Information Theory

Lecture Notes

Stefan M. Moser

6^{th} Edition — 2018

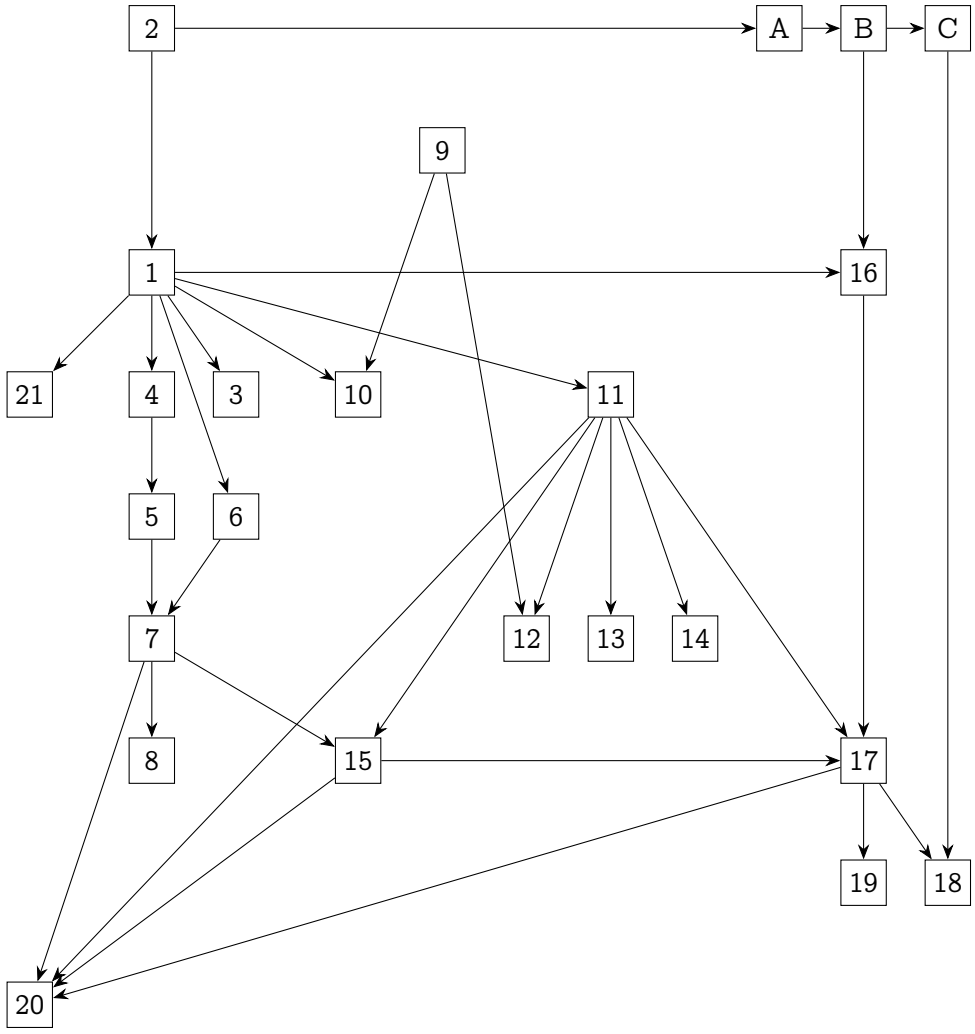
Signal and Information Processing Lab
ETH Zurich
Zurich, Switzerland

Institute of Communications Engineering
National Yang Ming Chiao Tung University (NYCU)
Hsinchu, Taiwan

6^{th} Edition — 2018.
Version 6.14.
Compiled on 14 September 2023.
For the latest version see https://moser-isi.ethz.ch/scripts.html

This book is typeset in the font family *Concrete* [Knu89] including both Concrete math and Euler math fonts.

Short Contents



Chapter dependency chart. An arrow has the meaning of “is required for”.

Contents

Preface

These lecture notes started out as handwritten guidelines that I used myself in class for teaching. As I got frequent and persistent requests from students attending the class to hand out these private notes in spite of their awful state (I still cannot really believe that any student was actually able to read them!), my students Lin Gu-Rong and Lin Hsuan-Yin took matters into their own hands and started to typeset my handwritten notes in LaTeX. These versions of notes then grew together with a couple of loose handouts (that complemented the textbook by Cover and Thomas [CT06] that I had been using as class textbook for several years) to a large pile of proper handouts and were used several times in combination with Cover and Thomas. During this time, the notes were constantly improved and extended. In this context I have to acknowledge the continued help of my students, in particular of Lin Hsuan-Yin and of Chang Hui-Ting, who typed the chapter about cryptography.

In fall 2008 my colleague Chen Po-Ning approached me and suggested to write a coding and information theory textbook for students with only little background in engineering or math. Together with three other colleagues we worked on this project for over two years until it got completed and published in 2012 [MC12]. This work had quite some impact on the presentation of some of the material of this class. In late 2010, I finally decided to compile all separate notes together, to add more detail in some places and rearrange the material, and to generate a proper lecture script that could be used as class textbook in future. Its first edition was used in class in fall 2011/2012 during which it underwent further revisions and improvements. In the third edition, I added one more chapter about the missing topic of error exponents, and the fourth edition additionally included a new chapter about polar codes. The fifth edition then saw strongly reorganized Chapter 6: it was split up into two chapters: one treating the case of compressing sources with memory (now Chapter 7) and one discussing the compression of infinite fixed sequences (now Chapter 8). Moreover, I also took out more advanced material from Chapter 1 and collected it in a new Chapter 3. Finally, in this sixth edition I realized that error exponents are a bit too advanced for an introductory course in information theory. So, when I started an overhaul of the advanced IT script [Mos22], I took the chance and moved the chapter on error exponents there.

In its current form, this script introduces the most important basics in information theory. Depending on the range and depth of the selected class material, it can be covered in about 20 to 30 lectures of two hours each. Roughly, the script can be divided into three main parts: Chapters 4–8 cover lossless data compression of discrete sources; Chapters 11–15 look at data transmission over discrete channels; and Chapters 16–19 deal with topics related to the Gaussian channel. Besides these main themes, the notes also briefly cover a couple of additional topics like convex optimization, gambling and horse betting, typicality, cryptography, and Gaussian random variables.

More in detail, the script starts with an introduction of the main quantities of information theory like entropy and mutual information in Chapter 1, followed by a very brief review of probability in Chapter 2. Chapter 3 introduces some more information measures and their connection to entropy. Apart from Section 3.1, this chapter can safely be omitted in a first reading.

In Chapters 4 and 5, lossless data compression of discrete memoryless sources is introduced including Huffman and Tunstall coding. Chapters 6, 7, and 8 extend these results to sources with memory and to universal data compression. Two different universal compression schemes are introduced: Elias–Willems coding and Lempel–Ziv coding. The tree-structured Lempel–Ziv coding is examined in Chapter 8 in a setting without a random source but a deterministic infinite source sequence.

As a preparation for later topics, in Chapter 9, we then discuss convex optimization and the Karush–Kuhn–Tucker (KKT) conditions. Chapter 10 is an interlude that introduces a quite different aspect of information theory: gambling and horse betting. It is quite separate from the rest of the script and only relies on some definitions from Chapter 1 and the KKT conditions from Chapter 9.

Next, Chapter 11 introduces the fundamental problem of data transmission and derives the channel coding theorem for discrete memoryless channels. In Chapter 12, we discuss the problem of computing capacity. Chapters 13 and 14 then describe two concrete examples of practical data transmission algorithms: Chapter 13 treats convolutional codes based on trellises and Chapter 14 gives a brief introduction to the capacity-achieving polar codes.

In Chapter 15, we combine source compression and channel transmission and discuss the basic problem of transmitting a given source over a given channel. In particular, we discuss the consequences of transmitting a source with an entropy rate being larger than capacity.

Before we extend the discussion of channel capacity to continuous alphabets (i.e., to the example of the Gaussian channel) in Chapter 17, we prepare the required background in Chapter 16. Chapters 18 and 19 then give short glimpses at continuous-time channels and at waterfilling, respectively.

Up to Chapter 19, I have avoided the usage of weak typicality. Chapter 20 corrects this omission. It introduces the asymptotic equipartition property

and typical sets, and then uses this new tool to re-derive several proofs of previous chapters (data compression, data transmission and joint source and channel coding). Note that for the basic understanding of the main concepts in this script, Chapter 20 is not necessary. Moreover, strong typicality is not covered here, but deferred to the second course *Advanced Topics in Information Theory* [Mos22] where it is treated in great detail.

Lastly, Chapter 21 presents a very brief overview of some of the most important basic results in cryptography.

I have made the experience that many students are mortally afraid of Gaussian random variables and Gaussian processes. I believe that this is mainly because they are never properly explained in undergraduate classes. Therefore I have decided to include a quite extensive coverage of them in the appendices, even though I usually do not teach them in class due to lack of time.

For a better understanding of the dependencies and relationships between the different chapters and of the prerequisites, on Page iv a dependency chart can be found.

I cannot and do not claim authorship for much of the material covered in here. My main contribution is the compilation, arrangement, and some more or less strong adaptation. There are many sources that I used as inspiration and from where I took the ideas of how to present information theory. Most important of all are my two teachers: Prof. James L. Massey during my master study and Prof. Amos Lapidoth during the time I was working on my Ph.D. Jim and Amos taught me most of what I know in information theory!

More in detail, I have used the following sources:

- The basic definitions in information theory (Chapters 1, 16, 20) are based on the textbook of Cover and Thomas [CT06]. Also from there come the treatment of horse betting (Chapter 10).

- The idea for the proof of the channel coding theorem (Chapters 11 and 17) using the beautifully simple threshold decoder was given to me by Tobias Koch, who got inspired by the work of Polyanskiy, Poor and Verdú [PPV10]. In principle, the threshold decoder goes back to a paper by Feinstein [Fei54].

- Most of the material about lossless data compression (Chapters 4–7) is very strongly inspired by the wonderful lecture notes of Jim Massey [Mas96]. In particular, I fully use the beautiful approach of trees to describe and analyze source codes. Also the chapter about computing capacity (Chapter 12), the introduction to convolutional codes (Chapter 13), and the overview over cryptography (Chapter 21) closely follow Jim's teaching style.

- The treatment of the tree-structured Lempel–Ziv universal data compression in Chapter 8 follows closely ideas I have learned from Emre Telatar.

- The material about the joint source and channel coding in Chapter 15 is inspired by the teaching of Chen Po-Ning [CA05a], [CA05b].

- For convex optimization I highly recommend the summary given by Bob Gallager in his famous textbook from 1968 [Gal68]. From this book also stems some inspiration on the presentation of the material about error exponents (in combination with Jim's lecture notes [Mas96]).

- Chapter 14 about polar codes is strongly inspired by lecture notes of Emre Telatar that I used in combination with the seminal paper by Erdal Arıkan [Arı09] and the paper [AT09] by Erdal and Emre.

- Finally, for a very exact, but still easy-to-understand treatment of stochastic processes (in particular Gaussian processes) I do not know any better book than *A Foundation in Digital Communication* by Amos Lapidoth [Lap17]. From this book stem Appendices A–C (with the exception of Section C.8 which is based on notes from Bob Gallager). Also the review of probability theory (Chapter 2) is strongly based on notes by Amos.

All of these sources are very inspiring and highly recommended for anyone who would like to learn more.

There are several important topics that are not covered in this script. Most notably, rate distortion theory is missing. While extremely beautiful and fundamental, rate distortion theory turns out to be harder to grasp than channel coding theory. Moreover, it is of less practical importance, and therefore I decided to defer it to the course *Advanced Topics in Information Theory* [Mos22]. In this subsequent course many more advanced topics are covered. In particular, it introduces strong typicality including Sanov's Theorem and the Conditional Limit Theorem, rate distortion theory, distributed source coding, and various multiple-user transmission schemes.

I would like to end this preface by saying thank you to the many readers who keep pointing out errors, typos, bad English, and bad explanations. I will keep working on these notes and try to improve them continually. So if you also find typos, errors, or if you have any comments, I would be very happy to hear them! Write to

moser@isi.ee.ethz.ch

Thank you!

To conclude I would like to express my deepest gratitude to Yin-Tai and Matthias, who were very patient with me whenever I sat writing on my computer with my thoughts far away...

Stefan M. Moser

Chapter 1

Shannon's Measure of Information

1.1 Motivation

We start by asking the question: What is information?

Let us consider some examples of sentences that contain some "information":

- The weather will be good tomorrow.
- The weather was bad last Sunday.
- The president of Taiwan will come to you tomorrow and will give you one million dollars.

The second statement seems not very interesting as you might already know what the weather has been like last Sunday. The last statement is much more exciting than the first two and therefore seems to contain much more information. But on the other hand do you actually believe it? Do you think it is likely that you will receive tomorrow one million dollars?

Let us make some easier examples:

- You ask: "Is the temperature in Taiwan currently above 30 degrees?"

  This question has only two possible answers: "yes" or "no".

- You ask: "The president of Taiwan has spoken with a certain person from Hsinchu today. With whom?"

  Here, the question has about 400,000 possible answers (since Hsinchu has about 400,000 inhabitants).

Obviously the second answer provides you with a much bigger amount of information than the first one. We learn the following:

The number of possible answers r should be linked to "information".

Let us have another example.

- You observe a gambler throwing a fair die. There are 6 possible outcomes $\{1,2,3,4,5,6\}$. You note the outcome and then tell it to a friend. By doing so you give your friend a certain amount of information.
- Next you observe the gambler throwing the die *three times*. Again, you note the three outcomes and tell them to your friend. Obviously, the amount of information that you give to your friend this time is three times as much as the first time.

So we learn:

"Information" should be additive in some sense.

Now we face a new problem: Regarding the example of the gambler above we see that in the first case we have $r = 6$ possible answers, while in the second case we have $r = 6^3 = 216$ possible answers. Hence in the second experiment there are 36 times more possible outcomes than in the first experiment. But we would like to have only a 3 times larger amount of information. So how do we resolve this?

A quite obvious idea is to use a logarithm. If we take the logarithm of the number of possible answers, then the exponent 3 will become a factor 3, exactly as we wish: $\log_b 6^3 = 3 \cdot \log_b 6$.

Precisely these observations were made by Ralph Hartley in 1928 in Bell Labs [Har28]. He gave the following definition.

Definition 1.1. We define the following measure of information:

$$\tilde{\mathrm{I}}(U) \triangleq \log_b r, \tag{1.1}$$

where r is the number of all possible outcomes of a random message U.

Using this definition we can confirm that it has the wanted property of additivity: If $U_1, \ldots, U_n$ are n random messages each taking value in an r-ary alphabet, then

$$\tilde{\mathrm{I}}(U_1, U_2, \ldots, U_n) = \log_b r^n = n \cdot \log_b r = n\, \tilde{\mathrm{I}}(U_1). \tag{1.2}$$

Hartley also correctly noted that the basis b of the logarithm is not really important for this measure. It only decides about the *unit of information*. So similarly to the fact that 1 km is the same distance as 1000 m, using

different b only causes a change of units without actually changing the amount of information described.

For two important and one unimportant special cases of b it has been agreed to use the following names for these units:

$b = 2$ ($\log_2$):	bit,
$b = e$ ($\ln$):	nat (because of "natural logarithm"),
$b = 10$ ($\log_{10}$):	Hartley.

Note that the unit Hartley has been chosen in honor of the first researcher who has made a first (partially correct) attempt of defining information. Unfortunately, as nobody in the world ever uses the basis $b = 10$, this honor is a bit questionable...

The measure $\tilde{\mathrm{I}}(U)$ is the right answer to many technical problems.

Example 1.2. A village has 8 telephones. How long must the phone number be? Or asked differently: How many bits of information do we need to send to the central office so that we are connected to a particular phone?

$$8 \text{ phones} \implies \log_2 8 = 3 \text{ bits}. \tag{1.3}$$

We choose the following phone numbers:

$$\{000, 001, 010, 011, 100, 101, 110, 111\}. \tag{1.4}$$

◊

In spite of its usefulness, Hartley's definition had no effect whatsoever in the world. That's life... On the other hand, it must be admitted that Hartley's definition has a fundamental flaw. To realize that something must be wrong, note that according to (1.1) the *smallest* nonzero amount of information is $\log_2 2 = 1$ bit. This might sound like only a small amount of information, but actually 1 bit can be *a lot of* information! As an example consider the 1-bit (yes or no) answer if a man asks a woman whether she wants to marry him... If you still don't believe that 1 bit is a huge amount of information, consider the following example.

Example 1.3. Currently there are about 7'621'000'000 persons living on our planet [Pop18]. How long must a binary telephone number U be if we want to be able to connect to every person?

According to Hartley we need

$$\tilde{\mathrm{I}}(U) = \log_2(7'621'000'000) \approx 32.8 \text{ bits}. \tag{1.5}$$

So with only 33 bits we can address every single person on this planet! Or, in other words, we only need 33 times 1 bit in order to distinguish every living human being. ◊

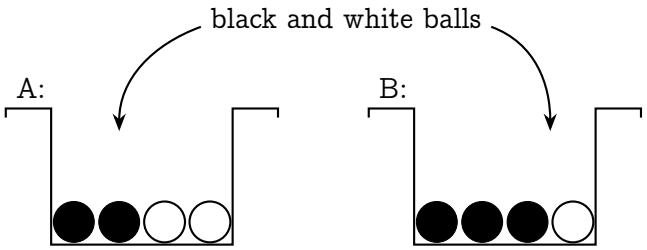


Figure 1.1: Two hats with four balls each.

We see that 1 bit is a lot of information and it cannot be that this is the smallest amount of (nonzero) information.

To understand more deeply what is wrong with Hartley's information measure, consider the two hats shown in Figure 1.1. Each hat contains four balls where the balls can be either white or black. Let us draw one ball at random and let U be the color of the ball. In hat A we have $r = 2$ colors: black and white, i.e., $\tilde{\mathrm{I}}(U_{\mathrm{A}}) = \log_2 2 = 1$ bit. In hat B we also have $r = 2$ colors and hence also $\tilde{\mathrm{I}}(U_{\mathrm{B}}) = 1$ bit. But obviously, we get less information if in hat B black shows up, since we somehow expect black to show up in the first place. Black is much more *likely*!

We realize the following:

> A proper measure of information needs to take into account the probabilities of the various possible events.

This has been observed for the first time by **Claude Elwood Shannon** in 1948 in his landmark paper: "A Mathematical Theory of Communication" [Sha48]. This paper has been like an explosion in the research community![1]

Before 1948, the engineering community was mainly interested in the behavior of a sinusoidal waveform that is passed through a communication system. Shannon, however, asked why we want to transmit a deterministic sinusoidal signal. The receiver already knows in advance that it will be a sinus, so it is much simpler to generate one at the receiver directly rather than to transmit it over a channel. In other words, Shannon had the fundamental insight that we need to consider *random* messages rather than deterministic messages whenever we deal with information.

[1] Besides the amazing accomplishment of inventing information theory, at the age of 21 Shannon also "invented" the computer in his *Master* thesis. He proved that electrical circuits can be used to perform logical and mathematical operations, which was the foundation of digital computing and digital circuit theory. It is probably the most important Master thesis of the 20th century! Incredible, isn't it?

Let us go back to the example of the hats in Figure 1.1 and have a closer look at hat B:

- There is one chance out of four possibilities that we draw a white ball.

  Since we would like to use Hartley's measure here, we recall that the quantity r inside the logarithm in (1.1) is "the number of all possible outcomes of a random message". Hence, from Hartley's point of view, we will see one realization out of r possible realizations. Translated to the case of the white ball, we see that we have one realization out of four possible realizations, i.e.,

  $$\log_2 4 = 2 \text{ bits} \tag{1.6}$$

  of information.

- On the other hand, there are three chances out of four that we draw a black ball.

  Here we cannot use Hartley's measure directly. But it is possible to translate the problem into a form that makes it somehow accessible to Hartley. We need to "normalize" the statement into a form that gives us *one* realization out of r. This can be done if we divide everything by 3, the number of black balls: We have 1 chance out of $\frac{4}{3}$ possibilities (whatever this means...) that we draw a black ball. Stated differently, we have one realization out of $\frac{4}{3}$ possible "realizations", i.e.,

  $$\log_2 \frac{4}{3} = 0.415 \text{ bits} \tag{1.7}$$

  of information.

So now we have two different values depending on what color we get. How shall we combine them to one value that represents the information? The most obvious choice is to average it, i.e., we weigh the different information values according to their probabilities of occurrence:

$$\frac{1}{4} \cdot 2 \text{ bits} + \frac{3}{4} \cdot 0.415 \text{ bits} = 0.811 \text{ bits} \tag{1.8}$$

or

$$\frac{1}{4} \cdot \log_2 4 + \frac{3}{4} \cdot \log_2 \frac{4}{3} = 0.811 \text{ bits}. \tag{1.9}$$

We see the following:

Shannon's measure of information is an "average Hartley information":

$$\sum_{i=1}^{r} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{r} p_i \log_2 p_i, \tag{1.10}$$

where p_i denotes the probability of the ith possible outcome.

We end this introductory section by pointing out that the given three motivating ideas, i.e.,

1. the number of possible answers r should be linked to "information";
2. "information" should be additive in some sense; and
3. a proper measure of information needs to take into account the probabilities of the various possible events,

are not sufficient to exclusively specify (1.10). In Appendix 1.A we will give some more information on why Shannon's measure should be defined like (1.10) and not differently. However, the true justification is — as always in engineering or physics — that Shannon's definition turns out to be useful.

1.2 Uncertainty or Entropy

1.2.1 Definition

We now formally define the Shannon measure of "self-information of a source". Due to its relationship with a corresponding concept in different areas of physics, Shannon called his measure *entropy*. We will stick to this name as it is standard in the whole literature. However note that *uncertainty* would be a far more precise description.

Definition 1.4. The *uncertainty* or *entropy* of a discrete random variable (RV) U that takes value in the set $\mathcal{U}$ (also called *alphabet* $\mathcal{U}$) is defined as

$$\mathrm{H}(U) \triangleq -\sum_{u \in \mathrm{supp}(P_U)} P_U(u) \log_b P_U(u), \tag{1.11}$$

where $P_U(\cdot)$ denotes the *probability mass function (PMF)*[2] of the RV U, and where the *support* of P_U is defined as

$$\mathrm{supp}(P_U) \triangleq \{u \in \mathcal{U} \colon P_U(u) > 0\}. \tag{1.12}$$

[2] Note that sometimes (but only if it is clear from the argument!) we will drop the subscript of the PMF: $P(u) = P_U(u)$.

Another, more mathematical, but often very convenient form to write the entropy is by means of expectation:

$$\mathrm{H}(U) = \mathrm{E}_U[-\log_b P_U(U)]. \tag{1.13}$$

Be careful about the two capital U: one denotes the name of the PMF, the other is the RV that is averaged over.

Remark 1.5. We have on purpose excluded the cases for which $P_U(u) = 0$ so that we do not get into trouble with $\log_b 0 = -\infty$. On the other hand, we also note that $P_U(u) = 0$ means that the symbol u never shows up. It therefore should not contribute to the uncertainty in any case. Luckily this is the case:

$$\lim_{t \downarrow 0} t \log_b t = 0, \tag{1.14}$$

i.e., we do not need to worry about this case. △

So we note the following:

> We will usually neglect to mention "support" when we sum over $P_U(u) \cdot \log_b P_U(u)$, i.e., we implicitly assume that we exclude all u with zero probability $P_U(u) = 0$.

As in the case of the Hartley measure of information, b denotes the *unit* of uncertainty:

$$\begin{aligned} b = 2: &\quad \text{bit},\\ b = e: &\quad \text{nat},\\ b = 10: &\quad \text{Hartley}. \end{aligned} \tag{1.15}$$

If the base of the logarithm is not specified, then we can choose it freely ourselves. However, note that the units are very important! A statement "$\mathrm{H}(U) = 0.43$" is completely meaningless. Since

$$\log_b \xi = \frac{\ln \xi}{\ln b}, \tag{1.16}$$

(with $\ln(\cdot)$ denoting the natural logarithm) 0.43 could mean anything as, e.g.,

$$\text{if } b = 2: \quad \mathrm{H}(U) = 0.43 \text{ bits}, \tag{1.17}$$

$$\text{if } b = e: \quad \mathrm{H}(U) = 0.43 \text{ nats} \approx 0.620 \text{ bits}, \tag{1.18}$$

$$\text{if } b = 256 = 2^8: \quad \mathrm{H}(U) = 0.43 \text{ "bytes"} = 3.44 \text{ bits}. \tag{1.19}$$

This is the same idea as 100 m not being the same distance as 100 km.

So remember:

If we do not specify the base of the logarithm, then the reader can choose the unit freely. However, never forget to add the units, once you write some concrete numbers!

Note that the term *bits* is used in two ways: Its first meaning is the *unit* of entropy when the base of the logarithm is chosen to be 2. Its second meaning is *binary digits*, i.e., in particular the number of digits of a binary codeword.

Remark 1.6. We like to remark here that while we often leave the choice of a unit (i.e., the base to the logarithm) up to the reader, we always assume that $b > 1$. If we chose a base smaller than one, weird things happen like negative monotonicity (having a decreasing value for H is equivalent with having an increasing uncertainty) or the logarithm being convex instead of concave. △

Remark 1.7. It is worth mentioning that if all r events are equally likely, Shannon's definition of entropy reduces to Hartley's measure:

$$p_i = \frac{1}{r},\ \forall i\colon \qquad \mathrm{H}(U) = -\sum_{i=1}^{r} \frac{1}{r} \log_b \frac{1}{r} = \frac{1}{r} \log_b r \underbrace{\sum_{i=1}^{r} 1}_{=r} = \log_b r. \tag{1.20}$$

△

Remark 1.8. Be careful not to confuse *uncertainty* with *information*! For motivation purposes, in Section 1.1 we have talked a lot about "information". However, what we actually meant there is "self-information" or more nicely put "uncertainty". You will learn soon that gaining information is equivalent to reducing uncertainty. △

Another important observation is that the entropy of U does not depend on the different possible values that U can take on, but only on the *probabilities* of these values. Hence,

$$U \in \{ \underbrace{1}_{\substack{\text{with}\\ \text{prob. } \frac{1}{2}}}, \underbrace{2}_{\substack{\text{with}\\ \text{prob. } \frac{1}{3}}}, \underbrace{3}_{\substack{\text{with}\\ \text{prob. } \frac{1}{6}}} \} \tag{1.21}$$

and

$$V \in \{ \underbrace{34}_{\substack{\text{with}\\ \text{prob. } \frac{1}{2}}}, \underbrace{512}_{\substack{\text{with}\\ \text{prob. } \frac{1}{3}}}, \underbrace{981}_{\substack{\text{with}\\ \text{prob. } \frac{1}{6}}} \} \tag{1.22}$$

have both the same entropy, which is

$$\mathrm{H}(U) = \mathrm{H}(V) = -\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{3}\log_2 \frac{1}{3} - \frac{1}{6}\log_2 \frac{1}{6} \approx 1.46 \text{ bits}. \tag{1.23}$$

This actually also holds true if we consider a random vector:

$$\mathbf{W} \in \left\{ \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{\substack{\text{with} \\ \text{prob. } \frac{1}{2}}}, \underbrace{\begin{pmatrix} 0 \\ 5 \end{pmatrix}}_{\substack{\text{with} \\ \text{prob. } \frac{1}{3}}}, \underbrace{\begin{pmatrix} 13 \\ 1 \end{pmatrix}}_{\substack{\text{with} \\ \text{prob. } \frac{1}{6}}} \right\}, \tag{1.24}$$

i.e., $\mathrm{H}(\mathbf{W}) = \mathrm{H}(U) = \mathrm{H}(V)$. Hence we can easily extend our definition to random vectors.

Definition 1.9. The *uncertainty* or *entropy* of a discrete random vector[3] $\mathbf{W} = (X, Y)^{\mathsf{T}}$ is defined as

$$\mathrm{H}(\mathbf{W}) = \mathrm{H}(X, Y) \triangleq \mathsf{E}_{X,Y}[-\log_b P_{X,Y}(X, Y)] \tag{1.25}$$

$$= - \sum_{(x,y) \in \mathrm{supp}(P_{X,Y})} P_{X,Y}(x, y) \log_b P_{X,Y}(x, y). \tag{1.26}$$

Here $P_{X,Y}(\cdot, \cdot)$ denotes the *joint probability mass function* of (X, Y) (see Section 2.2 for a review of discrete RVs).

1.2.2 Binary Entropy Function

One special case of entropy is so important that we introduce a specific name.

Definition 1.10. If U is *binary* with two possible values u_1 and u_2, $\mathcal{U} = \{u_1, u_2\}$, such that $\Pr[U = u_1] = p$ and $\Pr[U = u_2] = 1 - p$, then

$$\mathrm{H}(U) = \mathrm{H}_{\mathrm{b}}(p) \tag{1.27}$$

where $\mathrm{H}_{\mathrm{b}}(\cdot)$ is called the *binary entropy function* and is defined as

$$\mathrm{H}_{\mathrm{b}}(p) \triangleq -p \log_2 p - (1 - p) \log_2 (1 - p), \quad p \in [0, 1]. \tag{1.28}$$

The function $\mathrm{H}_{\mathrm{b}}(\cdot)$ is shown in Figure 1.2.

Exercise 1.11. *Show that the maximal value of* $\mathrm{H}_{\mathrm{b}}(p)$ *is 1 bit and is taken on for* $p = \frac{1}{2}$. ◊

1.2.3 The Information Theory Inequality

The following inequality does not really have a name, but since it is so important in information theory, we will follow Prof. James L. Massey, former professor at ETH in Zurich, and call it the *Information Theory Inequality* or the *IT Inequality*.

[3] Note that $(\cdot)^{\mathsf{T}}$ denotes the *transpose* of a vector.

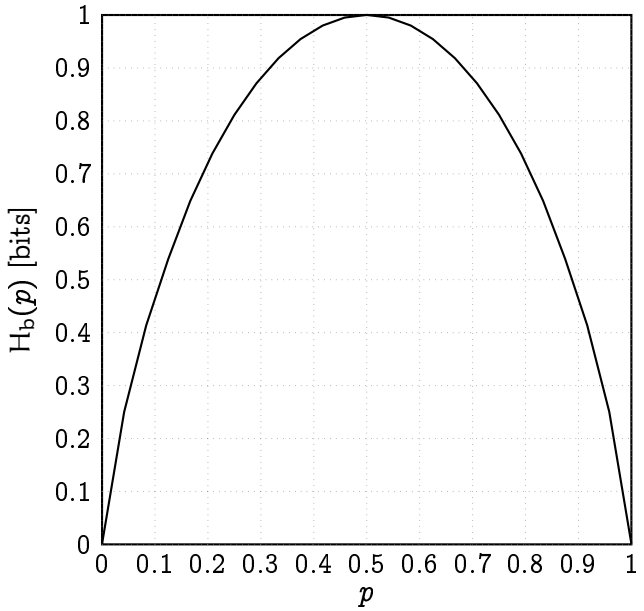


Figure 1.2: Binary entropy function $H_b(p)$ as a function of the probability p.

Proposition 1.12 (IT Inequality). For any base $b > 1$ and any $\xi > 0$,

$$\left(1 - \frac{1}{\xi}\right) \log_b e \le \log_b \xi \le (\xi - 1) \log_b e \tag{1.29}$$

with equalities on both sides if, and only if, $\xi = 1$.

Proof: Actually, Figure 1.3 can be regarded as a proof. For those readers who would like a formal proof, we provide next a mathematical derivation. We start with the upper bound. First note that

$$\log_b \xi \big|_{\xi=1} = 0 = (\xi - 1) \log_b e \big|_{\xi=1}. \tag{1.30}$$

Then have a look at the derivatives:

$$\frac{\mathrm{d}}{\mathrm{d}\xi}(\xi - 1) \log_b e = \log_b e \tag{1.31}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}\xi} \log_b \xi = \frac{1}{\xi} \log_b e \begin{cases} > \log_b e & \text{if } 0 < \xi < 1, \\ < \log_b e & \text{if } \xi > 1. \end{cases} \tag{1.32}$$

Hence, the two functions coincide at $\xi = 1$, and the linear function is above the logarithm for all other values.

To prove the lower bound again note that

$$\left(1 - \frac{1}{\xi}\right) \log_b e \bigg|_{\xi=1} = 0 = \log_b \xi \big|_{\xi=1} \tag{1.33}$$

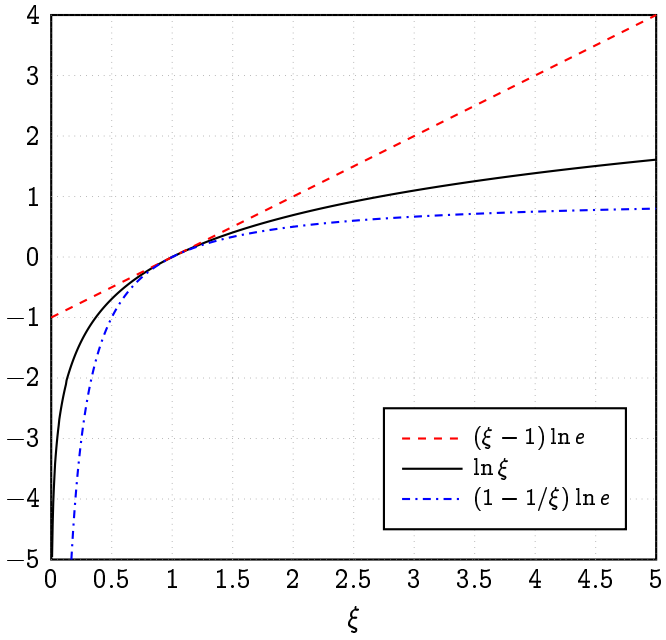


Figure 1.3: Illustration of the IT Inequality.

and

$$\frac{\mathrm{d}}{\mathrm{d}\xi}\left(1-\frac{1}{\xi}\right)\log_b e = \frac{1}{\xi^2}\log_b e \begin{cases} > \frac{\mathrm{d}}{\mathrm{d}\xi}\log_b \xi = \frac{1}{\xi}\log_b e & \text{if } 0<\xi<1, \\ < \frac{\mathrm{d}}{\mathrm{d}\xi}\log_b \xi = \frac{1}{\xi}\log_b e & \text{if } \xi>1, \end{cases} \tag{1.34}$$

similarly to above. □

1.2.4 Bounds on $\mathrm{H}(U)$

Theorem 1.13. If U has r possible values, then

$$0 \le \mathrm{H}(U) \le \log r, \tag{1.35}$$

where

$$\mathrm{H}(U) = 0 \qquad \text{if, and only if,} \qquad P_U(u) = 1 \quad \text{for some } u, \tag{1.36}$$

$$\mathrm{H}(U) = \log r \qquad \text{if, and only if,} \qquad P_U(u) = \frac{1}{r} \quad \forall\, u. \tag{1.37}$$

Proof: Since $0 \le P_U(u) \le 1$, we have

$$-P_U(u)\log_2 P_U(u) \begin{cases} = 0 & \text{if } P_U(u) = 1, \\ > 0 & \text{if } 0 < P_U(u) < 1. \end{cases} \tag{1.38}$$

Hence, $\mathrm{H}(U) \ge 0$. Equality can only be achieved if $-P_U(u)\log_2 P_U(u) = 0$ for all $u \in \mathrm{supp}(P_U)$, i.e., $P_U(u) = 1$ for all $u \in \mathrm{supp}(P_U)$.

To derive the upper bound we use a trick that is quite common in information theory: We take the difference and try to show that it must be nonpositive:

$$\begin{aligned}
\mathrm{H}(U) - \log r &= -\sum_{u\in\mathrm{supp}(P_U)} P_U(u)\log P_U(u) - \log r && (1.39)\\
&= -\sum_{u\in\mathrm{supp}(P_U)} P_U(u)\log P_U(u) - \sum_{u\in\mathrm{supp}(P_U)} P_U(u)\log r && (1.40)\\
&= -\sum_{u\in\mathrm{supp}(P_U)} P_U(u)\log\big(P_U(u)\cdot r\big) && (1.41)\\
&= \sum_{u\in\mathrm{supp}(P_U)} P_U(u)\log\Big(\underbrace{\frac{1}{r\cdot P_U(u)}}_{\triangleq\xi}\Big) && (1.42)\\
&\le \sum_{u\in\mathrm{supp}(P_U)} P_U(u)\Big(\frac{1}{r\cdot P_U(u)}-1\Big)\cdot\log e && (1.43)\\
&= \Bigg(\sum_{u\in\mathrm{supp}(P_U)}\frac{1}{r} - \underbrace{\sum_{u\in\mathrm{supp}(P_U)} P_U(u)}_{=1}\Bigg)\cdot\log e && (1.44)\\
&= \Bigg(\frac{1}{r}\sum_{u\in\mathrm{supp}(P_U)} 1 - 1\Bigg)\log e && (1.45)\\
&\le \Bigg(\frac{1}{r}\sum_{u\in\mathcal{U}} 1 - 1\Bigg)\log e && (1.46)\\
&= \Big(\frac{1}{r}\cdot r - 1\Big)\log e && (1.47)\\
&= (1-1)\log e = 0. && (1.48)
\end{aligned}$$

Here, (1.43) follows from the IT Inequality (Proposition 1.12); and in (1.46) we change the summation from $u\in\mathrm{supp}(P_U)$ to go over the whole alphabet $u\in\mathcal{U}$, i.e., we include additional (nonnegative) terms in the sum. Hence, $\mathrm{H}(U)\le\log r$.

Equality can only be achieved if

1. in (1.43), in the IT Inequality $\xi=1$, i.e., if
$$\frac{1}{r\cdot P_U(u)}=1 \implies P_U(u)=\frac{1}{r}, \tag{1.49}$$
for all u; and if
2. in (1.46), the support of U contains all elements of the alphabet $\mathcal{U}$, i.e.,[4]
$$|\mathrm{supp}(P_U)| = |\mathcal{U}| = r. \tag{1.50}$$

Note that if Condition 1 is satisfied, Condition 2 is also satisfied. □

[4] By $|\mathcal{U}|$ we denote the number of elements in the set $\mathcal{U}$.

1.2.5 Conditional Entropy

Similar to probability of random vectors, there is nothing really new about conditional probabilities given that a particular *event* $Y = y$ has occurred.

Definition 1.14. The *conditional entropy* or *conditional uncertainty* of the RV X given the *event* $Y = y$ is defined as

$$\mathrm{H}(X|Y=y) \triangleq -\sum_{x\in\mathrm{supp}(P_{X|Y}(\cdot|y))} P_{X|Y}(x|y)\log P_{X|Y}(x|y) \tag{1.51}$$

$$= \mathsf{E}\Big[-\log P_{X|Y}(X|Y)\,\Big|\,Y=y\Big]. \tag{1.52}$$

Note that the definition is identical to before apart from that everything is conditioned on the event $Y = y$.

From Theorem 1.13 we immediately get the following.

Corollary 1.15. If X has r possible values, then

$$0 \le \mathrm{H}(X|Y=y) \le \log r; \tag{1.53}$$

$$\mathrm{H}(X|Y=y) = 0 \quad \text{if, and only if,} \quad P(x|y) = 1 \quad \text{for some } x; \tag{1.54}$$

$$\mathrm{H}(X|Y=y) = \log r \quad \text{if, and only if,} \quad P(x|y) = \frac{1}{r} \quad \forall\, x. \tag{1.55}$$

Note that the conditional entropy given the event $Y = y$ is a function of y. Since Y is also a RV, we can now average over all possible events $Y = y$ according to the probabilities of each event. This will lead to the *averaged conditional entropy*.

Definition 1.16. The *conditional entropy* or *conditional uncertainty* of the RV X given the *random variable* Y is defined as

$$\mathrm{H}(X|Y) \triangleq \sum_{y\in\mathrm{supp}(P_Y)} P_Y(y)\cdot \mathrm{H}(X|Y=y) \tag{1.56}$$

$$= \mathsf{E}_Y[\mathrm{H}(X|Y=y)] \tag{1.57}$$

$$= -\sum_{(x,y)\in\mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y)\log P_{X|Y}(x|y) \tag{1.58}$$

$$= \mathsf{E}\Big[-\log P_{X|Y}(X|Y)\Big]. \tag{1.59}$$

The following observations should be straightforward:

- $P_Y(y)P_{X|Y}(x|y) = P_{X,Y}(x,y)$;
- $0 \le \mathrm{H}(X|Y) \le \log r$, where r is the number of values that the RV X can take on;
- $\mathrm{H}(X|Y) = 0$ if, and only if, $P_{X|Y}(x|y) = 1$ for some x and $\forall\, y$;
- $\mathrm{H}(X|Y) = \log r$ if, and only if, $P_{X|Y}(x|y) = \frac{1}{r}$, $\forall\, x$ and $\forall\, y$;

- $\mathrm{H}(X|Y) \neq \mathrm{H}(Y|X)$ (however, we will see later that $\mathrm{H}(X) - \mathrm{H}(X|Y) = \mathrm{H}(Y) - \mathrm{H}(Y|X)$).

Next we get the following very important theorem.

Theorem 1.17 (Conditioning Reduces Entropy).
For any two discrete RVs X and Y,

$$\mathrm{H}(X|Y) \le \mathrm{H}(X) \tag{1.60}$$

with equality if, and only if, X and Y are statistically independent $X \perp\!\!\!\perp Y$.

Proof: Again we use the same trick as above and prove the inequality by showing that the difference is nonpositive. We start by noting that

$$\begin{aligned}
\mathrm{H}(X) &= -\sum_{x\in\mathrm{supp}(P_X)} P_X(x)\log P_X(x)\cdot 1 && (1.61)\\
&= -\sum_{x\in\mathrm{supp}(P_X)} P_X(x)\log P_X(x) \underbrace{\sum_{y\in\mathrm{supp}(P_{Y|X}(\cdot|x))} P_{Y|X}(y|x)}_{=1} && (1.62)\\
&= -\sum_{x\in\mathrm{supp}(P_X)}\sum_{y\in\mathrm{supp}(P_{Y|X}(\cdot|x))} P_X(x)P_{Y|X}(y|x)\log P_X(x) && (1.63)\\
&= -\sum_{(x,y)\in\mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y)\log P_X(x) && (1.64)
\end{aligned}$$

such that

$$\begin{aligned}
\mathrm{H}(X|Y) - \mathrm{H}(X) &= -\sum_{(x,y)\in\mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y)\log P_{X|Y}(x|y)\\
&\quad + \sum_{(x,y)\in\mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y)\log P_X(x) && (1.65)\\
&= \sum_{(x,y)\in\mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y)\log\frac{P_X(x)}{P_{X|Y}(x|y)} && (1.66)\\
&= \mathsf{E}\left[\log\frac{P_X(X)}{P_{X|Y}(X|Y)}\right]. && (1.67)
\end{aligned}$$

Note that it is always possible to enlarge the summation of an expectation to include more random variables, i.e.,

$$\mathsf{E}_X[f(X)] = \mathsf{E}_{X,Y}[f(X)] = \mathsf{E}[f(X)]. \tag{1.68}$$

Hence, the expression (1.67) could be much easier derived using the expectation notation:

$$\mathrm{H}(X|Y) - \mathrm{H}(X) = \mathsf{E}\Big[-\log P_{X|Y}(X|Y)\Big] - \mathsf{E}[-\log P_X(X)] \tag{1.69}$$

$$= \mathsf{E}\left[\log \frac{P_X(X)}{P_{X|Y}(X|Y)}\right]. \tag{1.70}$$

So, we have the following derivation:

$$\begin{aligned}
&\mathrm{H}(X|Y) - \mathrm{H}(X)\\
&= \mathsf{E}\left[\log \frac{P_X(X)}{P_{X|Y}(X|Y)}\right] && (1.71)\\
&= \mathsf{E}\left[\log \frac{P_X(X)\cdot P_Y(Y)}{P_{X|Y}(X|Y)\cdot P_Y(Y)}\right] && (1.72)\\
&= \mathsf{E}\left[\log \frac{P_X(X)P_Y(Y)}{P_{X,Y}(X,Y)}\right] && (1.73)\\
&= \sum_{(x,y)\in \mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y) \log \frac{P_X(x)P_Y(y)}{P_{X,Y}(x,y)} && (1.74)\\
&\le \sum_{(x,y)\in \mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y)\left(\frac{P_X(x)P_Y(y)}{P_{X,Y}(x,y)} - 1\right)\cdot \log e && (1.75)\\
&= \sum_{(x,y)\in \mathrm{supp}(P_{X,Y})} \big(P_X(x)P_Y(y) - P_{X,Y}(x,y)\big)\cdot \log e && (1.76)\\
&= \left(\sum_{(x,y)\in \mathrm{supp}(P_{X,Y})} P_X(x)P_Y(y) - 1\right)\cdot \log e && (1.77)\\
&\le \left(\sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_X(x)P_Y(y) - 1\right)\cdot \log e && (1.78)\\
&= \left(\sum_{x\in\mathcal{X}} P_X(x) \sum_{y\in\mathcal{Y}} P_Y(y) - 1\right)\cdot \log e && (1.79)\\
&= (1-1)\log e = 0. && (1.80)
\end{aligned}$$

Here, (1.75) again follows from the IT Inequality (Proposition 1.12) and (1.78) holds because we add additional terms to the sum. Hence, $\mathrm{H}(X|Y) \le \mathrm{H}(X)$.

Equality can be achieved if, and only if,

1. in (1.75), in the IT Inequality $\xi = 1$, i.e., if

$$\frac{P(x)P(y)}{P(x,y)} = 1 \implies P(x)P(y) = P(x,y) \tag{1.81}$$

   for all x, y (which means that $X \perp\!\!\!\perp Y$);

2. in (1.78), $P(x)\cdot P(y) = 0$ for $P(x,y) = 0$.

Note that if Condition 1 is satisfied, Condition 2 is also satisfied. □

Remark 1.18. Attention: The *conditioning reduces entropy*-rule only applies to *random variables*, not to events! In particular,

$$\mathrm{H}(X|Y=y) \lessgtr \mathrm{H}(X). \tag{1.82}$$

To understand why this is the case, consider the following example. △

Example 1.19. Let us consider three different species of foxes: The *red fox* is the most common fox and can be found all over Europe, Asia and North America; the *white*[5] *fox* is common in the northern and Arctic areas of Europe, Asia and America; and the *gray fox* is exclusively found in North and Central America. It seems difficult to find exact numbers, but for the sake of argument, let us assume that the worldwide fox population consists of about 70% red, 20% white, and 10% gray foxes.

So, let X be the type of fox I randomly observe. The uncertainty about X is thus

$$\mathrm{H}(X) = -0.7\log 0.7 - 0.2\log 0.2 - 0.1\log 0.1 \approx 1.16 \text{ bits}. \tag{1.83}$$

Now let Y be the country in which I have observed the fox. If I tell you Y, your uncertainty about X changes. Examples:

- $Y =$ Switzerland: Since in Switzerland there only live red foxes, the uncertainty of X given this event equals zero:

  $$\mathrm{H}(X|Y=\text{Switzerland}) = -1\log 1 = 0 \text{ bits}, \tag{1.84}$$

  which is obviously less than $\mathrm{H}(X)$. So, the side-information "Switzerland" completely removes any uncertainty about X.

- $Y =$ Canada: Canada is home to all three types of foxes. Again, it is difficult to find exact numbers, but let us assume we have 40% red foxes, 40% white foxes, and 20% gray foxes in Canada. Thus,

  $$\mathrm{H}(X|Y=\text{Canada}) = -0.4\log 0.4 - 0.4\log 0.4 - 0.2\log 0.2 \approx 1.52 \text{ bits}, \tag{1.85}$$

  which is considerably *larger* than $\mathrm{H}(X) \approx 1.16$ bits. So, the side-information "Canada" made it actually harder to guess which fox was observed. The worst case would be a country with an equal distribution of the three types of foxes, in which case the conditional entropy would be $\log 3 \approx 1.58$ bits.

[5] The name is slightly misleading as the white fox has white fur only in winter. In summer it changes color and turns a dark brown.

So we see that depending on the value of Y, the uncertainty about X might be reduced or increased. However, we do encounter more cases similar to Switzerland than to Canada (e.g., $Y =$ Iceland will also reduce the uncertainty to zero because in Iceland only the white fox can be found), and thus it is more likely to have an uncertainty reduction than an uncertainty increase: From Theorem 1.17 we know that *on average* the knowledge of Y will reduce our uncertainty about X: $\mathrm{H}(X|Y) \le \mathrm{H}(X)$. ◊

1.2.6 Extensions to More RVs

We can now easily extend entropy and conditional entropy to more RVs. We only show some examples involving three RVs. You should have no troubles to extend it to an arbitrary number of RVs and events.

Definition 1.20. The conditional entropy of a RV X conditional on the RV Y and the event $Z = z$ is defined as

$$\begin{aligned}
&\mathrm{H}(X|Y, Z=z)\\
&\quad\triangleq \mathrm{E}\Big[-\log P_{X|Y,Z}(X|Y,z)\,\Big|\,Z=z\Big] && (1.86)\\
&\quad= -\sum_{(x,y)\in\operatorname{supp}(P_{X,Y|Z}(\cdot,\cdot|z))} P_{X,Y|Z}(x,y|z)\log P_{X|Y,Z}(x|y,z) && (1.87)\\
&\quad= \mathrm{E}_Y[\mathrm{H}(X|Y=y,Z=z)\,|\,Z=z] && (1.88)\\
&\quad= \sum_{y\in\operatorname{supp}(P_{Y|Z}(\cdot|z))} P_{Y|Z}(y|z)\,\mathrm{H}(X|Y=y,Z=z) && (1.89)\\
&\quad= \sum_{y\in\operatorname{supp}(P_{Y|Z}(\cdot|z))} P_{Y|Z}(y|z)\Bigg(-\sum_{x\in\operatorname{supp}(P_{X|Y,Z}(\cdot|y,z))} P_{X|Y,Z}(x|y,z)\cdot\\
&\qquad\qquad\cdot\log P_{X|Y,Z}(x|y,z)\Bigg) && (1.90)\\
&\quad= -\sum_{\substack{y\in\operatorname{supp}(P_{Y|Z}(\cdot|z))\\ x\in\operatorname{supp}(P_{X|Y,Z}(\cdot|y,z))}} \underbrace{P_{Y|Z}(y|z)P_{X|Y,Z}(x|y,z)}_{=P_{X,Y|Z}(x,y|z)}\log P_{X|Y,Z}(x|y,z) && (1.91)\\
&\quad= -\sum_{(x,y)\in\operatorname{supp}(P_{X,Y|Z}(\cdot,\cdot|z))} P_{X,Y|Z}(x,y|z)\log P_{X|Y,Z}(x|y,z). && (1.92)
\end{aligned}$$

The conditional entropy of X conditional on the RVs Y and Z is defined as

$$\begin{aligned}
\mathrm{H}(X|Y,Z) &\triangleq \mathrm{E}_Z[\mathrm{H}(X|Y,Z=z)] && (1.93)\\
&= \mathrm{E}\Big[-\log P_{X|Y,Z}(X|Y,Z)\Big] && (1.94)\\
&= -\sum_{(x,y,z)\in\operatorname{supp}(P_{X,Y,Z})} P_{X,Y,Z}(x,y,z)\log P_{X|Y,Z}(x|y,z). && (1.95)
\end{aligned}$$

The properties generalize analogously:

- $\mathrm{H}(X|Y,Z) \le \mathrm{H}(X|Z)$;

- $\mathrm{H}(X|Y,Z=z) \le \mathrm{H}(X|Z=z)$;
- but not (necessarily) $\mathrm{H}(X|Y,Z=z) \not\le \mathrm{H}(X|Y)$.

Note that the easiest way of remembering the definition of a complex entropy expression is to use the notation with the expected value. For example, $\mathrm{H}(X,Y,Z|U,V,W=w)$ is given as

$$\mathrm{H}(X,Y,Z|U,V,W=w) \triangleq \mathrm{E}\Big[-\log P_{X,Y,Z|U,V,W}(X,Y,Z|U,V,w)\Big|W=w\Big] \tag{1.96}$$

where the expectation is over the joint PMF of (X,Y,Z,U,V) conditional on the event $W=w$.

1.2.7 Chain Rule

Theorem 1.21 (Chain Rule).
Let $X_1,\dots,X_n$ be n discrete RVs with a joint PMF $P_{X_1,\dots,X_n}$. Then

$$\begin{aligned}&\mathrm{H}(X_1,X_2,\dots,X_n)\\ &\quad=\mathrm{H}(X_1)+\mathrm{H}(X_2|X_1)+\cdots+\mathrm{H}(X_n|X_1,X_2,\dots,X_{n-1}) &(1.97)\\ &\quad=\sum_{k=1}^{n}\mathrm{H}(X_k|X_1,X_2,\dots,X_{k-1}). &(1.98)\end{aligned}$$

Proof: This follows directly from the chain rule for PMFs:

$$P_{X_1,\dots,X_n}=P_{X_1}\cdot P_{X_2|X_1}\cdot P_{X_3|X_1,X_2}\cdots P_{X_n|X_1,\dots,X_{n-1}}. \tag{1.99}$$

□

Example 1.22. Let (X_1,X_2) take on the values $(0,0)$, $(1,1)$, $(1,0)$ equally likely with probability $\frac{1}{3}$. Then

$$\mathrm{H}(X_1,X_2)=\log 3\approx 1.58 \text{ bits}. \tag{1.100}$$

We also immediately see that $P_{X_1}(0)=\frac{1}{3}$ and $P_{X_1}(1)=\frac{2}{3}$. Hence,

$$\mathrm{H}(X_1)=-\frac{1}{3}\log\frac{1}{3}-\frac{2}{3}\log\frac{2}{3}=\mathrm{H_b}\Big(\frac{1}{3}\Big)\approx 0.91 \text{ bits}. \tag{1.101}$$

Moreover, $P_{X_2|X_1}(0|0)=1$, $P_{X_2|X_1}(1|0)=0$, such that

$$\mathrm{H}(X_2|X_1=0)=0 \text{ bits}, \tag{1.102}$$

and $P_{X_2|X_1}(0|1)=\frac{1}{2}$ and $P_{X_2|X_1}(1|1)=\frac{1}{2}$ such that

$$\mathrm{H}(X_2|X_1=1)=\log 2=1 \text{ bit}. \tag{1.103}$$

Using the definition of conditional entropy we then compute

$$\begin{aligned} \mathrm{H}(X_2|X_1) &= P_{X_1}(0)\,\mathrm{H}(X_2|X_1=0) + P_{X_1}(1)\,\mathrm{H}(X_2|X_1=1) && (1.104)\\ &= \frac{1}{3}\cdot 0 + \frac{2}{3}\cdot 1 \text{ bits} && (1.105)\\ &= \frac{2}{3} \text{ bits}. && (1.106) \end{aligned}$$

We finally use the chain rule to confirm the result we have computed above already directly:

$$\begin{aligned} \mathrm{H}(X_1,X_2) &= \mathrm{H}(X_1) + \mathrm{H}(X_2|X_1) && (1.107)\\ &= \mathrm{H_b}\left(\frac{1}{3}\right) + \frac{2}{3} \text{ bits} && (1.108)\\ &= -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} + \frac{2}{3} \text{ bits} && (1.109)\\ &= \frac{1}{3}\log_2 3 - \frac{2}{3}\log_2 2 + \frac{2}{3}\log_2 3 + \frac{2}{3}\log_2 2 && (1.110)\\ &= \log 3. && (1.111) \end{aligned}$$

$\Diamond$

1.3 Mutual Information

1.3.1 Definition

Finally, we come to the definition of information. The following definition is very intuitive: Suppose you have a RV X with a certain uncertainty $\mathrm{H}(X)$. The amount that another related RV Y can tell you about X is the *information* that Y gives you about X. How to measure it? Well, compare the uncertainty of X before and after you know Y. The difference is what you have learned!

Definition 1.23. The *mutual information* between the discrete RVs X and Y is given by

$$\mathrm{I}(X;Y) \triangleq \mathrm{H}(X) - \mathrm{H}(X|Y). \tag{1.112}$$

Note that $\mathrm{H}(X|Y)$ is the uncertainty about X when knowing Y.

Remark 1.24. Note that it is a *mutual* information, not an "information about X provided by Y". The reason for this name can be quickly understood if we consider the following. Using twice the chain rule for entropies (Theorem 1.21) we have:

$$\begin{aligned} \mathrm{H}(X,Y) &= \mathrm{H}(X) + \mathrm{H}(Y|X) = \mathrm{H}(Y) + \mathrm{H}(X|Y) && (1.113)\\ \Longrightarrow \quad \mathrm{H}(X) - \mathrm{H}(X|Y) &= \mathrm{H}(Y) - \mathrm{H}(Y|X) && (1.114)\\ \Longrightarrow \quad \mathrm{I}(X;Y) &= \mathrm{I}(Y;X) && (1.115) \end{aligned}$$

Hence, X will tell exactly the same about Y as Y tells about X. For example, assume X being the weather in Hsinchu and Y being the weather in Taichung. Knowing X will reduce your uncertainty about Y in the same way as knowing Y will reduce your uncertainty about X. △

The mutual information can be expressed in many equivalent forms. A particularly nice one can be derived as follows:[6]

$$\begin{aligned}
\mathrm{I}(X;Y) &= \mathrm{H}(X) - \mathrm{H}(X|Y) && (1.116)\\
&= \mathrm{E}[-\log P_X(X)] - \mathrm{E}\Big[-\log P_{X|Y}(X|Y)\Big] && (1.117)\\
&= \mathrm{E}\left[\log \frac{P_{X|Y}(X|Y)}{P_X(X)}\right] && (1.118)\\
&= \mathrm{E}\left[\log \frac{P_{X|Y}(X|Y)\cdot P_Y(Y)}{P_X(X)\cdot P_Y(Y)}\right] && (1.119)\\
&= \mathrm{E}\left[\log \frac{P_{X,Y}(X,Y)}{P_X(X)P_Y(Y)}\right] && (1.120)\\
&= \sum_{(x,y)\in \mathrm{supp}(P_{X,Y})} P_{X,Y}(x,y)\log \frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)}. && (1.121)
\end{aligned}$$

From the chain rule it follows that

$$\mathrm{H}(X|Y) = \mathrm{H}(X,Y) - \mathrm{H}(Y), \tag{1.122}$$

and thus we obtain from (1.116) one more form:

$$\mathrm{I}(X;Y) = \mathrm{H}(X) + \mathrm{H}(Y) - \mathrm{H}(X,Y). \tag{1.123}$$

This form can also be used for a Venn-diagram, as shown in Figure 1.4, and is particularly nice because it shows the mutual information's symmetry.

1.3.2 Properties

Since mutual information is closely related to entropy, we can easily derive some of its properties. The most important property is that mutual information cannot be negative.

Theorem 1.25. Let X and Y be two discrete RVs with mutual information $\mathrm{I}(X;Y)$. Then

$$0 \le \mathrm{I}(X;Y) \le \min\{\mathrm{H}(X), \mathrm{H}(Y)\}. \tag{1.124}$$

On the left-hand side, we achieve equality if, and only if, $P_{X,Y} = P_X \cdot P_Y$, i.e., if and only if $X \perp\!\!\!\perp Y$. On the right-hand side, we have equality if, and only if, either X determines Y or vice versa.

[6] Recall the way we can handle expectations shown in (1.68).

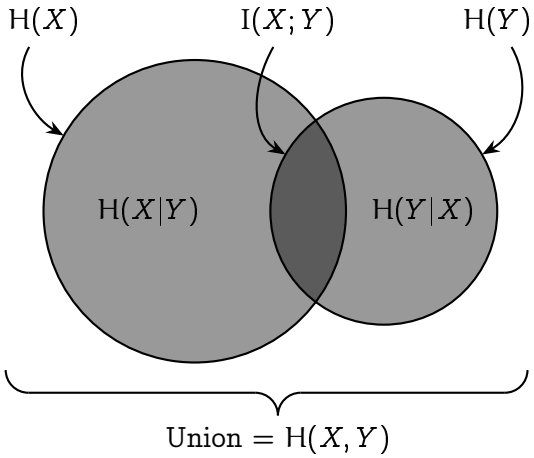


Figure 1.4: Diagram depicting mutual information and entropy in a set-theory way of thinking.

Proof: Because conditioning reduces entropy (Theorem 1.17), we have

$$\mathrm{I}(X;Y) = \mathrm{H}(Y) - \underbrace{\mathrm{H}(Y|X)}_{\le \mathrm{H}(Y)} \ge \mathrm{H}(Y) - \mathrm{H}(Y) = 0, \tag{1.125}$$

proving the lower bound. Note that we achieve equality if, and only if, X is independent of Y such that $\mathrm{H}(Y|X) = \mathrm{H}(Y)$.

To prove the upper bound, we recall the nonnegativity of entropy (Theorem 1.13) to obtain

$$\mathrm{I}(X;Y) = \mathrm{H}(X) - \underbrace{\mathrm{H}(X|Y)}_{\ge 0} \le \mathrm{H}(X), \tag{1.126}$$

$$\mathrm{I}(X;Y) = \mathrm{H}(Y) - \underbrace{\mathrm{H}(Y|X)}_{\ge 0} \le \mathrm{H}(Y). \tag{1.127}$$

We achieve equality if, and only if, $\mathrm{H}(X|Y) = 0$ or $\mathrm{H}(Y|X) = 0$, which according to Corollary 1.15 can only occur if $P(x|y) = 1$ for some x and for all y or if $P(y|x) = 1$ for some y and for all x, i.e., if Y determines X or vice versa. □

Note that the mutual information of a RV X about itself is simply its entropy:

$$\mathrm{I}(X;X) = \mathrm{H}(X) - \underbrace{\mathrm{H}(X|X)}_{=0} = \mathrm{H}(X). \tag{1.128}$$

Therefore $\mathrm{H}(X)$ sometimes is also referred to as *self-information*.

1.3.3 Conditional Mutual Information

Similarly to the entropy we can extend the mutual information to conditional versions. Since these definitions are basically only repetitions of the corresponding definitions of conditional entropies, we only state a few:

$$\mathrm{I}(X;Y|Z=z) \triangleq \mathrm{H}(X|Z=z) - \mathrm{H}(X|Y,Z=z); \tag{1.129}$$

$$\mathrm{I}(X;Y|Z) \triangleq \mathsf{E}_Z[\mathrm{I}(X;Y|Z=z)] \tag{1.130}$$

$$= \sum_z P_Z(z)(\mathrm{H}(X|Z=z) - \mathrm{H}(X|Y,Z=z)) \tag{1.131}$$

$$= \mathrm{H}(X|Z) - \mathrm{H}(X|Y,Z). \tag{1.132}$$

1.3.4 Chain Rule

Finally, also the chain rule of entropy directly extends to mutual information.

Theorem 1.26 (Chain Rule).

$$\begin{aligned}&\mathrm{I}(X;Y_1,Y_2,\dots,Y_n)\\ &\quad= \mathrm{I}(X;Y_1) + \mathrm{I}(X;Y_2|Y_1) + \cdots + \mathrm{I}(X;Y_n|Y_1,Y_2,\dots,Y_{n-1}) &(1.133)\\ &\quad= \sum_{k=1}^{n} \mathrm{I}(X;Y_k|Y_1,Y_2,\dots,Y_{k-1}). &(1.134)\end{aligned}$$

Proof: From the chain rule of entropy we have

$$\begin{aligned}&\mathrm{I}(X;Y_1,\dots,Y_n)\\ &\quad= \mathrm{H}(Y_1,\dots,Y_n) - \mathrm{H}(Y_1,\dots,Y_n|X) &(1.135)\\ &\quad= \mathrm{H}(Y_1) + \mathrm{H}(Y_2|Y_1) + \cdots + \mathrm{H}(Y_n|Y_{n-1},\dots,Y_1)\\ &\qquad - \big(\mathrm{H}(Y_1|X) + \mathrm{H}(Y_2|Y_1,X) + \cdots + \mathrm{H}(Y_n|Y_{n-1},\dots,Y_1,X)\big) &(1.136)\\ &\quad= \big(\mathrm{H}(Y_1) - \mathrm{H}(Y_1|X)\big) + \big(\mathrm{H}(Y_2|Y_1) - \mathrm{H}(Y_2|Y_1,X)\big) + \cdots\\ &\qquad + \big(\mathrm{H}(Y_n|Y_{n-1},\dots,Y_1) - \mathrm{H}(Y_n|Y_{n-1},\dots,Y_1,X)\big) &(1.137)\\ &\quad= \mathrm{I}(X;Y_1) + \mathrm{I}(X;Y_2|Y_1) + \cdots + \mathrm{I}(X;Y_n|Y_{n-1},\dots,Y_1) &(1.138)\\ &\quad= \sum_{k=1}^{n} \mathrm{I}(X;Y_k|Y_{k-1},\dots,Y_1). &(1.139)\end{aligned}$$

□

1.4 Comments on our Notation

1.4.1 General

We try to clearly distinguish between constant and random quantities. The basic rule here is

$$\begin{aligned} &\text{capital letter } X\text{: random variable,}\\ &\text{small letter } \quad x\text{: deterministic value.}\end{aligned}$$

For vectors or sequences bold face is used:

$$\begin{aligned} &\text{capital bold letter } \mathbf{X}\text{: random vector,}\\ &\text{small bold letter } \quad \mathbf{x}\text{: deterministic vector.}\end{aligned}$$

There are a few exceptions to this rule. Certain deterministic quantities are very standard in capital letters, so, to distinguish them from random variables, we use a different font. For example, the capacity is denoted by C (in contrast to a random variable C) or the codewords in Chapter 4 are D-ary (and not D-ary).

Moreover, matrices are also commonly depicted in capitals, but for them we use yet another font, e.g., C. Then, sets are denoted using a calligraphic font: $\mathcal{C}$. An example of a set is the alphabet $\mathcal{X}$ of a random variable X.

Finally, also the PMF is denoted by a capital letter P: The discrete random variable X has PMF $P_X(\cdot)$, where we normally will use the subscript X to indicate to which random variable the PMF belongs to. Sometimes, however, we also use $P(\cdot)$ or $Q(\cdot)$ without subscript to denote a generic PMF (see, e.g., Section 3.3 or Chapter 12). To avoid confusion, we shall never use RVs P or Q.

1.4.2 Entropy and Mutual Information

As introduced in Sections 1.2 and 1.3, respectively, entropy and mutual information are always shown in connection with RVs: $\mathrm{H}(X)$ and $\mathrm{I}(X;Y)$. But strictly speaking, they are functions of PMFs and not RVs: $\mathrm{H}(X)$ is a function of $P_X(\cdot)$ (see, e.g., (1.21)–(1.23)) and $\mathrm{I}(X;Y)$ is a function of $P_{X,Y}(\cdot,\cdot)$ (see, e.g., (1.121)). So in certain situations, it is more convenient to write H and I as functions of PMFs:

$$\mathrm{H}(Q(\cdot)) \text{ or } \mathrm{H}(Q) \quad \text{and} \quad \mathrm{I}(P_X, P_{Y|X}). \tag{1.140}$$

To emphasize the difference in notation, in this case we drop the use of the semicolon for the mutual information. However, for the entropy no such distinction is made. We hope that no confusion will arise as we never define RVs P or Q in the first place.

Finally, sometimes we write a PMF $P(\cdot)$ in a vector-like notation listing all possible probability values:

$$P_X(\cdot) = (p_1, p_2, \dots, p_r) \tag{1.141}$$

denotes the PMF of an r-ary RV X with probabilities $p_1, \dots, p_r$. The entropy of X can then be written in the following three equivalent forms

$$\mathrm{H}(X) = \mathrm{H}(P_X) = \mathrm{H}(p_1, \dots, p_r) \tag{1.142}$$

and is, of course, equal to

$$\mathrm{H}(p_1, \dots, p_r) = \sum_{i=1}^{r} p_i \log \frac{1}{p_i}. \tag{1.143}$$

1.A Appendix: Uniqueness of the Definition of Entropy

In Section 1.1 we have tried to motivate the definition of the entropy. Even though we succeeded partially, we were not able to give full justification of Definition 1.4. While Shannon did provide a mathematical justification [Sha48, Section 6], he did not consider it very important. We omit Shannon's argument, but instead we will now quickly summarize a slightly different result that was presented in 1956 by Aleksandr Khinchin. Khinchin specified four properties that entropy is supposed to have and then proved that, given these four properties, (1.11) is the only possible definition. We will now quickly summarize his result.

We define $\mathrm{H}_r(p_1, \dots, p_r)$ to be a function of r probabilities $p_1, \dots, p_r$ that sum up to 1:

$$\sum_{i=1}^{r} p_i = 1. \tag{1.144}$$

We further ask this function to satisfy the following four properties:

1. For any r, $\mathrm{H}_r(p_1, \dots, p_r)$ is continuous (i.e., a slight change to the values of p_i will only cause a slight change to H_r) and symmetric in $p_1, \dots, p_r$ (i.e., changing the order of the probabilities does not affect the value of H_r).

2. Any event of probability 0 does not contribute to H_r:

$$\mathrm{H}_{r+1}(p_1, \dots, p_r, 0) = \mathrm{H}_r(p_1, \dots, p_r). \tag{1.145}$$

3. H_r is maximized by the uniform distribution:

$$\mathrm{H}_r(p_1, \dots, p_r) \le \mathrm{H}_r\left(\frac{1}{r}, \dots, \frac{1}{r}\right). \tag{1.146}$$

4. If we partition the $m \cdot r$ possible outcomes of a random experiment into m groups, each group containing r elements, then we can do the experiment in two steps:

   (a) determine the group to which the actual outcome belongs,

   (b) find the outcome in this group.

   Let $p_{j,i}$, $1 \le j \le m$, $1 \le i \le r$, be the probabilities of the outcomes in this random experiment. Then the total probability of all outcomes in group j is

   $$q_j = \sum_{i=1}^{r} p_{j,i}, \tag{1.147}$$

   and the conditional probability of outcome i from group j is then given by

   $$\frac{p_{j,i}}{q_j}. \tag{1.148}$$

   Now $\mathrm{H}_{m\cdot r}$ can be written as follows:

   $$\begin{aligned}&\mathrm{H}_{m\cdot r}(p_{1,1}, p_{1,2}, \dots, p_{m,r})\\ &\quad = \mathrm{H}_m(q_1, \dots, q_m) + \sum_{j=1}^{m} q_j \, \mathrm{H}_r\left(\frac{p_{j,1}}{q_j}, \dots, \frac{p_{j,r}}{q_j}\right),\end{aligned} \tag{1.149}$$

   i.e., the uncertainty can be split into the uncertainty of choosing a group and the uncertainty of choosing one particular outcome of the chosen group, averaged over all groups.

Theorem 1.27. The only functions H_r that satisfy the above four conditions are of the form

$$\mathrm{H}_r(p_1, \dots, p_r) = -c \sum_{i=1}^{r} p_i \ln p_i \tag{1.150}$$

where the constant $c > 0$ decides about the units of H_r.

Proof: This theorem was proven by Khinchin in 1956, i.e., after Shannon had defined entropy. The article was firstly published in Russian [Khi56], and then in 1957 it was translated into English [Khi57]. We omit the details. □

Chapter 2

Review of Probability Theory

Information theory basically is applied probability theory. So it is quite important to feel comfortable with the main definitions and properties of probability. In this chapter we will very briefly review some of them and also quickly review a very important inequality with regard to expectations.

2.1 Discrete Probability Theory

Probability theory is based on three specific objects:

- the sample space Ω,
- the set of all events $\mathcal{F}$, and
- the probability measure $\Pr(\cdot)$.

The *sample space* Ω is the set of all possible outcomes of a random experiment. In discrete probability theory, the sample space is finite (i.e., $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$) or at most countably infinite (which we will indicate by $n = \infty$).

An *event* is any subset of Ω, including the *impossible event* $\emptyset$ (the empty subset of Ω) and the *certain event* Ω. An event occurs when the outcome of the random experiment is member of that event. All events are collected in the *set of events* $\mathcal{F}$, i.e., $\mathcal{F}$ is a set whose elements are sets themselves.

The *probability measure* $\Pr(\cdot)$ is a mapping that assigns to each event a real number (called the *probability* of that event) between 0 and 1 in such a way that

$$\Pr(\Omega) = 1 \tag{2.1}$$

and

$$\Pr(\mathcal{A} \cup \mathcal{B}) = \Pr(\mathcal{A}) + \Pr(\mathcal{B}) \quad \text{if } \mathcal{A} \cap \mathcal{B} = \emptyset, \tag{2.2}$$

where $\mathcal{A}, \mathcal{B} \in \mathcal{F}$ denote any two events. Notice that (2.2) implies

$$\Pr(\emptyset) = 0 \tag{2.3}$$

as we see by choosing $\mathcal{A} = \emptyset$ and $\mathcal{B} = \Omega$.

The *atomic events* are the events that contain only a single sample point, e.g., $\{\omega_i\}$. It follows from (2.2) that the numbers

$$p_i = \Pr(\{\omega_i\}), \quad i = 1, 2, \ldots, n \tag{2.4}$$

(i.e., the probabilities that the probability measure assigns to the atomic events) completely determine the probabilities of all events.

2.2 Discrete Random Variables

A *discrete random variable* is a mapping from the sample space into the real numbers. For instance, on the sample space $\Omega = \{\omega_1, \omega_2, \omega_3\}$ we might define the random variables X, Y and Z as

ω	$X(\omega)$
ω_1	-5
ω_2	0
ω_3	$+5$

ω	$Y(\omega)$
ω_1	1
ω_2	1
ω_3	0

ω	$Z(\omega)$
ω_1	13.4
ω_2	-102.44
ω_3	π

(2.5)

Note that the *range* (i.e., the set of possible values of the random variable) of these random variables are

$$X(\Omega) = \{-5, 0, +5\}, \tag{2.6}$$
$$Y(\Omega) = \{0, 1\}, \tag{2.7}$$
$$Z(\Omega) = \{-102.44, \pi, 13.4\}. \tag{2.8}$$

The *probability distribution* (or "frequency distribution") of a random variable X, denoted $P_X(\cdot)$, is closely related to the probability measure of the underlying random experiment: It is the mapping from $X(\Omega)$ into the interval $[0, 1]$ such that

$$P_X(x) \triangleq \Pr(\{\omega\colon X(\omega) = x\}), \tag{2.9}$$

i.e., it is the mapping that, for every $x \in X(\Omega)$, gives the probability that the random variable X takes on this value x. Usually, P_X is called the *probability mass function (PMF)* of X.

It is quite common to use a bit more sloppy notation and to neglect the mapping-nature of a random variable. I.e., we usually write X and not $X(\omega)$,

and for the event $\{\omega\colon X(\omega) = x\}$ we simply write $[X = x]$. So, the PMF is then expressed as

$$P_X(x) = \Pr[X = x]. \tag{2.10}$$

Note that we use square brackets instead of the round brackets in (2.9) to emphasize that behind X there still is a random experiment that causes the randomness of the function X. The range of the random variable is commonly denoted by

$$\mathcal{X} \triangleq X(\Omega) \tag{2.11}$$

and is called *alphabet* of X.

Note that it follows immediately from (2.9) that the PMF satisfies

$$P_X(x) \geq 0, \quad \text{all } x \in \mathcal{X} \tag{2.12}$$

and

$$\sum_x P_X(x) = 1, \tag{2.13}$$

where the summation in (2.13) is understood to be over all x in $\mathcal{X}$. Equations (2.12) and (2.13) are the only mathematical requirements on a probability distribution, i.e., any function which satisfies (2.12) and (2.13) is the PMF of some suitably defined random variable with some suitable alphabet.

In discrete probability theory, there is no fundamental distinction between a random variable X and a *random vector* $\mathbf{X}$: While the random variable takes value in $\mathbb{R}$, a random n-vector takes value in $\mathbb{R}^n$. So basically, the only difference is the alphabet of X and $\mathbf{X}$. However, if $X_1, X_2, \ldots, X_n$ are random variables with alphabets $\mathcal{X}_1, \ldots, \mathcal{X}_n$, it is often convenient to consider their *joint probability distribution* or *joint PMF* defined as the mapping from $\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n$ into the interval $[0, 1]$ such that

$$P_{X_1,X_2,\ldots,X_n}(x_1, x_2, \ldots, x_n) = \Pr[X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n]. \tag{2.14}$$

(Note that strictly speaking we assume here that these RVs share a common sample space Ω such that the joint PMF actually describes the probability of the event $\{\omega\colon X_1(\omega) = x_1\} \cap \{\omega\colon X_2(\omega) = x_2\} \cap \cdots \cap \{\omega\colon X_n(\omega) = x_n\}$.)

It follows again immediately that

$$P_{X_1,X_2,\ldots,X_n}(x_1, x_2, \ldots, x_n) \geq 0 \tag{2.15}$$

and that

$$\sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} P_{X_1,X_2,\ldots,X_n}(x_1, x_2, \ldots, x_n) = 1. \tag{2.16}$$

More interestingly, it follows from (2.14) that

$$\sum_{x_i} P_{X_1,X_2,\ldots,X_n}(x_1,x_2,\ldots,x_n) = P_{X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_n}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n). \tag{2.17}$$

The latter is called *marginal distribution* of the PMF of $\mathbf{X}$.

The random variables $X_1, X_2, \ldots, X_n$ are said to be *statistically independent* when

$$P_{X_1,X_2,\ldots,X_n}(x_1,x_2,\ldots,x_n) = P_{X_1}(x_1)\cdot P_{X_2}(x_2)\cdots P_{X_n}(x_n) \tag{2.18}$$

for all $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, \ldots, x_n \in \mathcal{X}_n$.

If X and Y are independent, then we also write $X \perp\!\!\!\perp Y$.

Suppose that g is a real-valued function whose domain includes $\mathcal{X}$. Then, the *expectation* of $g(X)$, denoted $\mathsf{E}[g(X)]$, is the real number

$$\mathsf{E}[g(X)] \triangleq \sum_x P_X(x)\, g(x). \tag{2.19}$$

The term *average* is synonymous with expectation. Similarly, when g is a real-valued function whose domain includes $\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n$, one defines

$$\mathsf{E}[g(X_1,X_2,\ldots,X_n)] \triangleq \sum_{x_1}\sum_{x_2}\cdots\sum_{x_n} P_{X_1,X_2,\ldots,X_n}(x_1,x_2,\ldots,x_n)\, g(x_1,x_2,\ldots,x_n). \tag{2.20}$$

It is often convenient to consider *conditional probability distributions*. If $P_X(x) > 0$, then one defines

$$P_{Y|X}(y|x) \triangleq \frac{P_{X,Y}(x,y)}{P_X(x)}. \tag{2.21}$$

It follows from (2.21) and (2.17) that

$$P_{Y|X}(y|x) \geq 0, \quad \text{for all } y \in \mathcal{Y} \tag{2.22}$$

and

$$\sum_y P_{Y|X}(y|x) = 1. \tag{2.23}$$

Thus, mathematically, there is no fundamental difference between a conditional probability distribution for Y (given, say, $X = x$) and the (unconditioned) probability distribution for Y. When $P_X(x) = 0$, we cannot of course use (2.21) to define $P_{Y|X}(y|x)$. It is often said that $P_{Y|X}(y|x)$ is "undefined" in this case, but it is better to say that $P_{Y|X}(y|x)$ can be arbitrarily specified, provided that (2.22) and (2.23) are satisfied by the specification. This latter

practice is often done in information theory to avoid having to treat as special cases those uninteresting situations where the conditioning event has zero probability.

Note that the concept of conditioning allows a more intuitive explanation of the definition of independence as given in (2.18). If two random variables X and Y are independent, i.e.,

$$P_{X,Y}(x,y) = P_X(x)P_Y(y), \tag{2.24}$$

then

$$P_{Y|X}(y|x) = \frac{P_{X,Y}(x,y)}{P_X(x)} = \frac{P_X(x)P_Y(y)}{P_X(x)} = P_Y(y). \tag{2.25}$$

Hence, we see that the probability distribution of Y remains unchanged irrespective of what value X takes on, i.e., they are independent!

If g is a real-valued function whose domain includes $\mathcal{X}$, then the *conditional expectation* of $g(X)$ given the occurrence of the event $\mathcal{A}$ is defined as

$$\mathsf{E}[g(X)\,|\,\mathcal{A}] \triangleq \sum_x g(x)\,\Pr[X=x\,|\,\mathcal{A}]. \tag{2.26}$$

Choosing $\mathcal{A} \triangleq \{Y=y_0\}$ for some random variable Y and some value $y_0 \in \mathcal{Y}$, we see that (2.26) implies

$$\mathsf{E}[g(X)\,|\,Y=y_0] = \sum_x g(x)P_{X|Y}(x|y_0). \tag{2.27}$$

More generally, when g is a real-valued function whose domain includes $\mathcal{X}\times\mathcal{Y}$, the definition (2.26) implies

$$\mathsf{E}[g(X,Y)\,|\,\mathcal{A}] = \sum_x \sum_y g(x,y)\,\Pr(\{X=x\}\cap\{Y=y\}\,|\,\mathcal{A}). \tag{2.28}$$

Again, nothing prevents us from choosing $\mathcal{A} \triangleq \{Y=y_0\}$, in which case (2.28) reduces to

$$\mathsf{E}[g(X,Y)\,|\,Y=y_0] = \sum_x g(x,y_0)P_{X|Y}(x|y_0), \tag{2.29}$$

as follows from the fact that $\Pr(\{X=x\}\cap\{Y=y\}\,|\,\{Y=y_0\})$ vanishes for all y except $y=y_0$, in which case it has the value $\Pr[X=x\,|\,Y=y_0] = P_{X|Y}(x|y_0)$. Multiplying both sides of (2.29) by $P_Y(y_0)$ and summing over y_0 gives the relation

$$\mathsf{E}[g(X,Y)] = \sum_y \mathsf{E}[g(X,Y)\,|\,Y=y]P_Y(y), \tag{2.30}$$

where we have used (2.20) and where we have changed the dummy variable of summation from y_0 to y for clarity. Similarly, conditioning on an event $\mathcal{A}$, we would obtain

$$\mathsf{E}[g(X,Y)\,|\,\mathcal{A}] = \sum_y \mathsf{E}\left[g(X,Y)\,\middle|\,\{Y=y\}\cap\mathcal{A}\right]\cdot \Pr[Y=y\,|\,\mathcal{A}], \quad (2.31)$$

which in fact reduces to (2.30) when one chooses $\mathcal{A}$ to be the certain event Ω. Both (2.30) and (2.31) are referred to as statements of the *theorem on total expectation*, and are exceedingly useful in the calculation of expectations.

A sequence $Y_1, Y_2, Y_3, \ldots$ of random variables is said to *converge in probability* to the random variable Y, denoted

$$Y = \operatorname*{plim}_{n\to\infty} Y_n, \quad (2.32)$$

if for every positive ϵ it is true that

$$\lim_{n\to\infty} \Pr[|Y - Y_n| < \epsilon] = 1. \quad (2.33)$$

Roughly speaking, the random variables $Y_1, Y_2, Y_3, \ldots$ converge in probability to the random variable Y if, for every large n, it is virtually certain that the random variable Y_n will take on a value very close to that of Y. Suppose that $X_1, X_2, X_3, \ldots$ is a sequence of statistically independent and identically-distributed (i.i.d.) random variables, let m denote their common expectation

$$m = \mathsf{E}[X_i], \quad (2.34)$$

and let

$$Y_n \triangleq \frac{X_1 + X_2 + \cdots + X_n}{n}. \quad (2.35)$$

Then the *weak law of large numbers* asserts that

$$\operatorname*{plim}_{n\to\infty} Y_n = m, \quad (2.36)$$

i.e., that this sequence $Y_1, Y_2, Y_3, \ldots$ of random variables converges in probability to (the constant random variable whose value is always) m. Roughly speaking, the law of large numbers states that, for every large n, it is virtually certain that $\frac{X_1+X_2+\cdots+X_n}{n}$ will take on a value close to m.

2.3 Continuous Random Variables

In contrast to a discrete random variable with a finite or countably infinite alphabet, a continuous random variable can take on uncountably many values. Unfortunately, once the sample space contains uncountably many elements, some subtle mathematical problems can pop up. So for example, the set of

all events $\mathcal{F}$ can not be defined anymore as set of all subsets, but needs to satisfy certain constraints, or the alphabet needs to be "decent" enough. In this very brief review we will omit these mathematical details. Moreover, we concentrate on a special family of random variables of uncountably infinite alphabet that satisfy some nice properties: *continuous random variables*.

A random variable X is called *continuous* if there exists a nonnegative function f_X, called *probability density function (PDF)*, such that for every subset of the alphabet $\mathcal{B} \subseteq \mathcal{X}$, we have

$$\Pr[X \in \mathcal{B}] = \int_{\mathcal{B}} f_X(x)\,\mathrm{d}x. \tag{2.37}$$

This function is called *density* because it describes the *probability per length*, similarly to a volume mass density giving the mass per volume, or the surface charge density giving the electrical charge per surface area. So for example,

$$\Pr[a \le X \le b] = \int_a^b f_X(x)\,\mathrm{d}x. \tag{2.38}$$

By definition we have

$$f_X(x) \ge 0, \quad x \in \mathcal{X}, \tag{2.39}$$

and from (2.37) it follows that

$$\int_{-\infty}^{\infty} f_X(x)\,\mathrm{d}x = \Pr[X \in \mathbb{R}] = 1. \tag{2.40}$$

These two properties fully correspond to (2.12) and (2.13), respectively. However, note that $f_X(x) > 1$ is possible, it is even possible that f_X is unbounded. As an example, consider the following PDF:

$$f_X(x) = \begin{cases} \frac{1}{4\sqrt{x}} & 0 < x \le 4, \\ 0 & \text{otherwise.} \end{cases} \tag{2.41}$$

Obviously, (2.39) is satisfied. We quickly check that also (2.40) holds:

$$\int_{-\infty}^{\infty} f_X(x)\,\mathrm{d}x = \int_0^4 \frac{1}{4\sqrt{x}}\,\mathrm{d}x = \left.\frac{\sqrt{x}}{2}\right|_{x=0}^{4} = \frac{\sqrt{4}}{2} - \frac{\sqrt{0}}{2} = 1. \tag{2.42}$$

The generalization to continuous random vectors is straightforward. We define the *joint PDF* of $X_1, \dots, X_n$ such that for any subset $\mathcal{B} \subseteq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$

$$\Pr[\mathbf{X} \in \mathcal{B}] = \iint \cdots \int_{\mathcal{B}} f_{X_1,\dots,X_n}(x_1, \dots, x_n)\,\mathrm{d}x_1\,\mathrm{d}x_2 \cdots \mathrm{d}x_n. \tag{2.43}$$

The definition of *expectation* for continuous random variables corresponds to (2.19):

$$\mathsf{E}[g(X)] \triangleq \int_{-\infty}^{\infty} f_X(x) g(x)\,\mathrm{d}x. \tag{2.44}$$

Similarly, also the definition of independence, conditional PDFs, and the Total Expectation Theorem can be taken over from the discrete probability theory.

2.4 Jensen Inequality

An important basic property of probability theory and convex or concave function is called *Jensen Inequality*. Note that in Chapter 9 we will study convexity and its properties more in detail.

Theorem 2.1 (Jensen Inequality). If $f(\cdot)$ is a concave function over an interval $\mathcal{I}$ and $X \in \mathcal{X}$ is a RV, where $\mathcal{X} \subset \mathcal{I}$, then

$$\mathsf{E}[f(X)] \le f(\mathsf{E}[X]). \tag{2.45}$$

Moreover, if $f(\cdot)$ is strictly concave (i.e., if $f(\cdot)$ is double-differentiable it satisfies $\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2} < 0$), then

$$\mathsf{E}[f(X)] = f(\mathsf{E}[X]) \iff X = \text{constant}. \tag{2.46}$$

If in the above "concave" is replaced by "convex", all inequalities have to be swapped, too.

Proof: A graphical explanation of why the Jensen Inequality holds is shown in Figure 2.1. For the special case when $f(\cdot)$ is double-differentiable, we can prove it as follows. Since $f(\cdot)$ is double-differentiable and concave, we know that $f''(\xi) \le 0$ for all $\xi \in \mathcal{I}$. So, using a Taylor expansion around a point $x_0 \in \mathcal{I}$ and the correction term, we see that for any $x \in \mathcal{I}$ and some ξ that lies between x and x_0,

$$\begin{aligned} f(x) &= f\big(x_0 + (x - x_0)\big) && (2.47)\\ &= f(x_0) + f'(x_0)(x - x_0) + \underbrace{f''(\xi)}_{\le 0} \underbrace{(x - x_0)^2}_{\ge 0} && (2.48)\\ &\le f(x_0) + f'(x_0)(x - x_0). && (2.49) \end{aligned}$$

Taking the expectation over both sides and choosing $x_0 \triangleq \mathsf{E}[X]$, we finally obtain

$$\begin{aligned} \mathsf{E}[f(X)] &\le f(x_0) + f'(x_0)(\mathsf{E}[X] - x_0) && (2.50)\\ &= f(\mathsf{E}[X]) + f'(\mathsf{E}[X])(\mathsf{E}[X] - \mathsf{E}[X]) && (2.51)\\ &= f(\mathsf{E}[X]), && (2.52) \end{aligned}$$

proving the claim. □

Remark 2.2. An easy way to remember the difference between "concave" and "convex" is to recall that a con*cave* function looks like "the entrance to a *cave*". △

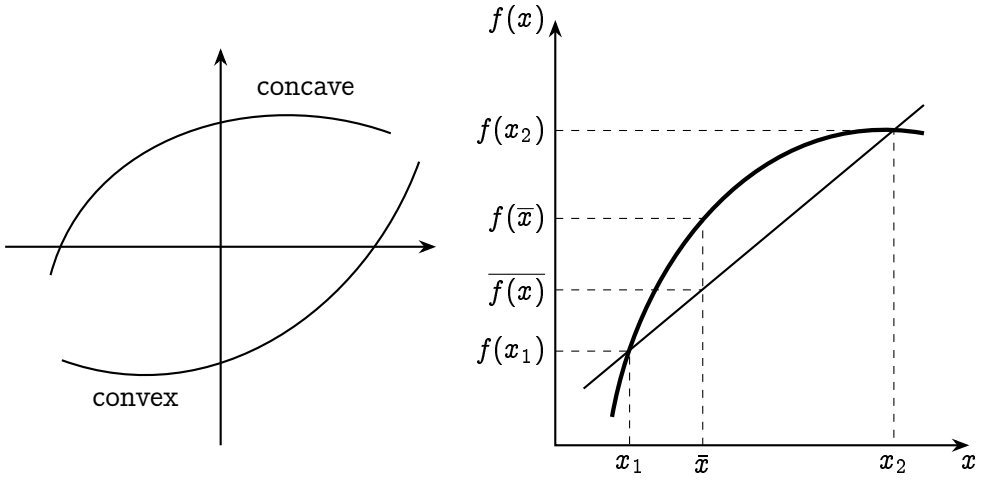


Figure 2.1: Graphical proof of the Jensen Inequality: Fix some x_1 and x_2 and let $\overline{x} \triangleq px_1 + (1-p)x_2$ for some $p \in [0,1]$. Then $f(\overline{x})$ is always *above* $\overline{f(x)} \triangleq pf(x_1) + (1-p)f(x_2)$.

Chapter 3

Entropy, Relative Entropy, and $\mathcal{L}_1$-Distance

In this chapter we will introduce some more information measures and show their relation to each other and to entropy. Moreover, we return to the question under which situation entropy is maximized.

Most material in this chapter is advanced and can therefore safely be omitted in a first reading, with the exception of the first section: relative entropy will show up from time to time throughout the script.

3.1 Relative Entropy

Beside the Shannon entropy as defined in Definition 1.4 that describes the uncertainty of a random experiment with given PMF, there also exist quantities that compare two random experiments (or rather the PMFs describing these experiments). In this class, we will touch on such quantities only very briefly.

Definition 3.1. Let $P(\cdot)$ and $Q(\cdot)$ be two PMFs over the same finite (or countably infinite) alphabet $\mathcal{X}$. The *relative entropy* or *Kullback–Leibler divergence* between $P(\cdot)$ and $Q(\cdot)$ is defined as

$$\mathscr{D}(P\|Q) \triangleq \sum_{x\in\operatorname{supp}(P)} P(x)\log\frac{P(x)}{Q(x)} = \mathsf{E}_P\left[\log\frac{P(X)}{Q(X)}\right]. \tag{3.1}$$

Remark 3.2. Note that $\mathscr{D}(P\|Q) = \infty$ if there exists an $x \in \operatorname{supp}(P)$ (i.e., $P(x) > 0$) such that $Q(x) = 0$:

$$P(x)\log\frac{P(x)}{0} = \infty. \tag{3.2}$$

So, strictly speaking, we should define relative entropy as follows:

$$\mathscr{D}(P\|Q) \triangleq \begin{cases} \sum_{x\in\operatorname{supp}(P)} P(x)\log\frac{P(x)}{Q(x)} & \text{if } \operatorname{supp}(P)\subseteq\operatorname{supp}(Q),\\ \infty & \text{otherwise,}\end{cases} \tag{3.3}$$

but again, we are lazy in notation and will usually simply write

$$\mathscr{D}(P\,\|\,Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}. \tag{3.4}$$

△

For us the most important property of $\mathscr{D}(\cdot\,\|\,\cdot)$ is its nonnegativity.

Theorem 3.3.

$$\mathscr{D}(P\,\|\,Q) \geq 0 \tag{3.5}$$

with equality if, and only if, $P(\cdot) = Q(\cdot)$.

Proof: In the case when $\operatorname{supp}(P) \not\subseteq \operatorname{supp}(Q)$, we have $\mathscr{D}(P\,\|\,Q) = \infty > 0$ trivially. So, we assume that $\operatorname{supp}(P) \subseteq \operatorname{supp}(Q)$. Then,

$$\begin{aligned}
-\mathscr{D}(P\,\|\,Q) &= \sum_{x\in\operatorname{supp}(P)} P(x) \log \underbrace{\frac{Q(x)}{P(x)}}_{\triangleq \xi} && (3.6)\\
&\leq \sum_{x\in\operatorname{supp}(P)} P(x)\left(\frac{Q(x)}{P(x)} - 1\right) \cdot \log e && (3.7)\\
&= \sum_{x\in\operatorname{supp}(P)} (Q(x) - P(x)) \log e && (3.8)\\
&= \Bigg(\underbrace{\sum_{x\in\operatorname{supp}(P)} Q(x)}_{\leq 1} - \underbrace{\sum_{x\in\operatorname{supp}(P)} P(x)}_{=1}\Bigg) \log e && (3.9)\\
&\leq (1-1)\cdot \log e && (3.10)\\
&= 0. && (3.11)
\end{aligned}$$

Here, (3.7) follows from the IT Inequality (Proposition 1.12) and (3.10) by adding additional terms to the sum. Hence, $\mathscr{D}(P\,\|\,Q) \geq 0$.

Equality can be achieved if, and only if,

1. in (3.7), in the IT Inequality $\xi = 1$, i.e., if $\frac{Q(x)}{P(x)} = 1$ for all $x \in \operatorname{supp}(P)$, i.e., if $P(x) = Q(x)$, for all x; and if
2. in (3.10), $\operatorname{supp}(P) = \operatorname{supp}(Q)$.

Note that if Condition 1 is satisfied, Condition 2 is also satisfied. □

It is quite tempting to think of $\mathscr{D}(P\,\|\,Q)$ as a "distance" between $P(\cdot)$ and $Q(\cdot)$, in particular because $\mathscr{D}(P\,\|\,Q)$ is nonnegative and is equal to zero only if $P(\cdot)$ is equal to $Q(\cdot)$. However, this is not correct because the relative entropy is not symmetric,

$$\mathscr{D}(P\,\|\,Q) \neq \mathscr{D}(Q\,\|\,P), \tag{3.12}$$

and because it does not satisfy the Triangle Inequality

$$\mathscr{D}(P_1 \| P_3) \lesseqgtr \mathscr{D}(P_1 \| P_2) + \mathscr{D}(P_2 \| P_3). \tag{3.13}$$

This is also the reason why it should *not* be called "Kullback–Leibler distance". Indeed, relative entropy behaves like a *squared* distance, so one should think of it as an "energy" rather than distance. (It actually describes the inefficiency of assuming that the PMF is $Q(\cdot)$ when the true distribution is $P(\cdot)$.)

We have actually encountered the relative entropy already, even then we did not point this out. From (1.121) it can be seen that mutual information actually is the relative entropy between the joint PMF $P_{X,Y}$ and the product of its marginals:

$$\mathrm{I}(X;Y) = \mathscr{D}(P_{X,Y} \| P_X \cdot P_Y). \tag{3.14}$$

Hence, $\mathrm{I}(X;Y)$ is the divergence between the joint distribution of X and Y (i.e., $P_{X,Y}$) and the joint distribution of X and Y if X and Y were independent (i.e., $P_X \cdot P_Y$). We thus see another explanation of mutual information: It describes how different the joint distribution of X and Y is from the situation then X and Y were independent of each (but with the same marginals).

3.2 $\mathcal{L}_1$-Distance

Another quantity that compares two PMFs is the *$\mathcal{L}_1$-distance*. In contrast to relative entropy, this is a true distance.

Definition 3.4. Let $P(\cdot)$ and $Q(\cdot)$ be two PMFs over the same finite (or countably infinite) alphabet $\mathcal{X}$. The *$\mathcal{L}_1$-distance* between $P(\cdot)$ and $Q(\cdot)$ is defined as

$$\mathscr{V}(P,Q) \triangleq \sum_{x\in\mathcal{X}} \big|P(x) - Q(x)\big|. \tag{3.15}$$

Here it is obvious from the definition that $\mathscr{V}(P,Q) \geq 0$ with equality if, and only if, $P(\cdot) = Q(\cdot)$. It is slightly less obvious that $\mathscr{V}(P,Q) \leq 2$.

Since $\mathscr{V}(\cdot,\cdot)$ satisfies all required conditions of a norm, it is correct to think of the $\mathcal{L}_1$-distance as a distance between $P(\cdot)$ and $Q(\cdot)$. It describes how similar (or different) two random experiments are.

Remark 3.5. The $\mathcal{L}_1$-distance is strongly related to the *total variation distance*:

$$\mathscr{V}_{\mathrm{tot}}(Q_1,Q_2) = \frac{1}{2}\,\mathscr{V}(Q_1,Q_2). \tag{3.16}$$

For more details, we refer to [Mos22, Section 2.7]. △

There exists a bound that links entropy, relative entropy and $\mathcal{L}_1$-distance:

Theorem 3.6. For any two PMFs $P(\cdot)$ and $Q(\cdot)$ over the same finite alphabet $\mathcal{X}$, it holds

$$\mathscr{D}(P\|Q) \le \mathrm{H}(Q) - \mathrm{H}(P) + \mathscr{V}(P,Q)\log\frac{1}{Q_{\min}}, \tag{3.17}$$

where

$$Q_{\min} \triangleq \min_{x\in\mathcal{X}} Q(x). \tag{3.18}$$

(See Section 1.4.2 for an explanation of the notation $\mathrm{H}(Q)$ and $\mathrm{H}(P)$.)

Proof:

$$\begin{aligned}
&\mathscr{D}(P\|Q)\\
&\quad= \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} && (3.19)\\
&\quad= \sum_{x\in\mathcal{X}} P(x)\log P(x) + \sum_{x\in\mathcal{X}} P(x)\log\frac{1}{Q(x)} && (3.20)\\
&\quad= -\mathrm{H}(P) + \sum_{x\in\mathcal{X}} \big(P(x)-Q(x)+Q(x)\big)\log\frac{1}{Q(x)} && (3.21)\\
&\quad= -\mathrm{H}(P) + \sum_{x\in\mathcal{X}} \big(P(x)-Q(x)\big)\log\frac{1}{Q(x)} + \sum_{x\in\mathcal{X}} Q(x)\log\frac{1}{Q(x)} && (3.22)\\
&\quad= -\mathrm{H}(P) + \sum_{x\in\mathcal{X}} \big(P(x)-Q(x)\big)\log\frac{1}{Q(x)} + \mathrm{H}(Q) && (3.23)\\
&\quad\le \mathrm{H}(Q) - \mathrm{H}(P) + \sum_{x\in\mathcal{X}} \big|P(x)-Q(x)\big|\log\frac{1}{Q(x)} && (3.24)\\
&\quad\le \mathrm{H}(Q) - \mathrm{H}(P) + \sum_{x\in\mathcal{X}} \big|P(x)-Q(x)\big|\log\frac{1}{Q_{\min}} && (3.25)\\
&\quad= \mathrm{H}(Q) - \mathrm{H}(P) + \mathscr{V}(P,Q)\log\frac{1}{Q_{\min}}. && (3.26)
\end{aligned}$$

□

Another inequality that links relative entropy and $\mathcal{L}_1$-distance is the *Pinsker Inequality* (see [Mos22, Section 2.8]). Moreover, some more properties of $\mathscr{V}(\cdot,\cdot)$ and its relation to entropy are discussed next.

3.3 Relations between Entropy and $\mathcal{L}_1$-Distance

3.3.1 Estimating PMFs

Suppose we have a RV $X \in \mathcal{X}$ with an unknown PMF $P(\cdot)$ that we would like to estimate, and suppose that by some estimation process we come up with an estimate $\hat{P}(\cdot)$ for the unknown $P(\cdot)$. If we now use $\hat{P}(\cdot)$ to compute the entropy $\mathrm{H}(\hat{X})$, then how good an approximation is this for the entropy $\mathrm{H}(X)$?

As we will show next, unfortunately, $\mathrm{H}(\hat{X})$ can be arbitrarily far from $\mathrm{H}(X)$ even if $P(\cdot)$ and $\hat{P}(\cdot)$ are very similar!

Theorem 3.7 ([HY10, Theorem 1]). Suppose $\epsilon > 0$ and $\delta > 0$ are given. Then for any PMF $\hat{P}(\cdot)$ with a support of size $\hat{r}$, there exists another PMF $P(\cdot)$ of support size $r \geq \hat{r}$ large enough such that

$$\mathscr{V}(P, \hat{P}) < \epsilon \tag{3.27}$$

but

$$\mathrm{H}(P) - \mathrm{H}(\hat{P}) > \delta. \tag{3.28}$$

We see that if we do not know the support size r of $P(\cdot)$, then even if our estimate $\hat{P}(\cdot)$ is arbitrarily close to the correct $P(\cdot)$ (with respect to the $\mathcal{L}_1$-distance), the difference between $\mathrm{H}(P)$ and $\mathrm{H}(\hat{P})$ remains unbounded.

Proof: For some $\hat{r}$, let

$$\hat{P}(\cdot) = (p_1, p_2, \ldots, p_{\hat{r}}), \tag{3.29}$$

where we have used the vector-like notation for the PMF introduced in Section 1.4.2. Moreover, let

$$P(\cdot) = \left(p_1 - \frac{p_1}{\sqrt{\log t}}, p_2 + \frac{p_1}{t\sqrt{\log t}}, \ldots, p_{\hat{r}} + \frac{p_1}{t\sqrt{\log t}}, \frac{p_1}{t\sqrt{\log t}}, \ldots, \frac{p_1}{t\sqrt{\log t}}\right) \tag{3.30}$$

be a PMF with $r = t+1 \geq \hat{r}$ probability masses, $t \in \mathbb{N}$. Note that $P(\cdot)$ indeed is a PMF:

$$\sum_{i=1}^{r} P(i) = \sum_{i=1}^{\hat{r}} p_i - \frac{p_1}{\sqrt{\log t}} + t \cdot \frac{p_1}{t\sqrt{\log t}} = 1 - \frac{p_1}{\sqrt{\log t}} + \frac{p_1}{\sqrt{\log t}} = 1. \tag{3.31}$$

For this choice of $P(\cdot)$ and $\hat{P(\cdot)}$ we have[1]

$$\mathscr{V}(P, \hat{P}) = \frac{p_1}{\sqrt{\log t}} + t \cdot \frac{p_1}{t\sqrt{\log t}} = \frac{2p_1}{\sqrt{\log t}}, \tag{3.32}$$

$$\begin{aligned} \mathrm{H}(\hat{P}) = &- \left(p_1 - \frac{p_1}{\sqrt{\log t}}\right) \log\left(p_1 - \frac{p_1}{\sqrt{\log t}}\right) \\ &- \sum_{i=2}^{\hat{r}} \left(p_i + \frac{p_1}{t\sqrt{\log t}}\right) \log\left(p_i + \frac{p_1}{t\sqrt{\log t}}\right) \\ &- (t + 1 - \hat{r}) \frac{p_1}{t\sqrt{\log t}} \log \frac{p_1}{t\sqrt{\log t}} \end{aligned} \tag{3.33}$$

$$\approx \mathrm{H}(P) + \frac{p_1}{\sqrt{\log t}} \log \frac{t\sqrt{\log t}}{p_1} \tag{3.34}$$

[1] For the definition of $\mathscr{V}$ it is assumed that $P(\cdot)$ and $\hat{P}(\cdot)$ take value in the same alphabet. We can easily fix by adding the right number zeros to the probability vector of $P(\cdot)$.

$$= \mathrm{H}(P) + \frac{p_1}{\sqrt{\log t}} \log t + \frac{p_1}{\sqrt{\log t}} \log \frac{\sqrt{\log t}}{p_1} \tag{3.35}$$

$$= \mathrm{H}(P) + p_1 \sqrt{\log t} + \frac{p_1}{\sqrt{\log t}} \log \frac{\sqrt{\log t}}{p_1} \tag{3.36}$$

$$\approx \mathrm{H}(P) + p_1 \sqrt{\log t}, \tag{3.37}$$

where the approximations become more accurate for larger values of t. If we let t become very large, then $\mathscr{V}(P, \hat{P})$ becomes arbitrarily small, while $p_1\sqrt{\log t}$ is unbounded. □

Note that if we fix the support size r, then Theorem 3.7 does not hold anymore, and we will show next that then the difference between $\mathrm{H}(X)$ and $\mathrm{H}(\hat{X})$ is bounded.

3.3.2 Extremal Entropy for given $\mathcal{L}_1$-Distance

We will next investigate how one needs to adapt a given PMF $P(\cdot)$ in order to maximize or minimize the entropy, when the allowed changes on the PMF are limited:

$$\max_{Q\colon \mathscr{V}(P,Q)\le\epsilon} \mathrm{H}(Q(\cdot)) \quad \text{or} \quad \min_{Q\colon \mathscr{V}(P,Q)\le\epsilon} \mathrm{H}(Q(\cdot)). \tag{3.38}$$

Recall that without any limitations on the PMF, we know from Theorem 1.13 that to maximize entropy we need a uniform distribution, while to minimize we make it extremely peaky with one value being 1 and the rest 0. Both such changes, however, will usually cause a large $\mathcal{L}_1$-distance. So the question is how to adapt a PMF without causing too much $\mathcal{L}_1$-distance, but to maximize (or minimize) the entropy.

In the remainder of this section, we assume that the given PMF $P(\cdot) = (p_1, \dots, p_{|\mathcal{X}|})$ is ordered such that

$$p_1 \ge p_2 \ge \cdots \ge p_r > 0 = p_{r+1} = \cdots = p_{|\mathcal{X}|}. \tag{3.39}$$

We again use r as the support size of $P(\cdot)$.

We will omit most proofs in this section and refer to [HY10] instead.

Theorem 3.8 ([HY10, Theorem 2]). Let $0 \le \epsilon \le 2$ and $P(\cdot)$ satisfying (3.39) be given. Here we must restrict $|\mathcal{X}|$ to be finite. Let $\mu, \nu \in \mathbb{R}$ be such that

$$\sum_{i=1}^{|\mathcal{X}|} (p_i - \mu)^+ = \frac{\epsilon}{2} \tag{3.40}$$

and

$$\sum_{i=1}^{|\mathcal{X}|} (\nu - p_i)^+ = \frac{\epsilon}{2}, \tag{3.41}$$

where

$$(\cdot)^+ \triangleq \max\{\cdot, 0\}. \tag{3.42}$$

If $\nu \geq \mu$, define $Q_{\max}(\cdot)$ to be the uniform distribution on $\mathcal{X}$,

$$Q_{\max}(i) \triangleq \frac{1}{|\mathcal{X}|}, \quad i = 1, \ldots, |\mathcal{X}|, \tag{3.43}$$

and if $\nu < \mu$, define $Q_{\max}(\cdot)$ as

$$Q_{\max}(i) \triangleq \begin{cases} \mu & \text{if } p_i > \mu, \\ p_i & \text{if } \nu \leq p_i \leq \mu, \\ \nu & \text{if } p_i < \nu, \end{cases} \quad i = 1, \ldots, |\mathcal{X}|. \tag{3.44}$$

Then

$$\max_{Q\colon \mathscr{V}(P,Q) \leq \epsilon} \mathrm{H}(Q(\cdot)) = \mathrm{H}(Q_{\max}(\cdot)). \tag{3.45}$$

Note the structure of the maximizing distribution: we cut the largest values of $P(\cdot)$ to a constant level μ and add this probability to the smallest values to make them all constant equal to ν. The middle range of the probabilities are not touched. So, under the constraint that we cannot twiddle $P(\cdot)$ too much, we should try to approach a uniform distribution by equalizing the extremes. See Figure 3.1 for an illustration of this.

It is quite obvious that $\mathrm{H}(Q_{\max})$ depends on the given ϵ. Therefore for a given $P(\cdot)$ and for $0 \leq \epsilon \leq 2$, we define

$$\psi_P(\epsilon) \triangleq \mathrm{H}(Q_{\max}(\cdot)) \tag{3.46}$$

with $Q_{\max}(\cdot)$ given in (3.43) and (3.44). One can show that $\psi_P(\epsilon)$ is a concave (and therefore continuous) and strictly increasing function in ϵ.

Theorem 3.9 ([HY10, Theorem 3]). Let $0 \leq \epsilon \leq 2$ and $P(\cdot)$ satisfying (3.39) be given ($|\mathcal{X}|$ can be infinite). If $1 - p_1 \leq \frac{\epsilon}{2}$, define

$$Q_{\min}(\cdot) \triangleq (1, 0). \tag{3.47}$$

Otherwise, let k be the largest integer such that

$$\sum_{i=k}^{r} p_i \geq \frac{\epsilon}{2} \tag{3.48}$$

and define $Q_{\min}(\cdot)$ as

$$Q_{\min}(i) \triangleq \begin{cases} p_1 + \frac{\epsilon}{2} & \text{if } i = 1, \\ p_i & \text{if } i = 2, \ldots, k-1, \\ \sum_{j=k}^{r} p_j - \frac{\epsilon}{2} & \text{if } i = k, \\ 0 & \text{if } i = k+1, \ldots, |\mathcal{X}|, \end{cases} \quad i = 1, \ldots, |\mathcal{X}|. \tag{3.49}$$

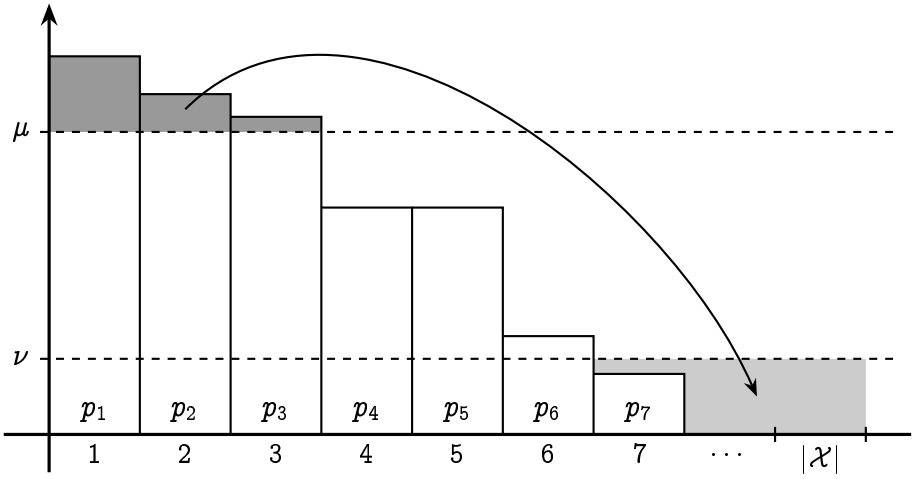


Figure 3.1: Example demonstrating how a PMF with seven nonzero probabilities is changed to maximize entropy under a $\mathcal{L}_1$-distance constraint ($|\mathcal{X}| = 9$, $r = 7$). The maximizing distribution is $Q_{\max}(\cdot) = (\mu, \mu, \mu, p_4, p_5, p_6, \nu, \nu, \nu)$.

Then

$$\min_{Q\colon\, \mathscr{V}(P,Q)\le\epsilon} \mathrm{H}(Q(\cdot)) = \mathrm{H}(Q_{\min}(\cdot)). \tag{3.50}$$

Note that to minimize entropy, we need to change the PMF to make it more concentrated. To this end, the few smallest probability values are set to zero and the corresponding amount is added to the single largest probability. The middle range of the probabilities are not touched. So, under the constraint that we cannot twiddle $P(\cdot)$ too much, we should try to approach the $(1, 0, \dots, 0)$-distribution by removing the tail and enlarge the largest peak. See Figure 3.2 for an illustration of this.

Also here, $\mathrm{H}(Q_{\min}(\cdot))$ depends on the given ϵ. For a given $P(\cdot)$ and for $0 \le \epsilon \le 2$, we define

$$\varphi_P(\epsilon) \triangleq \mathrm{H}(Q_{\min}(\cdot)) \tag{3.51}$$

with $Q_{\min}(\cdot)$ defined in (3.47)–(3.49). One can show that $\varphi_P(\epsilon)$ is a continuous and strictly decreasing function in ϵ.

We may, of course, also ask the question the other way around: For a given PMF $P(\cdot)$ and a given entropy $\mathrm{H}(X)$, what is the choice of a PMF $Q(\cdot)$ such that $\mathrm{H}(Q) = \mathrm{H}(X)$ is achieved and such that $Q(\cdot)$ is most similar to $P(\cdot)$ (with respect to $\mathcal{L}_1$-distance)?

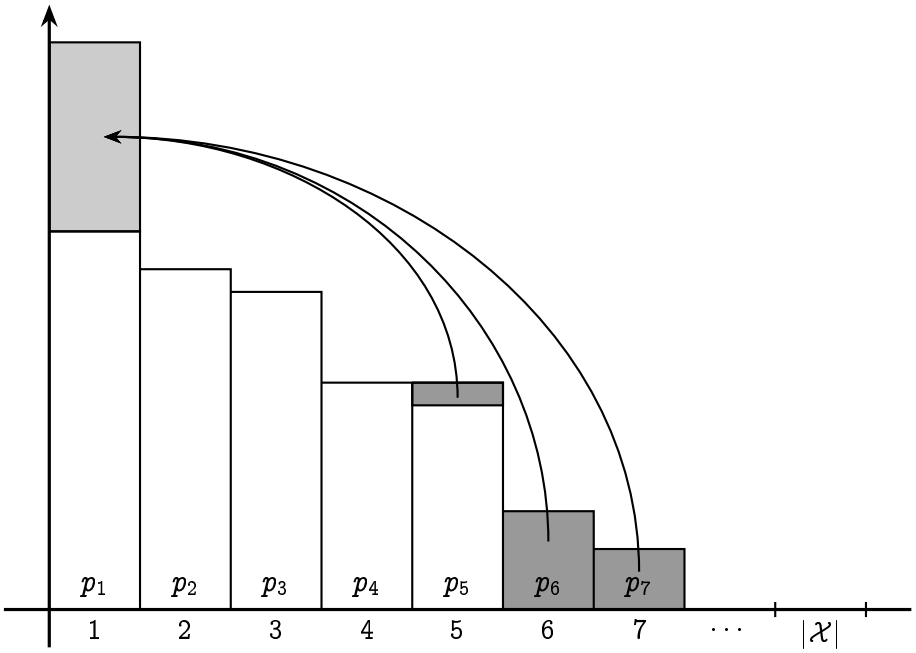


Figure 3.2: Example demonstrating how a PMF with seven nonzero probabilities is changed to minimize entropy under a $\mathcal{L}_1$-distance constraint ($r = 7$). The minimizing distribution is $Q_{\min}(\cdot) = (p_1 + \epsilon/2, p_2, p_3, p_4, p_5 + p_6 + p_7 - \epsilon/2, 0, 0, 0, 0)$.

Theorem 3.10 ([HY10, Theorem 4]). Let $0 \le t \le \log|\mathcal{X}|$ and $P(\cdot)$ satisfying (3.39) be given. Then

$$\min_{Q\colon\ \mathrm{H}(Q(\cdot))=t} \mathscr{V}(P,Q) = \begin{cases} 2(1-p_1) & \text{if } t=0,\\ \varphi_P^{-1}(t) & \text{if } 0<t\le \mathrm{H}(P(\cdot)),\\ \psi_P^{-1}(t) & \text{if } \mathrm{H}(P(\cdot))<t<\log|\mathcal{X}|,\\ \sum_{i=1}^{r}\left|p_i-\frac{1}{|\mathcal{X}|}\right|+\frac{|\mathcal{X}|-r}{|\mathcal{X}|} & \text{if } t=\log|\mathcal{X}| \end{cases} \tag{3.52}$$

with $\psi_P^{-1}(\cdot)$ and $\varphi_P^{-1}(\cdot)$ being the inverse of the functions defined in (3.46) and (3.51), respectively.

Note that this result actually is a direct consequence of Theorem 3.8 and 3.9 and the fact that $\psi_P(\epsilon)$ and $\varphi_P(\epsilon)$ both are continuous and monotonic functions that have a unique inverse.

3.3.3 Lower Bound on Entropy in Terms of $\mathcal{L}_1$-Distance

In Section 1.2.4 we have found the most general lower bound on entropy: $\mathrm{H}(X) \ge 0$. Using the results from the previous section, we can now improve

on this lower bound by taking into account the PMF of X.

Theorem 3.11. For a given $r \in \mathbb{N} \setminus \{1\}$, consider a RV X that takes value in an r-ary alphabet $\mathcal{X}$ with a PMF $P_X(\cdot) = (p_1, p_2, \dots, p_r)$. Then the entropy of X can be lower-bounded as follows:

$$\mathrm{H}(X) = \mathrm{H}(p_1, \dots, p_r) \ge \log r - \frac{r \log r}{2(r-1)} \sum_{i=1}^{r} \left| p_i - \frac{1}{r} \right|. \tag{3.53}$$

This lower bound has a beautiful interpretation: Let $X \in \mathcal{X}$ be an arbitrary RV and let U be uniformly distributed on the same alphabet $\mathcal{X}$. Then (3.53) can be rewritten as follows:

$$\mathrm{H}(U) - \mathrm{H}(X) \le \mathscr{V}(P_U, P_X) \cdot \frac{|\mathcal{X}| \log |\mathcal{X}|}{2(|\mathcal{X}| - 1)}. \tag{3.54}$$

Now recall that the entropy of a uniformly distributed RV is equal to the logarithm of the alphabet size, and if the distribution is not uniform, then the entropy is smaller. So, Theorem 3.11 gives an upper bound on this reduction in terms of the $\mathcal{L}_1$-distance between the PMF and the uniform PMF.

Proof: Suppose we can prove that

$$\varphi_{P_U}\left(\frac{2(r-1)}{r}(1-\xi) \right) \ge \xi \log r, \quad \forall\, \xi \in [0,1],\ r \in \mathbb{N} \setminus \{1\}. \tag{3.55}$$

Since $\varphi_{P_U}(\cdot)$ is monotonically decreasing, this means that

$$\frac{2(r-1)}{r}(1-\xi) \le \varphi_{P_U}^{-1}(\xi \log r), \tag{3.56}$$

and since $0 \le \mathrm{H}(P_X) \le \log r$, it then follows from Theorem 3.10 that

$$\begin{aligned}
\sum_{\ell=1}^{r} \left| p_\ell - \frac{1}{r} \right| \log r &= \mathscr{V}(P_U, P_X) \cdot \log r && (3.57)\\
&\ge \min_{Q:\ \mathrm{H}(Q) = \mathrm{H}(P_X)} \mathscr{V}(P_U, Q) \cdot \log r && (3.58)\\
&= \varphi_{P_U}^{-1}(\mathrm{H}(P_X)) \cdot \log r \qquad \text{(by Theorem 3.10)} && (3.59)\\
&= \varphi_{P_U}^{-1}\left(\frac{\mathrm{H}(P_X)}{\log r} \log r \right) \cdot \log r && (3.60)\\
&= \varphi_{P_U}^{-1}(\xi \log r)\Big|_{\xi = \frac{\mathrm{H}(P_X)}{\log r}} \cdot \log r && (3.61)\\
&\ge \frac{2(r-1)}{r}(1-\xi)\Big|_{\xi = \frac{\mathrm{H}(P_X)}{\log r}} \cdot \log r \quad \text{(by (3.56))} && (3.62)\\
&= \frac{2(r-1)}{r}\left(1 - \frac{\mathrm{H}(P_X)}{\log r} \right) \log r && (3.63)\\
&= \frac{2(r-1)}{r}(\log r - \mathrm{H}(P_X)), && (3.64)
\end{aligned}$$

from which follows (3.53).

Hence, it only remains to prove (3.55). To this end, recall the definition of $\varphi_{P_U}(\epsilon)$ in (3.51) with the $Q_{\min}(\cdot)$ given in (3.47)–(3.49) and where

$$\epsilon = \frac{2(r-1)}{r}(1-\xi) \tag{3.65}$$

for $\xi \in [0,1]$. Hence,

$$\frac{\epsilon}{2} = \frac{r-1}{r}(1-\xi) \le \frac{r-1}{r} = 1 - \frac{1}{r} = 1 - P_U(1) \tag{3.66}$$

and $Q_{\min}$ is defined by (3.49) exclusively:

$$\begin{aligned}\varphi_{P_U}(\epsilon) = &-\left(\frac{1}{r}+\frac{\epsilon}{2}\right)\log\left(\frac{1}{r}+\frac{\epsilon}{2}\right) + \frac{k-2}{r}\log r\\ &- \left(1-\frac{k-1}{r}-\frac{\epsilon}{2}\right)\log\left(1-\frac{k-1}{r}-\frac{\epsilon}{2}\right)\end{aligned} \tag{3.67}$$

with k being the largest integer satisfying (see (3.48))

$$\sum_{i=k}^{r}\frac{1}{r} = 1 - \frac{k-1}{r} \ge \frac{\epsilon}{2}, \tag{3.68}$$

i.e.,

$$k = \left\lfloor r\left(1-\frac{\epsilon}{2}\right)\right\rfloor + 1. \tag{3.69}$$

Proving (3.55) is thus equivalent to proving the nonnegativity of

$$f_r(\xi) \triangleq \frac{1}{\log r}\,\varphi_{P_U}\left(\frac{2(r-1)}{r}(1-\xi)\right) - \xi \tag{3.70}$$

for all $\xi \in [0,1]$ and for all $r \in \mathbb{N}\setminus\{1\}$. Using (3.67) and (3.69),

$$\begin{aligned}f_r(\xi) = &-\frac{1}{\log r}\left(1-\xi+\frac{\xi}{r}\right)\log\left(1-\xi+\frac{\xi}{r}\right) + \frac{1}{r}\lfloor 1+r\xi-\xi\rfloor - \frac{1}{r}\\ &- \frac{1}{\log r}\left(\frac{1}{r}+\xi-\frac{\xi}{r}-\frac{1}{r}\lfloor 1+r\xi-\xi\rfloor\right)\\ &\quad\cdot\log\left(\frac{1}{r}+\xi-\frac{\xi}{r}-\frac{1}{r}\lfloor 1+r\xi-\xi\rfloor\right) - \xi \qquad (3.71)\\ = &\,1-\xi-\frac{1}{r\log r}(r-r\xi+\xi)\log(r-r\xi+\xi)\\ &- \frac{1}{r\log r}(1+r\xi-\xi-\lfloor 1+r\xi-\xi\rfloor)\log(1+r\xi-\xi-\lfloor 1+r\xi-\xi\rfloor). \qquad (3.72)\end{aligned}$$

Note that $f_r(\xi)$ is continuous in ξ for all $\xi \in [0,1]$, but that its derivative is not continuous due to the floor-function.

Choose some ξ_{l}^* and ξ_{r}^* such that for all $\xi \in [\xi_{\mathrm{l}}^*, \xi_{\mathrm{r}}^*)$,

$$\lfloor 1 + r\xi - \xi \rfloor = \text{constant} \triangleq \beta \in \{1, \dots, r\}. \tag{3.73}$$

For such ξ, we have

$$\begin{aligned} f_r(\xi) &= 1 - \xi - \frac{1}{r \log r}(r - r\xi + \xi)\log(r - r\xi + \xi) \\ &\quad - \frac{1}{r \log r}(1 + r\xi - \xi - \beta)\log(1 + r\xi - \xi - \beta) \end{aligned} \tag{3.74}$$

with

$$\frac{\partial^2}{\partial \xi^2} f_r(\xi) = \frac{-(r-1)^2(r+1-\beta)}{r \ln r\,(r - r\xi + \xi)(\underbrace{1 + r\xi - \xi}_{\geq \beta} - \beta)} < 0, \quad \xi \in [\xi_{\mathrm{l}}^*, \xi_{\mathrm{r}}^*). \tag{3.75}$$

Hence $f_r(\xi)$ is concave over the interval $[\xi_{\mathrm{l}}^*, \xi_{\mathrm{r}}^*)$, and therefore, in this interval, $f_r(\xi)$ is lower-bounded by one of the boundary points $f_r(\xi_{\mathrm{l}}^*)$ or $f_r(\xi_{\mathrm{r}}^*)$.

So we investigate these boundary points for all possible intervals $[\xi_{\mathrm{l}}^*, \xi_{\mathrm{r}}^*)$. Let $\xi^* \in [0, 1]$ be such that

$$1 + r\xi^* - \xi^* \in \mathbb{N}, \tag{3.76}$$

i.e., for some $\ell \in \{1, \dots, r\}$, we have

$$\xi^* = \frac{\ell - 1}{r - 1}. \tag{3.77}$$

Also note that

$$\begin{aligned} &\lim_{\xi \downarrow \xi^*} (1 + r\xi - \xi - \lfloor 1 + r\xi - \xi \rfloor) \log(1 + r\xi - \xi - \lfloor 1 + r\xi - \xi \rfloor) \\ &\quad = \lim_{t \downarrow 0} t \log t = 0. \end{aligned} \tag{3.78}$$

Therefore, and by the continuity of $f_r(\xi)$, we have

$$f_r(\xi^*) = 1 - \xi^* - \frac{1}{r \log r}(r - r\xi^* + \xi^*)\log(r - r\xi^* - \xi^*) \tag{3.79}$$

$$= 1 - \frac{\ell - 1}{r - 1} - \frac{1}{r \log r}(r - \ell + 1)\log(r - \ell + 1) \tag{3.80}$$

$$\triangleq \tilde{f}(\ell). \tag{3.81}$$

We extend the definition of $\tilde{f}(\ell)$ to $\ell \in [1, r]$ and prove concavity:

$$\frac{\partial^2}{\partial \ell^2} \tilde{f}(\ell) = \frac{-1}{r \ln r\,(r - \ell + 1)} < 0. \tag{3.82}$$

Thus, $\tilde{f}(\ell)$ is lower-bounded by one of the boundary points $\tilde{f}(1)$ or $\tilde{f}(r)$. Since

$$\tilde{f}(1) = \tilde{f}(r) = 0, \tag{3.83}$$

we have finally shown that $f_r(\xi) \geq 0$. This completes the proof. □

An immediate consequence of this lower bound on the entropy is an upper bound on the relative entropy between a general P and the uniform distribution.

Corollary 3.12. For a given $r \in \{2,3,\ldots\}$, consider a RV X that takes value in an r-ary alphabet $\mathcal{X}$ with a PMF $P_X(\cdot) = (p_1, p_2, \ldots, p_r)$. Then the relative entropy between $P_X(\cdot)$ and the uniform distribution over $\mathcal{X}$ can be upper-bounded as follows:

$$\mathscr{D}(P_X \,\|\, P_U) = \mathscr{D}\Big(p_1,\ldots,p_r \Big\| \frac{1}{r},\ldots,\frac{1}{r}\Big) \le \frac{r\log r}{2(r-1)} \sum_{i=1}^{r} \Big|p_i - \frac{1}{r}\Big|. \tag{3.84}$$

Proof: Note that the entropy of X can be understood as the entropy of a uniform RV minus the relative entropy between the actual distribution of X and the uniform distribution:

$$\begin{aligned}
\mathrm{H}(X) &= \mathrm{H}(p_1,\ldots,p_r) && (3.85)\\
&= \sum_{i=1}^{r} p_i \log \frac{1}{p_i} && (3.86)\\
&= \sum_{i=1}^{r} p_i \log\Big(\frac{1/r}{p_i}\cdot\frac{1}{1/r}\Big) && (3.87)\\
&= -\mathscr{D}\Big(p_1,\ldots,p_r \Big\| \frac{1}{r},\ldots,\frac{1}{r}\Big) + \sum_{i=1}^{r} p_i \log r && (3.88)\\
&= \log r - \mathscr{D}\Big(p_1,\ldots,p_r \Big\| \frac{1}{r},\ldots,\frac{1}{r}\Big). && (3.89)
\end{aligned}$$

The result now follows from (3.53). □

3.4 Maximum Entropy Distribution

Another interesting question to ask is the following: Given an alphabet and some constraints on some expectations of the RV, what PMF maximizes entropy? We already know that without any moment constraints, the uniform PMF maximizes entropy for a given finite alphabet (see Theorem 1.13). In the following we will generalize this result.

Theorem 3.13 (Maximum Entropy Distribution).
Let the RV X with PMF $p(\cdot)$ take value in the finite alphabet $\mathcal{X}$ and satisfy the following J constraints:

$$\mathrm{E}[r_j(X)] = \sum_{x\in\mathcal{X}} p(x) r_j(x) = \alpha_j, \quad \text{for } j = 1,\ldots,\mathrm{J}, \tag{3.90}$$

for some given functions $r_1(\cdot),\ldots,r_{\mathrm{J}}(\cdot)$ and for some given values $\alpha_1,\ldots,\alpha_{\mathrm{J}}$. Then $\mathrm{H}(X)$ is maximized if, and only if,

$$p(x) = p^*(x) \triangleq e^{\lambda_0 + \sum_{j=1}^{\mathrm{J}} \lambda_j r_j(x)} \tag{3.91}$$

assuming that $\lambda_0, \ldots, \lambda_J$ can be chosen such that (3.90) is satisfied and such that

$$\sum_{x \in \mathcal{X}} p^*(x) = 1. \tag{3.92}$$

Proof: Let $q(\cdot)$ be an arbitrary PMF for X satisfying (3.90), and assume that $p^*(\cdot)$ in (3.91) exists, i.e., it satisfies (3.90) and (3.92). Then

$$\begin{aligned}
\mathrm{H}_q(X) &= -\sum_{x \in \mathcal{X}} q(x) \ln q(x) && (3.93)\\
&= -\sum_{x \in \mathcal{X}} q(x) \ln\left(\frac{q(x)}{p^*(x)} p^*(x)\right) && (3.94)\\
&= -\underbrace{\mathscr{D}(q \,\|\, p^*)}_{\geq 0} - \sum_{x \in \mathcal{X}} q(x) \ln p^*(x) && (3.95)\\
&\leq -\sum_{x \in \mathcal{X}} q(x) \ln e^{\lambda_0 + \sum_{j=1}^{J} \lambda_j r_j(x)} && (3.96)\\
&= -\sum_{x \in \mathcal{X}} q(x) \left(\lambda_0 + \sum_{j=1}^{J} \lambda_j r_j(x)\right) && (3.97)\\
&= -\lambda_0 - \sum_{j=1}^{J} \lambda_j \, \mathsf{E}_q[r_j(X)] && (3.98)\\
&= -\lambda_0 - \sum_{j=1}^{J} \lambda_j \alpha_j \qquad (q(\cdot) \text{ satisfies (3.90)}) && (3.99)\\
&= -\lambda_0 \sum_{x \in \mathcal{X}} p^*(x) - \sum_{j=1}^{J} \lambda_j \, \mathsf{E}_{p^*}[r_j(X)] \quad (p^*(\cdot) \text{ satisfies (3.90)}) && (3.100)\\
&= -\sum_{x \in \mathcal{X}} p^*(x) \left(\lambda_0 + \sum_{j=1}^{J} \lambda_j r_j(x)\right) && (3.101)\\
&= -\sum_{x \in \mathcal{X}} p^*(x) \ln p^*(x) && (3.102)\\
&= \mathrm{H}_{p^*}(X). && (3.103)
\end{aligned}$$

Hence for any $q(\cdot)$ we have

$$\mathrm{H}_q(X) \leq \mathrm{H}_{p^*}(X), \tag{3.104}$$

i.e., $p^*(\cdot)$ maximizes entropy! □

As an example of how Theorem 3.13 could be used consider the example of a nonnegative integer-valued RV with some given mean α.

Corollary 3.14. Let X be a nonnegative integer-valued RV with mean α: $X \in \mathbb{N}_0$, $\mathsf{E}[X] = \alpha$. Then

$$\mathrm{H}(X) \le (\alpha+1)\log(\alpha+1) - \alpha\log\alpha. \tag{3.105}$$

Proof: Given $X \in \mathbb{N}_0$ and $\mathsf{E}[X] = \alpha$, we know from Theorem 3.13 that $p^*(x) = e^{\lambda_0+\lambda_1 x} = c\,e^{\lambda x}$ maximizes the entropy. We need to compute the constants c and λ. From

$$\sum_{x=0}^{\infty} c\,e^{\lambda x} \overset{!}{=} 1 \tag{3.106}$$

we get

$$c = \frac{1}{\sum_{x=0}^{\infty} e^{\lambda x}} = \frac{1}{\frac{1}{1-e^{\lambda}}} = 1 - e^{\lambda}. \tag{3.107}$$

From

$$\begin{aligned}
\mathsf{E}[X] &= \sum_{x=0}^{\infty} (1-e^{\lambda})\, e^{\lambda x} x && (3.108)\\
&= \sum_{x=0}^{\infty} (1-e^{\lambda}) \frac{\partial}{\partial\lambda} e^{\lambda x} && (3.109)\\
&= (1-e^{\lambda}) \frac{\partial}{\partial\lambda} \sum_{x=0}^{\infty} e^{\lambda x} && (3.110)\\
&= (1-e^{\lambda}) \frac{\partial}{\partial\lambda} \frac{1}{1-e^{\lambda}} && (3.111)\\
&= (1-e^{\lambda}) \frac{e^{\lambda}}{(1-e^{\lambda})^2} && (3.112)\\
&= \frac{e^{\lambda}}{1-e^{\lambda}} \overset{!}{=} \alpha && (3.113)
\end{aligned}$$

we get

$$\lambda = \ln\left(\frac{\alpha}{\alpha+1}\right). \tag{3.114}$$

Hence,

$$p^*(x) = \frac{1}{\alpha+1}\left(\frac{\alpha}{\alpha+1}\right)^x \tag{3.115}$$

and therefore

$$\begin{aligned}
\mathrm{H}_{p^*}(X) &= \mathsf{E}\left[-\log\left(\frac{1}{\alpha+1}\left(\frac{\alpha}{\alpha+1}\right)^X\right)\right] && (3.116)\\
&= \log(\alpha+1) - \mathsf{E}[X]\log\left(\frac{\alpha}{\alpha+1}\right) && (3.117)\\
&= \log(\alpha+1) - \alpha\log\left(\frac{\alpha}{\alpha+1}\right) && (3.118)\\
&= (\alpha+1)\log(\alpha+1) - \alpha\log\alpha. && (3.119)
\end{aligned}$$

□

Chapter 4

Data Compression: Efficient Coding of a Single Random Message

After having introduced the basic definitions of information theory, we now start with the first important practical problem: how to represent information in an efficient manner. We start with the most simple situation of a single random message, and then extend the setup to more practical cases of an information source without and finally with memory. We will see that some of the basic definitions of Chapter 1 are closely related to this fundamental problem of data compression.

4.1 A Motivating Example

You would like to set up your own phone system that connects you to your three best friends. The question is how to design efficient binary phone numbers. In Table 4.1 you find six different ways how you could choose them.

Note that in this example the phone number is a *codeword* for the person

Table 4.1: Binary phone numbers for a telephone system with three friends.

Friend		Alice	Bob	Carol
Probability		$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$
Phone numbers	(i)	0011	0011	1100
	(ii)	001101	001110	110000
	(iii)	0	1	10
	(iv)	00	11	10
	(v)	0	11	10
	(vi)	10	0	11

you want to talk to. The set of all phone numbers is called *code*. We also assume that you have different probabilities when calling your friends: Bob is your best friend whom you will call in 50% of the times. Alice and Carol are contacted with a frequency of 25% each.

So let us discuss the different designs in Table 4.1:

(i) In this design Alice and Bob have the same phone number. The system obviously will not be able to connect properly. We say that this code is *singular*.

(ii) This is much better, i.e., the code will actually work. However, the phone numbers are quite long and therefore the design is rather inefficient.

(iii) This code is much shorter and at the same time does not use the same codeword twice. Still, a closer look reveals that the system will not work: If you dial 10 this could mean "Carol" or also "Bob, Alice". Or in other words: The telephone system will never connect you to Carol, because once you dial 1, it will immediately connect you to Bob. So, even though the code is nonsingular, it is not *uniquely decodable* (see below for an exact definition).

(iv) This is the first quite efficient code that is functional. But we note something: When calling Alice, why do we have to dial two zeros? After the first zero it is already clear to whom we would like to be connected! We fix this in design (v).

(v) This is still uniquely decodable and obviously more efficient than (iv). Is it the most efficient code? No! Since Bob is called most often, he should be assigned the shortest codeword.

(vi) This is the optimal code. Note one interesting property: Even though the numbers do not all have the same length, once you finish dialing any of the three numbers, the system immediately knows that you have finished dialing. This is because no codeword is the *prefix*[1] of any other codeword, i.e., it never happens that the first few digits of one codeword are identical to another codeword. Such a code is called *prefix-free* (see Section 4.3 below). Note that (iii) was not prefix-free: 1 is a prefix of 10.

From this example we learn the following requirements that we impose on our code design:

- A code needs to be *nonsingular* and *uniquely decodable*, where we define the latter as follows.

[1] According to the Oxford English Dictionary, a *prefix* is a word, letter, or number placed before another.

Definition 4.1. A code is called *uniquely decodable* if a string of any finite number of arbitrarily concatenated codewords can be split up into constituent codewords in only one unique way.

We will come back to this issue of decodability in Section 4.10.

- A code should be short, i.e., we want to *minimize the average codeword length* $\mathsf{E}[L]$, which is defined as follows:

$$\mathsf{E}[L] \triangleq \sum_{i=1}^{r} p_i l_i. \tag{4.1}$$

Here p_i denotes the probability that the source emits the ith symbol, i.e., the probability that the ith codeword $\mathbf{c}_i$ is selected; l_i is the length of the ith codeword; and r is the number of codewords.

- We additionally require the code to be *prefix-free*. Note that this requirement is not necessary, but only convenient. However, we will later see that we lose nothing by asking for it.

4.2 A Coding Scheme

As we have seen in the motivating example, this chapter is concerned with efficient ways of representing data. To say it in an engineering way: we want to "compress data without losing any information".

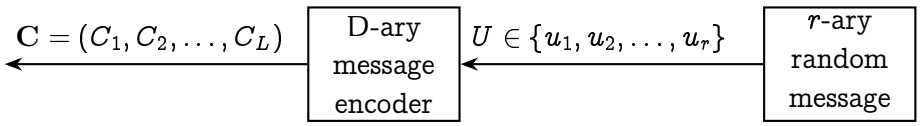


Figure 4.2: Basic data compression system for a single random message U.

To simplify the problem we will consider a rather abstract setup as shown in Figure 4.2. There, we have the following building blocks:

U: random message (random variable), taking value in an r-ary alphabet $\{u_1, u_2, \ldots, u_r\}$, with probability mass function (PMF) $P_U(\cdot)$:

$$P_U(u_i) = p_i, \quad i = 1, \ldots, r. \tag{4.2}$$

C_ℓ: codeword letter, taking value in the D-ary alphabet $\{0, 1, \ldots, \mathrm{D}-1\}$.

$\mathbf{C}$: codeword, taking a different value depending on the value of the message U. Actually, the codeword is a deterministic function of U, i.e., it is only random because U is random.

L: codeword length (length of $\mathbf{C}$). It is random because U is random and because the different codewords do not necessarily all have the same length.

Remark 4.2. We note the following:

- For every message u_i we have a codeword $\mathbf{c}_i$ of length l_i. Hence, $\mathbf{c}_i$ is a representation of u_i in the sense that we want to be able to recover u_i from $\mathbf{c}_i$.
- The complete set of codewords is called a *code* for U.
- For a concrete system to work, not only the code needs to be known, but also the assignment rule that maps the message to a codeword. We call the complete system *coding scheme*.
- The quality of a coding scheme is its average length $\mathsf{E}[L]$: the shorter, the better! (Remember, we are trying to compress the data!)
- Since the code letters are D-ary, the code is called a D-*ary code*. △

To summarize:

> The codeword $\mathbf{C}$ is a one-to-one representation of the random message U. We try to minimize the expected length of $\mathbf{C}$, i.e., we "compress" the data U.

Remark 4.3. In principle it is no problem if some message u_i has a zero probability. Obviously it should be assigned a long codeword as u_i will (with probability 1) not occur. However, when assigning a codeword for zero-probability messages, we will mess up a fair comparison between different codes. For example, if we use a provably optimal code, but then increase the codeword length of the codeword for the message that never occurs by a factor 100. The expected codeword length still is the same, so this new code performs equally well. So it is also optimal, even though it contains one codeword that is tremendously longer than in the original code. So, shall we really consider these two codes to be the same?

To avoid such issues, in the following we will always assume that all messages have nonzero probability (or, equivalently, we will remove from the message alphabet those symbols that have zero probability). △

4.3 Prefix-Free or Instantaneous Codes

Consider the following code with four codewords:

$$\begin{aligned} \mathbf{c}_1 &= 0 \\ \mathbf{c}_2 &= 10 \\ \mathbf{c}_3 &= 110 \\ \mathbf{c}_4 &= 111 \end{aligned} \tag{4.3}$$

Note that the zero serves as a kind of "comma": Whenever we receive a zero (or the code has reached length 3), we know that the codeword has finished. However, this comma still contains useful information about the message as there is still one codeword without it! This is another example of a prefix-free code. We recall the following definition.

Definition 4.4. A code is called *prefix-free* (or sometimes also *instantaneous*) if no codeword is the prefix of another codeword.

The name *instantaneous* is motivated by the fact that for a prefix-free code we can decode instantaneously once we have received a codeword and do not need to wait for later until the decoding becomes unique. Unfortunately, in literature one also finds people calling a prefix-free code a *prefix code*. This name is quite confusing because rather than having prefixes it is the point of the code to have *no* prefix! We shall stick to the name *prefix-free codes*.

Consider next the following example:

$$\begin{aligned} \mathbf{c}_1 &= 0 \\ \mathbf{c}_2 &= 01 \\ \mathbf{c}_3 &= 011 \\ \mathbf{c}_4 &= 111 \end{aligned} \tag{4.4}$$

This code is not prefix-free (0 is a prefix of 01 and 011; 01 is a prefix of 011), but it is still uniquely decodable.

Exercise 4.5. *Given the code in* (4.4)*, decode the sequence* 0011011110. ◊

Note the drawback of the code design in (4.4): The receiver needs to wait and see how the sequence continues before it can make a unique decision about the decoding. The code is *not instantaneously* decodable.

Apart from the fact that they can be decoded instantaneously, another nice property of prefix-free codes is that they can very easily be represented by leaves of *decision trees*. To understand this we will next make a small detour and talk about trees and their relation to codes.

4.4 Trees and Codes

The following definition is quite straightforward.

Definition 4.6. A D-*ary tree* consists of a root with some branches, nodes, and leaves in such a way that the root and every node have exactly D children.

As example, in Figure 4.3 a binary tree is shown, with two branches stemming forward from every node.

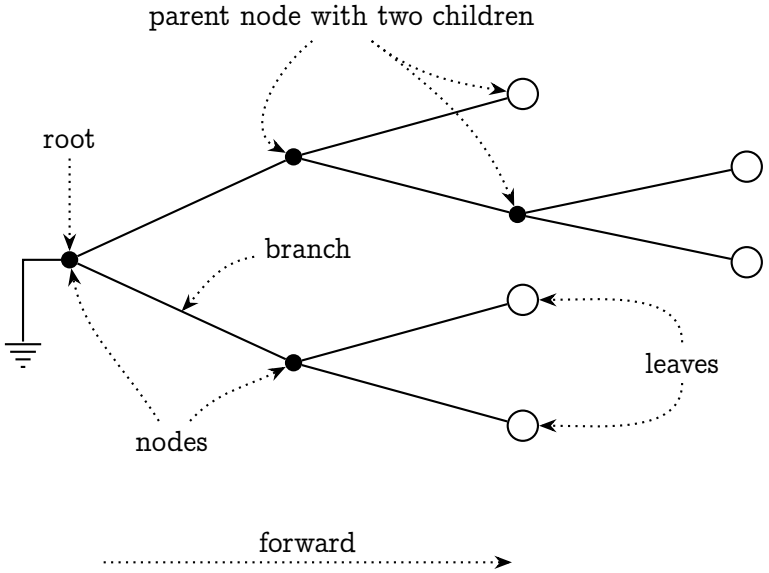


Figure 4.3: A binary tree with four *nodes* (including the *root*—the node that is grounded), and five *leaves*. Note that we always clearly distinguish between *nodes* and *leaves*: A node always has children, while a leaf always is an "end-point" in the tree.

The clue to this section is to note that *any* D-ary code can be represented as a D-ary tree. The D branches stemming from each node stand for the D different possible symbols in the code's alphabet, so that when we walk along the tree starting from the root, each symbol in a codeword is regarded as a decision which of the D branches to take. Hence, every codeword can be represented by a particular path traversing through the tree.

As an example, Figure 4.4 shows the binary (i.e., $\mathrm{D} = 2$) tree of a binary code with five codewords, and Figure 4.5 shows a ternary ($\mathrm{D} = 3$) tree with six codewords. Note that we need to keep branches that are not used in order to make sure that the tree is D-ary.

In Figure 4.6, we show the tree describing the prefix-free code given in (4.3). Note that here every codeword is a leaf. This is no accident.

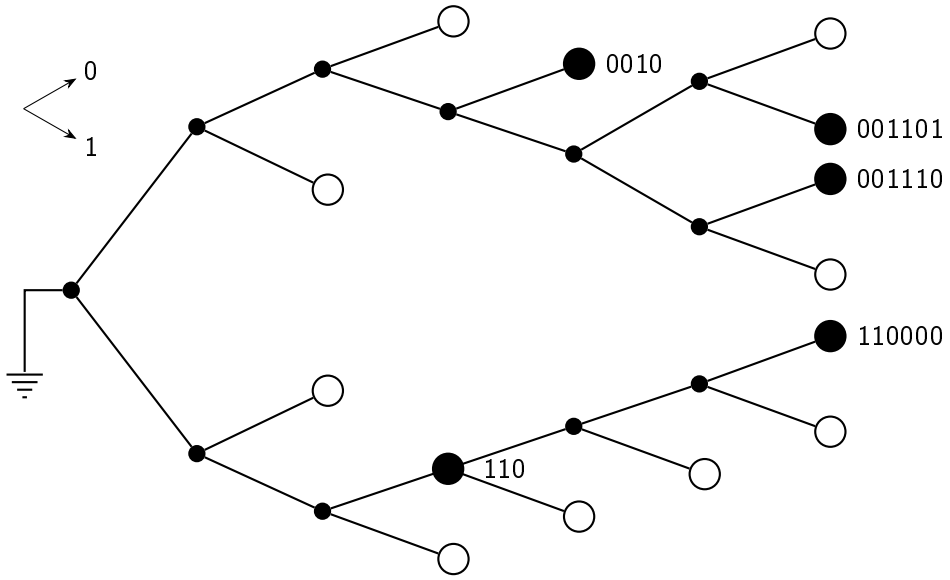


Figure 4.4: An example of a binary tree with five codewords: 110, 0010, 001101, 001110, and 110000. At every node, going upwards corresponds to a 0, and going downwards corresponds to a 1. The node with the ground symbol is the root of the tree indicating the starting point.

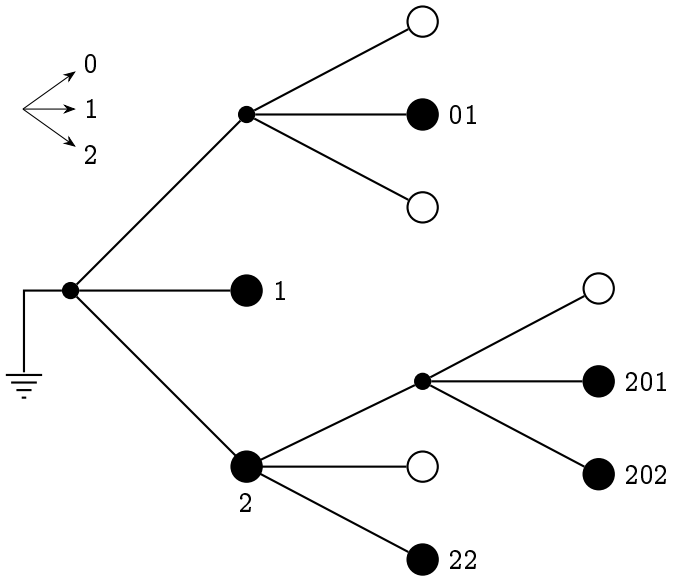


Figure 4.5: An example of a ternary tree with six codewords: 1, 2, 01, 22, 201, and 202. At every node, going upwards corresponds to a 0, taking the middle branch corresponds to a 1, and going downwards corresponds to a 2.

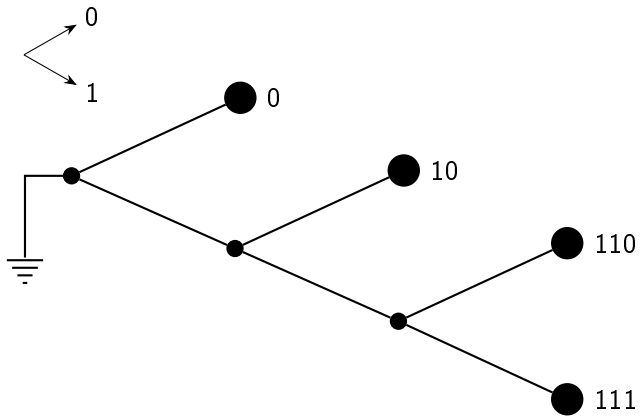


Figure 4.6: Decision tree corresponding to the prefix-free code given in (4.3).

Lemma 4.7. A D-ary code $\{\mathbf{c}_1, \ldots, \mathbf{c}_r\}$ is prefix-free if, and only if, in its D-ary tree every codeword is a leaf. (But not every leaf necessarily is a codeword.)

Exercise 4.8. *Prove Lemma 4.7.*

Hint: Carefully think about the definition of prefix-free codes (Definition 4.4). ◊

As mentioned, the D-ary tree of a prefix-free code might contain leaves that are not codewords. Such leaves are called *unused leaves*. Some more examples of trees of prefix-free and non-prefix-free codes are shown in Figure 4.7.

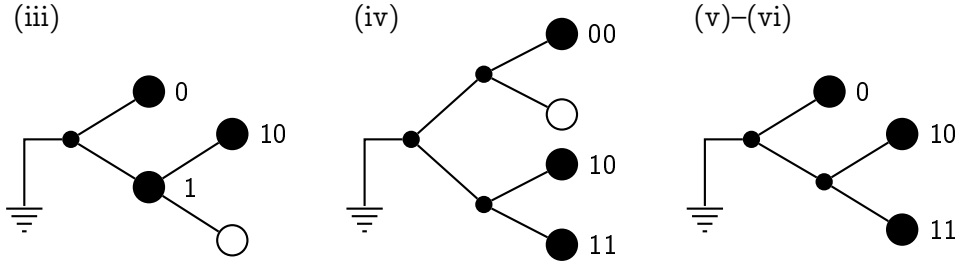


Figure 4.7: Examples of codes and its trees. The examples are taken from Table 4.1.

An important concept of trees is the *depths* of their leaves.

Definition 4.9. The *depth of a leaf* in a D-ary tree is the number of steps it takes when walking from the root forward to the leaf.

As an example consider again Figure 4.7. Tree (iv) has four leaves, all of them at depth 2. Both tree (iii) and tree (v)-(vi) have three leaves, one at depth 1 and two at depth 2.

We will now derive some interesting properties of trees. Since codes can be represented by trees, we will then be able to apply these properties directly to codes.

Lemma 4.10 (Leaf Counting and Leaf Depth Lemma). The number of leaves n and their depths $l_1, l_2, \ldots, l_n$ in a D-ary tree satisfy:

$$n = 1 + \mathrm{N}(\mathrm{D} - 1), \tag{4.5}$$

$$\sum_{i=1}^{n} \mathrm{D}^{-l_i} = 1, \tag{4.6}$$

where N is the number of nodes (including the root).

Proof: By *extending a leaf* we mean changing a leaf into a node by adding D branches that stem forward. In that process

- we reduce the number of leaves by 1,
- we increase the number of nodes by 1, and
- we increase the number of leaves by D,

i.e., in total we gain 1 node and $\mathrm{D} - 1$ leaves. This process is depicted graphically in Figure 4.8 for the case of $\mathrm{D} = 3$.

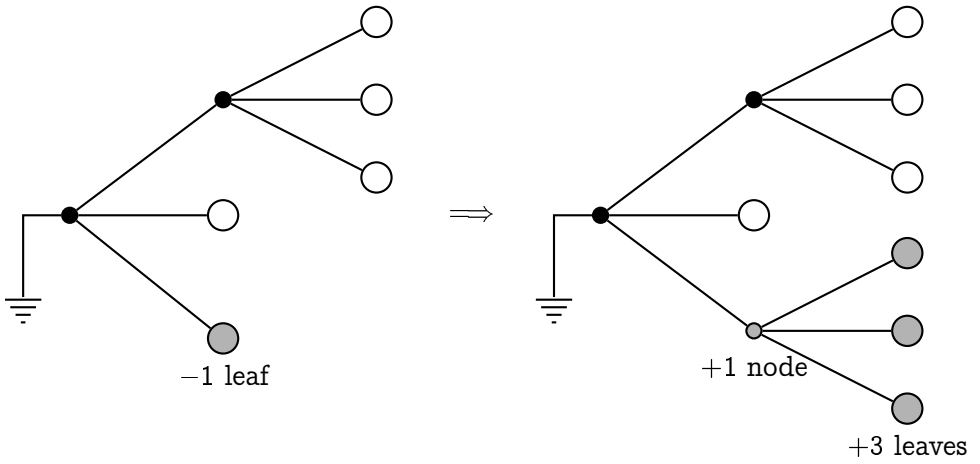


Figure 4.8: Extending a leaf in a ternary tree ($\mathrm{D} = 3$): The total number of nodes is increased by one and the total number of leaves is increased by $\mathrm{D} - 1 = 2$.

To prove the first statement (4.5), we start with the extended root, i.e., at the beginning, we have the root and $n = \mathrm{D}$ leaves. In this case, we have $\mathrm{N} = 1$ and (4.5) is satisfied. Now we can grow any tree by continuously extending some leaf, every time increasing the number of leaves by $\mathrm{D}-1$ and the number of nodes by 1. We see that (4.5) remains valid. By induction this proves the first statement.

We will prove the second statement (4.6) also by induction. We again start with the extended root.

1. An extended root has D leaves, all at depth 1: $l_i = 1$. Hence,
$$\sum_{i=1}^{n} \mathrm{D}^{-l_i} = \sum_{i=1}^{\mathrm{D}} \mathrm{D}^{-1} = \mathrm{D} \cdot \mathrm{D}^{-1} = 1, \tag{4.7}$$
i.e., for the extended root (4.6) is satisfied.

2. Suppose $\sum_{i=1}^{n} \mathrm{D}^{-l_i} = 1$ holds for an arbitrary D-ary tree with n leaves. Now we extend one leaf, say the nth leaf.[2] We get a new tree with $n' = n - 1 + \mathrm{D}$ leaves where the D new leaves all have depths $l_n + 1$:
$$\sum_{i=1}^{n'} \mathrm{D}^{-l'_i} = \underbrace{\sum_{i=1}^{n-1} \mathrm{D}^{-l_i}}_{\substack{\text{unchanged}\\ \text{leaves}}} + \mathrm{D} \cdot \underbrace{\mathrm{D}^{-(l_n+1)}}_{\substack{\text{new leaves}\\ \text{at depth}\\ l_n+1}} \tag{4.8}$$
$$= \sum_{i=1}^{n-1} \mathrm{D}^{-l_i} + \mathrm{D}^{-l_n} \tag{4.9}$$
$$= \sum_{i=1}^{n} \mathrm{D}^{-l_i} = 1. \tag{4.10}$$
Here the last equality follows from our assumption that $\sum_{i=1}^{n} \mathrm{D}^{-l_i} = 1$. Hence by extending one leaf the second statement continues to hold.

3. Since any tree can be grown by continuously extending some leaf, the proof follows by induction. □

We are now ready to apply our first insights about trees to codes.

4.5 Kraft Inequality

The following theorem is very useful because it gives us a way to find out whether a prefix-free code exists or not.

[2] Since the tree is arbitrary, it does not matter how we number the leaves.

Theorem 4.11 (Kraft Inequality).
There exists a D-ary prefix-free code with r codewords of lengths $l_1, l_2, \ldots, l_r \in \mathbb{N}$ if, and only if,

$$\sum_{i=1}^{r} \mathrm{D}^{-l_i} \le 1. \tag{4.11}$$

If (4.11) is satisfied with equality, then there are no unused leaves in the tree.

Example 4.12. Let $l_1 = 3$, $l_2 = 4$, $l_3 = 4$, $l_4 = 4$, $l_5 = 4$, and consider a binary code $\mathrm{D} = 2$. Then

$$2^{-3} + 4 \cdot 2^{-4} = \frac{1}{8} + \frac{4}{16} = \frac{3}{8} \le 1, \tag{4.12}$$

i.e., there exists a binary prefix-free code with the above codeword lengths.

On the other hand, we cannot find any binary prefix-free code with five codewords of lengths $l_1 = 1$, $l_2 = 2$, $l_3 = 3$, $l_4 = 3$, and $l_5 = 4$ because

$$2^{-1} + 2^{-2} + 2 \cdot 2^{-3} + 2^{-4} = \frac{17}{16} > 1. \tag{4.13}$$

But, if we instead look for a ternary prefix-free code ($\mathrm{D} = 3$) with five codewords of these lengths, we will be successful because

$$3^{-1} + 3^{-2} + 2 \cdot 3^{-3} + 3^{-4} = \frac{43}{81} \le 1. \tag{4.14}$$

These examples are shown graphically in Figure 4.9. ◊

Proof of the Kraft Inequality: We prove the two directions separately:

$\Longrightarrow$: Suppose that there exists a D-ary prefix-free code with the given codeword lengths. From Lemma 4.7 we know that all r codewords of a D-ary prefix-free code are leaves in a D-ary tree. The total number n of (used and unused) leaves in this tree can therefore not be smaller than r,

$$r \le n. \tag{4.15}$$

Hence,

$$\sum_{i=1}^{r} \mathrm{D}^{-l_i} \le \sum_{i=1}^{n} \mathrm{D}^{-l_i} = 1, \tag{4.16}$$

where the last equality holds because of the Leaf Depth Lemma (Lemma 4.10).

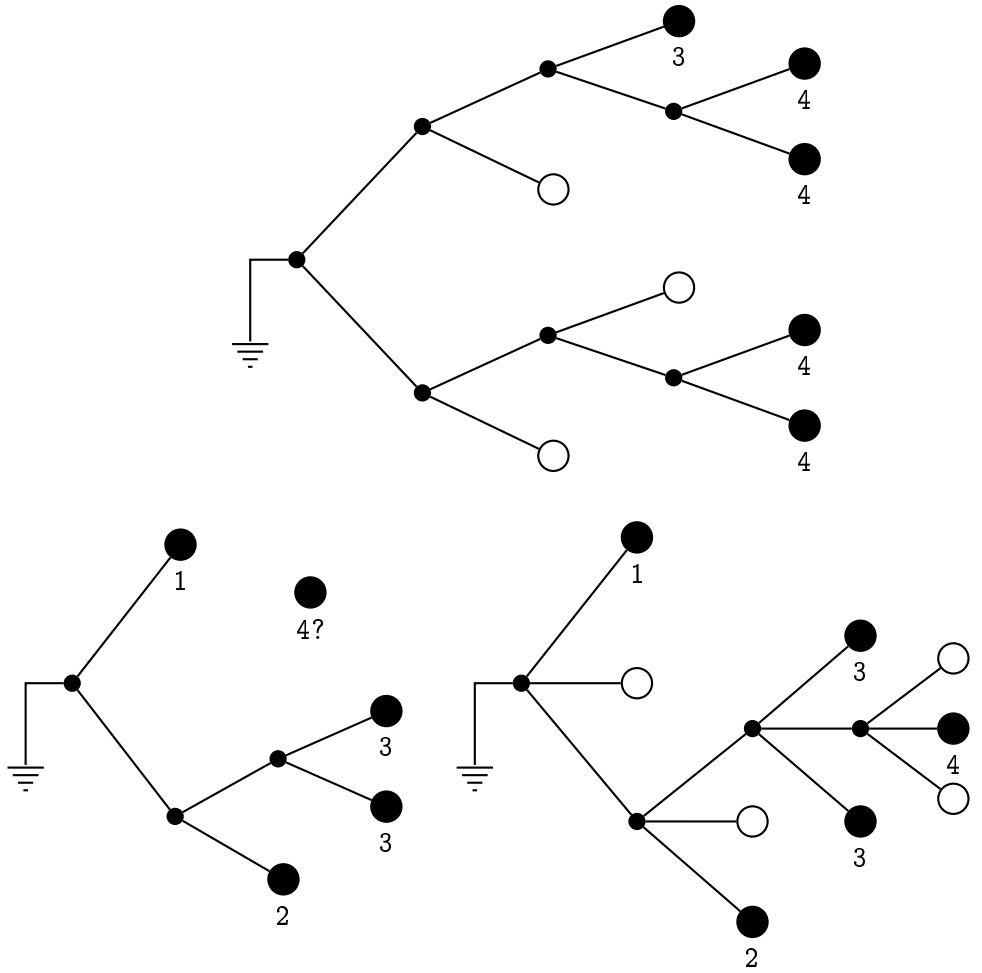


Figure 4.9: Illustration of the Kraft Inequality according to Example 4.12.

$\Longleftarrow$: Suppose that $\sum_{i=1}^{r} \mathrm{D}^{-l_i} \le 1$. We now can construct a D-ary prefix-free code as follows:

Step 1: Start with the extended root, i.e., a tree with D leaves, set $i = 1$, and assume without loss of generality that $l_1 \le l_2 \le \cdots \le l_r$.

Step 2: If there is an unused leaf at depth l_i, put the ith codeword there. Note that there could be none because l_i can be strictly larger than the current depth of the tree. In this case, extend any unused leaf to depth l_i, and put the ith codeword to one of the new leaves.

Step 3: If $i = r$, stop. Otherwise $i \to i + 1$ and go to Step 2.

We only need to check that Step 2 is always possible, i.e., that there is always some unused leaf available. To this end, note that if we get to

Step 2, we have already put $i-1$ codewords into the tree. From the Leaf Depth Lemma (Lemma 4.10) we know that

$$1 = \sum_{j=1}^{n} \mathrm{D}^{-\tilde{l}_j} = \underbrace{\sum_{j=1}^{i-1} \mathrm{D}^{-l_j}}_{\text{used leaves}} + \underbrace{\sum_{j=i}^{n} \mathrm{D}^{-\tilde{l}_j}}_{\text{unused leaves}}, \tag{4.17}$$

where $\tilde{l}_j$ are the depths of the leaves in the tree at that moment, i.e., $(\tilde{l}_1, \ldots, \tilde{l}_{i-1}) = (l_1, \ldots, l_{i-1})$ and $\tilde{l}_i, \ldots, \tilde{l}_n$ are the depths of the (so far) unused leaves. Now note that by assumption $i \le r$, i.e.,

$$\sum_{j=1}^{i-1} \mathrm{D}^{-l_j} < \sum_{j=1}^{r} \mathrm{D}^{-l_j} \le 1, \tag{4.18}$$

where the last inequality follows by assumption. Hence, in (4.17), the term "used leaves" is strictly less than 1, and thus the term "unused leaves" must be strictly larger than 0, i.e., there still must be some unused leaves available! □

Remark 4.13. The beauty of the above proof is that it is *constructive*, giving us an algorithm how to build a corresponding code tree for given codeword lengths. Another, nonconstructive proof of the Kraft Inequality is as follows: Let $l_{\max}$ be the length of the longest codeword and consider the full D-ary tree to depth $l_{\max}$. This tree contains $\mathrm{D}^{l_{\max}}$ leaves. Any codeword at depth l_i blocks $\mathrm{D}^{l_{\max}-l_i}$ of these leaves because in a prefix-free code no codeword can have other codewords further down in the tree. Thus,

$$\sum_{i=1}^{r} \mathrm{D}^{l_{\max}-l_i} \le \mathrm{D}^{l_{\max}}. \tag{4.19}$$

The Kraft Inequality now follows directly by dividing both sides by $\mathrm{D}^{l_{\max}}$. △

4.6 Trees with Probabilities

We have seen already in Section 4.1 that for codes it is important to consider the probabilities of the codewords. We therefore now introduce probabilities in our trees.

Definition 4.14. A *tree with probabilities* is a finite tree with probabilities assigned to each node and leaf such that

- the probability of a node is the sum of the probabilities of its children, and
- the root has probability 1.

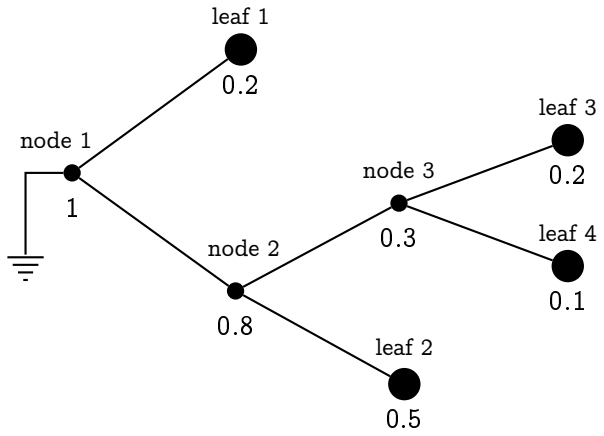


Figure 4.10: An example of a binary tree with probabilities.

An example of a binary tree with probabilities is given in Figure 4.10. Note that the node probabilities can be seen as the overall probability of passing through this node when making a random walk from the root to a leaf. For the example of Figure 4.10, we have an 80% chance that our path will go through node 2, and with a chance of 30% we will pass through node 3. By definition, the probability of ending up in a certain leaf is given by the leaf probability, e.g., in Figure 4.10, we have a 10% chance to end up in leaf 4. Obviously, since we always start our random walk at the root, the probability that our path goes through the root is always 1.

To clarify our notation we will use the following conventions:

- n denotes the total number of leaves;
- p_i, $i = 1, \ldots, n$, denote the probabilities of the leaves;
- N denotes the number of nodes (including the root, but excluding the leaves); and
- P_ℓ, $\ell = 1, \ldots, \mathrm{N}$, denote the probabilities of the nodes, where by definition $\mathrm{P}_1 = 1$ is the root probability.

Moreover, we will use $q_{\ell,j}$ to denote the probability of the jth node/leaf that is one step forward from node ℓ (the jth child of node ℓ), where $j = 0, 1, \ldots, \mathrm{D}-1$. I.e., we have

$$\sum_{j=0}^{\mathrm{D}-1} q_{\ell,j} = \mathrm{P}_\ell. \tag{4.20}$$

Since in a prefix-free code all codewords are leaves and we are particularly interested in the average codeword length, we are very much interested in the

average depth of the leaves in a tree (where for the averaging operation we use the probabilities in the tree). Luckily, there is an elegant way to compute this average depth as shown in the following lemma.

Lemma 4.15 (Path Length Lemma). In a rooted tree with probabilities, the average depth $\mathsf{E}[L]$ of the leaves is equal to the sum of the probabilities of all nodes (including the root):

$$\mathsf{E}[L] = \sum_{\ell=1}^{\mathsf{N}} \mathsf{P}_\ell. \tag{4.21}$$

Example 4.16. Consider the tree of Figure 4.10. We have four leaves: one at depth $l_1 = 1$ with a probability $p_1 = 0.2$, one at depth $l_2 = 2$ with a probability $p_2 = 0.5$, and two at depth $l_3 = l_4 = 3$ with probabilities $p_3 = 0.2$ and $p_4 = 0.1$, respectively. Hence, the average depth of the leaves is

$$\mathsf{E}[L] = \sum_{i=1}^{4} p_i l_i = 0.2 \cdot 1 + 0.5 \cdot 2 + 0.2 \cdot 3 + 0.1 \cdot 3 = 2.1. \tag{4.22}$$

According to Lemma 4.15 this must be equal to the sum of the node probabilities:

$$\mathsf{E}[L] = \mathsf{P}_1 + \mathsf{P}_2 + \mathsf{P}_3 = 1 + 0.8 + 0.3 = 2.1. \tag{4.23}$$

$\Diamond$

Proof of Lemma 4.15: The lemma is easiest understood when looking at a particular example. Let us again consider the tree of Figure 4.10. When computing the average depth of the leaves, we sum all probabilities of the leaves weighted with the corresponding depth. So, e.g., the probability $p_1 = 0.2$ of leaf 1 — being at depth 1 — has weight 1, i.e., it needs to be counted once only. If we now look at the sum of the node probabilities, then we note that p_1 indeed is only counted once as it only is part of the probability of the root $\mathsf{P}_1 = 1$.

Leaf 2, on the other hand, is at depth 2 and therefore its probability $p_2 = 0.5$ has weight 2, or in other words, p_2 needs to be counted twice. Again, in the sum of the node probabilities, this is indeed the case because p_2 is contained both in the root probability $\mathsf{P}_1 = 1$ and also in the probability of the second node $\mathsf{P}_2 = 0.8$.

Finally, the probabilities of leaf 3 and leaf 4, $p_3 = 0.2$ and $p_4 = 0.1$, are counted three times as they are part of P_1, P_2, and P_3:

$$\mathsf{E}[L] = 2.1 \tag{4.24}$$

$$= 1 \cdot 0.2 + 2 \cdot 0.5 + 3 \cdot 0.2 + 3 \cdot 0.1 \tag{4.25}$$
$$= 1 \cdot (0.2 + 0.5 + 0.2 + 0.1) + 1 \cdot (0.5 + 0.2 + 0.1) + 1 \cdot (0.2 + 0.1) \tag{4.26}$$
$$= 1 \cdot \mathrm{P}_1 + 1 \cdot \mathrm{P}_2 + 1 \cdot \mathrm{P}_3 \tag{4.27}$$
$$= \mathrm{P}_1 + \mathrm{P}_2 + \mathrm{P}_3. \tag{4.28}$$

□

Since we have started talking about probabilities and since this is an information theory course, it should not come as a big surprise that we next introduce the concept of entropy in our trees.

Definition 4.17. The *leaf entropy* is defined as

$$\mathrm{H}_{\text{leaf}} \triangleq -\sum_{i=1}^{n} p_i \log p_i. \tag{4.29}$$

Definition 4.18. The *branching entropy* H_ℓ of node ℓ is defined as

$$\mathrm{H}_\ell \triangleq -\sum_{j=0}^{\mathrm{D}-1} \frac{q_{\ell,j}}{\mathrm{P}_\ell} \log \frac{q_{\ell,j}}{\mathrm{P}_\ell}. \tag{4.30}$$

Note that $\frac{q_{\ell,j}}{\mathrm{P}_\ell}$ is the conditional probability of going along the jth branch given that we are at node ℓ (normalization!). Further note that, as usual, we implicitly exclude zero probability values. In particular, we do not consider the situation where a node has zero probability, but assume that in this case this node with all its children (and their children, etc.) is removed from the tree and replaced by an unused leaf.

So far we do not see yet what the use of these definitions will be, but we compute an example anyway.

Example 4.19. Consider the tree in Figure 4.11. We have

$$\mathrm{H}_{\text{leaf}} = -0.4 \log 0.4 - 0.1 \log 0.1 - 0.5 \log 0.5 \approx 1.361 \text{ bits}; \tag{4.31}$$
$$\mathrm{H}_1 = -\frac{0.4}{1} \log \frac{0.4}{1} - \frac{0.6}{1} \log \frac{0.6}{1} \approx 0.971 \text{ bits}; \tag{4.32}$$
$$\mathrm{H}_2 = -\frac{0.1}{0.6} \log \frac{0.1}{0.6} - \frac{0.5}{0.6} \log \frac{0.5}{0.6} \approx 0.650 \text{ bits}; \tag{4.33}$$
$$\mathrm{H}_3 = -\frac{0.1}{0.1} \log \frac{0.1}{0.1} = 0 \text{ bits}. \tag{4.34}$$

◊

We will next prove a very interesting relationship between the leaf entropy and the branching entropy that will turn out to be fundamental for the understanding of codes.

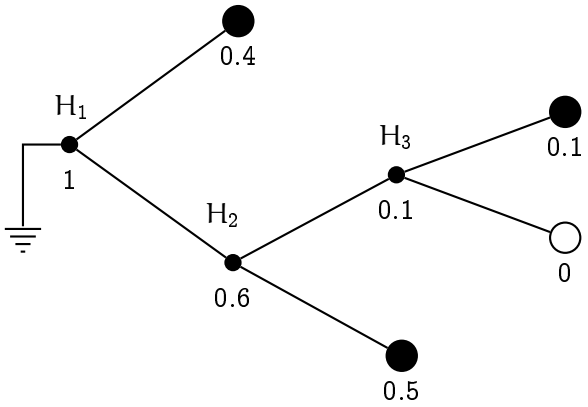


Figure 4.11: A binary tree with probabilities.

Theorem 4.20 (Leaf Entropy Theorem).
In any tree with probabilities it holds that

$$\mathrm{H}_{\text{leaf}} = \sum_{\ell=1}^{\mathrm{N}} \mathrm{P}_\ell \mathrm{H}_\ell. \tag{4.35}$$

Proof: First, we recall that by definition of trees and trees with probabilities we have for every node ℓ:

$$\mathrm{P}_\ell = \sum_{j=0}^{\mathrm{D}-1} q_{\ell,j} \tag{4.36}$$

(see (4.20)). Using the definition of branching entropy, we get

$$\begin{aligned}
\mathrm{P}_\ell \mathrm{H}_\ell &= \mathrm{P}_\ell \cdot \left(- \sum_{j=0}^{\mathrm{D}-1} \frac{q_{\ell,j}}{\mathrm{P}_\ell} \log \frac{q_{\ell,j}}{\mathrm{P}_\ell} \right) && (4.37)\\
&= - \sum_{j=0}^{\mathrm{D}-1} q_{\ell,j} \log \frac{q_{\ell,j}}{\mathrm{P}_\ell} && (4.38)\\
&= - \sum_{j=0}^{\mathrm{D}-1} q_{\ell,j} \log q_{\ell,j} + \sum_{j=0}^{\mathrm{D}-1} q_{\ell,j} \underbrace{\log \mathrm{P}_\ell}_{\substack{\text{indep.}\\ \text{of } j}} && (4.39)\\
&= - \sum_{j=0}^{\mathrm{D}-1} q_{\ell,j} \log q_{\ell,j} + \log \mathrm{P}_\ell \cdot \underbrace{\sum_{j=0}^{\mathrm{D}-1} q_{\ell,j}}_{=\mathrm{P}_\ell} && (4.40)\\
&= - \sum_{j=0}^{\mathrm{D}-1} q_{\ell,j} \log q_{\ell,j} + \mathrm{P}_\ell \log \mathrm{P}_\ell, && (4.41)
\end{aligned}$$

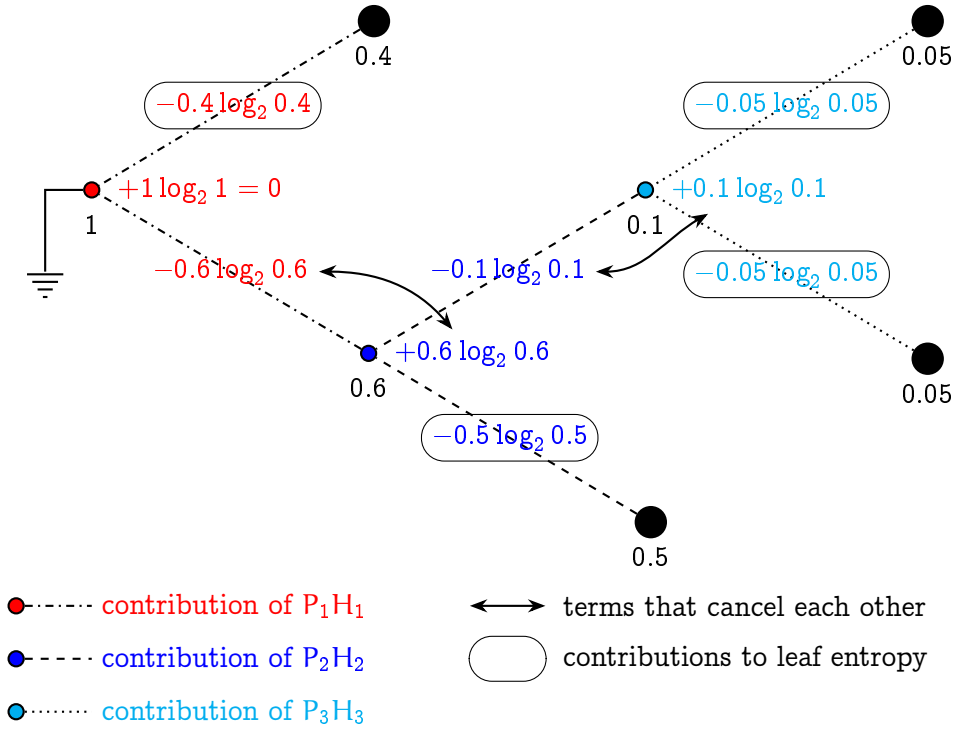


Figure 4.12: Graphical proof of the Leaf Entropy Theorem. In this example there are three nodes. We see that all contributions cancel apart from the root node (whose contribution is 0) and the leaves.

where the last equality follows from (4.36).

Hence, for every node $\tilde{\ell}$, we see that it will contribute to $\sum_{\ell=1}^{N} P_\ell H_\ell$ twice:

- Firstly it will add $P_{\tilde{\ell}} \log P_{\tilde{\ell}}$ when the node counter ℓ passes through $\tilde{\ell}$; and
- secondly it will subtract $q_{\ell,j} \log q_{\ell,j} = P_{\tilde{\ell}} \log P_{\tilde{\ell}}$ when the node counter ℓ points to the parent node of $\tilde{\ell}$.

Hence, the contributions of all nodes will be canceled out — apart from the root that does not have a parent. The root only contributes $P_1 \log P_1$ for $\ell = 1$. However, since $P_1 = 1$, we have $P_1 \log P_1 = 1 \log 1 = 0$. So the root does not contribute either.

It only remains to consider the leaves. Note that the node counter ℓ will not pass through leaves by definition. Hence, a leaf only contributes when the node counter points to its parent node and its contribution is $-q_{\ell,j} \log q_{\ell,j} = -p_i \log p_i$. Since the sum of all $-p_i \log p_i$ equals the leaf entropy by definition, this proves the claim.

In Figure 4.12 we have tried to depict the argumentation of this proof graphically. □

Example 4.21. We continue with Example 4.19:

$$\begin{aligned} P_1H_1 + P_2H_2 + P_3H_3 &= 1 \cdot 0.971 + 0.6 \cdot 0.650 + 0.1 \cdot 0 \text{ bits} && (4.42)\\ &= 1.361 \text{ bits} = H_{\text{leaf}} && (4.43) \end{aligned}$$

as we have expected. ◊

4.7 What We Cannot Do: Fundamental Limitations of Source Coding

The main strength of information theory is that it can provide some fundamental statements about what is possible to achieve and what is not possible. So a typical information theoretic result will consist of an upper bound and a lower bound or — as it is common to call these two parts — an achievability part and a converse part. The achievability part of a theorem tells us what we can do, and the converse part tells us what we cannot do.

Sometimes, the theorem also will tell us how to do it, but usually the result is theoretic in the sense that it only proves what is possible without actually saying how it could be done. To put it pointedly: Information theory tells us what is possible, coding theory tells us how to do it.

In this section we will now derive our first converse part, i.e., we will prove some fundamental limitation about the efficiency of source coding. Let us quickly summarize what we know about codes and their corresponding trees:

- We would like to use prefix-free codes.
- Every prefix-free code can be represented by a tree where every codeword corresponds to a leaf in the tree.
- Every codeword has a certain probability corresponding to the probability of the source symbol it represents.
- Unused leaves can be regarded as symbols that never occur, i.e., we assign probability zero to them.

From these observations we realize that the entropy $H(U)$ of a random message U with probabilities $p_1, \dots, p_r$ and the leaf entropy of the corresponding tree are the same:

$$H_{\text{leaf}} = H(U). \tag{4.44}$$

Note that the unused leaves do not contribute to H_{leaf} since they have zero probability.

Moreover, the average codeword length $E[L]$ is equivalent to the average depth of the leaves. (Again we can ignore the unused leaves because they have probability zero and therefore do not contribute to the average.)

Now note that since we consider D-ary trees where each node branches into D different children, we know from the properties of entropy that the branching entropy can never be larger than $\log \mathrm{D}$ (see Theorem 1.13):

$$\mathrm{H}_\ell \le \log \mathrm{D}. \tag{4.45}$$

Hence, using this together with the Leaf Entropy Theorem (Theorem 4.20) and the Path Length Lemma (Lemma 4.15) we get:

$$\mathrm{H}(U) = \mathrm{H}_{\text{leaf}} = \sum_{\ell=1}^{\mathrm{N}} \mathrm{P}_\ell \mathrm{H}_\ell \le \sum_{\ell=1}^{\mathrm{N}} \mathrm{P}_\ell \cdot \log \mathrm{D} = \log \mathrm{D} \cdot \sum_{\ell=1}^{\mathrm{N}} \mathrm{P}_\ell = \log \mathrm{D} \cdot \mathsf{E}[L] \tag{4.46}$$

and hence

$$\boxed{\mathsf{E}[L] \ge \frac{\mathrm{H}(U)}{\log \mathrm{D}}.} \tag{4.47}$$

Note that the logarithm to compute the entropy and the logarithm to compute $\log \mathrm{D}$ must use the same basis, however, it does not matter which basis is chosen.

This is the *converse part* of the *coding theorem for a single random message*. It says that whatever code you try to design, the average codeword length of *any* D-ary prefix-free code for an r-ary random message U cannot be smaller than the entropy of U (using the correct units)!

Note that to prove this statement we have not designed any code, but instead we have been able to prove something that holds for *every* code that exists.

When do we have equality? From the derivation just above we see that we have equality if the branching entropy always is $\mathrm{H}_\ell = \log \mathrm{D}$, i.e., the branching probabilities are all uniform. This is only possible if p_i is a negative integer power of D for all i:

$$p_i = \mathrm{D}^{-\nu_i} \tag{4.48}$$

with $\nu_i \in \mathbb{N}$ a natural number (and, of course, if we design an optimal code).

In Appendix 4.A we show an alternative proof of (4.47) that uses the fact that relative entropy is nonnegative.

4.8 What We Can Do: Analysis of Some Good Codes

In practice it is not only important to know where the limitations are, but perhaps even more so to know how close we can get to these limitations. So, as a next step, we would like to finally start thinking about how to design codes and then try to analyze their performance.

4.8.1 Shannon-Type Codes

We have already understood that a good code should assign a short codeword to a message with high probability, and use the long codewords for unlikely messages only. So we try to design a simple code following this main design principle.

Assume that $P_U(u_i) = p_i > 0$ for all i since we do not care about messages with zero probability. Then, for every message u_i define

$$l_i \triangleq \left\lceil \frac{\log \frac{1}{p_i}}{\log \mathrm{D}} \right\rceil, \tag{4.49}$$

(where $\lceil \xi \rceil$ denotes the smallest integer not smaller than ξ) and choose an arbitrary unique prefix-free codeword of length l_i. Any code generated like that is called *Shannon-type code*.

We need to show that such a code always exists. Note that by definition in (4.49)

$$l_i \geq \frac{\log \frac{1}{p_i}}{\log \mathrm{D}} = \log_{\mathrm{D}} \frac{1}{p_i}. \tag{4.50}$$

Hence, we have

$$\sum_{i=1}^{r} \mathrm{D}^{-l_i} \leq \sum_{i=1}^{r} \mathrm{D}^{-\log_{\mathrm{D}} \frac{1}{p_i}} = \sum_{i=1}^{r} \mathrm{D}^{\log_{\mathrm{D}} p_i} = \sum_{i=1}^{r} p_i = 1 \tag{4.51}$$

and the Kraft Inequality (Theorem 4.11) is always satisfied. So we know that we can always find a Shannon-type code.

Example 4.22. As an example, consider a random message U with four symbols having probabilities

$$p_1 = 0.4, \quad p_2 = 0.3, \quad p_3 = 0.2, \quad p_4 = 0.1, \tag{4.52}$$

i.e., $\mathrm{H}(U) \approx 1.846$ bits. We design a binary Shannon-type code:

$$l_1 = \left\lceil \log_2 \frac{1}{0.4} \right\rceil = 2, \tag{4.53}$$

$$l_2 = \left\lceil \log_2 \frac{1}{0.3} \right\rceil = 2, \tag{4.54}$$

$$l_3 = \left\lceil \log_2 \frac{1}{0.2} \right\rceil = 3, \tag{4.55}$$

$$l_4 = \left\lceil \log_2 \frac{1}{0.1} \right\rceil = 4. \tag{4.56}$$

Note that

$$2^{-2} + 2^{-2} + 2^{-3} + 2^{-4} = \frac{11}{16} < 1, \tag{4.57}$$

i.e., such a code does exist, as expected. A possible choice for such a code is shown in Figure 4.13. ◊

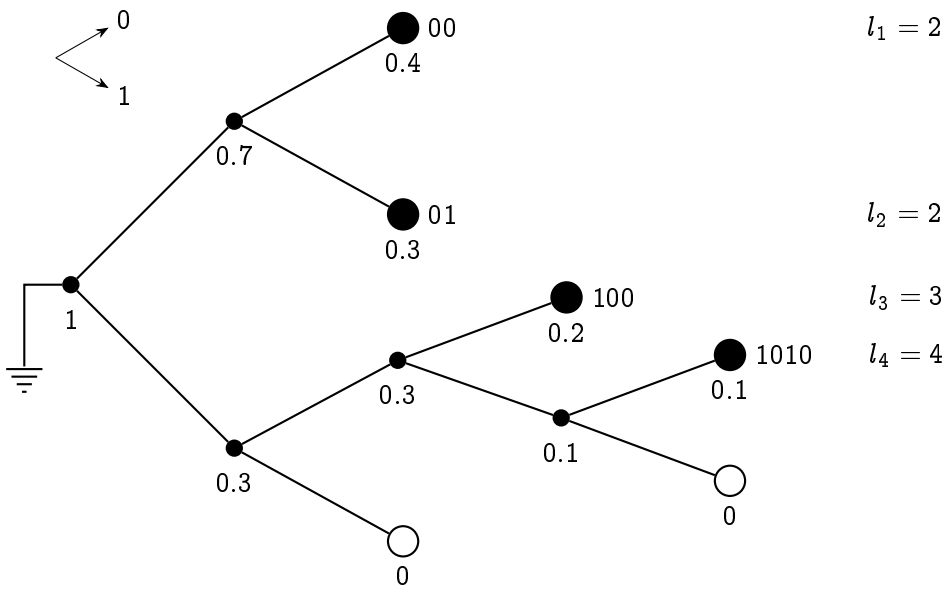


Figure 4.13: A Shannon-type code for the message U of Example 4.22.

Next, we would like to investigate how efficient such a code is. It turns out that it is very easy to find a good upper bound on the expected codeword length: By definition of l_i in (4.49),

$$l_i < \frac{\log \frac{1}{p_i}}{\log \mathrm{D}} + 1 \tag{4.58}$$

and therefore

$$\begin{aligned} \mathrm{E}[L] &= \sum_{i=1}^{r} p_i l_i && (4.59)\\ &< \sum_{i=1}^{r} p_i \left(\frac{\log \frac{1}{p_i}}{\log \mathrm{D}} + 1 \right) && (4.60)\\ &= \frac{1}{\log \mathrm{D}} \underbrace{\sum_{i=1}^{r} p_i \log \frac{1}{p_i}}_{=\mathrm{H}(U)} + \underbrace{\sum_{i=1}^{r} p_i}_{=1} && (4.61)\\ &= \frac{\mathrm{H}(U)}{\log \mathrm{D}} + 1, && (4.62) \end{aligned}$$

where we have used the definition of entropy. We see that a Shannon-type code (even though it is *not* an optimal code!) approaches the ultimate lower bound (4.47) by less than 1. An optimal code will be even better than that!

Example 4.23. We return to Example 4.22 and compute the efficiency of the Shannon-type code designed before:

$$\mathrm{E}[L] = 1 + 0.7 + 0.3 + 0.3 + 0.1 = 2.4. \tag{4.63}$$

As predicted the average codeword length satisfies

$$1.846 \le 2.4 < 1.846 + 1. \tag{4.64}$$

But obviously the code shown in Figure 4.14 is much better than the Shannon-type code. Its performance is $\mathrm{E}[L] = 2 < 2.4$. So in this example, the Shannon-type code is quite far from being optimal. ◊

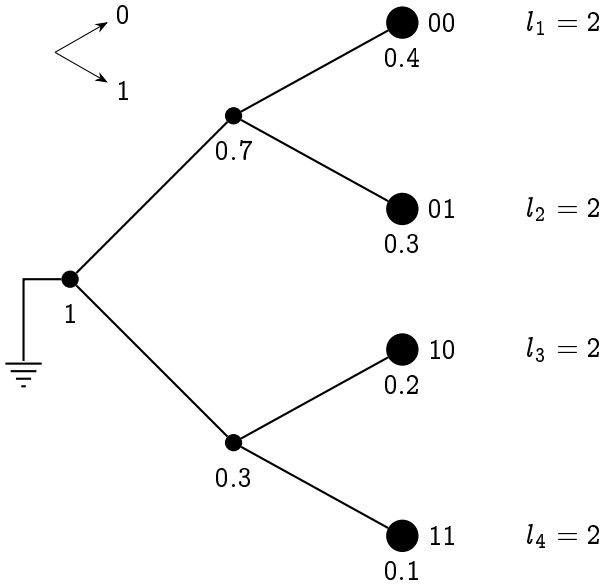


Figure 4.14: Another binary prefix-free code for U of Example 4.22.

4.8.2 Shannon Code

Shannon actually did not directly propose the Shannon-type codes, but a special case of the Shannon-type codes that we call *Shannon code* [Sha48, Section 9]. In Shannon's version the codeword lengths were chosen to satisfy (4.49), but in addition Shannon also gave a clear rule on how to choose the codewords. But before we can introduce the details, we need to make a quick remark about our common decimal representation of real numbers (and the corresponding D-ary representations).

Remark 4.24 (Decimal and D-ary Representation of Real Numbers). We are all used to the decimal representation of real numbers: For example, $\frac{1}{2}$ can be written as 0.5 or $\frac{3}{16} = 0.1875$. We are also aware that some fractions need infinitely many digits like $\frac{1}{3} = 0.3333333\ldots = 0.\overline{3}$ (we use here an overline $\overline{x}$ to represent "infinite repetition of x"), or $\frac{103}{135} = 0.7\overline{629}$. Finally, irrational numbers need infinitely many digits without any repetitive patterns.

The problem is that in contrast to how it might seem, the decimal representation of a real number is not necessarily unique! Take the example of $\frac{1}{3}$: obviously, $3 \cdot \frac{1}{3} = 1$, but if we write this in decimal form, we get $3 \cdot 0.\overline{3} = 0.\overline{9}$, i.e., we see that $0.\overline{9} = 1$. The problem is that *infinitely* many 9 at the tail of a decimal representation is equivalent to increasing the digit in front of the tail by 1 and removing the tail, e.g.,

$$0.34\overline{9} = 0.35. \tag{4.65}$$

To get rid of this nonuniqueness, we will require that no decimal representation of a real number has a tail of infinitely many 9s.

This same discussion can directly be extended to the D-ary representation of a real number. There we do not allow any infinite tail of the digit $(\mathrm{D}-1)$. Consider, for example, the following quaternary $(\mathrm{D}=4)$ representation:

$$(0.201\overline{3})_{④} = (0.202)_{④} = (0.53125)_{⑩}. \tag{4.66}$$

The problem is most pronounced for the binary representation because there we only have zeros and ones: We do not allow any representation that "ends" in a tail of infinitely many 1s. △

Definition 4.25. The *Shannon code* is a prefix-free code that is constructed as follows:

Step 1: Arrange the symbols in order of decreasing probability.

Step 2: Compute the following cumulative probability[3]

$$\mathrm{F}_i \triangleq \sum_{j=1}^{i-1} p_j \tag{4.67}$$

and express it in its D-ary representation (where we do not allow a representation with a tail of infinitely many digits $(\mathrm{D}-1)$ in order to make sure that this representation is unique[4]). For example, $\mathrm{F}_i = 0.625$ in binary form is 0.101; or $\mathrm{F}_i = \frac{55}{81}$ in ternary form is 0.2001.

Step 3: Carry out the D-ary representation to exactly l_i positions after the comma, where l_i is defined in (4.49), i.e.,

$$l_i \triangleq \left\lceil \log_{\mathrm{D}} \frac{1}{p_i} \right\rceil. \tag{4.68}$$

[3]Note that this definition does not exactly match the definition of the cumulative distribution function (CDF) of a random variable.

[4]We would like to point out that here we do not consider the numerical stability of computers, which always only calculate with a limited precision. It is quite probable that the Shannon code could be adapted to show a better performance with respect to numerical precision problems.

Table 4.15: The binary Shannon code for the random message U of Example 4.26.

	u_1	u_2	u_3	u_4
p_i	0.4	0.3	0.2	0.1
F_i	0	0.4	0.7	0.9
binary representation	0.0	0.01100...	0.10110...	0.11100...
$\left\lceil \log_2 \frac{1}{p_i} \right\rceil$	2	2	3	4
shortened representation	0.00	0.01	0.101	0.1110
$\mathbf{c}_i$	00	01	101	1110

Table 4.16: The ternary Shannon code for the random message U of Example 4.26.

	u_1	u_2	u_3	u_4
p_i	0.4	0.3	0.2	0.1
F_i	0	0.4	0.7	0.9
ternary representation	0.0	0.10121...	0.20022...	0.22002...
$\left\lceil \log_3 \frac{1}{p_i} \right\rceil$	1	2	2	3
shortened representation	0.0	0.10	0.20	0.220
$\mathbf{c}_i$	0	10	20	220

Then remove the leading "0." and use the resulting digits as codeword $\mathbf{c}_i$ for u_i. E.g., the binary $\mathrm{F}_i = (0.101)_{②}$ with $l_i = 5$ will give a codeword $\mathbf{c}_i = 10100$; or the ternary $\mathrm{F}_i = (0.2001)_{③}$ with $l_i = 3$ will give $\mathbf{c}_i = 200$.

Example 4.26. Once again, we consider the random message U of Example 4.22. The binary Shannon code is shown in Table 4.15, and the ternary Shannon code in Table 4.16. ◊

Note that the ordering of the symbols according to decreasing probability is crucial because this makes sure that the codeword lengths l_i are monotonically increasing and this in turn ensures that the code is prefix-free. We prove this formally in the following lemma.

Lemma 4.27. The Shannon code as defined in Definition 4.25 is prefix-free.

Proof: Since we do not allow a D-ary representation with an infinite tail of digits $(\mathrm{D}-1)$, the D-ary representation of F_i is unique, i.e., there exist unique $a_i^{(1)}, a_i^{(2)}, \ldots \in \{0,\ldots,\mathrm{D}-1\}$ such that

$$\mathrm{F}_i = \sum_{j=1}^{\infty} a_i^{(j)} \mathrm{D}^{-j} \tag{4.69}$$

and such that

$$(a_i^{(j')}, a_i^{(j'+1)}, a_i^{(j'+2)}, \ldots) \neq (\mathrm{D}-1, \mathrm{D}-1, \mathrm{D}-1, \ldots) \tag{4.70}$$

for any $j' \in \mathbb{N}$. Moreover, from (4.50) it follows (solve for p_i!) that

$$p_i \geq \frac{1}{\mathrm{D}^{l_i}}. \tag{4.71}$$

By contradiction, assume that the codeword $\mathbf{c}_i$ is the prefix of another codeword $\mathbf{c}_{i'}$, $i \neq i'$. This means that the length l_i of $\mathbf{c}_i$ cannot be longer than the length $l_{i'}$ of $\mathbf{c}_{i'}$, which could happen in two cases:

- $i \leq i'$ and therefore by definition $l_i \leq l_{i'}$, or
- $i > i'$, but $l_i = l_{i'}$.

Since the latter case equals to the first if we exchange the role of i and i', we do not need to further pursue it, but can concentrate on the former case only. Now, since $\mathbf{c}_i$ is the prefix of $\mathbf{c}_{i'}$, we must have that

$$a_i^{(j)} = a_{i'}^{(j)}, \quad j = 1,\ldots,l_i. \tag{4.72}$$

Hence,

$$\begin{aligned}
\mathrm{F}_{i'} - \mathrm{F}_i &= \sum_{j=1}^{\infty} \Big(a_{i'}^{(j)} - a_i^{(j)}\Big) \mathrm{D}^{-j} && \text{(by (4.69))} && (4.73)\\
&= \sum_{j=l_i+1}^{\infty} \underbrace{\Big(a_{i'}^{(j)} - a_i^{(j)}\Big)}_{\leq \mathrm{D}-1} \mathrm{D}^{-j} && \text{(by (4.72))} && (4.74)\\
&< \sum_{j=l_i+1}^{\infty} (\mathrm{D}-1)\mathrm{D}^{-j} && && (4.75)\\
&= \sum_{j=l_i+1}^{\infty} \mathrm{D}^{-j+1} - \sum_{j=l_i+1}^{\infty} \mathrm{D}^{-j} && && (4.76)\\
&= \sum_{j'=l_i}^{\infty} \mathrm{D}^{-j'} - \sum_{j=l_i+1}^{\infty} \mathrm{D}^{-j} && && (4.77)\\
&= \mathrm{D}^{-l_i} + \sum_{j'=l_i+1}^{\infty} \mathrm{D}^{-j'} - \sum_{j=l_i+1}^{\infty} \mathrm{D}^{-j} && && (4.78)\\
&= \mathrm{D}^{-l_i}. && && (4.79)
\end{aligned}$$

Here, the strict inequality in (4.75) holds because the difference term can only be equal to $\mathrm{D}-1$ if $a_{i'}^{(j)} = \mathrm{D}-1$ (and $a_i^{(j)} = 0$) for *all* $j \geq l_i + 1$, which we have specifically excluded in (4.70) (see also Remark 4.24); and in (4.77), we renumber the first sum $j' \triangleq j - 1$.

On the other hand, by definition of F_i and $\mathrm{F}_{i'}$,

$$\begin{aligned} \mathrm{F}_{i'} - \mathrm{F}_i &= p_i + p_{i+1} + \cdots + p_{i'-1} && (4.80)\\ &\geq \mathrm{D}^{-l_i} + \mathrm{D}^{-l_{i+1}} + \cdots + \mathrm{D}^{-l_{i'-1}}, && (4.81) \end{aligned}$$

where the inequality follows by repeated application of (4.71).

Combining (4.79) and (4.81) yields the following contradiction:

$$\mathrm{D}^{-l_i} > \mathrm{D}^{-l_i} + \mathrm{D}^{-l_{i+1}} + \cdots + \mathrm{D}^{-l_{i'-1}}, \tag{4.82}$$

and thereby shows that $\mathbf{c}_i$ cannot be a prefix of $\mathbf{c}_{i'}$. This concludes the proof.[5] □

Obviously, the Shannon code also is a Shannon-type code (because (4.49) and (4.68) are identical) and therefore its performance is identical to the codes described in Section 4.8.1. So we do not actually need to bother about the exact definition of the codewords specified by Shannon, but could simply construct any tree as shown for the Shannon-type codes. The importance of the Shannon code, however, lies in the fact that it was the origin of *arithmetic coding*. We will come back to this in Section 5.3.

4.8.3 Fano Code

Around the same time as Shannon proposed the Shannon code, Fano suggested a different code that actually turns out to be very easy to implement in hardware [Fan49].

Definition 4.28. The *Fano code* is generated according to the following algorithm:

Step 1: Arrange the symbols in order of decreasing probability.

Step 2: Split the list of ordered symbols into D parts, with the total probabilities q_ℓ of each part being as similar as possible, i.e., such that

$$\frac{1}{\mathrm{D}} \sum_{\ell=1}^{\mathrm{D}} \sum_{\ell'=1}^{\mathrm{D}} \left| q_\ell - q_{\ell'} \right| \tag{4.83}$$

is minimized.

[5] Many thanks go to Tomas Kroupa for pointing out the nonuniqueness of the D-ary representation and for suggesting this proof.

Step 3: Assign the digit 0 to the first part, the digit 1 to the second part, ..., and the digit $D-1$ to the last part. This means that the codewords for the symbols in the first part will all start with 0, and the codewords for the symbols in the second part will all start with 1, etc.

Step 4: Recursively apply Step 2 and Step 3 to each of the D parts, subdividing each part into further D parts and adding bits to the codewords until each symbol is the single member of a part.

Note that effectively this algorithm constructs a tree and that therefore the Fano code is prefix-free.

Example 4.29. Let us generate a binary Fano code for a random message with five symbols having probabilities

$$\begin{aligned} &p_1 = 0.35, \qquad p_2 = 0.25, \qquad p_3 = 0.15,\\ &p_4 = 0.15, \qquad p_5 = 0.1. \end{aligned} \tag{4.84}$$

Since the symbols are already ordered in decreasing order of probability, Step 1 can be omitted. We hence want to split the list into two parts, both having as similar total probability as possible. If we split in $\{1\}$ and $\{2,3,4,5\}$, we have a total probability 0.35 on the left and 0.65 on the right; the split $\{1,2\}$ and $\{3,4,5\}$ yields 0.6 and 0.4; and $\{1,2,3\}$ and $\{4,5\}$ gives 0.75 and 0.25. We see that the second split is best. So we assign 0 as a first digit to $\{1,2\}$ and 1 to $\{3,4,5\}$.

Now we repeat the procedure with both subgroups. Firstly, we split $\{1,2\}$ into $\{1\}$ and $\{2\}$. This is trivial. Secondly, we split $\{3,4,5\}$ into $\{3\}$ and $\{4,5\}$ because 0.15 and 0.25 is closer to each other than 0.3 and 0.1 that we would have gotten by splitting into $\{3,4\}$ and $\{5\}$. Again we assign the corresponding codeword digits.

Finally, we split the last group $\{4,5\}$ into $\{4\}$ and $\{5\}$. We end up with the five codewords $\{00,01,10,110,111\}$. This whole procedure is shown in Figure 4.17.

In Figure 4.18 a ternary Fano code is constructed for the same random message. ◊

Exercise 4.30. *Construct the binary Fano code for the random message U of Example 4.22 with four symbols having probabilities*

$$p_1 = 0.4, \qquad p_2 = 0.3, \qquad p_3 = 0.2, \qquad p_4 = 0.1, \tag{4.85}$$

and compute its performance. Compare to the Shannon-type code designed in Example 4.22 and 4.23. ◊

Remark 4.31. We would like to point out that there are cases where the algorithm given in Definition 4.28 does not lead to a unique design: There might

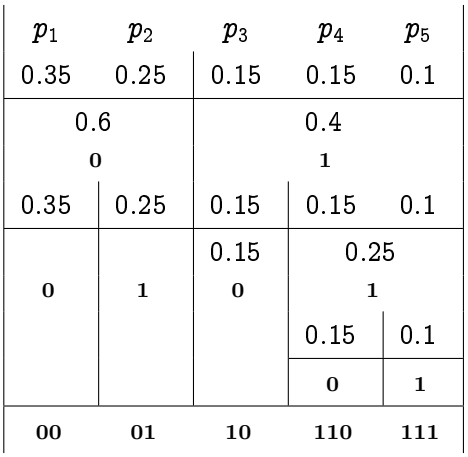


Figure 4.17: Construction of a binary Fano code according to Example 4.29.

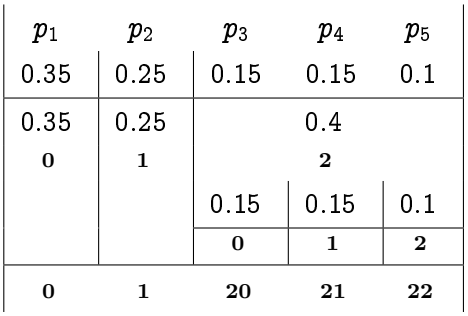


Figure 4.18: Construction of a ternary Fano code according to Example 4.29.

be two different ways of dividing the list into D parts such that the total probabilities are as similar as possible. Since the algorithm does not specify what to do in such a case, you are free to choose any possible way. Unfortunately, however, these different choices can lead to codes with different performance. As an example consider a random message U with seven possible symbols having the following probabilities:

$$\begin{aligned} &p_1 = 0.35, \quad p_2 = 0.3, \quad p_3 = 0.15, \quad p_4 = 0.05,\\ &p_5 = 0.05, \quad p_6 = 0.05, \quad p_7 = 0.05. \end{aligned} \tag{4.86}$$

Figure 4.19 and 4.20 show two different possible Fano codes for this random message. The first has an average codeword length of $\mathsf{E}[L] = 2.45$, while the latter performs better with $\mathsf{E}[L] = 2.4$. △

Exercise 4.32. *In total there are six different possible designs of a Fano code for the random message given in Remark 4.31. Design all of them and compare their performances!* ◊

Similarly to the Shannon-type codes, we also would like to have some performance bounds for the Fano code. When playing around with some

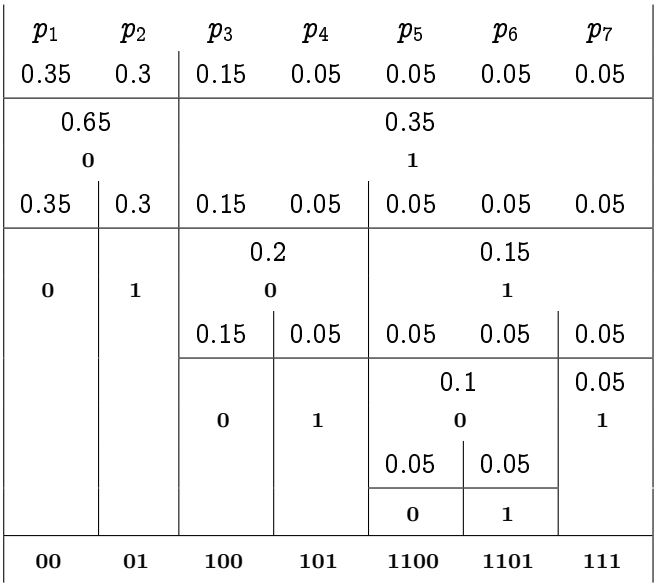


Figure 4.19: One possible Fano code for the random message given in (4.86).

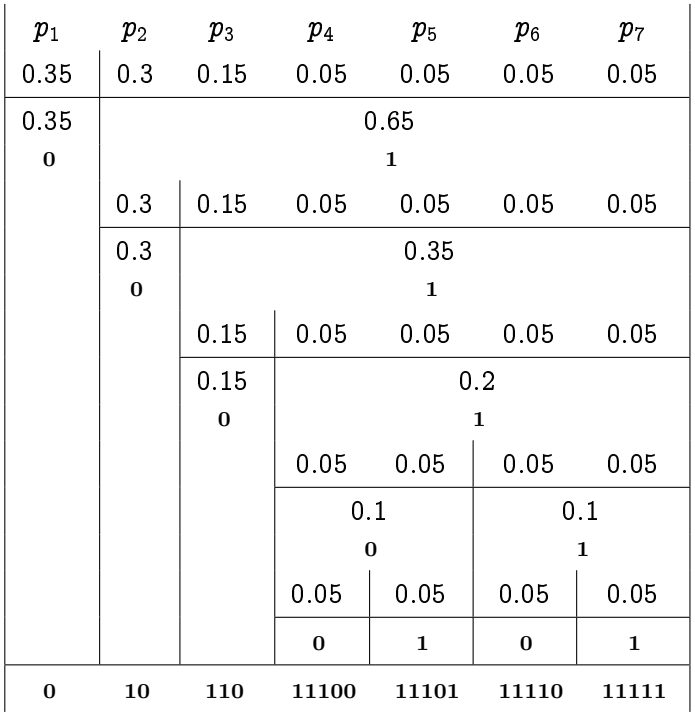


Figure 4.20: A second possible Fano code for the random message given in (4.86).

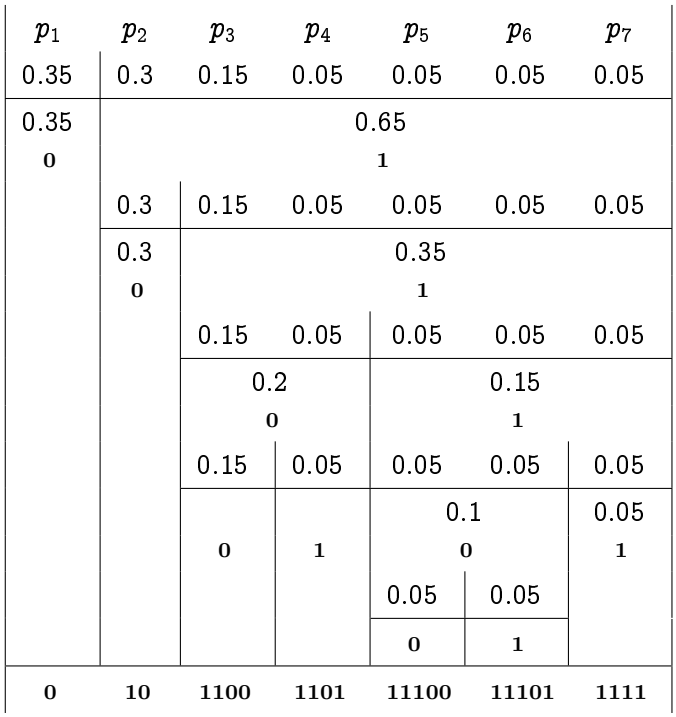


Figure 4.21: A third possible Fano code for the random message given in (4.86).

examples, one quickly gets the feeling that the Fano code is in general *better* than the Shannon-type codes. This seems reasonable because a Fano code will in general have fewer unused leaves. Thus, one would expect that also the Fano code satisfies the inequality (4.58). Unfortunately, this is not true, as can be seen from the following example.

Example 4.33. We construct a third possible Fano code for the random message given in (4.86), see Figure 4.21. In this Fano code, $l_3 = 4$, but

$$\left\lceil \log_2 \frac{1}{p_3} \right\rceil = \left\lceil \log_2 \frac{1}{0.15} \right\rceil = 3, \tag{4.87}$$

i.e., (4.58) is violated:

$$4 \not< 3 + 1. \tag{4.88}$$

◊

Exercise 4.34. *Construct a binary Fano code for a random message U with eight symbols of probabilities:*

$$\begin{aligned} &p_1 = 0.44, \quad p_2 = 0.14, \quad p_3 = 0.13, \quad p_4 = 0.13,\\ &p_5 = 0.13, \quad p_6 = 0.01, \quad p_7 = 0.01, \quad p_8 = 0.01. \end{aligned} \tag{4.89}$$

Note that l_5 violates (4.58)*. Compute the performance of this code. Is* (4.62) *satisfied?* ◊

In spite of the fact that (4.58) sometimes is violated, it turns out that we can prove a bound for the Fano codes that is even better than the bound given in (4.62).

Theorem 4.35. For $\mathrm{D} \in \{2,3\}$, the average codeword length $\mathsf{E}[L]$ of a D-ary Fano code for an r-ary random message U satisfies

$$\mathsf{E}[L] \le \frac{\mathrm{H}(U)}{\log \mathrm{D}} + 1 - p_{\min}, \tag{4.90}$$

where $p_{\min} \triangleq \min_i p_i$ denotes the smallest positive probability of U.

Proof: While this theorem only has been proven for $\mathrm{D} \in \{2,3\}$, it is conjectured to hold for all $\mathrm{D} \in \{2,3,\ldots\}$. The details are omitted. □

Keep in mind that neither Fano nor Shannon-type codes are optimal. The optimal code will perform at least as good. We will derive the optimal codes in Section 4.9.

Exercise 4.36. *Construct a ternary Fano code for the random message U of Example 4.22 and compute its performance.* ◊

Exercise 4.37. *Construct a ternary Fano code for a random message U with 14 symbols of probabilities:*

$$\begin{aligned} p_1 &= 0.3, & p_2 &= 0.12, & p_3 &= 0.11, & p_4 &= 0.11,\\ p_5 &= 0.1, & p_6 &= 0.03, & p_7 &= 0.03, & p_8 &= 0.03,\\ p_9 &= 0.03, & p_{10} &= 0.03, & p_{11} &= 0.03, & p_{12} &= 0.03,\\ p_{13} &= 0.03, & p_{14} &= 0.02. \end{aligned} \tag{4.91}$$

Note that l_5 violates (4.58)*. Compute the performance of this code and compare with* (4.62)*.* ◊

4.8.4 Coding Theorem for a Single Random Message

We summarize the so far most important results.

Theorem 4.38 (Coding Theorem for a Single Random Message).
The average codeword length $\mathsf{E}[L]$ of an optimal D-ary prefix-free code for an r-ary random message U satisfies

$$\frac{\mathrm{H}(U)}{\log \mathrm{D}} \le \mathsf{E}[L] < \frac{\mathrm{H}(U)}{\log \mathrm{D}} + 1 \tag{4.92}$$

with equality on the left if, and only if, $P_U(u_i) = p_i$ is a negative integer power of D, $\forall\, i$.

Moreover, this statement also holds true for Shannon-type codes, Shannon codes, and Fano codes.[6]

Proof: The lower bound follows from (4.47) that holds for all codes. The upper bound follows from the upper bound (4.62) on the performance of a Shannon code (which is not optimal, but whose performance cannot be better than the performance of an optimal code). Finally, that (4.92) also holds for Fano codes follows from Theorem 4.35 (which is proven for $\mathrm{D} \in \{2,3\}$, but only conjectured for $\mathrm{D} \geq 4$). □

This is the first step towards showing that the definition of $\mathrm{H}(U)$ is truly useful!

Remark 4.39 (Confusion of Names). Note that unfortunately the three codes from Sections 4.8.1, 4.8.2, and 4.8.3, i.e., the Shannon-type codes, the Shannon code, and the Fano codes are all known by the same name of *Shannon–Fano code*.[7] The reason for this confusion probably stems from the fact that firstly all three codes perform very similarly and secondly that in [Sha48, p. 17], Shannon refers to Fano's code construction algorithm (even though he then proposes the Shannon code).

To add to the confusion, sometimes the Shannon code is also known as *Shannon–Fano–Elias code* [CT06, Section 5.9], adding the name of Peter Elias. However, Prof. Elias from MIT definitely was not involved in the development of the Shannon code or the Fano code. Instead, he did make fundamental contributions to *arithmetic coding*, whose origins lie in the Shannon code. Indeed, sometimes Elias is even credited to be the inventor of arithmetic coding, but actually Elias denied this [Say99, Section 1.2]. The idea of arithmetic coding probably has come from Shannon himself during a talk that he gave at MIT. △

At this point, one should also ask the question what happens if we design a code according to a wrong distribution. So we design a Shannon-type code for the message U where we mistakenly assume that U has a distribution $P_{\tilde{U}}$ instead of the actual PMF P_U, i.e., we choose wrong codeword lengths

$$l_i \triangleq \left\lceil \frac{\log \frac{1}{\tilde{p}_i}}{\log \mathrm{D}} \right\rceil. \tag{4.93}$$

The performance of such a wrongly designed code can also be analyzed.

[6] In the case of Fano codes and $\mathrm{D} \geq 4$, the upper bound is only conjectured to hold.

[7] The names used in these lecture notes have been newly "invented" and are not used in the general literature like this.

Theorem 4.40 (Wrongly Designed Shannon-Type Code).
The performance of a Shannon-type code that is designed for the message $\tilde{U}$, but is used to describe the message U satisfies the following bounds:

$$\frac{\mathrm{H}(U)}{\log \mathrm{D}} + \frac{\mathscr{D}(P_U \,\|\, P_{\tilde{U}})}{\log \mathrm{D}} \le \mathsf{E}[L] < \frac{\mathrm{H}(U)}{\log \mathrm{D}} + \frac{\mathscr{D}(P_U \,\|\, P_{\tilde{U}})}{\log \mathrm{D}} + 1. \tag{4.94}$$

Here, $\mathscr{D}(\cdot\|\cdot)$ denotes *relative entropy* defined in Definition 3.1.

So we see that the relative entropy tells us how much we lose by designing according to the wrong distribution.

Proof: From (4.93), we note that

$$l_i = \left\lceil \frac{\log \frac{1}{\tilde{p}_i}}{\log \mathrm{D}} \right\rceil < \frac{\log \frac{1}{\tilde{p}_i}}{\log \mathrm{D}} + 1 \tag{4.95}$$

and obtain

$$\begin{aligned}
\mathsf{E}[L] &= \sum_{i=1}^{r} p_i l_i && (4.96)\\
&< \sum_{i=1}^{r} p_i \left(\frac{\log \frac{1}{\tilde{p}_i}}{\log \mathrm{D}} + 1 \right) && (4.97)\\
&= \frac{1}{\log \mathrm{D}} \sum_{i=1}^{r} p_i \log\left(\frac{1}{\tilde{p}_i} \cdot \frac{p_i}{p_i} \right) + \sum_{i=1}^{r} p_i && (4.98)\\
&= \frac{1}{\log \mathrm{D}} \sum_{i=1}^{r} p_i \log \frac{p_i}{\tilde{p}_i} + \frac{1}{\log \mathrm{D}} \sum_{i=1}^{r} p_i \log \frac{1}{p_i} + 1 && (4.99)\\
&= \frac{\mathscr{D}(P_U \,\|\, P_{\tilde{U}})}{\log \mathrm{D}} + \frac{\mathrm{H}(U)}{\log \mathrm{D}} + 1. && (4.100)
\end{aligned}$$

For the lower bound, we bound

$$l_i = \left\lceil \frac{\log \frac{1}{\tilde{p}_i}}{\log \mathrm{D}} \right\rceil \ge \frac{\log \frac{1}{\tilde{p}_i}}{\log \mathrm{D}}. \tag{4.101}$$

The rest is then completely analogous to (4.96)–(4.100). □

4.9 Optimal Codes: Huffman Code

Shannon actually did not find the optimal code design. He was working together with Prof. Robert M. Fano on the problem, but neither of them could solve it. So in 1951 Fano assigned the question to his students in an information theory class at MIT as a term paper. David A. Huffman, who had finished his B.S. and M.S. in electrical engineering and also served in the U.S. Navy

before he became a Ph.D. student at MIT, attended this course. Huffman tried a long time and was about to give up when he had the sudden inspiration to start building the tree backwards from leaves to root instead from root to leaves as everybody had been trying so far. Once he had understood this, he was quickly able to prove that his code was the most efficient one. Naturally, Huffman's term paper was later on published [Huf52].

To understand Huffman's design we firstly recall the average length of a code:

$$\mathsf{E}[L] = \sum_{i=1}^{r} p_i l_i. \tag{4.102}$$

Then we make the following general observation.

Lemma 4.41. Without loss in generality, we arrange the probabilities p_i in non-increasing order and consider an arbitrary code. If the lengths l_i of the codewords of this code are not in the opposite order, i.e., we do not have both

$$p_1 \ge p_2 \ge p_3 \ge \cdots \ge p_r \tag{4.103}$$

and

$$l_1 \le l_2 \le l_3 \le \cdots \le l_r, \tag{4.104}$$

then the code is not optimal and we can achieve a shorter average length by reassigning the codewords to different symbols.

Proof: To prove this claim, suppose that for some i and j with $i < j$ we have both

$$p_i > p_j \quad \text{and} \quad l_i > l_j. \tag{4.105}$$

In computing the average length, originally the sum in (4.102) contains, among others, the two terms

$$\text{old:} \quad p_i l_i + p_j l_j. \tag{4.106}$$

By interchanging the codewords for the ith and jth symbols, we get the terms

$$\text{new:} \quad p_i l_j + p_j l_i \tag{4.107}$$

while the remaining terms are unchanged. Subtracting the old from the new we see that

$$\begin{aligned} \text{new} - \text{old:} \quad (p_i l_j + p_j l_i) - (p_i l_i + p_j l_j) &= p_i(l_j - l_i) + p_j(l_i - l_j) && (4.108)\\ &= (p_i - p_j)(l_j - l_i) && (4.109)\\ &< 0, && (4.110) \end{aligned}$$

where the last inequality follows from (4.105). Thus we can decrease the average codeword length by interchanging the codewords for the ith and jth symbols. Hence the new code with exchanged codewords for the ith and jth symbols is better than the original code — which therefore cannot have been optimal. □

To simplify matters, we develop the optimal Huffman code firstly for the binary case $\mathrm{D} = 2$.

The clue of binary Huffman coding lies in two basic observations. The first observation is as follows.

Lemma 4.42. The binary tree of an optimal binary prefix-free code has no unused leaves.

Proof: Suppose that the tree of an optimal code has an unused leaf. Then we can delete this leaf and put its sibling to their parent node, see Figure 4.22. By doing so we reduce $\mathrm{E}[L]$, which contradicts our assumption that the original code was optimal. □

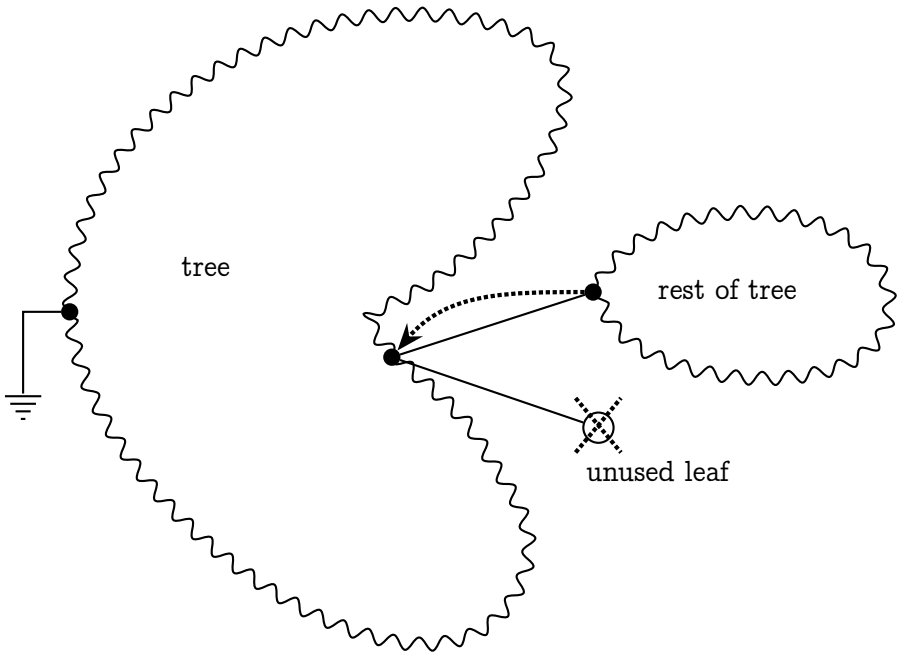


Figure 4.22: Code performance and unused leaves: By deleting the unused leaf and moving its sibling to their parent, we can improve on the code's performance.

Example 4.43. As an example consider the two codes given in Figure 4.23, both of which have three codewords. Code (i) has an average length[8] of $\mathsf{E}[L] = 2$, and code (ii) has an average length of $\mathsf{E}[L] = 1.5$. Obviously, code (ii) performs better. ◊

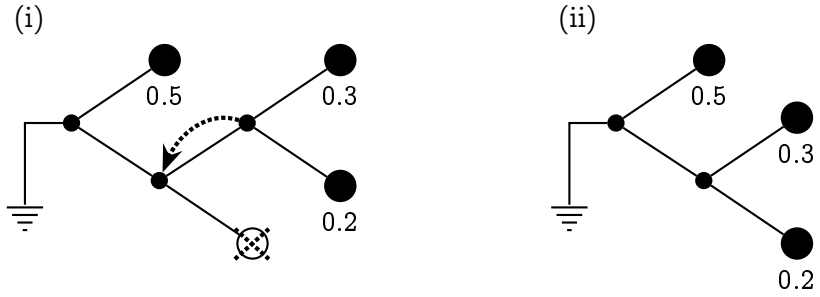


Figure 4.23: Improving a code by removing an unused leaf.

The second observation basically says that the two most unlikely symbols must have the longest codewords.

Lemma 4.44. There exists an optimal binary prefix-free code such that the two least likely codewords only differ in the last digit, i.e., the two most unlikely codewords are siblings.

Proof: Since we consider an optimal code and because of Lemma 4.41, the codewords that correspond to the two least likely symbols must be the longest codewords. Now assume for a moment that these two longest codewords do *not* have the same length. In this case, the sibling node of the longest codeword must be unused, which is a contradiction to Lemma 4.42. Hence, we understand the two least likely symbols must have the two longest codewords and are of identical length.

Now, if they have the same parent node, we are done. If they do not have the same parent node, this means that there exist other codewords of the same maximum length. In this case we can simply swap two codewords of equal maximum length in such a way that the two least likely codewords have the same parent, and we are done. □

Because of Lemma 4.42 and the Path Length Lemma (Lemma 4.15), we see that the construction of an optimal binary prefix-free code for an r-ary random message U is equivalent to constructing a binary tree with r leaves such that the sum of the probabilities of the nodes is minimum when the

[8] Remember Lemma 4.15 to compute the average codeword length: summing the node probabilities. In code (i) we have $\mathsf{P}_1 = 1$ and $\mathsf{P}_2 = \mathsf{P}_3 = 0.5$ (note that the unused leaf has by definition zero probability) and in code (ii) $\mathsf{P}_1 = 1$ and $\mathsf{P}_2 = 0.5$.

leaves are assigned the probabilities $P_U(u_i)$ for $i = 1, 2, \ldots, r$:

$$\mathsf{E}[L] = \underbrace{\sum_{\ell=1}^{\mathsf{N}} \mathsf{P}_\ell}_{\longrightarrow \text{minimize!}}. \tag{4.111}$$

But Lemma 4.44 tells us how we may choose one node in an optimal code tree, namely as parent of the two least likely leaves u_{r-1} and u_r:

$$\mathsf{P}_{\mathsf{N}} = P_U(u_{r-1}) + P_U(u_r). \tag{4.112}$$

So we have fixed one P_ℓ in (4.111) already. But, if we now pruned our binary tree at this node to make it a leaf with probability $p = P_U(u_{r-1}) + P_U(u_r)$, it would become one of $r-1$ leaves in a new tree. Completing the construction of the optimal code would then be equivalent to constructing a binary tree with these $r-1$ leaves such that the sum of the probabilities of the nodes is minimum. Again Lemma 4.44 tells us how to choose one node in this new tree. Etc. We have thus proven the validity of the following algorithm.

Huffman's Algorithm for Optimal Binary Codes:

Step 1: Create r leaves corresponding to the r possible symbols and assign their probabilities $p_1, \ldots, p_r$. Mark these leaves as *active*.

Step 2: Create a new node that has the two **least likely** *active* leaves or nodes as children. *Activate* this new node and *deactivate* its children.

Step 3: If there is only one *active* node left, root it. Otherwise, go to Step 2.

Example 4.45. In Figure 4.24 we show the procedure of producing a binary Huffman code for the random message U of Example 4.22 with four possible symbols with probabilities $p_1 = 0.4$, $p_2 = 0.3$, $p_3 = 0.2$, $p_4 = 0.1$. We see that the average codeword length of this Huffman code is

$$\mathsf{E}[L] = 0.4 \cdot 1 + 0.3 \cdot 2 + 0.2 \cdot 3 + 0.1 \cdot 3 = 1.9. \tag{4.113}$$

Using Lemma 4.15 this can be computed much easier as

$$\mathsf{E}[L] = \mathsf{P}_1 + \mathsf{P}_2 + \mathsf{P}_3 = 1 + 0.6 + 0.3 = 1.9. \tag{4.114}$$

$\Diamond$

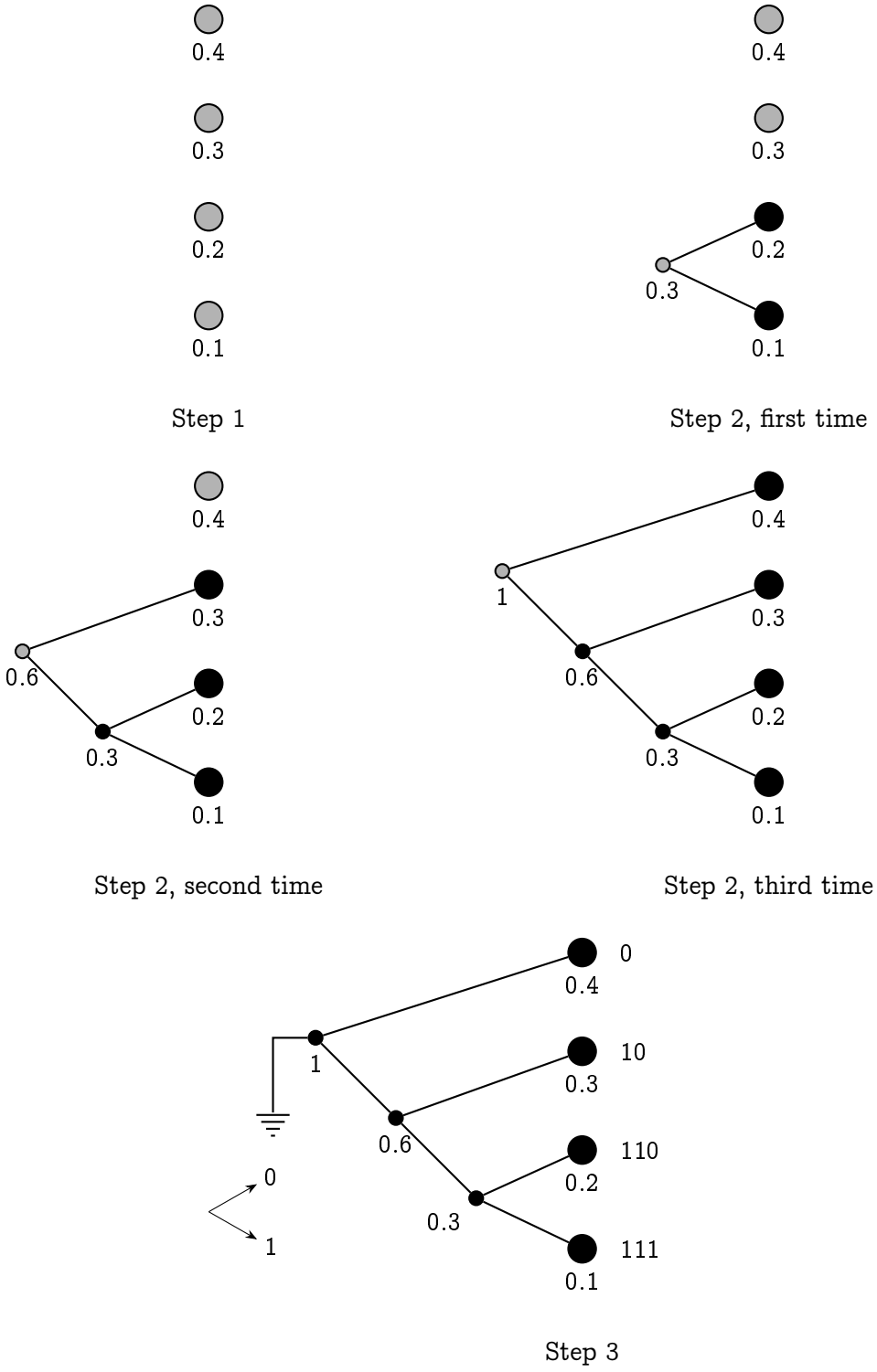


Figure 4.24: Creation of a binary Huffman code. *Active* nodes and leaves are shaded.

Remark 4.46. Note that similarly to the construction of the Fano code, the code design process is not unique in several respects. Firstly, the assignment of the 0 or 1 digits to the codewords at each forking stage is arbitrary, but this produces only trivial differences. Usually, we will stick to the convention that going upwards corresponds to 0 and downwards to 1. Secondly, when there are more than two least likely *active* nodes/leaves, it does not matter which we choose to combine. The resulting codes can have codewords of different lengths, however, in contrast to the case of the Fano code, the average codeword length will always be identical! The reason for this is clear: We have proven that the Huffman algorithm results in an optimal code! △

Example 4.47. As an example of different Huffman encodings of the same random message, let $p_1 = 0.4$, $p_2 = 0.2$, $p_3 = 0.2$, $p_4 = 0.1$, $p_5 = 0.1$. Figure 4.25 shows three different Huffman codes for this message. All of them have the same performance $\mathsf{E}[L] = 2.2$. ◊

Exercise 4.48. *Try to generate all three codes of Example 4.47 (see Figure 4.25) yourself.* ◊

We now turn to the general case with $\mathrm{D} \geq 2$. What happens, e.g., if $\mathrm{D} = 3$? In Example 4.45 with four codewords, we note that there does not exist any ternary code with 4 leaves (see the Leaf Counting Lemma, Lemma 4.10). Hence, the ternary Huffman code for a random message with four symbols must have some unused leaves! Obviously, the number of unused leaves should be minimized as they are basically a waste. Moreover, a leaf should only be unused if it has the longest length among all leaves in the tree.[9]

So we realize that, because we are growing the optimal code tree from its leaves, we need to find out how many unused leaves are needed *before* we start! To do this, we recall the Leaf Counting Lemma (Lemma 4.10): For the example of $\mathrm{D} = 3$ and $r = 4$, we have

$$n = 1 + \mathrm{N}(\mathrm{D} - 1) = 1 + 2\mathrm{N}, \tag{4.115}$$

i.e., $n = 3, 5, 7, 9, \ldots$ Hence, the smallest number of leaves that is not less than $r = 4$ is $n = 5$.

We make the following observation.

Lemma 4.49. There are at most $\mathrm{D} - 2$ unused leaves in an optimal D-ary prefix-free code, and there exists an optimal code where all unused leaves have the same parent node.

[9] Note that a unused leaf is equivalent to a message of zero probability: It never shows up! But from Lemma 4.41 we have understood that for messages with low probability we should assign long codewords.

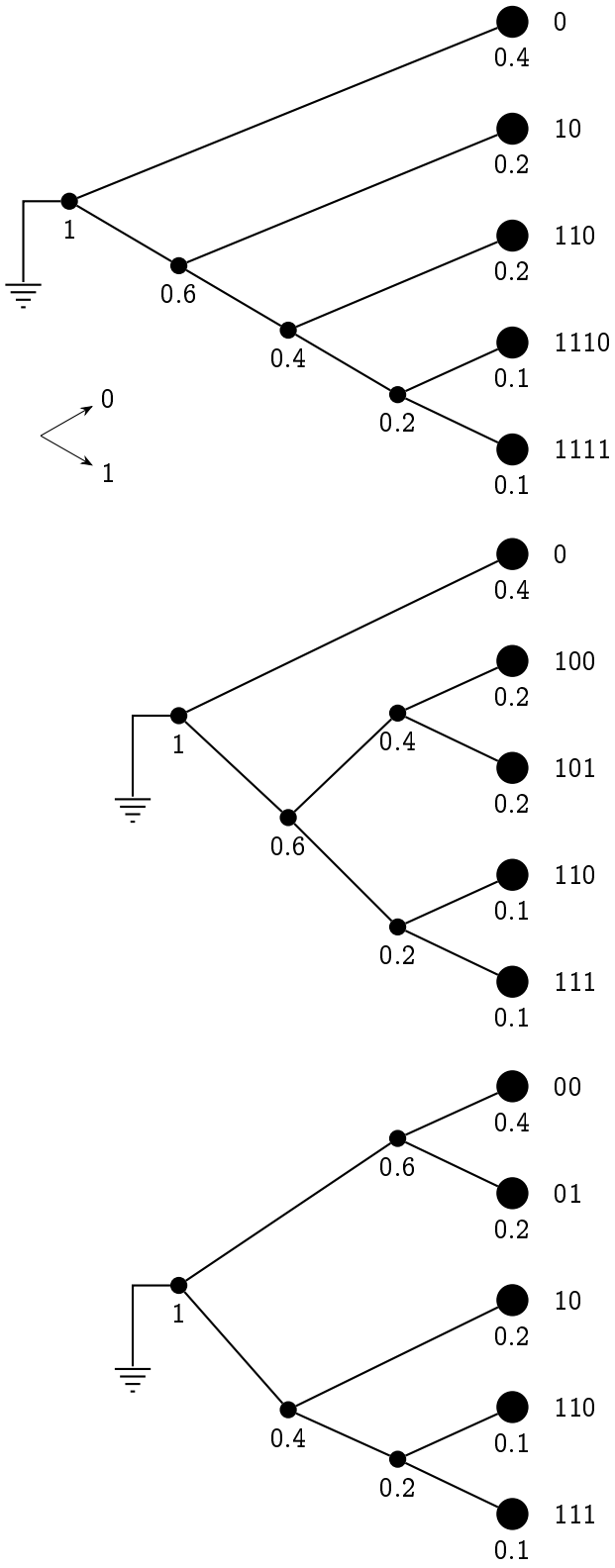


Figure 4.25: Different binary Huffman codes for the same random message.

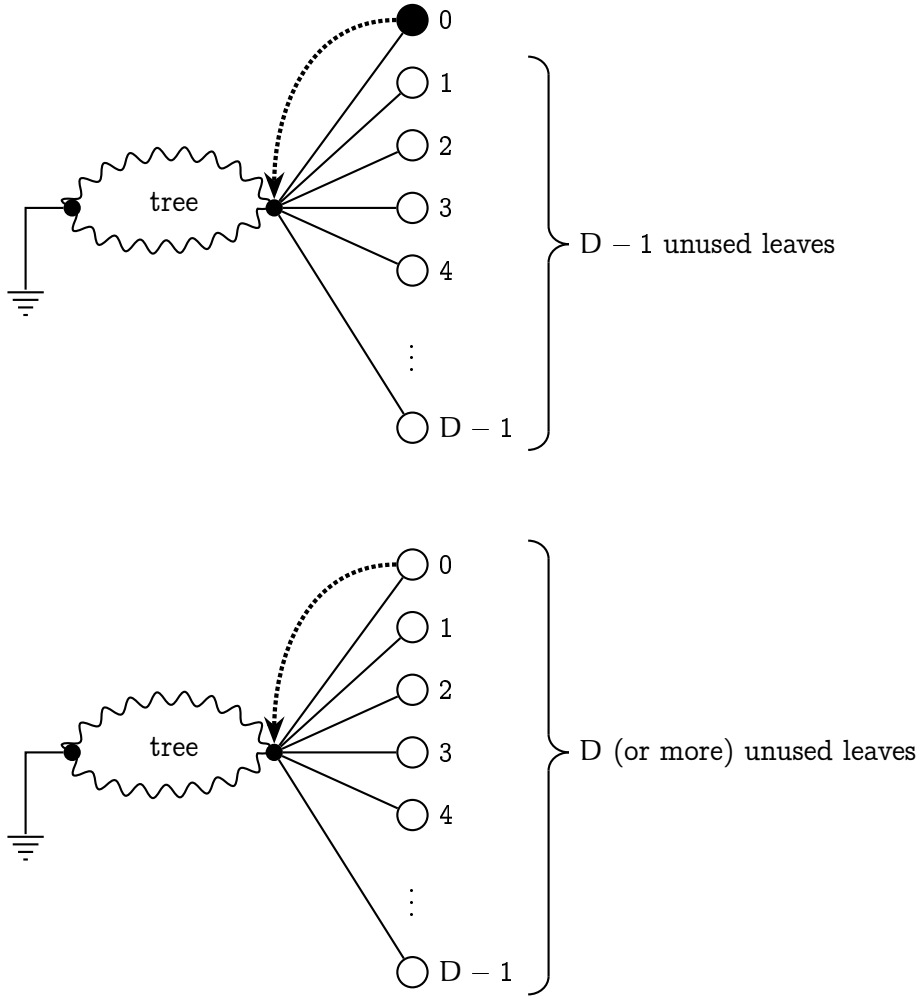


Figure 4.26: Illustration of cases when there are more than D−2 unused leaves in a D-ary tree.

Proof: The arguments are basically the same as in the binary case. If there are $D-1$ or more unused leaves in an optimal tree, then they can be deleted, which will increase the efficiency of a code. See Figure 4.26 for an illustration. The argument of the second statement is identical to the argument of Lemma 4.44. □

Next we want to derive a formula that allows us to directly compute the number of unused leaves. Let R be the number of unused leaves. By Lemma 4.49 we know that $0 \le R < D-1$, and from the Leaf Counting Lemma (Lemma 4.10) we get:

$$R = (\#\text{ of leaves in the D-ary tree}) - (\#\text{ of used leaves}) \quad (4.116)$$
$$= n - r \quad (4.117)$$

$$= 1 + N(D-1) - r \tag{4.118}$$
$$= D + (N-1)(D-1) - r \tag{4.119}$$
$$\Longrightarrow\ D - r = -(N-1)(D-1) + R \tag{4.120}$$

where $0 \le R < D-1$. This looks exactly like Euclid's Division Theorem when dividing $D-r$ by $D-1$! Hence, using $\mathscr{R}_x(y)$ to denote the remainder when y is divided by x, we finally get

$$R = \mathscr{R}_{D-1}(D-r) \tag{4.121}$$
$$= \mathscr{R}_{D-1}(D-r+(r-D)(D-1)) \tag{4.122}$$
$$= \mathscr{R}_{D-1}((r-D)(D-2)), \tag{4.123}$$

where the second equality follows because $\mathscr{R}_{D-1}(\ell(D-1)) = 0$ for all $\ell \in \mathbb{N}$.

Lemma 4.50. The number of unused leaves in the tree of an optimal D-ary prefix-free code is

$$R = \mathscr{R}_{D-1}((r-D)(D-2)). \tag{4.124}$$

The final observation now follows the lines of Lemma 4.44.

Lemma 4.51. There exists an optimal D-ary prefix-free code for a random message U with r possible values such that the $D-R$ least likely codewords differ only in their last digit. Here R is given in (4.124).

The Path Length Lemma (Lemma 4.15) tells us that we need to minimize the sum of node probabilities, and Lemma 4.51 tells us how to choose one such optimal node, namely, a node that combines the $D-R$ least likely codewords and the R unused leaves. If we prune the tree at this new node, Lemma 4.49 tells us that there will be no unused leaves in the remaining D-ary tree. So we continue to create nodes by combining the D least likely leaves in the remaining tree and then prune the tree at this node.

We have justified the following algorithm of designing an optimal code.

Huffman's Algorithm for Optimal D-ary Codes:

Step 1: Create r leaves corresponding to the r possible symbols and assign their probabilities $p_1, \ldots, p_r$. Compute

$$\mathrm{R} \triangleq \mathscr{R}_{\mathrm{D}-1}((r-\mathrm{D})(\mathrm{D}-2)), \tag{4.125}$$

create R leaves, and assign the probability 0 to them. Mark all leaves as *active*.

Step 2: Create a new node that has the D **least likely** *active* leaves or nodes as children. *Activate* this new node and *deactivate* its children.

Step 3: If there is only one *active* node left, root it. Otherwise go to Step 2.

Example 4.52. We return to Example 4.22 and design an optimal ternary ($\mathrm{D} = 3$) code. Note that $r = 4$, i.e.,

$$\mathrm{R} = \mathscr{R}_2\big((4-3)\cdot(3-2)\big) = 1. \tag{4.126}$$

In Figure 4.27 we show the procedure of designing this Huffman code. We get $\mathsf{E}[L] = 1.3$. Note that

$$\frac{\mathrm{H}(U)}{\log_2 3} = 1.16 \le \mathsf{E}[L] < \frac{\mathrm{H}(U)}{\log_2 3} + 1 = 2.16 \tag{4.127}$$

as expected.

As a warning to the reader we show in Figure 4.28 how Huffman's algorithm does **not** work: It is quite common to forget about the computation of the number of unused leaves in advance, and this will lead to a very inefficient code with the most efficient codeword not being used! ◊

Example 4.53. We consider a random message U with seven symbols having probabilities

$$\begin{aligned} &p_1 = 0.4, \quad p_2 = 0.1, \quad p_3 = 0.1, \quad p_4 = 0.1,\\ &p_5 = 0.1, \quad p_6 = 0.1, \quad p_7 = 0.1, \end{aligned} \tag{4.128}$$

i.e., $\mathrm{H}(U) \approx 2.52$ bits. We firstly design a binary Fano code, see Figure 4.29. The corresponding tree is shown in Figure 4.30. Note that the construction algorithm is not unique in this case: In the second round we could split the second group either to $\{3,4\}$ and $\{5,6,7\}$ or $\{3,4,5\}$ and $\{6,7\}$. Both ways will result in a code of identical performance. The same situation also occurs in the third round.

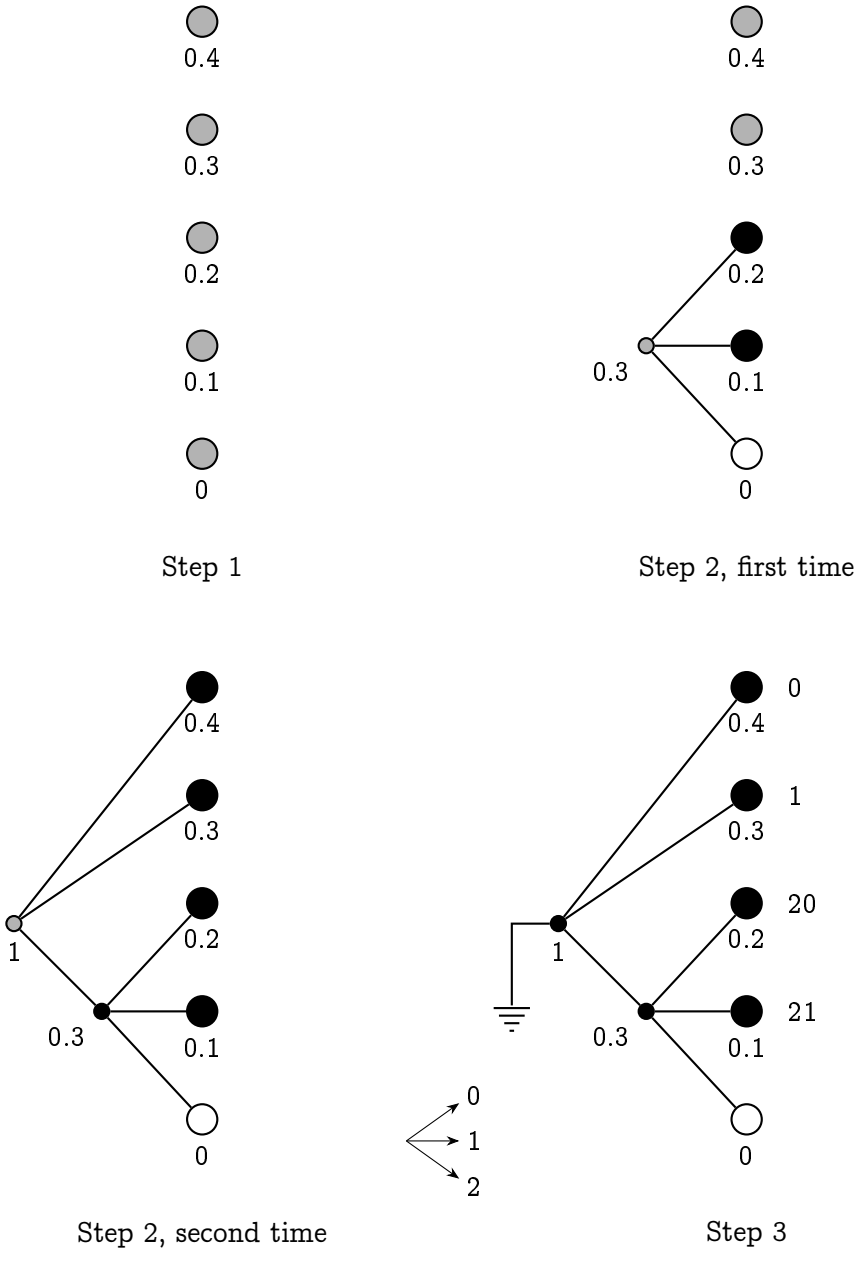


Figure 4.27: Creation of a ternary Huffman code. *Active* nodes and leaves are shaded.

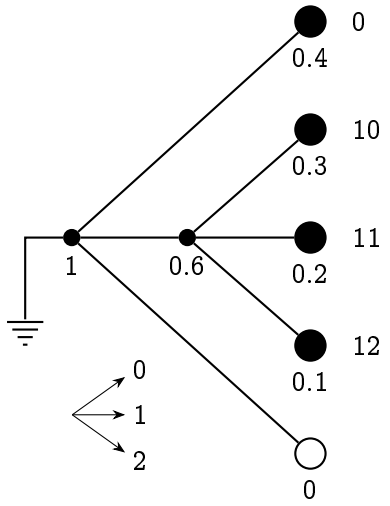


Figure 4.28: An example of a wrong application of Huffman's algorithm. Here it was forgotten to compute the number of unused leaves. We get $\mathsf{E}[L] = 1.6$. Note that one of the shortest (most valuable) codeword 2 is not used!

p_1	p_2	p_3	p_4	p_5	p_6	p_7
0.4	0.1	0.1	0.1	0.1	0.1	0.1
0.5				0.5		
0				**1**		
0.4	0.1	0.1	0.1	0.1	0.1	0.1
		0.2			0.3	
0	**1**	**0**			**1**	
		0.1	0.1	0.1	0.1	0.1
				0.2		0.1
		0	**1**	**0**		**1**
				0.1	0.1	
				0	**1**	
00	**01**	**100**	**101**	**1100**	**1101**	**111**

Figure 4.29: Construction of a binary Fano code for Example 4.53.

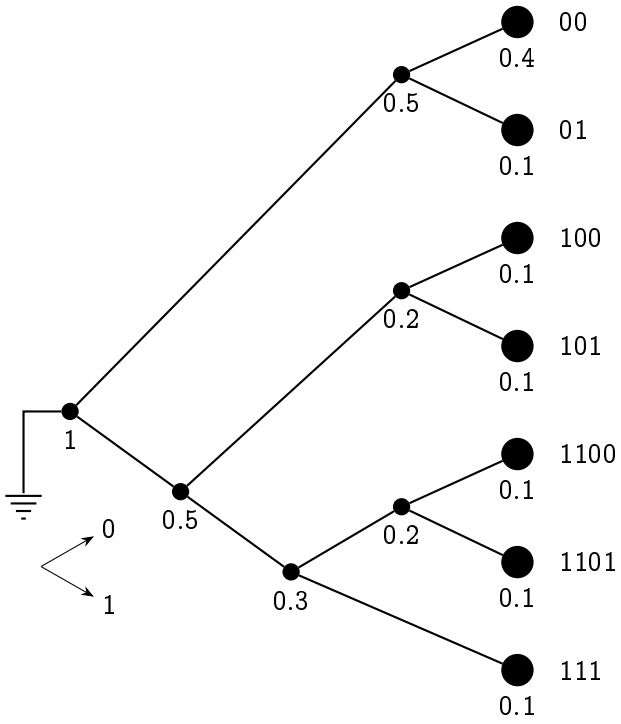


Figure 4.30: A Fano code for the message U of Example 4.53.

The efficiency of this Fano code is

$$\mathsf{E}[L] = 1 + 0.5 + 0.5 + 0.2 + 0.3 + 0.2 = 2.7, \tag{4.129}$$

which satisfies as predicted

$$2.52 \le 2.7 < 3.52. \tag{4.130}$$

Next, we design a binary Shannon code for the same random message. The details are given in Table 4.31. Its performance is less good

$$\mathsf{E}[L] = 0.4 \cdot 2 + 0.6 \cdot 4 = 3.2, \tag{4.131}$$

but must, of course, still satisfy

$$2.52 \le 3.2 < 3.52. \tag{4.132}$$

Finally, a corresponding binary Huffman code for U is shown in Figure 4.32. Its performance is $\mathsf{E}[L] = 2.6$, i.e., it is better than both the Fano and the Shannon code, but it still holds that

$$2.52 \le 2.6 < 3.52. \tag{4.133}$$

$\Diamond$

Table 4.31: The binary Shannon code for the random message U of Example 4.53.

	u_1	u_2	u_3	u_4	u_5	u_6	u_7
p_i	0.4	0.1	0.1	0.1	0.1	0.1	0.1
F_i	0	0.4	0.5	0.6	0.7	0.8	0.9
binary representation	0.0	0.01100	0.1	0.10011	0.10110	0.11001	0.11100
$\left\lceil \log_2 \frac{1}{p_i} \right\rceil$	2	4	4	4	4	4	4
shortened representation	0.00	0.0110	0.1000	0.1001	0.1011	0.1100	0.1110
$\mathbf{c}_i$	00	0110	1000	1001	1011	1100	1110

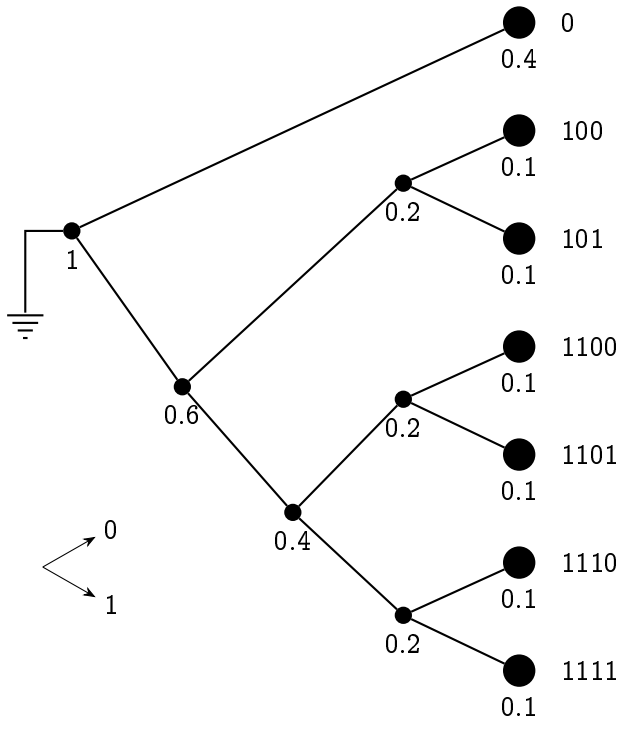


Figure 4.32: A binary Huffman code for the message U of Example 4.53.

Exercise 4.54. *Design binary and ternary Huffman, Shannon, and Fano codes for the random message U with probabilities*

$$\begin{aligned} &p_1 = 0.25, \quad p_2 = 0.2, \quad p_3 = 0.2, \quad p_4 = 0.1,\\ &p_5 = 0.1, \quad p_6 = 0.1, \quad p_7 = 0.05, \end{aligned} \tag{4.134}$$

and compare their performances. ◊

4.10 Types of Codes

Note that in Section 4.1 we have restricted ourselves to prefix-free codes. So, up to now we have only proven that Huffman codes are the optimal codes *under the assumption that we restrict ourselves to prefix-free codes*. We would now like to show that Huffman codes are actually optimal among all useful codes.

Table 4.33: Various codes for a random message with four possible values.

U	Code (i)	Code (ii)	Code (iii)	Code (iv)
a	0	0	10	0
b	0	010	00	10
c	1	01	11	110
d	1	10	110	111

To this end, we need to come back to a more precise definition of "useful codes", i.e., we continue the discussion that we have started in Section 4.1. Let us consider an example with a random message U with four different symbols and let us design various codes for this message as shown in Table 4.33.

We discuss these different codes:

Code (i) is useless because some codewords are used for more than one symbol. Such a code is called *singular*.

Code (ii) is nonsingular. But we have another problem. If we receive 010 we have three different possibilities how to decode it: It could be (010) giving us b, or it could be (0)(10) leading to ad, or it could be (01)(0) corresponding to ca. Even though nonsingular, this code is not *uniquely decodable* and therefore in practice similarly useless as code (i).[10]

Code (iii) is uniquely decodable, even though it is not prefix-free! To see this, note that in order to distinguish between c and d we only need to wait

[10] Note that adding a comma between the codewords is not allowed because in this case we change the code to be *ternary*, i.e., the codewords contain three different letters '0', '1', and ',' instead of only two '0' and '1'.

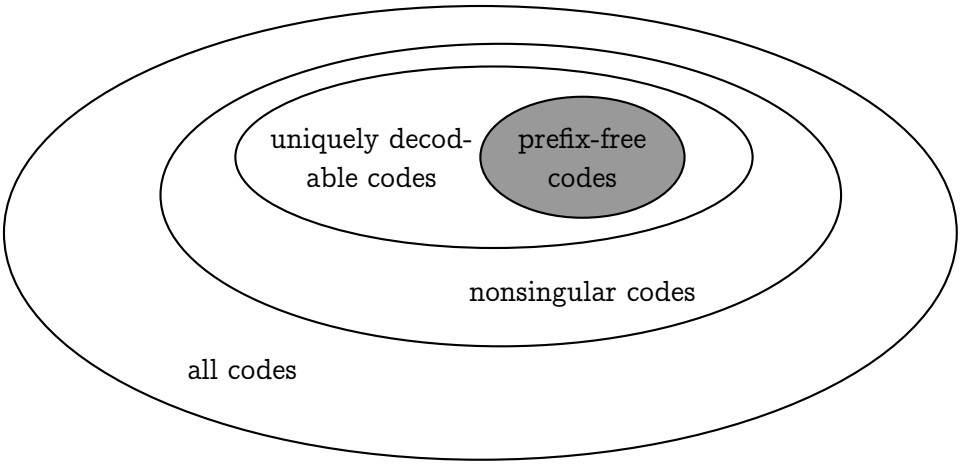


Figure 4.34: Set of all codes.

for the next 1 to show up: If the number of zeros in between is even, we decode 11, otherwise we decode 110. Example:

$$11000010 = (11)(00)(00)(10) \implies cbba, \tag{4.135}$$
$$110000010 = (110)(00)(00)(10) \implies dbba. \tag{4.136}$$

So in a uniquely decodable, but not prefix-free code we may have to delay the decoding until later.

Code (iv) is prefix-free and therefore trivially uniquely decodable.

We see that the set of all possible codes can be grouped as shown in Figure 4.34. We are only interested in the uniquely decodable codes. But so far we have restricted ourselves to prefix-free codes. So the following question arises: Is there a uniquely decodable code that is not prefix-free, but that has a better performance than the best prefix-free code (i.e., the corresponding Huffman code)?

Luckily the answer to this question is no, i.e., the Huffman codes are the best uniquely decodable codes. This can be seen from the following theorem.

Theorem 4.55 (McMillan's Theorem). The codeword lengths l_i of any uniquely decodable code must satisfy the Kraft Inequality

$$\sum_{i=1}^{r} \mathrm{D}^{-l_i} \le 1. \tag{4.137}$$

Why does this help answering our question about the most efficient uniquely decodable code? Well, note that we know from Theorem 4.11 that every prefix-free code also satisfies (4.137). So, for any uniquely decodable, but non-prefix-free code with given codeword lengths, one can find another code with

the same codeword lengths that is prefix-free! But if the codeword lengths are the same, also the performance is identical! Hence, there is no gain in designing a non-prefix-free code.

Proof of Theorem 4.55: Suppose we are given a random message U that takes on r possible values $u \in \mathcal{U}$ (here the set $\mathcal{U}$ denotes the message alphabet). Suppose further that we have a *uniquely decodable* code that assigns to every possible symbol $u \in \mathcal{U}$ a certain codeword of length $l(u)$.

Now choose an arbitrary positive integer ν and design a new code for a vector of ν symbols $\mathbf{u} = (u_1, u_2, \dots, u_\nu) \in \mathcal{U}^\nu = \mathcal{U} \times \cdots \times \mathcal{U}$ by simply *concatenating* the original codewords.

Example 4.56. Consider a binary message with the possible values $u = a$ or $u = b$ ($\mathcal{U} = \{a, b\}$). We design a code for this message with the two codewords $\{01, 1011\}$. Choose $\nu = 3$, i.e., now we have eight possible symbols

$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\} \tag{4.138}$$

and the corresponding codewords are

$$\{010101, 01011011, 01101101, 0110111011, 10110101, \\ 1011011011, 1011101101, 101110111011\}. \tag{4.139}$$

◊

The clue observation now is that because the original code was uniquely decodable, it immediately follows that this new concatenated code also must be uniquely decodable.

Exercise 4.57. *Why? Explain this clue observation!* ◊

The lengths of the new codewords are

$$\tilde{l}(\mathbf{u}) = \sum_{j=1}^{\nu} l(u_j). \tag{4.140}$$

Let $l_{\max}$ be the maximal codeword length of the original code. Then the new code has a maximal codeword length $\tilde{l}_{\max}$ satisfying

$$\tilde{l}_{\max} = \nu l_{\max}. \tag{4.141}$$

We now compute:

$$\left(\sum_{u\in\mathcal{U}} \mathrm{D}^{-l(u)}\right)^{\nu} = \left(\sum_{u_1\in\mathcal{U}} \mathrm{D}^{-l(u_1)}\right)\left(\sum_{u_2\in\mathcal{U}} \mathrm{D}^{-l(u_2)}\right)\cdots\left(\sum_{u_\nu\in\mathcal{U}} \mathrm{D}^{-l(u_\nu)}\right) \tag{4.142}$$

$$= \sum_{u_1\in\mathcal{U}}\sum_{u_2\in\mathcal{U}}\cdots\sum_{u_\nu\in\mathcal{U}} \mathrm{D}^{-l(u_1)}\mathrm{D}^{-l(u_2)}\cdots\mathrm{D}^{-l(u_\nu)} \tag{4.143}$$

$$= \sum_{\mathbf{u}\in\mathcal{U}^\nu} \mathrm{D}^{-l(u_1)-l(u_2)-\cdots-l(u_\nu)} \tag{4.144}$$

$$= \sum_{\mathbf{u}\in\mathcal{U}^\nu} \mathrm{D}^{-\sum_{j=1}^{\nu} l(u_j)} \tag{4.145}$$

$$= \sum_{\mathbf{u}\in\mathcal{U}^\nu} \mathrm{D}^{-\tilde{l}(\mathbf{u})} \tag{4.146}$$

$$= \sum_{m=1}^{\tilde{l}_{\max}} w(m)\,\mathrm{D}^{-m} \tag{4.147}$$

$$= \sum_{m=1}^{\nu l_{\max}} w(m)\,\mathrm{D}^{-m}. \tag{4.148}$$

Here (4.143) follows by writing the exponentiated sum as a product of ν sums; in (4.144) we combine the ν sums over $u_1,\dots,u_\nu$ into one huge sum over the ν-vector $\mathbf{u}$; (4.146) follows from (4.140); in (4.147) we rearrange the order of the terms by collecting all terms with the same exponent together where $w(m)$ counts the number of such terms with equal exponent, i.e., $w(m)$ denotes the number of codewords of length m in the new code; and in the final equality (4.148) we use (4.141).

Now note that since the new concatenated code is uniquely decodable, every codeword of length m is used at most once. In total there are only D^m different sequences of length m, i.e., we know that

$$w(m) \le \mathrm{D}^m. \tag{4.149}$$

Thus,

$$\left(\sum_{u\in\mathcal{U}} \mathrm{D}^{-l(u)}\right)^{\nu} = \sum_{m=1}^{\nu l_{\max}} w(m)\,\mathrm{D}^{-m} \le \sum_{m=1}^{\nu l_{\max}} \mathrm{D}^m \mathrm{D}^{-m} = \nu l_{\max} \tag{4.150}$$

or

$$\sum_{u\in\mathcal{U}} \mathrm{D}^{-l(u)} \le (\nu l_{\max})^{\frac{1}{\nu}}. \tag{4.151}$$

At this stage we are back at an expression that depends only on the original uniquely decodable code. So forget about the trick with the new concatenated code, but simply note that we have shown that for any uniquely decodable code and any positive integer ν, expression (4.151) must hold! Note that we can choose ν freely here!

Note further that for any finite value of $l_{\max}$ we have

$$\lim_{\nu\to\infty} (\nu l_{\max})^{\frac{1}{\nu}} = \lim_{\nu\to\infty} e^{\frac{1}{\nu}\log(\nu l_{\max})} = e^0 = 1. \tag{4.152}$$

Hence, by choosing ν extremely large (i.e., we let ν tend to infinity) we have

$$\sum_{u\in\mathcal{U}} \mathrm{D}^{-l(u)} \le 1 \tag{4.153}$$

as we wanted to prove. □

Since we have now proven that there cannot exist any uniquely decodable code with a better performance than some prefix-free code, it now also follows that the converse of the coding theorem for a single random message (Theorem 4.38), i.e., the lower bound in (4.92), holds for any uniquely decodable code.

4.A Appendix: Alternative Proof for the Converse Part of the Coding Theorem for a Single Random Message

Consider the D-ary tree of *any* D-ary prefix-free code for an r-ary random message U. Let $w_1, \dots, w_n$ be the leaves of this tree, ordered in such a way that w_i is the leaf corresponding to the message u_i, $i = 1, \dots, r$, and $w_{r+1}, \dots, w_n$ are the unused leaves.

Now define two new probability distributions:

$$P_W(w_i) = \begin{cases} P_U(u_i) = p_i & i = 1, \dots, r, \\ 0 & i = r+1, \dots, n \end{cases} \tag{4.154}$$

and

$$P_{\tilde{W}}(w_i) = \mathrm{D}^{-l_i}, \quad i = 1, \dots, n, \tag{4.155}$$

where l_i denotes the depth of leaf w_i. Note that by the Leaf Depth Lemma (Lemma 4.10)

$$\sum_{i=1}^{n} P_{\tilde{W}}(w_i) = \sum_{i=1}^{n} \mathrm{D}^{-l_i} = 1. \tag{4.156}$$

Now compute the relative entropy between these two distributions and use the fact that the relative entropy is always nonnegative (Theorem 3.3):

$$\begin{aligned} 0 \le \mathscr{D}(P_W \,\|\, P_{\tilde{W}}) &= \sum_{i=1}^{r} P_W(w_i) \log \frac{P_W(w_i)}{P_{\tilde{W}}(w_i)} && (4.157)\\ &= \sum_{i=1}^{r} P_W(w_i) \log P_W(w_i) - \sum_{i=1}^{r} P_W(w_i) \log \mathrm{D}^{-l_i} && (4.158)\\ &= \sum_{i=1}^{r} p_i \log p_i + \log \mathrm{D} \sum_{i=1}^{r} p_i l_i && (4.159)\\ &= -\mathrm{H}(U) + \log \mathrm{D} \cdot \mathsf{E}[L]. && (4.160) \end{aligned}$$

Hence,

$$\mathsf{E}[L] \ge \frac{\mathrm{H}(U)}{\log \mathrm{D}}. \tag{4.161}$$

Chapter 5

Data Compression: Efficient Coding of a Memoryless Random Source

So far we have only considered a single random message, but in reality we are much more likely to encounter a situation where we have a *stream* of messages that should be encoded continuously. Luckily we have prepared ourselves for this situation already by considering prefix-free codes only, which make sure that a sequence of codewords can be separated easily into the individual codewords.

5.1 Discrete Memoryless Source

We start with the simpler case where the random source is memoryless, i.e., each symbol that is emitted by the source is independent of all past symbols. A formal definition is given as follows.

Definition 5.1. An r-ary *discrete memoryless source (DMS)* is a device whose output is a sequence of random messages $U_1, U_2, U_3, \ldots$, where

- each U_k can take on r different values with probability[1] $p_1, \ldots, p_r$, and
- the different messages U_k are independent of each other.

Hence, a DMS produces a sequence of *independent and identically distributed (IID)* r-ary random variables.

Note that the DMS is memoryless in the sense that U_k does not depend on the past and also that the distribution of U_k does not change with time.

The obvious way of designing a compression system for such a source is to design a Huffman code for U, continuously use it for each message U_k, and concatenate the codewords together. The receiver can easily separate

[1] We again assume that these probabilities are positive. See Remark 4.3.

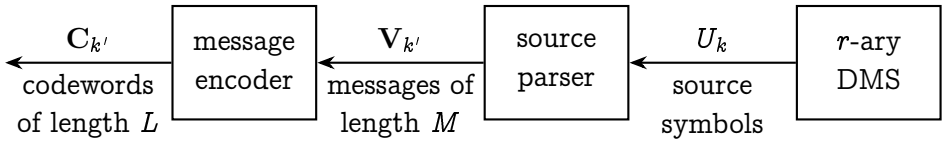


Figure 5.1: A coding scheme for an information source: The source parser groups the source output sequence $\{U_k\}$ into messages $\{\mathbf{V}_{k'}\}$. The message encoder then assigns a codeword $\mathbf{C}_{k'}$ to each possible message $\mathbf{V}_{k'}$.

the codewords (because the Huffman code is prefix-free) and decode them to recover the sequence of messages $\{U_k\}$.

However, the question is whether this is the most efficient approach. Note that it is also possible to combine two or more messages into a new vector message

$$(U_k, U_{k+1}, \dots, U_{k+m-1})$$

and then design a Huffman code for these combined vector messages. We will show below that this approach is actually more efficient.

So we consider an extended compression system as shown in Figure 5.1, where before the message encoder we have added a *source parser*. A source parser is a device that groups several incoming source letters U_k together to a new message $\mathbf{V}_{k'}$. This grouping must be an invertible function in order to make sure that the message can be split back into its components, however, it is possible that the length M of the message $\mathbf{V}_{k'}$ (i.e., the number of letters that are combined to the message $\mathbf{V}_{k'}$) is random: We can easily make it depend on the value of the (random) source output.

Before we consider such variable-length parsers, in the next section we start with the simplest parser that produces messages of constant length: the M-*block parser* that simply always groups exactly M source letters together. As an example assume $\mathrm{M} = 3$:

$$\underbrace{U_1, U_2, U_3}_{\mathbf{V}_1}, \underbrace{U_4, U_5, U_6}_{\mathbf{V}_2}, \underbrace{U_7, U_8, U_9}_{\mathbf{V}_3}, \dots \tag{5.1}$$

Note that since $\{U_k\}$ is IID, also $\{\mathbf{V}_{k'}\}$ is IID. So we can drop the time index and only consider one such block message. Note that if we choose $\mathrm{M} = 1$, we are back at the situation without source parser.

5.2 Block–to–Variable-Length Coding of a DMS

Consider an M-block parser $\mathbf{V} = (U_1, U_2, \dots, U_{\mathrm{M}})$ with M fixed. Then the random message $\mathbf{V}$ takes on r^{M} different values and has the following entropy:

$$\mathrm{H}(\mathbf{V}) = \mathrm{H}(U_1, U_2, \dots, U_{\mathrm{M}}) \tag{5.2}$$

$$= \mathrm{H}(U_1) + \mathrm{H}(U_2|U_1) + \cdots + \mathrm{H}(U_{\mathrm{M}}|U_1, U_2, \ldots, U_{\mathrm{M}-1}) \tag{5.3}$$
$$= \mathrm{H}(U_1) + \mathrm{H}(U_2) + \cdots + \mathrm{H}(U_{\mathrm{M}}) \tag{5.4}$$
$$= \mathrm{M} \cdot \mathrm{H}(U), \tag{5.5}$$

where (5.3) follows from the chain rule; (5.4) from the first I in IID; and (5.5) from the second I and the D in IID.

Now we decide to use an optimal code (i.e., the Huffman code[2]) or at least a good code (e.g., the Shannon or the Fano code) for the message $\mathbf{V}$. From the coding theorem for a single random message (Theorem 4.38) we know that such a code satisfies the following inequality:

$$\frac{\mathrm{H}(\mathbf{V})}{\log \mathrm{D}} \le \mathsf{E}[L] < \frac{\mathrm{H}(\mathbf{V})}{\log \mathrm{D}} + 1. \tag{5.6}$$

Next we note that it is not really fair to compare $\mathsf{E}[L]$ for different values of M because for larger M, $\mathsf{E}[L]$ also will be larger. So, to be correct we should compute the average codeword length necessary to describe *one* source symbol. Since $\mathbf{V}$ contains M source symbols U_k, the correct measure of performance is $\frac{\mathsf{E}[L]}{\mathrm{M}}$.

Hence, we divide the whole expression (5.6) by M:

$$\frac{\mathrm{H}(\mathbf{V})}{\mathrm{M}\log \mathrm{D}} \le \frac{\mathsf{E}[L]}{\mathrm{M}} < \frac{\mathrm{H}(\mathbf{V})}{\mathrm{M}\log \mathrm{D}} + \frac{1}{\mathrm{M}} \tag{5.7}$$

and make use of (5.5):

$$\frac{\mathrm{M}\,\mathrm{H}(U)}{\mathrm{M}\log \mathrm{D}} \le \frac{\mathsf{E}[L]}{\mathrm{M}} < \frac{\mathrm{M}\,\mathrm{H}(U)}{\mathrm{M}\log \mathrm{D}} + \frac{1}{\mathrm{M}} \tag{5.8}$$

i.e.,

$$\frac{\mathrm{H}(U)}{\log \mathrm{D}} \le \frac{\mathsf{E}[L]}{\mathrm{M}} < \frac{\mathrm{H}(U)}{\log \mathrm{D}} + \frac{1}{\mathrm{M}}. \tag{5.9}$$

We have proven the following important result, also known as Shannon's *source coding theorem*.

Theorem 5.2 (Block–to–Variable-Length Coding Theorem for a DMS).
There exists a D-ary prefix-free code of an M-block message from a DMS such that the average number of D-ary code digits per source letter satisfies:

$$\frac{\mathsf{E}[L]}{\mathrm{M}} < \frac{\mathrm{H}(U)}{\log \mathrm{D}} + \frac{1}{\mathrm{M}} \tag{5.10}$$

where $\mathrm{H}(U)$ is the entropy of a single source letter.

Conversely, for **every** uniquely decodable D-ary code of an M-block

[2]Note that since we now design the Huffman code for the M-block message $\mathbf{V}$, in Huffman's algorithm we have to replace r by r^{M}; see, e.g., (4.125).

message it must be true that

$$\frac{\mathrm{E}[L]}{\mathrm{M}} \geq \frac{\mathrm{H}(U)}{\log \mathrm{D}}. \tag{5.11}$$

We would like to briefly discuss this result. The main point to note here is that by choosing M large enough, we can approach the lower bound $\frac{\mathrm{H}(U)}{\log \mathrm{D}}$ *arbitrarily closely* when using a Huffman, a Shannon-type or a Fano code. Hence, the entropy $\mathrm{H}(U)$ is the ultimate limit of compression and precisely describes the amount of information that is packed in the output of the discrete memoryless source $\{U_k\}$! In other words we can compress any DMS to $\mathrm{H}(U)$ bits (entropy measured in bits) or $\frac{\mathrm{H}(U)}{\log \mathrm{D}}$ D-ary code letters on average, but not less. This is the first real justification of the usefulness of the definition of entropy.

We also see that in the end it does not make a big difference whether we use a Huffman code or any of the suboptimal good codes (Shannon-type, Shannon, or Fano codes) as all approach the ultimate limit for M large enough.

On the other hand, note the price we have to pay: By making M large we not only increase the number of possible messages and thereby make the code complicated, but we also introduce *delay* in the system, because the encoder can only encode the message *after it has received the complete block of* M *source symbols*! Basically the more closely we want to approach the ultimate limit of entropy, the larger is our potential delay in the system.

5.3 Arithmetic Coding

5.3.1 Introduction

In Section 5.2 we have described a general coding scheme for a DMS that is based on an M-block parser and a Huffman code (or similar) for the corresponding block messages. We have shown that this system works very efficiently and achieves a compression rate that is less than $\frac{1}{\mathrm{M}}$ of a code digit per source letter away from the theoretical minimum.

Unfortunately, the complexity of such a system very quickly becomes completely infeasible: Imagine for a moment a ternary ($r = 3$) DMS $\{U_k\}$ taking value in $\{a, b, c\}$ whose output is parsed into blocks of length $\mathrm{M} = 1000$. This will result in $3^{1000} \approx 10^{477}$ different possible source sequences! These sequences firstly need to be sorted according to their probabilities, and then for every one of them the corresponding Huffman codeword needs to be designed and stored. No whatever large and fast computer is capable of doing this.

Moreover, even if we could set up the system, we would not be happy with it because the system always needs to wait until the complete source sequence

of length $M = 1000$ has arrived at the encoder before the corresponding codeword can be found. So, we have long delays in the system.

Therefore the question arises if there exists a practical system that does not suffer from these complexity and delay issues, but still performs similarly efficient as shown in Theorem 5.2. And indeed, such systems do exist. One very beautiful and elegant such coding scheme is the so-called *arithmetic coding*. In its basic principle it is a generalization of the Shannon code to a DMS, however, it introduces some tweaks that allow to avoid complexity and delay.

5.3.2 Encoding

Recall that a Shannon code firstly orders all possible messages according to probability and then uses the D-ary representation of the cumulative probability $\mathrm{F}_j \triangleq \sum_{j'=1}^{j-1} p_{j'}$ to generate the corresponding codeword.

Thus, for the ordering of the messages, we first need to compute the probability of *every* possible message, which is infeasible. Arithmetic coding avoids this pitfall by ordering the sequences according to alphabet and not probability (*lexicographic ordering*). For example, for the ternary source $\{U_k\}$ mentioned above, we have the following order:

$$\begin{aligned}
\mathbf{u}_1 &= aa\ldots aaa\\
\mathbf{u}_2 &= aa\ldots aab\\
\mathbf{u}_3 &= aa\ldots aac\\
\mathbf{u}_4 &= aa\ldots aba\\
\mathbf{u}_5 &= aa\ldots abb\\
&\vdots\\
\mathbf{u}_{3^M-2} &= cc\ldots cca\\
\mathbf{u}_{3^M-1} &= cc\ldots ccb\\
\mathbf{u}_{3^M} &= cc\ldots ccc
\end{aligned}$$

Thus the cumulative probability $\mathrm{F}_{\mathbf{u}_j}$ will now be computed according to this alphabetical order:

$$\mathrm{F}_{\mathbf{u}_j} \triangleq \sum_{j'=1}^{j-1} p_{\mathbf{u}_{j'}}, \quad j = 1,\ldots,r^M, \tag{5.12}$$

where $p_{\mathbf{u}_j}$ is the probability of the output sequence $\mathbf{u}_j$:

$$p_{\mathbf{u}_j} \triangleq \prod_{k=1}^{M} P_U(u_{j,k}), \quad j = 1,\ldots,r^M. \tag{5.13}$$

So far this does not seem to help much because it still looks as if we need to compute $p_{\mathbf{u}_{j'}}$ for all $\mathbf{u}_{j'}$ that lie alphabetically before $\mathbf{u}_j$. However, this

is not the case: It turns out that because of the alphabetical ordering, for a any given output sequence $\mathbf{u}$ one can compute $\mathrm{F}_{\mathbf{u}}$ directly without having to consider any other sequences at all!

The trick lies in an iterative procedure using the components of $\mathbf{u}$. To understand this, we return to our ternary example from above, look at $\mathbf{u} = bbac$ and assume that we have computed F_{bba} already. It is now easy to see that we can derive F_{bbac} directly from F_{bba} and p_{bba}:

$$\begin{aligned}
\mathrm{F}_{bbac} &= \Pr(\{aaaa, \dots, bbab\}) && (5.14)\\
&= \Pr(\text{all length-4 seq. alphabetically before } bba_) \\
&\quad + \Pr(\{bbaa, bbab\}) && (5.15)\\
&= (p_{aaaa} + \cdots + p_{aaac}) + (p_{aaba} + \cdots + p_{aabc}) + \cdots \\
&\quad + (p_{baca} + \cdots + p_{bacc}) + \Pr(\{bbaa, bbab\}) && (5.16)\\
&= p_{aaa} \cdot \underbrace{(p_a + \cdots + p_c)}_{=1} + p_{aab} \cdot \underbrace{(p_a + \cdots + p_c)}_{=1} + \cdots \\
&\quad + p_{bac} \cdot \underbrace{(p_a + \cdots + p_c)}_{=1} + \Pr(\{bbaa, bbab\}) && (5.17)\\
&= \underbrace{p_{aaa} + p_{aab} + \cdots + p_{bac}}_{=\mathrm{F}_{bba}} + \Pr(\{bbaa, bbab\}) && (5.18)\\
&= \mathrm{F}_{bba} + p_{bba} \cdot \underbrace{(p_a + p_b)}_{=\mathrm{F}_c} && (5.19)\\
&= \mathrm{F}_{bba} + p_{bba} \cdot \mathrm{F}_c. && (5.20)
\end{aligned}$$

Hence, we see that we can compute the cumulative probability directly by iteratively updating the values of F and p with the current output of the DMS: For every output u_k ($k = 1, \dots, \mathrm{M}$) of the DMS we compute

$$\begin{aligned}
p_{u_1,\dots,u_k} &= p_{u_1,\dots,u_{k-1}} \cdot p_{u_k}, && (5.21)\\
\mathrm{F}_{u_1,\dots,u_k} &= \mathrm{F}_{u_1,\dots,u_{k-1}} + p_{u_1,\dots,u_{k-1}} \cdot \mathrm{F}_{u_k}. && (5.22)
\end{aligned}$$

Note that this not only enables us to compute $\mathrm{F}_{\mathbf{u}}$ directly, but we can do the computation immediately whenever a new source symbol becomes available, i.e., we do not suffer from any delays.

Unfortunately, if we now use the D-ary representation of $\mathrm{F}_{\mathbf{u}}$ and truncate it to $l_{\mathbf{u}} = \left\lceil \log_{\mathrm{D}} \frac{1}{p_{\mathbf{u}}} \right\rceil$ to get the codeword for $\mathbf{u}$, then the code will not be prefix-free anymore. (This happens because we do no longer have the property that $l_{\mathbf{u}_j}$ is increasing with j, since the corresponding $p_{\mathbf{u}_j}$ are not sorted.)

To solve this problem, the second main idea of arithmetic coding is the insight that we only need to make the codewords slightly longer,

$$\tilde{l}_{\mathbf{u}} \triangleq \left\lceil \log_{\mathrm{D}} \frac{1}{p_{\mathbf{u}}} \right\rceil + 1, \tag{5.23}$$

and that instead of using the $\tilde{l}_{\mathbf{u}}$-truncated D-ary representation of $\mathsf{F}_{\mathbf{u}}$, we need to use the $\tilde{l}_{\mathbf{u}}$-truncated D-ary representation of

$$\tilde{\mathsf{F}}_{\mathbf{u}} \triangleq \mathsf{F}_{\mathbf{u}} + \mathrm{D}^{-\tilde{l}_{\mathbf{u}}}. \tag{5.24}$$

(Note that this basically means that instead of rounding down (truncating) $\mathsf{F}_{\mathbf{u}}$, we are rounding it up; see the discussion in the proof of Lemma 5.3 below.)

To shed some more light on this, it is best to investigate why this code indeed is prefix-free.

Lemma 5.3 (The Arithmetic Code is Prefix-Free). The set of codewords defined as $\tilde{l}_{\mathbf{u}}$-truncated D-ary representations of $\tilde{\mathsf{F}}_{\mathbf{u}}$ (where $\tilde{l}_{\mathbf{u}}$ is defined in (5.23) and where $\tilde{\mathsf{F}}_{\mathbf{u}}$ is defined in (5.12) and (5.24)) is prefix-free.

Proof: To start we note that the operation of truncating a fraction f at the l-th position can be written mathematically as follows:

$$\lfloor f \cdot 10^{l} \rfloor \cdot 10^{-l}. \tag{5.25}$$

In general, when instead of a decimal fraction we consider the D-ary representation of f, the base 10 must be changed to base D:

$$\lfloor f \cdot \mathrm{D}^{l} \rfloor \cdot \mathrm{D}^{-l}. \tag{5.26}$$

Thus, the codewords of arithmetic coding are defined as

$$0.\mathbf{c}_{\mathbf{u}} \triangleq \lfloor \tilde{\mathsf{F}}_{\mathbf{u}} \cdot \mathrm{D}^{\tilde{l}_{\mathbf{u}}} \rfloor \cdot \mathrm{D}^{-\tilde{l}_{\mathbf{u}}}. \tag{5.27}$$

Now we associate to every M-block message $\mathbf{u}$ an interval:

$$\big(\mathsf{F}_{\mathbf{u}}, \mathsf{F}_{\mathbf{u}} + p_{\mathbf{u}}\big]. \tag{5.28}$$

Note that these intervals are disjoint and their union is $(0, 1]$.

We also associate an interval to each codeword $\mathbf{c}_{\mathbf{u}}$ of length $\tilde{l}_{\mathbf{u}}$:

$$\Big[0.\mathbf{c}_{\mathbf{u}}, 0.\mathbf{c}_{\mathbf{u}} + \mathrm{D}^{-\tilde{l}_{\mathbf{u}}}\Big). \tag{5.29}$$

(Note that here we write the numbers in D-ary format.) This interval contains all sequences $0.\mathbf{c}$ that have the first $\tilde{l}_{\mathbf{u}}$ digits identical to $0.\mathbf{c}_{\mathbf{u}}$. Hence, to show that arithmetic codes are prefix-free, it is sufficient to show that no two such codeword intervals overlap. This is indeed the case because each codeword interval is actually contained in the corresponding (disjoint!) source message interval (5.28), as can be seen as follows:

$$\begin{aligned}
0.\mathbf{c}_{\mathbf{u}} &= \lfloor \tilde{\mathsf{F}}_{\mathbf{u}} \cdot \mathrm{D}^{\tilde{l}_{\mathbf{u}}} \rfloor \cdot \mathrm{D}^{-\tilde{l}_{\mathbf{u}}} && \text{(by (5.27))} && (5.30)\\
&= \Big\lfloor (\mathsf{F}_{\mathbf{u}} + \mathrm{D}^{-\tilde{l}_{\mathbf{u}}}) \cdot \mathrm{D}^{\tilde{l}_{\mathbf{u}}} \Big\rfloor \cdot \mathrm{D}^{-\tilde{l}_{\mathbf{u}}} && \text{(by (5.24))} && (5.31)\\
&= \lfloor \mathsf{F}_{\mathbf{u}} \cdot \mathrm{D}^{\tilde{l}_{\mathbf{u}}} + 1 \rfloor \cdot \mathrm{D}^{-\tilde{l}_{\mathbf{u}}} && && (5.32)\\
&> (\mathsf{F}_{\mathbf{u}} \cdot \mathrm{D}^{\tilde{l}_{\mathbf{u}}}) \cdot \mathrm{D}^{-\tilde{l}_{\mathbf{u}}} && \text{(because } \lfloor x+1 \rfloor > x\text{)} && (5.33)\\
&= \mathsf{F}_{\mathbf{u}} && && (5.34)
\end{aligned}$$

and

$$\begin{aligned}
0.\mathbf{c_u} + \mathrm{D}^{-\tilde{l}_\mathbf{u}} &= \lfloor \tilde{\mathrm{F}}_\mathbf{u} \cdot \mathrm{D}^{\tilde{l}_\mathbf{u}} \rfloor \cdot \mathrm{D}^{-\tilde{l}_\mathbf{u}} + \mathrm{D}^{-\tilde{l}_\mathbf{u}} && \text{(by (5.27))} && (5.35)\\
&\le \tilde{\mathrm{F}}_\mathbf{u} \cdot \mathrm{D}^{\tilde{l}_\mathbf{u}} \cdot \mathrm{D}^{-\tilde{l}_\mathbf{u}} + \mathrm{D}^{-\tilde{l}_\mathbf{u}} && && (5.36)\\
&= \tilde{\mathrm{F}}_\mathbf{u} + \mathrm{D}^{-\tilde{l}_\mathbf{u}} && && (5.37)\\
&= \mathrm{F}_\mathbf{u} + \mathrm{D}^{-\tilde{l}_\mathbf{u}} + \mathrm{D}^{-\tilde{l}_\mathbf{u}} && \text{(by (5.24))} && (5.38)\\
&= \mathrm{F}_\mathbf{u} + 2\mathrm{D}^{-\tilde{l}_\mathbf{u}} && && (5.39)\\
&\le \mathrm{F}_\mathbf{u} + \mathrm{D} \cdot \mathrm{D}^{-\tilde{l}_\mathbf{u}} && \text{(because } \mathrm{D} \ge 2) && (5.40)\\
&= \mathrm{F}_\mathbf{u} + \mathrm{D}^{1-\tilde{l}_\mathbf{u}} && && (5.41)\\
&= \mathrm{F}_\mathbf{u} + \mathrm{D}^{-\left\lceil \log_\mathrm{D} \frac{1}{p_\mathbf{u}} \right\rceil} && \text{(by (5.23))} && (5.42)\\
&\le \mathrm{F}_\mathbf{u} + \mathrm{D}^{-\log_\mathrm{D} \frac{1}{p_\mathbf{u}}} && && (5.43)\\
&= \mathrm{F}_\mathbf{u} + p_\mathbf{u}. && && (5.44)
\end{aligned}$$

Thus,

$$\left[0.\mathbf{c_u}, 0.\mathbf{c_u} + \mathrm{D}^{-\tilde{l}_\mathbf{u}}\right) \subset \left(\mathrm{F}_\mathbf{u}, \mathrm{F}_\mathbf{u} + p_\mathbf{u}\right] \tag{5.45}$$

as claimed. Therefore, no codeword can be a prefix of another codeword.[3] □

So we see that by the sacrifice of one additional codeword digit we can get a very efficient encoding algorithm that can compress the outcome of a memoryless source on the fly.

Before we summarize this algorithm and analyze its efficiency, let us go through a small example.

Example 5.4. We again assume a ternary DMS $\{U_k\}$ that emits an independent and identically distributed (IID) sequence of ternary random messages U_k with the following probabilities:

$$P_U(a) = 0.5, \quad P_U(b) = 0.3, \quad P_U(c) = 0.2. \tag{5.46}$$

Now we compute the binary ($\mathrm{D} = 2$) codeword of the length-10 ($\mathrm{M} = 10$) output sequence $\mathbf{u} = baabcabbba$: For the original Shannon code we would have to compute the probabilities of each of the $3^{10} = 59049$ different possible output sequences and then order them according to their probabilities. This needs a large amount of time and storage place.

For arithmetic coding, we do not order the sequences and do not need to bother with the probability of any other sequence, but we can directly compute the probability $p_{baabcabbba}$ and the cumulative probability $\mathrm{F}_{baabcabbba}$

[3] One could again require that no D-ary representations with an infinite tail of symbols $(\mathrm{D}-1)$ are allowed. Strictly speaking, however, this is not necessary because we take care of the ambiguity of the D-ary representation by adding the term $\mathrm{D}^{-\tilde{l}_{\mathbf{u}_j}}$ in (5.24).

of the given sequence recursively as follows:

$$\begin{aligned}
p_b &= P_U(b) && = 0.3, && (5.47a)\\
p_{ba} &= p_b \cdot P_U(a) && = 0.3 \cdot 0.5 && = 0.15, && (5.47b)\\
p_{baa} &= p_{ba} \cdot P_U(a) && = 0.15 \cdot 0.5 && = 0.075, && (5.47c)\\
p_{baab} &= p_{baa} \cdot P_U(b) && = 0.075 \cdot 0.3 && = 0.0225, && (5.47d)\\
p_{baabc} &= p_{baab} \cdot P_U(c) && = 0.0225 \cdot 0.2 && = 0.0045, && (5.47e)\\
p_{baabca} &= p_{baabc} \cdot P_U(a) && = 0.0045 \cdot 0.5 && = 0.00225, && (5.47f)\\
p_{baabcab} &= p_{baabca} \cdot P_U(b) && = 0.00225 \cdot 0.3 && = 0.000675, && (5.47g)\\
p_{baabcabb} &= p_{baabcab} \cdot P_U(b) && = 0.000675 \cdot 0.3 && = 0.0002025, && (5.47h)\\
p_{baabcabbb} &= p_{baabcabb} \cdot P_U(b) && = 0.0002025 \cdot 0.3 && = 0.00006075, && (5.47i)\\
p_{baabcabbba} &= p_{baabcabb} \cdot P_U(a) && = 0.00006075 \cdot 0.5 && = 0.000030375, && (5.47j)
\end{aligned}$$

and

$$\begin{aligned}
\mathsf{F}_b &= f_b && = 0.5, && (5.48a)\\
\mathsf{F}_{ba} &= \mathsf{F}_b + p_b \cdot f_a && = 0.5 + 0.3 \cdot 0 && \\
& && = 0.5, && (5.48b)\\
\mathsf{F}_{baa} &= \mathsf{F}_{ba} + p_{ba} \cdot f_a && = 0.5 + 0.15 \cdot 0 && \\
& && = 0.5, && (5.48c)\\
\mathsf{F}_{baab} &= \mathsf{F}_{baa} + p_{baa} \cdot f_b && = 0.5 + 0.075 \cdot 0.5 && \\
& && = 0.5375, && (5.48d)\\
\mathsf{F}_{baabc} &= \mathsf{F}_{baab} + p_{baab} \cdot f_c && = 0.5375 + 0.0225 \cdot 0.8 && \\
& && = 0.5555, && (5.48e)\\
\mathsf{F}_{baabca} &= \mathsf{F}_{baabc} + p_{baabc} \cdot f_a && = 0.5555 + 0.0045 \cdot 0 && \\
& && = 0.5555, && (5.48f)\\
\mathsf{F}_{baabcab} &= \mathsf{F}_{baabca} + p_{baabca} \cdot f_b && = 0.5555 + 0.00225 \cdot 0.5 && \\
& && = 0.556625, && (5.48g)\\
\mathsf{F}_{baabcabb} &= \mathsf{F}_{baabcab} + p_{baabcab} \cdot f_b && = 0.556625 + 0.000675 \cdot 0.5 && \\
& && = 0.5569625, && (5.48h)\\
\mathsf{F}_{baabcabbb} &= \mathsf{F}_{baabcabb} + p_{baabcabb} \cdot f_b && = 0.5569625 + 0.0002025 \cdot 0.5 && \\
& && = 0.55706375, && (5.48i)\\
\mathsf{F}_{baabcabbba} &= \mathsf{F}_{baabcabbb} + p_{baabcabbb} \cdot f_a && = 0.55706375 + 0.00006075 \cdot 0 && \\
& && = 0.55706375. && (5.48j)
\end{aligned}$$

Here instead of F_a, F_b, and F_c we have introduced $f_a \triangleq 0$, $f_b \triangleq p_a = 0.5$, and $f_c \triangleq p_a + p_b = 0.8$ to distinguish them better from the $\mathsf{F}_{\mathbf{u}}$ that is computed recursively.

Now we only need to compute the required length

$$\tilde{l}_{baabcabbba} = \left\lceil \log_{\mathrm{D}} \frac{1}{p_{baabcabbba}} \right\rceil + 1 = \left\lceil \log_2 \frac{1}{0.000030375} \right\rceil + 1 = 17, \quad (5.49)$$

find the binary representation of the shifted cumulative probability

$$\tilde{\mathrm{F}}_{baabcabbba} = 0.55706375 + 2^{-17} = 0.557071379394531192 \quad (5.50)$$

and shorten it to length 17. Note that it is actually easier to change the order: Firstly find the binary representation of $\mathrm{F}_{baabcabbba} = 0.55706375$, shorten it to 17, and then add 2^{-17}:

$$(0.55706375)_{10} = (0.10001110100110111\ldots)_2 \quad (5.51a)$$
$$(0.10001110100110111)_2 + 2^{-17} = (0.10001110100111000)_2. \quad (5.51b)$$

This results in the codeword 10001110100111000. ◊

We summarize the encoding procedure of arithmetic coding in the following algorithmic form.

D-ary **Arithmetic Encoding of a DMS**:

Step 1: Fix a blocklength M and decide on an alphabetical order on the source alphabet $\mathcal{U} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$. Then define

$$f_{\alpha_i} \triangleq \sum_{i'=1}^{i-1} P_U(\alpha_{i'}), \quad i = 1, \ldots, r, \quad (5.52)$$

and set $k \triangleq 1$, $p_0 \triangleq 1$, $\mathrm{F}_0 \triangleq 0$.

Step 2: Observe the kth source output letter u_k of the DMS and compute

$$p_k \triangleq p_{k-1} \cdot P_U(u_k), \quad (5.53)$$
$$\mathrm{F}_k \triangleq \mathrm{F}_{k-1} + p_{k-1} \cdot f_{u_k}. \quad (5.54)$$

Step 3: If $k < \mathrm{M}$, $k \to k+1$ and go to Step 2. Otherwise, compute

$$\tilde{l} \triangleq \left\lceil \log_{\mathrm{D}} \frac{1}{p_{\mathrm{M}}} \right\rceil + 1, \quad (5.55)$$

compute the D-ary representation of F_{M}, shorten it to length $\tilde{l}$, and add $\mathrm{D}^{-\tilde{l}}$ to it. Then put out the (length-$\tilde{l}$) D-ary sequence after the comma as codeword.

It remains to derive the performance. This is straightforward: Using (5.23) we get

$$\tilde{l}_{\mathbf{u}} = \left\lceil \log_{\mathrm{D}} \frac{1}{p_{\mathbf{u}}} \right\rceil + 1 < \frac{\log \frac{1}{p_{\mathbf{u}}}}{\log \mathrm{D}} + 2 \tag{5.56}$$

and therefore

$$\mathsf{E}[L] < \frac{\mathsf{E}\left[\log \frac{1}{p_{\mathbf{U}}}\right]}{\log \mathrm{D}} + 2 = \frac{\mathrm{H}(\mathbf{U})}{\log \mathrm{D}} + 2 = \frac{\mathrm{M}\,\mathrm{H}(U)}{\log \mathrm{D}} + 2, \tag{5.57}$$

i.e.,

$$\frac{\mathsf{E}[L]}{\mathrm{M}} < \frac{\mathrm{H}(U)}{\log \mathrm{D}} + \frac{2}{\mathrm{M}}. \tag{5.58}$$

Theorem 5.5 (Coding Theorem for Arithmetic Coding).
The average number of D-ary code digits per source letter of an arithmetic code satisfies

$$\frac{\mathrm{H}(U)}{\log \mathrm{D}} \le \frac{\mathsf{E}[L]}{\mathrm{M}} < \frac{\mathrm{H}(U)}{\log \mathrm{D}} + \frac{2}{\mathrm{M}}. \tag{5.59}$$

This is almost as good as in Theorem 5.2 that is based on Shannon, Fano, or Huffman coding, and it is asymptotically optimal! Note that both Shannon and Huffman coding are completely infeasible for large message lengths, and also Fano coding will quickly get to its limitations.

5.3.3 Decoding

Finally, we need to think about the decoding. Recall that one of the main points of arithmetic coding is that we do not want (and also not need) to keep a complete list of all codewords. So, without knowing how the codewords look like, how do we decode? Moreover, even though we know that the code is prefix-free, we do not even know where a codeword ends and the next starts. So we face quite a challenge here! Luckily, both problems can be solved in a kind of reverse process of the encoding.

In the following we assume that the (in alphabetical order) jth sequence $\mathbf{u}_j$ was encoded and that the first source symbol was α_i, i.e., $u_1 = \alpha_i$, and the second was $u_2 = \alpha'_i$.

Let us pretend for the moment the decoder knew the exact value of $\mathrm{F}_{\mathbf{u}_j}$ (instead of the truncated version of $\tilde{\mathrm{F}}_{\mathbf{u}_j}$). From (5.54) we know that

$$\mathrm{F}_{\mathbf{u}_j} = f_{u_1} + P_U(u_1) f_{u_2} + P_U(u_1)P_U(u_2) f_{u_3} + \cdots + P_U(u_1)\cdots P_U(u_{\mathrm{M}-1}) f_{u_{\mathrm{M}}}. \tag{5.60}$$

We see that $\mathsf{F}_{\mathbf{u}_j} \geq f_{u_1} = f_{\alpha_i}$. On the other hand, we must have $\mathsf{F}_{\mathbf{u}_j} < f_{\alpha_i} + P_U(\alpha_i) = f_{\alpha_{i+1}}$ because otherwise there would exist another source sequence starting with α_{i+1} instead of α_i that would result in the same $\mathsf{F}_{\mathbf{u}_j}$, contradicting the fact that our code is prefix-free.

So we see that we can gain f_{u_1} back from $\mathsf{F}_{\mathbf{u}_j}$: We simply need to search for the largest f_{α_i} that is smaller or equal to $\mathsf{F}_{\mathbf{u}_j}$. Once we have f_{u_1} and thus u_1, we can get rid of the influence of u_1:

$$\mathsf{F}_{\mathbf{u}'} = \frac{\mathsf{F}_{\mathbf{u}_j} - f_{u_1}}{P_U(u_1)} = f_{u_2} + P_U(u_2) f_{u_3} + \cdots + P_U(u_2)\cdots P_U(u_{\mathrm{M}-1}) f_{u_{\mathrm{M}}}, \quad (5.61)$$

where we write $\mathbf{u}' = (u_2, \ldots, u_{\mathrm{M}})$. This looks exactly as if we had only encoded the sequence $\mathbf{u}'$! Thus, using the same argumentation again, we know that

$$f_{\alpha_i'} \leq \mathsf{F}_{\mathbf{u}'} < f_{\alpha_i'} + P_U(\alpha_i') = f_{\alpha_{i+1}'} \quad (5.62)$$

and that we can decode u_2 the same way. Hence, we can continue this way and recursively gain back the complete sequence $\mathbf{u}_j$.

Now unfortunately, the decoder does not know $\mathsf{F}_{\mathbf{u}_j}$, but only $0.\mathbf{c}_{\mathbf{u}_j}$, which is a truncated version of $\tilde{\mathsf{F}}_{\mathbf{u}_j}$:

$$0.\mathbf{c}_{\mathbf{u}_j} = \mathsf{F}_{\mathbf{u}_j} + \mathrm{D}^{-\tilde{l}_{\mathbf{u}_j}} - \epsilon_{\mathbf{u}_j}, \quad (5.63)$$

where $\epsilon_{\mathbf{u}_j} \geq 0$ denotes the fraction that is lost by the truncation. Luckily, this is not a problem because we know from (5.45) that

$$\mathsf{F}_{\mathbf{u}_j} < 0.\mathbf{c}_{\mathbf{u}_j} < \mathsf{F}_{\mathbf{u}_{j+1}} \quad (5.64)$$

and therefore

$$f_{\alpha_i} \leq \mathsf{F}_{\mathbf{u}_j} < 0.\mathbf{c}_{\mathbf{u}_j} < \mathsf{F}_{\mathbf{u}_{j+1}} \leq f_{\alpha_{i+1}}. \quad (5.65)$$

(Here the last inequality follows again because we have proven that the code is prefix-free: were $\mathsf{F}_{\mathbf{u}_{j+1}} > f_{\alpha_{i+1}}$, then the codeword intervals belonging to some sequence starting with α_i and to some sequence starting with α_{i+1} would overlap, leading to prefixes.)

So we see that the truncated version of $\tilde{\mathsf{F}}_{\mathbf{u}_j}$ still lies within the same boundaries as $\mathsf{F}_{\mathbf{u}_j}$ and we can find f_{u_1} as the largest f_{α_i} that is smaller or equal to $0.\mathbf{c}_{\mathbf{u}_j}$, thereby gaining back $u_1 = \alpha_i$. Then we would like to remove the influence of u_1 and define

$$0.\mathbf{c}_{\mathbf{u}'} \triangleq \frac{0.\mathbf{c}_{\mathbf{u}_j} - f_{u_1}}{P_U(u_1)}. \quad (5.66)$$

Note that

$$\frac{0.\mathbf{c}_{\mathbf{u}_j} - f_{u_1}}{P_U(u_1)} = \frac{\mathsf{F}_{\mathbf{u}_j} + \mathrm{D}^{-\tilde{l}_{\mathbf{u}_j}} - \epsilon_{\mathbf{u}_j} - f_{u_1}}{P_U(u_1)} \tag{5.67}$$

$$= \mathsf{F}_{\mathbf{u}'} + \frac{\mathrm{D}^{-\left\lceil \log_{\mathrm{D}} \frac{1}{p_{\mathbf{u}_j}} \right\rceil - 1}}{P_U(u_1)} - \frac{\epsilon_{\mathbf{u}_j}}{P_U(u_1)} \tag{5.68}$$

$$\le \mathsf{F}_{\mathbf{u}'} + \frac{\mathrm{D}^{\log_{\mathrm{D}} p_{\mathbf{u}_j} - 1}}{P_U(u_1)} \tag{5.69}$$

$$\le \mathsf{F}_{\mathbf{u}'} + \frac{p_{\mathbf{u}_j}}{P_U(u_1)} \tag{5.70}$$

$$= \mathsf{F}_{\mathbf{u}'} + \frac{P_U(u_1)\cdots P_U(u_{\mathrm{M}})}{P_U(u_1)} \tag{5.71}$$

$$= \mathsf{F}_{\mathbf{u}'} + P_U(u_2)\cdots P_U(u_{\mathrm{M}}) \tag{5.72}$$

$$= \mathsf{F}_{\mathbf{u}'} + p_{\mathbf{u}'}, \tag{5.73}$$

where (5.67) follows from (5.63); (5.68) holds because of (5.61) and (5.23); and the Inequality (5.69) follows from dropping the ceiling operation and because $\epsilon_{\mathbf{u}_j} \ge 0$.

On the other hand, from (5.65) and (5.61) it can be seen that

$$\frac{0.\mathbf{c}_{\mathbf{u}_j} - f_{u_1}}{P_U(u_1)} > \frac{\mathsf{F}_{\mathbf{u}_j} - f_{u_1}}{P_U(u_1)} = \mathsf{F}_{\mathbf{u}'}. \tag{5.74}$$

Thus, we see that

$$\mathsf{F}_{\mathbf{u}'} < 0.\mathbf{c}_{\mathbf{u}'} \le \mathsf{F}_{\mathbf{u}'} + p_{\mathbf{u}'} \tag{5.75}$$

and therefore (5.64) still holds. So, we can keep decoding recursively to gain back the complete sequence $\mathbf{u}_j$.

Finally, there is the problem of the end of one codeword and the start of the next. If instead of the correct codeword $0.\mathbf{c}_{\mathbf{u}}$, we use a too long sequence $0.\mathbf{c}$ consisting of more than one codeword, then we again increase the value, $0.\mathbf{c}_{\mathbf{u}} \le 0.\mathbf{c}$. But since $0.\mathbf{c}_{\mathbf{u}}$ is truncated to $\tilde{l}_{\mathbf{u}}$, this increase is strictly less than $\mathrm{D}^{-\tilde{l}_{\mathbf{u}}}$, i.e., we have

$$0.\mathbf{c}_{\mathbf{u}} \le 0.\mathbf{c} < 0.\mathbf{c}_{\mathbf{u}} + \mathrm{D}^{-\tilde{l}_{\mathbf{u}}} \tag{5.76}$$

and we see again from (5.45) that we still are in the correct boundaries:

$$f_{\alpha_i} \le \mathsf{F}_{\mathbf{u}_j} < 0.\mathbf{c}_{\mathbf{u}_j} \le 0.\mathbf{c} < 0.\mathbf{c}_{\mathbf{u}} + \mathrm{D}^{-\tilde{l}_{\mathbf{u}}} < \mathsf{F}_{\mathbf{u}_{j+1}} \le f_{\alpha_{i+1}}. \tag{5.77}$$

Also (5.67)–(5.73) can be adapted accordingly:

$$\frac{0.\mathbf{c} - f_{u_1}}{P_U(u_1)} \le \frac{\mathsf{F}_{\mathbf{u}_j} + 2\mathrm{D}^{-\tilde{l}_{\mathbf{u}_j}} - \epsilon_{\mathbf{u}_j} - f_{u_1}}{P_U(u_1)} \tag{5.78}$$

$$\leq \mathrm{F}_{\mathbf{u}'} + \frac{\mathrm{D}\mathrm{D}^{\log_{\mathrm{D}} p_{\mathbf{u}_j}-1}}{P_U(u_1)} \tag{5.79}$$

$$= \mathrm{F}_{\mathbf{u}'} + \frac{P_U(u_1)\cdots P_U(u_{\mathrm{M}})}{P_U(u_1)} \tag{5.80}$$

$$= \mathrm{F}_{\mathbf{u}'} + p_{\mathbf{u}'}, \tag{5.81}$$

where the inequality follows by bounding $2 \leq \mathrm{D}$, dropping the ceiling, and bounding $\epsilon_{\mathbf{u}_j} \geq 0$.

Thus, our recursive scheme still works.

So, the decoder will firstly compute the length of the longest possible codeword, then it will pick this many digits from the received sequence and start the decoding procedure leading to (5.61). Once it has decoded a codeword and regained M source letters, it will compute the correct length of the decoded codeword and remove it from the sequence. Then it can restart the same procedure again.

The algorithm is summarized as follows.

Decoding of an D-ary Arithmetic Code:

Step 1: Given is the blocklength M, the DMS U with an alphabetical order on the source alphabet $\mathcal{U} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$, and a sequence $\{\mathbf{c}\}$ of D-ary bits representing one or several concatenated codewords. We compute

$$\tilde{l}_{\max} = \left\lceil \mathrm{M} \log_{\mathrm{D}} \frac{1}{\min_{i\in\{1,\ldots,r\}} P_U(\alpha_i)} \right\rceil + 1, \tag{5.82}$$

and take the $\tilde{l}_{\max}$ first digits $\mathbf{c}$ from $\{\mathbf{c}\}$. (If the sequence is shorter than $\tilde{l}_{\max}$ digits, simply take the complete sequence.) We define

$$f_{\alpha_i} \triangleq \sum_{i'=1}^{i-1} P_U(\alpha_{i'}), \quad i = 1, \ldots, r, \tag{5.83}$$

set $k \triangleq 1$, $p_0 = 1$, and compute the decimal representation of $\tilde{\mathrm{F}}_1 \triangleq (0.\mathbf{c})_{10}$.

Step 2: Find the largest f_{α_i} such that $f_{\alpha_i} \leq \tilde{\mathrm{F}}_k$. Put out α_i and compute

$$\tilde{\mathrm{F}}_{k+1} \triangleq \frac{\tilde{\mathrm{F}}_k - f_{\alpha_i}}{P_U(\alpha_i)}, \tag{5.84}$$

$$p_k \triangleq p_{k-1} \cdot P_U(\alpha_i). \tag{5.85}$$

Step 3: If $k < M$, $k \to k+1$ and go to Step 2. Otherwise, compute

$$\tilde{l} \triangleq \left\lceil \log_D \frac{1}{p_M} \right\rceil + 1 \tag{5.86}$$

and remove the first $\tilde{l}$ digits from $\{\mathbf{c}\}$. If there are still digits remaining, repeat the algorithm for the next codeword.

Note that also here we have almost no delay: Apart from a short wait at the very beginning, the algorithm can basically run continuously with every incoming codeword digit.

Example 5.6. We again consider the DMS from Example 5.4. We are given a stream of encoded source symbols based on an arithmetic code of blocklength $M = 3$. The first couple of digits of the received sequence are as follows: 10001110100...

We firstly compute

$$\tilde{l}_{\max} = \left\lceil 3 \log_2 \frac{1}{0.2} \right\rceil + 1 = 8 \tag{5.87}$$

and pick the first 8 digits from the given sequence. We compute:

$$\tilde{F}_1 \triangleq (0.10001110)_2 = (0.5546875)_{10}. \tag{5.88}$$

Using $f_a \triangleq 0$, $f_b \triangleq p_a = 0.5$, and $f_c \triangleq p_a + p_b = 0.8$, we see that the largest f_{α_i} such that $f_{\alpha_i} \le \tilde{F}_1$ is f_b. Hence, we put out b and update the value of $\tilde{F}_1$:

$$\begin{aligned}
&\text{put out } b && p_1 = 1 \cdot 0.3 = 0.3 && \tilde{F}_2 = (0.5546875 - 0.5)/0.3 \\
&&&&& = 0.182291\bar{6} && (5.89)\\
&\text{put out } a && p_2 = 0.3 \cdot 0.5 = 0.15 && \tilde{F}_3 = (0.182291\bar{6} - 0)/0.5 \\
&&&&& = 0.36458\bar{3} && (5.90)\\
&\text{put out } a && p_3 = 0.15 \cdot 0.5 = 0.075. && && (5.91)
\end{aligned}$$

Hence, the decoded source sequence is baa. The codeword length is

$$\tilde{l} = \left\lceil \log_2 \frac{1}{0.075} \right\rceil + 1 = 5, \tag{5.92}$$

i.e., the first 5 digits from the received sequence 10001 should be removed. The remaining sequence then is 110100... We could now continue to decode those digits. ◊

We would like to conclude the discussion of arithmetic coding by pointing out that there exist different variations of arithmetic coding schemes. For example, instead of adding $D^{-\tilde{l}_{\mathbf{u}}}$ to $F_{\mathbf{u}}$ as in (5.24), one could also add $\frac{1}{2}p_{\mathbf{u}}$ to $F_{\mathbf{u}}$. Another possibility is to use $F_{\mathbf{u}}$ directly without shift, but then not to

truncate it, but to round it up. These changes have no fundamental impact on the performance. However, we would like to mention that we have not worried about numerical precision: There are other versions of the encoding and decoding algorithm that are numerically more stable than those introduced here.

For an easy-to-read introduction to arithmetic coding including its history, the introductory chapter of the Ph.D. thesis of Jossy Sayir is highly recommended [Say99].

5.4 Variable-Length–to–Block Coding of a DMS

The variable length codewords that we have considered to this point are sometimes inconvenient in practice. For instance, if the codewords are stored in a computer memory, one would usually prefer to use codewords whose length coincides with the computer word-length. Or if the digits in the codeword are to be transmitted synchronously (say, at the rate of 2400 bits/s), one would usually not wish to buffer variable-length codewords to assure a steady supply of digits to be transmitted.

But it was precisely the variability of the codeword lengths that permitted us to encode a single random variable efficiently, such as the message $\mathbf{V}$ from an M-block parser for a DMS. How can we get similar coding efficiency for a DMS when all codewords are forced to have the same length? The answer is that we must assign codewords not to blocks of source letters but rather to variable-length sequences of source letters, i.e., we must do *variable-length parsing*.

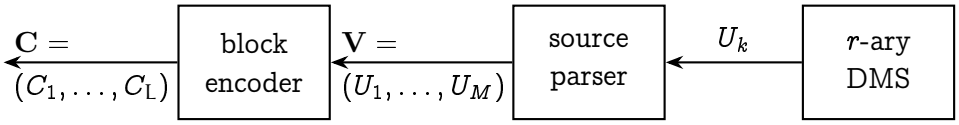


Figure 5.2: Variable-length–to–block coding of a discrete memoryless source: The source output sequence $\{U_k\}$ is parsed into messages $\mathbf{V}$ of variable length M, that are then assigned a codeword $\mathbf{C}$ of fixed length L.

So, we consider the coding of a variable number of source letters into an L-block codeword as indicated in Figure 5.2. Note that while an M-block parser in the situation of block–to–variable-length coding produces messages of equal length M, now the situation is more complicated with different messages of different length. We define the *message set* $\mathcal{V}$ to be the set of all possible (variable-length) output sequences of the source parser.

Note that since the DMS is memoryless, we may still concentrate on just one message $\mathbf{V} = [U_1, U_2, \ldots, U_M]$ and do not need to worry about the future

(the future is independent of the past!). Again we want to minimize the *average number of code digits per source letter*. In the case of a block–to–variable-length code this was

$$\frac{\mathsf{E}[L]}{\mathrm{M}}. \tag{5.93}$$

However, in our new situation of variable-length–to–block codes L is constant and M is variable. Hence, we now have to consider

$$\frac{\mathrm{L}}{\mathsf{E}[M]}, \tag{5.94}$$

as is proven in the following lemma.

Lemma 5.7. The quality criterion of a general coding scheme for a DMS according to Figure 5.1, where the r-ary source sequence $\{U_k\}$ is parsed into messages of variable length M, which are then mapped into D-ary codewords of variable length L, is the **average number of code digits per source letter**

$$\frac{\mathsf{E}[L]}{\mathsf{E}[M]}. \tag{5.95}$$

Proof: Since M source symbols are mapped into a codeword of length L, it is clear that we need $\frac{L}{M}$ code digits per source letter. However, L and M are random, i.e., we somehow need to compute an *averaged* version of this quality criterion. So why do we consider $\frac{\mathsf{E}[L]}{\mathsf{E}[M]}$ and not $\mathsf{E}\big[\frac{L}{M}\big]$?

First of all, note that $\frac{\mathsf{E}[L]}{\mathsf{E}[M]}$ and $\mathsf{E}\big[\frac{L}{M}\big]$ are *not* the same! If you are unsure about that, go back to the definition of expectation (see (2.19) and (2.44)) and think about linearity. You might also want to go back to the Jensen Inequality (Theorem 2.1) and think about it.

So, since they are not the same, which one should we use in our situation? To answer this, note that what we really want to do is to use our compression scheme many times and thereby minimize the *total* number of used code digits divided by the *total* number of encoded source letters:

$$\frac{L_1 + L_2 + \cdots + L_n}{M_1 + M_2 + \cdots + M_n}. \tag{5.96}$$

Here $M_{k'}$ denotes the length of the message during the k'th time we use the system, and $L_{k'}$ is the length of the corresponding codeword. If we now use the system for a long time, i.e., if we let n become large, then we know from the weak law of large numbers that

$$\frac{L_1 + L_2 + \cdots + L_n}{M_1 + M_2 + \cdots + M_n} = \frac{\frac{L_1+L_2+\cdots+L_n}{n}}{\frac{M_1+M_2+\cdots+M_n}{n}} \xrightarrow{n\to\infty} \frac{\mathsf{E}[L]}{\mathsf{E}[M]} \quad \text{in prob.} \tag{5.97}$$

Hence, the quantity of interest is $\frac{\mathsf{E}[L]}{\mathsf{E}[M]}$. □

So, we return to the case where the codewords all have constant length $\mathsf{E}[L] = \mathrm{L}$ and consider a simple example of a ternary DMS.

Example 5.8. Consider a ternary DMS $r = 3$ with source alphabet $\{a, b, c\}$. We decide to use the following message set:

$$\mathcal{V} \triangleq \{aaa, aab, aac, b, ca, cb, cc\}, \tag{5.98}$$

and assign them the following corresponding binary ($\mathrm{D} = 2$) codewords of fixed length $\mathrm{L} = 3$:

$$\mathcal{C} \triangleq \{001, 010, 011, 100, 101, 110, 111\}. \tag{5.99}$$

So, for example, the source output sequence $aabcccabca\ldots$ is split as follows

$$aab|cc|ca|b|ca|\ldots \tag{5.100}$$

and is mapped into

$$010111101100101\ldots \tag{5.101}$$

Will this system perform well? This of course depends on the probability distribution of the source! However, if the long messages are very likely, i.e., if a is much more likely than b, then it should not be too bad. Concretely, if $\mathsf{E}[M]$ is large, then $\frac{\mathrm{L}}{\mathsf{E}[M]}$ is small, as it is wished for a good compression. ◊

Example 5.9. We make another example to point out some possible problems. Consider again $r = 3$ with source alphabet $\{a, b, c\}$, but now choose $\mathcal{V} \triangleq \{aa, b, cba, cc\}$ as message set. Then the sequence of source letters $aabcccabca\ldots$ cannot be split into different messages:

$$aa|b|cc|\underbrace{cabca}_{?}\ldots \tag{5.102}$$

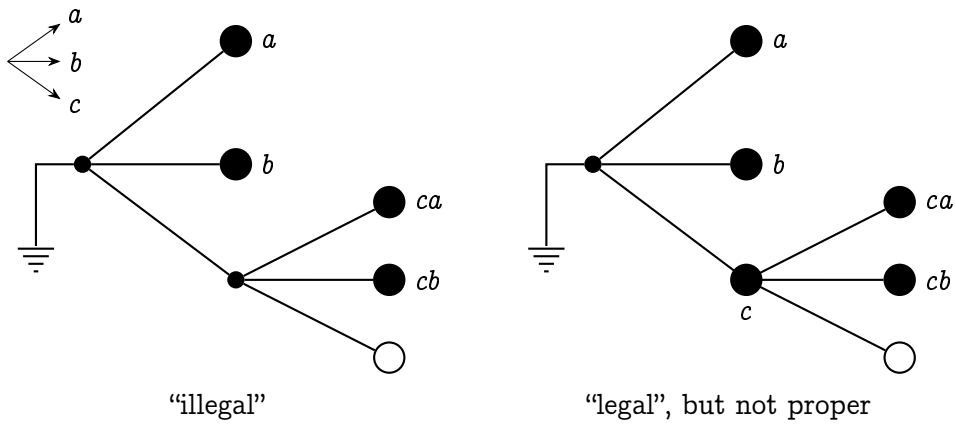
So this sequence cannot be parsed! What is wrong? ◊

We realize that, similarly to codes, also message sets must satisfy some properties. In particular we have the following condition on any "legal" message set:

> Every possible sequence of source symbols $U_1, U_2, \ldots$ must contain a message $\mathbf{V}$ as prefix so that we can parse every possible source sequence.

Since we have successfully been using the tool of rooted trees in order to describe the mapping from messages to codewords, we will now try to do the same thing also for the mapping of source sequences to messages. Obviously, *since we have an r-ary DMS, we need to consider r-ary trees!*

Example 5.10. We once more examine the ternary ($r = 3$) source with source alphabet $\{a, b, c\}$. In Figure 5.3 you see two examples of message sets that are represented by ternary trees. The first set is not allowed because not every possible sequence from the source can be parsed into messages: Any sequence containing two c in sequence fails to be parsed. The second one, however, is a legal message set once we define a clear rule how to deal with cases where the parsing is not unique. ◊



"illegal" "legal", but not proper

Figure 5.3: Two examples of message sets. The message set on the left cannot be used for a source parser because there is no valid parsing of the source sequence $(U_1, U_2, \ldots) = (c, c, \ldots)$. The message set on the right, however, can be used, perhaps with the rule *always choose the longest message if more than one parsing is possible*. Example: $c|ca|b|\ldots$ instead of $c|c|a|b|\ldots$

Similarly to our codes, we also would like to have a "prefix-free" property: We prefer a message set where the encoder can use the message immediately and does not need to wait longer. The message set on the right of Figure 5.3 is an example of a set that does not have this desired "prefix-free" property. In particular, if the parser sees a c at its input it cannot yet decide whether this will be a message, but first has to wait and see the following source symbol. We give the following definition.

Definition 5.11. An r-ary source message set is called *proper* if, and only if, it forms a *complete set of leaves* in an r-ary tree (i.e., all leaves are messages, no nodes are messages).

Example 5.12. An example of a proper message set for a quaternary ($r = 4$) source U is given in Figure 5.4. Note that in this tree we have already assigned the corresponding probabilities to each message, assuming that the PMF of the source U is $P_U(a) = 0.5$, $P_U(b) = P_U(c) = 0.2$ and $P_U(d) = 0.1$. ◊

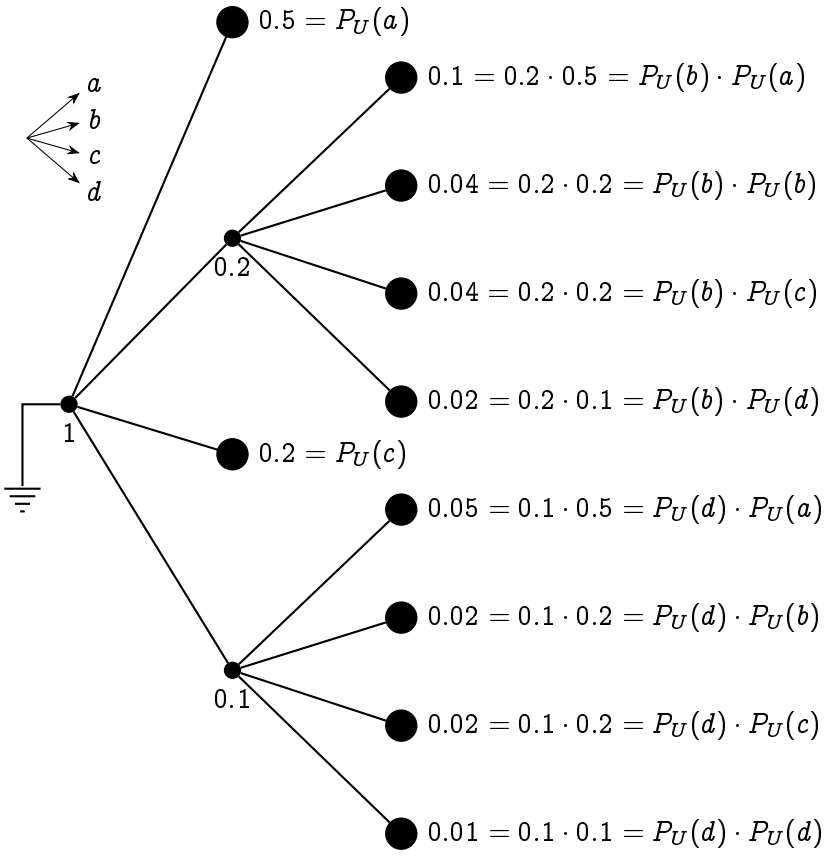


Figure 5.4: A proper message set for a quaternary source with PMF $P_U(a) = 0.5$, $P_U(b) = 0.2$, $P_U(c) = 0.2$, and $P_U(d) = 0.1$. The message $\mathbf{V}$ will take value in $\{a, ba, bb, bc, bd, c, da, db, dc, dd\}$.

Using our knowledge about trees, in particular, using the Leaf Entropy Theorem (Theorem 4.20) and the Path Length Lemma (Lemma 4.15), we can easily compute the entropy of the messages of a proper message set:

$$\begin{aligned} \mathrm{H}(\mathbf{V}) &= \mathrm{H}_{\text{leaf}} && (5.103)\\ &= \sum_{\ell=1}^{\mathrm{N}} \mathrm{P}_\ell \cdot \mathrm{H}_\ell && (5.104)\\ &= \sum_{\ell=1}^{\mathrm{N}} \mathrm{P}_\ell \cdot \mathrm{H}(U) && (5.105)\\ &= \mathrm{H}(U) \cdot \sum_{\ell=1}^{\mathrm{N}} \mathrm{P}_\ell && (5.106)\\ &= \mathrm{H}(U) \cdot \mathsf{E}[M]. && (5.107) \end{aligned}$$

Here, (5.103) follows from the fact that all possible values of $\mathbf{V}$ are leaves; (5.104) follows from the Leaf Entropy Theorem (Theorem 4.20); in (5.105) we

use the fact that the branching entropy is always the source entropy because the PMF of the source decides about the branching; and (5.107) follows from the Path Length Lemma (Lemma 4.15).

Hence we get the following theorem.

Theorem 5.13. The entropy of a proper message set $\mathrm{H}(\mathbf{V})$ for an r-ary DMS U is

$$\mathrm{H}(\mathbf{V}) = \mathrm{H}(U) \cdot \mathrm{E}[M] \tag{5.108}$$

where $\mathrm{E}[M]$ is the average message length.

Example 5.14. Consider the following ternary DMS: $P_U(a) = 0.1$, $P_U(b) = 0.3$, $P_U(c) = 0.6$. We would like to design a binary ($\mathrm{D} = 2$) block code for this source. Note that

$$\mathrm{H}(U) = -0.1\log 0.1 - 0.3\log 0.3 - 0.6\log 0.6 \approx 1.295 \text{ bits}. \tag{5.109}$$

Using the proper message set $\mathcal{V} = \{a, b, ca, cb, cc\}$ shown in Figure 5.5, we see that

$$\begin{aligned}
\mathrm{E}[M] &= 1 + 0.6 = 1.6, && (5.110)\\
\mathrm{H}(\mathbf{V}) &= -0.1\log 0.1 - 0.3\log 0.3 - 0.06\log 0.06 \\
&\quad - 0.18\log 0.18 - 0.36\log 0.36 && (5.111)\\
&\approx 2.073 \text{ bits}, && (5.112)\\
\mathrm{H}(U) \cdot \mathrm{E}[M] &\approx 1.295 \text{ bits} \cdot 1.6 = 2.073 \text{ bits}. && (5.113)
\end{aligned}$$

Note that since we have 5 different messages, we need to assign 5 codewords of equal length. The smallest possible choice for the codeword length will therefore be $\mathrm{L} = 3$. ◊

5.5 General Converse

Next note that from Theorem 5.13 we know

$$\mathrm{H}(\mathbf{V}) = \mathrm{H}(U)\,\mathrm{E}[M] \tag{5.114}$$

and from the converse part of the coding theorem for a single random message (Theorem 4.38) we have

$$\mathrm{E}[L] \geq \frac{\mathrm{H}(\mathbf{V})}{\log \mathrm{D}}. \tag{5.115}$$

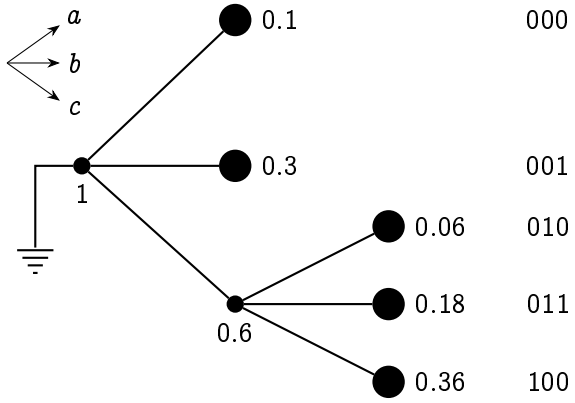


Figure 5.5: A proper message set for the DMS U of Example 5.14 and a possible binary block code.

Hence, we can derive a general converse:

$$\mathrm{E}[L] \geq \frac{\mathrm{H}(U)\,\mathrm{E}[M]}{\log \mathrm{D}}. \tag{5.116}$$

Theorem 5.15 (Converse to the General Coding Theorem for a DMS).
For any uniquely decodable D-ary code of any **proper** message set for an r-ary DMS U, we have

$$\frac{\mathrm{E}[L]}{\mathrm{E}[M]} \geq \frac{\mathrm{H}(U)}{\log \mathrm{D}}, \tag{5.117}$$

where $\mathrm{H}(U)$ is the entropy of a single source symbol, $\mathrm{E}[L]$ is the average codeword length, and $\mathrm{E}[M]$ is the average message length.

5.6 Optimal Message Sets: Tunstall Message Sets

We now would like to find an optimum way of designing proper message sets. From Lemma 5.7 we know that in order to *minimize* the average number of codewords digits per source letter, we have to *maximize* $\mathrm{E}[M]$.

Definition 5.16. A proper message set for an r-ary DMS with $n = 1 + \mathrm{N}(r-1)$ messages is a *Tunstall message set* if the corresponding r-ary tree can be formed, beginning with the extended root, by $\mathrm{N} - 1$ applications of the rule *"extend the most likely leaf"*.

As before, n denotes the number of leaves and N is the number of nodes (including the root).

Example 5.17. Consider a binary memoryless source (BMS), i.e., $r = 2$, with the following PMF:

$$P_U(u) = \begin{cases} 0.4 & u = 1, \\ 0.6 & u = 0. \end{cases} \tag{5.118}$$

Moreover, we would like to have a message set with $n = 5$ messages, i.e.,

$$n = 5 = 1 + \mathrm{N}(r-1) = 1 + \mathrm{N}, \tag{5.119}$$

i.e., we need to apply the rule "extend the most likely leaf" $\mathrm{N} - 1 = 3$ times. The corresponding Tunstall message set is shown in Figure 5.6. ◊

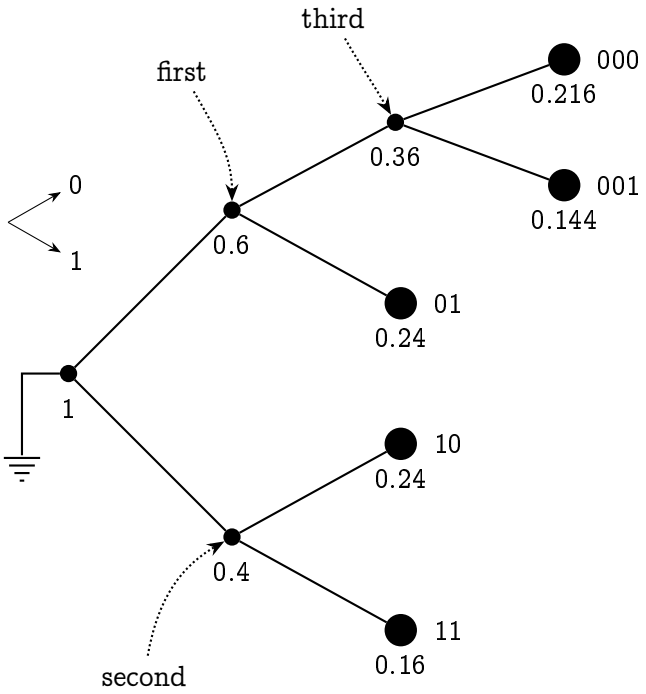


Figure 5.6: Example of a Tunstall message set for a BMS. The dashed arrows indicate the order in which the tree has been grown.

The following lemma follows directly from the definition of a Tunstall message set.

Lemma 5.18 (Tunstall Lemma). A proper message set for an r-ary DMS is a Tunstall message set if, and only if, in its r-ary tree every node is at least as likely as every leaf.

Proof: Consider growing a Tunstall message set by beginning from the extended root and repeatedly extending the most likely leaf. The extended

root trivially has the property that no leaf is more likely than its only node (the root itself). Suppose this property continues to hold for the first ℓ extensions. On the next extension, none of the r new leaves can be more likely than the node just extended and thus none is more likely than any of the old nodes since these were all at least as likely as the old leaf just extended. But none of the remaining old leaves is more likely than any old node nor more likely than the new node since this latter node was previously the most likely of the old leaves. Thus, the desired property holds also after $\ell + 1$ extensions. By induction, this property holds for every Tunstall message set.

Conversely, consider any proper message set with the property that, in its r-ary tree, no leaf is more likely than any intermediate node. This property still holds if we "prune" the tree by cutting off the r branches stemming from the least likely node. After enough such prunings, we will be left only with the extended root. But if we then re-grow the same tree by extending leaves in the reverse order to our pruning of nodes, we will at each step be extending the most likely leaf. Hence this proper message set was indeed a Tunstall message set, and the lemma is proved. □

Based on this lemma we can now prove the following theorem.

Theorem 5.19. A proper message set with n messages for an r-ary DMS maximizes $\mathsf{E}[M]$ over all such proper message sets if, and only if, it is a Tunstall message set.

Proof: The theorem follows from the Path Length Lemma (Lemma 4.15) and the Tunstall Lemma (Lemma 5.18). Consider the r-ary semi-infinite tree of the DMS. Note that all vertices of an r-ary message set are nodes in this semi-infinite tree. Since $\mathsf{E}[M] = \sum_\ell \mathsf{P}_\ell$, if we want to maximize $\mathsf{E}[M]$, we need to pick the N most likely nodes in this tree. This means that the $n = 1 + \mathsf{N}(r-1)$ leaves will all be less likely than these nodes. Hence, according to the Tunstall Lemma we have created a Tunstall message set. □

5.7 Optimal Variable-Length–to–Block Codes: Tunstall Codes

To find the optimal variable-length–to–block codes we already have almost all ingredients. In the last section we have proven that the optimal proper message sets are Tunstall message sets. There is only one question remaining: How large should the message set be, i.e., how shall we choose n?

Note that

- we want $\mathsf{E}[M]$ to be large, i.e., we want to choose n large;

- we can increase n in steps of size $(r-1)$ (see Leaf Counting Lemma, Lemma 4.10);
- but since the codeword length L and the size of the coding alphabet D are fixed, we have at most D^{L} possible codewords available.

So we realize that we should choose n such that it satisfies

$$\mathrm{D}^{\mathrm{L}} - (r-1) < n \le \mathrm{D}^{\mathrm{L}}. \tag{5.120}$$

We play around with this expression to get

$$-\mathrm{D}^{\mathrm{L}} + (r-1) > \quad -n \quad \ge -\mathrm{D}^{\mathrm{L}}, \tag{5.121}$$

$$\Longrightarrow \qquad (r-1) > \mathrm{D}^{\mathrm{L}} - n \ge 0. \tag{5.122}$$

Using the Leaf Counting Lemma (Lemma 4.10), i.e., $n = 1 + \mathrm{N}(r-1)$, we get

$$\mathrm{D}^{\mathrm{L}} - n = \mathrm{D}^{\mathrm{L}} - 1 - \mathrm{N}(r-1) \tag{5.123}$$

or

$$\mathrm{N}(r-1) + \underbrace{\mathrm{D}^{\mathrm{L}} - n}_{\triangleq \mathrm{R}} = \mathrm{D}^{\mathrm{L}} - 1, \tag{5.124}$$

where by (5.122) $0 \le \mathrm{R} < r-1$. So by Euclid's Division Theorem, we have that N is the quotient when $\mathrm{D}^{\mathrm{L}} - 1$ is divided by $r-1$. (Note that additionally $\mathrm{D}^{\mathrm{L}} \ge r$ is required because the smallest proper message set has r messages.)

Hence, we get the following algorithm.

Tunstall's Algorithm for Optimum D-ary L-Block Encoding of a Proper Message Set for an r-ary DMS:

Step 1: Check whether $\mathrm{D}^{\mathrm{L}} \ge r$. If not, stop because such coding is not possible. Otherwise compute the quotient N when $\mathrm{D}^{\mathrm{L}} - 1$ is divided by $r-1$:

$$\mathrm{N} \triangleq \left\lfloor \frac{\mathrm{D}^{\mathrm{L}} - 1}{r-1} \right\rfloor. \tag{5.125}$$

Step 2: Start with the extended root and construct the Tunstall message set of size $n = 1 + \mathrm{N}(r-1)$ by $\mathrm{N} - 1$ times extending the most likely leaf.

Step 3: Assign a distinct D-ary codeword of length L to each message.

Example 5.20. We again consider the BMS ($r = 2$) with $P_U(0) = 0.6$ and $P_U(1) = 0.4$, and construct a binary block code of length 3, i.e., $\mathrm{D} = 2$ and $\mathrm{L} = 3$. Then

$$\mathrm{D}^{\mathrm{L}} = 2^3 = 8 \geq r = 2 \implies \text{OK} \tag{5.126}$$

and

$$\mathrm{N} \triangleq \left\lfloor \frac{\mathrm{D}^{\mathrm{L}} - 1}{r - 1} \right\rfloor = \left\lfloor \frac{2^3 - 1}{2 - 1} \right\rfloor = 7, \tag{5.127}$$

i.e., starting with the extended root, we need to extend the most likely leaf six times. The corresponding Tunstall message set is shown in Figure 5.7.

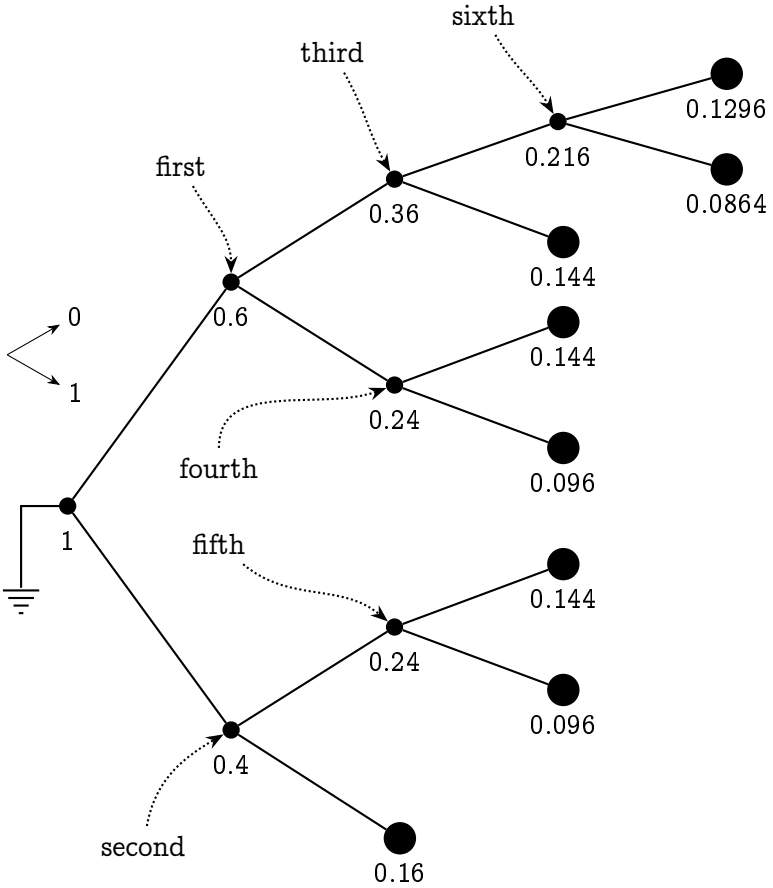


Figure 5.7: Tunstall message set for the BMS of Example 5.20. The dashed arrows indicate the order in which the tree has been grown.

We get

$$\mathsf{E}[M] = 3.056, \tag{5.128}$$

$$\mathsf{E}[L] = \mathrm{L} = 3, \tag{5.129}$$

$$\frac{\mathrm{H}(U)}{\log \mathrm{D}} \approx 0.971, \tag{5.130}$$

$$\Longrightarrow \quad \frac{\mathrm{L}}{\mathrm{E}[M]} \approx 0.982 \geq 0.971 \approx \frac{\mathrm{H}(U)}{\log \mathrm{D}}. \tag{5.131}$$

And finally, our choice of the Tunstall code is shown in Table 5.8.

Table 5.8: Tunstall code of Example 5.20.

Message	Codewords
0000	000
0001	001
001	010
010	011
011	100
100	101
101	110
11	111

Notice that if we had used "no coding at all" (which is possible here since the source alphabet and coding alphabet coincide), then we would have achieved trivially 1 code digit per source digit. It would be an economic question whether the 2% reduction in code digits offered by the Tunstall scheme in the above example was worth the cost of its implementation. In fact, we see that no coding scheme could "compress" this binary source by more than 3% so that our Tunstall code has not done too badly. ◊

There also exists a coding theorem for Tunstall codes, but it is less beautiful than the version for the Huffman codes.

Theorem 5.21 (The Variable-Length–to–Block Coding Theorem for a DMS). The ratio $\mathrm{E}[M]/\mathrm{L}$ of average message length to blocklength for an optimum D-ary block-L encoding of a proper message set for an r-ary DMS satisfies

$$\frac{\log \mathrm{D}}{\mathrm{H}(U)} - \frac{\log \frac{2}{p_{\min}}}{\mathrm{L}\,\mathrm{H}(U)} < \frac{\mathrm{E}[M]}{\mathrm{L}} \leq \frac{\log \mathrm{D}}{\mathrm{H}(U)}, \tag{5.132}$$

where $\mathrm{H}(U)$ is the entropy of a single source letter and where $p_{\min} = \min_u P_U(u)$ is the probability of the least likely source letter. (We actually have assumed here that $P_U(u) > 0$ for all u. If this is not the case, simply remove all letters with $P_U(u) = 0$ from the source alphabet.)

Proof: The right inequality follows directly from the general converse (Theorem 5.15). It remains to prove the left inequality.

The least likely message in the message set has probability $\mathrm{P}_\kappa \cdot p_{\min}$ where P_κ is the probability of the node from which it stems; but since there are n messages, this least likely message has probability at most $\frac{1}{n}$, so we must have

$$\mathrm{P}_\kappa \cdot p_{\min} \le \frac{1}{n}. \tag{5.133}$$

By the Tunstall Lemma (Lemma 5.18) no message (i.e., no leaf) has probability more than P_κ so that (5.133) implies

$$P_{\mathbf{V}}(\mathbf{v}) \le \mathrm{P}_\kappa \le \frac{1}{n p_{\min}}, \quad \forall\, \mathbf{v}, \tag{5.134}$$

which in turn implies

$$-\log P_{\mathbf{V}}(\mathbf{v}) \ge \log n - \log \frac{1}{p_{\min}}, \quad \forall\, \mathbf{v}. \tag{5.135}$$

Next, we recall that the number $n = 1 + \mathrm{N}(r-1)$ of messages was chosen with N as large as possible for D^{L} codewords so that surely (see (5.120))

$$n + (r-1) > \mathrm{D}^{\mathrm{L}}. \tag{5.136}$$

But $n \ge r$ so that (5.136) further implies

$$2n \ge n + r > \mathrm{D}^{\mathrm{L}} + 1 > \mathrm{D}^{\mathrm{L}}, \tag{5.137}$$

which, upon substitution in (5.135), gives

$$\mathrm{H}(\mathbf{V}) = \mathrm{E}[-\log P_{\mathbf{V}}(\mathbf{v})] \ge \log \frac{2n}{2} - \log \frac{1}{p_{\min}} > \mathrm{L} \log \mathrm{D} - \log \frac{2}{p_{\min}}. \tag{5.138}$$

Making use of Theorem 5.13, we see that (5.138) is equivalent to

$$\mathrm{E}[M]\, \mathrm{H}(U) > \mathrm{L} \log \mathrm{D} - \log \frac{2}{p_{\min}}. \tag{5.139}$$

Dividing now by $\mathrm{L}\, \mathrm{H}(U)$ gives the left inequality in (5.132). □

We see that by making L large we can again approach the upper bound in (5.132) arbitrarily closely. So, it is possible to approach the fundamental limit both with Huffman codes and Tunstall codes. Usually a Huffman code is slightly more efficient than the corresponding Tunstall code with the same number of codewords. The most efficient way is, of course, to combine a Tunstall message set with a Huffman code. This will then lead to an optimal *variable-length–to–variable-length code* (see exercises).[4]

In spite of its fundamental importance, Tunstall's work was never published in the open literature. Tunstall's doctoral thesis [Tun67], which contains this work, lay unnoticed for many years before it became familiar to information theorists.

[4] What happens if we do block–to–block coding? In general this will result in lossy compression!

Remark 5.22. Tunstall's work shows how to do optimal variable-length–to–block coding of a DMS, but only when one has decided to use a *proper* message set. A legal, but not proper message set — denoted *quasi-proper message set* (such as the message set on the right of Figure 5.3) — in which there is at least one message on each path from the root to a leaf in its r-ary tree, but sometimes more than one message, can also be used. Quite surprisingly (as Exercises 5.25 below will show), one can sometimes do better variable-length–to–block coding of a DMS with a quasi-proper message set than one can do with a proper message set having the same number of messages. Perhaps even more surprisingly, nobody today knows how to do optimal variable-length–to–block coding of a DMS with a given number of messages when quasi-proper message sets are allowed.

Note, however, that one cannot use less than $(\log \mathrm{D})/\mathrm{H}(U)$ source letters per D-ary code digit on the average, i.e., the converse (Theorem 5.15) still holds true also for quasi-proper message sets. The proof of this most general converse will be deferred to the advanced information theory course. It is based on the concept of *rate distortion theory* (also see the discussion on p. 367). △

5.8 Efficiency of a Source Coding Scheme

It is common to define an efficiency of a coding scheme.

Definition 5.23. The *efficiency* η of a source coding scheme is defined by the ratio of the lower bound $\mathrm{H}(U)/\log \mathrm{D}$ in the general coding theorem (Theorem 5.15) to the achieved average number of codeword digits per source letter $\mathrm{E}[L]/\mathrm{E}[M]$:

$$\eta \triangleq \frac{\mathrm{H}(U) \cdot \mathrm{E}[M]}{\log \mathrm{D} \cdot \mathrm{E}[L]}. \tag{5.140}$$

Note that $0 \le \eta \le 1$, where 1 denotes full efficiency because in this case the code achieves the best possible theoretical lower bound exactly.

Remark 5.24. We end this chapter with a final remark. All the systems we have discussed here contain one common drawback: We have always assumed that the probability statistics of the source is known in advance and can be used in the design of the system. In a practical situation this is often not the case. For example, what is the probability distribution of digitized speech in a telephone system? Or of English ASCII text in comparison to French ASCII text? Or of different types of music? A really practical system should work independently of the source, i.e., it should estimate the probabilities of the source symbols on the fly and adapt to it automatically. Such a system is called a *universal compression scheme*. Such systems do exist, and we

will discuss some examples in Chapter 7 in the even more general context of sources with memory. Beforehand, we need to discuss memory, and how to compute the entropy of sources with memory. △

Exercise 5.25 (Quasi-Proper Message Sets). *Consider a BMS with* $P_U(0) = 0.9$ *and* $P_U(1) = 0.1$.

1. *Find the binary (*$\mathrm{L} = 2$*)-block Tunstall code for this source and confirm that its efficiency is* $\eta = 0.635$.

Consider the binary message set shown in Figure 5.9. Although this mes-

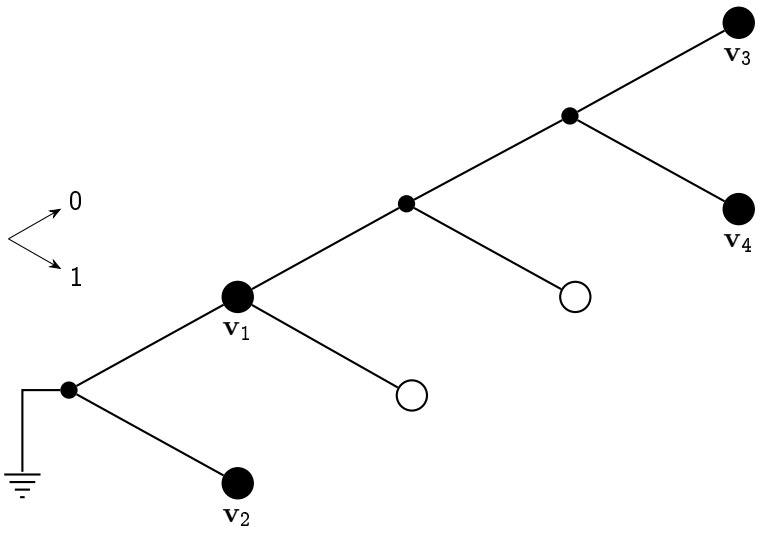


Figure 5.9: Quasi-proper message set.

sage set is not proper, it could in fact be used in practice if we adopted the convention that, when more than one choice is possible, we always choose the longest *message. Thus, if the source output sequence was* $0001001\ldots$*, the resulting sequence of messages would be* $\mathbf{v}_4, \mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_2, \ldots$ *Notice that we must sometimes examine digits beyond the end of a message before we choose that message. We call such a message* quasi-proper*: A* quasi-proper messages set *is a message set that is not prefix-free but that has the property that every path from the root to a leaf in its* r*-ary tree contains at least one message node.*

Note that in general it is not easy to compute the probabilities of the messages in a quasi-proper message set (try it!), and thus there is no simple direct way to analyze the performance of such a variable-length–to–block coding system. However, in the given example here, we can use a trick to convert the given quasi-proper message set into a proper one.

2. *Show that the given quasi-proper message set is equivalent to a proper message set with $n = 5$ messages in which each message in the latter message set is a sequence of one or more messages from the former message set. Draw the binary tree for this equivalent proper message set and find its average message length $\mathsf{E}[M]$.*

   Note that the source sequence 01 *will always be identified as the message sequence $\mathbf{v}_1, \mathbf{v}_2$; that the source sequence* 001 *will always be identified as the message sequence $\mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_2$, etc. Moreover, also note that, because we are using blocklength $\mathsf{L} = 2$ coding of the quasi-proper message set, the message $[\mathbf{v}_1, \mathbf{v}_2]$ in the equivalent proper message set will have a codeword length $2\mathsf{L} = 4$.*

3. *Find the average codeword length $\mathsf{E}[L]$ for the five codewords that are assigned to the messages in the equivalent proper message set.*

4. *Finally, confirm that the efficiency of the blocklength $\mathsf{L} = 2$ binary coding of the given quasi-proper message set is $\eta = 0.644$.*

This shows that the efficiency of the ($\mathsf{L} = 2$)-block coding based on the quasi-proper message set is higher than the efficiency of the best ($\mathsf{L} = 2$)-block coding with a proper message set. ◊

Chapter 6

Stochastic Processes and Entropy Rate

Most real information sources exhibit substantial memory so that the DMS is not an appropriate model for them. We now introduce two models for sources with memory that are sufficiently general to satisfactorily model most real information sources, but that are still simple enough to be analytically tractable: *discrete stationary sources* and *Markov sources*.

6.1 Discrete Stationary Sources

We begin with the following very general definition.

Definition 6.1. A *discrete stochastic process* or *discrete random process* $\{X_k\}$ is a sequence of RVs

$$\ldots, X_{-3}, X_{-2}, X_{-1}, X_0, X_1, X_2, \ldots \tag{6.1}$$

with a certain joint probability distribution.

Note that the word *stochastic* is just fancy for *random*. Moreover, we would like to point out that in our notation k denotes the discrete time, $\{X_k\}$ is a discrete-time stochastic process, and X_k denotes the RV of this process at time k.

It is quite obvious that it is difficult to handle the vast amount of all possible discrete stochastic processes. Simply a proper definition that contains the complete joint distribution of the infinite process is a very difficult task indeed! We need to introduce additional restrictions in order to make them more accessible for analysis.

One such restriction is to make sure that the probability laws reigning the process do not change over time. The process is then called *stationary*.

Definition 6.2. A stochastic process $\{X_k\}$ is called *stationary* or *strict-sense stationary (SSS)* if the joint distribution of any subset of the sequence does not change with time: $\forall\,\tau \in \mathbb{Z}$, $\forall\, n \in \mathbb{N}$, $\forall\, k_1, \ldots, k_n$ where $k_j \in \mathbb{Z}$ $(j = 1, \ldots, n)$:

$$\Pr[X_{k_1} = \alpha_1, \ldots, X_{k_n} = \alpha_n] = \Pr[X_{k_1+\tau} = \alpha_1, \ldots, X_{k_n+\tau} = \alpha_n], \quad \forall\,(\alpha_1, \ldots, \alpha_n). \tag{6.2}$$

In particular, note that this definition means that for any stationary stochastic process we have

$$\Pr[X_k = \alpha] = \Pr[X_1 = \alpha], \quad \forall\, k, \alpha. \tag{6.3}$$

Sometimes, strict-sense stationarity is a too strong restriction. Then, weak stationarity might be a solution.

Definition 6.3. A stochastic process $\{X_k\}$ is called *weakly stationary* or *wide-sense stationary (WSS)* if the first and second moments of the process do not depend on time, i.e.,

$$\mathsf{E}[X_k] = \text{constant, not depending on } k, \tag{6.4}$$

$$\mathsf{K}_{XX}(k, j) \triangleq \mathsf{E}\big[(X_k - \mathsf{E}[X_k])(X_j - \mathsf{E}[X_j])\big] = \mathsf{K}_{XX}(k - j). \tag{6.5}$$

Note that by definition any stationary process is also weakly stationary, but not *vice versa*.

The following definition is a straightforward application of the definition of stochastic processes and stationarity.

Definition 6.4. An r-ary *discrete stationary source (DSS)* is a device that emits a sequence of r-ary RVs such that this sequence forms a stationary stochastic process.

6.2 Markov Processes

In engineering we often have the situation that a RV is processed by a series of (random or deterministic) processors, as shown in Figure 6.1. In this case

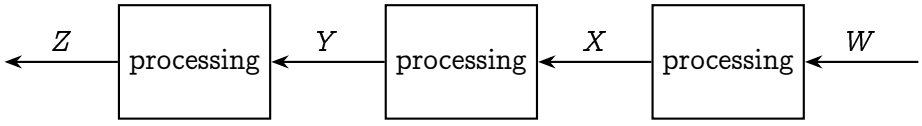


Figure 6.1: A series of processing black boxes with a random input.

Z depends on W, but only via its intermediate products X and Y. In other

words, once we know Y, it does not help us to also know X or W to improve our prediction about Z.

Another way of looking at the same thing is that the system has limited memory: Once we know W we do not need to know anymore what happened *before*! The process "forgets" about its past.

This basic idea can be generalized to a so-called *Markov process*.

Definition 6.5. A discrete stochastic process $\{X_k\}$ is called *Markov process of memory* μ if

$$\begin{aligned} &P_{X_k|X_{k-1},X_{k-2},\ldots,X_{k-\mu},\ldots}(x_k|x_{k-1},x_{k-2},\ldots,x_{k-\mu},\ldots) \\ &\quad = P_{X_k|X_{k-1},\ldots,X_{k-\mu}}(x_k|x_{k-1},\ldots,x_{k-\mu}), \end{aligned} \tag{6.6}$$

for all k and for all $(x_k, x_{k-1}, \ldots)$.

Hence we only need to keep track of the last μ time steps and can forget about anything older than μ time-steps before now!

Remark 6.6. Every Markov process of memory μ can be reduced to a Markov process of memory 1 by *enlarging its alphabet*. To show this, we give an instructive example. △

Example 6.7. Consider $r = 2$ with the alphabet $\{a, b\}$, and let $\mu = 2$, i.e., it is sufficient to know the last two time steps, e.g., $P_{X_3|X_2,X_1}(x_3|x_2,x_1)$. Now define $V_k \triangleq (X_{2k}, X_{2k-1})$ and note that $\{V_k\}$ takes value in the larger alphabet $\{aa, ab, ba, bb\}$ (i.e., $r' = 4$). To check how big the memory of this new process $\{V_k\}$ is, we compute (for simplicity of notation, we drop the PMF's arguments):

$$\begin{aligned} &P_{V_2|V_1,V_0,V_{-1},\ldots} \\ &\quad = P_{X_4,X_3|X_2,X_1,X_0,\ldots} && && (6.7)\\ &\quad = P_{X_3|X_2,X_1,X_0,\ldots} \cdot P_{X_4|X_3,X_2,X_1,X_0,\ldots} && \text{(by chain rule)} && (6.8)\\ &\quad = P_{X_3|X_2,X_1} \cdot P_{X_4|X_3,X_2,X_1} && \text{(by Markovity of } \{X_k\}) && (6.9)\\ &\quad = P_{X_4,X_3|X_2,X_1} && \text{(by chain rule)} && (6.10)\\ &\quad = P_{V_2|V_1}. && && (6.11) \end{aligned}$$

Hence, $\{V_k\}$ is again Markov, but of memory $\mu' = 1$! ◊

We usually will only consider Markov processes of memory 1 and simply call them *Markov processes*.

Definition 6.8. A Markov process is called *time-invariant* (or *homogeneous*) if the **conditional** probability distribution does not depend[1] on time k:

$$P_{X_k|X_{k-1}}(\alpha|\beta) = P_{X_2|X_1}(\alpha|\beta), \quad \forall\, k, \alpha, \beta. \tag{6.12}$$

[1] In the case of a Markov process of memory $\mu > 1$, this definition is accordingly adapted to $P_{X_k|X_{k-1},\ldots,X_{k-\mu}}(\alpha|\boldsymbol{\beta}) = P_{X_{\mu+1}|X_\mu,\ldots,X_1}(\alpha|\boldsymbol{\beta})$ for all k, α, $\boldsymbol{\beta}$.

We will always assume that a Markov process is time-invariant.

Remark 6.9. We make the following observations:

- For a time-invariant Markov process the transition probabilities from a certain state ℓ to a certain another state $\tilde{\ell}$ always remains the same: $p_{\ell,\tilde{\ell}}$.
- These transition probabilities $p_{\ell,\tilde{\ell}}$ are usually stored in the *transition probability matrix*

$$\mathsf{P} \triangleq \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,m} \end{pmatrix}. \tag{6.13}$$

- A time-invariant Markov process can easily be depicted by a *state diagram*, see the following Example 6.10. △

Example 6.10. Consider a binary source ($r = 2$) with alphabet $\{a, b\}$. Assume that

$$P_{X_2|X_1}(a|a) = \frac{1}{2}, \qquad P_{X_2|X_1}(a|b) = \frac{3}{4}, \tag{6.14}$$

$$P_{X_2|X_1}(b|a) = \frac{1}{2}, \qquad P_{X_2|X_1}(b|b) = \frac{1}{4}. \tag{6.15}$$

Then

$$\mathsf{P} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & \frac{1}{4} \end{pmatrix} \tag{6.16}$$

and the corresponding state transition diagram is shown in Figure 6.2. ◊

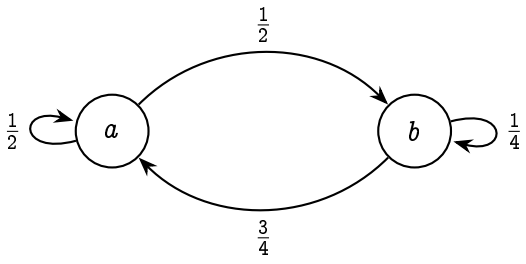


Figure 6.2: State transition diagram of a binary ($r = 2$) Markov source with transition probability matrix given in (6.16).

So we see that in order to uniquely define a Markov process, we certainly need to specify the set of all states $\mathcal{S}$ and the transition probabilities P, or

equivalently, the state transition diagram. But is this sufficient? Is a Markov process completely specified by states and transition probabilities?

We remember that a stochastic process is specified by the joint distribution of any subset of RVs (see Definition 6.1). So let us check the joint distribution of a Markov process $\{X_k\}$ and see if P fully specifies it. As an example, we pick three RVs:

$$\begin{aligned} P_{X_1,X_2,X_3}(\alpha,\beta,\gamma) &= P_{X_1}(\alpha)\cdot P_{X_2|X_1}(\beta|\alpha)\cdot P_{X_3|X_2,X_1}(\gamma|\beta,\alpha) && (6.17)\\ &= P_{X_1}(\alpha)\cdot P_{X_2|X_1}(\beta|\alpha)\cdot P_{X_3|X_2}(\gamma|\beta) && (6.18)\\ &= \underbrace{P_{X_1}(\alpha)}_{\text{not specified yet!}}\cdot\underbrace{P_{X_2|X_1}(\beta|\alpha)}_{\mathsf{P}}\cdot\underbrace{P_{X_2|X_1}(\gamma|\beta)}_{\mathsf{P}}. && (6.19)\end{aligned}$$

We see that the joint probability distribution is not completely specified: We are missing the starting distribution P_{X_1}!

Lemma 6.11 (Specification of a Markov Process). A time-invariant Markov process is completely specified by the following three items:

1. set of all states $\mathcal{S}$;
2. transition probabilities P;
3. starting distribution $P_{X_1}(\cdot)$.

Remark 6.12. Note that if we consider a Markov source of memory $\mu > 1$, then the transition probabilities are given by all possible values of

$$P_{X_{\mu+1}|X_\mu,\ldots,X_1}(\cdot|\cdot,\ldots,\cdot), \tag{6.20}$$

and the starting distribution corresponds to the distribution of the first μ time-steps:

$$P_{X_1,\ldots,X_\mu}(\cdot,\ldots,\cdot). \tag{6.21}$$

But, as mentioned, most of the time we will consider Markov processes of memory 1. △

A very interesting choice for the starting distribution is the so-called *steady-state distribution*:

Definition 6.13. A distribution $\pi(\cdot)$ that has the property that a transition will keep it unchanged,

$$\sum_\alpha \pi(\alpha)P_{X_k|X_{k-1}}(\beta|\alpha) = \pi(\beta), \quad \forall\,\beta \tag{6.22}$$

is called *steady-state distribution*. Usually, (6.22) is written in vector notation:[2]

$$\boldsymbol{\pi}^{\mathsf{T}} \cdot \mathsf{P} = \boldsymbol{\pi}^{\mathsf{T}}. \tag{6.23}$$

Example 6.14. Let us look at the example of Figure 6.2. What is the steady-state distribution? We write down (6.23) explicitly:

$$\begin{aligned} P_{X_2}(a) &= P_{X_1}(a) \cdot P_{X_2|X_1}(a|a) + P_{X_1}(b) \cdot P_{X_2|X_1}(a|b) && (6.24)\\ &= \pi_a \cdot \frac{1}{2} + \pi_b \cdot \frac{3}{4} \overset{!}{=} \pi_a, && (6.25)\\ P_{X_2}(b) &= P_{X_1}(a) \cdot P_{X_2|X_1}(b|a) + P_{X_1}(b) \cdot P_{X_2|X_1}(b|b) && (6.26)\\ &= \pi_a \cdot \frac{1}{2} + \pi_b \cdot \frac{1}{4} \overset{!}{=} \pi_b. && (6.27) \end{aligned}$$

Together with the normalization $\pi_a + \pi_b = 1$, this equation system can be solved:

$$\begin{aligned} \pi_a &= \frac{3}{5}, && (6.28)\\ \pi_b &= \frac{2}{5}. && (6.29) \end{aligned}$$

◊

We have learned the following.

Lemma 6.15. The steady-state distribution $\boldsymbol{\pi}$ can be computed by the **balance equations**:

$$\boldsymbol{\pi}^{\mathsf{T}} = \boldsymbol{\pi}^{\mathsf{T}} \cdot \mathsf{P} \tag{6.30}$$

and the **normalization equation**:

$$\sum_{j=1}^{m} \pi_j = 1. \tag{6.31}$$

We will now state the most important result concerning Markov processes that explains why they are so popular in engineering. We will only state the results and omit the proofs. They can be found in many textbooks about probability, e.g., in [BT02].

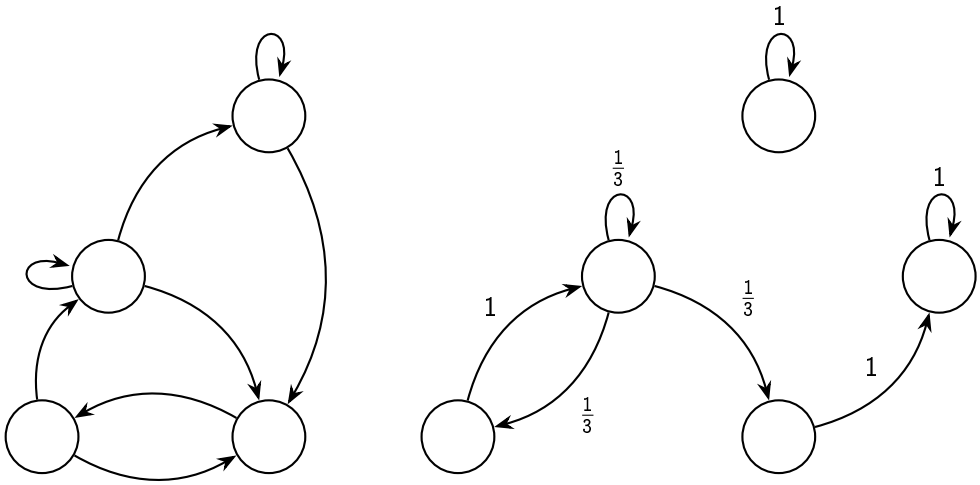
We first need two more definitions.

Definition 6.16. A Markov process is called *irreducible* if it is possible (i.e., with positive probability) to go from any state to any other state in a finite number of steps.

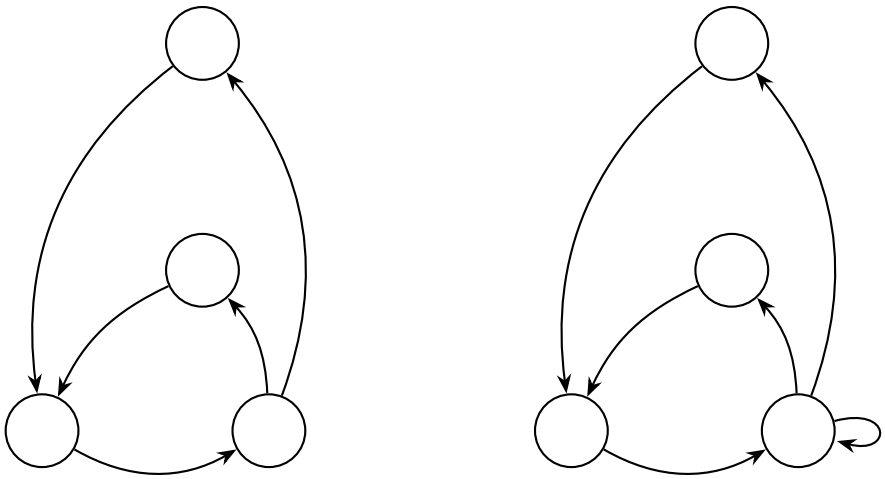
[2] Note again that in the case of Markov processes with memory $\mu > 1$ the steady-state distribution is a joint distribution of μ variables $\pi(\cdot,\ldots,\cdot) = P_{X_k,\ldots,X_{k-\mu+1}}(\cdot,\ldots,\cdot)$. In this case we will not use the vector notation.

Definition 6.17. An irreducible Markov process is called *periodic* if the states can be grouped into disjoint subsets so that all transitions from one subset lead to the next subset.

Figure 6.3 shows some examples of irreducible, reducible, periodic and aperiodic Markov processes, respectively.



irreducible reducible



periodic aperiodic

Figure 6.3: Examples of irreducible, reducible, periodic and aperiodic Markov processes, respectively.

And now we are ready for the most important theorem concerning Markov processes.

Theorem 6.18 (Markovity & Stationarity).
Let us consider a time-invariant Markov process that is irreducible and aperiodic. Then

1. the steady-state distribution $\pi(\cdot)$ is unique;
2. independently of the starting distribution $P_{X_1}(\cdot)$, the distribution $P_{X_k}(\cdot)$ will converge to the steady-state distribution $\pi(\cdot)$ for $k \to \infty$;
3. the Markov process is stationary if, and only if, the starting distribution $P_{X_1}(\cdot)$ is chosen to be the steady-state distribution $\pi(\cdot)$:

$$P_{X_1}(\alpha) = \pi(\alpha), \quad \forall\, \alpha. \tag{6.32}$$

Table 6.4: The state probability distribution of a Markov source converges to its steady-state distribution.

$P_{X_k}(\cdot)$	$k=1$	$k=2$	$k=3$	$k=4$
$\Pr[X_k=a]$	1	$\frac{1}{2}=0.5$	$\frac{5}{8}=0.625$	$\frac{19}{32}=0.59375$
$\Pr[X_k=b]$	0	$\frac{1}{2}=0.5$	$\frac{3}{8}=0.375$	$\frac{13}{32}=0.40625$

$P_{X_k}(\cdot)$	$k=5$	...	$k=\infty$
$\Pr[X_k=a]$	$\frac{77}{128}=0.6015625$	...	$\frac{3}{5}=0.6$
$\Pr[X_k=b]$	$\frac{51}{128}=0.3984375$	...	$\frac{2}{5}=0.4$

Example 6.19. Returning to the example of Figure 6.2, we already know that $\boldsymbol{\pi} = \left(\frac{3}{5}, \frac{2}{5}\right)^{\mathsf{T}}$. By Theorem 6.18 we know that if we start in state a, after a while we will have a probability of about $\frac{3}{5}$ to be in state a and a probability of about $\frac{2}{5}$ to be in state b. This is shown in Table 6.4. ◊

Remark 6.20. Note that *time-invariance* and *stationarity* are not the same thing! For stationarity it is necessary that a Markov is time-invariant, however, a time-invariant Markov process is not necessarily stationary! △

6.3 Entropy Rate

So now that we have defined general sources with memory, the question arises what the "uncertainty" of such a source is, i.e., what is the "entropy" $\mathrm{H}(\{X_k\})$ of a stochastic process $\{X_k\}$?

Example 6.21. If $\{X_k\}$ is IID, then defining $\mathrm{H}(\{X_k\}) = \mathrm{H}(X_1)$ would make sense as this corresponds to the situation we have considered so far. On the other hand, if the source has memory, this choice will lead to a rather strange result. To see this take as an example $P_{X_1}(0) = P_{X_1}(1) = \frac{1}{2}$, i.e., $\mathrm{H}(X_1) = 1$ bit. Assume further that

$$\begin{aligned} P_{X_2|X_1}(0|0) &= 0, & P_{X_2|X_1}(1|0) &= 1,\\ P_{X_2|X_1}(0|1) &= 0, & P_{X_2|X_1}(1|1) &= 1, \end{aligned} \tag{6.33}$$

(see Figure 6.5). From this we can now compute $P_{X_2}(1) = 1$, which means

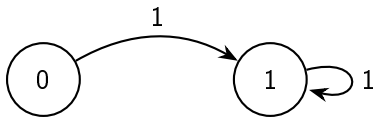


Figure 6.5: Transition diagram of a source with memory.

that $\mathrm{H}(X_2) = 0$, $\mathrm{H}(X_2|X_1) = 0$, and $\mathrm{H}(X_1, \ldots, X_n) = 1$ bit. So it is clear that it is definitely not correct to define the "entropy" to be 1 bit, as only the first step contains any uncertainty and then we know for sure that the source remains in state 1 forever. ◊

Latest by now it should be obvious that the amount of uncertainty coming from a source with memory strongly depends on the memory.

We give the following definition.

Definition 6.22. The *entropy rate* (i.e., the entropy per source symbol) of any stochastic process $\{X_k\}$ is defined as

$$\mathrm{H}(\{X_k\}) \triangleq \lim_{n\to\infty} \frac{\mathrm{H}(X_1, X_2, \ldots, X_n)}{n} \tag{6.34}$$

if the limit exists.

Remark 6.23. Unfortunately, this limit does not always exist, as can be seen from the following example. Let $\{X_k\}$ be independent, but for each k the

distribution is different:

$$\Pr[X_k=1]=\begin{cases}\frac{1}{2} & \text{if } k=1,\\ \frac{1}{2} & \text{if } \lceil \log_2 \log_2 k\rceil \text{ is even } (k\geq 2),\\ 0 & \text{if } \lceil \log_2 \log_2 k\rceil \text{ is odd } (k\geq 2),\end{cases} \tag{6.35}$$

$$\Pr[X_k=0]=1-\Pr[X_k=1]. \tag{6.36}$$

Then we see that

$$\mathrm{H}(X_k)=1 \text{ bit} \quad \text{for } k=1,\ 2\ (=2^{2^0}),\ 5\text{–}16\ (=2^{2^2}),\ 257\text{–}65536\ (=2^{2^4}),\ (2^{2^5}+1)\text{–}2^{2^6},\ \ldots \tag{6.37}$$

$$\mathrm{H}(X_k)=0 \text{ bits} \quad \text{for } k=3,\ 4\ (=2^{2^1}),\ 17\text{–}256\ (=2^{2^3}),\ 65537\text{–}2^{2^5},\ (2^{2^6}+1)\text{–}2^{2^7},\ \ldots \tag{6.38}$$

The changing points are at $2^{2^0}, 2^{2^1}, 2^{2^2}, 2^{2^3}, 2^{2^4}, 2^{2^5}, 2^{2^6}, 2^{2^7}, \ldots$ If we now look at

$$f(n)\triangleq \frac{1}{n}\mathrm{H}(X_1,\ldots,X_n)=\frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(X_k), \tag{6.39}$$

we note that $f(1)=f(2)=1$ bit, then it will decrease $f(3)=\frac{2}{3}$ bits, $f(4)=\frac{2}{4}=\frac{1}{2}$ bits, then it will increase again $f(5)=\frac{3}{5}$ bits, $f(6)=\frac{4}{6}$ bits, $\ldots$, until the next changing point $f(16)=\frac{14}{16}$ bits, where it turns again and decreases, etc. So, $f(n)$ is continuously alternating between increasing and slowly tending towards 1 bit and decreasing and slowly tending towards 0. Now note that the intervals between the changing points are growing double-exponentially, so that the time between each change is getting larger and larger. Therefore, $f(n)$ has enough "time" to actually approach the corresponding limit 0 or 1 closer and closer each time, before turning away again. Hence, we see that $f(\cdot)$ oscillates between 0 and 1 forever and does not converge. △

Question: Why not define the entropy rate as

$$\tilde{\mathrm{H}}(\{X_k\})\triangleq \lim_{n\to\infty}\mathrm{H}(X_n|X_{n-1},X_{n-2},\ldots,X_1)? \tag{6.40}$$

The (partial) answer lies in the following theorem.

Theorem 6.24 (Entropy Rate of a DSS).
For a stationary stochastic process (or DSS) the entropy rate $\mathrm{H}(\{X_k\})$ always exists and its value is identical to $\tilde{\mathrm{H}}(\{X_k\})$:

$$\mathrm{H}(\{X_k\})=\lim_{n\to\infty}\frac{\mathrm{H}(X_1,\ldots,X_n)}{n}=\lim_{n\to\infty}\mathrm{H}(X_n|X_{n-1},\ldots,X_1). \tag{6.41}$$

Furthermore,

1. $\mathrm{H}(X_n|X_{n-1},\ldots,X_1)$ is nonincreasing in n;
2. $\frac{1}{n}\mathrm{H}(X_1,\ldots,X_n)$ is nonincreasing in n;
3. $\mathrm{H}(X_n|X_{n-1},\ldots,X_1) \le \frac{1}{n}\mathrm{H}(X_1,\ldots,X_n), \quad \forall\, n \ge 1.$

Before we can prove this important result, we need another small lemma.

Lemma 6.25 (Cesáro Mean). If

$$\lim_{n\to\infty} a_n = a \tag{6.42}$$

then

$$b_n \triangleq \frac{1}{n}\sum_{k=1}^{n} a_k \to a \tag{6.43}$$

as $n \to \infty$.

Proof: By assumption, $a_n \to a$. This — in mathematical language — means: $\forall\,\epsilon > 0$, $\exists\ \mathrm{N}_\epsilon$ such that $|a_n - a| \le \epsilon$, for $n > \mathrm{N}_\epsilon$. We now want to show that for any $\epsilon_1 > 0$ and for n large enough we have

$$|b_n - a| \le \epsilon_1. \tag{6.44}$$

To this end, note that for $n > \mathrm{N}_\epsilon$

$$\begin{aligned}
|b_n - a| &= \left|\frac{1}{n}\sum_{k=1}^{n} a_k - a\right| && (6.45)\\
&= \left|\frac{1}{n}\sum_{k=1}^{n} a_k - \frac{1}{n}\sum_{k=1}^{n} a\right| && (6.46)\\
&= \left|\frac{1}{n}\sum_{k=1}^{n} (a_k - a)\right| && (6.47)\\
&\le \frac{1}{n}\sum_{k=1}^{n} |a_k - a| && (6.48)\\
&= \frac{1}{n}\sum_{k=1}^{\mathrm{N}_\epsilon} |a_k - a| + \frac{1}{n}\sum_{k=\mathrm{N}_\epsilon+1}^{n} \underbrace{|a_k - a|}_{\le \epsilon} && (6.49)\\
&\le \frac{1}{n}\underbrace{\sum_{k=1}^{\mathrm{N}_\epsilon} |a_k - a|}_{\text{constant}} + \underbrace{\frac{n-\mathrm{N}_\epsilon}{n}}_{\le 1}\cdot\epsilon && (6.50)\\
&\le \epsilon + \epsilon = 2\epsilon \triangleq \epsilon_1, \quad \text{for } n \text{ large enough}, && (6.51)
\end{aligned}$$

where the inequality (6.48) follows from the Triangle Inequality. Since $\epsilon > 0$ is arbitrary, also ϵ_1 is arbitrary and therefore the claim is proven. □

Proof of Theorem 6.24: We start with the second part of the Theorem and there with a proof of Part 1:

$$\begin{aligned} \mathrm{H}(X_n|X_{n-1},\dots,X_1) &= \mathrm{H}(X_{n+1}|X_n,\dots,X_2) && (6.52)\\ &\ge \mathrm{H}(X_{n+1}|X_n,\dots,X_2,X_1) && (6.53) \end{aligned}$$

where (6.52) follows from stationarity and (6.53) follows from conditioning that reduces entropy.

To prove Part 3 we rely on the chain rule and on Part 1:

$$\begin{aligned} \frac{1}{n}\mathrm{H}(X_1,\dots,X_n) &= \frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(X_k|X_{k-1},\dots,X_1) \qquad \text{(chain rule)} && (6.54)\\ &\ge \frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(X_n|X_{n-1},\dots,X_1) \qquad \text{(Part 1)} && (6.55)\\ &= \mathrm{H}(X_n|X_{n-1},\dots,X_1). && (6.56) \end{aligned}$$

To prove Part 2 we rely on the chain rule (equality in (6.57)), on Part 1 (inequality in (6.58)), and on Part 3 (inequality in (6.59)):

$$\begin{aligned} \mathrm{H}(X_1,\dots,X_n,X_{n+1}) &= \mathrm{H}(X_1,\dots,X_n)+\mathrm{H}(X_{n+1}|X_n,\dots,X_1) && (6.57)\\ &\le \mathrm{H}(X_1,\dots,X_n)+\mathrm{H}(X_n|X_{n-1},\dots,X_1) && (6.58)\\ &\le \mathrm{H}(X_1,\dots,X_n)+\frac{1}{n}\mathrm{H}(X_1,\dots,X_n) && (6.59)\\ &= \frac{n+1}{n}\mathrm{H}(X_1,\dots,X_n), && (6.60) \end{aligned}$$

i.e.,

$$\frac{1}{n+1}\mathrm{H}(X_1,\dots,X_{n+1}) \le \frac{1}{n}\mathrm{H}(X_1,\dots,X_n). \tag{6.61}$$

Finally, to prove (6.41) note that by Parts 1 and 2, both $\mathrm{H}(X_n|X_{n-1},\dots,X_1)$ and $\frac{1}{n}\mathrm{H}(X_1,\dots,X_n)$ are nonincreasing, but bounded below by zero (because entropies are always nonnegative). Hence, they must converge. This shows that the limit exists. Now it only remains to show that they converge to the same limit. To this end, we rely on the Cesáro mean (Lemma 6.25):

$$\begin{aligned} \mathrm{H}(\{X_k\}) &= \lim_{n\to\infty}\frac{1}{n}\mathrm{H}(X_1,\dots,X_n) && (6.62)\\ &= \lim_{n\to\infty}\overbrace{\frac{1}{n}\sum_{k=1}^{n}\underbrace{\mathrm{H}(X_k|X_{k-1},\dots,X_1)}_{\triangleq a_k}}^{\triangleq b_n} && (6.63)\\ &= \lim_{n\to\infty}\mathrm{H}(X_n|X_{n-1},\dots,X_1) \qquad \text{(Lemma 6.25)} && (6.64)\\ &= \tilde{\mathrm{H}}(\{X_k\}). && (6.65) \end{aligned}$$

□

Note the following direct consequence of Theorem 6.24.

Corollary 6.26. For a stationary stochastic process $\{X_k\}$ it holds that for any $n \in \mathbb{N}$,

$$\mathrm{H}(X_n|X_{n-1},\dots,X_1) \le \frac{1}{n}\,\mathrm{H}(X_1,\dots,X_n) \le \mathrm{H}(X_1). \tag{6.66}$$

Proof: The first inequality simply is repeated from Theorem 6.24 (Part 3), and the second inequality holds because $\frac{1}{n}\,\mathrm{H}(X_1,\dots,X_n)$ is nonincreasing and thus maximized for $n = 1$. □

Beside stationary processes we can also prove for a Markov process that the entropy rate always is defined (even if the Markov process is not stationary).

Theorem 6.27 (Entropy Rate of a Markov Source).
For a time-invariant, irreducible, and aperiodic Markov process $\{X_k\}$ of memory μ the entropy rate $\mathrm{H}(\{X_k\})$ always exists and its value is identical to $\tilde{\mathrm{H}}(\{X_k\})$. In particular, the entropy rate can be computed as

$$\mathrm{H}(\{X_k\}) = \mathrm{H}(X_{\mu+1}|X_\mu,\dots,X_1)\Big|_{P_{X_\mu,\dots,X_1}(\cdot,\dots,\cdot)=\pi(\cdot,\dots,\cdot)} \tag{6.67}$$

where, when computing the conditional entropy $\mathrm{H}(X_{\mu+1}|X_\mu,\dots,X_1)$, we need to choose $P_{X_\mu,\dots,X_1}(\cdot,\dots,\cdot)$ to be the steady-state distribution $\pi(\cdot,\dots,\cdot)$.

In the case where the memory has length $\mu = 1$, this simplifies to

$$\mathrm{H}(\{X_k\}) = \mathrm{H}(X_2|X_1)\big|_{P_{X_1}(\cdot)=\pi(\cdot)} \tag{6.68}$$

with $\pi(\cdot) = \boldsymbol{\pi}$ denoting the steady-state distribution.

Thus we see that the entropy rate of a Markov process does *not* depend on the starting distribution, but only on the transition probabilities (and the steady-state distribution that is computed from the transition probabilities).

Remark 6.28. Before we prove Theorem 6.27, we would like to give a warning: The reader should be aware that even though we assume a time-invariant Markov process, i.e., we assume that

$$P_{X_n|X_{n-1},\dots,X_{n-\mu}}(\alpha|\boldsymbol{\beta}) = P_{X_{\mu+1}|X_\mu,\dots,X_1}(\alpha|\boldsymbol{\beta}) \tag{6.69}$$

for all $\alpha \in \mathcal{X}$ and $\boldsymbol{\beta} \in \mathcal{X}^\mu$, we have

$$\mathrm{H}(X_n|X_{n-1},\dots,X_{n-\mu}) \neq \mathrm{H}(X_{\mu+1}|X_\mu,\dots,X_1) \tag{6.70}$$

in general! The reason lies in the fact that for the computation of the conditional entropy $\mathrm{H}(X_n|X_{n-1},\dots,X_{n-\mu})$ it is not sufficient to know merely

the conditional PMF $P_{X_n|X_{n-1},\ldots,X_{n-\mu}}$, but we actually need the joint PMF $P_{X_{n-\mu},\ldots,X_n}$. This joint PMF, however, depends on n. The inequality in (6.70) can only be replaced by equality if the starting distribution equals to the steady-state distribution, i.e., only if the Markov process happens to be stationary. △

Proof of Theorem 6.27: The basic idea behind this result is the fact that by Theorem 6.18, a time-invariant, irreducible, and aperiodic Markov process will always converge to the steady-state distribution. A Markov process based on the steady-state distribution, however, looks stationary and therefore should have an entropy rate that is well-defined.

We start with $\tilde{\mathrm{H}}(\{X_k\})$:

$$\begin{aligned}
\tilde{\mathrm{H}}(\{X_k\}) &= \lim_{n\to\infty} \mathrm{H}(X_n|X_{n-1},\ldots,X_1) && (6.71)\\
&= \lim_{n\to\infty} \mathrm{H}(X_n|X_{n-1},\ldots,X_{n-\mu}) && (6.72)\\
&= \lim_{n\to\infty} \mathrm{H}(X_n|X_{n-\mu},\ldots,X_{n-1}) && (6.73)\\
&= \lim_{n\to\infty} \sum_{\boldsymbol{\alpha}} P_{X_{n-\mu},\ldots,X_{n-1}}(\boldsymbol{\alpha})\, \mathrm{H}\big(X_n\big|(X_{n-\mu},\ldots,X_{n-1})=\boldsymbol{\alpha}\big) && (6.74)\\
&= \lim_{n\to\infty} \sum_{\boldsymbol{\alpha}} P_{X_{n-\mu},\ldots,X_{n-1}}(\boldsymbol{\alpha})\, \mathrm{H}\big(X_{\mu+1}\big|(X_1,\ldots,X_\mu)=\boldsymbol{\alpha}\big) && (6.75)\\
&= \sum_{\boldsymbol{\alpha}} \underbrace{\Big(\lim_{n\to\infty} P_{X_{n-\mu},\ldots,X_{n-1}}(\boldsymbol{\alpha})\Big)}_{=\pi(\boldsymbol{\alpha})} \mathrm{H}\big(X_{\mu+1}\big|(X_1,\ldots,X_\mu)=\boldsymbol{\alpha}\big) && (6.76)\\
&= \sum_{\boldsymbol{\alpha}} \pi(\boldsymbol{\alpha})\, \mathrm{H}\big(X_{\mu+1}\big|(X_1,\ldots,X_\mu)=\boldsymbol{\alpha}\big) && (6.77)\\
&= \mathrm{H}(X_{\mu+1}|X_1,\ldots,X_\mu)\Big|_{P_{X_1,\ldots,X_\mu}(\cdot,\ldots,\cdot)=\pi(\cdot,\ldots,\cdot)}. && (6.78)
\end{aligned}$$

Here, (6.72) follows from Markovity; in (6.73) we rearrange the order of $X_{n-\mu},\ldots,X_{n-1}$ for notational reasons; (6.74) is due to definition of conditional entropy; (6.75) follows from time-invariance; in (6.76) we take the limit into the sum since only $P_{X_{n-1},\ldots,X_{n-\mu}}$ depends on n; (6.77) is the crucial step: Here we use the fact that for a time-invariant, irreducible, and aperiodic Markov process of memory μ, the distribution $P_{X_{n-1},\ldots,X_{n-\mu}}$ converges to the steady-state distribution $\boldsymbol{\pi}$; and in (6.78) we again use the definition of conditional entropy. This proves the first part.

Next we prove that the entropy rate is identical to the above:

$$\begin{aligned}
&\mathrm{H}(\{X_k\})\\
&\quad= \lim_{n\to\infty} \frac{1}{n}\, \mathrm{H}(X_1,\ldots,X_n) && (6.79)\\
&\quad= \lim_{n\to\infty} \frac{1}{n} \big(\mathrm{H}(X_{\mu+1},\ldots,X_n|X_1,\ldots,X_\mu) + \mathrm{H}(X_1,\ldots,X_\mu)\big) && (6.80)\\
&\quad= \lim_{n\to\infty} \frac{n-\mu}{n-\mu}\cdot\frac{1}{n}\, \mathrm{H}(X_{\mu+1},\ldots,X_n|X_1,\ldots,X_\mu)
\end{aligned}$$

$$+ \underbrace{\lim_{n\to\infty} \frac{1}{n} \mathrm{H}(X_1,\ldots,X_\mu)}_{=0} \tag{6.81}$$

$$= \lim_{n\to\infty} \underbrace{\frac{n-\mu}{n}}_{\to 1} \cdot \frac{1}{n-\mu} \mathrm{H}(X_{\mu+1},\ldots,X_n|X_1,\ldots,X_\mu) \tag{6.82}$$

$$= \lim_{n\to\infty} \frac{1}{n-\mu} \sum_{k=\mu+1}^{n} \mathrm{H}(X_k|X_1,\ldots,X_{k-1}) \tag{6.83}$$

$$= \lim_{n\to\infty} \frac{1}{n-\mu} \sum_{k=\mu+1}^{n} \mathrm{H}(X_k|X_{k-\mu},\ldots,X_{k-1}) \tag{6.84}$$

$$= \lim_{n\to\infty} \frac{1}{n-\mu} \sum_{k=\mu+1}^{n} \sum_{\boldsymbol{\alpha}} P_{X_{k-\mu},\ldots,X_{k-1}}(\boldsymbol{\alpha})\, \mathrm{H}(X_k\big|(X_{k-\mu},\ldots,X_{k-1})=\boldsymbol{\alpha}) \tag{6.85}$$

$$= \lim_{n\to\infty} \sum_{\boldsymbol{\alpha}} \frac{1}{n-\mu} \sum_{k=\mu+1}^{n} P_{X_{k-\mu},\ldots,X_{k-1}}(\boldsymbol{\alpha})\, \mathrm{H}(X_{\mu+1}\big|(X_1,\ldots,X_\mu)=\boldsymbol{\alpha}) \tag{6.86}$$

$$= \sum_{\boldsymbol{\alpha}} \underbrace{\left(\lim_{n\to\infty} \frac{1}{n-\mu} \sum_{k=\mu+1}^{n} P_{X_{k-\mu},\ldots,X_{k-1}}(\boldsymbol{\alpha}) \right)}_{=\pi(\boldsymbol{\alpha})} \mathrm{H}(X_{\mu+1}\big|(X_1,\ldots,X_\mu)=\boldsymbol{\alpha}) \tag{6.87}$$

$$= \sum_{\boldsymbol{\alpha}} \pi(\boldsymbol{\alpha})\, \mathrm{H}(X_{\mu+1}\big|(X_1,\ldots,X_\mu)=\boldsymbol{\alpha})) \tag{6.88}$$

$$= \mathrm{H}(X_{\mu+1}|X_1,\ldots,X_\mu)\Big|_{P_{X_1,\ldots,X_\mu}(\cdot,\ldots,\cdot)=\pi(\cdot,\ldots,\cdot)}. \tag{6.89}$$

Here, (6.80) follows from the chain rule; in (6.82) we use that $\mathrm{H}(X_1,\ldots,X_\mu)$ is independent of n and finite; (6.83) again follows from the chain rule; (6.84) from Markovity; in (6.86) we swap the order of finite summations and use the time-invariance of the Markov process; in (6.87) we use the fact that $\mathrm{H}(X_{\mu+1}|(X_1,\ldots,X_\mu)=\boldsymbol{\alpha})$ does not depend on k or n; and (6.88) again follows because for a time-invariant, irreducible, and aperiodic Markov process of memory μ the distribution $P_{X_{k-\mu},\ldots,X_{k-1}}$ converges to the steady-state distribution $\boldsymbol{\pi}$ and because of the Cesáro mean (Lemma 6.25). □

Chapter 7

Data Compression: Efficient Coding of a Random Source with Memory

In Chapter 6 we have introduced some types of sources with memory. Now, we will start thinking about ways of how to compress them.

7.1 Block–to–Variable-Length Coding of a DSS

We start with a reminder.

Remark 7.1. An r-ary *discrete stationary source (DSS)* $\{U_k\}$ is a device that emits a sequence $U_1, U_2, \ldots$ of r-ary RVs such that this sequence is a stationary stochastic process. Hence, every DSS has an entropy rate

$$\mathrm{H}(\{U_k\}) = \lim_{n\to\infty} \frac{1}{n} \mathrm{H}(U_1, \ldots, U_n) = \lim_{n\to\infty} \mathrm{H}(U_n | U_{n-1}, \ldots, U_1). \tag{7.1}$$

△

For simplicity, at the moment we will only consider block parsing of the output of the DSS, but variable-length parsers are also often used in practice. So, we consider now the system shown in Figure 7.1.

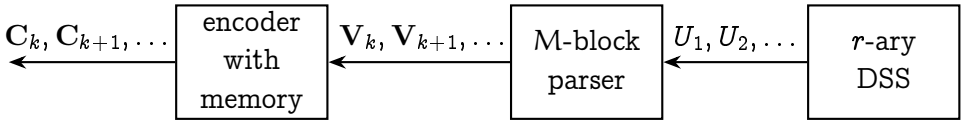


Figure 7.1: A general compression scheme for a source with memory: The D-ary code is, as usual, assumed to be prefix-free, but the encoder is allowed to have memory. For the moment, we assume an M-block parser.

We note that since the source has memory, the messages have memory, too, i.e., the different $\mathbf{V}_k$ depend on each other. Hence, to make the encoder

efficient, it should use this additional information provided by the memory! This means that we allow the encoder to take previous messages into account when encoding the message $\mathbf{V}_k$. In principle, the encoder could use a different prefix-free code for each possible sequence of values for the previous messages $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_{k-1}$. Note that at the time when the decoder should decode the codeword $\mathbf{C}_k$, it already has decoded $\mathbf{C}_1, \ldots, \mathbf{C}_{k-1}$ to $\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}$. Hence, the decoder also knows the previous messages and therefore the code used for $\mathbf{V}_k$. It therefore can decode $\mathbf{C}_k$.

Claim 7.2 (Optimal Block–to–Variable-Length Coding of a DSS). We consider an M-block parser that generates the block messages

$$\mathbf{V}_k = (\underbrace{U_{k\mathrm{M}-\mathrm{M}+1}, \ldots, U_{k\mathrm{M}-1}, U_{k\mathrm{M}}}_{\text{length } \mathrm{M}}), \quad \forall\, k. \tag{7.2}$$

For every k, we take the conditional distribution of the current message $\mathbf{V}_k$ conditional on the past $(\mathbf{V}_{k-1}, \ldots, \mathbf{V}_1) = (\mathbf{v}_{k-1}, \ldots, \mathbf{v}_1)$

$$P_{\mathbf{V}_k|\mathbf{V}_{k-1},\ldots,\mathbf{V}_1}(\cdot|\mathbf{v}_{k-1}, \ldots, \mathbf{v}_1) \tag{7.3}$$

to design an optimal D-ary Huffman code for $\mathbf{V}_k$ and then encode $\mathbf{V}_k$ with this code. Note that this Huffman code will be different for each k.

The receiver also knows the past $(\mathbf{v}_{k-1}, \ldots, \mathbf{v}_1)$ because it has decoded that beforehand, and therefore also knows which code has been used to encode $\mathbf{V}_k$, and hence can decode it.

Such a system is called **adaptive Huffman coding**.

We immediately realize that adaptive Huffman coding is not a practical system because the effort involved is rather big: We have to redesign our code for *every* message!

Nevertheless we will quickly analyze the performance of such an adaptive Huffman coding system. The analysis is quite straightforward as we can base it on our knowledge of the performance of a Huffman code. From the coding theorem for a single random message (Theorem 4.38) it follows that the length L_k of the codeword $\mathbf{C}_k$ for the message $\mathbf{V}_k$ must satisfy

$$\begin{aligned}
&\frac{\mathrm{H}(\mathbf{V}_k|(\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}) = (\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}))}{\log \mathrm{D}} \\
&\quad \le \mathsf{E}[L_k \,|\, (\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}) = (\mathbf{v}_1, \ldots, \mathbf{v}_{k-1})] &(7.4)\\
&\quad < \frac{\mathrm{H}(\mathbf{V}_k|(\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}) = (\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}))}{\log \mathrm{D}} + 1. &(7.5)
\end{aligned}$$

We first investigate the lower bound (7.4). Taking the expectation over $\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}$ (i.e., multiplying the expression by $P_{\mathbf{V}_1,\ldots,\mathbf{V}_{k-1}}(\mathbf{v}_1, \ldots, \mathbf{v}_{k-1})$

and summing over all values $\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}$) gives

$$\begin{aligned}
\mathsf{E}[L_k] &= \mathsf{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_{k-1}}[\mathsf{E}[L_k \,|\, \mathbf{V}_1, \ldots, \mathbf{V}_{k-1}]] && (7.6)\\
&= \sum_{\mathbf{v}_1,\ldots,\mathbf{v}_{k-1}} P_{\mathbf{V}_1,\ldots,\mathbf{V}_{k-1}}(\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}) \\
&\qquad \cdot \mathsf{E}[L_k \,|\, (\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}) = (\mathbf{v}_1, \ldots, \mathbf{v}_{k-1})] && (7.7)\\
&\geq \sum_{\mathbf{v}_1,\ldots,\mathbf{v}_{k-1}} P_{\mathbf{V}_1,\ldots,\mathbf{V}_{k-1}}(\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}) \\
&\qquad \cdot \frac{\mathrm{H}(\mathbf{V}_k | (\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}) = (\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}))}{\log \mathrm{D}} && (7.8)\\
&= \frac{\mathsf{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_{k-1}}[\mathrm{H}(\mathbf{V}_k | (\mathbf{V}_1, \ldots, \mathbf{V}_{k-1}) = (\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}))]}{\log \mathrm{D}} && (7.9)\\
&= \frac{\mathrm{H}(\mathbf{V}_k | \mathbf{V}_1, \ldots, \mathbf{V}_{k-1})}{\log \mathrm{D}} && (7.10)\\
&= \frac{\mathrm{H}(U_{(k-1)\mathrm{M}+1}, \ldots, U_{k\mathrm{M}} | U_1, \ldots, U_{\mathrm{M}}, \ldots, U_{(k-2)\mathrm{M}+1}, \ldots, U_{(k-1)\mathrm{M}})}{\log \mathrm{D}} && (7.11)\\
&= \frac{\mathrm{H}(U_{(k-1)\mathrm{M}+1} | U_1, \ldots, U_{(k-1)\mathrm{M}}) + \cdots + \mathrm{H}(U_{k\mathrm{M}} | U_1, \ldots, U_{k\mathrm{M}-1})}{\log \mathrm{D}} && (7.12)\\
&\geq \frac{\mathrm{M} \cdot \mathrm{H}(U_{k\mathrm{M}} | U_1, \ldots, U_{k\mathrm{M}-1})}{\log \mathrm{D}} && (7.13)\\
&\geq \frac{\mathrm{M}\, \mathrm{H}(\{U_k\})}{\log \mathrm{D}}. && (7.14)
\end{aligned}$$

Here, (7.6) follows from the law of total expectation; (7.8) follows from (7.4); in (7.11) we use the definition of the messages (7.2); the subsequent equality (7.12) follows from the chain rule; and in the subsequent inequalities (7.13) and (7.14) we use that $\mathrm{H}(U_j | U_{j-1}, \ldots, U_1)$ is monotonically nonincreasing and converges to the entropy rate $\mathrm{H}(\{U_k\})$ (Theorem 6.24). Hence, we have

$$\frac{\mathsf{E}[L_k]}{\mathrm{M}} \geq \frac{\mathrm{H}(\{U_k\})}{\log \mathrm{D}}. \tag{7.15}$$

Similarly, again using the monotonicity of $\mathrm{H}(U_j | U_{j-1}, \ldots, U_1)$, we obtain from (7.5)

$$\begin{aligned}
\mathsf{E}[L_k] &< \frac{\mathrm{H}(U_{(k-1)\mathrm{M}+1} | U_1, \ldots, U_{(k-1)\mathrm{M}}) + \cdots + \mathrm{H}(U_{k\mathrm{M}} | U_1, \ldots, U_{k\mathrm{M}-1})}{\log \mathrm{D}} + 1 && (7.16)\\
&\leq \frac{\mathrm{M}\, \mathrm{H}(U_{(k-1)\mathrm{M}+1} | U_1, \ldots, U_{(k-1)\mathrm{M}})}{\log \mathrm{D}} + 1, && (7.17)
\end{aligned}$$

i.e.,

$$\frac{\mathsf{E}[L_k]}{\mathrm{M}} < \frac{\mathrm{H}(U_{(k-1)\mathrm{M}+1} | U_1, \ldots, U_{(k-1)\mathrm{M}})}{\log \mathrm{D}} + \frac{1}{\mathrm{M}}. \tag{7.18}$$

Since $\mathrm{H}\big(U_{(k-1)\mathrm{M}+1}\big|U_1,\ldots,U_{(k-1)\mathrm{M}}\big)$ converges monotonically to $\mathrm{H}(\{U_k\})$, for every $\epsilon>0$ there must exist some M_0 large enough such that for all $\mathrm{M}\ge\mathrm{M}_0$, we have

$$\frac{\mathrm{E}[L_k]}{\mathrm{M}}<\frac{\mathrm{H}(\{U_k\})}{\log \mathrm{D}}+\epsilon. \tag{7.19}$$

We summarize this in the following theorem.

Theorem 7.3 (Block–to–Variable-Length Coding Theorem for a DSS).
For every $\epsilon>0$, one can choose an $\mathrm{M}\in\mathbb{N}$ large enough such that there exists a D-ary prefix-free code for the kth M-block message of a DSS $\{U_k\}$ with an average codeword length per source letter satisfying

$$\frac{\mathrm{E}[L_k]}{\mathrm{M}}<\frac{\mathrm{H}(\{U_k\})}{\log \mathrm{D}}+\epsilon. \tag{7.20}$$

Conversely, every uniquely decodable D-ary code of the kth M-block message of a DSS $\{U_k\}$ has an average codeword length per source symbol that satisfies

$$\frac{\mathrm{E}[L_k]}{\mathrm{M}}\ge\frac{\mathrm{H}(\{U_k\})}{\log \mathrm{D}}. \tag{7.21}$$

This holds even if we allow the code to depend on previous messages.

The alert reader might object that strictly speaking we have not proven that adaptive Huffman coding indeed is an optimal approach. However, we hope that our argument of choosing the optimal code at every step based on the complete knowledge at that time is convincing. Readers who would like to get more insight are referred to Section 20.8.

In spite of the success of having derived Theorem 7.3, we still feel uneasy because an adaptive Huffman coding scheme is completely infeasible in practice. Moreover, we also have to address the problem that we assume a *known* source distribution. Both issues can be resolved by so-called *universal coding schemes*.

7.2 Elias–Willems Universal Block–To–Variable-Length Coding

We have already mentioned that adaptive Huffman coding is very expensive and often even not feasible. However, there is one more very fundamental problem with Huffman codes: We need to know the exact probability distribution of the DSS. In practice we usually have no clue of the statistical distribution of the source we would like to compress.

We would like to find a compression scheme that does compress as efficiently as adaptive Huffman coding, but without the need of knowing the distribution of the source (and, if possible, in a more feasible and practical way).

The clue idea here is to try to find the distribution "on the fly".

Example 7.4. Consider a binary DSS $U_k \in \{a, b\}$ and assume that it produces the following sequence:

$$aaabaabaaaabababbaaaaba \tag{7.22}$$

We note that a seems to have a higher probability than b! So how can we make use of such observations? ◊

7.2.1 Recency Rank Calculator

The basic idea is shown in Figure 7.2. We use a new device called *recency rank calculator*. To understand how it works, firstly note that there are r^{M} different possible messages that can be at the input of the recency rank calculator. We assume that the system has been running for a long time, so that all possible messages have occurred at least once. Then at time k, the message that has been seen *most recently* has *recency rank 1*, the next *different* message before the most recent one has *recency rank 2*, etc.

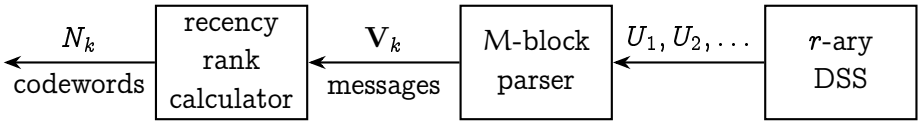


Figure 7.2: A first idea for a universal compression system that relies on a new device called *recency rank calculator*.

Example 7.5. Assume a binary source ($r = 2$) and a parser producing block messages of length $\mathrm{M} = 2$. Then we have $2^2 = 4$ possible messages $\{00, 01, 10, 11\}$ at the input of the recency rank calculator. Suppose that the sequence of past messages before time k is

$$\ldots \big|\mathbf{V}_{k-4}\big|\mathbf{V}_{k-3}\big|\mathbf{V}_{k-2}\big|\mathbf{V}_{k-1}\big|^{\overset{\text{now}}{\downarrow}} = \ldots \big|11\big|00\big|11\big|10\big|01\big|11\big|01\big|01\big|^{\overset{\text{now}}{\downarrow}} \tag{7.23}$$

Then we have the recency rank list shown in Table 7.3. We note that the recency rank calculator simply orders all possible messages in the order of their last occurrence. ◊

Table 7.3: Example of a recency rank list according to the situation shown in (7.23).

Message	Recency Rank at Time k
00	4
01	1
10	3
11	2

Let us make this definition formal.

Definition 7.6. A *recency rank calculator* is a device that assigns the actual recency rank N_k to the message $\mathbf{V}_k$, and **afterwards** updates the recency rank list.

Example 7.7. To continue with our example, assume that $\mathbf{V}_k = 10$. Hence, from the current recency rank list in Table 7.3 we see that $N_k = 3$. So, the recency rank calculator puts out $N_k = 3$ and then updates the recency rank list as shown in Table 7.4. ◊

Table 7.4: Updated recency rank list.

Message	Recency Rank at Time $k+1$
00	4
01	2
10	1
11	3

According to Figure 7.2, the codeword for $\mathbf{V}_k$ is now simply the number N_k. Since the decoder has decoded $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_{k-1}$ before it needs to decode N_k, it can easily keep track of the recency rank list in the same way as the encoder and therefore can correctly decode $\mathbf{V}_k$.

Note that at the start of the system, we do not yet have a recency rank list. But this is not really a problem as we simply will define a default starting value of the recency rank list. An example is shown in Table 7.5.

What is the intuition behind this system? Note that messages that occur often have small recency rank. Hence we use a shorter codeword (small number) for them. Messages that show up only rarely will have large N_k, i.e., we assign them a long codeword (large number). So it looks like that we are successfully compressing the source without knowing its distribution! Moreover,

Table 7.5: Example of a default starting recency rank list.

Message	Recency Rank at Time 1 (by definition)
00	1
01	2
10	3
11	4

if the source actually were not stationary, but kept changing its distribution over time, the system would also be able to adapt to these changes because the recency rank list will reflect the change as well.[1]

However, we do have one *serious problem*: The numbers N_k are not prefix-free! As a matter of fact, they are not even uniquely decodable!

Example 7.8. In our example we have four recency ranks: $1, 2, 3, 4$. Their binary representation is then $1, 10, 11, 100$, respectively. So how shall we decode a received sequence 111? It could be $1, 11$ or $11, 1$ or $1, 1, 1$! ◊

We realize:

> We need a prefix-free code for the positive integers!

7.2.2 Codes for Positive Integers

Standard Code

We start with the usual binary representation of the positive integers and call this code the *standard code* $\mathrm{B}_0(n)$. It is shown in Table 7.6.

Table 7.6: *Standard code*: usual binary representation of positive integers.

n	1	2	3	4	5	6	7	8	9	10	$\cdots$
$\mathrm{B}_0(n)$	1	10	11	100	101	110	111	1000	1001	1010	$\cdots$

As mentioned above, this code is not prefix-free (not even uniquely decodable).

[1] Here we need to assume that the change will not be too fast, as the recency rank list needs some time to adapt to a new distribution.

We observe that every codeword has a 1 as its first digit. Moreover, we note that the length $\mathrm{L}_{\mathrm{B}_0}(n)$ of the codeword $\mathrm{B}_0(n)$ is given by

$$\mathrm{L}_{\mathrm{B}_0}(n) = \lfloor \log_2 n \rfloor + 1. \tag{7.24}$$

Be careful: $\lfloor \log_2 n \rfloor + 1$ is not the same as $\lceil \log_2 n \rceil$. To see this, compare, e.g., the values for $n = 4$.

First Elias Code

Elias had the idea [Eli75] to make the standard code prefix-free by simply adding a prefix in front of every codeword consisting of $\mathrm{L}_{\mathrm{B}_0}(n) - 1$ zeros. We call this code the *first Elias code* $\mathrm{B}_1(n)$. The first couple of codewords are listed in Table 7.7.

Table 7.7: *First Elias code*: Add a prefix of $\mathrm{L}_{\mathrm{B}_0}(n) - 1$ zeros to $\mathrm{B}_0(n)$.

n	1	2	3	4	5	6
$\mathrm{B}_0(n)$	1	10	11	100	101	110
$\mathrm{L}_{\mathrm{B}_0}(n) - 1$	0	1	1	2	2	2
$\mathrm{B}_1(n)$	1	010	011	00100	00101	00110
n	7	8	9	10	11	$\cdots$
$\mathrm{B}_0(n)$	111	1000	1001	1010	1011	$\cdots$
$\mathrm{L}_{\mathrm{B}_0}(n) - 1$	2	3	3	3	3	$\cdots$
$\mathrm{B}_1(n)$	00111	0001000	0001001	0001010	0001011	$\cdots$

Obviously, the code $\mathrm{B}_1(n)$ is prefix-free because the number of leading zeros in a codeword tells us exactly how many digits will come after the inevitable leading 1 of the standard code.

Example 7.9. The sequence of codewords

$$0010101100110001110\ldots \tag{7.25}$$

is unambiguously recognizable as

$$00101|011|00110|00111|0\ldots = 5|3|6|7|\ldots \tag{7.26}$$

◊

Unfortunately, the codewords $\mathrm{B}_1(n)$ are almost twice as long as the codewords $\mathrm{B}_0(n)$:

$$\mathrm{L}_{\mathrm{B}_1}(n) = \mathrm{L}_{\mathrm{B}_0}(n) - 1 + \mathrm{L}_{\mathrm{B}_0}(n) = 2\mathrm{L}_{\mathrm{B}_0}(n) - 1 = 2\lfloor \log_2 n \rfloor + 1. \tag{7.27}$$

Second Elias Code

To solve the problem of the inefficient length, Elias came up with another idea [Eli75]: Instead of using zeros as prefix to tell the length of the codeword, we use $\mathrm{B}_1(\mathrm{L}_{\mathrm{B}_0}(n))$ to tell how many digits are coming. Moreover, since $\mathrm{B}_1(\cdot)$ is prefix-free, we do not need to use the leading 1 of $\mathrm{B}_0(n)$. This code is called the *second Elias code* and is given in Table 7.8.

Table 7.8: *Second Elias code*: The length is encoded by the first Elias code, the number by the standard code:

$$\mathrm{B}_2(n) = \mathrm{B}_1(\mathrm{L}_{\mathrm{B}_0}(n)) \cup [\mathrm{B}_0(n) \text{ without leading } 1].$$

n	1	2	3	4	5	6
$\mathrm{B}_0(n)$	1	10	11	100	101	110
$\mathrm{L}_{\mathrm{B}_0}(n)$	1	2	2	3	3	3
$\mathrm{B}_1(\mathrm{L}_{\mathrm{B}_0}(n))$	1	010	010	011	011	011
$\mathrm{B}_2(n)$	1	0100	0101	01100	01101	01110
n	7	8	9	10	11	$\cdots$
$\mathrm{B}_0(n)$	111	1000	1001	1010	1011	$\cdots$
$\mathrm{L}_{\mathrm{B}_0}(n)$	3	4	4	4	4	$\cdots$
$\mathrm{B}_1(\mathrm{L}_{\mathrm{B}_0}(n))$	011	00100	00100	00100	00100	$\cdots$
$\mathrm{B}_2(n)$	01111	00100000	00100001	00100010	00100011	$\cdots$

Example 7.10. The codeword for $n = 7$ in the standard representation is $\mathrm{B}_0(7) = 111$ with codeword length $\mathrm{L}_{\mathrm{B}_0}(7) = 3$. Now we use the first Elias code to write 3 as $\mathrm{B}_1(3) = 011$, remove the leading 1 from 111, and get as a new codeword for $n = 7$:

$$\mathrm{B}_2(7) = 011\,11. \tag{7.28}$$

$\Diamond$

Note that because the code $\mathrm{B}_1(n)$ is prefix-free, we can recognize the first part $\mathrm{B}_1(\mathrm{L}_{\mathrm{B}_0}(n))$ of the codeword $\mathrm{B}_2(n)$ as soon as we see the last digit of $\mathrm{B}_1(\mathrm{L}_{\mathrm{B}_0}(n))$. We decode this part and then immediately know the number $\mathrm{L}_{\mathrm{B}_0}(n)$ of digits in the remainder of the codeword. Thus $\mathrm{B}_2(n)$ is indeed prefix-free.

How efficient is $\mathrm{B}_2(n)$? We get

$$\begin{aligned}
\mathrm{L}_{\mathrm{B}_2}(n) &= \mathrm{L}_{\mathrm{B}_1}(\mathrm{L}_{\mathrm{B}_0}(n)) + \mathrm{L}_{\mathrm{B}_0}(n) - 1 && (7.29)\\
&= \mathrm{L}_{\mathrm{B}_1}(\lfloor \log_2 n \rfloor + 1) + \lfloor \log_2 n \rfloor + 1 - 1 && (7.30)\\
&= 2\lfloor \log_2(\lfloor \log_2 n \rfloor + 1) \rfloor + 1 + \lfloor \log_2 n \rfloor && (7.31)\\
&\approx \log_2 n, \quad \text{for large } n. && (7.32)
\end{aligned}$$

So, because $\log_2 \log_2 n$ is extremely slowly growing, the second Elias code is, at least for large n, far better than the first. As a matter of fact we have

$$\mathrm{L}_{\mathrm{B}_2}(n) > \mathrm{L}_{\mathrm{B}_1}(n), \qquad \text{only for } n = 2, 3, \ldots, 15; \tag{7.33}$$

$$\mathrm{L}_{\mathrm{B}_2}(n) = \mathrm{L}_{\mathrm{B}_1}(n), \qquad \text{for } n = 16, 17, \ldots, 31; \tag{7.34}$$

$$\mathrm{L}_{\mathrm{B}_2}(n) < \mathrm{L}_{\mathrm{B}_1}(n), \qquad \text{for } n \geq 32. \tag{7.35}$$

Asymptotically, the second Elias code has the same efficiency as the standard code in spite of the overhead that we added to make it prefix-free.

We will now use the second Elias code to build a block–to–variable-length coding scheme.

7.2.3 Elias–Willems Block–to–Variable-Length Coding for a DSS

It is now straightforward to combine the idea of a recency rank calculator with the prefix-free code of Elias. See Figure 7.9 for the corresponding system.

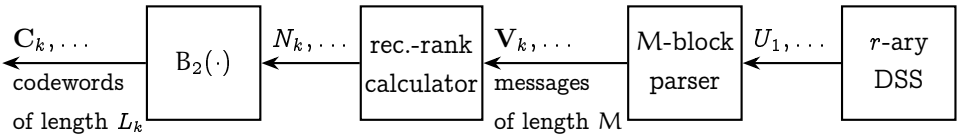


Figure 7.9: An Elias–Willems compression scheme.

Let us compute the efficiency of this system. We suppose that encoding has been in progress for a time sufficiently long for all message values to have appeared at least once in the past so that all recency ranks are genuine.

Define Δ_k to be the time to the most recent occurrence of $\mathbf{V}_k$: If $\mathbf{V}_k = \mathbf{v}$, $\mathbf{V}_{k-\delta} = \mathbf{v}$, $\mathbf{V}_j \neq \mathbf{v}$ for all $j = k-1, \ldots, k-\delta+1$, then $\Delta_k = \delta$.

Example 7.11. If

$$\ldots |\mathbf{V}_{k-4}|\mathbf{V}_{k-3}|\mathbf{V}_{k-2}|\mathbf{V}_{k-1}|\mathbf{V}_k = \ldots |11|00|11|\underbrace{10|01|11|01|01|10}_{\Delta_k} \tag{7.36}$$

then $\Delta_k = 5$. ◊

Note that

$$N_k \leq \Delta_k, \tag{7.37}$$

because only the values of $\mathbf{V}_{k-1}, \ldots, \mathbf{V}_{k-\Delta_k+1}$ (which might not all be different!) could have smaller recency ranks than $\mathbf{V}_k$.

We assume that the DSS is *ergodic*, i.e., time statistics are equal to the probabilistic statistics. Then we know that for a very long sequence, the number of occurrences of $\mathbf{v}$ divided by the whole length of the sequence is

approximately $P_{\mathbf{V}}(\mathbf{v})$. Hence the average spacing between these occurrences is $\frac{1}{P_{\mathbf{V}}(\mathbf{v})}$:

$$\mathsf{E}[\Delta_k \,|\, \mathbf{V}_k = \mathbf{v}] = \frac{1}{P_{\mathbf{V}}(\mathbf{v})}. \tag{7.38}$$

The binary codeword $\mathbf{C}_k$ for the message $\mathbf{V}_k$ is $\mathrm{B}_2(N_k)$. Therefore its length L_k satisfies

$$\begin{aligned} L_k &= \mathrm{L}_{\mathrm{B}_2}(N_k) && (7.39)\\ &\le \mathrm{L}_{\mathrm{B}_2}(\Delta_k) && (7.40)\\ &= \lfloor \log_2 \Delta_k \rfloor + 2\lfloor \log_2(\lfloor \log_2 \Delta_k \rfloor + 1)\rfloor + 1 && (7.41)\\ &\le \log_2 \Delta_k + 2\log_2(\log_2 \Delta_k + 1) + 1, && (7.42) \end{aligned}$$

where the first inequality follows from (7.37) and the second from dropping the flooring-operations. Taking the conditional expectation, given that $\mathbf{V}_k = \mathbf{v}$, results in

$$\begin{aligned} &\mathsf{E}[L_k \,|\, \mathbf{V}_k = \mathbf{v}]\\ &\quad\le \mathsf{E}[\log_2 \Delta_k \,|\, \mathbf{V}_k = \mathbf{v}] + 2\,\mathsf{E}\big[\log_2(\log_2 \Delta_k + 1)\,\big|\,\mathbf{V}_k = \mathbf{v}\big] + 1 && (7.43)\\ &\quad\le \log_2(\mathsf{E}[\Delta_k \,|\, \mathbf{V}_k = \mathbf{v}]) + 2\log_2\big(\log_2(\mathsf{E}[\Delta_k \,|\, \mathbf{V}_k = \mathbf{v}]) + 1\big) + 1 && (7.44)\\ &\quad= -\log_2 P_{\mathbf{V}}(\mathbf{v}) + 2\log_2(-\log_2 P_{\mathbf{V}}(\mathbf{v}) + 1) + 1. && (7.45) \end{aligned}$$

Here (7.44) follows from the Jensen Inequality (Theorem 2.1) and the concavity of the logarithm; and (7.45) follows from (7.38).

Taking the expectation over $\mathbf{V}_k$ and invoking the Total Expectation Theorem finally yields:

$$\begin{aligned} \mathsf{E}[L_k] &= \mathsf{E}_{\mathbf{V}}[\mathsf{E}[L_k \,|\, \mathbf{V}_k = \mathbf{v}]] && (7.46)\\ &\le \mathsf{E}[-\log_2 P_{\mathbf{V}}(\mathbf{V})] + 2\,\mathsf{E}\big[\log_2(-\log_2 P_{\mathbf{V}}(\mathbf{V}) + 1)\big] + 1 && (7.47)\\ &\le \underbrace{\mathsf{E}[-\log_2 P_{\mathbf{V}}(\mathbf{V})]}_{=\mathrm{H}(\mathbf{V})} + 2\log_2\big(\underbrace{\mathsf{E}[-\log_2 P_{\mathbf{V}}(\mathbf{V})]}_{=\mathrm{H}(\mathbf{V})} + 1\big) + 1 && (7.48)\\ &= \mathrm{H}(\mathbf{V}) + 2\log_2(\mathrm{H}(\mathbf{V}) + 1) + 1 && (7.49)\\ &= \mathsf{M}\cdot\frac{1}{\mathsf{M}}\,\mathrm{H}(U_1,\ldots,U_{\mathsf{M}}) + 2\log_2\Big(\mathsf{M}\cdot\frac{1}{\mathsf{M}}\,\mathrm{H}(U_1,\ldots,U_{\mathsf{M}}) + 1\Big) + 1 && (7.50)\\ &\le \mathsf{M}\cdot\frac{\mathrm{H}(U_1,\ldots,U_{\mathsf{M}})}{\mathsf{M}} + 2\log_2(\mathsf{M}\cdot\mathrm{H}(U_1) + 1) + 1, && (7.51) \end{aligned}$$

where (7.47) follows from (7.45), where in (7.48) we have used the Jensen Inequality once more, and where the last inequality (7.51) follows from Corollary 6.26 (because the source is stationary). Hence we have

$$\frac{\mathsf{E}[L_k]}{\mathsf{M}} \le \frac{\mathrm{H}(U_1,\ldots,U_{\mathsf{M}})}{\mathsf{M}} + \frac{2}{\mathsf{M}}\log_2(\mathsf{M}\cdot\mathrm{H}(U_1) + 1) + \frac{1}{\mathsf{M}}. \tag{7.52}$$

Recall that the entropy rate of a stationary source always exists and is given by

$$\mathrm{H}(\{U_k\}) = \lim_{\mathrm{M}\to\infty} \frac{\mathrm{H}(U_1,\ldots,U_\mathrm{M})}{\mathrm{M}}. \tag{7.53}$$

Thus for $\mathrm{M}\to\infty$ we have

$$\lim_{\mathrm{M}\to\infty} \frac{\mathsf{E}[L_k]}{\mathrm{M}} \le \lim_{\mathrm{M}\to\infty} \Bigg\{ \underbrace{\frac{\mathrm{H}(U_1,\ldots,U_\mathrm{M})}{\mathrm{M}}}_{\to \mathrm{H}(\{U_k\})} + \underbrace{\frac{2}{\mathrm{M}} \overbrace{\log_2\bigl(\mathrm{M}\cdot \mathrm{H}(U_1)+1\bigr)}^{\text{grows more slowly than } \mathrm{M}} + \frac{1}{\mathrm{M}}}_{\to 0} \Bigg\} \tag{7.54}$$

$$= \mathrm{H}(\{U_k\}). \tag{7.55}$$

When we compare this with the converse of the coding theorem for block–to–variable-length coding for a DSS (Theorem 7.3), we see that this system is asymptotically (i.e., for large M) optimal.

Theorem 7.12 (Elias–Willems Block–to–Variable-Length Coding Theorem for a DSS).
For the Elias–Willems **binary** coding scheme of M-block messages from an r-ary ergodic DSS, the codeword length L_k (for all k sufficiently large) satisfies:

$$\mathrm{H}(\{U_k\}) \le \frac{\mathsf{E}[L_k]}{\mathrm{M}} \le \frac{\mathrm{H}(U_1,\ldots,U_\mathrm{M})}{\mathrm{M}} + \frac{2}{\mathrm{M}}\log_2\biggl(\mathrm{M}\cdot\frac{\mathrm{H}(U_1,\ldots,U_\mathrm{M})}{\mathrm{M}}+1\biggr)+\frac{1}{\mathrm{M}} \tag{7.56}$$

where all entropies are given in bits. By choosing M sufficiently large, $\frac{\mathsf{E}[L_k]}{\mathrm{M}}$ can be made arbitrarily close to $\mathrm{H}(\{U_k\})$.

We would like to add that this theorem also holds if the DSS is *nonergodic*, but the proof is more complicated. Moreover, as already mentioned, it usually will even work if the source is not stationary. However, in such a case it will be very hard to prove anything.

Since this coding scheme works without knowing the source statistics, it is called a *universal coding scheme*. The Elias–Willems coding scheme is an example of a universal block–to–variable-length coding scheme. It is very elegant in description and also analysis. However, in practice it is not widely used. Some of the most commonly used schemes are universal variable-length–to–block coding schemes called *Lempel–Ziv coding schemes*. They are also asymptotically optimal. We will describe them next.

7.3 Sliding Window Lempel–Ziv Universal Coding Scheme

In real life, the most often used compression algorithms are based on the Lempel–Ziv universal coding schemes. There are many variations, but in general one can distinguish two main types: the *sliding window Lempel–Ziv coding scheme* (published it in 1977) and the *tree-structured Lempel–Ziv coding scheme* (published in 1978).

The latter coding scheme will be discussed and analyzed in Chapter 8. In this section we will briefly describe the former, but will refrain from making a detailed performance analysis because it is inherently related to the Elias-Willems coding scheme and therefore has a similar performance. We will explain the main idea using the example of a binary source $U_k \in \{A, B\}$.

Description

- Fix a window size w (in the following we will use the example of $w = 5$) and a maximum match length $l_{\max}$, and start parsing the source sequence.
- Consider the length-w window before the current position

$$\underset{\text{window}}{\uparrow} \qquad \overset{\text{now}}{\downarrow}$$
$$\ldots A\,A\,\underline{B\,A\,A\,B\,B}\,A\,A\,B\,A\,B\,A\,B\,A\,A\ldots \tag{7.57}$$

  and try to find the largest match of any sequence starting in the window that is identical to the sequence starting on the current position:

$$\text{current position:} \qquad \overbrace{A\,A\,B}^{\text{match}}\,A\,B\,A\,B\ldots \tag{7.58}$$
$$\text{4 positions before:} \qquad \underbrace{A\,A\,B}_{\text{match}}\,B\,A\,A\,B\,A\,B\,A\,B\ldots \tag{7.59}$$

  Note that a match may overlap with the current position! For example,

$$\underset{\text{window}}{\uparrow} \qquad \overset{\text{now}}{\downarrow}$$
$$\ldots B\,\underline{B\,A\,A\,B\,B}\,A\,B\,B\,A\,B\,A\,A\ldots \tag{7.60}$$

  has a match of length 5 at position 3:

$$\text{current position:} \qquad \overbrace{A\,B\,B\,A\,B}^{\text{match}}\,A\,A\ldots \tag{7.61}$$
$$\text{3 positions before:} \qquad \underbrace{A\,B\,B\,A\,B}_{\text{match}}\,B\,A\,B\,A\,A\ldots \tag{7.62}$$

- If there exists such a match, the encoder gives out the starting position of this match in the window (in the example of above: 3 before the current position) and the length of the match (in the example from above: 5), and then moves the current position the number of the match length positions ahead. Note that after $l_{\max}$ matching source letters the match is stopped even if there were further matching letters available.
- If there is no such match, the encoder puts out the uncompressed source letter at the current position, and then moves the current position one letter ahead.
- To distinguish between these two cases (i.e., either putting out a match position and a match length or putting out an uncompressed source letter), we need a flag bit: A flag 0 tells that the codeword describes an uncompressed source letter and a flag 1 tells that the codeword describes a match within the window.
- Note that in the case of a match, the match position and the match length will be encoded into two D-ary phrases of length $\lceil \log_{\mathrm{D}}(w) \rceil$ and $\lceil \log_{\mathrm{D}}(l_{\max}) \rceil$, respectively, where a 1 is mapped to $00\cdots0$ and w (or $l_{\max}$, respectively) is mapped to $\mathrm{DD}\cdots\mathrm{D}$.
- Also note that in the case of an uncompressed source letter, the source letter needs to be mapped into a D-ary phrase of length $\lceil \log_{\mathrm{D}}(|\mathcal{U}|) \rceil$, where $|\mathcal{U}|$ denotes the alphabet size of the source. This mapping has to be specified in advance.

Example 7.13. We still keep the example of a binary source $U_k \in \{A, B\}$ and again choose the window size $w = 4$. Moreover, we fix $l_{\max} = 8$ and choose binary codewords $\mathrm{D} = 2$. Assume that the source produces the following sequence:

$$ABBABBABBBAABABA. \tag{7.63}$$

The window of the Lempel–Ziv algorithm now slides through the given sequence as follows:

$$\begin{aligned}
&\underline{\quad}|A\ B\ B\ A\ B\ B\ A\ B\ B\ B\ A\ A\ B\ A\ B\ A\\
&\underline{\quad\ }|A|\\
&\underline{\quad\ \ }|A|B|\\
&\underline{\quad\ \ \ }|A|B|B|\\
&\quad\ |A|B|B|A\ B\ \underline{B\ A\ B\ B}|\\
&\quad\ |A|B|B|A\ B\ B\ A\ \underline{B\ B}|B\ A|\\
&\quad\ |A|B|B|A\ B\ B\ A\ B\ \underline{B}|B\ A|A|\\
&\quad\ |A|B|B|A\ B\ B\ A\ B\ B|B\ \underline{A}|A|B\ A|\\
&\quad\ |A|B|B|A\ B\ B\ A\ B\ B|B\ A|A|\underline{B\ A}|B\ A|
\end{aligned} \tag{7.64}$$

This leads to the following description:

$$\begin{array}{llll}(0,A) & (0,B) & (1,1,1) & (1,3,6)\\ (1,4,2) & (1,1,1) & (1,3,2) & (1,2,2).\end{array} \tag{7.65}$$

We map A to 0, and B to 1. Moreover, we need $\lceil \log_2(4) \rceil = 2$ digits for the match position (where $1 \mapsto 00$, $2 \mapsto 01$, $3 \mapsto 10$, and $4 \mapsto 11$) and $\lceil \log_2(8) \rceil = 3$ digits for the match length (where $1 \mapsto 000, \ldots, 8 \mapsto 111$). Hence, the description (7.65) is translated into

$$\begin{array}{llll}(0,0) & (0,1) & (1,00,000) & (1,10,101)\\ (1,11,001) & (1,00,000) & (1,10,001) & (1,01,001),\end{array} \tag{7.66}$$

i.e., the codeword put out by the encoder is

$$000110000011010111100110000011000110100 1. \tag{7.67}$$

Note that in this small toy example, the window size, the maximum match length, and the length of the encoded sequence are too short to be able to provide a significant compression. Indeed, the encoded sequence is longer than the original sequence.

Also note that it may be better to send short matches uncompressed and only compress longer matches (i.e., we use flag 0 more often). For example, in (7.66) the third part $(1,00,000)$ would actually be more efficiently encoded as $(0,1)$. ◊

Similarly to the Elias–Willems coding scheme, the sliding window Lempel–Ziv algorithm relies on the repeated occurrence of strings. So its analysis is related to the analysis shown in Section 7.2.3 and we therefore omit it. Practical implementations of it can be found, e.g., in `gzip` and `pkzip`.

Chapter 8

Data Compression: Efficient Coding of an Infinitely Long Fixed Sequence

So far (i.e., in Chapters 4–7) we have assumed that we are given a certain source with a known statistical behavior. We then tried to design a good compression scheme using the statistical description of the source. Only in Section 7.2 we have seen a first design of a compression scheme that did not rely on the specific distribution of the source (although we still required the source to be stationary), thereby finding a *universal* coding scheme.

In this chapter, we will completely depart from this approach and instead try to design a system which represents some sequence without considering how the sequence was produced. In this context, it thus makes more sense to completely abandon the statistical model and instead to assume that there is only one, infinitely long, fixed r-ary sequence $\mathbf{u}$ that shall be represented efficiently.

Of course, without a statistical model, we also seem to lose the ability to evaluate the *efficiency* of our system (there is no entropy rate to compare with and no probability measure allowing us to compute an expected performance, etc.). To compensate for that we introduce a trick: we start by analyzing the *best* possible system among all systems based on *finite state machines* with a given number S of states. The motivation behind the restriction on the number of states is quite a practical one: no real computing device can have infinite memory and precision. (On the other hand, we neither will be able to compress an *infinite* sequence, but we will always only compress a finite part of it.) The restriction on the memory is not very stringent though, because while we ask S to be finite, we do allow it to be exceedingly large. Moreover, we even allow this *best* system to be designed specifically for the given infinite sequence $\mathbf{u}$, i.e., we are allowed to custom-design it for the specific given sequence!

Then, in a second step, we propose a concrete algorithm — the tree-structured Lempel–Ziv algorithm — that is designed *without* knowledge of

$\mathbf{u}$, but that is not a finite state machine, i.e., for an infinite input sequence it might use infinite memory.[1] We will then analyze this algorithm and show that it performs as well as the best encoder based on finite state machines.

8.1 Information-Lossless Finite State Encoders

Finite state machines are widely used in many different situations. We will consider them only in our case of a device that compresses a given input string $\mathbf{u}$. We therefore use the slightly less common notation $\mathbf{c}$ for its output sequence ('c' stands for 'compressed output sequence' and is in agreement with our notation from earlier chapters where $\mathbf{c}$ stood for 'codeword').

Definition 8.1. A *finite state encoder (FSE)* with S states is defined as a device with an r-ary input u_k, a D-ary output string $\mathbf{c}_k$, and an S-ary internal state s_k such that given an input u_k and a current state s_k, the output is computed as

$$\mathbf{c}_k = f_{\text{FSE,out}}(s_k, u_k), \quad k \in \mathbb{N}, \tag{8.1}$$

and the next state is computed as

$$s_{k+1} = f_{\text{FSE,state}}(s_k, u_k), \quad k \in \mathbb{N}, \tag{8.2}$$

for some given functions $f_{\text{FSE,out}}$ and $f_{\text{FSE,state}}$. The output $\mathbf{c}_k$ is a finite string that potentially can also be the null string.

We will be sloppy and denote by $f_{\text{FSE,out}}(s_1, u_1^n)$ the output sequence $\mathbf{c}_1^n$ generated by the finite state machine for a starting state s_1 and for an input sequence $u_1^n = (u_1, \ldots, u_n)$; similarly, $f_{\text{FSE,out}}(s_1, u_1^n)$ denotes the state s_{n+1}. Note that the length of $\mathbf{c}_1^n$ can be larger or smaller than n because for each k, $\mathbf{c}_k$ is a finite string of length larger or equal to zero.

In order to be able to undo the compression, we need a *unique decodability condition* on the FSE: Basically, we want to be able to reconstruct $\mathbf{u}$ from the output sequence $\mathbf{c}$ (and the knowledge of the starting state s_1). In the following we will even allow some FSE for which this is not necessarily true.

Definition 8.2. A FSE is called *information lossless* if for any two distinct input sequences u_1^n and $\tilde{u}_1^{\tilde{n}}$ and any starting state s_1 either

$$f_{\text{FSE,out}}(s_1, u_1^n) \neq f_{\text{FSE,out}}(s_1, \tilde{u}_1^{\tilde{n}}) \tag{8.3}$$

or

$$f_{\text{FSE,state}}(s_1, u_1^n) \neq f_{\text{FSE,state}}(s_1, \tilde{u}_1^{\tilde{n}}). \tag{8.4}$$

[1] As in reality we will never encode an infinite sequence anyway, we do not need to worry about that.

Quite obviously, if a FSE is not information lossless, then we have no hope to recover the input from the output. However, even if a FSE is information lossless, it might happen that two different input sequences (and the same starting state) result in the same output sequence, but only a different final state. Since we do not observe the states, we will in this case not be able to recover the input. We nevertheless allow such encoders and thereby set the performance goal for the tree-structured Lempel–Ziv algorithm even higher.

We next define the *compressibility* of an infinite string $\mathbf{u}$ with respect to information-lossless FSEs.

Definition 8.3. The *minimum compression ratio* of a sequence u_1^n with respect to all information-lossless FSEs with S or less states is defined as

$$\rho_S(u_1^n) \triangleq \min_{f_{\text{FSE,out}},\, f_{\text{FSE,state}},\, s_1} \frac{1}{n}\, l_{\text{FSE}}(u_1^n), \tag{8.5}$$

where $l_{\text{FSE}}(u_1^n)$ denotes the length of the D-ary output sequence $\mathbf{c}_1^n$ that is generated by the FSE with input u_1^n, and where the minimum is over the set of all information-lossless FSEs with S or less states and all starting states.

Note that since it is always possible to design an FSE such that each r-ary input symbol is simply represented as a D-ary sequence, it follows that

$$\rho_S(u_1^n) \le \frac{1}{n} \cdot n\lceil \log_D(r) \rceil = \lceil \log_D(r) \rceil. \tag{8.6}$$

Definition 8.4. The *compressibility* of $\mathbf{u}$ with respect to information-lossless FSEs with S or less states is then defined as

$$\rho_S(\mathbf{u}) \triangleq \overline{\lim_{n\to\infty}}\, \rho_S(u_1^n) \tag{8.7}$$

and the *compressibility* of $\mathbf{u}$ with respect to all information-lossless encoders is defined as[2]

$$\rho(\mathbf{u}) \triangleq \lim_{S\to\infty} \rho_S(\mathbf{u}). \tag{8.8}$$

We see that in the definition of the compressibility of $\mathbf{u}$ we basically drop all constraints on the system apart from that it must be (almost) uniquely decodable and that its memory is arbitrary but finite.

8.2 Distinct Parsing

Definition 8.5. A *parsing* of a sequence is a division of the sequence into strings separated by commas. A *distinct parsing* is a parsing such that no two strings are the same.

[2] Note that since $\rho_S(\mathbf{u})$ is nonincreasing in S and bounded from below by zero, the limit exists.

Example 8.6. $0, 11, 11$ is a parsing and $0, 1, 111$ is a distinct parsing of 01111. $\Diamond$

Definition 8.7. Let $b(u_1^n)$ denote the *maximum number of distinct strings* that the sequence u_1^n can be parsed into, including the null string.

We will see that this number is fundamental to the compressibility of $\mathbf{u}$.

Lemma 8.8 (Maximum Number of Distinct Strings vs. Sequence Length). The maximum number of distinct strings $b(u_1^n)$ of an r-ary sequence u_1^n of length n satisfies the following bound:

$$n > b(u_1^n) \log_r \left(\frac{b(u_1^n)}{r^3} \right). \tag{8.9}$$

Proof: We start by quickly listing all possible r-ary strings of length *less* than some integer m:

$$\begin{aligned}
&\text{empty string;} && \text{(8.10a)}\\
&0, 1, \dots, r-1; && \text{(8.10b)}\\
&00, 01, \dots, 0(r-1); \dots; (r-1)0, \dots, (r-1)(r-1); && \text{(8.10c)}\\
&\vdots\\
&\underbrace{00\dots0}_{m-1 \text{ digits}}, \dots, \underbrace{(r-1)(r-1)\dots(r-1)}_{m-1 \text{ digits}}. && \text{(8.10d)}
\end{aligned}$$

The total number of all these strings is

$$\sum_{j=0}^{m-1} r^j = \frac{r^m - 1}{r-1} \tag{8.11}$$

and the total length of all these strings put together is

$$\sum_{j=0}^{m-1} j r^j = m \frac{r^m}{r-1} - \frac{r}{r-1} \frac{r^m - 1}{r-1}. \tag{8.12}$$

Now, suppose u_1^n is parsed into $b \geq 1$ distinct strings. We would like to find an upper bound on this number b. Obviously, b is largest if the strings in the parsing are as short as possible. But since we consider a *distinct* parsing, every string can occur at most once. So the maximum number occurs if *all* strings of length less than some number m show up and the rest of the sequence is parsed into strings of length m. In other words, we write the integer b as

$$b = \sum_{j=0}^{m-1} r^j + \mathrm{R} \tag{8.13}$$

where $m \geq 0$ is chosen as large as possible and where $0 \leq \mathrm{R} < r^m$.

To illustrate this, let's make a brief example: Suppose we have a ternary sequence ($r = 3$) and $b = 31$. From (8.11) we see that the number of ternary strings of length less than 3 is 13 and the number of ternary strings of length less than 4 is 40. Hence, the largest m is $m = 3$ resulting in $\mathrm{R} = 18$, i.e.,

$$31 = 3^0 + 3^1 + 3^2 + 18. \tag{8.14}$$

If we increase b, R would increase until b reaches 40, when m is increased to $m = 4$ and R is set to 0.

So far we have been looking for the largest number b of distinct strings for a given length n of the sequence. But equivalently, we can turn the problem around and look for the smallest length n for a given number b of distinct strings. In that case it is again obvious that n is smallest if it so happens that all distinct strings are the smallest, i.e., we are back to (8.13). Thus, in general,

$$n \geq \sum_{j=0}^{m-1} j r^j + m\mathrm{R}. \tag{8.15}$$

Using (8.11), (8.12), and (8.13) we can further bound this as follows:

$$\begin{aligned}
n &\geq \sum_{j=0}^{m-1} j r^j + m\mathrm{R} && (8.16)\\
&= m\frac{r^m}{r-1} - \frac{r}{r-1}\frac{r^m-1}{r-1} + m\mathrm{R} && (8.17)\\
&> m\frac{r^m-1}{r-1} - \frac{r}{r-1}\frac{r^m-1}{r-1} + m\mathrm{R} && (8.18)\\
&= m\sum_{j=0}^{m-1} r^j - \frac{r}{r-1}\sum_{j=0}^{m-1} r^j + m\mathrm{R} && (8.19)\\
&= m(b-\mathrm{R}) - \frac{r}{r-1}(b-\mathrm{R}) + m\mathrm{R} && (8.20)\\
&= mb - \underbrace{\frac{r}{r-1}}_{\leq 2} b + \underbrace{\frac{r}{r-1}}_{\geq 0}\mathrm{R} && (8.21)\\
&\geq mb - 2b && (8.22)\\
&= (m-2)b. && (8.23)
\end{aligned}$$

On the other hand, from (8.13) and from the fact that $\mathrm{R} < r^m$ it follows that

$$b < \sum_{j=0}^{m} r^j = \frac{r^{m+1}-1}{r-1} \tag{8.24}$$

and thus

$$r^{m+1} > (r-1)b + 1 > b, \tag{8.25}$$

i.e.,

$$m+1 > \log_r(b). \tag{8.26}$$

Plugging this into (8.23) then finally yields

$$n > (\log_r(b)-3)b = b\log_r\left(\frac{b}{r^3}\right). \tag{8.27}$$

Be replacing b by its largest possible value $b(u_1^n)$ we obtain the claimed result. □

Corollary 8.9. For n large enough, it holds that

$$b(u_1^n) < \frac{2n}{\log_r(n)-3}, \quad n \text{ large enough}, \tag{8.28}$$

i.e., $b(u_1^n) \in O(\frac{n}{\log_r(n)})$ (see Appendix 14.A for the definition of the big-O notation).

Proof: We set $n' \triangleq \frac{n}{r^3}$ and $b' \triangleq \frac{b(u_1^n)}{r^3}$. Then (8.9) reads

$$n' > b'\log_r(b'). \tag{8.29}$$

Now let's choose n large enough to guarantee that n' is large enough such that

$$\sqrt{n'} \le \frac{2n'}{\log_r(n')}. \tag{8.30}$$

Then either $b' < \sqrt{n'}$ or $b' \ge \sqrt{n'}$. In the former case,

$$b' < \sqrt{n'} \le \frac{2n'}{\log_r(n')} \tag{8.31}$$

by assumption. In the latter case, it follows from (8.29) and from the said assumption $b' \ge \sqrt{n'}$ that

$$b' < \frac{n'}{\log_r(b')} \le \frac{n'}{\log_r(\sqrt{n'})} = \frac{2n'}{\log_r(n')}. \tag{8.32}$$

Hence, in either case, we have

$$b' < \frac{2n'}{\log_r(n')} \tag{8.33}$$

i.e.,

$$b < \frac{2n}{\log_r\left(\frac{n}{r^3}\right)}. \tag{8.34}$$

□

8.3 Analysis of Information-Lossless Finite State Encoders

We consider now an information-lossless FSE with S states that is fed u_1^n. To analyze its performance, we split u_1^n into its maximum number $b = b(u_1^n)$ of distinct strings $\mathbf{v}_1, \ldots, \mathbf{v}_b$ and consider the output of the FSE for each string separately. Since the output of the FSE depends not only on its input, but also on the starting state, we introduce $b_{s,\tilde{s}}$ to be the number of strings $\mathbf{v}_k$ that find the FSE in state s and leave the FSE in state $\tilde{s}$. Because the FSE is information lossless and because these input strings are distinct, the corresponding output sequences are all distinct. Let $l_{s,\tilde{s}}$ be the total length of these output sequences. Since they are distinct, we may apply Lemma 8.8 (with $l_{s,\tilde{s}}$ substituting n, $b_{s,\tilde{s}}$ substituting $b(u_1^n)$, and because the output is D-ary, D substituting r):

$$l_{s,\tilde{s}} > b_{s,\tilde{s}} \log_{\mathrm{D}}\left(\frac{b_{s,\tilde{s}}}{\mathrm{D}^3}\right). \tag{8.35}$$

The length of the complete output sequence $\mathbf{c}_1^n$ is then the sum of the $l_{s,\tilde{s}}$, i.e.,

$$\begin{aligned}
l_{\mathrm{FSE}}(u_1^n) &= \sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} l_{s,\tilde{s}} && (8.36)\\
&> \sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} b_{s,\tilde{s}} \log_{\mathrm{D}}\left(\frac{b_{s,\tilde{s}}}{\mathrm{D}^3}\right) && (8.37)\\
&= b \sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} \frac{b_{s,\tilde{s}}}{b} \log_{\mathrm{D}}\left(\frac{b_{s,\tilde{s}}}{b}\frac{b}{\mathrm{D}^3}\right) && (8.38)\\
&= b \sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} \frac{b_{s,\tilde{s}}}{b} \log_{\mathrm{D}}\left(\frac{b_{s,\tilde{s}}}{b}\right) + b \sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} \frac{b_{s,\tilde{s}}}{b} \log_{\mathrm{D}}\left(\frac{b}{\mathrm{D}^3}\right) && (8.39)\\
&= b \sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} \frac{b_{s,\tilde{s}}}{b} \log_{\mathrm{D}}\left(\frac{b_{s,\tilde{s}}}{b}\right) + \left(\sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} b_{s,\tilde{s}}\right) \log_{\mathrm{D}}\left(\frac{b}{\mathrm{D}^3}\right) && (8.40)\\
&= -b\,\mathrm{H}_{\mathrm{D}}\left(\left\{\frac{b_{s,\tilde{s}}}{b}\right\}\right) + b \log_{\mathrm{D}}\left(\frac{b}{\mathrm{D}^3}\right) && (8.41)\\
&\geq -b \log_{\mathrm{D}}(\mathrm{S}^2) + b \log_{\mathrm{D}}\left(\frac{b}{\mathrm{D}^3}\right) && (8.42)\\
&= b \log_{\mathrm{D}}\left(\frac{b}{\mathrm{S}^2\mathrm{D}^3}\right) && (8.43)\\
&= b(u_1^n) \log_{\mathrm{D}}\left(\frac{b(u_1^n)}{\mathrm{S}^2\mathrm{D}^3}\right). && (8.44)
\end{aligned}$$

Note that

$$\sum_{s=1}^{\mathrm{S}} \sum_{\tilde{s}=1}^{\mathrm{S}} b_{s,\tilde{s}} = b \tag{8.45}$$

and thus $\frac{b_{s,\tilde{s}}}{b}$ behaves like a probability distribution. The "entropy" introduced in (8.41) has no real meaning here, but is used as a way to find the lower

bound in (8.42): entropy is upper-bounded by the logarithm of its alphabet size (which here is S^2).

Recalling Definition 8.3 and 8.4 we hence obtain

$$\rho_S(\mathbf{u}) = \overline{\lim_{n\to\infty}} \min_{f_{\text{FSE,out}},\, f_{\text{FSE,state}},\, s_1} \frac{1}{n} l_{\text{FSE}}(u_1^n) \tag{8.46}$$

$$\geq \overline{\lim_{n\to\infty}} \frac{1}{n} b(u_1^n) \log_D\left(\frac{b(u_1^n)}{S^2 D^3}\right) \tag{8.47}$$

$$= \overline{\lim_{n\to\infty}} \frac{1}{n} b(u_1^n) \log_D(b(u_1^n)) - \underbrace{\lim_{n\to\infty} \frac{1}{n} b(u_1^n) \log_D(S^2 D^3)}_{=0 \text{ by Corollary 8.9}} \tag{8.48}$$

$$= \overline{\lim_{n\to\infty}} \frac{1}{n} b(u_1^n) \log_D(b(u_1^n)). \tag{8.49}$$

Since the right-hand side does not depend on S, we finally have found the following lower-bound on the compressibility of $\mathbf{u}$.

Theorem 8.10. The compressibility of the infinite r-ary sequence $\mathbf{u}$ with respect to information-lossless FSEs is lower-bounded as follows:

$$\rho(\mathbf{u}) \geq \overline{\lim_{n\to\infty}} \frac{1}{n} b(u_1^n) \log_D(b(u_1^n)). \tag{8.50}$$

8.4 Tree-Structured Lempel–Ziv Universal Coding Scheme

The main idea of the tree-structured Lempel–Ziv coding scheme is to generate a dictionary for the source and constantly updating it. The output sequence then depends on the current version of the stored dictionary. The algorithm starts with a dictionary just consisting of all r-ary words of length 1. To each word in the dictionary it assigns a D-ary codeword of length $\lceil \log_D(r) \rceil$ in lexicographic order. Then it operates as follows: It parses the input. As soon as it recognizes a word in its dictionary, it puts out the corresponding codeword and afterwards updates the dictionary by replacing the just recognized word with all its single-letter extensions. Then it also updates the codeword assignments: it newly assigns D-ary codewords of length $\lceil \log_D(\text{dictionary size}) \rceil$ in lexicographic order. Then it continues scanning the input.

Note that the dictionary can be represented as an r-ary tree, whose leaves are the current dictionary entries. An example of this is shown in Figures 8.1 and 8.2. There the example of a ternary sequence *aaaccacc* is parsed into a, aa, c, ca, cc. Note that in each stage, to each leaf (word in the dictionary) a fixed-length D-ary sequence is assigned in lexicographical order. If we consider binary codewords ($D = 2$), we have

- in stage (a): $\{a \mapsto 00, b \mapsto 01, c \mapsto 10\}$;
- in stage (b): $\{aa \mapsto 000, ab \mapsto 001, ac \mapsto 010, b \mapsto 011, c \mapsto 100\}$;

- in stage (c): $\{aaa \mapsto 000, aab \mapsto 001, aac \mapsto 010, ab \mapsto 011, ac \mapsto 100, b \mapsto 101, c \mapsto 110\}$;
- in stage (d): $\{aaa \mapsto 0000, aab \mapsto 0001, aac \mapsto 0010, ab \mapsto 0011, ac \mapsto 0100, b \mapsto 0101, ca \mapsto 0110, cb \mapsto 0111, cc \mapsto 1000\}$;
- in stage (e): $\{aaa \mapsto 0000, aab \mapsto 0001, aac \mapsto 0010, ab \mapsto 0011, ac \mapsto 0100, b \mapsto 0101, caa \mapsto 0110, cab \mapsto 0111, cac \mapsto 1000, cb \mapsto 1001, cc \mapsto 1010\}$.

Hence, the output of the tree-structured Lempel–Ziv encoder is

$$00\ 000\ 110\ 0110\ 1010 \tag{8.51}$$

where we have added spaces purely for the sake of better readability.

Note that because the dictionary is only updated *after* the recognized word is encoded, the decoder can easily keep track of the encoder's operation. It will simply use the current dictionary to decode the codeword and then use the decoded word to update the dictionary in the same fashion: When decoding the sequence (8.51)

$$0000011001101010 \tag{8.52}$$

it first takes the dictionary (a), prunes 00 from the sequence and decodes it to a. Then it updates the dictionary in the same way as the encoder to (b). Now it can prune 000 from the sequence and decode it to aa. Etc.

The tree-structured Lempel–Ziv coding scheme is used, e.g., in `compress`, in the GIF-format of images, and in certain modems.

8.5 Analysis of Tree-Structured Lempel–Ziv Coding

Suppose the tree-structured Lempel–Ziv coding scheme parses the sequence u_1^n into b_{LZ} words $\mathbf{v}_1, \ldots, \mathbf{v}_{b_{\mathrm{LZ}}}$. Then we can write

$$u_1^n = \lambda \mathbf{v}_1 \mathbf{v}_2 \ldots \mathbf{v}_{b_{\mathrm{LZ}}}, \tag{8.53}$$

where λ denotes the null string. By construction, the first $b_{\mathrm{LZ}} - 1$ strings are distinct, however, the last may not be distinct from the others. If we concatenate the last two strings and instead also count the null string at the beginning, we obtain a distinct parsing of u_1^n into b_{LZ} distinct words. Therefore, by Definition 8.7, it must hold that

$$b_{\mathrm{LZ}} \le b(u_1^n). \tag{8.54}$$

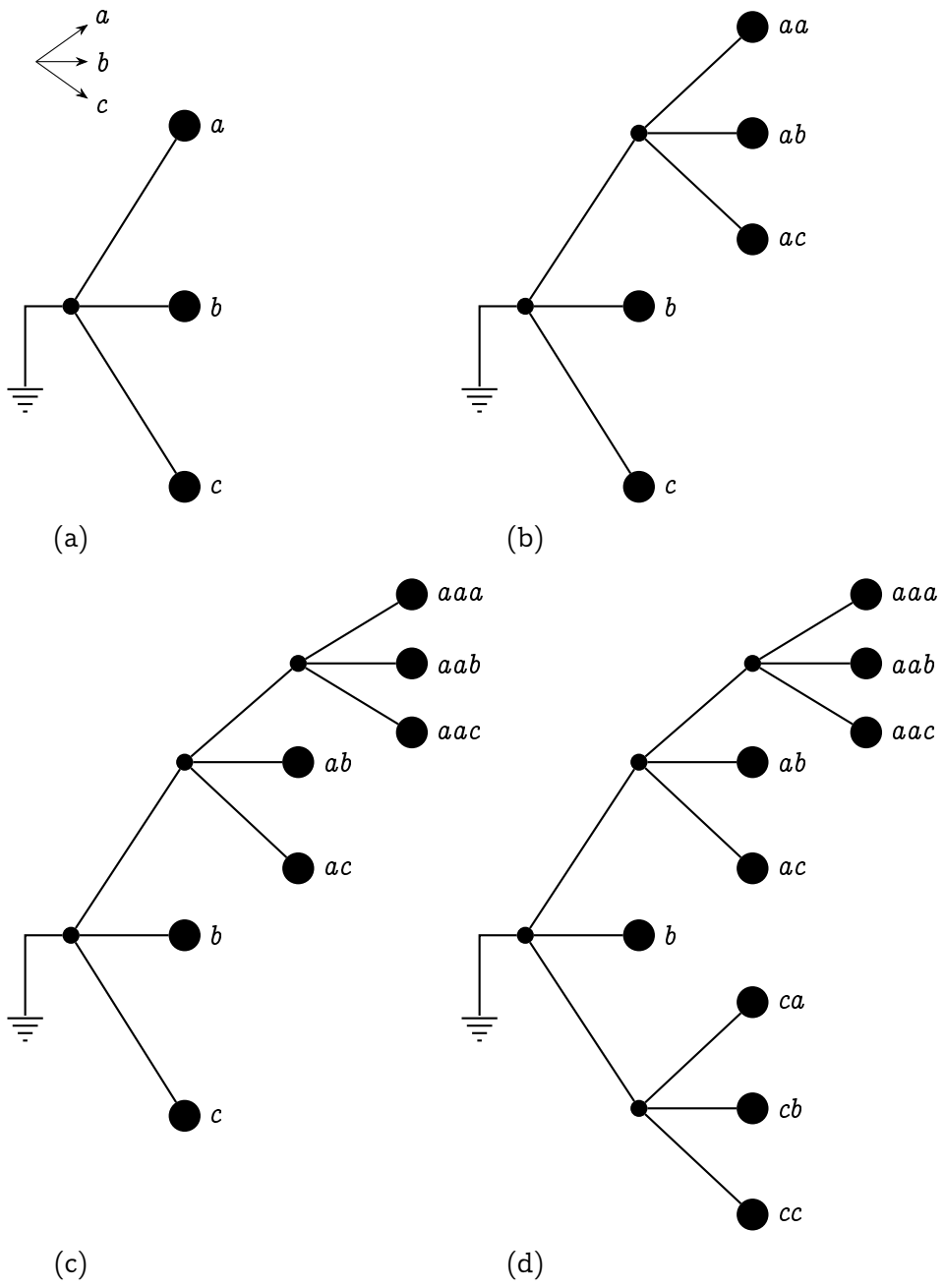


Figure 8.1: Parsing of the ternary sequence $aaaccacc$ with the tree-structured Lempel–Ziv algorithm: the sequence is parsed into the strings a, aa, c, ca, cc. The figure shows how the dictionary evolves. In (a), the initial dictionary is shown. In (b), the dictionary is shown after reading a; (c) shows the dictionary after reading aaa; (d) shows the dictionary after reading $aaac$. In Figure 8.2 (e), the dictionary is shown after reading $aaacca$; and Figure 8.2 (f) shows the dictionary after reading $aaaccacc$.

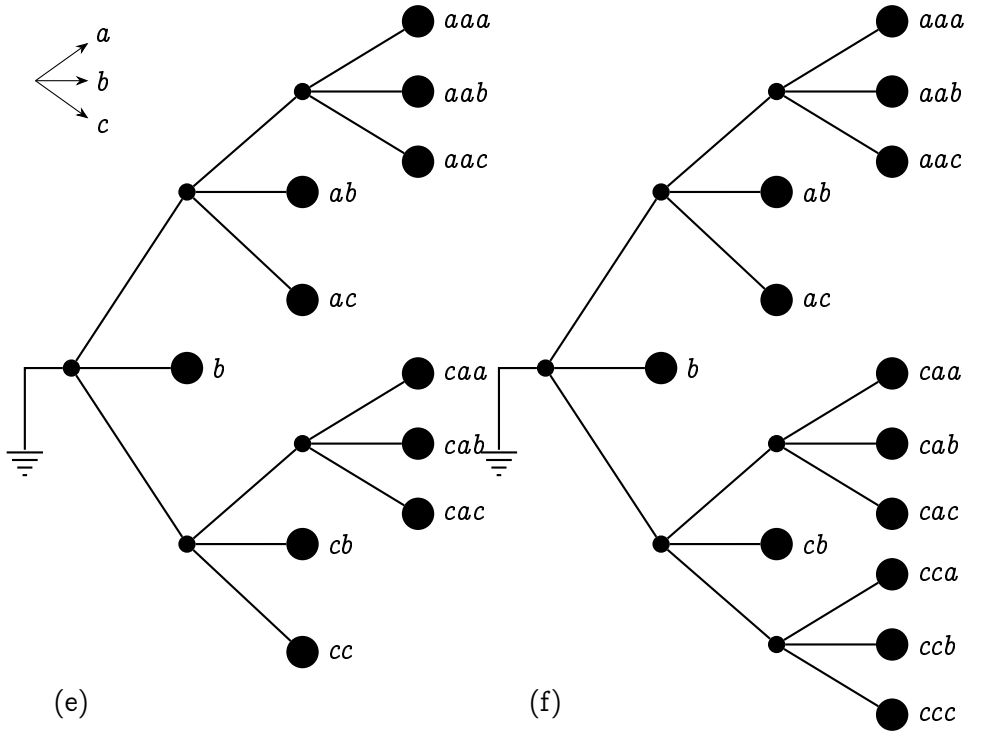


Figure 8.2: Second part of parsing of the ternary sequence $aaaccacc$ with the tree-structured Lempel–Ziv algorithm. See Figure 8.1 for an explanation.

Since each parse extends the dictionary by $r-1$ entries, the size of the dictionary used to encode the last string is

$$\begin{aligned}\text{dictionary end size} &= r+(b_{\mathrm{LZ}}-1)(r-1) && (8.55)\\ &= rb_{\mathrm{LZ}}-b_{\mathrm{LZ}}+1 && (8.56)\\ &\le rb_{\mathrm{LZ}} && (8.57)\\ &\le rb(u_1^n), && (8.58)\end{aligned}$$

where the last inequality follows from (8.54). Let $l_{\mathrm{LZ}}(u_1^n)$ be the length of the output sequence generated by the tree-structured Lempel–Ziv encoding scheme when seeing the input u_1^n. Since the codewords grow in length as the dictionary grows, we can bound $l_{\mathrm{LZ}}(\mathbf{c}_1^n)$ as follows:

$$\begin{aligned}l_{\mathrm{LZ}}(u_1^n) &\le b_{\mathrm{LZ}}\cdot\lceil\log_{\mathrm{D}}(\text{end dictionary size})\rceil && (8.59)\\ &\le b_{\mathrm{LZ}}\cdot\lceil\log_{\mathrm{D}}\big(rb(u_1^n)\big)\rceil && (8.60)\\ &\le b(u_1^n)\cdot\big(\log_{\mathrm{D}}\big(rb(u_1^n)\big)+1\big) && (8.61)\\ &= b(u_1^n)\cdot\big(\log_{\mathrm{D}}\big(rb(u_1^n)\big)+\log_{\mathrm{D}}(\mathrm{D})\big) && (8.62)\\ &= b(u_1^n)\log_{\mathrm{D}}\big(\mathrm{D}rb(u_1^n)\big). && (8.63)\end{aligned}$$

Dividing by n, letting n tend to infinity, and using Corollary 8.9, we obtain

$$\begin{aligned}\overline{\lim_{n\to\infty}}\,\frac{1}{n}\,l_{\mathrm{LZ}}(u_1^n) &\le \overline{\lim_{n\to\infty}}\,\frac{1}{n}\,b(u_1^n)\log_{\mathrm{D}}(\mathrm{D}rb(u_1^n)) && (8.64)\\ &= \overline{\lim_{n\to\infty}}\,\frac{1}{n}\,b(u_1^n)\log_{\mathrm{D}}(b(u_1^n)) + \overline{\lim_{n\to\infty}}\,\frac{1}{n}\,b(u_1^n)\log_{\mathrm{D}}(\mathrm{D}r) && (8.65)\\ &= \overline{\lim_{n\to\infty}}\,\frac{1}{n}\,b(u_1^n)\log_{\mathrm{D}}(b(u_1^n)). && (8.66)\end{aligned}$$

Comparing with Theorem 8.10, we see that as n tends to infinity, the tree-structured Lempel–Ziv encoding scheme achieves the performance of the best finite state encoding (FSE) scheme!

In particular, if the sequence $\mathbf{u}$ is the output of a random information source that is stationary and ergodic, one could design a FSE that implements the adaptive Huffman coding scheme (see Theorem 7.3) designed for this source and for a large enough block parser M that will guarantee that a compression is achieved that with high probability is arbitrarily close to the source's entropy rate. Theorem 8.10 and (8.66) now show that the tree-structured Lempel–Ziv encoding scheme will compress such a sequence to the entropy rate, too.

Theorem 8.11 (Performace of Tree-Structured Lempel–Ziv Encoding). Asymptotically, as n tends to infinity, the achieved compression of the tree-structured Lempel–Ziv coding scheme is as good the compression of the best finite state encoding scheme with an arbitrary, but fixed number of states.

When the tree-structured Lempel–Ziv coding scheme is applied to the output of a stationary and ergodic source, then

$$\overline{\lim_{n\to\infty}}\,\frac{1}{n}\,l_{\mathrm{LZ}}(u_1^n) = \mathrm{H}(\{U_k\}) \qquad \text{in probability.} \tag{8.67}$$

Note that the tree-structured Lempel–Ziv encoder is *not* a FSE because the algorithm uses up infinite memory since it keeps track of an ever growing tree. So basically, we have the following trade-off: If we want to compress some infinite sequence $\mathbf{u}$, we could either take the "off the shelf" Lempel–Ziv algorithm or we could custom design a finite state encoder with a finite (but arbitrarily large) number of states. In the long run, both will achieve the same performance. So, picking the Lempel–Ziv coding scheme will be much more preferable because then we do not need to custom design anything, and we do not need to know the particular sequence $\mathbf{u}$ it will be used on. Again, we have found a universal encoding scheme.

Chapter 9

Optimizing Probability Vectors over Concave Functions: Karush–Kuhn–Tucker Conditions

9.1 Introduction

In information theory (and also in many situations elsewhere), we often encounter the problem that we need to optimize a function of a probability mass function (PMF). A famous example of such a problem is the computation of *channel capacity* (see Chapter 12), which involves maximizing mutual information over the probability distribution of the channel input.

In general, such problems are rather difficult as we are not only looking for the optimal value that maximizes a given merit function, but we need to find the optimal *distribution*, i.e., an optimal *vector* where the components of the vector need to satisfy some constraints, namely, they must be nonnegative and need to sum to one. In the lucky situation when the merit function is *concave* (or *convex*), however, the problem actually simplifies considerably. In the following we will show how to solve this special case.

So, the problem under consideration is as follows. Let $f\colon \mathcal{R}_{\mathrm{L}} \to \mathbb{R}$ be a concave function of a probability vector[1] $\boldsymbol{\alpha} \in \mathcal{R}_{\mathrm{L}}$. We want to find the maximum value of $f(\cdot)$ over all possible ("legal") values of $\boldsymbol{\alpha}$:

$$\max_{\substack{\boldsymbol{\alpha}\\ \text{s.t. } \alpha_\ell \geq 0,\ \ell=1,\ldots,\mathrm{L}\\ \text{and } \sum_{\ell=1}^{\mathrm{L}} \alpha_\ell = 1}} f(\boldsymbol{\alpha}). \tag{9.1}$$

[1] By *probability vector* we mean a vector whose components are nonnegative and sum to 1, see Definition 9.3. Also $\mathcal{R}_{\mathrm{L}}$ is defined below, see Definition 9.5.

9.2 Convex Regions and Concave Functions

We start by a quick review of convex regions and concave functions.

Definition 9.1. A region $\mathcal{R} \subset \mathbb{R}^L$ is said to be *convex* if for each pair of points in $\mathcal{R}$, the straight line between those points stays in $\mathcal{R}$:

$$\forall \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}: \quad \theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta} \in \mathcal{R}, \quad \text{for } 0 \le \theta \le 1. \tag{9.2}$$

Example 9.2. Some examples of convex or nonconvex regions are shown in Figure 9.1. ◊

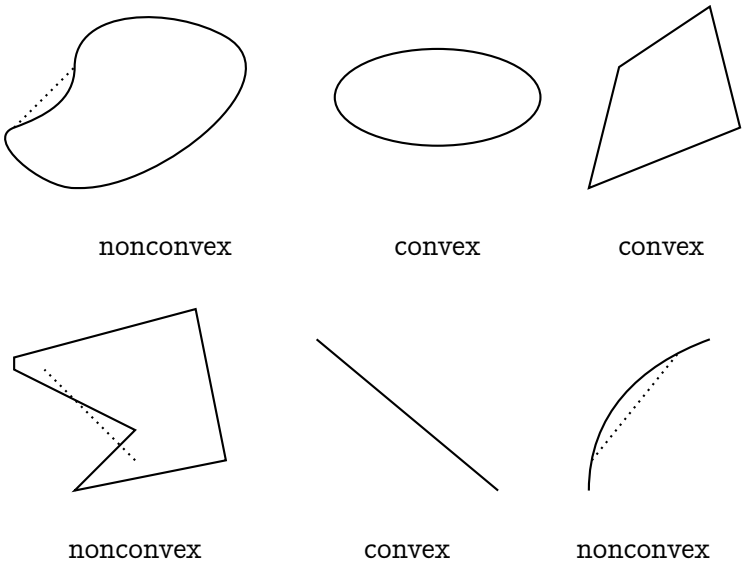


Figure 9.1: Examples of convex and nonconvex regions in $\mathbb{R}^2$.

We are especially interested in one particular convex region: the *region of probability vectors*.

Definition 9.3. A vector is called *probability vector* if all its components are nonnegative and they sum up to 1.

A probability vector is an elegant way to write a probability distribution over a finite alphabet.

Example 9.4. If a RV X takes value in $\{1,2,3,4,5\}$ with probabilities

$$P_X(x) = \begin{cases} \frac{1}{2} & \text{if } x = 1, \\ \frac{1}{4} & \text{if } x = 2, \\ \frac{1}{8} & \text{if } x = 3, \\ \frac{1}{16} & \text{if } x = 4, \\ \frac{1}{16} & \text{if } x = 5, \end{cases} \tag{9.3}$$

then its distribution can be written as probability vector

$$\mathbf{p} = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}\right)^{\mathsf{T}}. \tag{9.4}$$

◊

Definition 9.5. For any $\mathrm{L} \in \mathbb{N}$, we define the *region of probability vectors* $\mathcal{R}_{\mathrm{L}} \subset \mathbb{R}^{\mathrm{L}}$ as

$$\mathcal{R}_{\mathrm{L}} \triangleq \left\{\mathbf{p} \in \mathbb{R}^{\mathrm{L}} \colon p_\ell \geq 0, \ell \in \{1, \ldots, \mathrm{L}\}, \text{ and } \sum_{\ell=1}^{\mathrm{L}} p_\ell = 1\right\}. \tag{9.5}$$

Lemma 9.6. The region of probability vectors $\mathcal{R}_{\mathrm{L}}$ is convex.

Proof: Let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}_{\mathrm{L}}$ be probability vectors. Define $\boldsymbol{\gamma} \triangleq \theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}$ for any $\theta \in [0,1]$. We need to show that $\boldsymbol{\gamma}$ is also a probability vector. Obviously, $\gamma_\ell \geq 0$ for all $\ell \in \{1, \ldots, \mathrm{L}\}$. Moreover,

$$\sum_{\ell=1}^{\mathrm{L}} \gamma_\ell = \sum_{\ell=1}^{\mathrm{L}} (\theta\alpha_\ell + (1-\theta)\beta_\ell) \tag{9.6}$$

$$= \theta \underbrace{\sum_{\ell=1}^{\mathrm{L}} \alpha_\ell}_{=1} + (1-\theta) \underbrace{\sum_{\ell=1}^{\mathrm{L}} \beta_\ell}_{=1} \tag{9.7}$$

$$= \theta + (1-\theta) = 1. \tag{9.8}$$

Hence, all components of $\boldsymbol{\gamma}$ are nonnegative and they sum up to 1, i.e., $\boldsymbol{\gamma} \in \mathcal{R}_{\mathrm{L}}$. □

Definition 9.7. Let $\mathcal{R} \subset \mathbb{R}^{\mathrm{L}}$ be a convex region. A real-valued function $f \colon \mathcal{R} \to \mathbb{R}$ is said to be *concave over a convex region* $\mathcal{R}$ if $\forall\, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}$ and $0 < \theta < 1$:

$$\theta f(\boldsymbol{\alpha}) + (1-\theta) f(\boldsymbol{\beta}) \leq f(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}). \tag{9.9}$$

It is called *strictly concave* if the above inequality is strict.

Note that a function can only be defined to be concave *over a convex region* because otherwise the right-hand side of (9.9) might not be defined for some values of $0 < \theta < 1$.

Also note that the definition of *convex* functions is identical apart from a reversed sign. This means that if $f(\cdot)$ is concave, then $-f(\cdot)$ is convex and vice-versa. So, all results can easily be adapted for convex functions and we only need to treat concave functions here.

Example 9.8. See Figure 9.2 for an example of a concave function. ◊

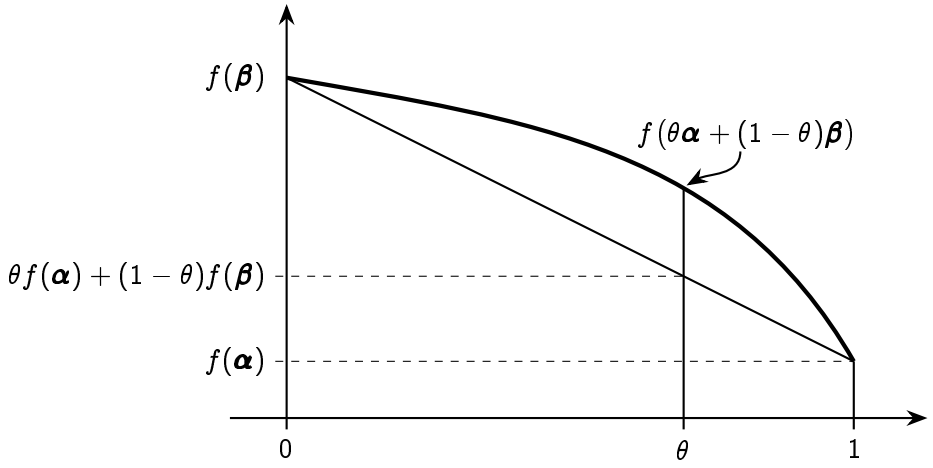


Figure 9.2: Example of a concave function.

Lemma 9.9. Concave functions have the following properties:

1. If $f_1(\cdot), \ldots, f_n(\cdot)$ are concave and $c_1, \ldots, c_n$ are positive numbers, then

$$\sum_{k=1}^{n} c_k f_k(\cdot) \tag{9.10}$$

   is concave with strict concavity if at least one $f_k(\cdot)$ is strictly concave.

2. If $f(\cdot)$ is monotonically increasing and concave over a convex region $\mathcal{R}_1$, and $g(\cdot)$ is concave over $\mathcal{R}_2$ and has an image $\mathcal{I} \subseteq \mathcal{R}_1$, then $f(g(\cdot))$ is concave.

3. For a one-dimensional vector α, if

$$\frac{\partial^2 f(\alpha)}{\partial \alpha^2} \le 0 \tag{9.11}$$

   everywhere in an interval, then $f(\alpha)$ is concave in this interval.

4. **Jensen Inequality:** for any concave function $f(\cdot)$ and any random vector $\mathbf{X} \in \mathbb{R}^{\mathrm{L}}$ taking on M values,[2] we have

$$\mathsf{E}[f(\mathbf{X})] \le f(\mathsf{E}[\mathbf{X}]). \tag{9.12}$$

Proof: We only have a short look at the last property. Let $(\theta_1, \ldots, \theta_{\mathrm{M}})$ be the probabilities of the M values of $\mathbf{X}$, i.e., $\theta_m \ge 0$ and $\sum_{m=1}^{\mathrm{M}} \theta_m = 1$.

[2] Actually, the Jensen Inequality also holds for random vectors with an alphabet of infinite size or even with a continuous alphabet. Recall the discussion in Section 2.4.

Let $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_M$ be a set of vectors in a convex region $\mathcal{R} \subset \mathbb{R}^L$ where $f(\cdot)$ is concave. Then

$$\sum_{m=1}^{M} \theta_m f(\boldsymbol{\alpha}_m) \le f\left(\sum_{m=1}^{M} \theta_m \boldsymbol{\alpha}_m\right) \tag{9.13}$$

by definition of concavity (to be precise, for $M = 2$ it is by definition, for $M > 2$ we use the definition recursively). Let $\mathbf{X}$ be a random vector that takes on the values $\boldsymbol{\alpha}_m$ with probability θ_m, $m = 1, \ldots, M$. The result now follows from (9.13). □

9.3 Maximizing Concave Functions

Why are concave functions interesting? We will see that they are relatively easy to maximize over a given region.

We firstly consider the simpler situation where we do not worry about the normalization of a probability vector's components (i.e., we ignore that the components need to sum to 1). Instead, let $\mathcal{R} \subset \mathbb{R}^L$ be a convex region, let $\boldsymbol{\alpha} \in \mathcal{R}$ be an L-dimensional real vector, and let $f(\boldsymbol{\alpha})$ be a concave function over $\mathcal{R}$. We would now like to find the maximum of f over the convex region $\mathcal{R}$.

In a first straightforward attempt to solve the problem we remember from calculus that an optimum is reached if the derivative is zero. So we ask the optimum vector $\boldsymbol{\alpha}^*$ to satisfy the following conditions:

$$\frac{\partial f(\boldsymbol{\alpha}^*)}{\partial \alpha_\ell^*} \stackrel{!}{=} 0, \quad \ell \in \{1, \ldots, L\}. \tag{9.14}$$

But there is a problem with these conditions: The set of equations (9.14) might not have any solution, or, if it has a solution in $\mathbb{R}$, this solution might not be in $\mathcal{R}$!

Luckily, we have at least the following lemma.

Lemma 9.10. If there exists a solution $\boldsymbol{\alpha}^* \in \mathbb{R}^L$ that simultaneously satisfies (9.14) and the constraint $\boldsymbol{\alpha}^* \in \mathcal{R}$, then this solution $\boldsymbol{\alpha}^*$ maximizes $f(\cdot)$.

Proof: By contradiction assume that there exists a $\boldsymbol{\beta} \in \mathcal{R}$, $\boldsymbol{\beta} \neq \boldsymbol{\alpha}^*$, with $f(\boldsymbol{\beta}) > f(\boldsymbol{\alpha}^*)$. Then the line from $f(\boldsymbol{\alpha}^*)$ to $f(\boldsymbol{\beta})$ is increasing. But because $f(\cdot)$ is concave, the slope of $f(\cdot)$ at $\boldsymbol{\alpha}^*$ must be larger than the slope of the line. This is a contradiction to (9.14). □

So what about the situation when we cannot find a correct solution to (9.14)? Since we have assumed that $f(\cdot)$ is concave we note that such a situation can only happen if either $f(\cdot)$ has no "peak" where the derivative is zero, or this peak occurs outside of the given convex region $\mathcal{R}$. See Figure 9.3 for an illustration. In that case, $f(\cdot)$ is maximized by an $\boldsymbol{\alpha}^*$ on the boundary

of the convex region $\mathcal{R}$ (where we can think of infinity to be a "boundary", too).

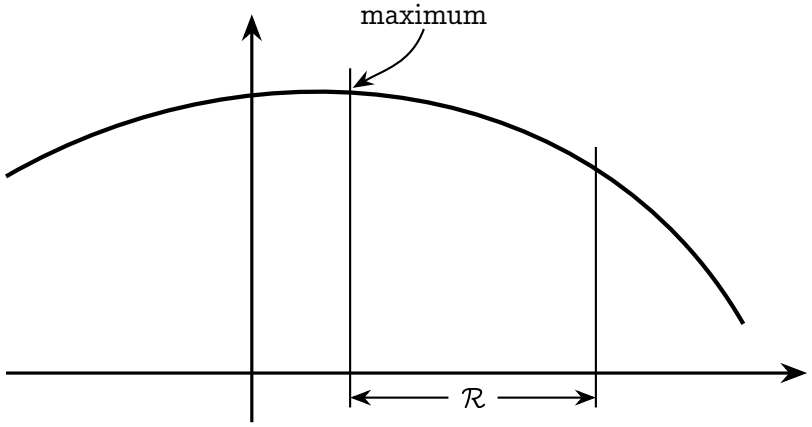


Figure 9.3: Maximum is achieved on the boundary of the region.

Note that in such a case $\boldsymbol{\alpha}^*$ can only maximize $f(\cdot)$ over $\mathcal{R}$ if $f(\cdot)$ is *decreasing* when going from the boundary point $\boldsymbol{\alpha}^*$ into the region! This gives us an idea on how to fix (9.14): Instead of (9.14), an optimal vector $\boldsymbol{\alpha}^*$ should satisfy

$$\frac{\partial f(\boldsymbol{\alpha}^*)}{\partial \alpha_\ell^*} \stackrel{!}{=} 0, \quad \forall\, \ell \text{ such that } \alpha_\ell^* \text{ is inside the region } \mathcal{R}, \tag{9.15a}$$

$$\frac{\partial f(\boldsymbol{\alpha}^*)}{\partial \alpha_\ell^*} \stackrel{!}{\leq} 0, \quad \forall\, \ell \text{ such that } \alpha_\ell^* \text{ is on the boundary of } \mathcal{R}. \tag{9.15b}$$

We will prove (9.15) later. By the way, why do we have "$\leq$" in (9.15b) and not "$\geq$"? Note that the maximum might also occur on the right boundary, in which case we want the derivative be positive! The answer here lies in the way how we compute the derivative. See Appendix 9.A for more details.

Now note that we are actually interested in the situation of probability vectors, where we have seen that the region of probability vectors is convex. Since $\alpha_\ell \geq 0$, its boundary is $\alpha_\ell = 0$, $\ell \in \{1, \ldots, \mathrm{L}\}$. However, beside this nonnegativity condition, probability vectors also have to satisfy

$$\sum_{\ell=1}^{\mathrm{L}} \alpha_\ell = 1. \tag{9.16}$$

Unfortunately, this condition is not a boundary because the conditions on the different components are highly dependent.

So how shall we include this condition into our maximization problem?

We use an old trick deep in the mathematicians' box of tricks: Since we want (9.16) to hold, we also have that

$$\sum_{\ell=1}^{\mathrm{L}} \alpha_\ell - 1 = 0 \tag{9.17}$$

or, for any constant $\lambda \in \mathbb{R}$,

$$\lambda\left(\sum_{\ell=1}^{\mathrm{L}} \alpha_\ell - 1\right) = 0. \tag{9.18}$$

Hence, any $\boldsymbol{\alpha}$ that maximizes $f(\cdot)$ must also maximize the function

$$\mathrm{F}(\boldsymbol{\alpha}) \triangleq f(\boldsymbol{\alpha}) - \lambda\left(\sum_{\ell=1}^{\mathrm{L}} \alpha_\ell - 1\right). \tag{9.19}$$

And this has to be true for **every** choice of λ! Now the trick is to try to find an $\boldsymbol{\alpha}^*(\lambda)$ that maximizes $\mathrm{F}(\cdot)$ instead of $f(\cdot)$ and to ignore the constraint (9.16) for the moment. Note that such an optimal $\boldsymbol{\alpha}^*(\lambda)$ will depend on λ and it will in general not satisfy (9.16). But since we can choose λ freely, we can tweak it in such a way that the solution also satisfies (9.16). So, then we have found one solution $\boldsymbol{\alpha}^*$ that maximizes $\mathrm{F}(\cdot)$ and satisfies (9.16). However, in exactly this case, $\boldsymbol{\alpha}^*$ also maximizes $f(\cdot)$. We are done!

So, the problem hence reduces to finding the maximum of (9.19). We therefore apply (9.15) to $\mathrm{F}(\cdot)$ instead of $f(\cdot)$. Noting that

$$\frac{\partial \mathrm{F}(\boldsymbol{\alpha})}{\partial \alpha_\ell} = \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell} - \lambda, \tag{9.20}$$

we obtain from (9.15):

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell} \overset{!}{=} \lambda \qquad \forall\, \ell \text{ such that } \alpha_\ell > 0, \tag{9.21a}$$

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell} \overset{!}{\le} \lambda \qquad \forall\, \ell \text{ such that } \alpha_\ell = 0, \tag{9.21b}$$

where λ is chosen such that (9.16) is satisfied. The mathematicians call the constant λ the *Lagrange multiplier*. Let us now summarize and formally prove our insights in the following theorem.

Theorem 9.11 (Karush–Kuhn–Tucker (KKT) Conditions).
Let $f\colon \mathcal{R}_{\mathrm{L}} \to \mathbb{R}$ be concave over the convex region $\mathcal{R}_{\mathrm{L}}$ of L-dimensional probability vectors. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{\mathrm{L}})^{\mathsf{T}} \in \mathcal{R}_{\mathrm{L}}$ be a probability vector. Assume that $\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell}$ are defined and continuous over $\mathcal{R}_{\mathrm{L}}$ with the possible exception that

$$\lim_{\alpha_\ell \downarrow 0} \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell} = +\infty \tag{9.22}$$

is allowed. Then

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell} \overset{!}{=} \lambda \qquad \forall\, \ell \text{ such that } \alpha_\ell > 0, \tag{9.23a}$$

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell} \overset{!}{\le} \lambda \qquad \forall\, \ell \text{ such that } \alpha_\ell = 0, \tag{9.23b}$$

are **necessary** and **sufficient** conditions on $\boldsymbol{\alpha}$ to maximize $f(\cdot)$ over $\mathcal{R}_{\mathrm{L}}$. Note that the Lagrange multiplier λ needs to be chosen such that

$$\sum_{\ell=1}^{\mathrm{L}} \alpha_\ell = 1. \tag{9.24}$$

Proof: **Sufficiency:** Assume (9.23a) and (9.23b) hold for some λ and $\boldsymbol{\alpha}$. Then we want to show that, for any $\boldsymbol{\beta}$, $f(\boldsymbol{\beta}) \le f(\boldsymbol{\alpha})$. We know from concavity that for any $\theta \in [0,1]$

$$\theta f(\boldsymbol{\beta}) + (1-\theta) f(\boldsymbol{\alpha}) \le f(\theta\boldsymbol{\beta} + (1-\theta)\boldsymbol{\alpha}). \tag{9.25}$$

Hence,

$$\begin{aligned}
f(\boldsymbol{\beta}) - f(\boldsymbol{\alpha}) &\le \frac{f(\theta\boldsymbol{\beta} + (1-\theta)\boldsymbol{\alpha}) - f(\boldsymbol{\alpha})}{\theta} && (9.26)\\
&= \frac{f(\boldsymbol{\alpha} + \theta(\boldsymbol{\beta}-\boldsymbol{\alpha})) - f(\boldsymbol{\alpha})}{\theta} && (9.27)\\
&\to \sum_{\ell=1}^{\mathrm{L}} \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell}(\beta_\ell - \alpha_\ell) \qquad \text{for } \theta \downarrow 0 && (9.28)\\
&\le \sum_{\ell=1}^{\mathrm{L}} \lambda(\beta_\ell - \alpha_\ell) && (9.29)\\
&= \lambda\left(\sum_{\ell=1}^{\mathrm{L}} \beta_\ell - \sum_{\ell=1}^{\mathrm{L}} \alpha_\ell\right) && (9.30)\\
&= \lambda(1-1) = 0. && (9.31)
\end{aligned}$$

Here in (9.28) we choose θ to tend to 0. This will cause the expression to tend to the sum of the derivatives. If this expression looks confusing to you, you might remember the one-dimensional case, which should look familiar:

$$\begin{aligned}
\lim_{\theta\downarrow 0} \frac{f(\theta\beta + (1-\theta)\alpha) - f(\alpha)}{\theta} &= \lim_{\theta\downarrow 0} \frac{f(\alpha + \theta(\beta-\alpha)) - f(\alpha)}{\theta(\beta-\alpha)}(\beta-\alpha) && (9.32)\\
&= \frac{\partial f(\alpha)}{\partial \alpha}(\beta-\alpha). && (9.33)
\end{aligned}$$

The subsequent inequality (9.29) follows by assumption from (9.23a) and (9.23b), and the last equality follows because both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are probability vectors, which sum to 1.

Necessity: Let $\boldsymbol{\alpha}$ maximize f and assume for the moment that the derivatives of f are continuous at $\boldsymbol{\alpha}$. Since $\boldsymbol{\alpha}$ maximizes f, we have for any $\boldsymbol{\gamma}$:

$$f(\boldsymbol{\gamma}) - f(\boldsymbol{\alpha}) \le 0. \tag{9.34}$$

Now choose $\boldsymbol{\gamma} \triangleq \theta\boldsymbol{\beta} + (1-\theta)\boldsymbol{\alpha}$ for some $\boldsymbol{\beta}$ and let $\theta \downarrow 0$. Then similarly to above we get

$$f(\theta\boldsymbol{\beta} + (1-\theta)\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}) \le 0 \tag{9.35}$$

$$\Longrightarrow \quad \frac{f(\theta\boldsymbol{\beta} + (1-\theta)\boldsymbol{\alpha}) - f(\boldsymbol{\alpha})}{\theta} \le 0 \tag{9.36}$$

$$\Longrightarrow \quad \sum_{\ell=1}^{\mathrm{L}} \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell}(\beta_\ell - \alpha_\ell) \le 0, \quad \text{as } \theta \downarrow 0. \tag{9.37}$$

Since $\boldsymbol{\alpha}$ is a probability vector, its components must sum to 1. Hence, there must be at least one component that is strictly larger than zero. For convenience, let us assume that this component is α_1. We define $\mathbf{i}^{(k)}$ to be a unit vector with a 1 in position k and zeros everywhere else. Then we fix some $k \in \{1, \ldots, \mathrm{L}\}$ and choose

$$\boldsymbol{\beta} \triangleq \boldsymbol{\alpha} + \epsilon\mathbf{i}^{(k)} - \epsilon\mathbf{i}^{(1)}. \tag{9.38}$$

This is a probability vector as long as ϵ is chosen correctly, as can be seen as follows:

$$\sum_{\ell=1}^{\mathrm{L}} \beta_\ell = \underbrace{\sum_{\ell=1}^{\mathrm{L}} \alpha_\ell}_{=1} + \epsilon \sum_{\ell=1}^{\mathrm{L}} (i_\ell^{(k)} - i_\ell^{(1)}) = 1 + \epsilon(1-1) = 1, \tag{9.39}$$

i.e., $\boldsymbol{\beta}$ always sums to 1. So we only have to make sure that the components are nonnegative. But the components of $\boldsymbol{\alpha}$ are for sure nonnegative. So for a nonnegative choice of ϵ we are only in danger at the first component where we subtract ϵ. But we have assumed that α_1 is strictly larger than 0. Hence, as long as we choose ϵ smaller than α_1, we are OK: we fix ϵ such that $0 < \epsilon \le \alpha_1$. Finally, we choose $\lambda \triangleq \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_1}$.

With these choices we can evaluate (9.37):

$$\sum_{\ell=1}^{\mathrm{L}} \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell}(\beta_\ell - \alpha_\ell) = \epsilon\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k} - \epsilon\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_1} = \epsilon\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k} - \epsilon\lambda \overset{!}{\le} 0. \tag{9.40}$$

So, when we divide by ϵ (which is positive!), we get

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k} \overset{!}{\le} \lambda \tag{9.41}$$

which corresponds to (9.23b).

Now note that if $\alpha_k > 0$ then ϵ can also be chosen to be negative with $\boldsymbol{\beta}$ still having all components nonnegative. In this case we can choose ϵ such that $-\alpha_k \le \epsilon < 0$. If we then divide (9.40) by ϵ, we get[3]

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k} \overset{!}{\ge} \lambda. \tag{9.42}$$

Equations (9.41) and (9.42) can only be satisfied simultaneously if

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k} \overset{!}{=} \lambda, \tag{9.43}$$

which corresponds to (9.23a).

It only remains to consider the case where the derivative is unbounded if one component tends to zero. Hence, assume that for a given k, $\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k} = +\infty$ for $\alpha_k \downarrow 0$. Again, we assume $\alpha_1 > 0$ and therefore $\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_1}$ must be continuous and well-behaved by assumption, i.e., $k \neq 1$.[4] We now show that $\boldsymbol{\alpha}$ with $\alpha_k = 0$ cannot achieve the maximum:

$$\begin{aligned}
&\frac{f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)} - \epsilon \mathbf{i}^{(1)}) - f(\boldsymbol{\alpha})}{\epsilon} \\
&= \frac{f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)} - \epsilon \mathbf{i}^{(1)}) - f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)})}{\epsilon} + \frac{f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)}) - f(\boldsymbol{\alpha})}{\epsilon} && (9.44)\\
&= -\frac{f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)} + (-\epsilon)\mathbf{i}^{(1)}) - f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)})}{(-\epsilon)} + \frac{f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)}) - f(\boldsymbol{\alpha})}{\epsilon} && (9.45)\\
&\overset{\epsilon \downarrow 0}{\to} \underbrace{-\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_1}}_{\substack{> -\infty \\ \text{by assumption} \\ \text{because } \alpha_1 > 0}} + \underbrace{\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k}}_{\substack{= \infty \\ \text{by assumption}}} = \infty > 0. && (9.46)
\end{aligned}$$

Hence,

$$f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(k)} - \epsilon \mathbf{i}^{(1)}) > f(\boldsymbol{\alpha}) \tag{9.47}$$

and $\boldsymbol{\alpha}$ cannot achieve the maximum. □

9.A Appendix: Slope Paradox

The aware reader might want to adapt the example of Figure 9.3 slightly to get a situation shown in Figure 9.4.

Now obviously the slope at the boundary point where the maximum is achieved is *positive* and not negative as claimed in (9.15b). So it seems that (9.15b) is wrong for this case. How can we save ourselves?

[3] Note that since we divide by a negative value, we have to change the inequality sign!

[4] Note that we only allow the derivative to become infinite for α_ℓ tending to zero. Since we have here that $\alpha_1 > 0$ this derivative must be well-behaved!

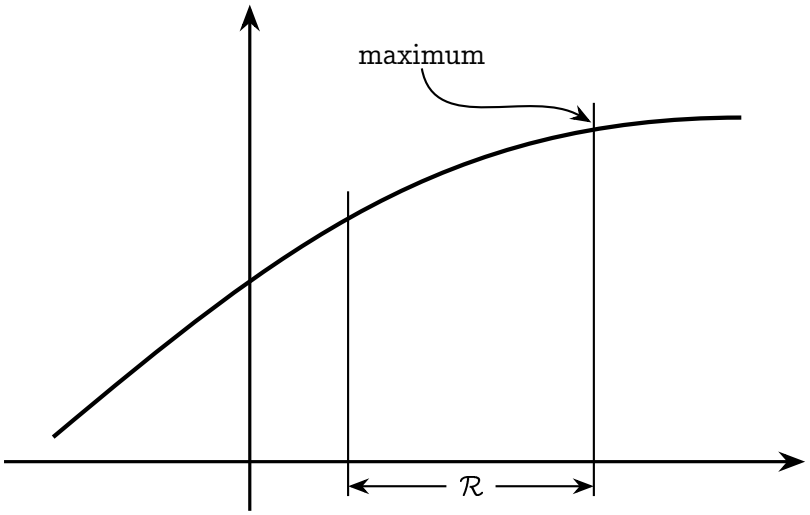


Figure 9.4: Slope paradox: The maximum is achieved on the boundary of the region, but now the slope seems to be positive instead of negative as we have claimed in (9.15b).

The clue lies in the way we compute derivatives. The actual idea behind (9.15a) and (9.15b) was as follows:

slope is zero	for all components that lie *inside* the region,	(9.48)
slope is decreasing when walking from the boundary *into* the region	for all components that are on the boundary.	(9.49)

Now, strictly speaking, our function f is only defined in the region $\mathcal{R}$. Hence we need to compute the slope accordingly. In the situation of Figure 9.3 the slope is computed as follows:

$$\lim_{\epsilon \downarrow 0} \frac{f(\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(\ell)}) - f(\boldsymbol{\alpha})}{\epsilon} = \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell}, \tag{9.50}$$

which then leads to (9.15).

In the case of Figure 9.4, however, this approach is not possible because $\boldsymbol{\alpha} + \epsilon \mathbf{i}^{(\ell)} \notin \mathcal{R}$. Instead, there we need to compute the slope as follows:

$$\lim_{\epsilon \downarrow 0} \frac{f(\boldsymbol{\alpha} - \epsilon \mathbf{i}^{(\ell)}) - f(\boldsymbol{\alpha})}{\epsilon} = -\lim_{\epsilon \downarrow 0} \frac{f(\boldsymbol{\alpha} - \epsilon \mathbf{i}^{(\ell)}) - f(\boldsymbol{\alpha})}{-\epsilon} = -\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell}, \tag{9.51}$$

i.e., condition (9.15b) then actually becomes

$$\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell} \geq 0. \tag{9.52}$$

So, our paradox is solved if we understand that by $\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_\ell}$ we mean the derivative that must be computed from the boundary towards *inside* the convex region. We learn that we always need to check the concrete case and choose the sign of the inequality accordingly!

Note that in the case of the region of probability vectors $\mathcal{R}_{\mathrm{L}}$, we did it correctly because the situation corresponds to Figure 9.3: The region is only bounded *below* by zero, i.e., the way we state equation (9.15b) is correct (as it must have been, since we have proven it in Theorem 9.11).

Chapter 10

Gambling and Horse Betting

In this chapter we make an excursion into an area that at first sight seems to have little to do with information theory: gambling. However, this impression is wrong! It turns out that we can describe optimal betting strategies very elegantly using some of our information-theoretic quantities.

The setup of the gambling in this chapter is very simple. As basic example we assume a horse gambling game, where one can bet money on a horse in a horse race and will receive money if the horse wins. However, in concept the ideas that are presented here can also be applied in a much wider context, e.g., to the stock-markets or to large scale investment portfolios.

10.1 Problem Setup

Consider a set $\mathcal{X}$ of m horses. Let the random variable $X \in \mathcal{X}$ be the winning horse with PMF

$$P_X(x) = p(x) = \begin{cases} p_i & \text{if } x = i, \\ 0 & \text{otherwise,} \end{cases} \tag{10.1}$$

where p_i denotes the probability that horse i wins. Moreover, let o_i denote the return per dollar (odds) if horse i wins.

Example 10.1. Assume we bet 5 dollars on horse 1 with odds $o_1 = 2$. Then, if horse 1 wins, we will receive 10 dollars; if horse 1 does not win, our 5 dollars are lost and we receive nothing. ♢

We use $\mathbf{b}$ to denote the betting strategy, i.e., b_i is the *proportion* of the gambler's total wealth that he bets on horse i. This means that $\mathbf{b}$ is a *percentage*, i.e., the gambler needs to decide first how much money he wants to risk in gambling and then decide how to bet with this money. Or in mathematical terms:

$$\sum_{i=1}^{m} b_i = 1. \tag{10.2}$$

Remark 10.2. For convenience we use two equivalent forms of notation: a vector notation and a functional notation. For example, in the case of the betting strategy we have

$b_i = b(i)$ the proportion of the money that is bet on horse i,
$\mathbf{b} = b(\cdot)$ the complete betting strategy.

Similarly we write $p_i = p(i)$, $\mathbf{p} = p(\cdot)$, $o_i = o(i)$, and $\mathbf{o} = o(\cdot)$. △

10.2 Optimal Gambling Strategy

How shall we now invest our money? Shall we bet on the horse that has highest chance of winning? Or shall we rather favor the horse with the best odds?

Example 10.3. Consider a race with $m = 3$ horses, with the following winning probabilities:

$$p_1 = 0.7, \qquad p_2 = 0.2, \qquad p_3 = 0.1, \tag{10.3}$$

and with the following odds:

$$o_1 = 1.2, \qquad o_2 = 4.5, \qquad o_3 = 8.5. \tag{10.4}$$

On which horse would you put your money? On horse 1 that has the highest winning probability? Or on horse 3 that has the highest odds?

Actually, if you are a slightly more capable gambler, you will bet your money on the horse with the highest *expected return*, i.e., the one with the largest product $p_i o_i$:

$$p_1 o_1 = 0.84, \qquad p_2 o_2 = 0.9, \qquad p_3 o_3 = 0.85, \tag{10.5}$$

i.e., on the second horse. But note that also this strategy is risky because you might lose all your money in one blow! Hence, we realize that the really careful gambler should look at the *long-term* expected return. ◊

So, we realize that we need to keep participating in many races. For the moment we will assume that all races are IID, so we therefore also assume that we stick with the same betting strategy during all races. Let S_n be the *growth factor* of our wealth after participating in n races, and let X_k denote the winning horse in race k.

Example 10.4. Returning to the horse race of Example 10.3, assume that you bet 50% of your money on horse 1 and 50% on horse 2. Hence,

$$b_1 = 0.5, \qquad b_2 = 0.5, \qquad b_3 = 0. \tag{10.6}$$

Assume further that you decide to gamble with a total of $\gamma_0 = 800$ dollars.

It so happens that in the first race horse 2 wins, $X_1 = 2$. Hence, the $b_1 \cdot \gamma_0 = 0.5 \cdot 800 = 400$ dollars that you have bet on horse 1 are lost. But the other 400 dollars that you have bet on horse 2 are multiplied by a factor of 4.5, i.e., you get $4.5 \cdot 400 = 1800$ dollars back. Therefore your current wealth now is

$$\gamma_1 = \gamma_0 \cdot \underbrace{b(2) \cdot o(2)}_{= s_1} = 800 \cdot 0.5 \cdot 4.5 = 1800. \tag{10.7}$$

Note that the growth factor turned out to be $s_1 = 2.25$.

Let's say that in the second race horse 1 wins, $X_2 = 1$. This time the $b_2 \cdot \gamma_1 = 0.5 \cdot 1800 = 900$ dollars that you have bet on horse 2 are lost, but you gain $o_1 \cdot (b_1 \cdot \gamma_1) = 1.2 \cdot (0.5 \cdot 1800)$ dollars from horse 1. So now your wealth is

$$\gamma_2 = \gamma_1 \cdot b(1) \cdot o(1) = 1800 \cdot 0.5 \cdot 1.2 = 1080. \tag{10.8}$$

Note that using (10.7) we can write this also as follows:

$$\gamma_2 = \underbrace{\gamma_0 \cdot s_1}_{= \gamma_1} \cdot b(1) \cdot o(1) = \gamma_0 \cdot \underbrace{s_1 \cdot b(1) \cdot o(1)}_{= s_2}, \tag{10.9}$$

where s_2 denotes the growth factor of your wealth during the first two races: $s_2 = 2.25 \cdot 0.6 = 1.35$.

Suppose that in the third race, horse 1 wins again, $X_3 = 1$. Your wealth therefore develops as follows:

$$\gamma_3 = \gamma_2 \cdot b(1) \cdot o(1) = \gamma_0 \cdot \underbrace{s_2 \cdot b(1) \cdot o(1)}_{= s_3}. \tag{10.10}$$

And so on. ◊

From Example 10.4 we now understand that

$$\begin{aligned} S_1 &= b(X_1)\, o(X_1), && (10.11)\\ S_2 &= S_1 \cdot b(X_2)\, o(X_2), && (10.12)\\ S_3 &= S_2 \cdot b(X_3)\, o(X_3), && (10.13)\\ &\vdots \\ \Longrightarrow\ S_n &= \prod_{k=1}^{n} b(X_k)\, o(X_k) && (10.14) \end{aligned}$$

and that the wealth after n races is

$$\gamma_n = \gamma_0 \cdot S_n. \tag{10.15}$$

You note that γ_n only depends on the starting wealth γ_0 and the growth factor S_n. We therefore usually directly look at S_n without worrying about the exact amount of money we use for betting.

Example 10.5. We briefly return to Example 10.4 and point out that once horse 3 wins (which is unlikely but possible!), you are finished: Since $b(3) = 0$, your wealth is reduced to zero and you cannot continue to bet. So it is quite obvious that the chosen betting strategy is not good in the long run. ◊

We go back to our question of deciding about a good betting strategy. From the discussion above we conclude that we would like to find a betting strategy that maximizes S_n, or rather, since S_n is random, that maximizes the expected value of S_n for large n. Unfortunately, the expression (10.14) is rather complicated. In order to simplify it, we introduce a logarithm:

$$\begin{aligned}
\frac{1}{n}\log_2 S_n &= \frac{1}{n}\log_2\left(\prod_{k=1}^{n} b(X_k)\,o(X_k)\right) && (10.16)\\
&= \frac{1}{n}\sum_{k=1}^{n}\log_2\big(b(X_k)\,o(X_k)\big) && (10.17)\\
&\overset{n\to\infty}{\to} \mathsf{E}\big[\log_2\big(b(X)\,o(X)\big)\big] \qquad \text{in probability} && (10.18)
\end{aligned}$$

by the weak law of large numbers. So we define the following new quantity.

Definition 10.6. We define the *doubling rate* $W(\mathbf{b},\mathbf{p},\mathbf{o})$ as

$$\begin{aligned}
W(\mathbf{b},\mathbf{p},\mathbf{o}) &\triangleq \mathsf{E}\big[\log_2\big(b(X)\,o(X)\big)\big] && (10.19)\\
&= \sum_x p(x)\log_2\big(b(x)\,o(x)\big). && (10.20)
\end{aligned}$$

From (10.18) we see that, for n large, we have with high probability

$$S_n \approx 2^{nW(\mathbf{b},\mathbf{p},\mathbf{o})}. \tag{10.21}$$

This also explains the name of W: If $W = 1$, then we expect to *double* our wealth after each race; if $W = 0.1$, we expect to *double* our wealth after every 10 races; and if W is negative, we will be broke sooner or later...

So our new aim is to maximize the doubling rate by choosing the best possible betting strategy $\mathbf{b}$:

$$\begin{aligned}
W^*(\mathbf{p},\mathbf{o}) &\triangleq \max_{\mathbf{b}} W(\mathbf{b},\mathbf{p},\mathbf{o}) && (10.22)\\
&= \max_{\mathbf{b}} \sum_x p(x)\log_2\big(b(x)\,o(x)\big) && (10.23)\\
&= \max_{\mathbf{b}} \left\{\sum_x p(x)\log_2 b(x) + \sum_x p(x)\log_2 o(x)\right\} && (10.24)\\
&= \max_{\mathbf{b}} \sum_x p(x)\log_2 b(x) + \underbrace{\sum_x p(x)\log_2 o(x)}_{\substack{\text{independent of } \mathbf{b}\\ \Longrightarrow \text{ we ignore the odds!}}} && (10.25)
\end{aligned}$$

$$= \max_{\mathbf{b}} \sum_x p(x) \log_2 \left(\frac{b(x)}{p(x)} p(x) \right) + \mathsf{E}[\log_2 o(X)] \tag{10.26}$$

$$= \max_{\mathbf{b}} \{ - \mathscr{D}(\mathbf{p} \| \mathbf{b}) - \mathsf{H}(X) \} + \mathsf{E}[\log_2 o(X)] \tag{10.27}$$

$$= - \min_{\mathbf{b}} \underbrace{\mathscr{D}(\mathbf{p} \| \mathbf{b})}_{\substack{= 0 \text{ for} \\ \mathbf{b}^* = \mathbf{p}}} - \mathsf{H}(X) + \mathsf{E}[\log_2 o(X)] \tag{10.28}$$

$$= -\mathsf{H}(X) + \mathsf{E}[\log_2 o(X)]. \tag{10.29}$$

Here we used relative entropy defined in Definition 3.1.

Hence, we get the following theorem.

Theorem 10.7 (Proportional Gambling is Log-Optimal).
Consider an IID horse race $X \sim \mathbf{p}$ with odds $\mathbf{o}$. The optimum doubling rate is given by

$$W^*(\mathbf{p}, \mathbf{o}) = \mathsf{E}[\log_2 o(X)] - \mathsf{H}(X) \tag{10.30}$$

(where the entropy must be specified in bits) and is achieved by **proportional betting**:

$$\mathbf{b}^* = \mathbf{p}. \tag{10.31}$$

Note the really interesting fact that optimal gambling ignores the odds!

Example 10.8. Consider a race with 2 horses. Horse 1 wins with probability p, $\frac{1}{2} \le p < 1$; and horse 2 wins with probability $1 - p$. Assume even odds $o_1 = o_2 = 2$. If you like risk, you put all your money on horse 1 as this gives highest expected return. But then

$$W([1,0],[p,1-p],[2,2]) = \mathsf{E}\left[\log_2 (b(X)\, o(X))\right] \tag{10.32}$$

$$= p \log_2 (1 \cdot 2) + (1-p) \log_2 (0 \cdot 2) \tag{10.33}$$

$$= -\infty, \tag{10.34}$$

i.e., on the long run your wealth will be

$$S_n \approx 2^{nW} = 2^{-\infty} = 0. \tag{10.35}$$

Actually, this will happen very soon, a couple of races in general is enough...

If you are very much afraid of risk, put half on horse 1 and half on horse 2:

$$S_n = \prod_{k=1}^{n} b(X_k)\, o(X_k) = \prod_{k=1}^{n} \frac{1}{2} \cdot 2 = 1, \tag{10.36}$$

i.e., you will not lose nor gain anything, but for sure keep your money the way it is. The doubling rate in this case is

$$W\Big(\Big[\frac{1}{2},\frac{1}{2}\Big],[p,1-p],[2,2]\Big)=p\log_2\Big(\frac{1}{2}\cdot 2\Big)+(1-p)\log_2\Big(\frac{1}{2}\cdot 2\Big)=0. \tag{10.37}$$

The optimal thing to do is proportional betting:

$$b_1=p,\qquad b_2=1-p, \tag{10.38}$$

which yields a doubling rate

$$W^*\big([p,1-p],[2,2]\big)=\mathsf{E}[\log_2 o(X)]-\mathsf{H}(X)=1-\mathrm{H_b}(p). \tag{10.39}$$

Then your wealth will grow exponentially fast to infinity:

$$S_n\approx 2^{n(1-\mathrm{H_b}(p))}. \tag{10.40}$$

$\lozenge$

10.3 Bookie's Perspective

So far we have concentrated on the gambler's side of the game and simply assumed that the odds were given. However, in an actual horse race, there must be a bookie that offers the game, i.e., the odds.

How shall the bookie decide on the odds o_i that he offers? He, of course, wants to maximize his earning, i.e., minimize the gambler's winning. However, he has no control over the gambler's betting strategy and needs to announce his odds *before* he learns the gambler's bets!

Instead of working with the odds o_i directly, it will turn out to be easier to work with a normalized reciprocal version of it. We define

$$r_i\triangleq\frac{1}{o_i}\cdot\frac{1}{\sum_{j=1}^m\frac{1}{o_j}} \tag{10.41}$$

such that

$$o_i=\frac{c}{r_i} \tag{10.42}$$

with

$$c\triangleq\frac{1}{\sum_{j=1}^m\frac{1}{o_j}}. \tag{10.43}$$

Hence, $(\mathbf{r},c)$ is equivalent to $\mathbf{o}$, i.e., by specifying $\mathbf{r}$ and c the bookie also specifies the odds $\mathbf{o}$.

It is important to note that

$$\sum_{i=1}^{m} r_i = \sum_{i=1}^{m} \frac{1}{o_i} \cdot \frac{1}{\sum_{j=1}^{m} \frac{1}{o_j}} = \frac{1}{\sum_{j=1}^{m} \frac{1}{o_j}} \sum_{i=1}^{m} \frac{1}{o_i} = 1 \tag{10.44}$$

like a proper probability distribution.

We use (10.42) to write

$$\begin{aligned} W(\mathbf{b}, \mathbf{p}, \mathbf{o}) &= \sum_{i=1}^{m} p_i \log_2(b_i o_i) && (10.45)\\ &= \sum_{i=1}^{m} p_i \log_2\left(b_i \frac{c}{r_i}\right) && (10.46)\\ &= \sum_{i=1}^{m} p_i \log_2 c + \sum_{i=1}^{m} p_i \log_2\left(\frac{b_i}{p_i} \cdot \frac{p_i}{r_i}\right) && (10.47)\\ &= \log_2 c - \mathscr{D}(\mathbf{p}\|\mathbf{b}) + \mathscr{D}(\mathbf{p}\|\mathbf{r}). && (10.48) \end{aligned}$$

Note that the bookie has no control over $\mathbf{b}$, but he can minimize the doubling rate $W(\mathbf{b}, \mathbf{p}, \mathbf{o})$ by choosing $\mathbf{r} = \mathbf{p}$! We have proven the following result.

Theorem 10.9. The optimal strategy of the bookie is to offer odds that are **inverse proportional** to the winning probabilities.

Moreover, the bookie can influence the doubling rate by the constant c.

- If $c = 1$, i.e., $\sum_{i=1}^{m} \frac{1}{o_i} = 1$, we have *fair odds*. In this case there exists a betting strategy that contains no risk (but also gives no return): simply bet

  $$b_i = \frac{1}{o_i}. \tag{10.49}$$

  Note that we can do this because we have assumed that $c = 1$, i.e.,

  $$\sum_{i=1}^{m} \frac{1}{o_i} = 1. \tag{10.50}$$

  Then for all n

  $$S_n = \prod_{k=1}^{n} b(X_k)\, o(X_k) = \prod_{k=1}^{n} \frac{1}{o(X_k)}\, o(X_k) = \prod_{k=1}^{n} 1 = 1, \tag{10.51}$$

  i.e., the doubling rate for this risk-free betting strategy is 0.

  In general the doubling rate in the case of fair odds is

$$W(\mathbf{b},\mathbf{p},\mathbf{o}) = \mathscr{D}(\mathbf{p}\|\mathbf{r}) - \mathscr{D}(\mathbf{p}\|\mathbf{b}) \tag{10.52}$$

and has a beautiful interpretation: In reality neither the gambler nor the bookie know the real winning distribution $\mathbf{p}$. But for both it is optimal to choose $\mathbf{b} = \mathbf{p}$ or $\mathbf{r} = \mathbf{p}$, respectively. Hence $\mathbf{b}$ and $\mathbf{r}$ represent the gambler's and the bookie's estimate of the true winning distribution, respectively. The doubling rate is the difference between the "distance" of the bookie's estimate from the true distribution and the "distance" of the gambler's estimate from the true distribution.

A gambler can only make money if his estimate of the true winning distribution is better than the bookie's estimate.

This is a *fair* competition between the gambler and the bookie, and therefore the name "fair odds".

- If $c > 1$, i.e., $\sum_{i=1}^{m} \frac{1}{o_i} < 1$, we have *superfair odds*. In this — not very realistic — scenario, the gambler can win money without risk by making a *Dutch book*:

  $$b_i = \frac{c}{o_i}. \tag{10.53}$$

  He then gets for all n

  $$S_n = \prod_{k=1}^{n} b(X_k)\,o(X_k) = \prod_{k=1}^{n} \frac{c}{o(X_k)}\,o(X_k) = \prod_{k=1}^{n} c = c^n. \tag{10.54}$$

  However, note that a Dutch book is not log-optimal. To maximize W, the gambler still needs to do proportional betting (in which case he ignores the odds).

- If $c < 1$, i.e., $\sum_{i=1}^{m} \frac{1}{o_i} > 1$, we have *subfair odds*. That is often the case in reality: The bookie takes a cut from all bets. This case is more difficult to analyze. We will see in Section 10.6 that in this case it might be clever *not* to bet at all!

10.4 Uniform Fair Odds

Before we start to investigate the realistic situation of subfair odds, we would like to quickly point out some nice properties of the special case of *uniform fair odds:*

$$o_i = m \tag{10.55}$$

where m denotes the number of horses. (Note that

$$\sum_{i=1}^{m} \frac{1}{o_i} = \sum_{i=1}^{m} \frac{1}{m} = 1, \tag{10.56}$$

i.e., these really are fair odds.) Then

$$W^*(\mathbf{p}, \mathbf{o}) = \sum_{i=1}^{m} p_i \log_2 m - \mathrm{H}(X) \tag{10.57}$$

$$= \log_2 m - \mathrm{H}(X) \tag{10.58}$$

and we have the following result.

Theorem 10.10 (Conservation Theorem). For uniform fair odds (10.55), we have

$$W^*(\mathbf{p}, \mathbf{o}) + \mathrm{H}(X) = \log_2 m = \text{constant}, \tag{10.59}$$

i.e., the sum of doubling rate and entropy rate is constant.

Hence, we see that we can make most money from races with low entropy rate. (On the other hand, if the entropy rate is low, then the bookie won't be so stupid to offer uniform odds...)

10.5 What About Not Gambling?

So far we have always assumed that we use all our money in every race. However, in the situation of subfair odds, where the bookie takes some fixed amount of the gambler's bet for himself for sure, we should allow the gambler to retain some part of his money. Let $b(0) = b_0$ be that part that is not put into the game.

In this situation, the growth factor given in (10.14) will become

$$S_n = \prod_{k=1}^{n} \big(b(0) + b(X_k)\, o(X_k)\big) \tag{10.60}$$

and the Definition 10.6 of the doubling rate is changed to

$$W(\mathbf{b}, \mathbf{p}, \mathbf{o}) \triangleq \mathsf{E}\big[\log_2\big(b(0) + b(X)\, o(X)\big)\big] = \sum_{i=1}^{m} p_i \log_2(b_0 + b_i o_i). \tag{10.61}$$

Let us see how our results adapt to these new definitions:

- **Fair odds:** We have already seen that the bet $b_i = \frac{1}{o_i}$ is risk-free (but also gives no gain). Hence, instead of taking the portion b_0 of our money out of the game, we might also simply distribute b_0 proportionally to $\frac{1}{o_i}$ among all horses. This yields the same effect!

So we see that every betting strategy with $b_0 > 0$ can be replaced by another betting strategy with $b_0 = 0$ and with identical risk and identical performance. Because obviously any betting strategy with $b_0 = 0$ is obviously identical to a betting strategy without b_0 in the first place, the optimal solution that we have derived in Section 10.2 still is optimal, i.e., proportional betting $b_i = p_i$ (and $b_0 = 0$) is log-optimal (but not risk-free).

- **Superfair odds:** Here we would be silly to retain some money as we can increase our money risk-free! So the optimal solution is to have $b_0 = 0$ and always use all money. Hence, using the same argument as above for fair odds, proportional betting is log-optimal (but not risk-free).

- **Subfair odds:** This case is much more complicated. In general it is optimal to retain a certain portion of the money. How much depends on the expected returns $p_i o_i$. Basically, we need to maximize

$$W(\mathbf{b}, \mathbf{p}, \mathbf{o}) = \sum_{i=1}^{m} p_i \log_2(b_0 + b_i o_i) \tag{10.62}$$

under the constraints

$$b_0 \geq 0, \tag{10.63}$$
$$b_i \geq 0, \quad i = 1, \ldots, m, \tag{10.64}$$
$$b_0 + \sum_{i=1}^{m} b_i = 1. \tag{10.65}$$

As shown in the following section, this problem can be solved using the *Karush–Kuhn–Tucker (KKT) conditions* (Theorem 9.11 in Chapter 9).

10.6 Optimal Gambling for Subfair Odds

We apply the KKT conditions to the optimization problem (10.62)–(10.65) and get the following:

$$\frac{\partial W}{\partial b_0} = \sum_{i=1}^{m} \frac{p_i}{b_0 + b_i o_i} \log_2 e \begin{cases} = \lambda & \text{if } b_0 > 0, \\ \leq \lambda & \text{if } b_0 = 0, \end{cases} \tag{10.66}$$

$$\frac{\partial W}{\partial b_\ell} = \frac{p_\ell o_\ell}{b_0 + b_\ell o_\ell} \log_2 e \quad \begin{cases} = \lambda & \text{if } b_\ell > 0, \\ \leq \lambda & \text{if } b_\ell = 0, \end{cases} \quad \ell = 1, \ldots, m, \tag{10.67}$$

where λ must be chosen such that (10.65) is satisfied. We define $\nu \triangleq \frac{\log_2 e}{\lambda}$ and rewrite (10.66) and (10.67) as follows:

$$\sum_{i=1}^{m} \frac{\nu p_i}{b_0 + b_i o_i} \begin{cases} = 1 & \text{if } b_0 > 0, \\ \le 1 & \text{if } b_0 = 0, \end{cases} \tag{10.68}$$

$$\frac{\nu p_i o_i}{b_0 + b_i o_i} \begin{cases} = 1 & \text{if } b_i > 0, \\ \le 1 & \text{if } b_i = 0, \end{cases} \quad i = 1, \ldots, m. \tag{10.69}$$

We now try to solve these expressions for $\mathbf{b}$.

- From (10.69) we learn that, if $b_i > 0$,

$$b_i = \frac{1}{o_i}(\nu p_i o_i - b_0), \tag{10.70}$$

and if $b_i = 0$,

$$b_i \ge \frac{1}{o_i}(\nu p_i o_i - b_0). \tag{10.71}$$

In other words, if $\nu p_i o_i \ge b_0$ we get the case $b_i > 0$; and if $\nu p_i o_i < b_0$, we have $b_i = 0$. Hence,

$$b_i = \frac{1}{o_i}(\nu p_i o_i - b_0)^+ \tag{10.72}$$

where

$$(\cdot)^+ \triangleq \max\{\cdot, 0\}. \tag{10.73}$$

- Next we will prove that (under the assumption of subfair odds) we always have $b_0 > 0$. To this end, assume $b_0 = 0$. Then from (10.72) we see that

$$b_i = \frac{1}{o_i}(\nu p_i o_i - 0)^+ = \nu p_i. \tag{10.74}$$

Using this and our assumption $b_0 = 0$ in (10.68) we get

$$1 \ge \sum_{i=1}^{m} \frac{\nu p_i}{b_0 + b_i o_i} = \sum_{i=1}^{m} \frac{\nu p_i}{0 + \nu p_i o_i} = \sum_{i=1}^{m} \frac{1}{o_i} \overset{\text{sub-fair}}{>} 1, \tag{10.75}$$

where the last inequality follows from the fact that we have assumed subfair odds. Since $1 > 1$ is not possible, we have a contradiction, and therefore we must have $b_0 > 0$.

Hence, we have proven the following fact.

In a horse race with subfair odds, it is never optimal to invest all money.

- So, $b_0 > 0$. Let

$$\mathcal{B} \triangleq \{i \in \{1,\ldots,m\} \colon b_i > 0\} \tag{10.76}$$

be the set of all horses that we bet money on. We now rewrite (10.68) as follows:

$$\begin{aligned}
1 &= \sum_{i=1}^{m} \frac{\nu p_i}{b_0 + b_i o_i} && (10.77)\\
&= \sum_{i\in\mathcal{B}} \frac{\nu p_i}{b_0 + b_i o_i} + \sum_{i\in\mathcal{B}^{\mathrm{c}}} \frac{\nu p_i}{b_0 + b_i o_i} && (10.78)\\
&= \sum_{i\in\mathcal{B}} \frac{\nu p_i}{b_0 + \nu p_i o_i - b_0} + \sum_{i\in\mathcal{B}^{\mathrm{c}}} \frac{\nu p_i}{b_0 + 0} && (10.79)\\
&= \sum_{i\in\mathcal{B}} \frac{1}{o_i} + \frac{\nu}{b_0} \underbrace{\sum_{i\in\mathcal{B}^{\mathrm{c}}} p_i}_{=1-\sum_{i\in\mathcal{B}} p_i}, && (10.80)
\end{aligned}$$

where in first sum of (10.79) we have used (10.72). Hence,

$$b_0 = \nu \cdot \frac{1 - \sum_{i\in\mathcal{B}} p_i}{1 - \sum_{i\in\mathcal{B}} \frac{1}{o_i}}. \tag{10.81}$$

Using (10.72) and (10.81) in (10.65) then yields

$$\begin{aligned}
1 &= b_0 + \sum_{i=1}^{m} b_i && (10.82)\\
&= b_0 + \sum_{i\in\mathcal{B}} \frac{1}{o_i}(\nu p_i o_i - b_0) && (10.83)\\
&= b_0 \left(1 - \sum_{i\in\mathcal{B}} \frac{1}{o_i}\right) + \nu \sum_{i\in\mathcal{B}} p_i && (10.84)\\
&= \nu \cdot \frac{1 - \sum_{i\in\mathcal{B}} p_i}{1 - \sum_{i\in\mathcal{B}} \frac{1}{o_i}} \cdot \left(1 - \sum_{i\in\mathcal{B}} \frac{1}{o_i}\right) + \nu \sum_{i\in\mathcal{B}} p_i && (10.85)\\
&= \nu \left(1 - \sum_{i\in\mathcal{B}} p_i\right) + \nu \sum_{i\in\mathcal{B}} p_i = \nu, && (10.86)
\end{aligned}$$

i.e., $\nu = 1$.

So we finally have arrived at the following solution:

$$b_0 = \frac{1 - \sum_{i\in\mathcal{B}} p_i}{1 - \sum_{i\in\mathcal{B}} \frac{1}{o_i}} \tag{10.87}$$

and

$$b_i = \frac{1}{o_i}\big(p_i o_i - b_0\big)^+ \tag{10.88}$$

for $i \in \{1,\dots,m\}$.

We see that we have a recursive definition here: The values of b_i depend on b_0, but b_0 in turn depends (via $\mathcal{B}$) on b_i. So the remaining question is how to find the optimal set $\mathcal{B}$ of horses we do bet on. We cannot compute this optimal set directly, but we can find a recursive procedure to figure it out.

Firstly, note that from (10.88) we see that the horses with $b_i > 0$ must have an expected return $p_i o_i$ larger than b_0. Hence, we will only (if at all) bet on horses with large expected return! Secondly, note that b_0 is a common threshold to all horses, i.e., if we bet on horse i and horse j has an expected return larger than horse i, $p_j o_j \geq p_i o_i$, then we will also bet on horse j.

So, we start by ordering the horses according to expected return $p_i o_i$. Then we try recursively:

Step 1: Set $\mathcal{B} = \{\}$ and compute b_0 from (10.87).

Step 2: Check whether the best horse (i.e., the one with the largest expected return) has an expected return $p_i o_i$ larger than b_0. If yes, add this horse to $\mathcal{B}$, recompute b_0 and repeat Step 2 with the remaining horses. If no, stop.

This leads to the following algorithm that finds the optimal betting strategy for horse racing with subfair odds.

Theorem 10.11 (Optimal Gambling for Subfair Odds).
Consider an IID horse race $X \sim \mathbf{p}$ with odds $\mathbf{o}$. If the gambler is allowed to withhold the fraction b_0 of his wealth, then an optimal betting strategy that maximizes the doubling rate can be found as follows:

- Order the horses according to the expected return $p_i o_i$:

$$p_1 o_1 \geq p_2 o_2 \geq \cdots \geq p_m o_m. \tag{10.89}$$

- Define

$$\beta_t \triangleq \frac{1-\sum_{j=1}^{t-1} p_j}{1-\sum_{j=1}^{t-1}\frac{1}{o_j}} \qquad \text{for } t=1,\dots,m. \tag{10.90}$$

  Note that $\beta_1 = 1$.

- Find the smallest t such that $\beta_t \geq p_t o_t$. If $\beta_t < p_t o_t$ for all $t = 1,\dots,m$ (which can only happen if the odds are fair or superfair!), then set $t = m+1$ and $\beta_t = 0$.

- Assign

$$b_0 = \beta_t, \tag{10.91a}$$

$$b_i = p_i - \frac{\beta_t}{o_i} \qquad \text{for } i=1,\dots,t-1, \tag{10.91b}$$

$$b_i = 0 \qquad \text{for } i=t,\dots,m. \tag{10.91c}$$

Remark 10.12. Note that it is very well possible that the solution is $b_0 = 1$ and $b_i = 0$ ($i = 1,\dots,m$), i.e., that the optimal betting strategy is *not* to gamble at all! This is the case if the expected return of all horses is less than (or equal to) 1: $p_i o_i \leq 1$, $\forall\, i$. If the expected return of at least one horse is strictly larger than 1, then $b_0 < 1$ and we will gamble.

Also note that if we apply the algorithm to a game that is fair or superfair, we will in general regain our previously found solution $b_0 = 0$ and $b_i = p_i$, $\forall\, i$. The only exceptions are some cases of fair odds where the algorithm will result in $b_0 > 0$ and $b_i = 0$ for some horses. This will happen if the algorithm at a certain stage finds $\beta_t = p_i o_i$ (with equality!). However, in such a case the resulting doubling rate is identical to the doubling rate achieved by proportional betting. △

Example 10.13. We go back to Example 10.3 with $m = 3$ horses with

$$p_1 = 0.7, \qquad p_2 = 0.2, \qquad p_3 = 0.1, \tag{10.92}$$

and

$$o_1 = 1.2, \qquad o_2 = 4.5, \qquad o_3 = 8.5. \tag{10.93}$$

Firstly we order the horses according to expected return

$$p_1 o_1 = 0.84, \qquad p_2 o_2 = 0.9, \qquad p_3 o_3 = 0.85, \tag{10.94}$$

i.e., we have the new order $(2,3,1)$. But then we note that the largest expected return is less than 1, i.e., in the above algorithm the smallest t such that $\beta_t \geq p_t o_t$ is $t = 1$ and we realize that the optimal solution is not to bet at all.

We change the example slightly by offering better odds:

$$o_1 = 1.2, \qquad o_2 = 6, \qquad o_3 = 10. \tag{10.95}$$

the game still is subfair because

$$\sum_{i=1}^{3} \frac{1}{o_i} = 1.1 > 1, \tag{10.96}$$

but now the expected returns are

$$p_2 o_2 = 1.2, \qquad p_3 o_3 = 1, \qquad p_1 o_1 = 0.84. \tag{10.97}$$

The computation of the algorithm is shown in Table 10.1. It turns out that the smallest t such that $\beta_t \geq p_t o_t$ is $t = 3$ and we have (again using the original names of the horses):

$$b_0 = \beta_3 = 0.955, \tag{10.98}$$

$$b_2 = p_2 - \frac{\beta_3}{o_2} = 0.2 - \frac{0.955}{6} = 0.041, \tag{10.99}$$

$$b_3 = p_3 - \frac{\beta_3}{o_3} = 0.1 - \frac{0.955}{10} = 0.00455, \tag{10.100}$$

$$b_1 = 0. \tag{10.101}$$

Hence, we see that we only bet on the two unlikely horses with much higher odds. However, we also see that we gamble with only about 5% of our money.

The doubling rate is now given as

$$W^*(\mathbf{p}, \mathbf{o}) = \sum_{i=1}^{m} p_i \log_2(b_0 + b_i o_i) \tag{10.102}$$

$$= 0.7 \log_2(0.955) + 0.2 \log_2(0.955 + 0.041 \cdot 6) + 0.1 \log_2(0.955 + 0.00455 \cdot 10) \tag{10.103}$$

$$= 0.00563. \tag{10.104}$$

This is very small, but it is positive. I.e., if we play long enough (it might be very long!), we will make money. ♢

10.7 Gambling with Side-Information

Let us return to the situation where we have to bet with all money. So far we have assumed that all horse races are independent of each other and have the same distribution. This is of course not very realistic. In order to loosen this stringent assumption, as a first step, we introduce side-information about the horse race. Later on, we will be able to use this to deal with memory in the

Table 10.1: Derivation of optimal gambling for a subfair game.

original name	2	3	1
new name t	1	2	3
p_i	0.2	0.1	0.7
o_i	6	10	1.2
$p_i o_i$	1.2	1.0	0.84
	$\vee$	$\vee$	$\wedge$
β_t	1	0.96	0.955

system: We simply take the past racing results as the side-information. Note that at the moment we still keep the assumption of independent races.

Assume now that we have available some side-information Y about which horse is going to win: Given is some joint probability distribution $P_{X,Y}$. Depending on the current value of the side-information $Y=y$, we can adapt our betting strategy, i.e., our betting strategy is a function of y: $b(\cdot|y)$. To find the optimal strategy, we need to redo all derivations from above. This basically happens in an identical manner, with the only difference that everything is now conditioned on $Y=y$. Hence, we basically get the same results.

Since we assume that the races are still IID, i.e., (X_k, Y_k) are jointly IID, we have

$$S_n = \prod_{k=1}^{n} b(X_k|Y_k)\, o(X_k) \tag{10.105}$$

and

$$\begin{aligned}
\frac{1}{n}\log_2 S_n &= \frac{1}{n}\log_2\left(\prod_{k=1}^{n} b(X_k|Y_k)\, o(X_k)\right) && (10.106)\\
&= \frac{1}{n}\sum_{k=1}^{n}\log_2\big(b(X_k|Y_k)\, o(X_k)\big) && (10.107)\\
&\overset{n\to\infty}{\to} \mathsf{E}_{X,Y}\big[\log_2\big(b(X|Y)\, o(X)\big)\big] \qquad \text{in probability.} && (10.108)
\end{aligned}$$

We define the conditional doubling rate

$$W(\mathbf{b}(\cdot|\cdot), P_{X,Y}, \mathbf{o}) \triangleq \mathsf{E}_{X,Y}\big[\log_2\big(b(X|Y)\, o(X)\big)\big] \tag{10.109}$$

and obtain

$$\begin{aligned}
&W(\mathbf{b}(\cdot|\cdot), P_{X,Y}, \mathbf{o})\\
&\quad = \sum_{x,y} P_{X,Y}(x,y)\log_2\big(b(x|y)\, o(x)\big) && (10.110)
\end{aligned}$$

$$= \sum_{x,y} P_{X,Y}(x,y) \log_2 b(x|y) + \sum_{x,y} P_{X,Y}(x,y) \log_2 o(x) \tag{10.111}$$

$$= \sum_y P_Y(y) \sum_x P_{X|Y}(x|y) \log_2 \left(b(x|y) \cdot \frac{P_{X|Y}(x|y)}{P_{X|Y}(x|y)} \right) + \sum_x P_X(x) \log_2 o(x) \tag{10.112}$$

$$= \sum_y P_Y(y) \sum_x P_{X|Y}(x|y) \log_2 P_{X|Y}(x|y) - \sum_y P_Y(y) \sum_x P_{X|Y}(x|y) \log_2 \left(\frac{P_{X|Y}(x|y)}{b(x|y)} \right) + \mathsf{E}[\log_2 o(X)] \tag{10.113}$$

$$= -\mathsf{H}(X|Y) - \underbrace{\mathsf{E}_Y\Big[\mathscr{D}\Big(P_{X|Y}(\cdot|Y)\Big\|\mathbf{b}(\cdot|Y)\Big)\Big]}_{\geq 0} + \mathsf{E}[\log_2 o(X)] \tag{10.114}$$

$$\leq -\mathsf{H}(X|Y) + \mathsf{E}[\log_2 o(X)], \tag{10.115}$$

where the inequality can be achieved with equality if, and only if, we choose $b(\cdot|\cdot) = P_{X|Y}(\cdot|\cdot)$.

In summary, we have the following result.

Theorem 10.14 (Betting with Side-Information).
Consider an IID horse race X with odds $\mathbf{o}$ with access to some side-information Y about X, where $(X,Y) \sim P_{X,Y}$. The optimal doubling rate

$$W^*(X|Y) \triangleq \max_{\mathbf{b}(\cdot|\cdot)} W(\mathbf{b}(\cdot|\cdot), P_{X,Y}, \mathbf{o}) \tag{10.116}$$

is given by

$$W^*(X|Y) = \mathsf{E}[\log_2 o(X)] - \mathsf{H}(X|Y) \tag{10.117}$$

(where the entropy must be specified in bits) and is achievable by proportional betting:

$$b^*(x|y) = P_{X|Y}(x|y), \quad \forall\, x, y. \tag{10.118}$$

Here we have introduced a new notation:

$$W^*(X) \triangleq W^*(P_X, \mathbf{o}), \tag{10.119}$$

$$W^*(X|Y) \triangleq W^*(P_{X,Y}, \mathbf{o}). \tag{10.120}$$

So, how much does side-information help? To see this, we compute

$$\Delta W^* \triangleq W^*(X|Y) - W^*(X) \tag{10.121}$$

$$= \mathsf{E}[\log_2 o(X)] - \mathsf{H}(X|Y) - (\mathsf{E}[\log_2 o(X)] - \mathsf{H}(X)) \tag{10.122}$$

$$= \mathrm{H}(X) - \mathrm{H}(X|Y) \tag{10.123}$$
$$= \mathrm{I}(X;Y). \tag{10.124}$$

Hence, we have shown the following beautiful theorem.

Theorem 10.15. The increase ΔW^* in the optimal doubling rate due to the side-information Y for an IID horse race X is

$$\Delta W^* = \mathrm{I}(X;Y). \tag{10.125}$$

10.8 Dependent Horse Races

The most common type of side-information is the past performance of the horses. Hence, we will now drop the IID assumption, but instead assume that $\{X_k\}$ is a stationary stochastic process. Then from Theorem 10.14 we immediately get

$$W^*(X_k|X_{k-1},\ldots,X_1) = \mathrm{E}[\log_2 o(X_k)] - \mathrm{H}(X_k|X_{k-1},\ldots,X_1) \tag{10.126}$$

which is achieved by proportional betting

$$b^*(x_k|x_{k-1},\ldots,x_1) = P(x_k|x_{k-1},\ldots,x_1). \tag{10.127}$$

The question is how to define a corresponding doubling rate in this new context. To answer this, we again need to consider the growth factor of our wealth. Note that the growth factor is now more complicated because our betting strategy changes in each step:

$$S_n^* = \prod_{k=1}^{n} b^*(X_k|X_{k-1},\ldots,X_1)\, o(X_k) \tag{10.128}$$
$$= \prod_{k=1}^{n} P(X_k|X_{k-1},\ldots,X_1)\, o(X_k). \tag{10.129}$$

Of course, as before, S_n^* is a random number. However, while in the case of an IID process we could use the weak law of large numbers to show that for large n, S_n^* will converge to a constant, we cannot do this anymore here, because the weak law of large number does not hold for a general stationary process $\{X_k\}$.

Instead we compute the expected value of $\frac{1}{n}\log_2 S_n^*$:

$$\begin{aligned}
&\mathsf{E}\left[\frac{1}{n}\log_2 S_n^*\right] \\
&\quad= \mathsf{E}\left[\frac{1}{n}\sum_{k=1}^{n}\log_2\big(P(X_k|X_{k-1},\ldots,X_1)\,o(X_k)\big)\right] && (10.130)\\
&\quad= \frac{1}{n}\sum_{k=1}^{n}\mathsf{E}[\log_2 P(X_k|X_{k-1},\ldots,X_1)] + \frac{1}{n}\sum_{k=1}^{n}\mathsf{E}[\log_2 o(X_k)] && (10.131)\\
&\quad= -\frac{1}{n}\sum_{k=1}^{n}\mathsf{H}(X_k|X_{k-1},\ldots,X_1) + \frac{1}{n}\sum_{k=1}^{n}\mathsf{E}[\log_2 o(X_1)] && (10.132)\\
&\quad= -\frac{1}{n}\mathsf{H}(X_1,\ldots,X_n) + \mathsf{E}[\log_2 o(X_1)] && (10.133)\\
&\quad\overset{n\to\infty}{\to} -\mathsf{H}(\{X_k\}) + \mathsf{E}[\log_2 o(X_1)]. && (10.134)
\end{aligned}$$

Here, (10.132) follows from the definition of entropy and from stationarity; (10.133) follows from the chain rule; and the last equality (10.134) is due to the definition of entropy rate.

Hence, we have shown that for $n \gg 1$ we have

$$\mathsf{E}\left[\frac{1}{n}\log_2 S_n^*\right] \approx \mathsf{E}[\log_2 o(X_1)] - \mathsf{H}(\{X_k\}). \tag{10.135}$$

If we are unhappy about this much weaker statement, we can fix the situation by assuming that $\{X_k\}$ is also *ergodic*. Then, the weak law of large numbers can be applied again and we get

$$\begin{aligned}
\frac{1}{n}\log_2 S_n^* &= \frac{1}{n}\sum_{k=1}^{n}\log_2 P(X_k|X_{k-1},\ldots,X_1) + \frac{1}{n}\sum_{k=1}^{n}\log_2 o(X_k) && (10.136)\\
&\overset{n\to\infty}{\to} \lim_{n\to\infty}\mathsf{E}[\log_2 P(X_n|X_{n-1},\ldots,X_1)] \\
&\quad + \lim_{n\to\infty}\mathsf{E}[\log_2 o(X_n)] \qquad \text{in probability} && (10.137)\\
&= -\lim_{n\to\infty}\mathsf{H}(X_n|X_{n-1},\ldots,X_1) + \mathsf{E}[\log_2 o(X_1)] && (10.138)\\
&= -\mathsf{H}(\{X_k\}) + \mathsf{E}[\log_2 o(X_1)]. && (10.139)
\end{aligned}$$

Here, (10.137) follows from ergodicity; in (10.138) we rely on the definition of entropy and use stationarity; and the last equality (10.139) follows from the second equivalent definition of entropy rate for stationary processes.

So we get the following result.

Theorem 10.16. Consider a stationary and ergodic horse race $\{X_k\}$ with odds $\mathbf{o}$. Then we have for large n

$$S_n \approx 2^{nW^*(\{X_k\})} \qquad \text{in probability,} \tag{10.140}$$

where

$$W^*(\{X_k\}) \triangleq \mathsf{E}[\log_2 o(X_1)] - \mathsf{H}(\{X_k\}) \tag{10.141}$$

(where the entropy rate must be specified in bits).

Remark 10.17. As a matter of fact, both IID processes and stationary and ergodic processes satisfy the strong law of large numbers. So in the whole chapter we can replace "in probability" with "almost surely" (or, equivalently, "with probability 1") everywhere. △

Example 10.18. Instead of horses we consider cards. Assume that we have 52 cards, 26 black and 26 red. The cards are well shuffled, and then opened one by one. We bet on the color, with fair odds

$$o(\text{red}) = o(\text{black}) = 2. \tag{10.142}$$

We offer two different strategies:

1. We bet sequentially: We know that it is optimal to bet proportionally, taking the known past into account as side-information. Hence, at the beginning we bet

   $$b(X_1 = \text{red}) = b(X_1 = \text{black}) = \frac{26}{52} = \frac{1}{2}. \tag{10.143}$$

   Then for the second card we bet either, if $X_1 = \text{red}$,

   $$b(X_2 = \text{red}) = \frac{25}{51}, \quad b(X_2 = \text{black}) = \frac{26}{51}, \tag{10.144}$$

   or, if $X_1 = \text{black}$,

   $$b(X_2 = \text{red}) = \frac{26}{51}, \quad b(X_2 = \text{black}) = \frac{25}{51}. \tag{10.145}$$

   etc.

2. We bet on the entire sequence of 52 cards at once. There are $\binom{52}{26}$ different possible sequences of 25 red and 26 black cards, all equally likely. Hence we bet

   $$b(x_1, \ldots, x_{52}) = \frac{1}{\binom{52}{26}} \tag{10.146}$$

   on each of these sequences.

Both strategies are equivalent! To see this note, e.g., that half of the $\binom{52}{26}$ sequences will start with red, i.e., in the second strategy we actually bet half on "red in the first card", which is identical to the first strategy.

How much are we going to win? Well, at the end one sequence must show up, so we get for this sequence 2^{52} (52 times the odds 2) times our money we have bet on this sequence, which is $1/\binom{52}{26}$, and for the rest we get nothing. Hence

$$S_{52}^* = 2^{52} \cdot \frac{1}{\binom{52}{26}} \approx 9.08. \tag{10.147}$$

Note that in this special problem we actually again have a risk-free situation! We will multiply our money by a factor of 9 for sure. But this is not that surprising when you realize that once all 26 cards of one color are opened, we know the outcome of the rest for sure. This will happen latest with the second last card, but usually even earlier. ◊

Chapter 11

Data Transmission over a Noisy Digital Channel

11.1 Problem Setup

We have already mentioned that information theory can be roughly split into three main areas:

- *Source coding* asks the question of how to compress data *efficiently*.
- *Channel coding* ask the question of how to transmit data *reliably* over a channel.
- *Cryptography* asks the question of how to transmit data in a *secure* way such that nobody unwanted can read it.

So far we have had a quite close first look onto the first topic. Now we go to the second one. We consider the general system shown in Figure 11.1.

We will simplify this general system model in several ways: Firstly, we assume that all values of the random message M that is to be transmitted over the channel have the *same equal probability*. This might sound like a strong restriction, but actually it is not. The reason is that the output of any good data compression scheme is (almost) *memoryless and uniformly distributed*!

Remark 11.1 (Compressed data is IID and uniform). Note that the output of an IID uniform source cannot be compressed. The reason is that every block message of length M is equally likely which means that the Huffman code cannot apply shorter messages to more likely messages. This can also be seen directly from the converse of the coding theorem (e.g., Theorem 5.15 or 7.3). A similar argument can be used in the case of a Tunstall code.

Any other source can be compressed. So assume for the moment that the output of a perfect (ideal) compressor has memory and/or is not uniformly

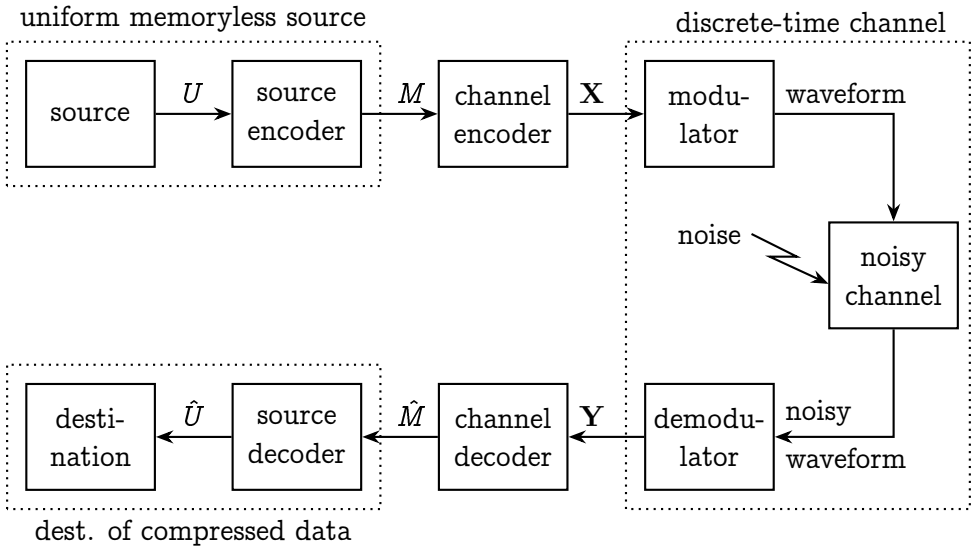


Figure 11.1: Most general system model in information theory.

distributed. In this case we could design another compressor specifically for this output and compress the data even further. But this is a contradiction to the assumption that the first compressor already is perfect! △

Note that in general the random message M emitted by the source might be a sequence of digits (in practice usually a sequence of binary digits). However, we do not really care about the form of the message, but simply assume that there is only a finite number of possible messages and that therefore we can number them from 1 to M.

Definition 11.2. The *source* emits a random message M that is uniformly distributed over the (finite) message set $\mathcal{M} = \{1, 2, \ldots, \mathrm{M}\}$. This means we have $|\mathcal{M}| = \mathrm{M}$ possible messages that are all equally likely.

Secondly, we simplify our system model by ignoring modulation and demodulation. This means that we think of the modulator and the demodulator to be part of the channel and only consider a *discrete-time channel*, i.e., the channel input and output are sequences of RVs $\{X_k\}$ and $\{Y_k\}$, respectively, where k denotes the discrete time.

Even more abstractly and generally, we give the following "engineering definition" of a channel.

Definition 11.3. A *channel* is that part of a communication system that the engineer either is unwilling or unable to change.

Example 11.4. To explain this notion, we give two (quite practical) examples:

- In the design of some communication system, we usually must restrict ourselves to a certain frequency band that is specified by the regulatory administration of the country. Hence, the constraint that we have to use this band becomes part of the channel. Note that there might be other frequency bands that would be better suited to the task, but we are not allowed to use them: We are unable to change that part of our communication system.

- Suppose we want to set up a simple communication system between two locations. Suppose further that we have a certain cheap transmitter available and do not want to buy a new one. Hence, the fact that we use exactly this transmitter becomes part of the channel. There might be much better transmitters available, but we are unwilling to change to those. ◊

So we see that once the channel is given, the remaining freedom can then be used to transmit information.

The only question that remains now is how we can translate this engineering definition into something that is easier for a mathematical analysis. The trick here is to abstract the channel further to its three most important features:

- What can we put into the channel?

- What can come out of the channel?

- What is the (probabilistic) relation between the input and the output?

To simplify our life we will make a further assumption: We assume that the channel is *memoryless*, i.e., the output of the channel only depends on the current input and not on the past inputs. We give the following definition.

Definition 11.5. A *discrete memoryless channel (DMC)* is a channel specified by

- an *input alphabet* $\mathcal{X}$, which is a set of symbols that the channel accepts as input;

- an *output alphabet* $\mathcal{Y}$, which is a set of symbols that the channel can produce at its output; and

- a *conditional probability distribution* $P_{Y|X}(\cdot|x)$ for all $x \in \mathcal{X}$ such that

$$\begin{aligned} &P_{Y_k|X_1,X_2,\text{...},X_k,Y_1,Y_2,\text{...},Y_{k-1}}(y_k|x_1,x_2,\dots,x_k,y_1,y_2,\dots,y_{k-1}) \\ &\quad = P_{Y|X}(y_k|x_k), \qquad \forall\, k. \end{aligned} \tag{11.1}$$

  This conditional distribution $P_{Y|X}(\cdot|\cdot)$ is sometimes also called *channel law*. It describes the probabilistic connection between input and output.

So mathematically a DMC is simply a conditional probability distribution that tells us the distribution of the output for every possible different input letter.

We would like to point out the following important observations from the definition of a DMC:

- The current output Y_k depends only on the current input x_k (the channel is *memoryless*).
- For a particular input x_k, the distribution of the output Y_k does not change over time (the channel is *time-invariant*). This can be seen from our notation: We wrote $P_{Y|X}(y_k|x_k)$ and not $P_{Y_k|X_k}(y_k|x_k)$!
- For a given input x, we do not know the output for sure (due to the random noise), but we are only given a certain probability distribution. Hopefully, this distribution is not the same for different inputs, as otherwise we will never be able to recover the input from the distorted output!

Example 11.6. A very typical example of a DMC is the *binary symmetric channel (BSC)* shown in Figure 11.2. It is a binary channel, i.e., both input

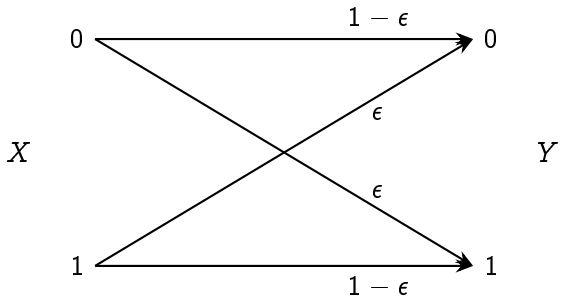


Figure 11.2: Binary symmetric channel (BSC).

and output can only take on two different values, i.e., $\mathcal{X} = \mathcal{Y} = \{0, 1\}$. We see that if $x = 0$ the probability of the output Y being unchanged is $1 - \epsilon$. The probability of a flip is ϵ. The channel is called *symmetric* because the probability of a flip is the same for $x = 0$ to $Y = 1$ and $x = 1$ to $Y = 0$. We see that

$$P_{Y|X}(0|0) = 1 - \epsilon, \qquad P_{Y|X}(1|0) = \epsilon, \tag{11.2}$$

$$P_{Y|X}(0|1) = \epsilon, \qquad P_{Y|X}(1|1) = 1 - \epsilon. \tag{11.3}$$

Another often used channel is the *binary erasure channel (BEC)* shown in Figure 11.3. This channel has the same binary input as the BSC, but the output alphabet is ternary: If we receive "?", then we know that something

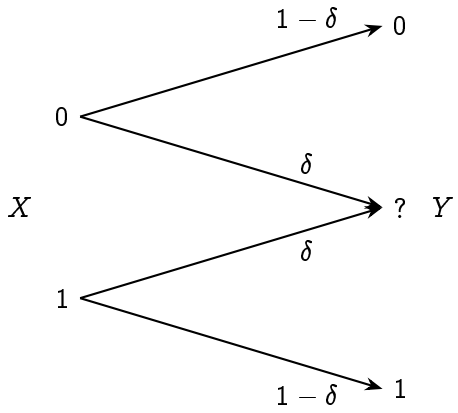


Figure 11.3: Binary erasure channel (BEC).

went wrong and the input symbol was lost. The probability for such a lost symbol is δ.

Note that this seems to be a better behavior than for the BSC, where we never can be sure if a 1 really is a 1. In the BEC we never are going to confuse $Y = 0$ with $Y = 1$. ◊

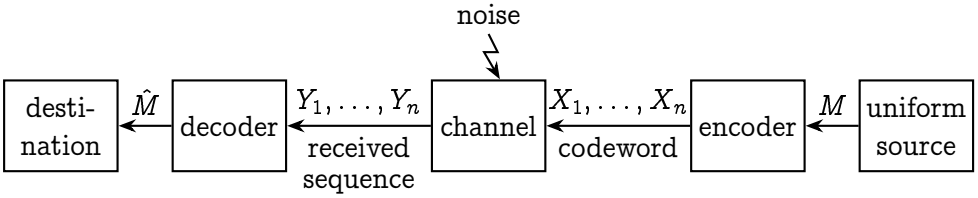


Figure 11.4: Simplified discrete-time system model.

Adding these two simplifications to our system model in Figure 11.1, we now get the simplified, abstract system model shown in Figure 11.4. The only two devices in this model that we have not yet described are the *encoder* and the *decoder*.

Definition 11.7. The *channel encoder* is a device that accepts a message m as input and generates from it a sequence of channel input symbols at the output. Usually, of course, we will have a different output sequence for every different message m.

We will assume that all output sequences have the same length n. These output sequences are denoted *codewords* of *blocklength* n.

Mathematically, the encoder is a *deterministic* mapping

$$\phi\colon \mathcal{M} \to \mathcal{X}^n \tag{11.4}$$

that assigns for every message m a codeword $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{X}^n$. The set of all codewords $\mathscr{C}$ is called *codebook* or simply *code*.

Definition 11.8. The *channel decoder* is a device that receives a length-n channel output sequence (i.e., a distorted codeword) and then needs to make a decision on what message has been sent.

Of course we should design this device in a clever way so as to optimize some cost function. Usually, the decoder is designed such that the probability of a message error is minimized. For more details see Section 11.3.

Note that this optimization is in general very difficult, but it can be done during the design phase of the system. Once we have figured out the optimal way of decoding, we do not change the decoder anymore. I.e., mathematically, the decoder is a *deterministic* mapping

$$\psi \colon \mathcal{Y}^n \to \hat{\mathcal{M}}. \tag{11.5}$$

Here usually $\hat{\mathcal{M}} = \mathcal{M}$, but sometimes we might have $\hat{\mathcal{M}} = \mathcal{M} \cup \{0\}$ where 0 corresponds to a declaration of an error.

In short: A decoder takes any possible channel output sequence $\mathbf{y} = (y_1, \dots, y_n)$ and assigns to it a *guess* $\hat{m}$ which message has been transmitted.

11.2 Discrete Memoryless Channels

Next, we are going to investigate the DMC more in detail and make one more important assumption.

Definition 11.9. We say that a DMC is used *without feedback* if

$$P(x_k|x_1, \dots, x_{k-1}, y_1, \dots, y_{k-1}) = P(x_k|x_1, \dots, x_{k-1}), \quad \forall\, k, \tag{11.6}$$

i.e., X_k depends only on past inputs (by choice of the encoder), but not on past outputs. In other words, there is no feedback link from the receiver back to the transmitter that would inform the transmitter about the past outputs.

Remark 11.10. It is important to realize that even though we assume the channel to be memoryless, we do *not* restrict the encoder to be memoryless! Actually, memory in the encoder will be crucial for our transmission system! As a very simple example consider a codebook with only two codewords of length $n = 3$:

$$\mathscr{C} = \{000, 111\}. \tag{11.7}$$

(This code is called *three-times repetition code*.) In this case if $x_1 = 1$, we know for sure that $x_2 = x_3 = 1$, i.e., we have very strong memory between the different inputs x_1^n! △

In this class we will most of the time (i.e., always, unless we explicitly state it) assume that we do not have any feedback links.

We now prove the following theorem.

Theorem 11.11. If a DMC is used without feedback, then

$$P(y_1,\dots,y_n|x_1,\dots,x_n)=\prod_{k=1}^{n} P_{Y|X}(y_k|x_k), \quad \forall\, n\geq 1. \tag{11.8}$$

Proof: The proof only needs the definition of a DMC (11.1) and the assumption that we do not have any feedback (11.6):

$$\begin{aligned}
&P(x_1,\dots,x_n,y_1,\dots,y_n)\\
&\quad = P(x_1)\cdot P(y_1|x_1)\cdot P(x_2|x_1,y_1)\cdot P(y_2|x_1,x_2,y_1)\cdots\\
&\qquad \cdot P(x_n|x_1,\dots,x_{n-1},y_1,\dots,y_{n-1})\cdot P(y_n|x_1,\dots,x_n,y_1,\dots,y_{n-1}) && (11.9)\\
&\quad = P(x_1)\cdot P(y_1|x_1)\cdot P(x_2|x_1)\cdot P(y_2|x_1,x_2,y_1)\cdots\\
&\qquad \cdot P(x_n|x_1,\dots,x_{n-1})\cdot P(y_n|x_1,\dots,x_n,y_1,\dots,y_{n-1}) && (11.10)\\
&\quad = P(x_1)\cdot P_{Y|X}(y_1|x_1)\cdot P(x_2|x_1)\cdot P_{Y|X}(y_2|x_2)\cdots\\
&\qquad \cdot P(x_n|x_1,\dots,x_{n-1})\cdot P_{Y|X}(y_n|x_n) && (11.11)\\
&\quad = P(x_1)\cdot P(x_2|x_1)\cdots P(x_n|x_1,\dots,x_{n-1})\cdot \prod_{k=1}^{n} P_{Y|X}(y_k|x_k) && (11.12)\\
&\quad = P(x_1,\dots,x_n)\cdot \prod_{k=1}^{n} P_{Y|X}(y_k|x_k), && (11.13)
\end{aligned}$$

where the first equality (11.9) follows from the chain rule, the second (11.10) because we do not have feedback, and the third (11.11) from the definition of a DMC.

Hence

$$\frac{P(x_1,\dots,x_n,y_1,\dots,y_n)}{P(x_1,\dots,x_n)}=P(y_1,\dots,y_n|x_1,\dots,x_n)=\prod_{k=1}^{n} P_{Y|X}(y_k|x_k). \tag{11.14}$$

□

While in source coding we have seen that the entropy rate $\mathrm{H}(\{U_k\})$ was the crucial definition, it probably will not come as a big surprise that in channel coding this role is taken over by the mutual information $\mathrm{I}(X;Y)$, i.e., the amount of information that Y reveals about X (or vice versa). So next we will investigate the mutual information between the channel input and output of a DMC.

Lemma 11.12. The mutual information $\mathrm{I}(X;Y)$ between the input X and output Y of a DMC that is used without feedback is a function of the channel law $P_{Y|X}(\cdot|\cdot)$ and the PMF of the input $P_X(\cdot)$:

$$\mathrm{I}(X;Y) = f(P_X, P_{Y|X}). \tag{11.15}$$

To be precise we have

$$\mathrm{I}(X;Y) = \mathsf{E}_X\Big[\mathscr{D}(P_{Y|X}(\cdot|X) \,\|\, (P_X P_{Y|X})(\cdot))\Big], \tag{11.16}$$

where we have defined

$$(P_X P_{Y|X})(y) \triangleq \sum_{x'} P_X(x') P_{Y|X}(y|x'), \quad \forall\, y, \tag{11.17}$$

and were we make use of relative entropy (Definition 3.1).

Proof: We remember that mutual information can be written using the relative entropy $\mathscr{D}(\cdot\,\|\,\cdot)$ (see the discussion around (3.14)):

$$\begin{aligned}
\mathrm{I}(X;Y) &= \mathscr{D}(P_{X,Y} \,\|\, P_X \cdot P_Y) && (11.18)\\
&= \sum_{(x,y)} P_{X,Y}(x,y) \log \frac{P_{X,Y}(x,y)}{P_X(x) P_Y(y)} && (11.19)\\
&= \sum_x P_X(x) \sum_y P_{Y|X}(y|x) \log \frac{P_{Y|X}(y|x)}{P_Y(y)} && (11.20)\\
&= \sum_x P_X(x) \sum_y P_{Y|X}(y|x) \log \frac{P_{Y|X}(y|x)}{\sum_{x'} P_X(x') P_{Y|X}(y|x')} && (11.21)\\
&= \mathsf{E}_X\Big[\mathscr{D}(P_{Y|X}(\cdot|X) \,\|\, (P_X P_{Y|X})(\cdot))\Big]. && (11.22)
\end{aligned}$$

□

Hence, we learn that the mutual information is a function of the channel law $P_{Y|X}(\cdot|\cdot)$, which is given to us via the channel, and of the distribution on the input, which can be chosen freely by us via the design of the encoder.

Based on the fact that we can design the encoder, but have to accept the channel that we get, the following definition is now quite natural.

Definition 11.13. We define the *information capacity* of a given DMC as

$$\mathrm{C}_{\inf} \triangleq \max_{P_X(\cdot)} \mathrm{I}(X;Y). \tag{11.23}$$

Note that this is a purely mathematical definition without any engineering meaning so far!

11.3 Coding for a DMC

So we have introduced our system model (see Figure 11.4) and have understood some issues about the channel. We are now ready to think about the design of an encoder and decoder for a given DMC. Recalling our mathematical definitions of the encoder and decoder in Definitions 11.7 and 11.8, we see that such a design basically is equivalent to specifying a codebook $\mathscr{C}$, an encoding function $\phi(\cdot)$ and a decoding function $\psi(\cdot)$. Hence, we give the following definition.

Definition 11.14. An (M, n) *coding scheme* for a DMC $(\mathcal{X}, \mathcal{Y}, P_{Y|X})$ consists of

- the message set $\mathcal{M} = \{1, \ldots, \mathrm{M}\}$,
- the codebook (or code) $\mathscr{C}$ of all length-n codewords,
- an encoding function $\phi(\cdot)$, and
- a decoding function $\psi(\cdot)$.

Remark 11.15. We make a short comment about our choice of name. It is an unfortunate habit of many information theorists (including [CT06]) to call the (M, n) coding scheme an "(M, n) code". The problem is that the term *code* is used already by all coding theorists to denote the codebook, i.e., the unsorted list of codewords. We only mention the *Hamming code* and the *Reed-Solomon code* as examples. To avoid confusion between the two groups of people, we will always try to use *codebook* for the list of codewords and *coding scheme* for the complete system consisting of the codebook and the mappings. △

As one sees from its name "(M, n) coding scheme", the two crucial parameters of interest of a coding scheme are the number of possible messages M and the codeword length (blocklength) n. Usually we would like to make M large, because this way we can transmit more information over the channel, and the blocklength n should be short so that it does not take too long (we do not need too many discrete time-steps) to transmit the codeword. In other words,

- we have M equally likely messages, i.e., the entropy of the message is $\mathrm{H}(M) = \log_2 \mathrm{M}$ bits;
- we need n transmissions of a channel input symbol X_k over the channel in order to transmit the complete message.

Hence, we transmit on average $\frac{\mathrm{H}(M)}{n} = \frac{\log_2 \mathrm{M}}{n}$ bits per each transmission of a channel symbol.

This insight is crucial and leads to the following fundamental definition.

Definition 11.16. The *rate*[1] R of an (M, n) coding scheme is defined as

$$\mathrm{R} \triangleq \frac{\log_2 M}{n} \text{ bits/transmission.} \tag{11.24}$$

The rate is one of the most important parameter of any coding scheme. It tells us how much information is transmitted every time the channel is used given the particular coding scheme. The unit "bits/transmission" sometimes is also called "bits/channel use" or "bits/symbol".

So far we have only worried about the encoder. However, at least as important is the decoder. If the decoder is not able to gain back the transmitted message, the coding scheme is not very useful! Or to say it in other words: The definition of the transmission rate R only makes sense if the message really arrives at the destination, i.e., if the receiver does not make too many decoding errors.

So the second crucial parameter of a coding scheme is its performance measured in its probability of making an error. To explain more about this, we give the following definitions.

Definition 11.17. Given that message $M = m$ has been sent, we define λ_m to be the *error probability* (or *block error probability*):

$$\lambda_m \triangleq \Pr\big[\psi(\mathbf{Y}) \neq m \,\big|\, \mathbf{x} = \phi(m)\big] \tag{11.25}$$

$$= \sum_{\mathbf{y}\in\mathcal{Y}^n} P(\mathbf{y}|\mathbf{x}(m))\, \mathbb{1}\{\psi(\mathbf{y}) \neq m\}, \tag{11.26}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function

$$\mathbb{1}\{\text{statement}\} \triangleq \begin{cases} 1 & \text{if statement is true,} \\ 0 & \text{if statement is wrong.} \end{cases} \tag{11.27}$$

The *maximum error probability* $\lambda^{(n)}$ for an (M, n) coding scheme is defined as

$$\lambda^{(n)} \triangleq \max_{m\in\mathcal{M}} \lambda_m. \tag{11.28}$$

The *average error probability* $P_{\mathrm{e}}^{(n)}$ for an (M, n) coding scheme is defined as

$$P_{\mathrm{e}}^{(n)} \triangleq \frac{1}{M} \sum_{m=1}^{M} \lambda_m. \tag{11.29}$$

Now we know all main definitions to formulate our main objective.

[1] We define the rate here using a logarithm of base 2. However, we can use any logarithm as long as we adapt the units accordingly.

The main objective of channel coding is to try to find an (M, n) coding scheme such that the maximum error probability $\lambda^{(n)}$ is as small as possible and the rate R is as large as possible.

Note that in order to achieve our main objective, we cannot only vary the design of the codewords in the codebook $\mathscr{C}$, but also the blocklength n, the number of codewords M, and (coupled with these two quantities) also the rate $\mathrm{R} = \frac{\log \mathrm{M}}{n}$.

Now it is quite obvious that if we fix M and let n grow that the error probability can be made arbitrarily small. Take the example of a binary repetition code of length n (compare with Remark 11.10) and only two possible messages $\mathrm{M} = 2$: the larger n is, the more certain the receiver becomes about the transmitted message. However, such a system is very inefficient because we end up transmitting only 1 single bit, but use n transmissions. The rate of this scheme tends to zero!

Much more interesting is a system that actually transmits at a constant rate R. This then means that for growing n, also M must grow. Usually, we formulate this in a way that we choose a blocklength n and a fixed rate R. The number of needed codewords is then

$$\mathrm{M} = 2^{n\mathrm{R}}. \tag{11.30}$$

Here we have assumed that the rate R is given in bits/channel use, i.e., the logarithm is to the base 2. If this is not the case, we will have to change the form to, e.g., $e^{n\mathrm{R}}$ for the case of a rate measured in nats.[2]

Next we ask the question how the decoder needs to be designed such that the error probability is minimized. Given that the random message $M = m$ has been fed to the encoder, the decoder receives the sequence $\mathbf{Y} = (Y_1, \ldots, Y_n)$, which has a conditional distribution

$$P(\mathbf{Y}|\mathbf{X}(m)) = \prod_{k=1}^{n} P_{Y|X}\big(Y_k\big|X_k(m)\big). \tag{11.31}$$

The receiver needs to guess which message has been sent, and we want to do this in a way that minimizes the probability of decoding error or, equivalently, maximizes the probability of a correct decoding:

$$\begin{aligned}\Pr(\text{correct decoding}) &= \sum_{\mathbf{y}} P_{\mathbf{Y}}(\mathbf{y}) \Pr(\text{correct decoding}|\,\mathbf{Y}=\mathbf{y}) && (11.32)\\ &= \sum_{\mathbf{y}} P_{\mathbf{Y}}(\mathbf{y}) \Pr[M = \psi(\mathbf{y})|\,\mathbf{Y}=\mathbf{y}]. && (11.33)\end{aligned}$$

[2] Note that we actually also need to include a ceiling-operation in (11.30) to make sure that M is an integer.

This is maximized if for each $\mathbf{y}$ we choose $\psi(\mathbf{y})$ such as to maximize[3] the conditional probability $\Pr[M=\psi(\mathbf{y})|\,\mathbf{Y}=\mathbf{y}]$:

$$\psi(\mathbf{y}) \triangleq \operatorname*{argmax}_{m\in\mathcal{M}} P_{M|\mathbf{Y}}(m|\mathbf{y}). \tag{11.34}$$

Such a decoder is called *maximum a posteriori (MAP) decoder*. It is the best decoder. Its form can be further changed as follows:

$$\begin{aligned}
\psi(\mathbf{y}) &\triangleq \operatorname*{argmax}_{m\in\mathcal{M}} P_{M|\mathbf{Y}}(m|\mathbf{y}) && (11.35)\\
&= \operatorname*{argmax}_{m\in\mathcal{M}} \frac{P_{M,\mathbf{Y}}(m,\mathbf{y})}{P_{\mathbf{Y}}(\mathbf{y})} && (11.36)\\
&= \operatorname*{argmax}_{m\in\mathcal{M}} \frac{P_M(m)P_{\mathbf{Y}|M}(\mathbf{y}|m)}{P_{\mathbf{Y}}(\mathbf{y})} && (11.37)\\
&= \operatorname*{argmax}_{m\in\mathcal{M}} \{P_M(m)P_{\mathbf{Y}|M}(\mathbf{y}|m)\} && (11.38)\\
&= \operatorname*{argmax}_{m\in\mathcal{M}} \{P_M(m)P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(m))\}, && (11.39)
\end{aligned}$$

where (11.38) follows because $P_{\mathbf{Y}}(\mathbf{y})$ is not a function of m. If we now assume (as we usually do) that M is uniform, i.e., $P_M(m)=1/2^{n\mathrm{R}}$, we get

$$\begin{aligned}
\psi(\mathbf{y}) &= \operatorname*{argmax}_{m\in\mathcal{M}} \left\{\frac{1}{2^{n\mathrm{R}}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(m))\right\} && (11.40)\\
&= \operatorname*{argmax}_{m\in\mathcal{M}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(m)), && (11.41)
\end{aligned}$$

where (11.41) follows because $1/2^{n\mathrm{R}}$ is not a function of m. A decoder that implements the rule given in (11.41) is called *maximum likelihood (ML) decoder*.

An ML decoder is equivalent to the optimum MAP decoder if (and only if!) the messages are uniformly distributed. However, even if this were not the case, people often prefer the suboptimal ML decoder over the optimal MAP decoder, because for the MAP decoder it is necessary to know the source distribution $P_M(\cdot)$, which usually is not the case. Moreover, if the source changes, the decoder needs to be adapted as well, which is very inconvenient for a practical system design: We definitely do not want to change our radio receiver simply because the type of transmitted music is different on Sundays than during the week...!

An ML decoder, on the other hand, does not depend on the source, and since a uniform source is kind of the "most difficult" case anyway, the ML decoder is like a worst-case design: The decoder is optimized for the most difficult source.

[3] Note that strictly speaking argmax_m is not defined if the maximum-achieving value m is not unique. However, in that case, the performance of the decoder does not change irrespective of which m among all maximum-achieving values is chosen. It is usual to define that argmax_m will pick a value m among all optimal values uniformly at random.

Irrespective of the type of decoder used, it is often convenient to describe it using *decoding regions*.

Definition 11.18. The mth *decoding region* is defined as

$$\mathcal{D}_m \triangleq \{\mathbf{y}\colon \psi(\mathbf{y}) = m\}, \tag{11.42}$$

i.e., it is the set of all those channel output sequences that will be decoded to the message m.

Even though it is not necessarily correct, it sometimes is helpful for the understanding to think of the channel as a device that adds noise to the transmitted codeword. The decoding regions can then be thought of as clouds around the codewords. If the noise is not too strong, then $\mathbf{Y}$ will be relatively "close" to the transmitted codeword $\mathbf{x}(m)$, so a good decoder should look for the "closest" codeword to a received $\mathbf{Y}$. See Figure 11.5 for an illustration of this idea.

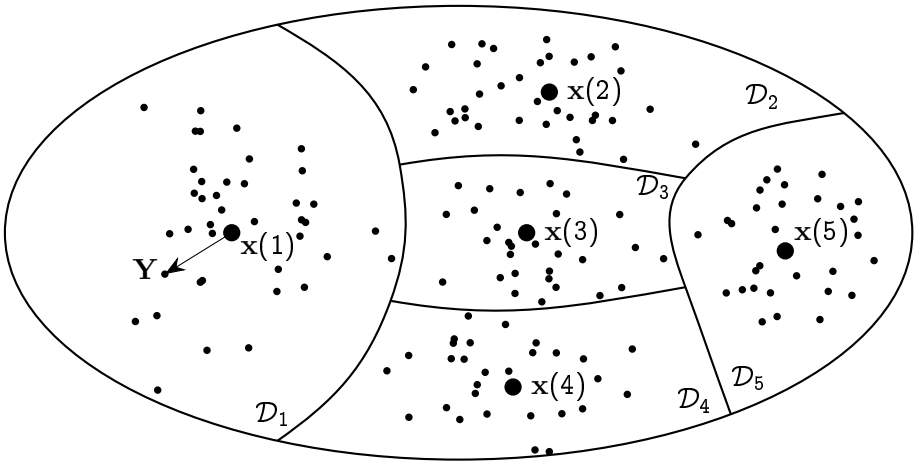


Figure 11.5: Decoding regions of a code with five codewords.

With the help of the decoding region, we can now rewrite the error probability λ_m as follows:

$$\lambda_m = \sum_{\mathbf{y}\notin\mathcal{D}_m} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(m)). \tag{11.43}$$

Unfortunately, in spite of its simplicity, the evaluation of this expression for an ML decoder is rather difficult in general. So it can be useful to have a bound, particularly an upper bound, at hand.

11.4 Bhattacharyya Bound

For the situation of a code with only two codewords, there exists a very elegant bound on the error probability λ_m under an ML decoder. According to its discoverer, this bound has the difficult name *Bhattacharyya Bound*.

In order to derive it, we assume that we have a code with only two codewords: $\mathbf{x}(1)$ and $\mathbf{x}(2)$. There are only two decoding regions that must satisfy $\mathcal{D}_1^{\mathrm{c}} = \mathcal{D}_2$. So, (11.43) simplifies significantly:

$$\lambda_2 = \sum_{\mathbf{y}\in\mathcal{D}_1} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2)) \tag{11.44}$$

and similar for λ_1.

Now recall that the ML rule looks for the codeword that maximize the conditional probability:

$$\begin{aligned}
\mathbf{y}\in\mathcal{D}_1 &\implies P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1)) \ge P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2)) && (11.45)\\
&\implies \sqrt{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1))} \ge \sqrt{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2))} && (11.46)\\
&\implies \sqrt{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1))\,P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2))} \ge P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2)). && (11.47)
\end{aligned}$$

(Note that the inverse direction might not hold because we have not specified what happens in a case of a tie when $P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1)) = P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2))$.) Using (11.47) in (11.44) then yields

$$\lambda_2 = \sum_{\mathbf{y}\in\mathcal{D}_1} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2)) \le \sum_{\mathbf{y}\in\mathcal{D}_1} \sqrt{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1))\,P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2))}. \tag{11.48}$$

A similar derivation will lead to

$$\lambda_1 = \sum_{\mathbf{y}\in\mathcal{D}_2} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1)) \le \sum_{\mathbf{y}\in\mathcal{D}_2} \sqrt{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1))\,P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2))}, \tag{11.49}$$

and adding (11.48) and (11.49) we obtain

$$\lambda_1+\lambda_2 \le \sum_{\mathbf{y}} \sqrt{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(1))\,P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}(2))}. \tag{11.50}$$

If we now assume that the code is used on a DMC without feedback, we can further simplify this as follows:

$$\begin{aligned}
\lambda_m &\le \lambda_1+\lambda_2 && (11.51)\\
&\le \sum_{y_1,\ldots,y_n} \sqrt{\prod_{k=1}^{n} P_{Y|X}(y_k|x_k(1))\,P_{Y|X}(y_k|x_k(2))} && (11.52)\\
&= \prod_{k=1}^{n} \sum_{y} \sqrt{P_{Y|X}(y|x_k(1))\,P_{Y|X}(y|x_k(2))}. && (11.53)
\end{aligned}$$

Here in the last step we did the following type of algebraic transformation:

$$a_1b_1+a_1b_2+a_2b_1+a_2b_2 = (a_1+a_2)(b_1+b_2). \tag{11.54}$$

We have shown the following bound.

Theorem 11.19 (Bhattacharyya Bound).
The worst case block error probability of a code with two codewords that is used over a DMC without feedback and that is decoded with ML decoding is upper-bounded as follows:

$$\lambda^{(n)} \le \prod_{k=1}^{n} \sum_{y} \sqrt{P_{Y|X}(y|x_k(1))\, P_{Y|X}(y|x_k(2))}. \tag{11.55}$$

In practice, one often encounters channels with a binary input alphabet. For such channels, Theorem 11.19 can be expressed even more simply.

Definition 11.20. For a DMC with a binary input alphabet $\mathcal{X} = \{0, 1\}$, we define the *Bhattacharyya distance* $\mathrm{D_B}$ as

$$\mathrm{D_B} \triangleq -\log_2\left(\sum_{y} \sqrt{P_{Y|X}(y|0)\, P_{Y|X}(y|1)}\right). \tag{11.56}$$

For a code used on a binary-input DMC, we then have for $x_k(1) \neq x_k(2)$,

$$\sum_{y} \sqrt{P_{Y|X}(y|x_k(1))\, P_{Y|X}(y|x_k(2))} = \sum_{y} \sqrt{P_{Y|X}(y|0)\, P_{Y|X}(y|1)} \tag{11.57}$$

$$= 2^{-\mathrm{D_B}}, \tag{11.58}$$

and for $x_k(1) = x_k(2)$,

$$\sum_{y} \sqrt{P_{Y|X}(y|x_k(1))\, P_{Y|X}(y|x_k(2))} = \sum_{y} P_{Y|X}(y|x_k(1)) = 1. \tag{11.59}$$

Hence, we see that for binary-input DMCs, Theorem 11.19 can be rewritten as follows.

Corollary 11.21. For a binary-input DMC and the code $\{\mathbf{x}(1), \mathbf{x}(2)\}$ the worst case block error probability (using an ML decoder) satisfies

$$\lambda^{(n)} \le \left(2^{-\mathrm{D_B}}\right)^{d_{\mathrm{H}}(\mathbf{x}(1),\mathbf{x}(2))}. \tag{11.60}$$

Example 11.22. Consider the BEC shown in Figure 11.6. We have

$$\mathrm{D_B} = -\log_2\left(\sqrt{(1-\delta)\cdot 0} + \sqrt{\delta\cdot\delta} + \sqrt{0\cdot(1-\delta)}\right) = -\log_2 \delta \tag{11.61}$$

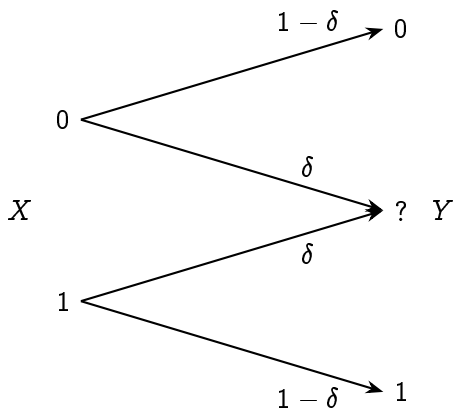


Figure 11.6: Binary erasure channel (BEC).

and therefore

$$\lambda^{(n)} \le 2^{d_{\mathrm{H}}(\mathbf{x}(1),\mathbf{x}(2))\log_2\delta} = \delta^{d_{\mathrm{H}}(\mathbf{x}(1),\mathbf{x}(2))}. \tag{11.62}$$

◊

11.5 Operational Capacity

We have seen in Section 11.3 that it is our goal to make the maximum probability of error as small as possible. It should be quite obvious though that unless we have a very special (and very unrealistic) channel it will never be possible to make the error probability equal to zero. It also turns out that the analysis of the optimal system is horribly difficult and even not tractable by simulation or numerical search.

Shannon's perhaps most important contribution was a way around this problem. He realized that in general a system should perform better if we make n larger. This is quite obvious if we keep the number of codewords M constant. In this case we can make the distance between the different codewords larger and larger, and thereby the probability that the decoder will mix them up due to the noise will become smaller and smaller. However, if we keep M constant while making n larger, we implicitly let the rate R tend to zero (see (11.24) or (11.30)). Hence, ultimately, for very large n we transmit almost no information anymore.

At the time of Shannon every engineer was convinced that the only way of improving the error probability was to reduce the rate. Shannon, on the other hand, had the following incredible insight:

It is possible to reduce the error probability *without reducing the rate*!

If you think this is trivially obvious, you probably have not understood the implications of this statement...

Using this insight, Shannon then found a way around the obstacle that it is basically impossible to analyze or even find an optimal coding scheme for a given DMC. His trick was not to try to analyze a system for a fixed rate and blocklength, but to make the blocklength n arbitrarily large and at the same time ask the error probability to become arbitrarily small. He gave the following two crucial definitions.

Definition 11.23. A rate R is said to be *achievable* on a given DMC if there exists a sequence of $\big(\lceil 2^{nR}\rceil, n\big)$ coding schemes such that the maximum error probability $\lambda^{(n)}$ tends to 0 as $n \to \infty$.

Definition 11.24. The *operational capacity* C_{op} of a DMC is defined to be the supremum of all achievable rates.

Hence we see that the capacity is the highest amount of information (in bits) that can be transmitted over the channel per transmission *reliably* in the sense that the error probability can be made arbitrarily small if we make n sufficiently large.

Note that Shannon on purpose ignored two issues that actually are important in practice. Firstly, he completely ignores delay in the system. If we make n very large, this means that at a first step, the encoder needs to wait until he gets the complete random message M which usually is encoded as a binary string, i.e., if the number of messages M is very large (which will be the case if n is large!) we have to wait for a long binary sequence. Next the codeword needs to be transmitted which takes n transmissions. The receiver can only decode once it has received the full message $(Y_1, \ldots, Y_n)$. So we see that large n causes large delays in the system.[4]

Secondly, Shannon does not worry about the practicability of a scheme, i.e., he does not bother about the computational complexity of the system.[5]

Before we start proving Shannon's main results, we would like to recapitulate some of the important insights we have seen so far:

- If we fix n, but increase the number of messages M (i.e., we increase the rate R), we need to choose more codewords in the same n-dimensional vector space. Thereby we increase the chance that the decoder will mix them up. Hence we enlarge the error probability.

[4] The delays you usually experience when making an international telephone call that is routed via satellites are not mainly caused by the distance, but by the used channel coding with large blocklength!

[5] To see the problem, just imagine how an ML decoder will look like for codewords of blocklength $n = 100'000\ldots$

- However, if we increase both the number of messages M and n (such that the rate R remains constant), then it might be possible that we increase the space between the different codewords in the n-dimensional vector space (even though the number of codewords is increasing!). If this is the case, then we actually reduce the chance of a mix up and thereby the error probability.

- Whatever we do, there is no way of having a system with exactly zero error probability.

11.6 Two Important Lemmas

Before we can figure out how to compute the operational capacity, we need two important lemmas. The first one relates error probability with entropy.

Lemma 11.25 (Fano Inequality). Let U and $\hat{U}$ be L-ary ($\mathrm{L} \geq 2$) RVs taking value in the same alphabet.[6] Let the error probability be defined as

$$P_{\mathrm{e}} \triangleq \Pr[U \neq \hat{U}]. \tag{11.63}$$

Then

$$\mathrm{H_b}(P_{\mathrm{e}}) + P_{\mathrm{e}} \log(\mathrm{L}-1) \geq \mathrm{H}(U|\hat{U}). \tag{11.64}$$

Proof: Define the indicator RV (it indicates an error!)

$$Z \triangleq \begin{cases} 1 & \text{if } \hat{U} \neq U, \\ 0 & \text{if } \hat{U} = U. \end{cases} \tag{11.65}$$

Then

$$P_Z(1) = P_{\mathrm{e}}, \tag{11.66}$$
$$P_Z(0) = 1 - P_{\mathrm{e}}, \tag{11.67}$$
$$\mathrm{H}(Z) = \mathrm{H_b}(P_{\mathrm{e}}). \tag{11.68}$$

Now we use the chain rule to derive the following:

$$\mathrm{H}(U, Z|\hat{U}) = \mathrm{H}(U|\hat{U}) + \underbrace{\mathrm{H}(Z|U, \hat{U})}_{=0} = \mathrm{H}(U|\hat{U}) \tag{11.69}$$

[6] You may think of $\hat{U}$ to be a *decision* or *guess* about U.

and

$$\begin{aligned}
\mathrm{H}(U,Z|\hat{U}) &= \mathrm{H}(Z|\hat{U}) + \mathrm{H}(U|\hat{U},Z) && (11.70)\\
&\le \mathrm{H}(Z) + \mathrm{H}(U|\hat{U},Z) && (11.71)\\
&= \mathrm{H}_{\mathrm{b}}(P_{\mathrm{e}}) + P_Z(0)\underbrace{\mathrm{H}(U|\hat{U},Z=0)}_{\substack{=0\\ \text{because } U=\hat{U}}} + P_Z(1)\underbrace{\mathrm{H}(U|\hat{U},Z=1)}_{\substack{\le \log(\mathrm{L}-1)\\ \text{because } U\neq\hat{U}}} && (11.72)\\
&\le \mathrm{H}_{\mathrm{b}}(P_{\mathrm{e}}) + P_{\mathrm{e}}\log(\mathrm{L}-1), && (11.73)
\end{aligned}$$

where the first inequality follows from conditioning that cannot increase entropy. □

Example 11.26. Let $\mathrm{H}(U|\hat{U}) = \frac{1}{2}$ bits for binary random variables U and $\hat{U}$. Then the Fano Inequality tells us that

$$\mathrm{H}_{\mathrm{b}}(P_{\mathrm{e}}) \ge \frac{1}{2} \text{ bits}, \tag{11.74}$$

i.e.,

$$0.11 \le P_{\mathrm{e}} \le 0.89. \tag{11.75}$$

♢

Often, we use the Fano Inequality in a looser form.

Corollary 11.27. Let U and $\hat{U}$ be L-ary ($\mathrm{L} \ge 2$) RVs taking value in the same alphabet and let the error probability be defined as in (11.63). Then

$$\log 2 + P_{\mathrm{e}}\log \mathrm{L} \ge \mathrm{H}(U|\hat{U}) \tag{11.76}$$

or

$$P_{\mathrm{e}} \ge \frac{\mathrm{H}(U|\hat{U}) - \log 2}{\log \mathrm{L}}. \tag{11.77}$$

Proof: This follows directly from (11.64) by bounding $\mathrm{H}_{\mathrm{b}}(P_{\mathrm{e}}) \le \log 2$ and $\log(\mathrm{L}-1) < \log \mathrm{L}$. □

The second lemma is related to our discussion of Markov processes. We give the following important definition.

Definition 11.28. We say that the three RVs X, Y, and Z form a *Markov chain* and write $X \multimap Y \multimap Z$ if

$$P(z|y,x) = P(z|y) \tag{11.78}$$

or, equivalently,

$$\mathrm{H}(Z|Y,X) = \mathrm{H}(Z|Y). \tag{11.79}$$

A Markov chain has a strong engineering meaning as it is implicitly given by a chain of processing boxes as shown in Figure 11.7.

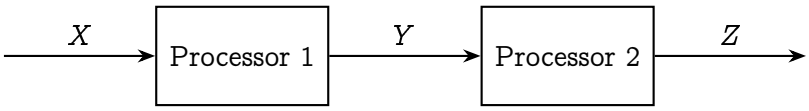


Figure 11.7: A Markov chain. Note that the processors can be deterministic or random.

Remark 11.29. An equivalent definition of a Markov chain is to say that conditionally on Y, the RVs X and Z are independent. We leave it to the reader to prove that this is equivalent to (11.78) and (11.79). △

Lemma 11.30 (Data Processing Inequality (DPI)).
If $X \multimap Y \multimap Z$, then

$$\mathrm{I}(X;Z) \le \mathrm{I}(X;Y) \tag{11.80a}$$
$$\mathrm{I}(X;Z) \le \mathrm{I}(Y;Z) \tag{11.80b}$$

Hence, processing "destroys" (or more precise: cannot "create") information.

Proof: Using Markovity and the fact that conditioning cannot increase entropy, we have

$$\begin{aligned}
\mathrm{I}(Y;Z) &= \mathrm{H}(Z) - \underbrace{\mathrm{H}(Z|Y)}_{=\mathrm{H}(Z|Y,X)} && (11.81)\\
&= \mathrm{H}(Z) - \underbrace{\mathrm{H}(Z|Y,X)}_{\le \mathrm{H}(Z|X)} && (11.82)\\
&\ge \mathrm{H}(Z) - \mathrm{H}(Z|X) && (11.83)\\
&= \mathrm{I}(X;Z). && (11.84)
\end{aligned}$$

□

11.7 Converse to the Channel Coding Theorem

Similarly to the chapters about source coding where we have proven a *source coding theorem*, we will now derive a *channel coding theorem*. And similarly to the source coding theorem, the channel coding theorem consists of a converse part that tells what we cannot do and an achievability part that tells what is possible. We start with the converse part.

Suppose someone gives you an achievable $(\lceil 2^{n\mathrm{R}}\rceil, n)$ coding scheme, i.e., a coding scheme (or rather a sequence of coding schemes) for which $P_{\mathrm{e}}^{(n)} \to 0$ as $n \to \infty$. Taking this scheme and remembering the definition of the code rate (Definition 11.16), we have

$$\begin{aligned}
\mathrm{R} &= \frac{\log 2^{n\mathrm{R}}}{n} && (11.85)\\
&\le \frac{\log\lceil 2^{n\mathrm{R}}\rceil}{n} && (11.86)\\
&= \frac{\log \mathrm{M}}{n} && (11.87)\\
&= \frac{1}{n}\,\mathrm{H}(M) && (11.88)\\
&= \frac{1}{n}\,\mathrm{H}(M) - \frac{1}{n}\,\mathrm{H}(M|\hat{M}) + \frac{1}{n}\,\mathrm{H}(M|\hat{M}) && (11.89)\\
&= \frac{1}{n}\,\mathrm{H}(M|\hat{M}) + \frac{1}{n}\,\mathrm{I}(M;\hat{M}) && (11.90)\\
&\le \frac{1}{n}\,\mathrm{H}(M|\hat{M}) + \frac{1}{n}\,\mathrm{I}(X_1^n;Y_1^n) && (11.91)\\
&\le \frac{\log 2}{n} + \underbrace{\frac{\log \mathrm{M}}{n}}_{\le \left(\mathrm{R}+\frac{1}{n}\right)\log 2} \cdot P_{\mathrm{e}}^{(n)} + \frac{1}{n}\,\mathrm{I}(X_1^n;Y_1^n) && (11.92)\\
&\le \frac{\log 2}{n} + \left(\mathrm{R}+\frac{1}{n}\right) P_{\mathrm{e}}^{(n)} \log 2 + \frac{1}{n}\,\mathrm{H}(Y_1^n) - \frac{1}{n}\,\mathrm{H}(Y_1^n|X_1^n) && (11.93)\\
&= \frac{\log 2}{n} + \left(\mathrm{R}+\frac{1}{n}\right) P_{\mathrm{e}}^{(n)} \log 2 + \frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(Y_k|Y_1^{k-1}) - \frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(Y_k|Y_1^{k-1},X_1^n) && (11.94)\\
&= \frac{\log 2}{n} + \left(\mathrm{R}+\frac{1}{n}\right) P_{\mathrm{e}}^{(n)} \log 2 + \frac{1}{n}\sum_{k=1}^{n}\underbrace{\mathrm{H}(Y_k|Y_1^{k-1})}_{\le \mathrm{H}(Y_k)} - \frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(Y_k|X_k) && (11.95)\\
&\le \frac{\log 2}{n} + \left(\mathrm{R}+\frac{1}{n}\right) P_{\mathrm{e}}^{(n)} \log 2 + \frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(Y_k) - \frac{1}{n}\sum_{k=1}^{n}\mathrm{H}(Y_k|X_k) && (11.96)\\
&= \frac{\log 2}{n} + \left(\mathrm{R}+\frac{1}{n}\right) P_{\mathrm{e}}^{(n)} \log 2 + \frac{1}{n}\sum_{k=1}^{n}\underbrace{\mathrm{I}(X_k;Y_k)}_{\le \max\limits_{P_X}\mathrm{I}(X;Y)} && (11.97)\\
&\le \frac{\log 2}{n} + \left(\mathrm{R}+\frac{1}{n}\right) P_{\mathrm{e}}^{(n)} \log 2 + \frac{1}{n}\sum_{k=1}^{n}\mathrm{C}_{\mathrm{inf}} && (11.98)\\
&= \frac{\log 2}{n} + \left(\mathrm{R}+\frac{1}{n}\right) P_{\mathrm{e}}^{(n)} \log 2 + \mathrm{C}_{\mathrm{inf}}. && (11.99)
\end{aligned}$$

Here, (11.88) follows because we assume that the messages are uniformly distributed; (11.91) follows by twice applying the Data Processing Inequality (Lemma 11.30, see Figure 11.4); the subsequent inequality (11.92) follows from

the Fano Inequality (Corollary 11.27) with $P_{\mathrm{e}}^{(n)} = \Pr[M \neq \hat{M}]$; for (11.93) we bound

$$\frac{1}{n}\log \mathrm{M} = \frac{1}{n}\log\lceil 2^{n\mathrm{R}}\rceil \le \frac{1}{n}\log(2^{n\mathrm{R}}+1) \le \frac{1}{n}\log(2^{n\mathrm{R}}\cdot 2) = \frac{n\mathrm{R}+1}{n}\log 2; \tag{11.100}$$

in (11.94) we use the chain rule; the equality in (11.95) is based on our assumption that we have a DMC without feedback; and (11.98) follows from the definition of the information capacity (Definition 11.13). Note that (11.92)–(11.99) show that

$$\frac{1}{n}\,\mathrm{I}(X_1^n; Y_1^n) \le \mathrm{C}_{\inf}. \tag{11.101}$$

We also point out that (11.95) does not hold if we drop the condition that the DMC is used without feedback! While it is true for any DMC that

$$\mathrm{H}(Y_k|Y_1^{k-1}, X_1^k) = \mathrm{H}(Y_k|X_k) \tag{11.102}$$

due to (11.1), we have here

$$\begin{aligned}
&\mathrm{H}(Y_k|Y_1^{k-1}, X_1^n)\\
&\quad = \mathrm{H}(Y_k|Y_1^{k-1}, X_1^k) - \mathrm{I}(X_{k+1}^n; Y_k|Y_1^{k-1}, X_1^k) &(11.103)\\
&\quad = \mathrm{H}(Y_k|X_k) - \mathrm{H}(X_{k+1}^n|Y_1^{k-1}, X_1^k) + \mathrm{H}(X_{k+1}^n|Y_1^k, X_1^k). &(11.104)
\end{aligned}$$

However, note that

$$\mathrm{H}(X_{k+1}^n|Y_1^{k-1}, X_1^k) = \mathrm{H}(X_{k+1}^n|Y_1^k, X_1^k) \tag{11.105}$$

if, and only if, (11.6) is satisfied, too.

Now recall that we have assumed an achievable coding scheme, i.e., the coding scheme under investigation is by assumption such that $P_{\mathrm{e}}^{(n)} \downarrow 0$ as $n \to \infty$. Thus, for $n \to \infty$, the first two terms in (11.99) tend to zero and we have shown that

$$\mathrm{R} \le \mathrm{C}_{\inf}. \tag{11.106}$$

We have proven the following theorem.

Theorem 11.31 (Converse Part of the Channel Coding Theorem for a DMC). Any coding scheme that has an average error probability $P_{\mathrm{e}}^{(n)}$ going to zero as the blocklength n tends to infinity must have a rate that is not larger than the information capacity:

$$\mathrm{R} \le \mathrm{C}_{\inf}. \tag{11.107}$$

Remark 11.32. Note that this is a *weak* converse. It is possible to prove a stronger version that shows that for rates above capacity, the error probability goes to 1 exponentially fast in n! △

Remark 11.33. Note that we have stated the converse using the average error probability $P_{\mathrm{e}}^{(n)}$. However, Theorem 11.31 also holds if we replace $P_{\mathrm{e}}^{(n)}$ by the maximum error probability $\lambda^{(n)}$. Indeed, since the average error probability certainly will tend to zero if the maximum error probability tends to zero, it follows that any coding scheme with vanishing maximum error probability also must satisfy the converse, i.e., also must have a rate $\mathrm{R} \leq \mathrm{C}_{\inf}$. △

Next we turn to the achievability part.

11.8 Channel Coding Theorem

We will now derive a lower bound to the achievable rates. To this end, we will need to design a coding scheme (codebook, encoder, and decoder) and then analyze its performance. The better this performance, the better the lower bound.

Our main ideas are as follows:

- The aim of our design is not to create a coding scheme (i.e., codebook, encoder and decoder) that can be used in practice, but that is easy to analyze.

- Since we have no clue how to design the codebook, we choose one *at random* from the huge set of all possible codebooks. Then we will analyze the performance of such a random codebook *on average*, and if it is good, then we know that there must exist at least some codebooks that are good!

  So, think of a storehouse that contains all possible codebooks. We go there and randomly pick one and analyze its performance, i.e., compute its average error probability. Then we pick another codebook and analyze its performance, etc. Finally, we average the performance over all such codebooks, i.e., we compute the average of the average error probability.

  This trick was Shannon's idea! It is one of the most crucial ideas in information theory and is called *random coding*.

- The encoder will be a simple mapping based on a lookup table. As mentioned above, such a design is highly inefficient in practice, but easy to analyze. Note that by itself the encoder is not random, but a deterministic mapping that maps any given message into a codeword.

However, since both the message and the codewords are random, the encoder output looks random, too.

- The decoder should be an ML decoder as discussed in Section 11.3. Unfortunately, an ML decoder turns out to be quite difficult to analyze. Therefore we will use a suboptimal decoder. If this decoder will perform well, then an optimal ML decoder will perform even better!

  Note that the decoder by itself is not random, but a deterministic mapping that maps a received sequence into a decision. But again, since message, codewords and channel output all are random, the output of the decoder looks random.

So, let us look at the details. We go through the following steps:

1: **Codebook Design:** We randomly generate a codebook. To this end, we fix a rate R, a codeword length n, and a distribution $P_X(\cdot)$. Then we "design" a $(\lceil 2^{n\mathrm{R}}\rceil, n)$ coding scheme by independently generating $\lceil 2^{n\mathrm{R}}\rceil$ length-n codewords according to the distribution

$$P_{\mathbf{X}}(\mathbf{x}) = \prod_{k=1}^{n} P_X(x_k). \tag{11.108}$$

Note that this codebook can be described by a matrix of $\lceil 2^{n\mathrm{R}}\rceil$ rows of length n:

$$\mathscr{C} = \begin{pmatrix} X_1(1) & X_2(1) & \cdots & X_n(1) \\ X_1(2) & X_2(2) & \cdots & X_n(2) \\ \vdots & \vdots & \ddots & \vdots \\ X_1(\lceil 2^{n\mathrm{R}}\rceil) & X_2(\lceil 2^{n\mathrm{R}}\rceil) & \cdots & X_n(\lceil 2^{n\mathrm{R}}\rceil) \end{pmatrix}, \tag{11.109}$$

where each row describes a codeword. Each entry of this matrix is generated IID $\sim P_X(\cdot)$, i.e.,

$$P(\mathscr{C}) \triangleq \Pr\left[\mathscr{C} = \begin{pmatrix} x_1(1) & \cdots & x_n(1) \\ \vdots & & \vdots \\ x_1(\lceil 2^{n\mathrm{R}}\rceil) & \cdots & x_n(\lceil 2^{n\mathrm{R}}\rceil) \end{pmatrix}\right] \tag{11.110}$$

$$= \prod_{m=1}^{\lceil 2^{n\mathrm{R}}\rceil} \prod_{k=1}^{n} P_X\big(x_k(m)\big). \tag{11.111}$$

Once the codebook has been generated, we reveal it to both transmitter and receiver. And of course they also know the channel law $P_{Y|X}(\cdot|\cdot)$ of the DMC.

2: **Encoder Design:** As always we assume that the source randomly picks a message $M \in \{1, \dots, \lceil 2^{nR} \rceil\}$ according to a uniform distribution

$$\Pr[M = m] = \frac{1}{\lceil 2^{nR} \rceil}. \tag{11.112}$$

The encoder is now very simple: Given the message m, it takes the mth codeword $\mathbf{X}(m)$ of $\mathscr{C}$ (the mth row of (11.109)) and sends it over the channel.

3: **Decoder Design:** The decoder receives the sequence $\mathbf{Y} = \mathbf{y}$, which has, conditional on the transmitted codeword $\mathbf{X}(m)$, the distribution

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{X}(m)) = \prod_{k=1}^{n} P_{Y|X}(y_k|X_k(m)). \tag{11.113}$$

The decoder needs to guess which message has been sent. The best thing to do is to minimize the probability of decoding error or, equivalently, maximizing the probability of a correct decoding as we have described in Section 11.3, i.e., the best thing would be to design the decoder as an ML decoder.

Unfortunately, an ML decoder is quite difficult to analyze.[7] We therefore will use another, suboptimal decoder, that is not very practical, but allows a very elegant and simple analysis. Moreover, even though it is suboptimal, its performance is good enough for our proof purposes.

Before we can explain the design of our decoder, we need to introduce a function of two vectors:

$$\mathrm{i}(\mathbf{x};\mathbf{y}) \triangleq \log \frac{P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}{P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}}(\mathbf{y})}, \quad (\mathbf{x},\mathbf{y}) \in \mathcal{X}^n \times \mathcal{Y}^n. \tag{11.114}$$

This function is called *instantaneous mutual information*. Its name stems from the fact that the mutual information $\mathrm{I}(\mathbf{X};\mathbf{Y})$ between the random vectors $\mathbf{X}$ and $\mathbf{Y}$ is given as the *average* of the instantaneous mutual information between all possible realizations of these vectors:

$$\mathrm{I}(\mathbf{X};\mathbf{Y}) = \mathsf{E}_{\mathbf{X},\mathbf{Y}}[\mathrm{i}(\mathbf{X};\mathbf{Y})] = \mathsf{E}\left[\log \frac{P_{\mathbf{X},\mathbf{Y}}(\mathbf{X},\mathbf{Y})}{P_{\mathbf{X}}(\mathbf{X})P_{\mathbf{Y}}(\mathbf{Y})}\right]. \tag{11.115}$$

Note that we will use the instantaneous mutual information in the context where $P_{\mathbf{X},\mathbf{Y}}$ denotes the joint PMF of our random codewords and our DMC without feedback:

$$P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) = P_{\mathbf{X}}(\mathbf{x}) \cdot P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \prod_{k=1}^{n} P_X(x_k) P_{Y|X}(y_k|x_k), \tag{11.116}$$

[7] It is not impossible to do it, though. Shannon in his 1948 paper [Sha48] does analyze the ML decoder.

where in the second equality we have made use of the memorylessness of the DMC without feedback (see Theorem 11.11) and of the IID codebook generation $P_{\mathbf{X}}(\mathbf{x}) = \prod_{k=1}^{n} P_X(x_k)$. For the same reason we also have

$$P_{\mathbf{Y}}(\mathbf{y}) = \prod_{k=1}^{n} P_Y(y_k), \tag{11.117}$$

where $P_Y(\cdot)$ is the marginal distribution of $P_{X,Y}(\cdot,\cdot) = P_X(\cdot)P_{Y|X}(\cdot|\cdot)$.

Hence, the instantaneous mutual information (11.114) can be rewritten and simplified as follows:

$$\begin{aligned} \imath(\mathbf{x};\mathbf{y}) &= \log \frac{P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})}{P_{\mathbf{Y}}(\mathbf{y})} && (11.118)\\ &= \log \frac{\prod_{k=1}^{n} P_{Y|X}(y_k|x_k)}{\prod_{k=1}^{n} P_Y(y_k)} && (11.119)\\ &= \sum_{k=1}^{n} \log \frac{P_{Y|X}(y_k|x_k)}{P_Y(y_k)} && (11.120)\\ &\triangleq \sum_{k=1}^{n} \imath(x_k;y_k), && (11.121) \end{aligned}$$

where (11.121) should be read as definition of $\imath(x_k;y_k)$.

Now we are ready to describe the design of our suboptimal decoder. Our decoder is a *threshold decoder* and works as follows. Given a received $\mathbf{Y} = \mathbf{y}$, the decoder computes the instantaneous mutual information between any codeword $\mathbf{X}(\tilde{m})$ and the received sequence $\mathbf{y}$: $\imath(\mathbf{X}(\tilde{m});\mathbf{y})$ for all $\tilde{m} = 1,\ldots,\lceil 2^{nR}\rceil$. Then it checks if it can find an $\hat{m}$ such that at the same time

$$\imath(\mathbf{X}(\hat{m});\mathbf{y}) > \log\beta \tag{11.122}$$

and

$$\imath(\mathbf{X}(\tilde{m});\mathbf{y}) \le \log\beta, \quad \forall\, \tilde{m} \neq \hat{m}, \tag{11.123}$$

for some specified threshold $\log\beta$ (we will describe our choice of β below). If such an $\hat{m}$ exists, the decoder will put out this $\hat{m}$. If not, it will put out $\hat{m} = 0$, i.e., it declares an error.

4: **Performance Analysis:** There is a decoding error if $\hat{M} \neq M$. (Note that if $\hat{M} = 0$, then for sure this is the case.) To analyze the error probability we now use the trick not to try to analyze one particular codebook, but instead the *average over all possible codebooks*, i.e., we also average over

the random codebook generation:

$$\begin{aligned}
\Pr(\text{error}) &= \Pr(\text{error, averaged over all codewords } \textit{and} \text{ -books}) && (11.124)\\
&= \sum_{\mathscr{C}} \underbrace{P(\mathscr{C})}_{\substack{\text{probability of}\\ \text{choosing codebook } \mathscr{C}}} \cdot \underbrace{P_{\mathrm{e}}^{(n)}(\mathscr{C})}_{\substack{\text{average error prob.}\\ \text{for given codebook } \mathscr{C}}} && (11.125)\\
&= \sum_{\mathscr{C}} P(\mathscr{C}) \cdot \frac{1}{\lceil 2^{nR}\rceil} \sum_{m=1}^{\lceil 2^{nR}\rceil} \lambda_m(\mathscr{C}) && (11.126)\\
&= \frac{1}{\lceil 2^{nR}\rceil} \sum_{m=1}^{\lceil 2^{nR}\rceil} \underbrace{\sum_{\mathscr{C}} P(\mathscr{C})\lambda_m(\mathscr{C})}_{\text{independent of } m} && (11.127)\\
&= \frac{1}{\lceil 2^{nR}\rceil} \sum_{m=1}^{\lceil 2^{nR}\rceil} \sum_{\mathscr{C}} P(\mathscr{C})\lambda_1(\mathscr{C}) && (11.128)\\
&= \sum_{\mathscr{C}} P(\mathscr{C})\lambda_1(\mathscr{C}) && (11.129)\\
&= \Pr(\text{error, averaged over all codebooks} \,|\, M=1). && (11.130)
\end{aligned}$$

Here, in (11.128) we use the fact that all codewords are IID randomly generated. Hence, since we average over all codebooks, the average error probability does not depend on which message has been sent. So without loss of generality we can assume that $M = 1$.

Let

$$\mathcal{F}_m \triangleq \Big\{ \imath(\mathbf{X}(m);\mathbf{Y}) > \log_2 \beta \Big\}, \quad m = 1, \ldots, \lceil 2^{nR}\rceil, \tag{11.131}$$

be the event that the instantaneous mutual information of the mth codeword and the received sequence $\mathbf{Y}$ is above the threshold. Note that for simplicity we choose a base 2 for the logarithms.

An error occurs if $\mathcal{F}_1^{\mathrm{c}}$ occurs (i.e., if the instantaneous mutual information of transmitted codeword and the received sequence is too small) or if $\mathcal{F}_2 \cup \mathcal{F}_3 \cup \cdots \cup \mathcal{F}_{\lceil 2^{nR}\rceil}$ occurs (i.e., if one or more wrong codewords have an instantaneous mutual information with the received sequence $\mathbf{Y}$ that is too big). Hence, we can write (11.130) as follows:

$$\begin{aligned}
\Pr(\text{error}) &= \Pr(\text{error} \,|\, M=1) && (11.132)\\
&= \Pr\Big(\mathcal{F}_1^{\mathrm{c}} \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \cdots \cup \mathcal{F}_{\lceil 2^{nR}\rceil} \,\Big|\, M=1\Big) && (11.133)\\
&\le \Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M=1) + \sum_{m=2}^{\lceil 2^{nR}\rceil} \Pr(\mathcal{F}_m \,|\, M=1) && (11.134)
\end{aligned}$$

where the inequality follows from the *Union Bound*: for any two events $\mathcal{A}$ and $\mathcal{B}$:

$$\Pr(\mathcal{A}\cup\mathcal{B}) = \Pr(\mathcal{A}) + \Pr(\mathcal{B}) - \Pr(\mathcal{A}\cap\mathcal{B}) \le \Pr(\mathcal{A}) + \Pr(\mathcal{B}). \tag{11.135}$$

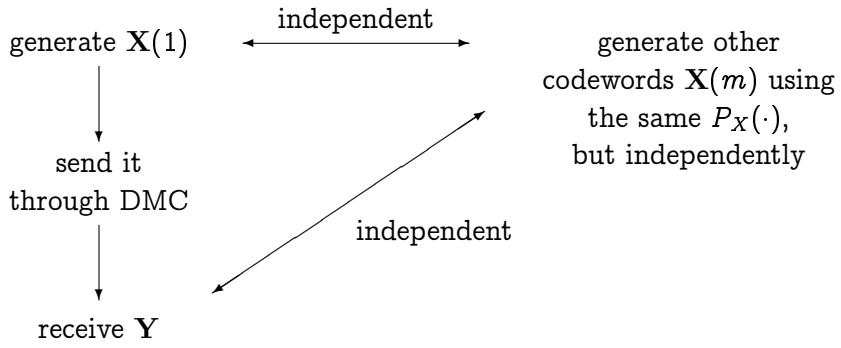


Figure 11.8: The received vector $\mathbf{Y}$ is independent of the codewords $\mathbf{X}(m)$, $m \geq 2$, that are not transmitted.

Note that $\mathbf{Y}$ is completely independent of $\mathbf{X}(m)$ for all $m \geq 2$ (see Figure 11.8 for an illustration of this fact). Hence, for $m \geq 2$,

$$\begin{aligned}
\Pr(\mathcal{F}_m \,|\, M=1) &= \Pr\bigl[\imath(\mathbf{X}(m);\mathbf{Y}) > \log_2 \beta \,\big|\, M=1\bigr] && (11.136)\\
&= \Pr\left[\log_2 \frac{P_{\mathbf{X},\mathbf{Y}}(\mathbf{X}(m),\mathbf{Y})}{P_{\mathbf{X}}(\mathbf{X}(m))P_{\mathbf{Y}}(\mathbf{Y})} > \log_2 \beta \,\middle|\, M=1\right] && (11.137)\\
&= \Pr\left[\frac{P_{\mathbf{X},\mathbf{Y}}(\mathbf{X}(m),\mathbf{Y})}{P_{\mathbf{X}}(\mathbf{X}(m))P_{\mathbf{Y}}(\mathbf{Y})} > \beta \,\middle|\, M=1\right] && (11.138)\\
&= \sum_{\substack{(\mathbf{x},\mathbf{y}) \text{ such that}\\ P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})>\beta P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}}(\mathbf{y})}} P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}}(\mathbf{y}) && (11.139)\\
&= \sum_{\substack{(\mathbf{x},\mathbf{y}) \text{ such that}\\ P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}}(\mathbf{y})<\frac{1}{\beta}P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}} \underbrace{P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}}(\mathbf{y})}_{<\frac{1}{\beta}P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})} && (11.140)\\
&< \sum_{\substack{(\mathbf{x},\mathbf{y}) \text{ such that}\\ P_{\mathbf{X}}(\mathbf{x})P_{\mathbf{Y}}(\mathbf{y})<\frac{1}{\beta}P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}} \frac{1}{\beta} P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) && (11.141)\\
&\leq \frac{1}{\beta} \sum_{(\mathbf{x},\mathbf{y})} P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) = \frac{1}{\beta}. && (11.142)
\end{aligned}$$

Here, the most important step is (11.139): There we make use of the fact that $\mathbf{X}(m) \perp\!\!\!\perp \mathbf{Y}$, i.e., their joint probability distribution is a product distribution. In the last inequality we upper-bound the sum by including all possible pairs $(\mathbf{x},\mathbf{y})$ without any restriction, such that the summation adds up to 1.

In order to bound the term $\Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M=1)$, we make a clever choice for our threshold β. We fix some $\epsilon > 0$ and choose

$$\beta \triangleq 2^{n(\mathrm{I}(X;Y)-\epsilon)}, \tag{11.143}$$

where $\mathrm{I}(X;Y)$ is the mutual information between a channel input symbol X and its corresponding output symbol Y when the input has the distribution $P_X(\cdot)$ (the same as used for the codebook generation!). Then

$$\begin{aligned}
&\Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M=1)\\
&\quad = \Pr\bigl[\mathrm{i}(\mathbf{X}(1);\mathbf{Y}) \le \log_2 \beta \,\big|\, M=1\bigr] && (11.144)\\
&\quad = \Pr\bigl[\mathrm{i}(\mathbf{X}(1);\mathbf{Y}) \le n\bigl(\mathrm{I}(X;Y)-\epsilon\bigr) \,\big|\, M=1\bigr] && (11.145)\\
&\quad = \Pr\left[\sum_{k=1}^{n} \mathrm{i}(X_k(1);Y_k) \le n\bigl(\mathrm{I}(X;Y)-\epsilon\bigr) \,\middle|\, M=1\right] && (11.146)\\
&\quad = \Pr\left[\frac{1}{n}\sum_{k=1}^{n} \mathrm{i}(X_k(1);Y_k) - \mathrm{I}(X;Y) \le -\epsilon \,\middle|\, M=1\right] && (11.147)\\
&\quad \le \Pr\left[\left|\frac{1}{n}\sum_{k=1}^{n} \mathrm{i}(X_k(1);Y_k) - \mathrm{I}(X;Y)\right| \ge \epsilon \,\middle|\, M=1\right], && (11.148)
\end{aligned}$$

where in (11.146) we have used (11.121).

Now recall the weak law of large numbers, which says that for a sequence $Z_1,\dots,Z_n$ of IID random variables of mean μ and finite variance and for any $\epsilon>0$,

$$\lim_{n\to\infty} \Pr\left[\left|\frac{Z_1+\cdots+Z_n}{n} - \mu\right| \ge \epsilon\right] = 0. \tag{11.149}$$

Applied to our case in (11.148), we have $Z_k \triangleq \mathrm{i}(X_k(1);Y_k)$ and $\mu = \mathrm{E}[Z_k] = \mathrm{E}\bigl[\mathrm{i}(X_k(1);Y_k)\bigr] = \mathrm{I}(X;Y)$. Hence, it follows from (11.148) that

$$\Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M=1) \to 0 \text{ for } n \uparrow \infty, \tag{11.150}$$

i.e., for n large enough, we have $\Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M=1) \le \epsilon$.

Combining these results with (11.134) and choosing n large enough, we now get

$$\begin{aligned}
\Pr(\text{error}) &\le \underbrace{\Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M=1)}_{\le \epsilon} + \sum_{m=2}^{\lceil 2^{n\mathrm{R}}\rceil} \underbrace{\Pr(\mathcal{F}_m \,|\, M=1)}_{< \frac{1}{\beta}} && (11.151)\\
&< \epsilon + \sum_{m=2}^{\lceil 2^{n\mathrm{R}}\rceil} \frac{1}{\beta} && (11.152)\\
&= \epsilon + \underbrace{\bigl(\lceil 2^{n\mathrm{R}}\rceil - 1\bigr)}_{\le 2^{n\mathrm{R}}} 2^{-n(\mathrm{I}(X;Y)-\epsilon)} && (11.153)\\
&\le \epsilon + 2^{n\mathrm{R}} \cdot 2^{-n(\mathrm{I}(X;Y)-\epsilon)} && (11.154)\\
&= \epsilon + 2^{-n(\mathrm{I}(X;Y)-\mathrm{R}-\epsilon)} && (11.155)\\
&\le 2\epsilon, && (11.156)
\end{aligned}$$

where in the last step again n needs to be sufficiently large and $\mathrm{I}(X;Y) - \mathrm{R} - \epsilon > 0$ so that the exponent is negative. Hence we see that as long as

$$\mathrm{R} < \mathrm{I}(X;Y) - \epsilon \tag{11.157}$$

for any $\epsilon > 0$ we can choose n large enough such that the average error probability, averaged over all codewords and all codebooks, is less than 2ϵ:

$$\Pr(\text{error}) \le 2\epsilon. \tag{11.158}$$

5: Strengthening: Now we strengthen the proof in several ways.

- Firstly, we would like to get rid of the average over the codebooks. Since our average error probability is small ($\le 2\epsilon$), there must exist at least one codebook $\mathscr{C}^*$ that has an equally small error probability, i.e., we can find a codebook $\mathscr{C}^*$ satisfying

$$P_{\mathrm{e}}^{(n)}(\mathscr{C}^*) \le 2\epsilon. \tag{11.159}$$

  Note that in our calculation of the average error probability we have included *all* possible codebooks, i.e., really stupid ones with, e.g., all codewords being the same (which means that the error probability for this codebook will be 1!) were not excluded.

- Secondly, we strengthen the proof to obtain a result with respect to the maximum error probability $\lambda^{(n)}$ instead of the average error probability $P_{\mathrm{e}}^{(n)}$. To this end, we consider the good codebook $\mathscr{C}^*$, for which we know that (11.159) holds, and order the codewords according to their error probabilities λ_m, $m = 1, \dots, \lceil 2^{n\mathrm{R}} \rceil$. Let $\tilde{\lambda} \triangleq \lambda_{\lfloor 2^{n\mathrm{R}}/2 \rfloor}$ be the error probability of the worst codeword of the better half. Then

$$\begin{aligned}
2\epsilon &\ge P_{\mathrm{e}}^{(n)}(\mathscr{C}^*) && (11.160)\\
&= \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{m=1}^{\lceil 2^{n\mathrm{R}} \rceil} \lambda_m && (11.161)\\
&= \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{\substack{m=1\\ \text{(better half)}}}^{\lfloor 2^{n\mathrm{R}}/2 \rfloor} \underbrace{\lambda_m}_{\ge 0} + \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{\substack{m=\lfloor 2^{n\mathrm{R}}/2 \rfloor + 1\\ \text{(worse half)}}}^{\lceil 2^{n\mathrm{R}} \rceil} \underbrace{\lambda_m}_{\ge \tilde{\lambda}} && (11.162)\\
&\ge \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{m=\lfloor 2^{n\mathrm{R}}/2 \rfloor + 1}^{\lceil 2^{n\mathrm{R}} \rceil} \tilde{\lambda} && (11.163)\\
&= \underbrace{\frac{\lceil 2^{n\mathrm{R}} \rceil - \lfloor 2^{n\mathrm{R}}/2 \rfloor}{\lceil 2^{n\mathrm{R}} \rceil}}_{\ge \frac{1}{2}} \cdot \tilde{\lambda} && (11.164)\\
&\ge \frac{\tilde{\lambda}}{2}. && (11.165)
\end{aligned}$$

Hence, $\tilde{\lambda} \le 4\epsilon$. So if we design a new codebook with only half the amount of codewords by taking the good codebook $\mathscr{C}^*$ and throwing away the worse half of the codewords, we get a new codebook $\tilde{\mathscr{C}}^*$ with a maximum error probability being very small:

$$\tilde{\lambda}^{(n)} \le 4\epsilon. \tag{11.166}$$

But by throwing away half of the codewords we have also changed the rate: The rate of $\tilde{\mathscr{C}}^*$ satisfies

$$\tilde{\mathsf{R}} = \frac{\log_2\bigl(\lfloor \frac{2^{n\mathsf{R}}}{2} \rfloor\bigr)}{n} = \frac{\log_2\bigl(\lfloor 2^{n\mathsf{R}-1} \rfloor\bigr)}{n} \le \frac{n\mathsf{R}-1}{n} = \mathsf{R} - \frac{1}{n}. \tag{11.167}$$

So we see that for large n, we do not lose much.

The condition (11.157) now becomes

$$\begin{aligned} \tilde{\mathsf{R}} &\le \mathsf{R} - \frac{1}{n} && (11.168)\\ &< \mathrm{I}(X;Y) - \epsilon - \frac{1}{n} && (11.169)\\ &\le \mathrm{I}(X;Y) - 2\epsilon, && (11.170) \end{aligned}$$

where in (11.170) we choose $n \ge \frac{1}{\epsilon}$.

- Thirdly, we note that we have not yet clearly specified how we want to choose $P_X(\cdot)$ in the codebook design process. Now we decide to choose it to be the information-capacity-achieving input distribution $P_X^*(\cdot)$, i.e., the distribution that achieves the maximum in

$$\mathrm{C}_{\inf} \triangleq \max_{P_X(\cdot)} \mathrm{I}(X;Y). \tag{11.171}$$

This choice allows us to loosen condition (11.170) to

$$\tilde{\mathsf{R}} < \mathrm{C}_{\inf} - 2\epsilon. \tag{11.172}$$

- Finally, we remember that ϵ is a parameter that we can choose freely. Hence, we can make it as small as we wish, so that condition (11.172) can be replaced by

$$\tilde{\mathsf{R}} < \mathrm{C}_{\inf}. \tag{11.173}$$

Thus, we have shown that it is possible to design a coding scheme with arbitrarily small maximum error probability if n is chosen large enough and if the rate is smaller than the information capacity.

We have proven the following very exciting result.

Theorem 11.34 (The Channel Coding Theorem for a DMC).
Let a DMC $(\mathcal{X}, \mathcal{Y}, P_{Y|X})$ be given. Define the capacity of this DMC as

$$\mathrm{C} \triangleq \max_{P_X(\cdot)} \mathrm{I}(X;Y) \tag{11.174}$$

where X and Y are the input and output RVs of this DMC. Then all rates below C are achievable, i.e., for every $\mathrm{R} < \mathrm{C}$ there exists a sequence of $(\lceil 2^{n\mathrm{R}} \rceil, n)$ coding schemes with maximum error probability $\lambda^{(n)} \to 0$ as the blocklength n gets very large.

Conversely, any sequence of $(\lceil 2^{n\mathrm{R}} \rceil, n)$ coding schemes with average error probability $P_{\mathrm{e}}^{(n)} \to 0$ must have a rate $\mathrm{R} \leq \mathrm{C}$.

We would like to comment on this, in this class probably most important result.

- We have proven that $\mathrm{C}_{\mathrm{inf}} = \mathrm{C}_{\mathrm{op}}$. Therefore, henceforth we will not make the distinction between information and operational capacity anymore, but simply call it *channel capacity* C.

- Note that the inequality $\mathrm{R} < \mathrm{C}$ is strict for the achievable rates! If $\mathrm{R} = \mathrm{C}$, the situation is unclear: Depending on the channel, C might be achievable or not.

- The result is counterintuitive: As long as $\mathrm{R} < \mathrm{C}$, we can increase the amount of codewords without having a higher error probability! Or equivalently: As long as $\mathrm{R} < \mathrm{C}$, we can make the error probability smaller without reducing the rate!

- To summarize: As long as we transmit not too much information per channel use, we can do it as reliable as you wish (give me any ϵ, e.g., $\epsilon = 10^{-100}$ is possible!), but if you are above C, you are not able to transmit reliably at all!

- This result was like a bombshell in 1948! Before 1948 the engineers believed that you need to increase your power or SNR (i.e., make the channel better!) in order to get better error probability. However, Shannon proved that you can do it without changing the channel (in particular, without increasing power).

- Where do we pay? (There is no free lunch!) Note that the result only holds for $n \to \infty$. So for a very small choice of ϵ, the necessary n might be very large. This will cause

  - a very complex system (how shall we decode codewords of length 1′000′000?);
  - very long delays (we need to wait until we have received the complete received sequence before we can decode!).

- The proof only shows that there *exists* a good code, but unfortunately does not give any clue on how to find it!

- In practice we have the additional problem that the encoding and decoding functions must be efficient. For example, a simple lookup table at the decoder for all possible received sequences will not work.

  Example 11.35. Assume a binary channel and a coding scheme with rate $\mathrm{R} = \frac{1}{4}$ and blocklength $n = 1024$. Then we have $\mathrm{M} = 2^{n\mathrm{R}} = 2^{256}$ possible messages, i.e., we need 256 bits to store the decision for each of the $2^n = 2^{1024}$ different possible received sequences. Hence the decoder needs a memory of size

  $$\begin{aligned} 2^{1024} \cdot 256 \text{ bits} = 2^{1032} \text{ bits} &= 10^{1032 \log_{10} 2} \text{ bits} \\ &= 4.60 \cdot 10^{310} \text{ bits} = 5.75 \cdot 10^{297} \text{ TB}. \end{aligned} \tag{11.175}$$

  Note that if we took all existing hard-drives worldwide together, we would not even reach 10^{30} TB for many years to come... ◊

- In practice it took about 50 years before the first practical codes were found that performed close to the limit predicted by Shannon. See the discussion at the end of Chapter 17.

Chapter 12

Computing Capacity

12.1 Introduction

We have now learned the definition and meaning of the channel capacity for a DMC. Next we need to tackle the problem of how we can actually compute its value. It turns out that in many cases this is very difficult. But there is still hope because we have a couple of tricks available that sometimes help.

We start by recalling the definition of capacity: For a given DMC with conditional probability distribution $P_{Y|X}$ the capacity is given as

$$\mathrm{C} = \max_{P_X(\cdot)} \mathrm{I}(X;Y). \tag{12.1}$$

So how can we compute it? This usually is difficult because

$$\mathrm{C} = \max_{P_X(\cdot)} \Big\{ \underbrace{\mathrm{H}(X)}_{\substack{\text{nice!}\\ \text{depends on}\\ P_X \text{ only}}} - \underbrace{\mathrm{H}(X|Y)}_{\substack{\text{very annoying!}\\ \text{looking backwards}\\ \text{through DMC!}}} \Big\} \tag{12.2}$$

$$= \max_{P_X(\cdot)} \Big\{ \underbrace{\mathrm{H}(Y)}_{\substack{\text{annoying!}\\ \text{need to compute } P_Y\\ \text{from } P_X \text{ and } P_{Y|X}}} - \underbrace{\mathrm{H}(Y|X)}_{\substack{\text{nice!}\\ \text{DMC}}} \Big\}. \tag{12.3}$$

We will show two different approaches on how to compute capacity. The first is based on some special symmetry assumption about the DMC under consideration (strongly and weakly symmetric DMCs). The second approach relies on the Karush–Kuhn–Tucker conditions and convexity.

12.2 Strongly Symmetric DMCs

Definition 12.1. A DMC is called *uniformly dispersive* if there exist numbers $p_1, p_2, \ldots, p_{|\mathcal{Y}|}$ such that

$$\Big[P_{Y|X}(y_1|x), P_{Y|X}(y_2|x), \ldots, P_{Y|X}(y_{|\mathcal{Y}|}|x)\Big]$$

is a permutation of $[p_1, p_2, \ldots, p_{|\mathcal{Y}|}]$ for all $x \in \mathcal{X}$.

Example 12.2. A BEC as shown in Figure 12.1 is uniformly dispersive. The corresponding numbers are $[p_1, p_2, p_3] = [1-\delta, \delta, 0]$. ◊

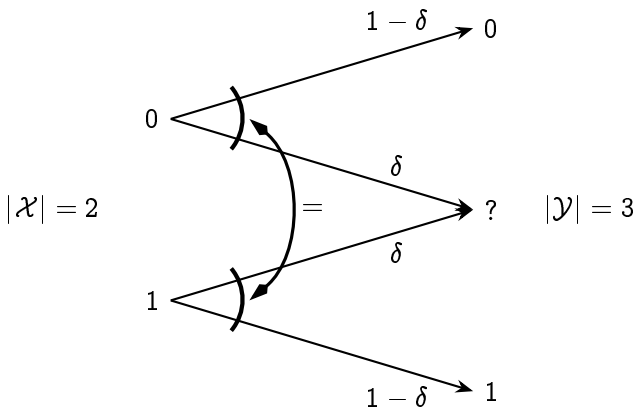


Figure 12.1: The BEC is uniformly dispersive.

Lemma 12.3. A uniformly dispersive DMC satisfies

$$\mathrm{H}(Y|X) = -\sum_{j=1}^{|\mathcal{Y}|} p_j \log p_j, \tag{12.4}$$

independently of $P_X(\cdot)$.

Proof: Using the definition of entropy, we have

$$\mathrm{H}(Y|X=x) = -\sum_{y} P_{Y|X}(y|x) \log P_{Y|X}(y|x) = -\sum_{j=1}^{|\mathcal{Y}|} p_j \log p_j, \tag{12.5}$$

where the second equality follows from the assumption that the DMC is uniformly dispersive. Note that this expression does not depend on x. Hence,

$$\mathrm{H}(Y|X) = \mathsf{E}_X[\mathrm{H}(Y|X=x)] = -\sum_{j=1}^{|\mathcal{Y}|} p_j \log p_j. \tag{12.6}$$

□

Definition 12.4. A DMC is called *uniformly focusing* if there exist numbers $r_1, r_2, \ldots, r_{|\mathcal{X}|}$ such that

$$[P_{Y|X}(y|x_1), \ldots, P_{Y|X}(y|x_{|\mathcal{X}|})]$$

is a permutation of $[r_1, r_2, \ldots, r_{|\mathcal{X}|}]$ for all $y \in \mathcal{Y}$.

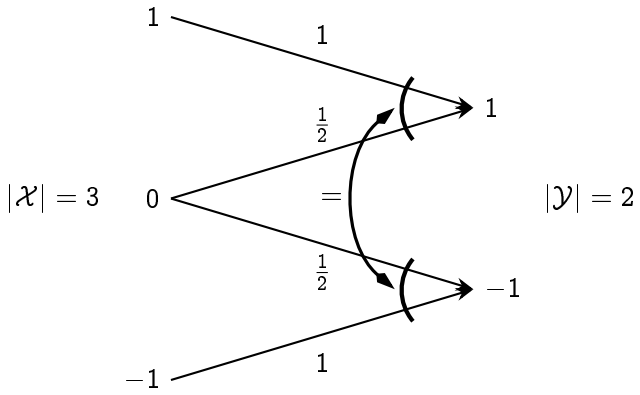


Figure 12.2: The uniform inverse BEC is uniformly focusing.

Example 12.5. The uniform inverse BEC as shown in Figure 12.2 is uniformly focusing. The corresponding numbers are $[r_1, r_2, r_3] = [1, \frac{1}{2}, 0]$. ◊

Lemma 12.6. A uniformly focusing DMC has the following properties:

1. If $P_X(\cdot)$ is uniform, then also $P_Y(\cdot)$ is uniform.
2. The maximum output entropy is

$$\max_{P_X(\cdot)} \mathrm{H}(Y) = \log|\mathcal{Y}| \tag{12.7}$$

   and is achieved by a uniform $P_X(\cdot)$ (but possibly also by other choices of $P_X(\cdot)$).

Proof: We start with the derivation of $P_Y(\cdot)$ if $P_X(\cdot)$ is uniform:

$$P_Y(y) = \sum_x P_X(x) P_{Y|X}(y|x) \tag{12.8}$$

$$= \sum_x \frac{1}{|\mathcal{X}|} P_{Y|X}(y|x) \tag{12.9}$$

$$= \frac{1}{|\mathcal{X}|} \sum_{\ell=1}^{|\mathcal{X}|} r_\ell \tag{12.10}$$

$$= \text{constant} = \frac{1}{|\mathcal{Y}|}. \tag{12.11}$$

Note that the r_ℓ do not in general sum to one, but to some general nonnegative number. However, if $P_Y(y)$ is constant for all values of y, then this constant must be $\frac{1}{|\mathcal{Y}|}$ because $P_Y(\cdot)$ must sum to 1.

To prove the second statement, we note that by the properties of entropy we have

$$\mathrm{H}(Y) \le \log|\mathcal{Y}|, \tag{12.12}$$

with equality if, and only if, Y is uniform. However, since we have just shown that if X is uniform, Y is uniform, too, we see that $\max_{P_X(\cdot)} \mathrm{H}(Y) = \log|\mathcal{Y}|$. □

Definition 12.7. A DMC is called *strongly symmetric*[1] if it is both uniformly dispersive and uniformly focusing.

Example 12.8. The BSC as shown in Figure 12.3 is strongly symmetric. The corresponding numbers are $[p_1, p_2] = [1-\epsilon, \epsilon]$ and $[r_1, r_2] = [1-\epsilon, \epsilon]$. ◊

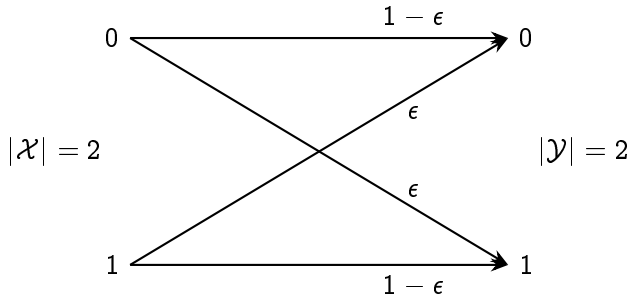


Figure 12.3: Example of a strongly symmetric DMC.

Theorem 12.9 (Capacity of a Strongly Symmetric DMC).
The capacity of a strongly symmetric DMC with input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$, and transition probabilities $p_1, \dots, p_{|\mathcal{Y}|}$ is given as

$$\mathrm{C} = \log|\mathcal{Y}| + \sum_{j=1}^{|\mathcal{Y}|} p_j \log p_j \tag{12.13}$$

and is achieved by a uniform input distribution

$$P_X(x) = \frac{1}{|\mathcal{X}|}, \quad x \in \mathcal{X} \tag{12.14}$$

(but possibly also by other distributions).

Proof: By definition of capacity we have

$$\mathrm{C} = \max_{P_X(\cdot)} \{\mathrm{H}(Y) - \mathrm{H}(Y|X)\} \tag{12.15}$$

$$= \max_{P_X(\cdot)} \mathrm{H}(Y) + \sum_{j=1}^{|\mathcal{Y}|} p_j \log p_j \tag{12.16}$$

[1] In some books strongly symmetric DMCs are called "symmetric DMCs" only. See, e.g., [CT06].

$$= \log|\mathcal{Y}| + \sum_{j=1}^{|\mathcal{Y}|} p_j \log p_j, \tag{12.17}$$

where the second equality (12.16) follows from Lemma 12.3 and the third equality (12.17) from Lemma 12.6. □

Example 12.10. Take again the BSC of Figure 12.3, with $|\mathcal{Y}| = 2$, $p_1 = 1-\epsilon$, $p_2 = \epsilon$. Then we have

$$\mathrm{C} = \log 2 + (1-\epsilon)\log(1-\epsilon) + \epsilon \log \epsilon = 1 - \mathrm{H_b}(\epsilon) \text{ bits}. \tag{12.18}$$

Note that

- if $\epsilon = 0$, then $\mathrm{C} = 1$ bit;
- if $\epsilon = 1$, then $\mathrm{C} = 1$ bit, too.

In the latter case the DMC deterministically swaps zeros and ones which can be repaired without problem (i.e., without error). ◊

12.3 Weakly Symmetric DMCs

The assumption of strong symmetry is pretty stringent. Most DMCs are not strongly symmetric. Luckily, there is a slightly less strong assumption for which we can compute the capacity, too.

Definition 12.11. A DMC $(\mathcal{X}, \mathcal{Y}, P_{Y|X})$ is called *weakly symmetric* if it can be split up into several strongly symmetric DMCs (called *subchannels*) in the following way:

- The input alphabets $\mathcal{X}_s$ of all subchannels are the same and equal to $\mathcal{X}$:

$$\mathcal{X}_s = \mathcal{X}, \quad \forall\, s. \tag{12.19}$$

- The output alphabets $\mathcal{Y}_s$ of the different subchannels are disjoint and add up to $\mathcal{Y}$:

$$\mathcal{Y}_s \cap \mathcal{Y}_{s'} = \emptyset, \quad \forall\, s \neq s', \tag{12.20}$$

$$\bigcup_s \mathcal{Y}_s = \mathcal{Y}. \tag{12.21}$$

- We can find a stochastic process that randomly selects one of the subchannels in such a way that the combination of selection process and subchannel yields the same conditional probability distribution as the channel law $P_{Y|X}$ of the DMC.

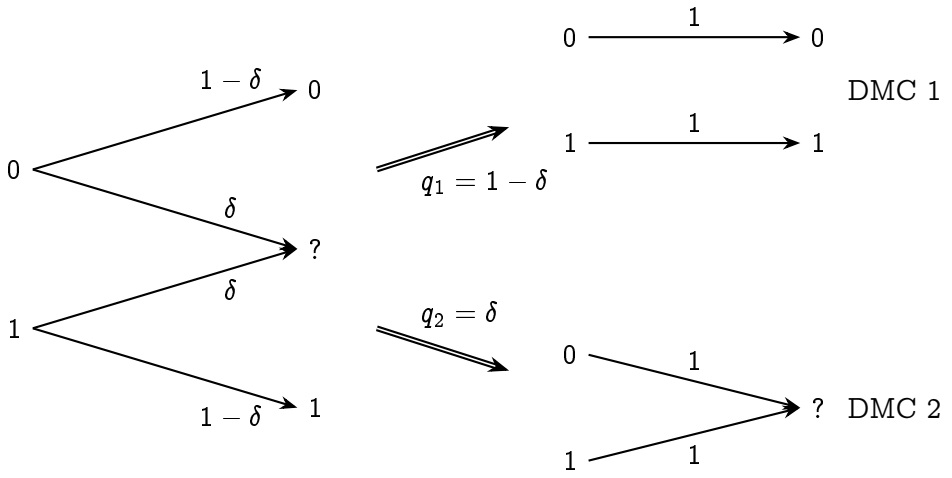


Figure 12.4: The BEC is split into two strongly symmetric DMCs.

Example 12.12. Take again the BEC of Figure 12.1. We split it up into two strongly symmetric DMCs as shown in Figure 12.4. The selection process chooses the first subchannel with probability $q_1 = 1 - \delta$ and the second with probability $q_2 = \delta$. ◊

It is fairly easy to test a DMC for weak symmetry. One merely needs to examine each subset of output letters with the same focusing, and then check to see whether each input letter has the same dispersion into this subset of output letters. Note that it is not hard to see that if a channel is weakly symmetric, it must be uniformly dispersive. We summarize this idea in the following algorithm.

Algorithm for checking whether a DMC is weakly symmetric:

Step 1: Check whether the channel is uniformly dispersive. If no $\Longrightarrow$ abort!

Step 2: Partition the output letters into *disjoint* sets $\mathcal{Y}_s$, $s = 1, \ldots, S$, in such a way that all output letters with the same focusing are combined together into the same set $\mathcal{Y}_s$.

Step 3: For each $\mathcal{Y}_s$, check whether all inputs have the same dispersion. If no $\Longrightarrow$ abort!

Step 4: The DMC is weakly symmetric! We have S subchannels with input alphabet $\mathcal{X}$ and output alphabet $\mathcal{Y}_s$, $s = 1, \ldots, S$. We

compute the selection probabilities of each subchannel as follows:

$$q_s \triangleq \sum_{y\in\mathcal{Y}_s} P_{Y|X}(y|x_1), \quad s = 1,\ldots,\mathrm{S}. \tag{12.22}$$

Then we normalize the conditional subchannel probabilities by dividing each original conditional probability by q_s:

$$P_{Y|X,S=s}(y|x) = \frac{P_{Y|X}(y|x)}{q_s}, \quad x\in\mathcal{X},\ y\in\mathcal{Y}_s, \tag{12.23}$$

for $s = 1,\ldots,\mathrm{S}$.

Since we know how to compute the capacity of every strongly symmetric subchannel, it is not surprising that the capacity of a weakly symmetric DMC can also be derived easily.

Theorem 12.13 (Capacity of a Weakly Symmetric DMC).
The capacity of a weakly symmetric DMC is

$$\mathrm{C} = \sum_{s=1}^{\mathrm{S}} q_s \mathrm{C}_s, \tag{12.24}$$

where C_s denotes the capacity of the sth strongly symmetric subchannel, and q_s is the selection probability of the sth subchannel.

Proof: We decompose the weakly symmetric DMC into S strongly symmetric DMCs, with the selection probabilities $q_1,\ldots,q_{\mathrm{S}}$. Define S to be the selection RV, i.e., if $S = s$, then the sth subchannel is selected. Since the output alphabets are disjoint, it follows that once we know Y, we also know S. Therefore, $\mathrm{H}(S|Y) = 0$, and therefore also $\mathrm{H}(S|Y,X) = 0$. Moreover, since $X \perp\!\!\!\perp S$ (the input is independent of the channel!), we also have $\mathrm{H}(S|X) = \mathrm{H}(S)$. Hence, we get

$$\begin{aligned}
\mathrm{H}(Y) &= \mathrm{H}(Y) + \mathrm{H}(S|Y) && (12.25)\\
&= \mathrm{H}(Y,S) && (12.26)\\
&= \mathrm{H}(S) + \mathrm{H}(Y|S) && (12.27)\\
&= \mathrm{H}(S) + \sum_{s=1}^{\mathrm{S}} P_S(s)\cdot \mathrm{H}(Y|S=s) && (12.28)\\
&= \mathrm{H}(S) + \sum_{s=1}^{\mathrm{S}} q_s\, \mathrm{H}(Y|S=s) && (12.29)
\end{aligned}$$

and

$$\begin{aligned} \mathrm{H}(Y|X) &= \mathrm{H}(Y|X) + \mathrm{H}(S|X,Y) && (12.30)\\ &= \mathrm{H}(Y,S|X) && (12.31)\\ &= \mathrm{H}(S|X) + \mathrm{H}(Y|S,X) && (12.32)\\ &= \mathrm{H}(S) + \sum_{s=1}^{\mathrm{S}} q_s\, \mathrm{H}(Y|X,S=s). && (12.33) \end{aligned}$$

Combined this yields

$$\begin{aligned} \mathrm{I}(X;Y) &= \mathrm{H}(Y) - \mathrm{H}(Y|X) && (12.34)\\ &= \sum_{s=1}^{\mathrm{S}} q_s \big(\mathrm{H}(Y|S=s) - \mathrm{H}(Y|X,S=s)\big) && (12.35)\\ &= \sum_{s=1}^{\mathrm{S}} q_s\, \mathrm{I}(X;Y|S=s), && (12.36) \end{aligned}$$

where $\mathrm{I}(X;Y|S=s)$ is the mutual information of the sth subchannel. We know that

$$\mathrm{I}(X;Y|S=s) \le \mathrm{C}_s \tag{12.37}$$

with equality if X is uniform. Since the uniform distribution on X *simultaneously* achieves capacity for all subchannels, we see that

$$\mathrm{C} = \max_{P_X(\cdot)} \mathrm{I}(X;Y) = \sum_{s=1}^{\mathrm{S}} q_s \mathrm{C}_s. \tag{12.38}$$

□

Remark 12.14. Note that the proof of Theorem 12.13 only works because the same input distribution achieves capacity for all subchannels. In general, if a DMC is split into several not necessarily strongly symmetric subchannels with the same input and disjoint output alphabets, $\sum_s q_s \mathrm{C}_s$ is an *upper bound to capacity*. △

Remark 12.15. Note that in [CT06, Section 7.2], only strongly symmetric channels are considered, and there they are called "symmetric". △

Example 12.16. We return to the BEC of Figure 12.1. The capacities of the two strongly symmetric subchannels are $\mathrm{C}_1 = 1$ bit and $\mathrm{C}_2 = 0$. Hence,

$$\mathrm{C}_{\mathrm{BEC}} = (1-\delta)\mathrm{C}_1 + \delta \mathrm{C}_2 = 1 - \delta \text{ bits}. \tag{12.39}$$

This is quite intuitive: Since the BEC erases a fraction of δ of all transmitted bits, we cannot have hope to be able to transmit more than a fraction $1-\delta$ bits on average. ◊

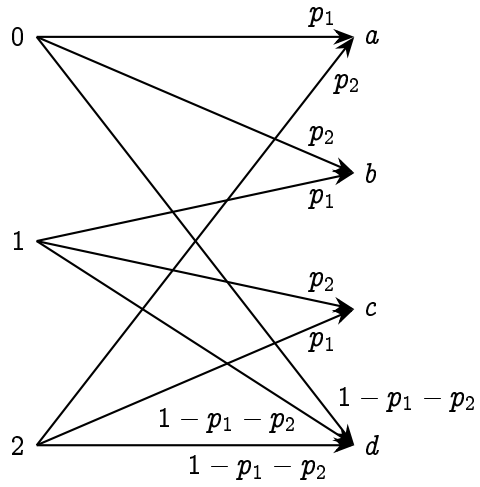


Figure 12.5: A weakly symmetric DMC.

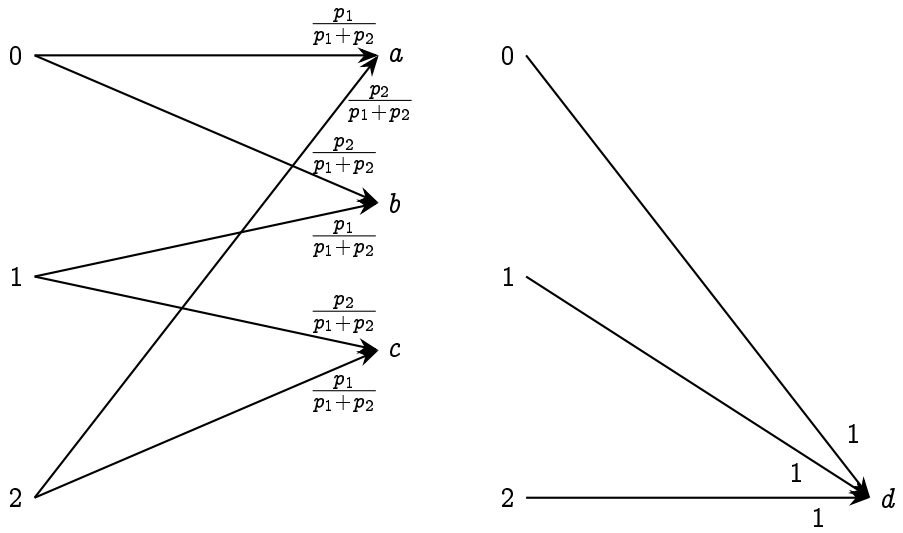


Figure 12.6: Two strongly symmetric subchannels.

Example 12.17. We would like to compute the capacity of the DMC shown in Figure 12.5.

Note that the DMC is not strongly symmetric because the focusing of the output letter d is different from the focusing of the other three output letters. However, since the dispersion is the same for all three input letters, we check whether we can split the channel into two strongly symmetric DMCs: According to the focusing we create two subchannels with $\mathcal{Y}_1 = \{a, b, c\}$ and $\mathcal{Y}_2 = \{d\}$.

We double-check and see that both subchannels are uniformly dispersive:

In the first subchannel each input has the probabilities $(p_1, p_2, 0, 0)$ leaving the node, and in the second subchannel, each input sees the probability $1-p_1-p_2$ leaving its node. Thus, the DMC of Figure 12.5 indeed is weakly symmetric.

We now compute the selection probabilities:

$$q_1 = \sum_{y\in\mathcal{Y}_1} P_{Y|X}(y|0) = p_1 + p_2, \tag{12.40}$$

$$q_2 = \sum_{y\in\mathcal{Y}_2} P_{Y|X}(y|0) = 1 - p_1 - p_2. \tag{12.41}$$

These selection probabilities are also used to normalize the probabilities, and we obtain the two subchannels shown in Figure 12.6. The capacity of the channel thus is

$$\begin{aligned} \mathrm{C} &= q_1\mathrm{C}_1 + q_2\mathrm{C}_2 && (12.42)\\ &= (p_1+p_2)\biggl(\log 3 + \frac{p_1}{p_1+p_2}\log\frac{p_1}{p_1+p_2} + \frac{p_2}{p_1+p_2}\log\frac{p_2}{p_1+p_2}\biggr)\\ &\quad + (1-p_1-p_2)\cdot 0 && (12.43)\\ &= (p_1+p_2)\biggl(\log 3 - \mathrm{H_b}\Bigl(\frac{p_1}{p_1+p_2}\Bigr)\biggr). && (12.44) \end{aligned}$$

◊

12.4 Mutual Information and Convexity

In this section we will show that mutual information is concave in the input distribution. This will be used in Section 12.5 when we apply the Karush–Kuhn–Tucker conditions (Theorem 9.11) to the problem of computing capacity. We introduce a new notation here that we will use in combination with our notation so far: We will denote the input distribution $P_X(\cdot)$ of a DMC by $Q(\cdot)$, and the conditional channel distribution $P_{Y|X}(\cdot|\cdot)$ will be denoted by $W(\cdot|\cdot)$. Moreover, whenever a DMC has finite alphabets, we will use a vector- and matrix-notation $\mathbf{Q}$ and W, respectively.

Theorem 12.18. Consider a DMC with input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$, and transition probability matrix W with components $W(y|x)$, $y\in\mathcal{Y}$, $x\in\mathcal{X}$. Let $\mathbf{Q} = (Q(1),\dots,Q(|\mathcal{X}|))^{\mathsf{T}}$ be an arbitrary input probability vector. Then

$$\mathrm{I}(X;Y) = \mathrm{I}(\mathbf{Q},\mathrm{W}) \triangleq \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} Q(x)W(y|x)\log\frac{W(y|x)}{\sum_{x'\in\mathcal{X}}Q(x')W(y|x')} \tag{12.45}$$

is a **concave function** in $\mathbf{Q}$ (for a fixed W) and a **convex function** in W (for a fixed $\mathbf{Q}$).

Proof: We start with the first claim. We fix W and let $\mathbf{Q}_0$ and $\mathbf{Q}_1$ be two input distributions. We want to show that

$$\theta\,\mathrm{I}(\mathbf{Q}_0,\mathrm{W}) + (1-\theta)\,\mathrm{I}(\mathbf{Q}_1,\mathrm{W}) \le \mathrm{I}(\theta\mathbf{Q}_0 + (1-\theta)\mathbf{Q}_1,\mathrm{W}). \tag{12.46}$$

To this end, we define Z to be a binary RV with $P_Z(0) = \theta$ and $P_Z(1) = 1-\theta$ and define X such that, conditionally on Z,

$$X \sim \begin{cases} \mathbf{Q}_0 & \text{if } Z = 0, \\ \mathbf{Q}_1 & \text{if } Z = 1. \end{cases} \tag{12.47}$$

Note that the unconditional PMF of X then is

$$\mathbf{Q} = P_Z(0)\mathbf{Q}_0 + P_Z(1)\mathbf{Q}_1 \tag{12.48}$$
$$= \theta\mathbf{Q}_0 + (1-\theta)\mathbf{Q}_1. \tag{12.49}$$

We can think of this like two channels in series: a first channel (that is part of our encoder) with input Z and output X, and a second channel (the DMC) with input X and output Y (see Figure 12.7). Hence, we have a Markov chain $Z \multimap X \multimap Y$, i.e.,

$$\mathrm{I}(Z;Y|X) = 0. \tag{12.50}$$

We get

$$\mathrm{I}(X;Y) = \mathrm{I}(X;Y) + \mathrm{I}(Z;Y|X) \tag{12.51}$$
$$= \mathrm{I}(X,Z;Y) \tag{12.52}$$
$$= \underbrace{\mathrm{I}(Z;Y)}_{\geq 0} + \mathrm{I}(X;Y|Z) \tag{12.53}$$
$$\geq \mathrm{I}(X;Y|Z), \tag{12.54}$$

where the inequality follows because mutual information cannot be negative. Since

$$\mathrm{I}(X;Y) = \mathrm{I}(\mathbf{Q},\mathsf{W}) = \mathrm{I}(\theta\mathbf{Q}_0 + (1-\theta)\mathbf{Q}_1, \mathsf{W}) \tag{12.55}$$

and

$$\mathrm{I}(X;Y|Z) = P_Z(0)\,\mathrm{I}(X;Y|Z=0) + P_Z(1)\,\mathrm{I}(X;Y|Z=1) \tag{12.56}$$
$$= \theta\,\mathrm{I}(\mathbf{Q}_0,\mathsf{W}) + (1-\theta)\,\mathrm{I}(\mathbf{Q}_1,\mathsf{W}), \tag{12.57}$$

(12.54) proves the concavity of $\mathrm{I}(\mathbf{Q},\mathsf{W})$ in $\mathbf{Q}$.

To prove the second statement, fix $\mathbf{Q}$ and let W_0 and W_1 be two conditional channel distributions. We want to show that

$$\theta\,\mathrm{I}(\mathbf{Q},\mathsf{W}_0) + (1-\theta)\,\mathrm{I}(\mathbf{Q},\mathsf{W}_1) \geq \mathrm{I}(\mathbf{Q},\theta\mathsf{W}_0 + (1-\theta)\mathsf{W}_1). \tag{12.58}$$

To this end, we define a new channel W to be a random combination of W_0 and W_1 as shown in Figure 12.8. The RV Z is the random switch inside this new channel, i.e., conditionally on Z,

$$\mathsf{W} \triangleq \begin{cases} \mathsf{W}_0 & \text{if } Z = 0, \\ \mathsf{W}_1 & \text{if } Z = 1. \end{cases} \tag{12.59}$$

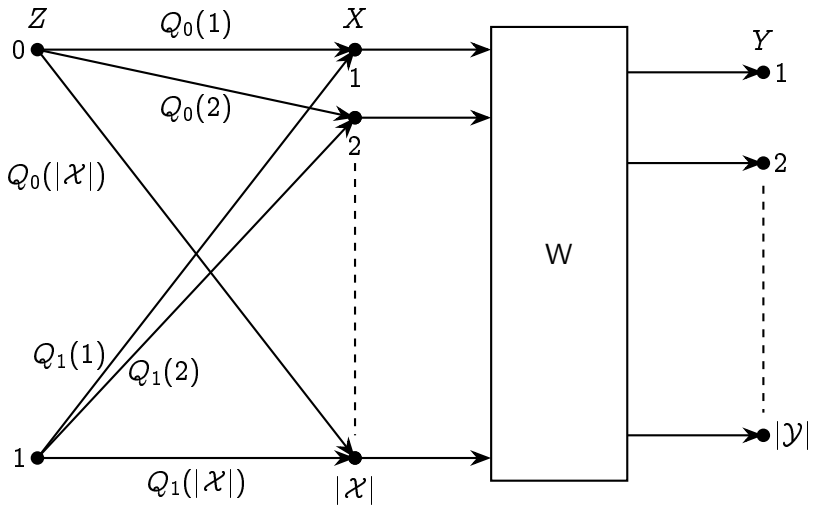


Figure 12.7: Mutual information is concave in the input. The encoder consists of random switch between two random distributions. Since such an encoder looks like a DMC with a binary input, the system looks like having two DMCs in series.

Note that the channel law of this new DMC W is

$$\mathsf{W} = P_Z(0)\mathsf{W}_0 + P_Z(1)\mathsf{W}_1 \tag{12.60}$$
$$= \theta\mathsf{W}_0 + (1-\theta)\mathsf{W}_1. \tag{12.61}$$

Also note that X (input) and Z (part of the new channel) are independent in this case. Therefore

$$\mathrm{I}(X;Y|Z) = \mathrm{I}(X;Y|Z) + \underbrace{\mathrm{I}(X;Z)}_{=0} \tag{12.62}$$
$$= \mathrm{I}(X;Y,Z) \tag{12.63}$$
$$= \mathrm{I}(X;Y) + \underbrace{\mathrm{I}(X;Z|Y)}_{\geq 0} \tag{12.64}$$
$$\geq \mathrm{I}(X;Y). \tag{12.65}$$

Since

$$\mathrm{I}(X;Y|Z) = P_Z(0)\,\mathrm{I}(X;Y|Z=0) + P_Z(1)\,\mathrm{I}(X;Y|Z=1) \tag{12.66}$$
$$= \theta\,\mathrm{I}(\mathbf{Q},\mathsf{W}_0) + (1-\theta)\,\mathrm{I}(\mathbf{Q},\mathsf{W}_1) \tag{12.67}$$

and

$$\mathrm{I}(X;Y) = \mathrm{I}(\mathbf{Q},\mathsf{W}) = \mathrm{I}(\mathbf{Q},\theta\mathsf{W}_0 + (1-\theta)\mathsf{W}_1), \tag{12.68}$$

(12.65) proves the convexity of $\mathrm{I}(\mathbf{Q},\mathsf{W})$ in W. □

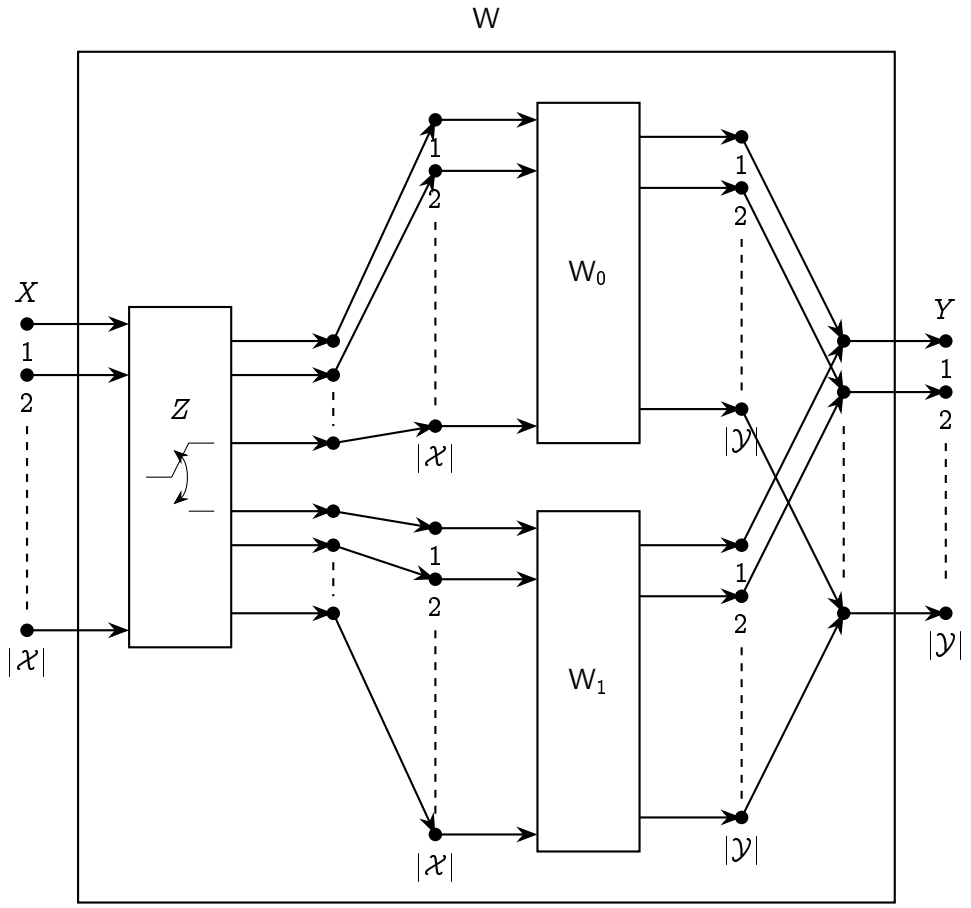


Figure 12.8: Mutual information is convex in the channel law. We have two parallel channels with a random switch.

12.5 Karush–Kuhn–Tucker Conditions

Recall that in Section 9.3 we have derived the Karush–Kuhn–Tucker (KKT) conditions to describe the optimal solution of an optimization problem involving the maximization of a concave function over a probability vector. We note that the computation of capacity is exactly such a problem:

$$\mathrm{C} = \max_{\mathbf{Q}} \mathrm{I}(\mathbf{Q}, \mathrm{W}) \tag{12.69}$$

where $\mathbf{Q}$ is a probability vector and where $\mathrm{I}(\mathbf{Q}, \mathrm{W})$ is concave in $\mathbf{Q}$.

Hence, we can use the KKT conditions to describe the capacity-achieving input distribution.

Theorem 12.19 (KKT Conditions for Capacity).
A set of necessary and sufficient conditions on an input probability vector $\mathbf{Q}$ to achieve the capacity of a DMC $(\mathcal{X}, \mathcal{Y}, \mathsf{W})$ is that for some number C,

$$\sum_y W(y|x) \log \frac{W(y|x)}{\sum_{x'} Q(x')W(y|x')} = \mathsf{C} \qquad \forall\, x \text{ with } Q(x) > 0, \tag{12.70}$$

$$\sum_y W(y|x) \log \frac{W(y|x)}{\sum_{x'} Q(x')W(y|x')} \le \mathsf{C} \qquad \forall\, x \text{ with } Q(x) = 0. \tag{12.71}$$

Moreover, the number C is the capacity of the channel.

More concisely, the KKT conditions can also be written as

$$\mathscr{D}(W(\cdot|x) \,\|\, (QW)(\cdot)) \begin{cases} = \mathsf{C} & \forall\, x \text{ with } Q(x) > 0, \\ \le \mathsf{C} & \forall\, x \text{ with } Q(x) = 0. \end{cases} \tag{12.72}$$

Proof: From Theorem 12.18 we know that the mutual information $\mathrm{I}(\mathbf{Q}, \mathsf{W})$ is concave in the input probability vector $\mathbf{Q}$. We are therefore allowed to apply the KKT conditions (Theorem 9.11). Hence, we need to compute the derivatives of $\mathrm{I}(\mathbf{Q}, \mathsf{W})$. To this end, note that $\mathrm{I}(\mathbf{Q}, \mathsf{W})$ can be written as follows:

$$\begin{aligned} \mathrm{I}(\mathbf{Q}, \mathsf{W}) &= \sum_x \sum_y Q(x)W(y|x) \log \frac{W(y|x)}{\sum_{x'} Q(x')W(y|x')} && (12.73) \\ &= \sum_{\substack{x \\ x \neq \tilde{x}}} \sum_y Q(x)W(y|x) \log \frac{W(y|x)}{\sum_{x'} Q(x')W(y|x')} \\ &\quad + \sum_y Q(\tilde{x})W(y|\tilde{x}) \log \frac{W(y|\tilde{x})}{\sum_{x'} Q(x')W(y|x')}, && (12.74) \end{aligned}$$

where where have split up one summation into $x = \tilde{x}$ and $x \neq \tilde{x}$ for $\tilde{x}$ being some fixed value. Then

$$\begin{aligned} &\frac{\partial}{\partial Q(\tilde{x})} \mathrm{I}(\mathbf{Q}, \mathsf{W}) \\ &\quad = -\sum_{\substack{x \\ x \neq \tilde{x}}} \sum_y Q(x)W(y|x) \cdot \frac{\sum_{x'} Q(x')W(y|x')}{W(y|x)} \cdot \frac{W(y|x)}{\left(\sum_{x'} Q(x')W(y|x')\right)^2} \\ &\qquad \cdot W(y|\tilde{x}) \log e \\ &\qquad + \sum_y W(y|\tilde{x}) \log \frac{W(y|\tilde{x})}{\sum_{x'} Q(x')W(y|x')} \\ &\qquad - \sum_y Q(\tilde{x})W(y|\tilde{x}) \cdot \frac{\sum_{x'} Q(x')W(y|x')}{W(y|\tilde{x})} \cdot \frac{W(y|\tilde{x})}{\left(\sum_{x'} Q(x')W(y|x')\right)^2} \\ &\qquad \cdot W(y|\tilde{x}) \log e \end{aligned} \tag{12.75}$$

$$= -\sum_x \sum_y Q(x)W(y|x) \cdot \frac{\sum_{x'} Q(x')W(y|x')}{W(y|x)} \cdot \frac{W(y|x)}{\left(\sum_{x'} Q(x')W(y|x')\right)^2} \cdot W(y|\tilde{x}) \log e + \sum_y W(y|\tilde{x}) \log \frac{W(y|\tilde{x})}{\sum_{x'} Q(x')W(y|x')} \tag{12.76}$$

$$= -\sum_y \sum_x Q(x)W(y|x) \cdot \frac{1}{\sum_{x'} Q(x')W(y|x')} \cdot W(y|\tilde{x}) \log e + \sum_y W(y|\tilde{x}) \log \frac{W(y|\tilde{x})}{\sum_{x'} Q(x')W(y|x')} \tag{12.77}$$

$$= -\sum_y \frac{\sum_x Q(x)W(y|x)}{\sum_{x'} Q(x')W(y|x')} \cdot W(y|\tilde{x}) \log e + \sum_y W(y|\tilde{x}) \log \frac{W(y|\tilde{x})}{\sum_{x'} Q(x')W(y|x')} \tag{12.78}$$

$$= -\log e + \sum_y W(y|\tilde{x}) \log \frac{W(y|\tilde{x})}{\sum_{x'} Q(x')W(y|x')} \begin{cases} = \lambda & \text{if } Q(\tilde{x}) > 0, \\ \le \lambda & \text{if } Q(\tilde{x}) = 0, \end{cases} \tag{12.79}$$

where in (12.79) we have applied the Karush–Kuhn–Tucker conditions with the Lagrange multiplier λ. Taking the constant $-\log e$ to the other side and defining $c \triangleq \lambda + \log e$ now finishes the proof of the first part.

It remains to show that c is the capacity C. To this end, take an optimal $\mathbf{Q}^*$ and average both sides of (12.79):

$$\sum_x Q^*(x) \sum_y W(y|x) \log \frac{W(y|x)}{\sum_{x'} Q^*(x')W(y|x')} = \sum_x Q^*(x) c \tag{12.80}$$

i.e.,

$$\mathrm{I}(\mathbf{Q}^*, \mathsf{W}) = c. \tag{12.81}$$

But since $\mathrm{I}(\mathbf{Q}^*, \mathsf{W}) = \mathrm{C}$, we see that c must be the capacity. □

Unfortunately, (12.70) and (12.71) are still difficult to solve. They are most useful in checking whether a given $\mathbf{Q}$ is optimal or not: We make a guess of how the optimal input distribution might look like. Then we check whether (12.70) and (12.71) are satisfied. If yes, then we are done and we know the capacity of the channel. If not, then we at least know one example of how the capacity-achieving input distribution does not look like...

Example 12.20. We again consider the binary symmetric channel (BSC). From symmetry we guess that the capacity-achieving input distribution should be uniform $Q(0) = Q(1) = \frac{1}{2}$. To check whether we guess correctly, we apply Theorem 12.19: Since $Q(0) = Q(1) > 0$ we only need (12.70). For $x = 0$ we

get

$$\begin{aligned}
&W(0|0)\log\frac{W(0|0)}{\frac{1}{2}W(0|0)+\frac{1}{2}W(0|1)}+W(1|0)\log\frac{W(1|0)}{\frac{1}{2}W(1|0)+\frac{1}{2}W(1|1)}\\
&\quad=(1-\epsilon)\log\frac{1-\epsilon}{\frac{1}{2}(1-\epsilon)+\frac{1}{2}\epsilon}+\epsilon\log\frac{\epsilon}{\frac{1}{2}(1-\epsilon)+\frac{1}{2}\epsilon} && (12.82)\\
&\quad=(1-\epsilon)\log(1-\epsilon)+\epsilon\log\epsilon-(1-\epsilon+\epsilon)\log\Big(\frac{1}{2}(1-\epsilon)+\frac{1}{2}\epsilon\Big) && (12.83)\\
&\quad=-\mathrm{H_b}(\epsilon)+\log 2 && (12.84)\\
&\quad=1-\mathrm{H_b}(\epsilon)\text{ bits.} && (12.85)
\end{aligned}$$

Similarly, for $x=1$:

$$\begin{aligned}
&W(0|1)\log\frac{W(0|1)}{\frac{1}{2}W(0|0)+\frac{1}{2}W(0|1)}+W(1|1)\log\frac{W(1|1)}{\frac{1}{2}W(1|0)+\frac{1}{2}W(1|1)}\\
&\quad=1-\mathrm{H_b}(\epsilon)\text{ bits.} && (12.86)
\end{aligned}$$

Since both expressions (12.85) and (12.86) are the same, our guess has been correct and the capacity is $\mathrm{C}=1-\mathrm{H_b}(\epsilon)$ bits. ◊

Chapter 13

Convolutional Codes

After having studied rather fundamental properties of channels and channel coding schemes, in this chapter we finally give a practical example of how one could try to actually build a channel coding scheme. Note that while we know that there must exist many schemes that work reliably, in practice we have the additional problem of the limited computational power available both at transmitter and receiver. This complicates the design of a channel coding scheme considerably.

The code that will be introduced in this chapter is called *convolutional code* or *trellis code*. It is a quite powerful scheme that is often used in practice. We will explain it using a particular example that will hopefully shed light on the basic ideas behind any such design.

13.1 Convolutional Encoder of a Trellis Code

We assume that we have a message source that generates a sequence of *information digits* $\{U_k\}$. We will assume that the information digits are binary, i.e., *information bits*, even though strictly speaking this is not necessary. But in practice this is the case anyway. These information bits are fed into a *convolutional encoder*. As an example consider the encoder shown in Figure 13.1. This encoder is a finite state machine that has (a finite) memory: Its current output depends on the current input and on a certain number of past inputs. In the example of Figure 13.1 its memory is 2 bits, i.e., it contains a shift-register that keeps stored the values of the last two information bits. Moreover, the encoder has several modulo-2 adders.

The output of the encoder are the *codeword bits* that will be then transmitted over the channel. In our example for every information bit, two codeword bits are generated. Hence the *encoder rate* is

$$R_t = \frac{1}{2} \text{ bits.} \tag{13.1}$$

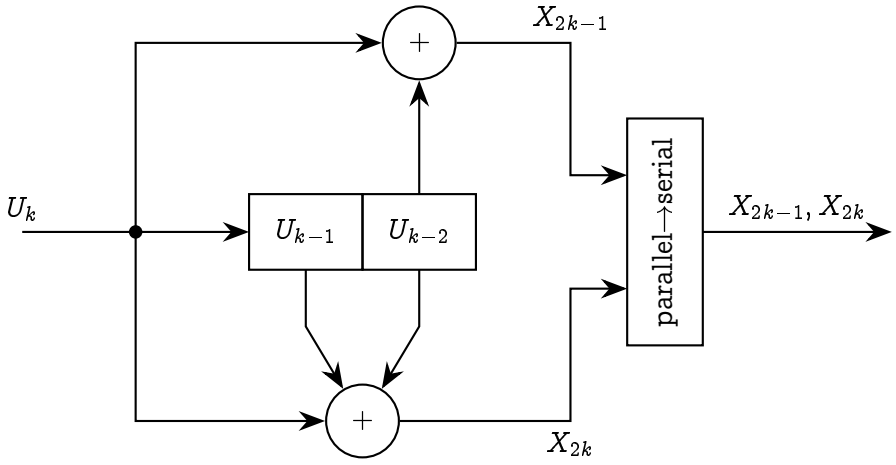


Figure 13.1: Example of a convolutional encoder.

In general the encoder can take n_i information bits to generate n_c codeword bits yielding an encoder rate of $R_t = \frac{n_i}{n_c}$ bits.

To make sure that the outcome of the encoder is a deterministic function of the sequence of input bits, we ask the memory cells of the encoder to contain zeros at the beginning of the encoding process. Moreover, once L_t information bits have been encoded, we stop the information bit sequence and will feed T dummy zero-bits as inputs instead, where T is chosen to be equal to the memory size of the encoder. These dummy bits will make sure that the state of the memory cells are turned back to zero. In the case of the encoder given in Figure 13.1 we have $T = 2$.

Example 13.1. For the encoder given in Figure 13.1 let $L_t = 3$ and $T = 2$. How many codeword bits do we generate? Answer: $n = 2 \cdot (3 + 2) = 10$ codeword bits. So our codewords have length $n = 10$. How many different messages do we encode in this way? Since we use $L_t = 3$ information bits, we have $2^{L_t} = 2^3 = 8$ possible messages. So in total, we have an actual coding rate of

$$R = \frac{\log_2(\#\text{messages})}{n} = \frac{\log_2 8}{10} = \frac{3}{10} \text{ bits/channel use.} \tag{13.2}$$

◊

In general we get the following actual coding rate:

$$R = \frac{L_t}{L_t + T} \cdot R_t. \tag{13.3}$$

These eight codewords can be described with the help of a binary tree. At each node we either turn upwards (corresponding to an input bit 1) or downwards (corresponding to an input bit 0). Since the output depends on the current state of the memory, we include the state in the graphical depiction of

the node. The branches will then be labeled according to the corresponding output bits. See Figure 13.2. Any code that can be described by a tree is called *tree code*. Obviously a tree code is a special case of a *block code* as all codewords have the same length.

Note that at depth 3 and further in the tree, the different states start to repeat itself and therefore (since the output bits only depend on the state and the input) also the output bits repeat. Hence there is no real need to draw all states many times, but instead we can collapse the tree into a so-called *trellis*; see Figure 13.3.

So, our tree code actually also is a *trellis code*. Note that every trellis code by definition is a tree code, but not necessarily vice versa. Moreover, note that we end up in the all-zero state because of our tail of $\mathsf{T} = 2$ dummy bits.

13.2 Decoder of a Trellis Code

A code that is supposed to be useful in practice needs to have a decoder with reasonable performance that can be implemented with reasonable efficiency. Luckily, for our trellis code, it is possible to implement not only a reasonable good decoder, but actually the optimal *maximum likelihood (ML) decoder* (see Section 11.3)!

Before we explain the decoder, we quickly give the following definitions.

Definition 13.2. The *Hamming distance* $d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$ between two sequences $\mathbf{x}$ and $\mathbf{y}$ is defined as the number of positions where $\mathbf{x}$ differ from $\mathbf{y}$.

The *Hamming weight* $w_{\mathrm{H}}(\mathbf{x})$ of a sequence $\mathbf{x}$ is defined as the number of nonzero positions of $\mathbf{x}$.

Hence, we see that

$$d_{\mathrm{H}}(\mathbf{x},\mathbf{y}) = w_{\mathrm{H}}(\mathbf{x}-\mathbf{y}). \tag{13.4}$$

To explain how the ML decoder of a trellis code works, we again use a concrete example: Let us assume that we use the trellis code of Figure 13.3 over a BSC with crossover probability $0 \le \epsilon < \frac{1}{2}$ as shown in Figure 13.4. For a given codeword $\mathbf{x}$, the probability of a received vector $\mathbf{y}$ is then given as

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = (1-\epsilon)^{n-d_{\mathrm{H}}(\mathbf{x},\mathbf{y})} \cdot \epsilon^{d_{\mathrm{H}}(\mathbf{x},\mathbf{y})} \tag{13.5}$$

$$= (1-\epsilon)^n \left(\frac{\epsilon}{1-\epsilon}\right)^{d_{\mathrm{H}}(\mathbf{x},\mathbf{y})}, \tag{13.6}$$

where $d_{\mathrm{H}}(\mathbf{x},\mathbf{y})$ denotes the *Hamming distance* defined above. Since we want to implement an ML decoder, we need to maximize this *a posteriori* probability given in (13.6):

$$\max_{\mathbf{x}\in\mathscr{C}} P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \max_{\mathbf{x}\in\mathscr{C}} \left((1-\epsilon)^n \left(\frac{\epsilon}{1-\epsilon}\right)^{d_{\mathrm{H}}(\mathbf{x},\mathbf{y})} \right). \tag{13.7}$$

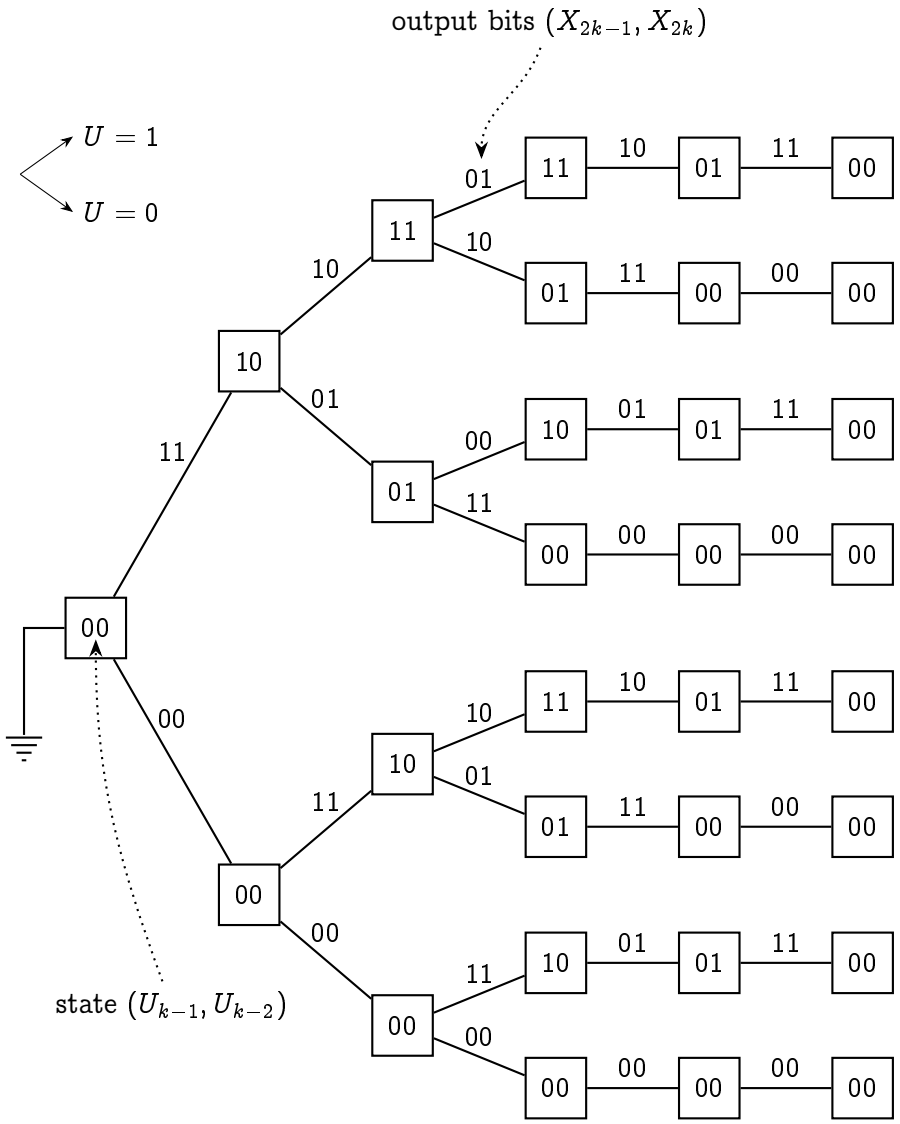


Figure 13.2: Tree depicting all codewords of the convolutional code: The bits on the branches form the codewords. The labels inside the boxes correspond to the current *state* of the encoder, i.e., the contents of the memory cells. The input bits are not depicted directly, but are shown indirectly as the choice of the path: Going upwards corresponds to an input bit 1, going downwards to 0. The last two input bits are by definition 0, so we only show one path there.

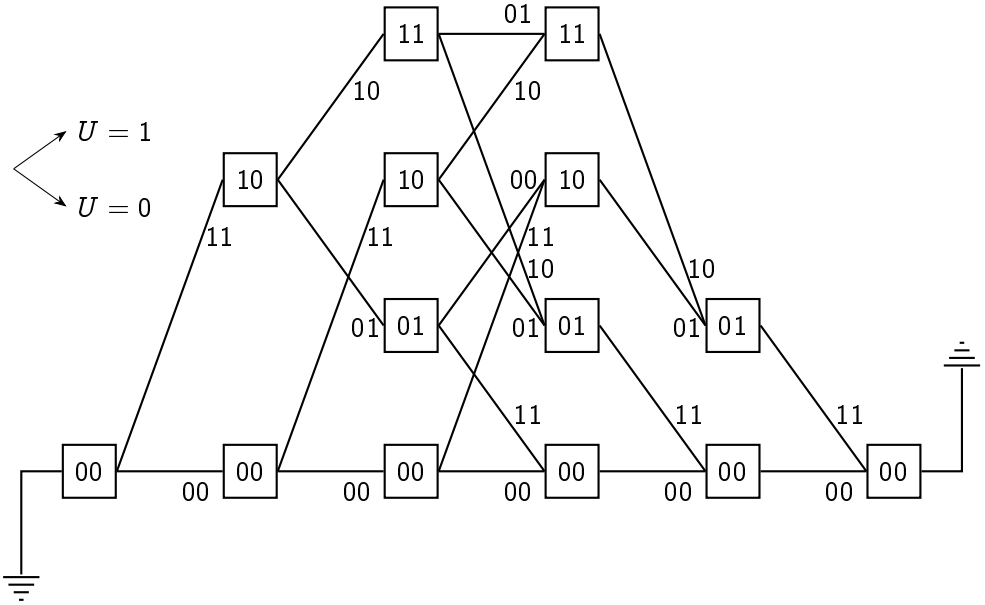


Figure 13.3: Trellis depicting all codewords of the convolutional code. A trellis is like a tree, but with identical states being collapsed into one state only. The final state is called *toor*, the inverse of *root*.

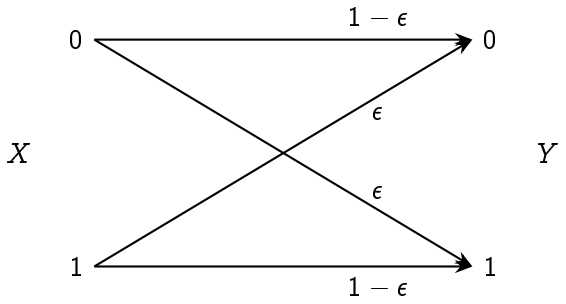


Figure 13.4: Binary symmetric channel (BSC).

Note that because we have assumed $\epsilon < \frac{1}{2}$, we see that $\frac{\epsilon}{1-\epsilon} < 1$, i.e., we achieve the maximum if we minimize $d_{\mathrm{H}}(\mathbf{x}, \mathbf{y})$.

So we can describe the ML decoder of a trellis code as follows:

> For a trellis code that is used on a BSC, an optimal decoder needs to find the path in the trellis (see Figure 13.3) that *maximizes the number of positions where the received vector* $\mathbf{y}$ *and the output bits of the path agree.*

This can be done in the following way: Given a received vector $\mathbf{y}$, we go through the trellis and write over each state the number of positions that agree with $\mathbf{y}$. Once we reach depth 4, this number will not be unique anymore because there are two paths arriving at each state. In this case we will compare the number we will get along both incoming paths and discard the smaller one because we know that whatever the optimal solution is, it definitely cannot be the path with the smaller (intermediate) number! We mark the discarded path with a cross.

If both incoming path result in an identical value, it doesn't matter which path to take and this might result in a nonunique optimal solution.

Once we reach the *toor* (the final state), we only need to go backwards through the trellis always following the not-discarded path.

Example 13.3. If we receive $\mathbf{y} = (0\,1\,0\,1\,0\,1\,0\,1\,1\,1)$, what is the ML decoding decision?

In Figure 13.5 we show the trellis including the number of agreeing bits written above each state and the discarded paths. We see that the optimal decoding yields

$$\hat{\mathbf{x}} = (1\,1\,0\,1\,0\,0\,0\,1\,1\,1), \tag{13.8}$$

which corresponds to an input vector $\hat{\mathbf{u}} = (1\,0\,1)$. ◊

The algorithm that we have described just now is called *Viterbi Algorithm*. It is an efficient implementation of an ML-decoder for a trellis code.

In our example of a BSC we have seen that the Hamming distance was the right *metric* to be used in the Viterbi Algorithm. But how do we choose a metric for a general DMC with binary input? To find out, let us list some properties such a metric should have:

- The metric should be additive (in $k = 1, \ldots, n$).
- The largest metric must belong to the path that an ML-decoder would choose.

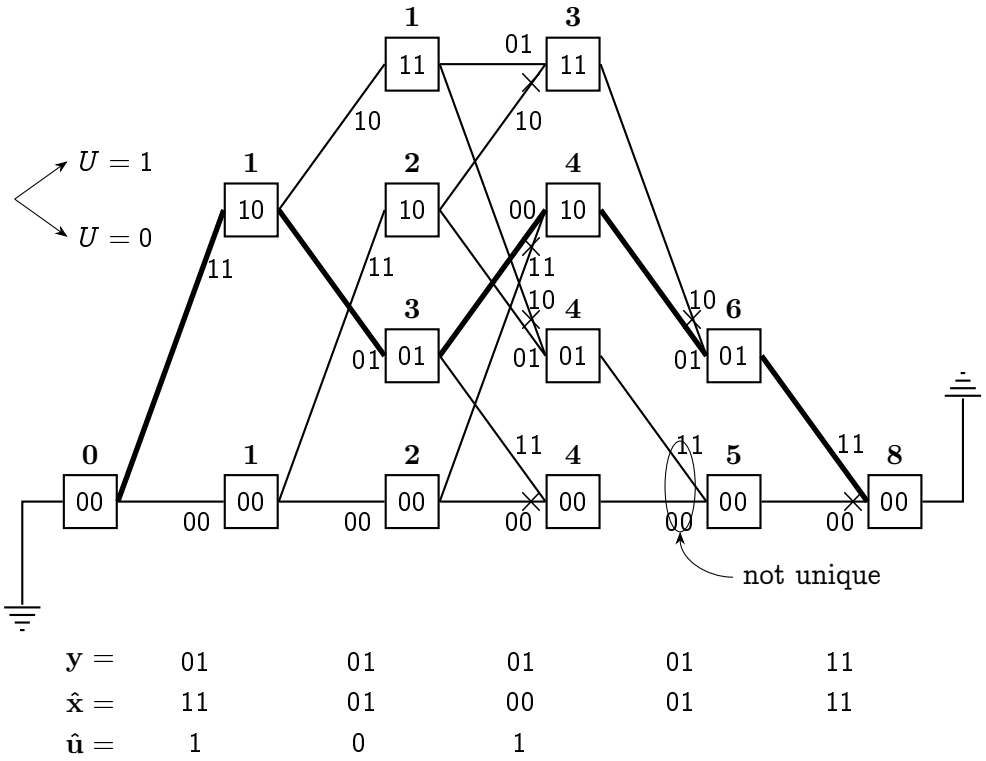


Figure 13.5: Decoding of a trellis code.

Note that for every $\mathbf{y}$, an ML-decoder chooses a $\mathbf{x}$ that maximizes

$$P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \prod_{k=1}^{n} P_{Y|X}(y_k|x_k) \tag{13.9}$$

where the equality follows because we assume a DMC without feedback. Unfortunately, this metric is not additive. To fix this, we take the logarithm:

$$\log P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \sum_{k=1}^{n} \log P_{Y|X}(y_k|x_k). \tag{13.10}$$

Hence, we could use

$$\log P_{Y|X}(y_k|x_k) \tag{13.11}$$

as metric. The only problem is that such a metric will yield very annoying numbers, making it very inconvenient for humans to compute. So, in order to simplify our life, we ask for the following requirements in addition to the two properties given above:

- For computational simplicity, the metric should consist of small nonnegative numbers.

- Moreover, the metric should be 0 as often as possible.

To find a way to change our metric (13.11) to satisfy these two additional constraints, note that

$$\operatorname*{argmax}_{\mathbf{x}}\Big\{\log P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})\Big\} = \operatorname*{argmax}_{\mathbf{x}}\left\{\alpha \log P_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) + \alpha \sum_{k=1}^{n} f(y_k)\right\} \tag{13.12}$$

for any $\alpha > 0$ and any function $f(\cdot)$ (note that $f(y_k)$ does not depend on $\mathbf{x}$!). Hence, instead of (13.11) we could use

$$d(x_k, y_k) = \alpha \log P_{Y|X}(y_k|x_k) + \alpha f(y_k) \tag{13.13}$$

as metric for an arbitrary choice of $\alpha > 0$ and $f(\cdot)$.

To make sure that $d(x_k, y_k)$ is nonnegative and equals to zero as often as possible, we choose

$$f(y_k) \triangleq -\log\Big(\min_x P_{Y|X}(y_k|x)\Big). \tag{13.14}$$

Finally, we choose α such that the different possible values of the metric are (almost) integers.

Example 13.4. Consider the binary symmetric erasure channel given in Figure 13.6. For this channel we have

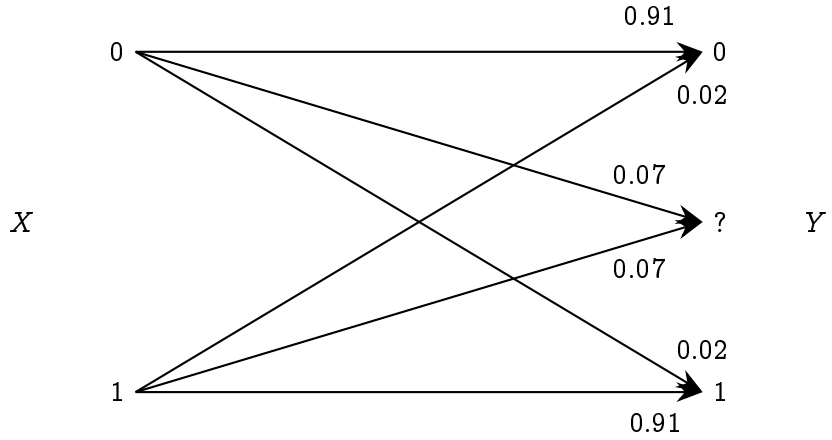


Figure 13.6: Binary symmetric erasure channel (BSEC).

$$\min_x P_{Y|X}(y|x) = \begin{cases} 0.02 & \text{for } y = 0, \\ 0.07 & \text{for } y = ?, \\ 0.02 & \text{for } y = 1. \end{cases} \tag{13.15}$$

Hence we get the (so far unscaled) metric shown in Table 13.7. We see that

Table 13.7: Viterbi metric for a BSEC, unscaled.

	$y=0$	$y=?$	$y=1$
$x=0$	$\log 0.91 - \log 0.02$ ≈ 5.50	$\log 0.07 - \log 0.07$ $= 0$	$\log 0.02 - \log 0.02$ $= 0$
$x=1$	$\log 0.02 - \log 0.02$ $= 0$	$\log 0.07 - \log 0.07$ $= 0$	$\log 0.91 - \log 0.02$ ≈ 5.50

Table 13.8: Viterbi metric for a BSEC.

	$y=0$	$y=?$	$y=1$
$x=0$	1	0	0
$x=1$	0	0	1

most entries are zero. The only two entries that are not zero, happen to have the same value, so we choose the scaling factor $\alpha = \frac{1}{5.50}$ and get our final metric shown in Table 13.8.

We now use the convolutional code of Example 13.3 on the BSEC of Figure 13.6 and assume that we receive $\mathbf{y} = (0\,1\,?\,1\,1\,?\,0\,1\,1\,1)$. Applying the Viterbi algorithm to the trellis of Figure 13.3 and using the metric of Table 13.8, we then obtain the optimal decoding shown in Figure 13.9. We see that the decoding in this case turns out to be not unique: We find two possible optimal solutions. ♢

13.3 Quality of a Trellis Code

Suppose you have come up with a certain design of a convolutional encoder. Now your boss wants to know how good your coding scheme is (of course when using the encoder together with a Viterbi decoder). One approach would be to implement the system and let it run for long time and then compute the average error probability. However, from a designer's point of view, it would be much more practical to have a quick and elegant way of getting the performance (or at least a bound on the performance) of the system without having to build it. This is the goal of this section: We will derive an upper bound on the bit error probability or *bit error rate* (BER) that can be easily computed using nothing but the design plan of our convolutional encoder and the specification of the channel.

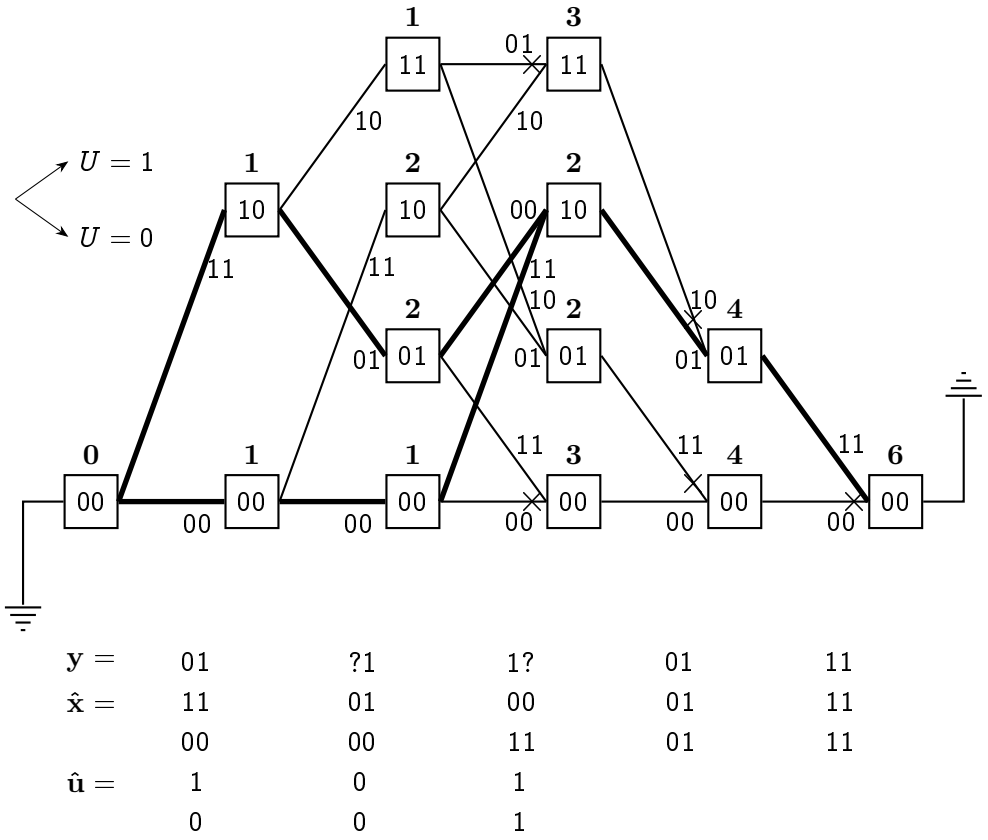


Figure 13.9: Viterbi decoding of convolutional code over BSEC.

While at the end the computation of this elegant bound is easy, the derivation of it is quite long as we need to make quite a few excursions to attain some necessary auxiliary results. We start with the idea of a *detour* in a trellis.

13.3.1 Detours in a Trellis

The performance of a trellis code depends on how "close" two different codewords are: If they are very close, then it is much more likely that the decoder confuses them due to the distortion caused by the channel. Since every codeword corresponds to a path in the trellis, we need to consider the "distance" of different paths.

To this end, assume for the moment that we transmit the all-zero codeword (which is a possible codeword of any trellis code!). If the decoder makes a mistake, the Viterbi decoder will make a *detour* in the trellis, i.e., it leaves the correct path and follows partially a wrong path.

To understand all possible detours that could occur, assume for the moment that the trellis goes on forever (i.e., assume $L_t = \infty$). In this case we

can describe all possible detours by fixing the starting point of the detour to node 1.

Definition 13.5. A *detour* is a path through the trellis starting at node 1 and ending as soon as we are back to the all-zero state.

An example of a detour is shown in Figure 13.10.

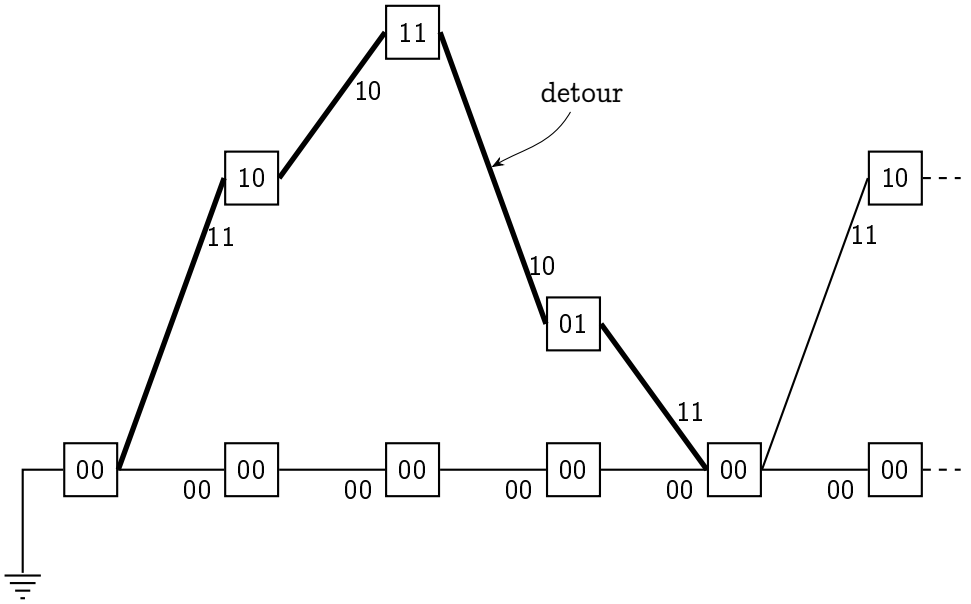


Figure 13.10: Detour in a trellis: The detour ends as soon as we get back to the correct path (the all-zero path in this case) for the first time, even if we will leave the correct path right again.

For a given detour the important questions are firstly how many bits are different in comparison to the correct path, and secondly how many information bit errors do occur along the way. Ideally, a detour has a large Hamming distance to the correct path.

Definition 13.6 (Counting Detours). For the infinite trellis of a given convolutional encoder, we define the parameter $a(d,i)$ to denote the number of detours starting at node 1 that have a Hamming distance d from the correct (all-zero) path and contain i information bit errors.

Example 13.7. Considering the convolutional encoder of Figure 13.1, it is not hard to see (compare with Figure 13.3) that

$$a(d,i) = 0, \qquad \text{for } d < 5 \text{ and all } i, \tag{13.16a}$$

$$a(5,1) = 1, \qquad \text{and} \tag{13.16b}$$

$$a(6,2) = 2. \tag{13.16c}$$

However, it can become rather tiring to try to find all possible detours by hand. ◊

In Section 13.3.2 we will find a way of deriving $a(d,i)$ graphically. Beforehand, however, we should investigate what happens if the correct path is not the all-zero path. It will turn out that $a(d,i)$ does not change irrespective which path through the trellis is assumed as the correct path! Concretely, we have the following lemma.

Lemma 13.8. Consider the $\mathrm{L_t} = \infty$ trellis for a given convolutional encoder. For any correct path and any integer j, the number of wrong paths beginning at node j that have Hamming distance d from the corresponding segment of the correct path and contain i information bits errors, is equal to $a(d,i)$. For every finite $\mathrm{L_t}$, $a(d,i)$ is an upper bound on this number.

Proof: Firstly note that in the infinite trellis, it does not matter where a detour starts, as from every node, the "remainder" of the trellis looks the same. So without loss of generality we can assume that the deviation from the correct path starts at node 1.

For an information sequence $\mathbf{u}$ denote by $g(\mathbf{u})$ the corresponding codeword generated by the convolutional encoder.

If $\mathbf{0}$ is the information sequence of the correct path, let $\tilde{\mathbf{u}}$ be the information sequence corresponding to a wrongly decoded codeword when making a detour beginning at node 1 (hence, $\tilde{\mathbf{u}}$ has a 1 at the first position!). Let

$$i = d_{\mathrm{H}}(\mathbf{0}, \tilde{\mathbf{u}}) = w_{\mathrm{H}}(\tilde{\mathbf{u}}) \tag{13.17}$$

be the number of bit errors caused by this wrong decoding and let

$$d = d_{\mathrm{H}}(g(\mathbf{0}), g(\tilde{\mathbf{u}})) = d_{\mathrm{H}}(\mathbf{0}, g(\tilde{\mathbf{u}})) = w_{\mathrm{H}}(g(\tilde{\mathbf{u}})) \tag{13.18}$$

be the Hamming distance between the two codewords.

Now, if we take a general information sequence $\mathbf{u}$ with corresponding codeword $g(\mathbf{u})$ as the correct path, let $\mathbf{u}'$ be the information sequence corresponding to a wrongly decoded codeword when making a deviation from the correct path beginning at node 1 (hence, $\mathbf{u}$ and $\mathbf{u}'$ differ at the first position!). Now, the number of information bit errors is

$$i = d_{\mathrm{H}}(\mathbf{u}, \mathbf{u}') = w_{\mathrm{H}}(\mathbf{u} \oplus \mathbf{u}') \tag{13.19}$$

and the Hamming distance between the two codeword is

$$\begin{aligned} d &= d_{\mathrm{H}}(g(\mathbf{u}), g(\mathbf{u}')) && (13.20)\\ &= d_{\mathrm{H}}(g(\mathbf{u}) \oplus g(\mathbf{u}), g(\mathbf{u}') \oplus g(\mathbf{u})) && (13.21)\\ &= d_{\mathrm{H}}(\mathbf{0}, g(\mathbf{u}') \oplus g(\mathbf{u})) && (13.22)\\ &= d_{\mathrm{H}}(\mathbf{0}, g(\mathbf{u}' \oplus \mathbf{u})) && (13.23)\\ &= w_{\mathrm{H}}(g(\mathbf{u}' \oplus \mathbf{u})). && (13.24) \end{aligned}$$

Here, in (13.21) we use the fact that adding the same sequence to both codewords will not change the Hamming distance; and in (13.23) we use that any convolutional encoder is linear in the sense that

$$g(\mathbf{u}' \oplus \mathbf{u}) = g(\mathbf{u}') \oplus g(\mathbf{u}). \tag{13.25}$$

Comparing (13.19) with (13.17) and (13.24) with (13.18), we see that for every given i, d, $\mathbf{u}$ and $\mathbf{u}'$, we can find an $\tilde{\mathbf{u}}$ (i.e., $\tilde{\mathbf{u}} = \mathbf{u}' \oplus \mathbf{u}$) such that i and d remain the same, but $\mathbf{u}$ is changed to $\mathbf{0}$. Hence, we can conclude that the number of deviations from a correct path $\mathbf{u}$ is the same as the number of detours (from a correct path $\mathbf{0}$), and therefore that $a(d,i)$ denotes the number of deviations from the correct path with i information bit errors and Hamming distance d for an arbitrary path $\mathbf{u}$. □

13.3.2 Counting Detours: Signalflowgraphs

As we now have understood the significance of $a(d,i)$, we would next like to try to find a way on how to compute it. The trick is to look at the convolutional encoder using a *state-transition diagram* that depicts all possible states and shows how the encoder swaps between them. For the example of the encoder of Figure 13.1 we have the state-transition diagram shown in Figure 13.11.

Now consider again the case when the all-zero codeword is the correct path. Then any detour starts from the all-zero state and will end in the all-zero state. Hence, we have the idea to break the all-zero state up into two states, a starting state and an ending state, and to transform the state-transition diagram into a *signalflowgraph*, where every state is transformed into a node and every transition into a directed edge; see Figure 13.12.

Moreover, each edge has assigned some multiplicative factor, where a term I denotes that an information bit error occurs when we pass along the corresponding edge, and a term D describes a codeword bit error caused by the detour that follows along this edge. If we take the product of all weight factors along a certain detour, then we will get a total factor $\mathrm{I}^i\mathrm{D}^d$, where the exponent i denotes the total number of information bit errors along this detour, and the exponent d denotes the total Hamming distance between the detour and the correct path.

Example 13.9. If we take the detour 'start→a→b→b→c→end', we get the product

$$\mathrm{ID}^2 \cdot \mathrm{ID} \cdot \mathrm{ID} \cdot \mathrm{D} \cdot \mathrm{D}^2 = \mathrm{I}^3\mathrm{D}^7, \tag{13.26}$$

which means that we have made three information bit errors and have a Hamming distance 7 to the correct all-zero path. ◊

Before we now learn how we can make use of this signalflowgraph, let us quickly repeat the basic definitions of signalflowgraphs.

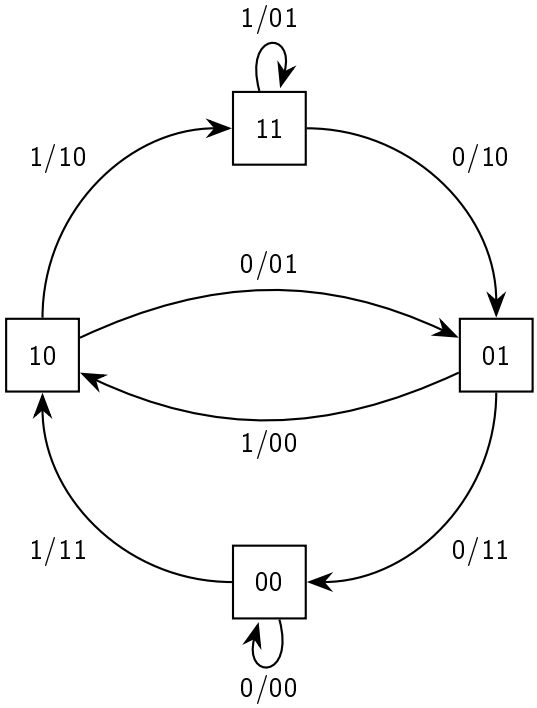


Figure 13.11: State-transition diagram of the encoder given in Figure 13.1. Each edge u/cc is labeled with the information bit u necessary in order to change state along this edge and the corresponding codeword bits cc generated by this transition.

Definition 13.10. A *signalflowgraph* consists of nodes and directed edges between nodes, where each edge has a weight factor. We define the following:

- A node i is *directly connected* with a node j if there exists a directed edge pointing from i to j.
- A node i is *connected* with node j if we can find a sequence of nodes ℓ_ν such that i is directly connected with ℓ_1; ℓ_1 is directly connected with ℓ_2; $\ldots$; $\ell_{\nu-1}$ is directly connected with ℓ_ν; and ℓ_ν is directly connected with j.
- An *open path* is a connection from a node i to a node j where each node on the way is only passed through once.
- A *loop* is an open path that returns to the starting node.
- A *path* from a node i to a node j is an arbitrary combination of open paths and loops that connects the starting node with the end node. So we can also say that node i is connected with node j if there exists a path from i to j.

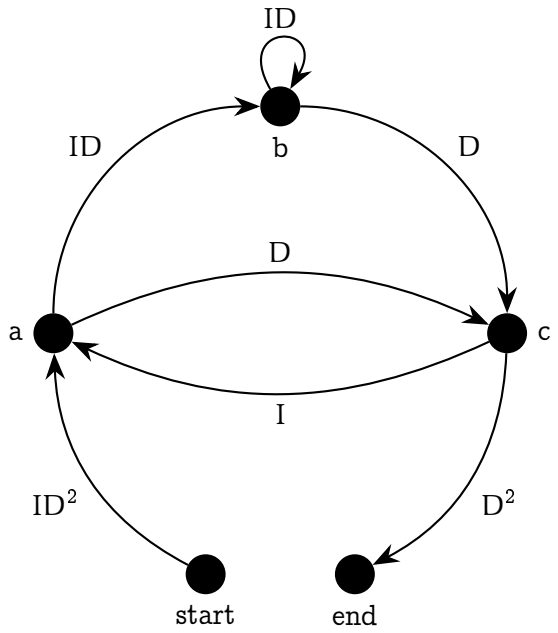


Figure 13.12: Signalflowgraph of the state-transition diagram that is cut open at the all-zero node.

- Each path has a *path gain* G that is the product of all weight factors of the edges along the path.
- We say that two paths *touch* each other if they share at least one node.
- The *transmission gain* T of a signalflowgraph between a starting node and an ending node is the sum of the path gains of all possible paths (including arbitrary numbers of loops) between the starting and the ending node.

Example 13.11. Consider the signalflowgraph shown in Figure 13.13. The path

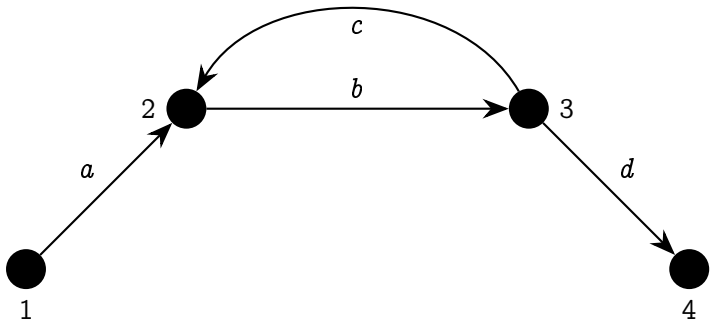


Figure 13.13: Example of a signalflowgraph.

'$1 \to 2 \to 3 \to 4$' is an open path, '$2 \to 3 \to 2$' is a loop, and '$1 \to 2 \to 3 \to 2 \to 3 \to 4$' is a path that is neither open nor a loop. The path gain of this last path is

$$\mathsf{G}_{1\to2\to3\to2\to3\to4} = a \cdot b^2 \cdot c \cdot d. \tag{13.27}$$

To compute the transmission gain of this signalflowgraph (assuming that node 1 is the starting node and node 4 the ending node), we need to list all possible paths from 1 to 4 with their path gains:

$$\begin{aligned}
\mathsf{G}_{1\to2\to3\to4} &= a \cdot b \cdot d, && (13.28)\\
\mathsf{G}_{1\to2\to3\to2\to3\to4} &= a \cdot b^2 \cdot c \cdot d, && (13.29)\\
\mathsf{G}_{1\to2\to3\to2\to3\to2\to3\to4} &= a \cdot b^3 \cdot c^2 \cdot d, && (13.30)\\
\mathsf{G}_{1\to2\to3\to2\to3\to2\to3\to2\to3\to4} &= a \cdot b^4 \cdot c^3 \cdot d, && (13.31)\\
&\vdots
\end{aligned}$$

Hence, we get

$$\begin{aligned}
\mathsf{T} &= \mathsf{G}_{1\to2\to3\to4} + \mathsf{G}_{1\to2\to3\to2\to3\to4} + \mathsf{G}_{1\to2\to3\to2\to3\to2\to3\to4}\\
&\quad + \mathsf{G}_{1\to2\to3\to2\to3\to2\to3\to2\to3\to4} + \cdots && (13.32)\\
&= abd + ab^2cd + ab^3c^2d + ab^4c^3d + \cdots && (13.33)\\
&= abd \sum_{i=0}^{\infty} b^i c^i && (13.34)\\
&= abd \cdot \frac{1}{1-bc} = \frac{abd}{1-bc}. && (13.35)
\end{aligned}$$

Note that we quite easily managed to compute the transmission gain for this example by listing the paths and cleverly arrange them. However, this was mainly due to the simplicity of the graph. In general, we need a more powerful tool. ◊

We will now state a general rule on how to compute the transmission gain of an arbitrary signalflowgraph.

Theorem 13.12 (Mason's Rule).
The transmission gain T of a signalflowgraph is given by

$$\mathsf{T} = \frac{\sum_{\text{all open paths } k} \mathsf{G}_k \Delta_k}{\Delta}. \tag{13.36}$$

Here the sum is over all possible open paths from starting node to end node, and G_k corresponds to the path gain of the kth of these open paths. The factor Δ is denoted **determinant** of the signalflowgraph and is

computed as follows:

$$\Delta \triangleq 1 - \sum_{\text{all loops } \ell} \mathsf{G}_\ell + \sum_{\substack{\text{all loops } \ell_1,\, \ell_2 \\ \ell_1 \text{ and } \ell_2 \text{ do not touch}}} \mathsf{G}_{\ell_1} \cdot \mathsf{G}_{\ell_2} - \sum_{\substack{\text{all loops } \ell_1,\, \ell_2,\, \ell_3 \\ \ell_1,\, \ell_2,\, \ell_3 \text{ do not touch}}} \mathsf{G}_{\ell_1} \cdot \mathsf{G}_{\ell_2} \cdot \mathsf{G}_{\ell_3} + \cdots - \cdots \tag{13.37}$$

Finally, the **cofactor** Δ_k of the kth open path is the determinant of that part of the graph that does **not** touch the kth path.

Example 13.13. We return to the signalflowgraph of Figure 13.13. The determinant of this graph is

$$\Delta = 1 - b \cdot c. \tag{13.38}$$

Since there is only one open path from 1 to 4 with $\mathsf{G} = abd$ and since this open path has cofactor $\Delta_1 = 1$ (the path touches all parts of the graph!), we get

$$\mathsf{T} = \frac{abd}{1 - bc}, \tag{13.39}$$

exactly as we have seen already in (13.35). ◊

Example 13.14. We make another, slightly more complicated example. Consider the signalflowgraph in Figure 13.14.

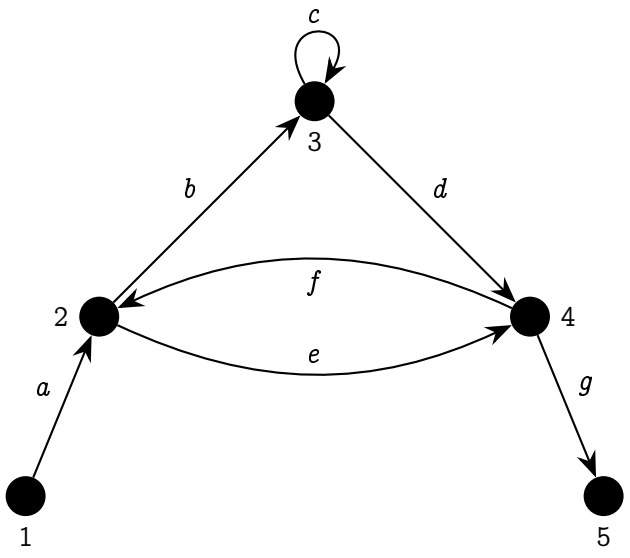


Figure 13.14: Another example of a signalflowgraph.

For this graph we have two open paths from 1 to 5 and therefore two corresponding cofactors:

$$\mathsf{G}_{1\to2\to4\to5} = aeg, \qquad \Delta_{1\to2\to4\to5} = 1-c, \tag{13.40}$$

$$\mathsf{G}_{1\to2\to3\to4\to5} = abdg, \qquad \Delta_{1\to2\to3\to4\to5} = 1. \tag{13.41}$$

Moreover, the determinant is

$$\Delta = 1 - c - bdf - ef + cef. \tag{13.42}$$

Note that we have three loops: $3\to3$, $2\to3\to4\to2$, and $2\to4\to2$; two of which do not touch each other.

Hence, using Mason's rule (Theorem 13.12) we get

$$\mathsf{T} = \frac{aeg(1-c)+abdg}{1-c-bdf-ef+cef}. \tag{13.43}$$

$\Diamond$

So, let us now return to the signalflowgraph of our convolutional encoder from Figure 13.12. We recall from Example 13.9 that a path gain describes the number of information bit errors and the Hamming distance of the corresponding detour with respect to the correct path. Moreover, the transmission gain by definition is the summation of all possible path gains from starting node to ending node. But this is exactly what we need for the analysis of our detours:

$$\mathsf{T} = \sum_{\substack{\text{all paths } k \text{ from}\\ \text{state } \mathbf{0} \text{ to state } \mathbf{0}}} \mathsf{G}_k \tag{13.44}$$

$$= \sum_{\text{all detours } \ell} \mathsf{I}^{i_\ell}\mathsf{D}^{d_\ell} \tag{13.45}$$

$$= \sum_{i=1}^{\infty}\sum_{d=1}^{\infty} a(d,i)\mathsf{I}^i\mathsf{D}^d. \tag{13.46}$$

Here the first equality follows from the definition of the transmission gain; the subsequent equality from the way how we transform the state-transition diagram into a signalflowgraph with the all-zero state cut open and changed into the starting and ending node and with each edge labeled by factors I and D depending on the caused errors; and in the final equality we rearrange the summation order and collect all terms with identical exponents i and d together, using $a(d,i)$ as the count of such identical detours.

We see that if we compute the transmission gain, we can read the wanted values of $a(d,i)$ directly out of the expression.

Example 13.15. We continue with Figure 13.12 and compute the transmission gain using Mason's rule (note that the structure of the signalflowgraph is

identical to the example in Figure 13.14):

$$\mathsf{T} = \frac{\mathsf{ID}^5 \cdot (1-\mathsf{ID}) + \mathsf{I}^2\mathsf{D}^6}{1-\mathsf{ID}-\mathsf{I}^2\mathsf{D}^2-\mathsf{ID}+\mathsf{I}^2\mathsf{D}^2} = \frac{\mathsf{ID}^5}{1-2\mathsf{ID}}. \tag{13.47}$$

If we write this as a geometrical series, we get

$$\mathsf{T} = \mathsf{ID}^5\Big(1+2\mathsf{ID}+4\mathsf{I}^2\mathsf{D}^2+8\mathsf{I}^3\mathsf{D}^3+\cdots\Big) \tag{13.48}$$

$$= \mathsf{ID}^5+2\mathsf{I}^2\mathsf{D}^6+4\mathsf{I}^3\mathsf{D}^7+8\mathsf{I}^4\mathsf{D}^8+\cdots, \tag{13.49}$$

from which we see that $a(5,1)=1$, $a(6,2)=2$, $a(7,3)=4$, $a(8,4)=8$, etc. ♢

13.3.3 Upper Bound on the Bit Error Probability of a Trellis Code

So now we have a way of deriving the number of "wrong paths", i.e., detours, and their impact concerning information bit errors and Hamming distance to the correct path. Next we will use the Bhattacharyya Bound derived in Section 11.4 to find a relation between these detours and the error probability. This will then lead to an upper bound on the bit error probability of a trellis code.

We make the following assumptions and definitions:

- We consider a convolutional encoder that is clocked L_t times using information bits as input and T times using dummy zero bits as input. Hence, every path in the corresponding trellis will pass through $\mathsf{L}_\mathsf{t}+\mathsf{T}$ nodes before it ends in the final all-zero node. We number these nodes $j=1,\dots,\mathsf{L}_\mathsf{t}+\mathsf{T}+1$. (See also Example 13.16 below.)

- We will assume that at every time step, k_0 information bits enter the encoder (so far, we have always assumed $k_0=1$, but we want to keep this analysis general). Hence, we are using in total $k_0\mathsf{L}_\mathsf{t}$ information bits and $k_0\mathsf{T}$ dummy bits.

- We denote by $P_{\mathrm{e},i}$ the probability of a decoding error for the ith information bit. Then the *average bit error probability* or *average bit error rate (BER)* is

$$P_\mathrm{b} \triangleq \frac{1}{k_0\mathsf{L}_\mathsf{t}} \sum_{i=1}^{k_0\mathsf{L}_\mathsf{t}} P_{\mathrm{e},i}. \tag{13.50}$$

- We recall that in the infinite trellis, the number of detours starting at node j is identical to the number of detours starting at node 1. We number these detours in the infinite trellis in some (arbitrary) way: $\ell=1,2,\dots$

- For each of these detours $\ell = 1, 2, \ldots$ let

$$\begin{aligned} d_\ell &\triangleq \text{Hamming distance between the } \ell\text{th detour and the} \\ &\qquad \text{corresponding segment of the correct path,} \\ i_\ell &\triangleq \text{number of information bit errors that occur along} \\ &\qquad \text{the } \ell\text{th detour.} \end{aligned}$$

- We define the following events: For any node[1] $j = 1, \ldots, \mathrm{L_t}$ and for any detour $\ell = 1, 2, \ldots$,

$$\mathcal{B}_{j,\ell} \triangleq \{\text{Viterbi decoder follows the } \ell\text{th detour starting at node } j\}, \tag{13.51}$$

and for any node $j = 1, \ldots, \mathrm{L_t}$,

$$\begin{aligned} \mathcal{B}_{j,0} \triangleq \{&\text{Viterbi decoder does not follow} \\ &\text{any detour starting at node } j\}. \end{aligned} \tag{13.52}$$

Note that $\mathcal{B}_{j,0}$ can occur for two different reasons: Either the Viterbi decoder does not make an error at node j or the Viterbi decoder is already on a detour that started earlier. Note further that for every node j, exactly one event $\mathcal{B}_{j,\ell}$ must occur, i.e.,

$$\sum_{\ell=0}^{\infty} \Pr(\mathcal{B}_{j,\ell}) = 1, \quad \text{for all } j = 1, \ldots, \mathrm{L_t}. \tag{13.53}$$

Note that, in general, the events $\mathcal{B}_{j,\ell}$ are highly dependent (e.g., if we start a detour at node j, we cannot start another one at node $j+1$).

- Let W_j be the number of information bit errors that occur when the Viterbi decoder follows a detour starting at node j. Note that W_j depends on which detour is chosen. Note further that $W_j = 0$ if the decoder does not start a detour at node j or it is already on a detour that started earlier.

  We see that

$$W_j = \begin{cases} 0 & \text{if } \mathcal{B}_{j,0} \text{ occurs,} \\ i_\ell & \text{if } \mathcal{B}_{j,\ell} \text{ occurs } (\ell \geq 1). \end{cases} \tag{13.54}$$

- Let V_i be a binary indicator random variable such that

$$V_i \triangleq \begin{cases} 1 & \text{if the } i\text{th information bit is in error,} \\ 0 & \text{otherwise.} \end{cases} \tag{13.55}$$

[1] At nodes $\mathrm{L_t}+1, \mathrm{L_t}+2, \ldots$ no detours can start because there we will be using dummy zero bits.

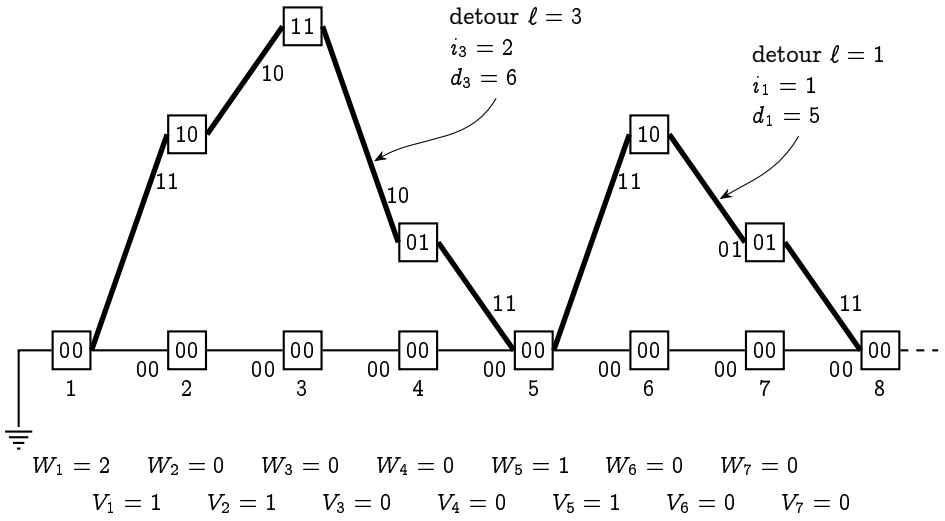


Figure 13.15: Example of a path in a trellis that contains two detours: The first detour (detour $\ell = 3$, although this numbering is arbitrary!) starts at node 1, the second detour (detour $\ell = 1$) at node 5.

Note that

$$\Pr[V_i = 1] = P_{\mathrm{e},i} \tag{13.56}$$

and

$$\mathsf{E}[V_i] = 1 \cdot \Pr[V_i = 1] + 0 \cdot \Pr[V_i = 0] = P_{\mathrm{e},i}. \tag{13.57}$$

Example 13.16. Consider the example shown in Figure 13.15. We see that the following events take place: $\mathcal{B}_{1,3}$, $\mathcal{B}_{2,0}$, $\mathcal{B}_{3,0}$, $\mathcal{B}_{4,0}$, $\mathcal{B}_{5,1}$, $\mathcal{B}_{6,0}$, $\mathcal{B}_{7,0}$. In total we make three information bit errors, the first, second, and the fifth bits are wrong. ◊

Putting everything together, we now get the following:

$$P_{\mathrm{b}} = \frac{1}{k_0 \mathrm{L_t}} \sum_{i=1}^{k_0 \mathrm{L_t}} P_{\mathrm{e},i} \tag{13.58}$$

$$= \frac{1}{k_0 \mathrm{L_t}} \sum_{i=1}^{k_0 \mathrm{L_t}} \mathsf{E}[V_i] \tag{13.59}$$

$$= \mathsf{E}\left[\frac{1}{k_0 \mathrm{L_t}} \sum_{i=1}^{k_0 \mathrm{L_t}} V_i\right], \tag{13.60}$$

where the last step follows because expectation is a linear operation. Note that $\sum_{i=1}^{k_0 \mathrm{L_t}} V_i$ denotes the total number of information bit errors. This number can

also be computed in a different way by summing over W_j:

$$\sum_{i=1}^{k_0 \mathrm{L_t}} V_i = \sum_{j=1}^{\mathrm{L_t}} W_j. \tag{13.61}$$

Hence we have

$$\begin{aligned} P_\mathrm{b} &= \mathsf{E}\left[\frac{1}{k_0 \mathrm{L_t}} \sum_{i=1}^{k_0 \mathrm{L_t}} V_i\right] && (13.62)\\ &= \mathsf{E}\left[\frac{1}{k_0 \mathrm{L_t}} \sum_{j=1}^{\mathrm{L_t}} W_j\right] && (13.63)\\ &= \frac{1}{k_0 \mathrm{L_t}} \sum_{j=1}^{\mathrm{L_t}} \mathsf{E}[W_j] && (13.64)\\ &= \frac{1}{k_0 \mathrm{L_t}} \sum_{j=1}^{\mathrm{L_t}} \sum_{\ell=0}^{\infty} \Pr(\mathcal{B}_{j,\ell})\, \mathsf{E}[W_j \,|\, \mathcal{B}_{j,\ell}] && (13.65)\\ &= \frac{1}{k_0 \mathrm{L_t}} \sum_{j=1}^{\mathrm{L_t}} \left(\Pr(\mathcal{B}_{j,0}) \cdot 0 + \sum_{\ell=1}^{\infty} \Pr(\mathcal{B}_{j,\ell})\, i_\ell \right) && (13.66)\\ &= \frac{1}{k_0 \mathrm{L_t}} \sum_{j=1}^{\mathrm{L_t}} \sum_{\ell=1}^{\infty} \Pr(\mathcal{B}_{j,\ell})\, i_\ell. && (13.67) \end{aligned}$$

Here, (13.64) follows again from the linearity of expectation; in (13.65) we use the theorem of total expectation (see Chapter 2), based on (13.53); and in (13.66) we apply (13.54).

Now, in order to get a grip on $\mathcal{B}_{j,\ell}$ and $\Pr(\mathcal{B}_{j,\ell})$, we need to simplify things:

- Firstly, we upper-bound the probability by including all possible (infinitely many) detours in the infinite trellis instead of only the (finite number of) possible detours that can start from node j in our finite trellis.
- Secondly, we upper-bound the probability further by completely ignoring the dependency of $\mathcal{B}_{j,\ell}$ on j, i.e., by ignoring the possibility that the decoder could already be on a detour.
- Thirdly, we upper-bound the probability even more by ignoring all other detours $\mathcal{B}_{j,\ell'}$, $\ell' \neq \ell$, i.e., we assume that the likelihood of the ℓth detour must not necessarily be the largest of all detours, but it is sufficient if it is larger than the corresponding correct path. In short: We reduce the problem to a two-codeword problem discussed in Section 11.4.

We can then bound according to Corollary 11.21:

$$\Pr(\mathcal{B}_{j,\ell}) \le \left(2^{-\mathrm{D_B}}\right)^{d_\ell}. \tag{13.68}$$

So we get

$$P_{\mathrm{b}} \le \frac{1}{k_0 \mathrm{L_t}} \sum_{j=1}^{\mathrm{L_t}} \sum_{\ell=1}^{\infty} \left(2^{-\mathrm{D_B}}\right)^{d_\ell} i_\ell \tag{13.69}$$

$$= \frac{1}{k_0 \mathrm{L_t}} \sum_{\ell=1}^{\infty} i_\ell \left(2^{-\mathrm{D_B}}\right)^{d_\ell} \underbrace{\sum_{j=1}^{\mathrm{L_t}} 1}_{=\mathrm{L_t}} \tag{13.70}$$

$$= \frac{1}{k_0} \sum_{\ell=1}^{\infty} i_\ell \left(2^{-\mathrm{D_B}}\right)^{d_\ell} \tag{13.71}$$

$$= \frac{1}{k_0} \sum_{i=1}^{\infty} \sum_{d=1}^{\infty} a(d,i)\, i \left(2^{-\mathrm{D_B}}\right)^{d}, \tag{13.72}$$

where in the last equality we have rearranged the summation order: Instead of summing over all possible detours, we group all detours with i information bit errors and Hamming distance d together.

Note that this upper bound holds for any $\mathrm{L_t}$, even for $\mathrm{L_t} = \infty$, i.e., the case when we do not insert the tail of dummy bits. Also note that $\mathrm{D_B}$ is a characteristic of the channel, while $a(d,i)$ and k_0 are characteristics of the encoder.

Now we only need to recall from Section 13.3.2 that

$$\mathrm{T}(\mathrm{D},\mathrm{I}) = \sum_{i=1}^{\infty} \sum_{d=1}^{\infty} a(d,i) \mathrm{I}^i \mathrm{D}^d, \tag{13.73}$$

take the derivative of this expression with respect to I:

$$\frac{\partial}{\partial \mathrm{I}} \mathrm{T}(\mathrm{D},\mathrm{I}) = \sum_{i=1}^{\infty} \sum_{d=1}^{\infty} a(d,i)\, i\, \mathrm{I}^{i-1} \mathrm{D}^d, \tag{13.74}$$

and compare it with (13.72) to get the following main result.

Theorem 13.17. The average bit error probability (BER) of a convolutional encoder that is used on a binary-input DMC without feedback is upper-bounded as follows:

$$P_{\mathrm{b}} \le \frac{1}{k_0} \left. \frac{\partial \mathrm{T}(\mathrm{D},\mathrm{I})}{\partial \mathrm{I}} \right|_{\mathrm{I}=1, \mathrm{D}=2^{-\mathrm{D_B}}} \tag{13.75}$$

where $\mathrm{T}(\mathrm{D},\mathrm{I})$ is the transmission gain of the signalflowgraph corresponding to the state transition diagram of the convolutional encoder; where $\mathrm{D_B}$ is the Bhattacharyya distance of the DMC; and where k_0 is the number of information bits that enter the encoder at each time step.

Example 13.18. We return to the main example of this chapter: the encoder given in Figure 13.1. From Example 13.15 we know that

$$\mathsf{T}(\mathsf{D},\mathsf{I}) = \frac{\mathsf{I}\mathsf{D}^5}{1-2\mathsf{I}\mathsf{D}}. \tag{13.76}$$

Hence,

$$\frac{\partial \mathsf{T}(\mathsf{D},\mathsf{I})}{\partial \mathsf{I}} = \frac{\mathsf{D}^5(1-2\mathsf{I}\mathsf{D}) + \mathsf{I}\mathsf{D}^5 2\mathsf{D}}{(1-2\mathsf{I}\mathsf{D})^2} = \frac{\mathsf{D}^5}{(1-2\mathsf{I}\mathsf{D})^2}. \tag{13.77}$$

Moreover, we have $k_0 = 1$.

We now assume that we use this encoder on a BEC with erasure probability δ. Then, from Example 11.22 we know that

$$2^{-\mathrm{D_B}} = \delta. \tag{13.78}$$

Using these values in (13.75) now gives the following bound:

$$P_{\mathrm{b}} \le \frac{\delta^5}{(1-2\delta)^2}. \tag{13.79}$$

For example, if $\delta = 0.1$, then $P_{\mathrm{b}} \le 1.56 \cdot 10^{-5}$. ◊

Since (13.75) is an upper-bound, any encoder design will be performing better than what Theorem 13.17 predicts. However, note that the bound (13.75) is in general quite tight. This means that if we design a system according to the upper-bound instead of the exact average bit error probability, our design will usually not result in unnecessary complexity.

Chapter 14

Polar Codes

In contrast to source coding where soon after the fundamental result (Theorem 5.2) was published by Shannon, an optimal scheme was found (Huffman coding), channel coding proved to be a much harder nut to crack. For years engineers tried to find a practical system (i.e., one with manageable complexity) that would approach the performance of an optimal coding scheme. The irony is that a randomly picked system will quite likely work very well (as can be seen from the random coding proof of Shannon!), but any such system is impossible in practice as there is no structure in the code that would allow efficient encoding and decoding. All investigated structured codes like, e.g., algebraic codes that use the structure of vector spaces and subspaces, turned out to be far from optimal.

The first real breakthrough was the discovery of *turbo codes* [BGT93] in 1993 (see the end of Chapter 17 for a more detailed discussion). Nevertheless, turbo codes (and also the even more efficient *low-density parity-check (LDPC) codes* [MN96], [DM98]) are not proven to be good (or even optimal), but are known to perform well simply from experience.

The first *provably* capacity-achieving coding scheme that at the same time also has decent complexity is *polar coding* introduced by Erdal Arıkan in 2007 [Arı09]. In this chapter we are going to study Arıkan's idea in its original form that was restricted to binary-input DMCs (with an arbitrary finite output alphabet).

Note that in this chapter we will again use the notation introduced in Section 12.4: W denotes the conditional probability distribution of the DMC.

14.1 Polar Transform

The basic idea of polar coding is the insight that there are basically two special types of DMCs for which communication is trivial:

- In a *perfect channel*, where the output Y determines the input X, we do not need to code at all because we will never make any mistakes.

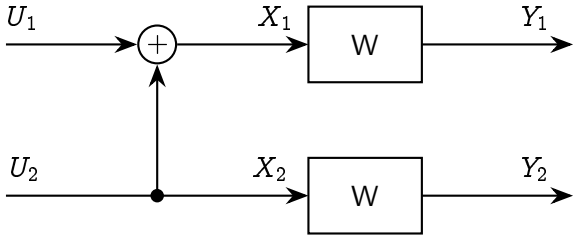


Figure 14.1: The polar transform underlying the construction of polar codes.

- In a *useless channel*, where the output Y is independent of the input X, we do not need to code either, because whatever we try, we will never be able to communicate anyway.

Arıkan now found a way of transforming two given identical DMCs W in such a way that one of the newly generated DMCs is better than the original channel, while the other is worse. By repeated application of this transformation process, we end up with a bunch of channels that are either perfect or useless, but nothing in between. This effect is called *polarization*. Once we have polarized all channels, we do not need to code anymore, but can directly transmit information bits over the perfect channels and dummy bits over the useless ones.

We will show that the fraction of channels that are perfect is equal to the mutual information of the original channel under a uniform input distribution. If the uniform input is capacity achieving, then the fraction of perfect channels equals the capacity of the original channel, and thus the ratio of the number of information bits transmitted over the perfect channels to the number of total transmitted bits matches exactly the capacity of the channel, i.e., the coding scheme is optimal.

We start with Arıkan's way of transforming the channels. To understand the following definition and its consequences, recall that in a coding scheme a given channel W is used n times to transmit a codeword. Instead of using the same channel n times in series, however, we can equivalently think about having a bank of n independent identical channels W that are used in parallel just once. Polar codes are more easily understood when using this second equivalent picture.

Definition 14.1. The *polar transform* $\mathsf{W} \rightsquigarrow (\mathsf{W}^-, \mathsf{W}^+)$ takes a binary-input DMC $\mathsf{W}\colon \{0,1\} \to \mathcal{Y}$ and produces the following two binary-input DMCs (with larger output alphabets!):

$$\mathsf{W}^-\colon \{0,1\} \to \mathcal{Y}^2\colon \qquad U_1 \mapsto (Y_1, Y_2), \tag{14.1}$$

$$\mathsf{W}^+\colon \{0,1\} \to \mathcal{Y}^2 \times \{0,1\}\colon \qquad U_2 \mapsto (Y_1, Y_2, U_1), \tag{14.2}$$

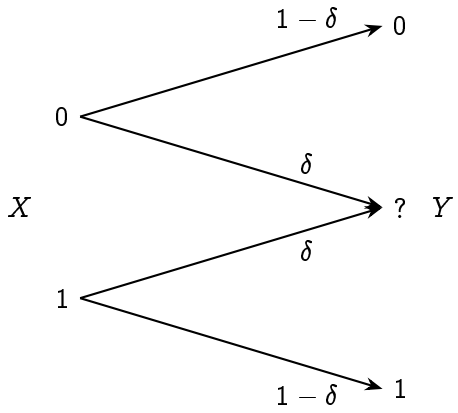


Figure 14.2: Binary erasure channel (BEC).

where Y_1 and Y_2 are the outputs of two identical copies of W with corresponding inputs $X_1 = U_1 \oplus U_2$ and $X_2 = U_2$, respectively (see Figure 14.1). Note that we will assume that U_1 and U_2 are independent uniform binary RVs.

We call the channels W^- and W^+ the *children channels* of W.

Note that in Definition 14.1 we describe a way of duplicating a given DMC. In reality, however, we do not "duplicate" a channel, but actually start with two identical copies of the same DMC W that are then transformed into W^- and W^+.

The following corollary is a direct consequence of Definition 14.1 and our assumption that U_1 and U_2 are uniform.

Corollary 14.2. The channel laws of W^- and W^+ are given as follows:

$$W^-(y_1,y_2|u_1) = \sum_{u_2} P_{U_2}(u_2)\, W(y_1|u_1\oplus u_2)\, W(y_2|u_2) \tag{14.3}$$

$$= \frac{1}{2}\, W(y_1|u_1)\, W(y_2|0) + \frac{1}{2}\, W(y_1|u_1\oplus 1)\, W(y_2|1), \tag{14.4}$$

$$W^+(y_1,y_2,u_1|u_2) = P_{U_1}(u_1)\, W(y_1|u_1\oplus u_2)\, W(y_2|u_2) \tag{14.5}$$

$$= \frac{1}{2}\, W(y_1|u_1\oplus u_2)\, W(y_2|u_2). \tag{14.6}$$

Example 14.3 (BEC). Consider the case of W being a binary erasure channel (BEC) with erasure probability δ as shown in Figure 14.2. What channels are W^- and W^+?

Recall that U_1 and U_2 are assumed to be chosen independently and both with equal probability of being 0 or 1.

The DMC W^- has input U_1 and output (Y_1, Y_2).

- If there is no erasure, we receive $(Y_1, Y_2) = (U_1 \oplus U_2, U_2)$ from which U_1 can be determined correctly.

- If there is an erasure on the first BEC, we receive $(Y_1, Y_2) = (?, U_2)$ from which U_1 cannot be determined.
- If there is an erasure on the second BEC, we receive $(Y_1, Y_2) = (U_1 \oplus U_2, ?)$ from which U_1 cannot be determined (because we assume that U_2 is uniform and independent of U_1).
- If there are two erasures, we receive $(Y_1, Y_2) = (?, ?)$ from which U_1 cannot be determined.

So if there are no erasures, we get the correct answer, otherwise we have no clue about the transmitted bit. Moreover, we are fully aware whether we can decode correctly, or not. Hence, we see that W^- is again a BEC, but with a worse erasure probability $1-(1-\delta)^2 = 2\delta - \delta^2$.

The DMC W^+, on the other hand, has input U_2 and output (Y_1, Y_2, U_1) (where U_1 is always available and does not suffer from a channel!).

- If there is no erasure, we receive $(Y_1, Y_2, U_1) = (U_1 \oplus U_2, U_2, U_1)$ from which U_2 can be determined correctly.
- If there is an erasure on the first BEC, we receive $(Y_1, Y_2, U_1) = (?, U_2, U_1)$ from which U_2 can be determined correctly.
- If there is an erasure on the second BEC, we receive $(Y_1, Y_2) = (U_1 \oplus U_2, ?, U_1)$ from which U_2 also can be determined correctly.
- If there are two erasures, we receive $(Y_1, Y_2) = (?, ?, U_1)$ from which U_2 cannot be determined.

So only if we have two erasures, we cannot figure out what has been transmitted, in all other cases we get the correct answer. Hence, we see that also W^+ is a BEC, but with a better erasure probability δ^2.

We could have found this also by using Corollary 14.2, but it is more cumbersome. We only show the situation for W^-. From Table 14.3 we see that we either have equal probabilities $W^-(y_1, y_2|0) = W^-(y_1, y_2|1)$ or one of them equals zero. This is exactly the structure of a BEC where

- $\{(0,0), (1,1)\}$ is taken as new symbol 0,
- $\{(0,1), (1,0)\}$ is taken as new symbol 1, and
- $\{(0,?), (?,0), (1,?), (?,1), (?,?)\}$ is taken as new symbol ?.

We also note that W^+ is better than W (it has a smaller erasure probability) and W^- is worse than W (it has a larger erasure probability). Moreover, W^+ is strictly better than W and W^- is strictly worse than W unless $\delta = 0$ or $\delta = 1$ in the first place, i.e., only if the BEC is already perfect or useless from

Table 14.3: Probability distribution of BEC^-.

y_1	y_2	$W^-(y_1,y_2|0)$	$W^-(y_1,y_2|1)$
0	0	$(1-\delta)^2/2$	0
0	?	$\delta(1-\delta)/2$	$\delta(1-\delta)/2$
0	1	0	$(1-\delta)^2/2$
?	0	$\delta(1-\delta)/2$	$\delta(1-\delta)/2$
?	?	δ^2	δ^2
?	1	$\delta(1-\delta)/2$	$\delta(1-\delta)/2$
1	0	0	$(1-\delta)^2/2$
1	?	$\delta(1-\delta)/2$	$\delta(1-\delta)/2$
1	1	$(1-\delta)^2/2$	0

the beginning. We will see next that this is no accident, but holds in principle for any choice of W.

Note that in general the type of channel is changed by the polar transform. For example, if W is a BSC, then neither W^- nor W^+ will be a BSC anymore. ◊

Example 14.3 demonstrates the main property of the polar transform: we see an extremalization in the sense that W^+ is "better" than W and W^- is "worse". We will next make this concept more precise.

Definition 14.4. For a binary-input DMC W, we define the *channel mutual information* I(W) as the mutual information between input and output of the DMC under the assumption that the input is uniformly distributed:

$$\mathrm{I}(\mathrm{W}) \triangleq \mathrm{I}(X;Y)\big|_{X\sim\mathcal{U}(\{0,1\})} \quad [\text{bits}]. \tag{14.7}$$

Note that I(W) is always measured in the unit of bits (logarithm to the basis of 2), but that we are lazy and often omit these units. Since the channel input X is a binary RV, I(W) is a number between 0 and 1 bit.

Theorem 14.5 (Channel Mutual Information and the Polar Transform).
For any DMC W, the channel mutual information of W and its children channels W^+ and W^- satisfy the following:

1. The average channel mutual information remains unchanged:
$$\frac{1}{2}\,\mathrm{I}(\mathsf{W}^-) + \frac{1}{2}\,\mathrm{I}(\mathsf{W}^+) = \mathrm{I}(\mathsf{W}). \tag{14.8}$$
2. We have "guaranteed progress" in the polarization unless the channel is already extremal:
$$\mathrm{I}(\mathsf{W}^+) \geq \mathrm{I}(\mathsf{W}) \geq \mathrm{I}(\mathsf{W}^-) \tag{14.9}$$
with equality if, and only if, W is either perfect ($\mathrm{I}(\mathsf{W}) = 1$) or useless ($\mathrm{I}(\mathsf{W}) = 0$).

The first property shows that the polar transform does not change the total amount of mutual information: The total mutual information of two DMCs W is the same as the total mutual information of the two children channels W^- and W^+. The second property guarantees that W^+ is strictly better than W and W^- is strictly worse than W (unless the extreme values 1 bit or 0 bits have already been reached).

Proof: Note that by definition
$$\mathrm{I}(\mathsf{W}^-) = \mathrm{I}(U_1; Y_1, Y_2), \tag{14.10}$$
$$\mathrm{I}(\mathsf{W}^+) = \mathrm{I}(U_2; Y_1, Y_2, U_1), \tag{14.11}$$
where $U_1 \perp\!\!\!\perp U_2$ and $U_1, U_2 \sim \mathcal{U}(\{0,1\})$. From the chain rule, it follows that
$$\begin{aligned}
\mathrm{I}(\mathsf{W}^-) + \mathrm{I}(\mathsf{W}^+) &= \mathrm{I}(U_1; Y_1, Y_2) + \mathrm{I}(U_2; Y_1, Y_2, U_1) && (14.12)\\
&= \mathrm{I}(U_1; Y_1, Y_2) + \mathrm{I}(U_2; U_1) + \mathrm{I}(U_2; Y_1, Y_2|U_1) && (14.13)\\
&= \mathrm{I}(U_1; Y_1, Y_2) + \mathrm{I}(U_2; Y_1, Y_2|U_1) && (14.14)\\
&= \mathrm{I}(U_1, U_2; Y_1, Y_2) && (14.15)\\
&= \mathrm{I}(X_1, X_2; Y_1, Y_2) && (14.16)\\
&= \mathrm{I}(X_1; Y_1) + \mathrm{I}(X_2; Y_2) && (14.17)\\
&= 2\,\mathrm{I}(\mathsf{W}). && (14.18)
\end{aligned}$$
Here, (14.14) holds because $U_1 \perp\!\!\!\perp U_2$; (14.16) holds because there is a one-to-one relation between (U_1, U_2) and (X_1, X_2); and (14.17) follows from the fact that we consider DMCs. This proves (14.8).

By dropping some arguments and by noting that $X_2 = U_2$, we obtain from (14.11)
$$\mathrm{I}(\mathsf{W}^+) = \mathrm{I}(U_2; Y_1, Y_2, U_1) \geq \mathrm{I}(U_2; Y_2) = \mathrm{I}(X_2; Y_2) = \mathrm{I}(\mathsf{W}). \tag{14.19}$$
This then combines with (14.18) to prove (14.9).

In order to investigate equality in (14.9), we rewrite (14.19) using the chain rule as follows:

$$\begin{aligned}
\mathrm{I}(U_2;Y_1,Y_2,U_1) &= \mathrm{I}(U_2;Y_2)+\mathrm{I}(U_2;U_1|Y_2)+\mathrm{I}(U_2;Y_1|U_1,Y_2) && (14.20)\\
&= \mathrm{I}(\mathsf{W})+\underbrace{\mathrm{I}(U_2,Y_2;U_1)}_{=0}-\underbrace{\mathrm{I}(U_1;Y_2)}_{=0}+\mathrm{I}(U_2;Y_1|U_1,Y_2) && (14.21)\\
&= \mathrm{I}(\mathsf{W})+\mathrm{I}(U_2;Y_1|U_1,Y_2) && (14.22)\\
&= \mathrm{I}(\mathsf{W})+\mathrm{I}(U_2\oplus U_1;Y_1|U_1,Y_2) && (14.23)\\
&= \mathrm{I}(\mathsf{W})+\mathrm{I}(X_1;Y_1|U_1,Y_2) && (14.24)\\
&= \mathrm{I}(\mathsf{W})+\mathrm{H}(Y_1|U_1,Y_2)-\mathrm{H}(Y_1|U_1,Y_2,X_1) && (14.25)\\
&= \mathrm{I}(\mathsf{W})+\mathrm{H}(Y_1|U_1,Y_2)-\mathrm{H}(Y_1|X_1) && (14.26)\\
&\ge \mathrm{I}(\mathsf{W})+\mathrm{H}(Y_1|U_1,Y_2,X_1)-\mathrm{H}(Y_1|X_1) && (14.27)\\
&= \mathrm{I}(\mathsf{W}). && (14.28)
\end{aligned}$$

Here, (14.22) is understood easiest by checking the dependences of U_1, U_2, and Y_2 in Figure 14.1; in (14.23) we add a known quantity to one of the arguments, which does not change the mutual information; (14.26) holds because of the basic properties of a DMC (compare with Theorem 11.11); and (14.27) follows from conditioning that reduces entropy.

The clue is now to realize that we have equality in (14.27) if, and only if, Y_1 is conditionally independent of X_1 given U_1 and Y_2. This can happen only in exactly two cases: Either W is useless, i.e., Y_1 is independent of X_1 and any other quantity related with X_1 such that all conditioning disappears and we have $\mathrm{H}(Y_1)-\mathrm{H}(Y_1)$ in (14.25) (this corresponds to the situation when $\mathrm{I}(\mathsf{W})=0$). Or W is perfect so that from Y_2 we can perfectly recover U_2 and — with the additional help of U_1 — also X_1 (this corresponds to the situation when $\mathrm{I}(\mathsf{W})=1$ bit). □

It can be shown that, for any given fixed channel mutual information $\mathrm{I}(\mathsf{W})$, the BEC yields the largest difference between $\mathrm{I}(\mathsf{W}^+)$ and $\mathrm{I}(\mathsf{W}^-)$ and the BSC yields the smallest difference. Any other DMC will yield a difference that lies somewhere in between. See Figure 14.4 for the corresponding plot.

Exercise 14.6. *In this exercise you are asked to recreate the boundary curves in Figure 14.4.*

1. *Start with* W *being a BEC with erasure probability* δ*. Recall that* $\mathrm{I}(\mathsf{W})=1-\delta$*, and then show that* $\mathrm{I}(\mathsf{W}^+)-\mathrm{I}(\mathsf{W}^-)=2\delta(1-\delta)$*.*

2. *For* W *being a BSC with crossover probability* ϵ*, recall that* $\mathrm{I}(\mathsf{W})=1-\mathrm{H_b}(\epsilon)$*. Then, defining* Z_1 *and* Z_2 *being independent binary RVs*

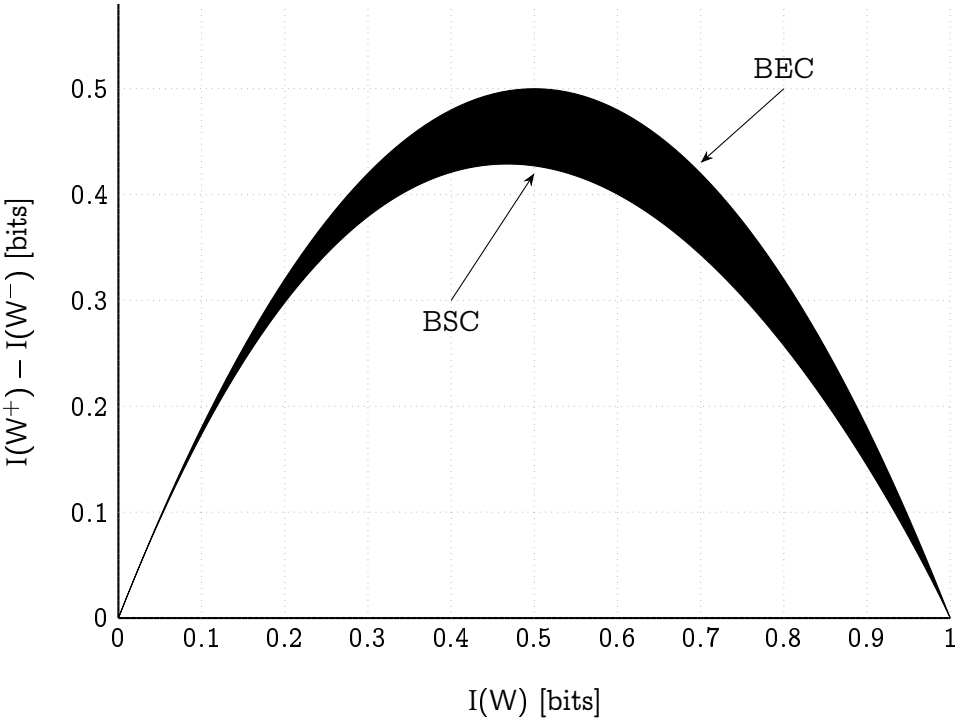


Figure 14.4: Difference of $I(W^+)$ and $I(W^-)$ as a function of $I(W)$. We see that unless the channel is extreme already, the difference is strictly positive. Moreover, the largest difference is achieved for a BEC, while the BSC yields the smallest difference.

with $\Pr[Z_i = 1] = \epsilon$*, explain the steps in the following derivation:*

$$
\begin{aligned}
&I(W^+) - I(W^-) \\
&= I(U_2; U_1 \oplus U_2 \oplus Z_1, U_2 \oplus Z_2, U_1) - I(U_1; U_1 \oplus U_2 \oplus Z_1, U_2 \oplus Z_2) && (14.29)\\
&= I(U_2; U_2 \oplus Z_1, U_2 \oplus Z_2) - I(U_1; U_1 \oplus U_2 \oplus Z_1 | U_2 \oplus Z_2) && (14.30)\\
&= H(U_2 \oplus Z_1) + H(U_2 \oplus Z_2 | U_2 \oplus Z_1) - H(Z_1) - H(Z_2) \\
&\quad - H(U_1 \oplus U_2 \oplus Z_1 | U_2 \oplus Z_2) + H(U_1 \oplus U_2 \oplus Z_1 | U_2 \oplus Z_2, U_1) && (14.31)\\
&= 1 + H(U_2 \oplus Z_2 | U_2 \oplus Z_1) - 2\, H_b(\epsilon) \\
&\quad - H(U_1 \oplus Z_2 \oplus Z_1 | U_2 \oplus Z_2) + H(U_2 \oplus Z_1 | U_2 \oplus Z_2) && (14.32)\\
&= 1 + H(Z_1 \oplus Z_2 | U_2 \oplus Z_1) - 2\, H_b(\epsilon) \\
&\quad - H(U_1 \oplus Z_2 \oplus Z_1) + H(Z_1 \oplus Z_2 | U_2 \oplus Z_2) && (14.33)\\
&= 1 + H(Z_2 \oplus Z_1) - 2\, H_b(\epsilon) - 1 + H(Z_2 \oplus Z_1) && (14.34)\\
&= 2\, H_b(2\epsilon(1-\epsilon)) - 2\, H_b(\epsilon). && (14.35)
\end{aligned}
$$

◊

14.2 Polarization

14.2.1 Recursive Application of the Polar Transform

Note that the polar transform changes the output alphabet of the newly created channels, but that the input alphabet remains binary. Hence, we can apply the polar transform also to the children channels W^- and W^+! In order to do so, we will need to have available two copies each of W^- and W^+, i.e., we firstly apply the polar transform twice to a total of four identical copies of W, see Figure 14.5.

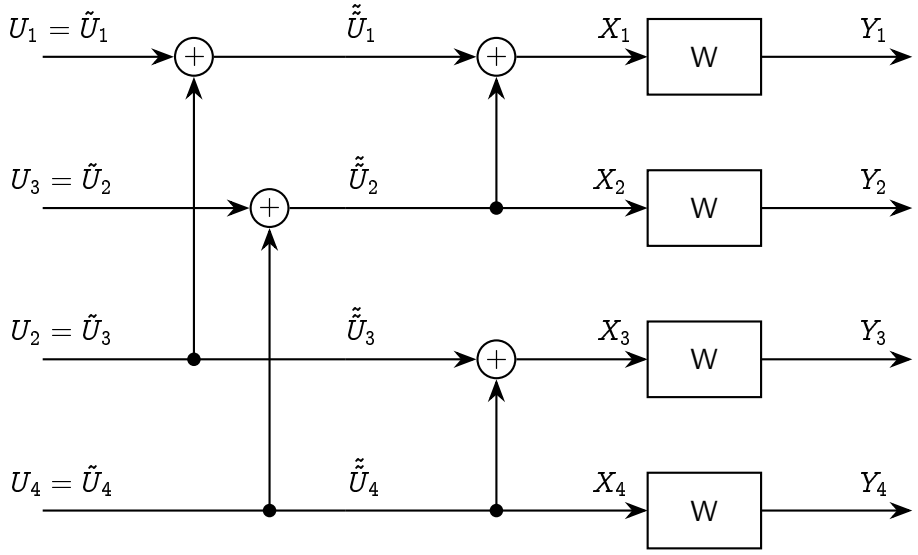


Figure 14.5: Second application of the polar transform to W^- and W^+.

- Apply the polar transform to W: the first copy of W has input X_1 and output Y_1, and the second copy has input X_2 and output Y_2:

$$\mathrm{W}^-\colon \tilde{\tilde{U}}_1 \mapsto (Y_1, Y_2), \qquad \text{where } X_1 = \tilde{\tilde{U}}_1 \oplus \tilde{\tilde{U}}_2, \tag{14.36}$$
$$\mathrm{W}^+\colon \tilde{\tilde{U}}_2 \mapsto (Y_1, Y_2, \tilde{\tilde{U}}_1), \quad \text{where } X_2 = \tilde{\tilde{U}}_2. \tag{14.37}$$

- Repeat with first copy of W having input X_3 and output Y_3, and second copy having input X_4 and output Y_4:

$$\mathrm{W}^-\colon \tilde{\tilde{U}}_3 \mapsto (Y_3, Y_4), \qquad \text{where } X_3 = \tilde{\tilde{U}}_3 \oplus \tilde{\tilde{U}}_4, \tag{14.38}$$
$$\mathrm{W}^+\colon \tilde{\tilde{U}}_4 \mapsto (Y_3, Y_4, \tilde{\tilde{U}}_3), \quad \text{where } X_4 = \tilde{\tilde{U}}_4. \tag{14.39}$$

- Apply the polar transform to W^-: the first copy (14.36) has input $\tilde{\tilde{U}}_1$ and output (Y_1, Y_2), and the second copy (14.38) has input $\tilde{\tilde{U}}_3$ and output

(Y_3, Y_4):

$$\mathsf{W}^{--}\colon \tilde{U}_1 \mapsto (Y_1, Y_2, Y_3, Y_4), \qquad \text{where } \tilde{\tilde{U}}_1 = \tilde{U}_1 \oplus \tilde{U}_3, \tag{14.40}$$
$$\mathsf{W}^{-+}\colon \tilde{U}_3 \mapsto (Y_1, Y_2, Y_3, Y_4, \tilde{U}_1), \quad \text{where } \tilde{\tilde{U}}_3 = \tilde{U}_3. \tag{14.41}$$

- Apply the polar transform to W^{+}: the first copy (14.37) has input $\tilde{\tilde{U}}_2$ and output $(Y_1, Y_2, \tilde{\tilde{U}}_1)$, and the second copy (14.39) has input $\tilde{\tilde{U}}_4$ and output $(Y_3, Y_4, \tilde{\tilde{U}}_3)$:

$$\mathsf{W}^{+-}\colon \tilde{U}_2 \mapsto (Y_1, Y_2, Y_3, Y_4, \tilde{\tilde{U}}_1, \tilde{\tilde{U}}_3), \qquad \text{where } \tilde{\tilde{U}}_2 = \tilde{U}_2 \oplus \tilde{U}_4, \tag{14.42}$$
$$\mathsf{W}^{++}\colon \tilde{U}_4 \mapsto (Y_1, Y_2, Y_3, Y_4, \tilde{\tilde{U}}_1, \tilde{\tilde{U}}_3, \tilde{U}_2), \quad \text{where } \tilde{\tilde{U}}_4 = \tilde{U}_4. \tag{14.43}$$

  Note that since there is a one-to-one relation between $(\tilde{\tilde{U}}_1, \tilde{\tilde{U}}_3)$ and $(\tilde{U}_1, \tilde{U}_3)$, this can be rewritten as

$$\mathsf{W}^{+-}\colon \tilde{U}_2 \mapsto (Y_1, Y_2, Y_3, Y_4, \tilde{U}_1, \tilde{U}_3), \qquad \text{where } \tilde{\tilde{U}}_2 = \tilde{U}_2 \oplus \tilde{U}_4, \tag{14.44}$$
$$\mathsf{W}^{++}\colon \tilde{U}_4 \mapsto (Y_1, Y_2, Y_3, Y_4, \tilde{U}_1, \tilde{U}_3, \tilde{U}_2), \quad \text{where } \tilde{\tilde{U}}_4 = \tilde{U}_4. \tag{14.45}$$

By renaming $U_1 \triangleq \tilde{U}_1$, $U_2 \triangleq \tilde{U}_3$, $U_3 \triangleq \tilde{U}_2$, and $U_4 \triangleq \tilde{U}_4$ (where we swap the indices 2 and 3 on purpose), we get the following four channels:

$$\mathsf{W}^{--}\colon U_1 \mapsto (Y_1, Y_2, Y_3, Y_4), \tag{14.46}$$
$$\mathsf{W}^{-+}\colon U_2 \mapsto (Y_1, Y_2, Y_3, Y_4, U_1), \tag{14.47}$$
$$\mathsf{W}^{+-}\colon U_3 \mapsto (Y_1, Y_2, Y_3, Y_4, U_1, U_2), \tag{14.48}$$
$$\mathsf{W}^{++}\colon U_4 \mapsto (Y_1, Y_2, Y_3, Y_4, U_1, U_2, U_3), \tag{14.49}$$

with the corresponding connections as shown in Figure 14.5.

We continue with the same procedure, taking two independent copies of these four channels and applying the polar transform to all four channels again. We obtain the eight channels shown in Figure 14.6.

Note again that we reorder the inputs on purpose in such a way that we get the channels

$$U_k \mapsto (Y_1^8, U_1^{k-1}), \quad k = 1, \ldots, 8. \tag{14.50}$$

Proposition 14.7 (Naming of Channels and Inputs). Let $n = 2^\ell$. In order to find out how the inputs $\tilde{U}_1^n$ (where $\tilde{U}_k$ corresponds to the new input of the kth channel, i.e., in the graphical representation as in Figure 14.6, $\tilde{U}_k$ "sits" on the same level as the channel input X_k and the channel output Y_k) need to be renamed and to which channel they belong, we use the following algorithm that is generally known as **bit reversal permutation**:

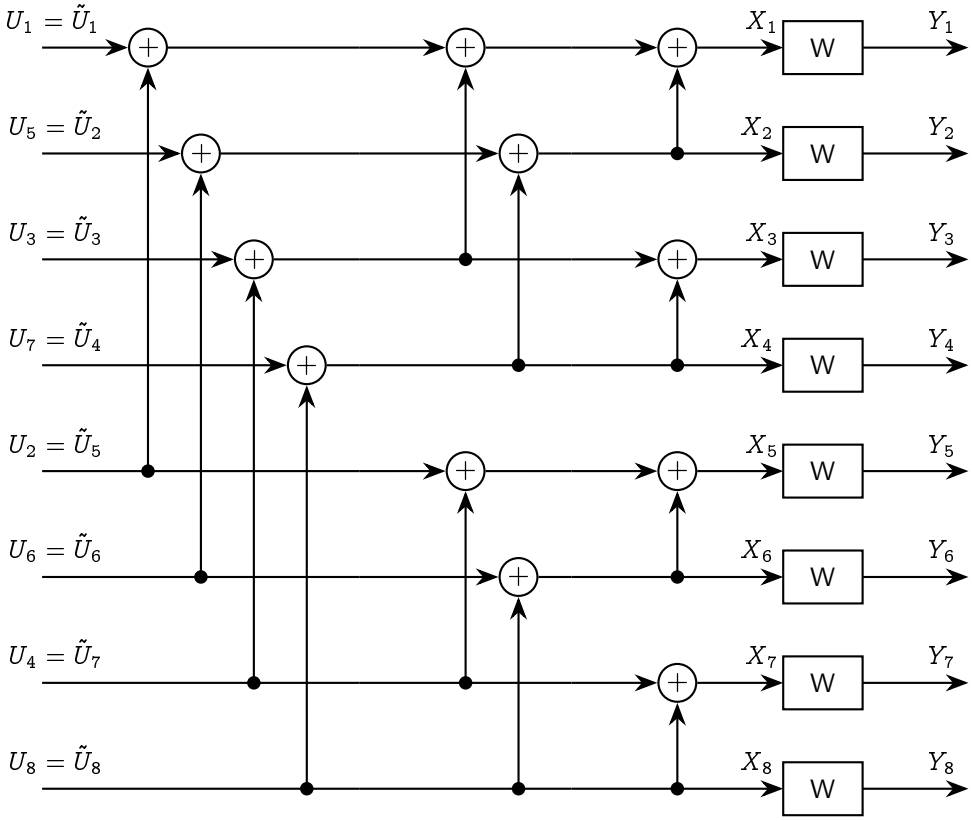


Figure 14.6: Third application of the polar transform.

For $\tilde{U}_k$, take the length-ℓ binary representation of $k-1$ and read it **backwards**. This yields a binary string $(s_1 \cdots s_\ell)$. Then add one to this binary string in order to obtain the binary number of the corresponding input $U_{(s_1 \cdots s_\ell)+1}$, or replace 0 by $-$ and 1 by $+$ to obtain the corresponding channel $\mathsf{W}^{s_1 \cdots s_\ell}$.

In general, we have

$$\mathsf{W}^{s_1 \cdots s_\ell} \colon U_{(s_1 \cdots s_\ell)+1} \mapsto \big(Y_1, \ldots, Y_{2^\ell}, U_1, \ldots, U_{(s_1 \cdots s_\ell)}\big), \qquad s_1, \ldots, s_\ell \in \{0 \text{ or } -, 1 \text{ or } +\}, \tag{14.51}$$

with

$$\mathrm{I}\big(\mathsf{W}^{s_1 \cdots s_\ell}\big) = \mathrm{I}\big(U_{(s_1 \cdots s_\ell)+1}; Y_1, \ldots, Y_{2^\ell}, U_1, \ldots, U_{(s_1 \cdots s_\ell)}\big). \tag{14.52}$$

Note that sometimes we also use the index number of $U_{(s_1 \cdots s_\ell)+1}$ as the name for the channel, i.e.,

$$\mathsf{W}_n^{((s_1 \cdots s_\ell)+1)} = \mathsf{W}^{s_1 \cdots s_\ell}, \tag{14.53}$$

in which case we always add the subscript n to clarify the blocklength.

Example 14.8. For example in Figure 14.6, $\tilde{U}_4$ gets the corresponding length-3 representation of $4-1=3$: (011). Read backwards this corresponds to $(s_1s_2s_3)=(110)$, or $s_1s_2s_3=++-$. Hence,

$$\mathsf{W}^{++-}\colon U_{(110)+1}=U_{(111)}=U_7\mapsto(Y_1^8,U_1^6), \tag{14.54}$$

which is also called $\mathsf{W}_8^{(7)}$.

As another example, consider the situation with $\ell=5$ (i.e., five recursive applications of the polar transform). There, $\tilde{U}_{19}$ is mapped to U_{10} and is used as input to channel $\mathsf{W}^{-+--+}=\mathsf{W}_{32}^{(10)}$ because the binary representation of $19-1=18$ is 10010, which backward is $(s_1\cdots s_5)=(01001)$. This corresponds to $-+---+$, and $(01001)+1=(01010)=10$:

$$\mathsf{W}^{-+--+}=\mathsf{W}_{32}^{(10)}\colon U_{10}\mapsto(Y_1^{32},U_1^9). \tag{14.55}$$

$\Diamond$

Lemma 14.9. For any $k\in\{1,\dots,n\}$, we have

$$\big(\mathsf{W}_n^{(k)}\big)^-=\mathsf{W}_{2n}^{(2k-1)}, \tag{14.56}$$

$$\big(\mathsf{W}_n^{(k)}\big)^+=\mathsf{W}_{2n}^{(2k)}. \tag{14.57}$$

Proof: Let $(s_1\cdots s_\ell)$ be the binary representation of $k-1$, i.e., according to Proposition 14.7 and again using $-$ for 0 and $+$ for 1,

$$\mathsf{W}_n^{(k)}=\mathsf{W}^{s_1\cdots s_\ell}. \tag{14.58}$$

Then

$$\big(\mathsf{W}_n^{(k)}\big)^-=\mathsf{W}^{s_1\cdots s_\ell-}=\mathsf{W}_{2n}^{(k^-)} \tag{14.59}$$

with

$$k^-=(s_1\cdots s_\ell\,0)+1=2(k-1)+1=2k-1. \tag{14.60}$$

Similarly,

$$\big(\mathsf{W}_n^{(k)}\big)^+=\mathsf{W}^{s_1\cdots s_\ell+}=\mathsf{W}_{2n}^{(k^+)} \tag{14.61}$$

with

$$k^+=(s_1\cdots s_\ell\,1)+1=\big(2(k-1)+1\big)+1=2k. \tag{14.62}$$

$\square$

We need to understand these transformations in more detail! Firstly note that if we only consider the last step in the recursive application of the polar transform we realize that the polar transform can also be seen from a vector

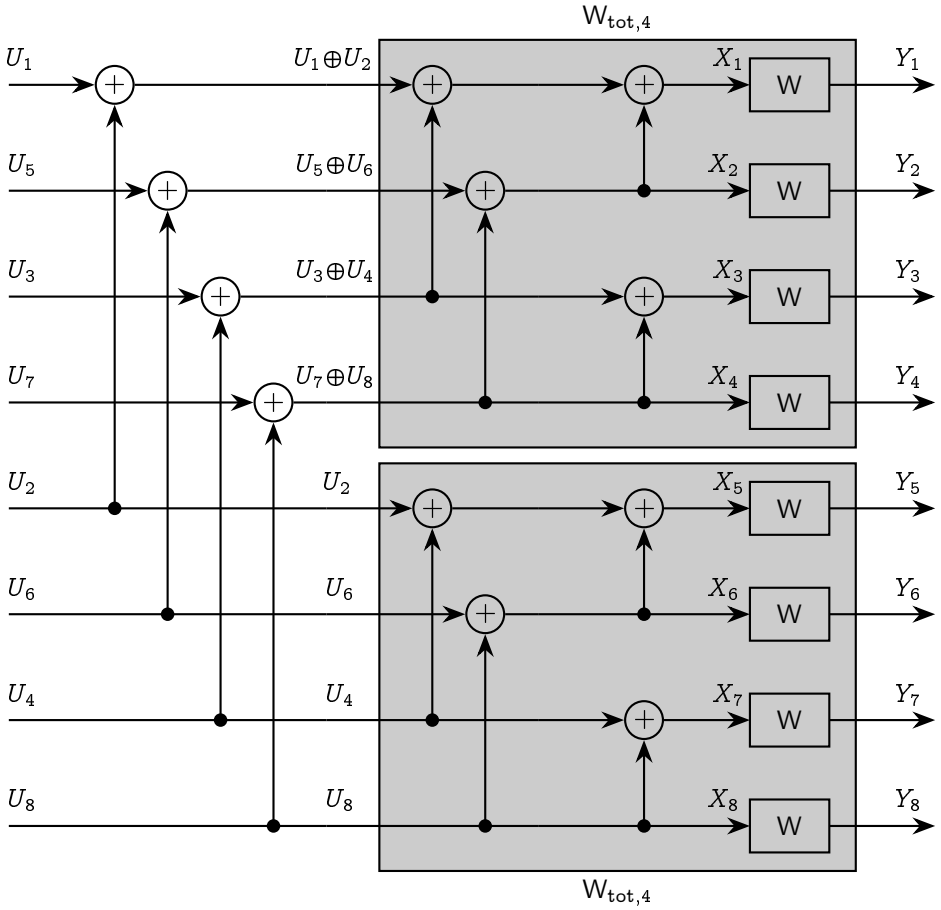


Figure 14.7: Vector channel $\mathsf{W}_{\mathrm{tot},8}$ that is created by application of the polar transform to (two copies of) $\mathsf{W}_{\mathrm{tot},4}$ (compare with Figure 14.6).

channel point of view with an input u_1^n and an output Y_1^n. For example, have a look at $\mathsf{W}_{\mathrm{tot},8}$ shown in Figure 14.7 that is created from an application of the polar transform to two copies of $\mathsf{W}_{\mathrm{tot},4}$ (note that Figure 14.7 simply redraws Figure 14.6!).

In general we have the picture shown in Figure 14.8. In particular, note how U_1^{2n} is split into $U_{1,\mathrm{even}}^{2n}$ and $U_{1,\mathrm{odd}}^{2n}$ and how the inputs of the upper copy of $\mathsf{W}_{\mathrm{tot},n}$ consist of $U_{2k-1} \oplus U_{2k}$, $k = 1, \ldots, n$. We are now ready to understand the following corollary of Lemma 14.9.

Corollary 14.10. For any $k \in \{1, \ldots, n\}$, we have

$$\begin{aligned} &W_{2n}^{(2k-1)}(y_1^{2n}, u_1^{2k-2} | u_{2k-1}) \\ &\quad = \frac{1}{2}\, W_n^{(k)}(y_1^n, u_{1,\mathrm{odd}}^{2k-2} \oplus u_{1,\mathrm{even}}^{2k-2} | u_{2k-1})\, W_n^{(k)}(y_{n+1}^{2n}, u_{1,\mathrm{even}}^{2k-2} | 0) \\ &\qquad + \frac{1}{2}\, W_n^{(k)}(y_1^n, u_{1,\mathrm{odd}}^{2k-2} \oplus u_{1,\mathrm{even}}^{2k-2} | u_{2k-1} \oplus 1)\, W_n^{(k)}(y_{n+1}^{2n}, u_{1,\mathrm{even}}^{2k-2} | 1), \end{aligned} \tag{14.63}$$

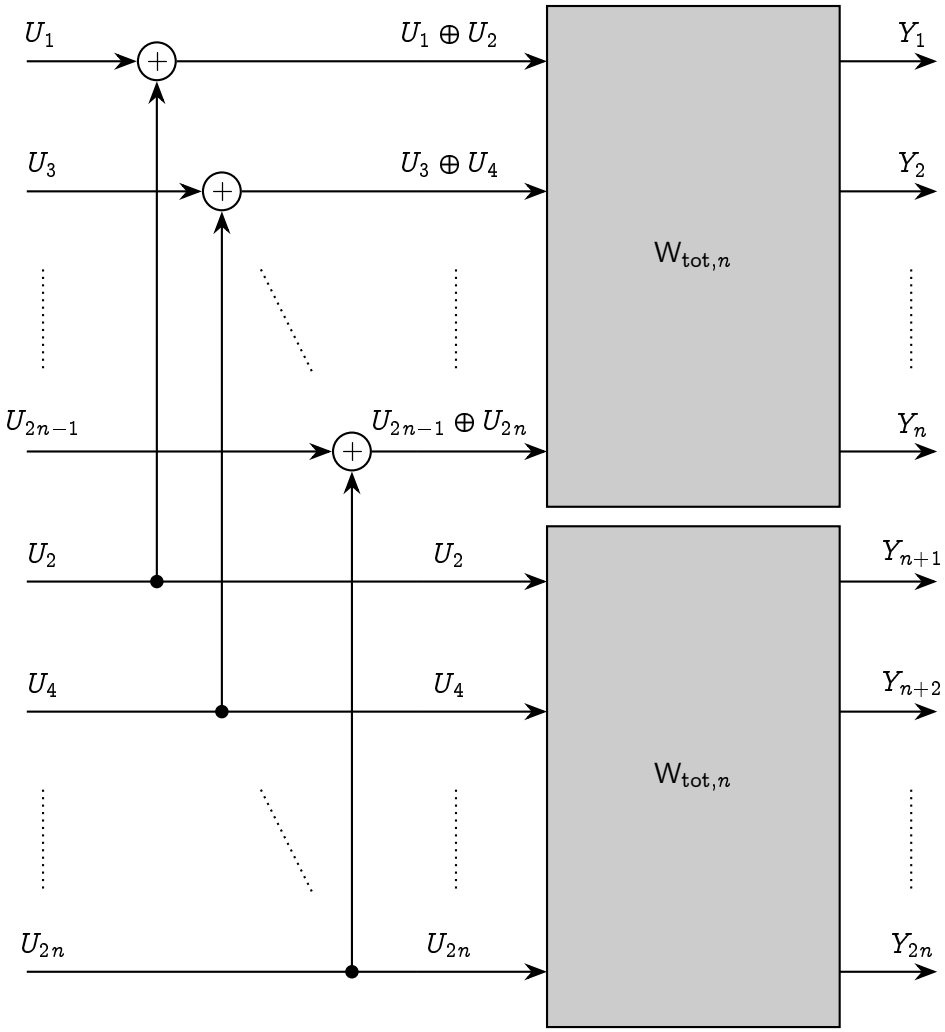


Figure 14.8: Vector channel $\mathsf{W}_{\mathrm{tot},2n}$ that is created by application of the polar transform to (two copies of) $\mathsf{W}_{\mathrm{tot},n}$.

$$W_{2n}^{(2k)}(y_1^{2n}, u_1^{2k-1}|u_{2k}) = \frac{1}{2} W_n^{(k)}(y_1^n, u_{1,\mathrm{odd}}^{2k-2} \oplus u_{1,\mathrm{even}}^{2k-2}|u_{2k-1} \oplus u_{2k})\, W_n^{(k)}(y_{n+1}^{2n}, u_{1,\mathrm{even}}^{2k-2}|u_{2k}). \quad (14.64)$$

Proof: From Lemma 14.9 we know that $\mathsf{W}_{2n}^{(2k-1)}$ is the minus-child of $\mathsf{W}_n^{(k)}$ and that $\mathsf{W}_{2n}^{(2k)}$ is the plus-child of $\mathsf{W}_n^{(k)}$. Thus, the given channel laws follow from Corollary 14.2 after the following substitutions: ($\rightsquigarrow$ stands for "is

replaced by")

$$\begin{aligned} W &\rightsquigarrow W_n^{(k)}, \\ W^- &\rightsquigarrow W_{2n}^{(2k-1)}, & W^+ &\rightsquigarrow W_{2n}^{(2k)}, \\ u_1 &\rightsquigarrow u_{2k-1}, & u_2 &\rightsquigarrow u_{2k}, \\ y_1 &\rightsquigarrow (y_1^n, u_{1,\text{odd}}^{2k-2} \oplus u_{1,\text{even}}^{2k-2}), & y_2 &\rightsquigarrow (y_{n+1}^{2n}, u_{1,\text{even}}^{2k-2}), \\ (y_1, y_2) &\rightsquigarrow (y_1^{2n}, u_1^{2k-2}). \end{aligned} \tag{14.65}$$

□

14.2.2 Matrix Notation

In practice it is often convenient to describe the mapping from $U_1, \ldots, U_n$ to $X_1, \ldots, X_n$ in a mathematical form instead of only giving a graphical definition as in Figures 14.5 or 14.6. Since the transformation is linear, the most obvious choice is to use matrices. Note that the polar transform given in Figure 14.1 can be described by the matrix

$$\mathsf{T} \triangleq \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{14.66}$$

such that[1]

$$\mathbf{X} = (X_1\, X_2) = (U_1\, U_2) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \mathbf{U}\mathsf{T}. \tag{14.67}$$

For a second application of the polar transform as given in Figure 14.5 we then use the Kronecker product $\otimes$ of matrices:

$$\mathbf{X} = \tilde{\mathbf{U}}(\mathsf{T} \otimes \mathsf{T}) \tag{14.68}$$

$$= \tilde{\mathbf{U}} \begin{pmatrix} \mathsf{T} & 0 \\ \mathsf{T} & \mathsf{T} \end{pmatrix} \tag{14.69}$$

$$= \tilde{\mathbf{U}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{14.70}$$

It only remains to apply the bit reversal reordering. This is done by the permutation matrix

$$\mathsf{P}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{14.71}$$

[1] It is common in the coding community to represent codewords by row vectors instead of the column vectors used in math in general. We will follow here the coding tradition.

i.e.,

$$\begin{aligned}\mathbf{X} &= (X_1\, X_2\, X_3\, X_4) && (14.72)\\ &= \mathbf{U}\mathsf{P}_4\mathsf{T}^{\otimes 2} && (14.73)\\ &= (U_1\, U_2\, U_3\, U_4)\begin{pmatrix}1&0&0&0\\1&0&1&0\\1&1&0&0\\1&1&1&1\end{pmatrix} && (14.74)\\ &= \mathbf{U}\mathsf{T}_4. && (14.75)\end{aligned}$$

In general, for $n = 2^\ell$, the transformation is given by the transformation matrix

$$\mathsf{T}_n \triangleq \mathsf{P}_n\, \mathsf{T}^{\otimes \ell}. \tag{14.76}$$

14.2.3 Are these Channels Realistic?

We have now understood how to transform n DMCs W into n children channels (14.51). However, so far it is not clear if these transformed channels are actually realistic in the sense that they can be used in a practical way.

The answer is slightly subtle: Since in reality we do not have access to U_1^{k-1} at the receiver, the transformed channels are not practical at all, but a mere theoretic exercise. However, if we restrict ourselves to a particular decoding structure, they are actually equivalent to a practical setup.

More specifically, we will focus on *successive cancellation decoding*.

Definition 14.11. A *successive cancellation decoder* decodes the bits in successive order, i.e., it firstly decodes the first information bit $\hat{U}_1$, based on the complete channel output Y_1^n, then it decodes the second information bit $\hat{U}_2$ based on the complete channel output Y_1^n and the already decoded first information bit $\hat{U}_1$, etc., until at last it decodes the last information bit $\hat{U}_n$ based on $(Y_1^n, \hat{U}_1^{n-1})$.

Note that a successive cancellation decoder is in general not optimal, i.e., it is not a MAP decoder. However, it is a decoding design that is very appealing to engineers because it splits the problem up: instead of having to decode everything in one blow, it approaches the problem bit by bit, always making use of the complete knowledge that is available at each step.

To understand why a successive cancellation decoding scheme solves the issue with the unknown channel outputs U_1^{k-1}, consider the (very theoretic) idea of a *genie-aided successive cancellation decoding scheme*.

A *genie* is a fictional fellow that helps the decoder by providing some information for free. Again, this is not realistic, but serves as an upper bound on the performance of a realistic decoder. In our case we assume that a genie

provides the decoder with the information of the *correct* values of U_k, but only *after* the decoder has decided on $\hat{U}_k$.

So, our fictional decoder consists of n decoding functions ψ_k, $k = 1, \ldots, n$:

$$\begin{aligned}
\hat{U}_1 &= \psi_1(Y_1, \ldots, Y_n), && \text{(14.77a)}\\
\hat{U}_2 &= \psi_2(Y_1, \ldots, Y_n, U_1), && \text{(14.77b)}\\
\hat{U}_3 &= \psi_3(Y_1, \ldots, Y_n, U_1, U_2), && \text{(14.77c)}\\
&\vdots\\
\hat{U}_n &= \psi_n(Y_1, \ldots, Y_n, U_1, \ldots, U_{n-1}), && \text{(14.77d)}
\end{aligned}$$

where the values of U_1^{n-1} at the receiver are provided by the genie.

On the other hand, while a realistic successive cancellation decoding scheme has no access to the *correct* values of U_1^{n-1}, it still can use the values that it has decoded already. Hence, using exactly the same decoding functions as for the genie-aided successive cancellation decoder (14.77), we define the following *realistic* successive cancellation decoding scheme:

$$\begin{aligned}
\hat{U}_1 &= \psi_1(Y_1, \ldots, Y_n), && \text{(14.78a)}\\
\hat{U}_2 &= \psi_2(Y_1, \ldots, Y_n, \hat{U}_1), && \text{(14.78b)}\\
\hat{U}_3 &= \psi_3(Y_1, \ldots, Y_n, \hat{U}_1, \hat{U}_2), && \text{(14.78c)}\\
&\vdots\\
\hat{U}_n &= \psi_n(Y_1, \ldots, Y_n, \hat{U}_1, \ldots, \hat{U}_{n-1}). && \text{(14.78d)}
\end{aligned}$$

In contrast to (14.77), here we have the problem of error propagation: if some decision $\hat{U}_k$ is wrong, it will negatively influence the remaining decoding decisions. But note that *if* the genie-aided decoder makes *no* mistake, then $\hat{U}_k = U_k$ in each round of the successive cancellation decoding and then neither does the realistic successive cancellation decoder (14.78) make any mistake!

On the other hand, if the genie-aided decoder makes some mistake, then the realistic successive cancellation decoder will make even more mistakes. However, we do not care how many mistakes we make, but we only worry whether we are correct or not, i.e., we only worry about the probability of making no error. Thus, the performance of the realistic decoder is actually identical to the genie-aided decoder, and hence our decoding scheme is realistic! We have proven the following lemma.

Lemma 14.12. The average error probability of a coding scheme with a genie-aided successive cancellation decoder is identical to a coding scheme with the same successive cancellation decoder without genie.

Note that the bit error rate of the two decoders given in (14.77) and (14.78) are different (since once a mistake happens, the realistic decoder will make

more errors than the genie-aided decoder due to error propagation). But once again: we are only interested in the *average codeword error* (or also called *block error*), which is identical for both decoders.

14.2.4 Polarization

We have already spoken about this main idea of *polarization* without actually properly defining or even proving it. Recall from Theorem 14.5 that applying the polar transform will result in a "guaranteed progress" in the sense of W^+ and W^- being strictly "better" and "worse", respectively, than W. To make this more precise, we give the following definitions.

Definition 14.13. For a fixed $\epsilon > 0$, we say that a binary-input DMC W is *ϵ-mediocre* if[2]

$$\epsilon < I(W) < 1 - \epsilon \tag{14.79}$$

If

$$I(W) \geq 1 - \epsilon \tag{14.80}$$

W is called *ϵ-good*, and if

$$I(W) \leq \epsilon \tag{14.81}$$

W is called *ϵ-bad*.

Let's return to Example 14.3.

Example 14.14 (BEC continued). We start with a mediocre BEC of erasure probability $\delta = 0.4$ and repeatedly apply the polar transform. In Figure 14.9, we plot the values of the channel mutual information I(W) for all channels after a specific number ℓ of recursive application of the polar transform. (Note that the channels are not numbered according to Proposition 14.7, but are sorted with respect to their channel mutual information.) We see that for large ℓ, roughly 40% of the channels have a channel mutual information very close to 0, and roughly 60% have a channel mutual information very close to 1 bit. There are only very few channels with a value in between. Thus, about 60% of the channels are good, about 40% of the channels are bad, and there remain almost no mediocre channels anymore. ◊

So we see that by repeatedly applying the polar transform we have succeeded in *polarizing* the channels, i.e., we end up with (almost) only perfect or useless channels.

More concretely, we have the following theorem.

[2] Recall that the mutual information is measured in bits.

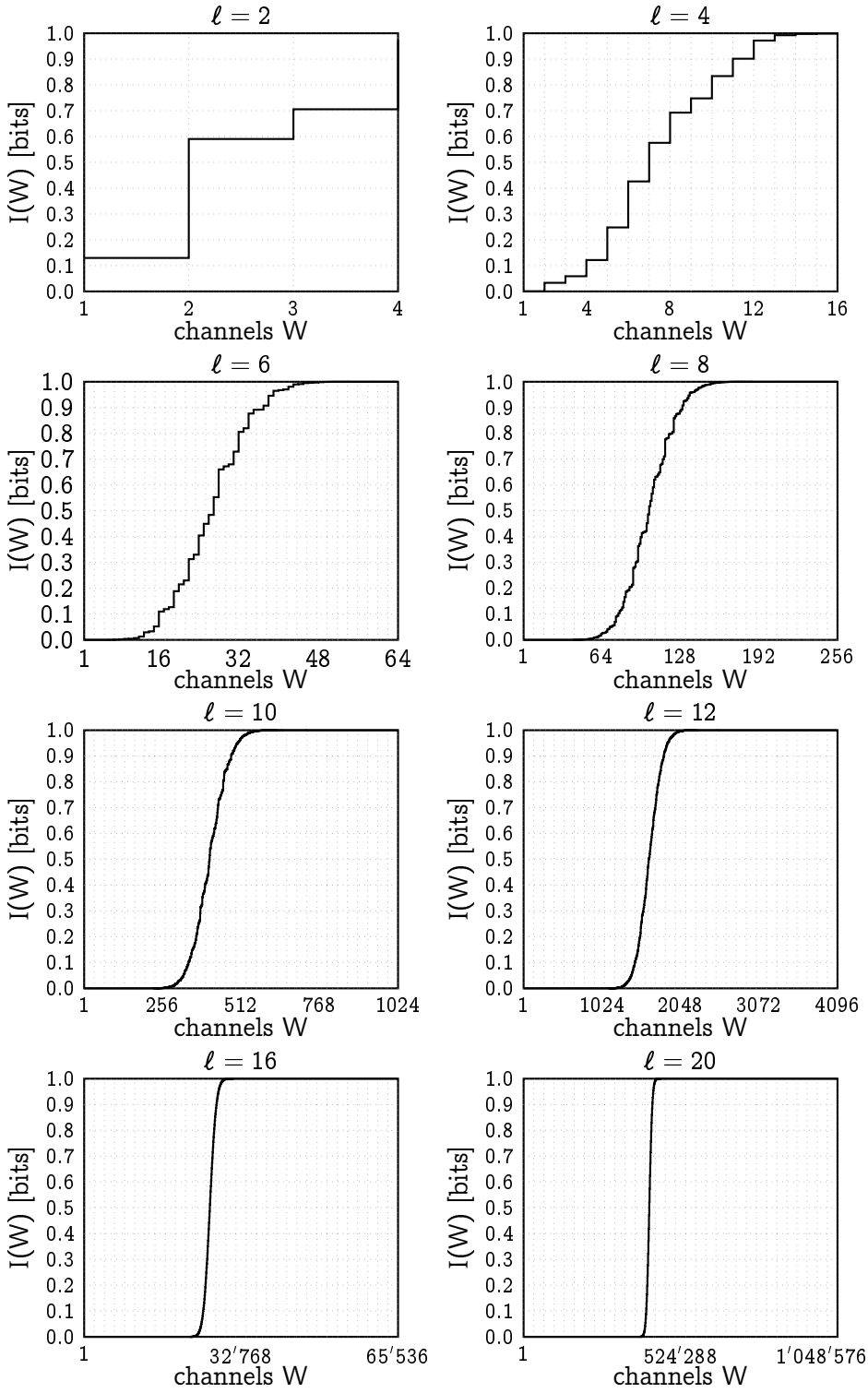


Figure 14.9: Polarization of $n = 2^\ell$ BECs of erasure probability $\delta = 0.4$ after ℓ recursive applications of the polar transform. The y-axis depicts the channel mutual information I(W) in bits, and the x-axis lists the channels that are sorted with respect to their channel mutual information.

Theorem 14.15 (Polarization Happens). For any binary-input DMC W and any $\epsilon > 0$, the fraction of ϵ-mediocre channels tends to zero as the number ℓ of polar transform applications gets large:

$$\lim_{\ell\to\infty} \frac{\big|\{s_1\cdots s_\ell\colon\ \mathrm{I}(\mathsf{W}^{s_1\cdots s_\ell}) \in (\epsilon, 1-\epsilon)\}\big|}{2^\ell} = 0. \tag{14.82}$$

Moreover, the fraction of ϵ-good channels tends to $\mathrm{I}(\mathsf{W})$,

$$\lim_{\ell\to\infty} \frac{\big|\{s_1\cdots s_\ell\colon\ \mathrm{I}(\mathsf{W}^{s_1\cdots s_\ell}) \in [1-\epsilon, 1]\}\big|}{2^\ell} = \mathrm{I}(\mathsf{W}), \tag{14.83}$$

and the fraction of ϵ-bad channels tends to $1-\mathrm{I}(\mathsf{W})$,

$$\lim_{\ell\to\infty} \frac{\big|\{s_1\cdots s_\ell\colon\ \mathrm{I}(\mathsf{W}^{s_1\cdots s_\ell}) \in [1-\epsilon, 1]\}\big|}{2^\ell} = 1-\mathrm{I}(\mathsf{W}). \tag{14.84}$$

We postpone the proof of this theorem to Section 14.2.5, but first investigate the special case of a BEC that is much easier to handle because for a BEC there exists a direct connection between $\mathrm{I}(\mathsf{W})$ and the probability of error.

Recall that if W is a BEC with erasure probability δ, then its children W^+ and W^- are again BECs with erasure probabilities

$$\delta^+ = \delta^2, \tag{14.85a}$$
$$\delta^- = 2\delta - \delta^2, \tag{14.85b}$$

respectively. Moreover, since for a BEC

$$\mathrm{I}(\mathsf{W}) = 1-\delta, \tag{14.86}$$

a BEC is ϵ-mediocre if, and only if, its erasure probability δ satisfies $\epsilon < \delta < 1-\epsilon$.

To help with our investigation of the number of ϵ-mediocre BECs, we next define the *ugliness* of a BEC.

Definition 14.16. The *ugliness* of a BEC W with erasure probability δ is defined as

$$\mu(\mathsf{W}) \triangleq \sqrt{4\delta(1-\delta)}. \tag{14.87}$$

Note that the ugliness is normalized such that it is a number between 0 and 1, i.e., a "maximally ugly" BEC has an ugliness of 1. Also note that the motivation behind this definition is that a BEC with an erasure probability δ close to 0 or close to 1 is not ugly, while an ugly BEC has a δ around $\frac{1}{2}$.

Now, by straightforward calculation, we see that

$$\begin{aligned}
\mu(\mathsf{W}^+) &= \sqrt{4\delta^+(1-\delta^+)} && (14.88)\\
&= \sqrt{4\delta^2(1-\delta^2)} && (14.89)\\
&= \sqrt{4\delta\delta(1+\delta)(1-\delta)} && (14.90)\\
&= \sqrt{4\delta(1-\delta)}\sqrt{\delta(1+\delta)} && (14.91)\\
&= \mu(\mathsf{W})\sqrt{\delta(1+\delta)}, && (14.92)\\
\mu(\mathsf{W}^-) &= \sqrt{4\delta^-(1-\delta^-)} && (14.93)\\
&= \sqrt{4(2\delta-\delta^2)(1-2\delta+\delta^2)} && (14.94)\\
&= \sqrt{4\delta(2-\delta)(1-\delta)^2} && (14.95)\\
&= \mu(\mathsf{W})\sqrt{(2-\delta)(1-\delta)}. && (14.96)
\end{aligned}$$

Thus, nothing can be said about the individual ugliness of a child channel of a BEC: a child can be less ugly or more ugly than its parent. On average, however, the ugliness of the children channels is reduced:

$$\begin{aligned}
\frac{1}{2}\mu(\mathsf{W}^+) + \frac{1}{2}\mu(\mathsf{W}^-) &= \mu(\mathsf{W})\frac{1}{2}\Big(\sqrt{\delta(1+\delta)} + \sqrt{(2-\delta)(1-\delta)}\Big) && (14.97)\\
&\le \mu(\mathsf{W})\sqrt{\frac{3}{4}}, && (14.98)
\end{aligned}$$

where the inequality follows from the fact that the function $\delta \mapsto \sqrt{\delta(1+\delta)} + \sqrt{(2-\delta)(1-\delta)}$ is strictly concave and symmetric around $\frac{1}{2}$, i.e., it achieves its maximum for $\delta = \frac{1}{2}$.

By recursive application of (14.98) to the children of a BEC, we thus obtain

$$\frac{1}{2^\ell}\sum_{(s_1\cdots s_\ell)\in\{+,-\}^\ell} \mu(\mathsf{W}^{s_1\cdots s_\ell}) \le \mu(\mathsf{W})\left(\frac{3}{4}\right)^{\frac{\ell}{2}}. \tag{14.99}$$

Also note that for an arbitrary child channel of ℓ applications of the polar transform to a BEC,

$$\begin{aligned}
\mu(\mathsf{W}^{s_1\cdots s_\ell}) &= \sqrt{4\delta^{s_1\cdots s_\ell}(1-\delta^{s_1\cdots s_\ell})} && (14.100)\\
&\ge \mathbb{1}\{\delta^{s_1\cdots s_\ell}\in(\epsilon,1-\epsilon)\}\cdot \min_{\delta^{s_1\cdots s_\ell}\in(\epsilon,1-\epsilon)} \sqrt{4\delta^{s_1\cdots s_\ell}(1-\delta^{s_1\cdots s_\ell})}\\
&\quad + \mathbb{1}\{\delta^{s_1\cdots s_\ell}\notin(\epsilon,1-\epsilon)\}\cdot \min_{\delta^{s_1\cdots s_\ell}\notin(\epsilon,1-\epsilon)} \sqrt{4\delta^{s_1\cdots s_\ell}(1-\delta^{s_1\cdots s_\ell})} && (14.101)\\
&= \mathbb{1}\{\delta^{s_1\cdots s_\ell}\in(\epsilon,1-\epsilon)\}\cdot\sqrt{4\epsilon(1-\epsilon)}, && (14.102)
\end{aligned}$$

where the first minimum is achieved for $\delta^{s_1\cdots s_\ell} = \epsilon$ and the second for $\delta^{s_1\cdots s_\ell} = 0$.

We now obtain the following chain of (in)equalities:

$$\frac{\bigl|\{s_1\cdots s_\ell\colon\ \mathrm{I}(\mathsf{W}^{s_1\cdots s_\ell})\in(\epsilon,1-\epsilon)\}\bigr|}{2^\ell}$$

$$=\frac{1}{2^\ell}\sum_{(s_1\cdots s_\ell)\in\{+,-\}^\ell}\mathbb{1}\{\mathrm{I}(\mathsf{W}^{s_1\cdots s_\ell})\in(\epsilon,1-\epsilon)\} \tag{14.103}$$

$$=\frac{1}{2^\ell}\sum_{(s_1\cdots s_\ell)\in\{+,-\}^\ell}\mathbb{1}\{\delta^{s_1\cdots s_\ell}\in(\epsilon,1-\epsilon)\} \tag{14.104}$$

$$\le\frac{1}{2^\ell}\sum_{(s_1\cdots s_\ell)\in\{+,-\}^\ell}\frac{\mu(\mathsf{W}^{s_1\cdots s_\ell})}{\sqrt{4\epsilon(1-\epsilon)}} \tag{14.105}$$

$$\le\frac{\mu(\mathsf{W})}{\sqrt{4\epsilon(1-\epsilon)}}\biggl(\frac{3}{4}\biggr)^{\frac{\ell}{2}} \tag{14.106}$$

$$=\frac{\sqrt{4\delta(1-\delta)}}{\sqrt{4\epsilon(1-\epsilon)}}\biggl(\frac{3}{4}\biggr)^{\frac{\ell}{2}}\overset{\ell\to\infty}{\to}0. \tag{14.107}$$

Here, the first inequality follows from (14.102), and the second from (14.99). This proves (14.82) (for the special case of a BEC).

Since in the limit when ℓ tends to infinity, no mediocre channels are left, the remaining channels must all be good or bad. Now note that the polar transform preserves the average erasure probability (see (14.85) or, related to this, Theorem 14.5):

$$\frac{1}{2}\delta^+ + \frac{1}{2}\delta^- = \delta. \tag{14.108}$$

Thus, as in the limit we end up with only BECs of erasure probability 0 or 1, the fraction of useless channels must be δ, and the fraction of perfect channels must be $1-\delta$. This proves (14.83) and (14.84) for the BEC.

14.2.5 Proof of Theorem 14.15

We prove the theorem by formulating it in a probabilistic manner: We claim that if out of the 2^ℓ channels we pick one channel uniformly at random, then with a probability that tends to 1 as ℓ gets large the channel mutual information is either close to 1 or close to 0.

We start by describing the recursive application process of the polar transform by a tree of channels as shown in Figure 14.10. We now randomly select a channel $\mathsf{W}^{s_1\cdots s_\ell}$ by following a random walk along the tree: at every fork we flip a fair coin to either go up $(+)$ or down $(-)$.

Concretely, let $\{S_\ell\}_{\ell\in\mathbb{N}}$ be a sequence of RVs that are IID $\sim\mathcal{U}(\{+,-\})$ and let $\mathbb{W}_\ell$ $(\ell=0,1,2,\ldots)$ denote the channel after the ℓth fork:

$$\mathbb{W}_0\triangleq\mathsf{W}, \tag{14.109}$$

$$\mathbb{W}_\ell=\mathbb{W}_{\ell-1}^{S_\ell} \tag{14.110}$$

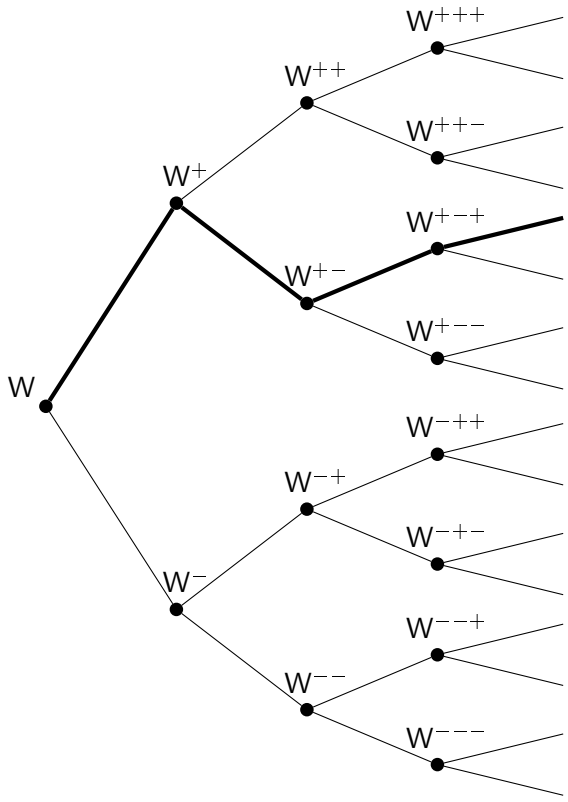


Figure 14.10: Tree of recursive applications of the polar transform.

$$= \begin{cases} \mathbb{W}_{\ell-1}^{+} & \text{if } S_\ell = +, \\ \mathbb{W}_{\ell-1}^{-} & \text{if } S_\ell = -, \end{cases} \quad \ell = 1, 2, \ldots \tag{14.111}$$

For example, for the bold path depicted in Figure 14.10 we have

$$\{S_1, S_2, S_3, S_4, \ldots\} = \{+, -, +, +, \ldots\} \tag{14.112}$$

and hence

$$\begin{aligned} &\mathbb{W}_0 = \mathsf{W}, && \mathbb{W}_1 = \mathsf{W}^{+}, && \mathbb{W}_2 = \mathsf{W}^{+-}, \\ &\mathbb{W}_3 = \mathsf{W}^{+-+}, && \mathbb{W}_4 = \mathsf{W}^{+-++}, && \ldots \end{aligned} \tag{14.113}$$

Moreover, we define the random process $\{I_\ell\}$ by setting

$$I_\ell \triangleq \mathsf{I}(\mathbb{W}_\ell). \tag{14.114}$$

We now note that, conditionally on a fixed start of a path $(\mathbb{W}_0, \ldots, \mathbb{W}_{\ell-1}) = (\mathsf{W}_0, \ldots, \mathsf{W}_{\ell-1})$, the expected value of I_ℓ only depends on $I_{\ell-1}$:

$$\begin{aligned} &\mathsf{E}[\mathsf{I}(\mathbb{W}_\ell) \,|\, (\mathbb{W}_0, \ldots, \mathbb{W}_{\ell-1}) = (\mathsf{W}_0, \ldots, \mathsf{W}_{\ell-1})] \\ &\quad = \Pr[S_\ell = +] \cdot \mathsf{I}(\mathsf{W}_{\ell-1}^{+}) + \Pr[S_\ell = -] \cdot \mathsf{I}(\mathsf{W}_{\ell-1}^{-}) \end{aligned} \tag{14.115}$$

$$= \frac{1}{2}\,\mathrm{I}(\mathsf{W}_{\ell-1}^{+}) + \frac{1}{2}\,\mathrm{I}(\mathsf{W}_{\ell-1}^{-}) \tag{14.116}$$
$$= \mathrm{I}(\mathsf{W}_{\ell-1}), \tag{14.117}$$

where the last equality follows from the first property in Theorem 14.5. Hence,

$$\mathsf{E}[I_\ell \,|\, I_0, \ldots, I_{\ell-1}] = I_{\ell-1} \tag{14.118}$$

or

$$\mathsf{E}[I_\ell - I_{\ell-1} \,|\, I_0, \ldots, I_{\ell-1}] = 0. \tag{14.119}$$

Using (14.119) with ℓ replaced by k, we now obtain for any $j < k$:

$$0 = 0 \cdot (I_j - I_{j-1}) \tag{14.120}$$
$$= \mathsf{E}[I_k - I_{k-1} \,|\, I_1, \ldots, I_{k-1}] \cdot (I_j - I_{j-1}) \qquad \text{(by (14.119))} \tag{14.121}$$
$$= \mathsf{E}[(I_k - I_{k-1}) \cdot (I_j - I_{j-1}) \,|\, I_0, \ldots, I_{k-1}], \qquad \text{(because } j < k\text{)} \tag{14.122}$$

and by taking the expectation over $I_0, \ldots, I_{k-1}$ hence also

$$\mathsf{E}[(I_k - I_{k-1}) \cdot (I_j - I_{j-1})] = 0, \quad j \neq k, \tag{14.123}$$

i.e., the differences $I_\ell - I_{\ell-1}$ are uncorrelated.

Now note that since $I_\ell \in [0,1]$ (we are lazy and omit the unit of bits), we have

$$1 \geq (I_\ell - I_0)^2 \tag{14.124}$$
$$= \left(\sum_{j=1}^{\ell} (I_j - I_{j-1})\right)^2 \tag{14.125}$$
$$= \left(\sum_{j=1}^{\ell} (I_j - I_{j-1})\right)\left(\sum_{j'=1}^{\ell} (I_{j'} - I_{j'-1})\right) \tag{14.126}$$
$$= \sum_{j=1}^{\ell}\sum_{j'=1}^{\ell} (I_j - I_{j-1})(I_{j'} - I_{j'-1}). \tag{14.127}$$

Here (14.125) follows from a telescoping sum:

$$I_\ell - I_0 = I_\ell - I_{\ell-1} + I_{\ell-1} - I_{\ell-2} + I_{\ell-2} - \cdots - I_1 + I_1 - I_0. \tag{14.128}$$

Taking expectation on both sides of (14.127) and using the uncorrelatedness (14.123) now yields

$$1 \geq \sum_{j=1}^{\ell}\sum_{j'=1}^{\ell} \mathsf{E}[(I_j - I_{j-1})(I_{j'} - I_{j'-1})] \tag{14.129}$$
$$= \sum_{j=1}^{\ell} \mathsf{E}\Big[(I_j - I_{j-1})^2\Big] \tag{14.130}$$
$$\triangleq \sum_{j=1}^{\ell} \sigma_j^2, \tag{14.131}$$

where the last equality should be read as definition of σ_j^2. This holds for any ℓ, i.e.,

$$\lim_{\ell\to\infty}\sum_{j=1}^{\ell}\sigma_j^2 \le 1, \tag{14.132}$$

which means that

$$\lim_{j\to\infty}\sigma_j^2 = 0. \tag{14.133}$$

Combined with the Markov Inequality (Lemma 20.2) this now allows us to show that for any $\epsilon > 0$

$$\begin{aligned}
\lim_{j\to\infty}\Pr\big[\big|I_j - I_{j-1}\big| > \epsilon\big] &= \lim_{j\to\infty}\Pr\big[|I_j - I_{j-1}|^2 > \epsilon^2\big] && (14.134)\\
&\le \lim_{j\to\infty}\frac{\mathsf{E}\big[(I_j - I_{j-1})^2\big]}{\epsilon^2} && (14.135)\\
&= \lim_{j\to\infty}\frac{\sigma_j^2}{\epsilon^2} && (14.136)\\
&= 0. && (14.137)
\end{aligned}$$

We conclude that the random sequence $\{I_j\}$ must converge.[3]

Note that in general a converging random sequence converges not to a constant, but to a random variable.[4] Let I_∞ be this RV. From (14.118) it follows that

$$\mathsf{E}[I_\ell] = \mathsf{E}[I_{\ell-1}], \quad \ell \in \mathbb{N}, \tag{14.138}$$

and hence

$$\mathsf{E}[I_\infty] = \mathsf{E}[I_0] = \mathrm{I}(\mathsf{W}). \tag{14.139}$$

Moreover, because of the guaranteed progress property (Theorem 14.5), I_∞ can only take value in the extremes: $I_\infty \in \{0,1\}$.

Finally, by explicitly computing the expectation (14.139):

$$\begin{aligned}
\mathrm{I}(\mathsf{W}) &= \mathsf{E}[I_\infty] && (14.140)\\
&= \Pr[I_\infty = 1]\cdot 1 + \Pr[I_\infty = 0]\cdot 0 && (14.141)\\
&= \Pr[I_\infty = 1] && (14.142)
\end{aligned}$$

we have derived the exact distribution of I_∞.

Note that the above proof introduces the important concept of a *random walk* in the tree of channels that is created by recursively applying the polar transform. The channel mutual information parameters $\{I_\ell\}$ of such a walk is a sequence of RVs that converges to I_∞, which is a binary RV taking value only in $\{0,1\}$.

[3] Readers who know about martingales might have realized that $\{I_j\}$ forms a martingale and that therefore the convergence could be directly argued based on the properties of martingales.

[4] For a discussion of convergence of random sequences, see Section 20.2.

14.2.6 Attempt on a Polar Coding Scheme for the BEC

Based on the insight from Theorem 14.15 that the channels polarize into good and bad channels only, we come up with the following simple coding scheme for a BEC with erasure probability δ:

- Fix a rate $\mathrm{R} < 1-\delta$ and a blocklength $n = 2^\ell$.
- We start with ℓ applications of the polar transform to $n = 2^\ell$ copies of the BEC W and obtain n children channels $\mathrm{W}^{-\cdots-}, \ldots, \mathrm{W}^{+\cdots+}$.
- We pick the $n\mathrm{R}$ best channels and send uncoded data over them. The inputs of the remaining channels are frozen to 0.
- Since the number of ϵ-good channels is approximately $n(1-\delta)$, all the $n\mathrm{R} < n(1-\delta)$ channels with uncoded data at the input are good.
- At the receiver, we successively decode $U_1, \ldots, U_n$ (where of course we are only really interested in those inputs that are not frozen).
- The block error probability of this scheme is upper-bounded by

$$\begin{aligned} P_{\mathrm{e}}^{(n)} &= \sum_{k\colon \text{uncoded data}} \delta_n^{(k)} + \sum_{k\colon \text{frozen bits}} 0 && (14.143)\\ &\le \sum_{k\colon \text{uncoded data}} \epsilon && (14.144)\\ &= n\mathrm{R}\epsilon, && (14.145) \end{aligned}$$

  where $\delta_n^{(k)}$ denotes the erasure probability of channel $\mathrm{W}_n^{(k)}$ and where the inequality follows from the fact that all channels with uncoded data at the input are ϵ-good.

And now we are quite a bit disappointed: even if ϵ is small, $n\mathrm{R}\epsilon$ might be large! It is not clear whether our scheme works or not...

We see that it is not sufficient to know that polarization happens, but we also need to know how fast this polarization occurs! In order to be able to investigate the polarization speed, we need to introduce one more fundamental quantity: the channel reliability.

14.3 Channel Reliability

All our investigations so far focused completely on the channel mutual information I(W), which is a very fundamental property of a DMC. Unfortunately, it is quite costly to track this value for all channels that result from recursive application of the polar transform and, moreover, (except for the special case of a BEC) it is not directly clear how I(W) is linked to the *probability of error*. Thus, we will next introduce a companion to the channel mutual information I(W): the channel reliability parameter.

Definition 14.17. For a binary-input DMC W we define the *channel reliability parameter* $Z(W)$ as

$$Z(W) \triangleq \sum_y \sqrt{W(y|0)\,W(y|1)}. \tag{14.146}$$

Remark 14.18. Note that the reliability parameter actually is the *exponentiated Bhattacharyya distance* defined in Definition 11.20. Hence, we know from Corollary 11.21 that $Z(W)$ is an upper bound on the error probability if we use two codewords that differ in only one position. Particularly, since for uncoded transmission any sequence is a codeword and thus the minimum distance indeed is 1, we understand that $Z(W)$ is an upper bound on the error probability for uncoded transmission. We will prove this formally in Theorem 14.27 in Section 14.4. △

In this section we will now investigate $Z(W)$. In particular, we would like to understand its connections to $I(W)$ and its reactions to applying the polar transform.

Lemma 14.19. The channel reliability parameter is a number between 0 and 1,

$$0 \le Z(W) \le 1, \tag{14.147}$$

where 0 stands for *fully reliable* (corresponding to the perfect channel) and 1 for *not reliable at all* (corresponding to the useless channel).

Proof: The nonnegativity follows directly from its definition. The upper bound follows from the Cauchy–Schwarz Inequality [Lap17, Theorem 3.3.1]:

$$\sum_y \sqrt{W(y|0)}\sqrt{W(y|1)} \le \sqrt{\sum_y W(y|0)}\sqrt{\sum_y W(y|1)} = 1 \cdot 1 = 1. \tag{14.148}$$

The fact that $Z(W) = 0$ corresponds to the perfect channel and $Z(W) = 1$ to the useless channel will be proven below in Corollary 14.22. □

Theorem 14.20 (Channel Reliability Parameter and the Polar Transform).
For any DMC W, the channel reliability parameter of W and its children channels W^+ and W^- satisfy the following:

1. It holds

$$Z(W^+) = Z^2(W), \tag{14.149}$$
$$Z(W^-) \le 2Z(W) - Z^2(W), \tag{14.150}$$

and thus by applying the polar transform the total reliability can

only improve:

$$\mathsf{Z}(\mathsf{W}^-) + \mathsf{Z}(\mathsf{W}^+) \le 2\mathsf{Z}(\mathsf{W}). \tag{14.151}$$

We have equality in (14.150) and (14.151) if, and only if, W is a BEC.

2. We have "guaranteed progress" in the polarization unless the channel is already extremal:

$$\mathsf{Z}(\mathsf{W}^+) \le \mathsf{Z}(\mathsf{W}) \le \mathsf{Z}(\mathsf{W}^-) \tag{14.152}$$

with equality if, and only if, W is either perfect ($\mathsf{Z}(\mathsf{W}) = 0$) or useless ($\mathsf{Z}(\mathsf{W}) = 1$).

Proof: The identity (14.149) follows from straightforward calculation:

$$\begin{aligned}
&\mathsf{Z}(\mathsf{W}^+)\\
&= \sum_{y_1}\sum_{y_2}\sum_{u_1} \sqrt{W^+(y_1,y_2,u_1|0)\,W^+(y_1,y_2,u_1|1)} && (14.153)\\
&= \sum_{y_1}\sum_{y_2}\sum_{u_1} \sqrt{P_{U_1}(u_1)\,W(y_1|u_1)\,W(y_2|0)\cdot P_{U_1}(u_1)\,W(y_1|u_1\oplus 1)\,W(y_2|1)} && (14.154)\\
&= \sum_{y_1}\sum_{y_2}\sum_{u_1} P_{U_1}(u_1)\sqrt{W(y_1|u_1)\,W(y_1|u_1\oplus 1)}\sqrt{W(y_2|0)\,W(y_2|1)} && (14.155)\\
&= \sum_{u_1} P_{U_1}(u_1)\sum_{y_1}\sqrt{W(y_1|0)\,W(y_1|1)}\sum_{y_2}\sqrt{W(y_2|0)\,W(y_2|1)} && (14.156)\\
&= 1\cdot \mathsf{Z}(\mathsf{W})\cdot\mathsf{Z}(\mathsf{W}) && (14.157)\\
&= \mathsf{Z}^2(\mathsf{W}). && (14.158)
\end{aligned}$$

For (14.150), we write

$$\begin{aligned}
\mathsf{Z}(\mathsf{W}^-) &= \sum_{y_1}\sum_{y_2}\sqrt{W^-(y_1,y_2|0)\,W^-(y_1,y_2|1)} && (14.159)\\
&= \sum_{y_1}\sum_{y_2}\sqrt{\Big(\frac{1}{2}W(y_1|0)\,W(y_2|0)+\frac{1}{2}W(y_1|1)\,W(y_2|1)\Big)}\\
&\qquad\cdot\sqrt{\Big(\frac{1}{2}W(y_1|1)\,W(y_2|0)+\frac{1}{2}W(y_1|0)\,W(y_2|1)\Big)} && (14.160)\\
&= \frac{1}{2}\sum_{y_1}\sum_{y_2}\sqrt{(\alpha\gamma+\beta\delta)(\beta\gamma+\alpha\delta)}, && (14.161)
\end{aligned}$$

where we have introduced the shorthands

$$\alpha \triangleq W(y_1|0), \qquad \beta \triangleq W(y_1|1), \tag{14.162}$$

$$\gamma \triangleq W(y_2|0), \qquad \delta \triangleq W(y_2|1). \tag{14.163}$$

Now note that

$$\begin{aligned}
&\Big((\sqrt{\alpha\gamma}+\sqrt{\beta\delta})(\sqrt{\beta\gamma}+\sqrt{\alpha\delta})-2\sqrt{\alpha\beta\gamma\delta}\Big)^2\\
&\quad=(\alpha\gamma+\beta\delta+2\sqrt{\alpha\beta\gamma\delta})(\beta\gamma+\alpha\delta+2\sqrt{\alpha\beta\gamma\delta})\\
&\qquad-4\sqrt{\alpha\beta\gamma\delta}\,(\gamma\sqrt{\alpha\beta}+\alpha\sqrt{\gamma\delta}+\beta\sqrt{\gamma\delta}+\delta\sqrt{\alpha\beta})+4\alpha\beta\gamma\delta && (14.164)\\
&\quad=(\alpha\gamma+\beta\delta)(\beta\gamma+\alpha\delta)+8\alpha\beta\gamma\delta\\
&\qquad+2\sqrt{\alpha\beta\gamma\delta}\Big(\alpha\gamma+\beta\delta+\beta\gamma+\alpha\delta-2(\gamma+\delta)\sqrt{\alpha\beta}-2(\alpha+\beta)\sqrt{\gamma\delta}\Big) && (14.165)\\
&\quad=(\alpha\gamma+\beta\delta)(\beta\gamma+\alpha\delta)\\
&\qquad+2\sqrt{\alpha\beta\gamma\delta}\Big((\alpha+\beta)(\gamma+\delta)-2(\gamma+\delta)\sqrt{\alpha\beta}-2(\alpha+\beta)\sqrt{\gamma\delta}+4\sqrt{\alpha\beta\gamma\delta}\Big) && (14.166)\\
&\quad=(\alpha\gamma+\beta\delta)(\beta\gamma+\alpha\delta)+2\sqrt{\alpha\beta\gamma\delta}\,\big(\sqrt{\alpha}-\sqrt{\beta}\big)^2\big(\sqrt{\gamma}-\sqrt{\delta}\big)^2 && (14.167)\\
&\quad\geq(\alpha\gamma+\beta\delta)(\beta\gamma+\alpha\delta), && (14.168)
\end{aligned}$$

and thus

$$\begin{aligned}
\mathsf{Z}(\mathsf{W}^-)&\leq\frac{1}{2}\sum_{y_1}\sum_{y_2}\Big((\sqrt{\alpha\gamma}+\sqrt{\beta\delta})(\sqrt{\beta\gamma}+\sqrt{\alpha\delta})-2\sqrt{\alpha\beta\gamma\delta}\Big) && (14.169)\\
&=\frac{1}{2}\sum_{y_1}\sum_{y_2}\Big(\gamma\sqrt{\alpha\beta}+\alpha\sqrt{\gamma\delta}+\beta\sqrt{\gamma\delta}+\delta\sqrt{\alpha\beta}-2\sqrt{\alpha\beta\gamma\delta}\Big) && (14.170)\\
&=\frac{1}{2}\big(\mathsf{Z}(\mathsf{W})+\mathsf{Z}(\mathsf{W})+\mathsf{Z}(\mathsf{W})+\mathsf{Z}(\mathsf{W})-2\mathsf{Z}^2(\mathsf{W})\big) && (14.171)\\
&=2\mathsf{Z}(\mathsf{W})-\mathsf{Z}^2(\mathsf{W}). && (14.172)
\end{aligned}$$

The left inequality in (14.152) follows directly from (14.149) and the fact that $\mathsf{Z}(\mathsf{W})\in[0,1]$. From this one also immediately sees that equality can only be achieved if $\mathsf{Z}(\mathsf{W})=0$ or $\mathsf{Z}(\mathsf{W})=1$.

The derivation of the right inequality in (14.152) is more involved and is postponed to Appendix 14.B. Note that if we have equality on the left in (14.152), then (14.151) shows that $\mathsf{Z}(\mathsf{W}^-)\leq\mathsf{Z}(\mathsf{W})$, which is only possible if we have equality $\mathsf{Z}(\mathsf{W}^-)=\mathsf{Z}(\mathsf{W})$. Hence, we achieve equality in both inequalities in (14.152) if, and only if, $\mathsf{Z}(\mathsf{W})$ equals 0 or 1. □

The alert reader has already recognized the similarities between Theorem 14.5 and Theorem 14.20. It seems quite intuitive that $\mathsf{Z}(\mathsf{W})\approx 0$ if, and only if, $\mathsf{I}(\mathsf{W})\approx 1$ bit, and that $\mathsf{Z}(\mathsf{W})\approx 1$ if, and only if, $\mathsf{I}(\mathsf{W})\approx 0$. To get a better understanding of the connection between these two channel parameters, we are next going to derive some more bounds.

Proposition 14.21 (Connection between $\mathsf{I}(\mathsf{W})$ and $\mathsf{Z}(\mathsf{W})$). For any DMC W,

$$1-\mathsf{Z}(\mathsf{W})\leq\mathsf{I}(\mathsf{W})\leq 1-\mathsf{Z}^2(\mathsf{W}) \tag{14.173}$$

or

$$1-\mathsf{I}(\mathsf{W})\leq\mathsf{Z}(\mathsf{W})\leq\sqrt{1-\mathsf{I}(\mathsf{W})}. \tag{14.174}$$

(Recall that I(W) is measured in bits.) Note that equality is achieved on the left if W is a BEC.

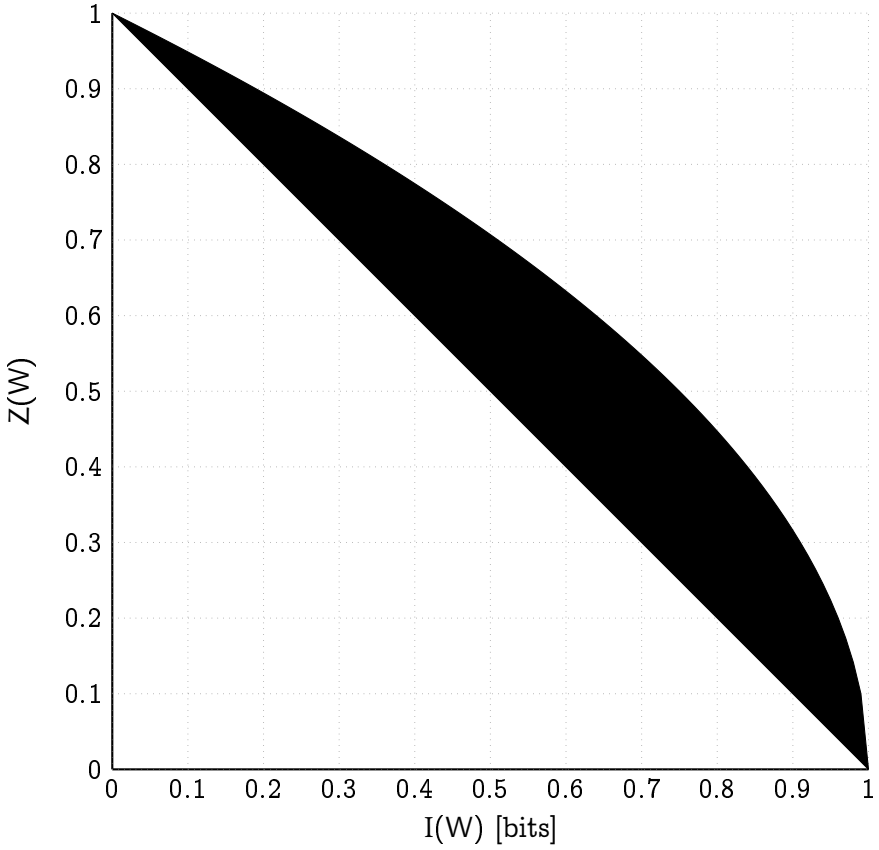


Figure 14.11: Possible values of $(\mathsf{I}(\mathsf{W}), \mathsf{Z}(\mathsf{W}))$. We see that the range is quite close to the diagonal $\mathsf{Z}(\mathsf{W}) = 1 - \mathsf{I}(\mathsf{W})$ (which is achieved for a BEC). In particular, $\mathsf{I}(\mathsf{W})$ is extremal if, and only if, $\mathsf{Z}(\mathsf{W})$ is extremal.

Proof: We start by recalling that since we assume a uniform binary input,

$$P_{X,Y}(x,y) = P_X(x) \cdot P_{Y|X}(y|x) = \frac{1}{2} W(y|x), \quad y \in \mathcal{Y},\ x \in \{0,1\}. \tag{14.175}$$

Thus,

$$\mathsf{Z}(\mathsf{W}) = \sum_y \sqrt{W(y|0)\, W(y|1)} \tag{14.176}$$

$$= 2 \sum_y \sqrt{\frac{1}{2} W(y|0)\, \frac{1}{2} W(y|1)} \tag{14.177}$$

$$= 2 \sum_y \sqrt{P_{X,Y}(0,y)\, P_{X,Y}(1,y)} \tag{14.178}$$

$$= 2\sum_y \sqrt{P_Y(y)\,P_{X|Y}(0|y)\,P_Y(y)\,P_{X|Y}(1|y)} \tag{14.179}$$

$$= \sum_y P_Y(y)\cdot 2\sqrt{P_{X|Y}(0|y)\,P_{X|Y}(1|y)} \tag{14.180}$$

$$= \mathsf{E}\Big[2\sqrt{P_{X|Y}(0|Y)\,P_{X|Y}(1|Y)}\Big] \tag{14.181}$$

$$= \mathsf{E}\Big[2\sqrt{p(Y)\,(1-p(Y))}\Big], \tag{14.182}$$

where we have introduced the shorthand

$$p(y) \triangleq P_{X|Y}(0|y), \quad y \in \mathcal{Y}. \tag{14.183}$$

Similarly,

$$\mathsf{I}(\mathsf{W}) = \mathsf{H}(X) - \mathsf{H}(X|Y) \tag{14.184}$$

$$= 1 - \mathsf{E}[\mathsf{H}(X|Y=y)] \text{ bits} \tag{14.185}$$

$$= 1 - \mathsf{E}\big[\mathsf{H}_{\mathsf{b}}(p(Y))\big] \text{ bits}. \tag{14.186}$$

We are now ready to prove the right inequality in (14.173), i.e., we need to show that

$$1 - \mathsf{Z}^2(\mathsf{W}) - \mathsf{I}(\mathsf{W}) \geq 0. \tag{14.187}$$

Note that by (14.182) and (14.186),

$$1 - \mathsf{I}(\mathsf{W}) - \mathsf{Z}^2(\mathsf{W})$$

$$= 1 - 1 + \mathsf{E}\big[\mathsf{H}_{\mathsf{b}}(p(Y))\big] - \Big(\mathsf{E}\Big[2\sqrt{p(Y)(1-p(Y))}\Big]\Big)^2 \tag{14.188}$$

$$\geq \mathsf{E}\big[\mathsf{H}_{\mathsf{b}}(p(Y))\big] - \mathsf{E}\Big[\Big(2\sqrt{p(Y)(1-p(Y))}\Big)^2\Big] \tag{14.189}$$

$$= \mathsf{E}\Big[\mathsf{H}_{\mathsf{b}}(p(Y)) - 4p(Y)(1-p(Y))\Big], \tag{14.190}$$

where the inequality follows from the Jensen Inequality (Theorem 2.1) and the fact that $z \mapsto z^2$ is convex. Thus, (14.187) is certainly satisfied if we can show that

$$\mathsf{H}_{\mathsf{b}}(p) - 4p(1-p) \geq 0, \quad \forall\, p \in [0,1]. \tag{14.191}$$

To this end, we define

$$f(p) \triangleq -p\log_2 p - (1-p)\log_2(1-p) - 4p(1-p) \tag{14.192}$$

and compute

$$\frac{\partial f}{\partial p} = -\log_2 p + \log_2(1-p) - 4 + 8p, \tag{14.193}$$

$$\frac{\partial^2 f}{\partial p^2} = -\frac{1}{\ln 2}\Big(\frac{1}{p} + \frac{1}{1-p}\Big) + 8, \tag{14.194}$$

$$\frac{\partial^3 f}{\partial p^3} = \frac{1}{\ln 2}\Big(\frac{1}{p^2} - \frac{1}{(1-p)^2}\Big). \tag{14.195}$$

When inspecting the third-order derivative, we see that the first-order derivative is strictly convex for $p < \frac{1}{2}$, and strictly concave for $p > \frac{1}{2}$. Hence, $\partial f/\partial p = 0$ can have at most one solution in each interval $(0, 1/2)$ and $(1/2, 1)$. Since the first-order derivative is zero at $p = 1/2$ and since the second-order derivative is positive at $p = 1/2$, we thus have exactly three extremal points of $f(p)$ in $(0, 1)$: a maximum in the interval $(0, 1/2)$, a minimum at $p = 1/2$, and a maximum in the interval $(1/2, 1)$. Combined with the fact that $f(0) = f(1) = 0$ this then proves that $f(p) \geq 0$ for all $p \in [0, 1]$.

We progress to prove the lower bound in (14.173), i.e., we next need to show that

$$\mathrm{I}(\mathsf{W}) + \mathrm{Z}(\mathsf{W}) - 1 \geq 0. \tag{14.196}$$

Again, by (14.182) and (14.186),

$$\begin{aligned}
&\mathrm{Z}(\mathsf{W}) + \mathrm{I}(\mathsf{W}) - 1 \\
&\quad = \mathsf{E}\Big[2\sqrt{p(Y)(1-p(Y))}\Big] + 1 - \mathsf{E}\big[\mathrm{H_b}(p(Y))\big] - 1 && (14.197)\\
&\quad = \mathsf{E}\Big[2\sqrt{p(Y)(1-p(Y))} - \mathrm{H_b}(p(Y))\Big]. && (14.198)
\end{aligned}$$

Thus, (14.196) holds if we can show that

$$2\sqrt{p(1-p)} - \mathrm{H_b}(p) \geq 0, \quad \forall\, p \in [0, 1]. \tag{14.199}$$

To this end, we define

$$f(p) \triangleq 2\sqrt{p(1-p)} + p\log_2 p + (1-p)\log_2(1-p) \tag{14.200}$$

and compute

$$\begin{aligned}
\frac{\partial f}{\partial p} &= \frac{1-2p}{\sqrt{p(1-p)}} + \log_2 p - \log_2(1-p), && (14.201)\\
\frac{\partial^2 f}{\partial p^2} &= \frac{1}{p(1-p)}\left(\frac{1}{\ln 2} - \frac{1}{2\sqrt{p(1-p)}}\right), && (14.202)\\
\frac{\partial^3 f}{\partial p^3} &= \frac{1-2p}{4p^2(1-p^2)}\left(\frac{3}{\sqrt{p(1-p)}} - \frac{4}{\ln 2}\right). && (14.203)
\end{aligned}$$

The remaining discussion is analogous to the paragraph after (14.195) and therefore omitted. □

Corollary 14.22. An immediate consequence of Proposition 14.21 is that

$$\begin{aligned}
\Big(\mathrm{I}(\mathsf{W}) = 0\Big) &\iff \Big(\mathrm{Z}(\mathsf{W}) = 1\Big), && (14.204)\\
\Big(\mathrm{I}(\mathsf{W}) = 1\Big) &\iff \Big(\mathrm{Z}(\mathsf{W}) = 0\Big). && (14.205)
\end{aligned}$$

Proof: Plug $\mathrm{I}(\mathsf{W}) = 0$ or $\mathrm{I}(\mathsf{W}) = 1$ into (14.173). □

Another immediate consequence of Proposition 14.21 and Corollary 14.22 is that polarization also happens with respect to $\mathrm{Z}(\mathsf{W})$.

Corollary 14.23 (Polarization of $\mathrm{Z}(\mathsf{W})$). For any binary-input DMC W and any $\epsilon > 0$,

$$\lim_{\ell\to\infty} \frac{\big|\{s_1 \cdots s_\ell \colon \mathrm{Z}(\mathsf{W}^{s_1\cdots s_\ell}) \in (\epsilon, 1-\epsilon)\}\big|}{2^\ell} = 0. \tag{14.206}$$

Moreover, the fraction of channels that are close to $\mathrm{Z}(\mathsf{W}^{s_1\cdots s_\ell}) = 0$ tends to $\mathrm{I}(\mathsf{W})$ and the fraction of channels that are close to $\mathrm{Z}(\mathsf{W}^{s_1\cdots s_\ell}) = 1$ tends to $1 - \mathrm{I}(\mathsf{W})$ as $\ell \to \infty$.

Proof: From Theorem 14.15 we know that $I_\ell = \mathrm{I}(\mathbb{W}_\ell)$ converges to a binary RV I_∞ taking value in $\{0, 1\}$ with probability $\Pr[I_\infty = 1] = \mathrm{I}(\mathsf{W})$. By (14.174) it now follows that also

$$Z_\ell \triangleq \mathrm{Z}(\mathbb{W}_\ell) \tag{14.207}$$

must converge to a binary RV with $\Pr[Z_\infty = 1] = 1 - \mathrm{I}(\mathsf{W})$. □

However, as discussed already, we need more: We need to understand how fast this polarization happens. The next theorem shows that the speed is actually almost exponentially fast.

Theorem 14.24 (Speed of Polarization [AT09]). Let W be any binary DMC, define $\{S_\ell\}_{\ell\in\mathbb{N}}$ and $\{\mathbb{W}_\ell\}_{\ell\in\mathbb{N}_0}$ as in Section 14.2.5, and let Z_ℓ be defined in (14.207). Then for any fixed $0 < \beta < \frac{1}{2}$,

$$\lim_{\ell\to\infty} \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta}}\Big] = \mathrm{I}(\mathsf{W}). \tag{14.208}$$

Conversely, if $\mathrm{I}(\mathsf{W}) < 1$, then for any $\beta > \frac{1}{2}$,

$$\lim_{\ell\to\infty} \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta}}\Big] = 0. \tag{14.209}$$

Proof: The proof of this result is rather involved. We therefore postpone it to Appendix 14.C. □

14.4 Polar Coding

In this section we now finally address polar coding. We start with a family of coding schemes that contain the polar coding schemes as a special case.

14.4.1 Coset Coding Scheme

The basic idea of polar coding is to repeatedly apply the polar transform to our DMC and then select among the polarized channels only those that are close to perfect. On those we transmit our message bits uncoded, while on the useless channels we transmit frozen bits. The challenge lies in figuring out which channels to use and which to freeze. So, at first we define a coding scheme with an arbitrary choice of frozen bits.

Definition 14.25. For some $n = 2^\ell$, $\ell \in \mathbb{N}$, we define the $(n, \mathsf{K}, \mathcal{F}, \mathbf{u}_\mathcal{F})$ *coset coding scheme* $\mathscr{C}_{\mathsf{T}_n}$ as follows: The length-n codeword $\mathbf{x}$ is given as

$$\mathbf{x} = \mathbf{u}\mathsf{T}_n, \tag{14.210}$$

where T_n describes the ℓ-fold transformation matrix (14.76) and where $\mathbf{u}$ is an n-vector consisting of $\mathsf{K} = n - |\mathcal{F}|$ information bits and of $|\mathcal{F}|$ *frozen bits*. The indices of the latter are given by the *frozen bits set* $\mathcal{F} \subseteq \{1, \ldots, n\}$, and their value is specified by the $|\mathcal{F}|$-vector $\mathbf{u}_\mathcal{F}$. The encoding function thus is fully specified by T_n, $\mathcal{F}$, and $\mathbf{u}_\mathcal{F}$, and the code rate is given by $\mathsf{R} = \frac{\mathsf{K}}{n}$ bits.

The decoding function is based on successive cancellation and is defined by the sequential decisions

$$\hat{U}_k \triangleq \begin{cases} u_k & \text{if } k \in \mathcal{F}, \\ \psi_k(Y_1^n, \hat{U}_1^{k-1}) & \text{if } k \notin \mathcal{F}, \end{cases} \tag{14.211}$$

where

$$\psi_k(y_1^n, \hat{u}_1^{k-1}) = \begin{cases} 0 & \text{if } \frac{W_n^{(k)}(y_1^n, \hat{u}_1^{k-1}|0)}{W_n^{(k)}(y_1^n, \hat{u}_1^{k-1}|1)} \geq 1, \\ 1 & \text{otherwise.} \end{cases} \tag{14.212}$$

Note that even though this looks like a bitwise ML decision it is not exactly so, because the future frozen bits $\mathbf{u}_{\mathcal{F} \cap \{j\colon j>k\}}$ are treated as random variables rather than as known bits.[5]

We remark that we will use $\mathcal{F}^\mathsf{c}$ to denote all indices of the information bits, i.e.,

$$\mathcal{F}^\mathsf{c} \triangleq \{1, \ldots, n\} \setminus \mathcal{F}. \tag{14.213}$$

[5] This suboptimality will allow us to derive efficient recursive formulas for the computation of these decisions. Luckily, the loss in performance will turn out to be small and we will still be able to achieve a rate I(W) with this coding scheme.

Example 14.26. Let $n = 8$ with

$$\mathsf{T}_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{14.214}$$

and suppose that $\mathcal{F} = \{1, 2, 4\}$ and $\mathbf{u}_{\mathcal{F}} = (0\,1\,0)$. Then the length-8 codeword is given as

$$\mathbf{x} = \mathbf{u}\mathsf{T}_8 \tag{14.215}$$

$$= (u_3\ u_5\ u_6\ u_7\ u_8) \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$+ (0\,1\,0) \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{14.216}$$

$$= (u_3\ u_5\ u_6\ u_7\ u_8) \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} + (1\,0\,0\,0\,1\,0\,0\,0). \tag{14.217}$$

Note that here 5 information bits are mapped into a codeword of length 8. Hence the rate of this coding scheme is $\mathsf{R} = \frac{5}{8}$ bits.

Also note that the naming *coset code* can now be understood: in math, a *coset* is defined as the set of elements that is generated by adding a constant value h to each element of a group $\mathcal{G}$. Here, the coset code $\mathscr{C}_{\mathsf{T}_n}$ is basically a K-dimensional subspace of $\{0,1\}^n$, where each element of the subspace is shifted by a constant value (which depends on the values of the frozen bits). ◊

14.4.2 Performance of Coset Coding

We use $P_{\mathrm{e}}^{(n)}(\mathsf{R}, \mathcal{F}, \mathbf{u}_{\mathcal{F}})$ to denote the average probability of (block) error for the $(n, \mathsf{K} = n\mathsf{R}, \mathcal{F}, \mathbf{u}_{\mathcal{F}})$ coset code, where we make the usual assumption that all information bits $\mathbf{u}_{\mathcal{F}^{\mathrm{c}}}$ are IID uniform. By $P_{\mathrm{e}}^{(n)}(\mathsf{R}, \mathcal{F})$ we denote the error probability that is uniformly averaged over all possible choices of $\mathbf{u}_{\mathcal{F}}$.

The key result for the analysis of coset coding schemes is as follows.

Theorem 14.27 (Performance of Coset Coding). For any binary DMC W and any choice of $n = 2^{\ell}$ and $\mathcal{F}$,

$$P_{\mathrm{e}}^{(n)}(\mathsf{R}, \mathcal{F}) \le \sum_{k \in \mathcal{F}^{\mathrm{c}}} \mathsf{Z}(\mathsf{W}_n^{(k)}). \tag{14.218}$$

Hence, there must exist some choice of $\mathbf{u}_{\mathcal{F}}$ such that

$$P_{\mathrm{e}}^{(n)}(\mathsf{R}, \mathcal{F}, \mathbf{u}_{\mathcal{F}}) \le \sum_{k \in \mathcal{F}^{\mathrm{c}}} \mathsf{Z}(\mathsf{W}_n^{(k)}). \tag{14.219}$$

We will show later that if the DMC W is weakly symmetric, then (14.219) holds for any choice of $\mathbf{u}_{\mathcal{F}}$.

Proof: The information bits are IID uniform, and we also choose the frozen bits uniformly at random:

$$\Pr[\mathbf{U}_{\mathcal{F}} = \mathbf{u}_{\mathcal{F}}] = \frac{1}{2^{n-\mathsf{K}}}, \quad \forall\, \mathbf{u}_{\mathcal{F}} \in \{0,1\}^{n-\mathsf{K}},$$

independent of the information bits. We now split the block error event up into the events that the *first* decision error occurs exactly at stage k, and then apply the union bound:

$$\begin{aligned}
&P_{\mathrm{e}}^{(n)}(\mathsf{R}, \mathcal{F}) \\
&= \Pr\left(\bigcup_{k \in \mathcal{F}^{\mathrm{c}}} \left\{U_1^{k-1} = \hat{U}_1^{k-1},\, U_k \neq \hat{U}_k\right\}\right) && (14.220)\\
&\le \sum_{k \in \mathcal{F}^{\mathrm{c}}} \Pr\left[U_1^{k-1} = \hat{U}_1^{k-1},\, U_k \neq \hat{U}_k\right] && (14.221)\\
&= \sum_{k \in \mathcal{F}^{\mathrm{c}}} \Pr\left[U_1^{k-1} = \hat{U}_1^{k-1},\, U_k \neq \psi_k(Y_1^n, \hat{U}_1^{k-1})\right] && (14.222)\\
&= \sum_{k \in \mathcal{F}^{\mathrm{c}}} \Pr\left[U_1^{k-1} = \hat{U}_1^{k-1},\, U_k \neq \psi_k(Y_1^n, U_1^{k-1})\right] && (14.223)\\
&\le \sum_{k \in \mathcal{F}^{\mathrm{c}}} \Pr\left[U_k \neq \psi_k(Y_1^n, U_1^{k-1})\right] && (14.224)\\
&\le \sum_{k \in \mathcal{F}^{\mathrm{c}}} \Pr\left[W_n^{(k)}(Y_1^n, U_1^{k-1} | U_k) \le W_n^{(k)}(Y_1^n, U_1^{k-1} | U_k \oplus 1)\right] && (14.225)\\
&= \sum_{k \in \mathcal{F}^{\mathrm{c}}} \sum_{y_1^n, u_1^k} \Pr\left[Y_1^n = y_1^n,\, U_1^k = u_1^k\right]
\end{aligned}$$

$$\cdot \mathbb{1}\Big\{W_n^{(k)}(y_1^n, u_1^{k-1}|u_k) \le W_n^{(k)}(y_1^n, u_1^{k-1}|u_k \oplus 1)\Big\} \tag{14.226}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \mathsf{E}\Big[\mathbb{1}\Big\{W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k) \le W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k \oplus 1)\Big\}\Big] \tag{14.227}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \mathsf{E}\left[\mathbb{1}\left\{\sqrt{W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k)} \le \sqrt{W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k \oplus 1)}\right\}\right] \tag{14.228}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \mathsf{E}\left[\mathbb{1}\left\{1 \le \sqrt{\frac{W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k \oplus 1)}{W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k)}}\right\}\right] \tag{14.229}$$

$$\le \sum_{k\in\mathcal{F}^{\mathrm{c}}} \mathsf{E}\left[\sqrt{\frac{W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k \oplus 1)}{W_n^{(k)}(Y_1^n, U_1^{k-1}|U_k)}}\right] \tag{14.230}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \sum_{y_1^n, u_1^k} \Pr[U_k = u_k] \Pr\left[Y_1^n = y_1^n,\ U_1^{k-1} = u_1^{k-1} \,\middle|\, U_k = u_k\right] \cdot \sqrt{\frac{W_n^{(k)}(y_1^n, u_1^{k-1}|u_k \oplus 1)}{W_n^{(k)}(y_1^n, u_1^{k-1}|u_k)}} \tag{14.231}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \sum_{y_1^n, u_1^k} \frac{1}{2}\, W_n^{(k)}(y_1^n, u_1^{k-1}|u_k) \sqrt{\frac{W_n^{(k)}(y_1^n, u_1^{k-1}|u_k \oplus 1)}{W_n^{(k)}(y_1^n, u_1^{k-1}|u_k)}} \tag{14.232}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \frac{1}{2} \sum_{u_k} \sum_{y_1^n, u_1^{k-1}} \sqrt{W_n^{(k)}(y_1^n, u_1^{k-1}|u_k)\, W_n^{(k)}(y_1^n, u_1^{k-1}|u_k \oplus 1)} \tag{14.233}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \frac{1}{2} \sum_{u_k} \sum_{y_1^n, u_1^{k-1}} \sqrt{W_n^{(k)}(y_1^n, u_1^{k-1}|0)\, W_n^{(k)}(y_1^n, u_1^{k-1}|1)} \tag{14.234}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \frac{1}{2} \sum_{u_k} \mathsf{Z}(\mathsf{W}_n^{(k)}) \tag{14.235}$$

$$= \sum_{k\in\mathcal{F}^{\mathrm{c}}} \mathsf{Z}(\mathsf{W}_n^{(k)}). \tag{14.236}$$

Here, (14.221) follows from the Union Bound (11.135); the subsequent equality (14.222) follows from the definition of the decoder (14.211) and the fact that $k \in \mathcal{F}^{\mathrm{c}}$; (14.223) holds because before up to stage k the decoder has made no errors $U_1^{k-1} = \hat{U}_1^{k-1}$; in (14.224) we upper-bound some probabilities by 1; (14.225) follows from decoding law (14.212), where we do not have equality because of the case when both conditional probabilities are equal; in (14.226) and (14.227) we write the probability as an expectation over the indicator function:

$$\Pr[X \in \mathcal{B}] = \sum_{x\in\mathcal{B}} \Pr[X = x] = \sum_{x\in\mathcal{X}} \Pr[X = x]\, \mathbb{1}\{x \in \mathcal{B}\} = \mathsf{E}[\mathbb{1}\{X \in \mathcal{B}\}]; \tag{14.237}$$

the inequality (14.230) follows because for any nonnegative RV X,

$$\mathbb{1}\{1 \le X\} = \begin{cases} 0 & \text{if } 0 \le X < 1, \\ 1 & \text{if } 1 \le X \end{cases} \tag{14.238}$$

$$\le X; \tag{14.239}$$

(14.235) follows from the definition of the channel reliability parameter for the binary-input DMC $\mathsf{W}_n^{(k)}$ (Definition 14.17); and the final equality uses that

$$\frac{1}{2} \sum_{u_k=0}^{1} 1 = 1. \tag{14.240}$$

□

Note that this results is useful in two ways: Firstly it gives an upper bound on the probability of error of coset coding schemes and thereby a lower bound on their performance. Secondly, it also gives a clue of how to design good coset coding schemes: we need to freeze those bits u_k that correspond to a large value of $\mathsf{Z}(\mathsf{W}_n^{(k)})$! This now directly leads us to the definition of polar coding schemes.

14.4.3 Polar Coding Schemes

Definition 14.28. Given a binary DMC W, an $(n, \mathsf{K}, \mathcal{F}, \mathbf{u}_{\mathcal{F}})$ coset coding scheme $\mathscr{C}_{\mathsf{T}_n}$ is called *polar coding scheme* $\mathscr{C}_{\text{polar}}^{(n)}(\mathsf{R}, \mathbf{u}_{\mathrm{F}})$ *for* W if $\mathcal{F}$ is chosen such that

$$\mathsf{Z}(\mathsf{W}_n^{(k)}) \ge \mathsf{Z}(\mathsf{W}_n^{(k')}), \quad \forall\, k \in \mathcal{F},\ k' \in \mathcal{F}^{\mathsf{c}}, \tag{14.241}$$

and where $\mathbf{u}_{\mathrm{F}} = \mathbf{u}_{\mathcal{F}}$ and $\mathsf{R} = \mathsf{K}/n$.

The block error probability of a polar coding scheme $\mathscr{C}_{\text{polar}}^{(n)}(\mathsf{R}, \mathbf{u}_{\mathrm{F}})$ is denoted by $P_{\mathsf{e}}^{(n)}(\mathsf{R}, \mathbf{u}_{\mathrm{F}})$. When averaged over all possible choices for the frozen bits vector $\mathbf{u}_{\mathrm{F}}$, the average block error probability is denoted by $P_{\mathsf{e}}^{(n)}(\mathsf{R})$.

We are now finally ready for the main result of this chapter.

Theorem 14.29 (Coding Theorem for Polar Coding).
Let W be any binary DMC with $\mathsf{I}(\mathsf{W}) > 0$. Fix some rate $\mathsf{R} < \mathsf{I}(\mathsf{W})$ and some $0 < \beta < \frac{1}{2}$. Then the polar coding schemes of rate $\mathsf{R} = \frac{\mathsf{K}}{n}$ satisfy

$$\lim_{n\to\infty} \frac{P_{\mathsf{e}}^{(n)}(\mathsf{R})}{2^{-n^\beta}} = 0, \tag{14.242}$$

i.e., the average error probability tends to zero subexponentially fast with an exponent that grows (almost) with $\sqrt{n}$. Using the little-o notation (see

Appendix 14.A) we have

$$P_{\mathrm{e}}^{(n)}(\mathsf{R}) \in \mathcal{O}\Big(2^{-n^{\beta}}\Big). \tag{14.243}$$

Hence, there must exist some (sequence of) frozen bits vectors $\mathbf{u}_{\mathrm{F}}$ such that

$$P_{\mathrm{e}}^{(n)}(\mathsf{R}, \mathbf{u}_{\mathrm{F}}) \in \mathcal{O}\Big(2^{-n^{\beta}}\Big). \tag{14.244}$$

Note that in case the capacity of W is achieved by a uniform input distribution, $\mathrm{I}(\mathsf{W}) = \mathrm{C}(\mathsf{W})$ and the polar coding scheme is capacity achieving! Particularly this is the case for weakly symmetric binary DMCs (see also Section 14.5).

In general the performance of polar codes depends on the frozen bits vector $\mathbf{u}_{\mathrm{F}}$, i.e., we need to find a good choice for it. We will see in Section 14.5 that for symmetric DMCs the dependence on the frozen bits vector $\mathbf{u}_{\mathrm{F}}$ disappears, i.e., the result above holds for any arbitrary choice of $\mathbf{u}_{\mathrm{F}}$.

In [Mos22] we will discuss *error exponents* that describe how fast the probability of error tends to zero as the blocklength grows to infinity. In general it can be proven that for any DMC there exist block coding schemes with an exponential decay of the error probability. Hence, the polar coding schemes fall short of these schemes (the error exponent of polar codes is zero!). Nevertheless a subexponential decay is already pretty fast and, e.g., far outperforms any polynomial decay.

Proof: The proof relies heavily on Theorem 14.24.

Let $0 < \beta < \frac{1}{2}$ and choose some β' such that

$$\beta < \beta' < \frac{1}{2}. \tag{14.245}$$

(Since β is strictly smaller than $\frac{1}{2}$, such a choice of β' is always possible.) Note that the random channel selection introduced in (14.109)–(14.111) results in a uniform distribution among the channels in

$$\mathcal{W} \triangleq \{\mathsf{W}^{-\cdots-}, \ldots, \mathsf{W}^{+\cdots+}\}. \tag{14.246}$$

Therefore the statement from Theorem 14.24 (applied to $0 < \beta' < \frac{1}{2}$) saying that

$$\lim_{\ell\to\infty} \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta'}}\Big] = \mathrm{I}(\mathsf{W}) \tag{14.247}$$

is equivalent to saying that (in the limit) a fraction of $\mathrm{I}(\mathsf{W})$ of all channels in $\mathcal{W}$ have a channel reliability parameter satisfying

$$Z\Big(\mathsf{W}_{2^\ell}^{(k)}\Big) \le 2^{-2^{\ell\beta'}}. \tag{14.248}$$

Since polar coding by definition always picks the channels from $\mathcal{W}$ with the smallest Z-values, Theorem 14.27 now guarantees that as long as $\mathrm{R} < \mathrm{I}(\mathrm{W})$ we have

$$\begin{aligned}
\varlimsup_{\ell\to\infty} \frac{1}{2^{-2^{\ell\beta}}} P_{\mathrm{e}}^{(2^\ell)}(\mathrm{R}) &\le \varlimsup_{\ell\to\infty} \frac{1}{2^{-2^{\ell\beta}}} \sum_{k\in\mathcal{F}_{\mathrm{opt}}^{\mathrm{c}}} \mathrm{Z}\Big(\mathrm{W}_{2^\ell}^{(k)}\Big) && (14.249)\\
&\le \varlimsup_{\ell\to\infty} \frac{1}{2^{-2^{\ell\beta}}} \sum_{k\in\mathcal{F}_{\mathrm{opt}}^{\mathrm{c}}} 2^{-2^{\ell\beta'}} && (14.250)\\
&= \varlimsup_{\ell\to\infty} \frac{1}{2^{-2^{\ell\beta}}} 2^{-2^{\ell\beta'}} \cdot \big|\mathcal{F}_{\mathrm{opt}}^{\mathrm{c}}\big| && (14.251)\\
&= \varlimsup_{\ell\to\infty} 2^{2^{\ell\beta}} \cdot 2^{-2^{\ell\beta'}} \cdot 2^\ell \cdot \mathrm{R} && (14.252)\\
&= \varlimsup_{\ell\to\infty} 2^{-2^{\ell\beta'}+2^{\ell\beta}+\ell} \cdot \mathrm{R} && (14.253)\\
&= 0. && (14.254)
\end{aligned}$$

Here, the first inequality follows from Theorem 14.27; the subsequent inequality from (14.248); the equality (14.252) holds because $|\mathcal{F}_{\mathrm{opt}}^{\mathrm{c}}| = \mathrm{K} = n\mathrm{R}$; and the final equality holds because (14.245) guarantees that in the exponent the term with a negative sign dominates all other terms. □

14.5 Polar Coding for Symmetric DMCs

In the following we give an alternative definition for weak symmetry of DMCs that is slightly more formal than Definition 12.11 from Chapter 12. We directly focus on binary DMCs.

Definition 14.30. A binary DMC W is called *weakly symmetric* if there exists a permutation $\pi_1(\cdot)\colon \mathcal{Y} \to \mathcal{Y}$ such that

$$\pi_1^{-1}(y) = \pi_1(y), \quad y \in \mathcal{Y}, \tag{14.255}$$

and

$$W(y|1) = W\big(\pi_1(y)\big|0\big), \quad y \in \mathcal{Y}. \tag{14.256}$$

Using $\pi_0(\cdot)$ to denote the identity permutation (i.e., the permutation that leaves everything the way it is), obviously, any binary DMC satisfies

$$W(y|x\oplus 0) = W\big(\pi_0(y)\big|x\big), \quad x \in \{0,1\},\ y \in \mathcal{Y}. \tag{14.257}$$

Thus, combining (14.256) and (14.257), we see that a weakly symmetric binary DMC satisfies for any $a \in \{0,1\}$,

$$W(y|x\oplus a) = W\big(\pi_a(y)\big|x\big), \quad x \in \{0,1\},\ y \in \mathcal{Y},\ a \in \{0,1\}. \tag{14.258}$$

To understand why Definition 14.30 and Definition 12.11 are equivalent, let $\mathcal{Y}_1, \ldots, \mathcal{Y}_S$ be the largest partition[6] of $\mathcal{Y}$ satisfying

$$\bigcup_{y \in \mathcal{Y}_s} \{\pi_1(y)\} = \mathcal{Y}_s, \quad s = 1, \ldots, S. \tag{14.259}$$

(The subsets $\mathcal{Y}_s$ correspond to the subchannels of Section 12.3.) Now note that for every $y \in \mathcal{Y}_s$ there exists a $\tilde{y} \in \mathcal{Y}_s$ with

$$W(y|0) = W(\tilde{y}|1), \tag{14.260}$$

i.e., each subchannels $\mathcal{Y}_s$ is uniformly focusing. Indeed, for $y \in \mathcal{Y}_s$,

$$W(y|0) = W(\pi_1(y)|1) \tag{14.261}$$

and $\pi_1(y) \in \mathcal{Y}_s$ by (14.259).

Moreover,

$$\bigcup_{y \in \mathcal{Y}_s} \{W(y|0)\} = \bigcup_{y \in \mathcal{Y}_s} \Big\{W\big(\pi_1(y)|1\big)\Big\} = \bigcup_{y \in \mathcal{Y}_s} \{W(y|1)\}, \tag{14.262}$$

where the first equality follows from (14.258) and the second from (14.259). Hence, each subchannel is also uniformly dispersive.

In the following we will assume that W is weakly symmetric, i.e., there exists a corresponding permutation $\pi_1(\cdot)$.

Remark 14.31. Note that in the remainder of this chapter, we will be lazy and use *symmetric*, when we actually mean *weakly symmetric*. △

Lemma 14.32. If W is symmetric, then also W^+ and W^- are symmetric.

Proof: Using that W is symmetric and using Corollary 14.2, we have for $a \in \{0,1\}$

$$\begin{aligned}
&W^-(y_1, y_2|u_1 \oplus a) \\
&\quad = \frac{1}{2} W(y_1|u_1 \oplus a \oplus 0) W(y_2|0) + \frac{1}{2} W(y_1|u_1 \oplus a \oplus 1) W(y_2|1) && (14.263)\\
&\quad = \frac{1}{2} W\big(\pi_a(y_1)\big|u_1 \oplus 0\big) W\big(\pi_0(y_2)\big|0\big) \\
&\qquad + \frac{1}{2} W\big(\pi_a(y_1)\big|u_1 \oplus 1\big) W\big(\pi_0(y_2)\big|1\big) && (14.264)\\
&\quad = W^-\big(\pi_a(y_1), \pi_0(y_2)\big|u_1\big) && (14.265)\\
&\quad = W^-\big(\boldsymbol{\pi}_{(a,0)}(y_1, y_2)\big|u_1\big), && (14.266)
\end{aligned}$$

where we have defined

$$\boldsymbol{\pi}_{(a_1,\ldots,a_n)}(\mathbf{y}) \triangleq \big(\pi_{a_1}(y_1), \ldots, \pi_{a_n}(y_n)\big)^{\mathsf{T}}. \tag{14.267}$$

[6] A *partition* of $\mathcal{Y}$ is a split of $\mathcal{Y}$ into subsets that are nonzero, disjoint, and add to the whole: $\bigcup_s \mathcal{Y}_s = \mathcal{Y}$. A partition is *largest* if it contains the largest number of subsets.

Hence, we have found a corresponding permutation for W^- showing that also W^- is symmetric.

Similarly,

$$\begin{aligned} W^+(y_1,y_2,u_1|u_2\oplus a) &= \frac{1}{2}\,W(y_2|u_1\oplus u_2\oplus a)\,W(y_2|u_2\oplus a) && (14.268)\\ &= \frac{1}{2}\,W\bigl(\pi_a(y_1)\bigr|u_1\oplus u_2\bigr)\,W\bigl(\pi_a(y_2)\bigr|u_2\bigr) && (14.269)\\ &= W^+\bigl(\pi_a(y_1),\pi_a(y_2),u_1\bigr|u_2\bigr) && (14.270)\\ &= W^+\bigl(\boldsymbol{\pi}_{(a,a,0)}(y_1,y_2,u_1)\bigr|u_2\bigr). && (14.271) \end{aligned}$$

□

Corollary 14.33. If W is symmetric, then all channels generated from it by the polar transform are symmetric. In particular, the channels $\mathsf{W}_n^{(k)}$ for $k=1,\ldots,n$ are symmetric, and also the induced vector channel with input $\mathbf{u}$ and output $\mathbf{y}$ is symmetric:

$$W_{\mathrm{tot},n}(\mathbf{y}|\mathbf{u}\oplus\mathbf{a}) = W_{\mathrm{tot},n}\bigl(\boldsymbol{\pi}_{\mathbf{a}\mathsf{T}_n}(\mathbf{y})\bigr|\mathbf{u}\bigr). \tag{14.272}$$

Proof: The fact that $\mathsf{W}_n^{(k)}$ is symmetric follows from a recursive application of Lemma 14.32.

Using the product notation

$$W^n(\mathbf{y}|\mathbf{x}) = \prod_{k=1}^n W(y_k|x_k), \tag{14.273}$$

we have

$$\begin{aligned} W_{\mathrm{tot},n}(\mathbf{y}|\mathbf{u}\oplus\mathbf{a}) &= W^n\bigl(\mathbf{y}\bigr|(\mathbf{u}\oplus\mathbf{a})\mathsf{T}_n\bigr) && (14.274)\\ &= W^n\bigl(\mathbf{y}\bigr|\mathbf{u}\mathsf{T}_n\oplus\mathbf{a}\mathsf{T}_n\bigr) && (14.275)\\ &= W^n\bigl(\boldsymbol{\pi}_{\mathbf{a}\mathsf{T}_n}(\mathbf{y})\bigr|\mathbf{u}\mathsf{T}_n\bigr) && (14.276)\\ &= W_{\mathrm{tot},n}\bigl(\boldsymbol{\pi}_{\mathbf{a}\mathsf{T}_n}(\mathbf{y})\bigr|\mathbf{u}\bigr). && (14.277) \end{aligned}$$

□

Recall that the DMC $\mathsf{W}_n^{(k)}$ has input u_k and output (Y_1^n,U_1^{k-1}). Its channel reliability parameter is therefore defined as

$$\mathsf{Z}\bigl(\mathsf{W}_n^{(k)}\bigr) = \sum_{y_1^n,u_1^{k-1}} \sqrt{W_n^{(k)}\bigl(y_1^n,u_1^{k-1}\bigr|0\bigr)\,W_n^{(k)}\bigl(y_1^n,u_1^{k-1}\bigr|1\bigr)}. \tag{14.278}$$

We define now a *conditional channel reliability parameter* as follows

$$\mathsf{Z}\Bigl(\mathsf{W}_n^{(k)}\Bigr|u_1^{k-1}\Bigr) \triangleq \sum_{y_1^n} \sqrt{W_n^{(k)}\bigl(y_1^n,u_1^{k-1}|0\bigr)\,W_n^{(k)}\bigl(y_1^n,u_1^{k-1}|1\bigr)}. \tag{14.279}$$

Proposition 14.34. For a symmetric binary DMC W and for any $k \in \{1, \ldots, n\}$, the conditional channel reliability parameter of $\mathsf{W}_n^{(k)}$ does not depend on the choice of u_1^{k-1}:

$$\mathsf{Z}\Big(\mathsf{W}_n^{(k)}\Big|u_1^{k-1}\Big) = \mathsf{Z}\Big(\mathsf{W}_n^{(k)}\Big|0_1^{k-1}\Big). \tag{14.280}$$

Proof: Fix some u_1^n. Then

$$\begin{aligned}
&\mathsf{Z}\Big(\mathsf{W}_n^{(k)}\Big|u_1^{k-1}\Big)\\
&\quad= \sum_{y_1^n} \sqrt{W_n^{(k)}(y_1^n, u_1^{k-1}|0)\, W_n^{(k)}(y_1^n, u_1^{k-1}|1)} && (14.281)\\
&\quad= \sum_{y_1^n} \sqrt{W_n^{(k)}(\boldsymbol{\pi}_{\mathbf{a}}(y_1^n), u_1^{k-1} \oplus a_1^{k-1}|0 \oplus a_k)}\\
&\qquad\qquad \cdot \sqrt{W_n^{(k)}(\boldsymbol{\pi}_{\mathbf{a}}(y_1^n), u_1^{k-1} \oplus a_1^{k-1}|1 \oplus a_k)} && (14.282)\\
&\quad= \sum_{y_1^n} \sqrt{W_n^{(k)}(\boldsymbol{\pi}_{\mathbf{u}}(y_1^n), 0_1^{k-1}|0 \oplus u_k)\, W_n^{(k)}(\boldsymbol{\pi}_{\mathbf{u}}(y_1^n), 0_1^{k-1}|1 \oplus u_k)} && (14.283)\\
&\quad= \sum_{y_1^n} \sqrt{W_n^{(k)}(\boldsymbol{\pi}_{\mathbf{u}}(y_1^n), 0_1^{k-1}|0)\, W_n^{(k)}(\boldsymbol{\pi}_{\mathbf{u}}(y_1^n), 0_1^{k-1}|1)} && (14.284)\\
&\quad= \sum_{\tilde{y}_1^n} \sqrt{W_n^{(k)}(\tilde{y}_1^n, 0_1^{k-1}|0)\, W_n^{(k)}(\tilde{y}_1^n, 0_1^{k-1}|1)} && (14.285)\\
&\quad= \mathsf{Z}\Big(\mathsf{W}_n^{(k)}\Big|0_1^{k-1}\Big). && (14.286)
\end{aligned}$$

Here (14.282) follows from Corollary 14.33; in (14.283) we choose $\mathbf{a} \triangleq \mathbf{u}$; and (14.285) holds because for a fixed $\mathbf{u}$, summing over all $\mathbf{y}$ is equivalent to summing over all $\boldsymbol{\pi}_{\mathbf{u}}(\mathbf{y})$. □

We are now ready for a restatement of Theorem 14.29.

Theorem 14.35 (Coding Theorem for Polar Coding on Symmetric DMCs). Let W be a symmetric binary DMC with positive capacity $\mathsf{C}(\mathsf{W}) > 0$. Fix some rate $\mathsf{R} < \mathsf{C}(\mathsf{W})$, some $0 < \beta < \frac{1}{2}$, and some frozen bits vector $\mathbf{u}_{\mathrm{F}}$. Then the polar coding scheme of rate R satisfies

$$\lim_{n\to\infty} \frac{P_{\mathrm{e}}^{(n)}(\mathsf{R}, \mathbf{u}_{\mathrm{F}})}{2^{-n^\beta}} = 0, \tag{14.287}$$

i.e., polar coding is capacity achieving irrespective of the choice of the frozen bits vector.

Proof: This follows directly from an adaptation of the proof of Theorem 14.29 using Proposition 14.34. □

14.6 Complexity Analysis

A capacity-achieving coding scheme is useless if the complexity of implementing and operating it is very high. Note that we know from Chapter 11 that a randomly generated codebook very likely is performing well. However, the complexity of encoding and decoding for such a random scheme is exuberant and therefore random codes are only of theoretical interest.

In this section we will now show that polar coding schemes can be operated at quite low costs.

14.6.1 Encoder

In principle the encoder is straightforward: we need to perform the matrix multiplication of the n-vector $\mathbf{u}$ (combined K information bits and $n-\mathsf{K}$ frozen bits) with T_n. In general this requires $\mathcal{O}(n^2)$ operations.[7] Luckily, we can improve on this by taking advantage of the recursive structure of our system. Recalling the discussion in Section 14.2.1 and, e.g., Figure 14.7, we note that the encoder consists of several blocks (see Figure 14.12): The first block P_n performs the bit reversal, and the recursive block G_n is responsible for the main part of the polar transform, calling itself twice with a smaller number of inputs.

In the following we assume that the complexity of the bit reversal is n operations, while each EXOR counts as 1 operation. Hence, the complexity of this encoder is

$$\chi_{\text{encoder}}(n) = n + \chi_{\mathsf{G}}(n) \tag{14.288}$$

where

$$\chi_{\mathsf{G}}(n) = \frac{n}{2} + 2\chi_{\mathsf{G}}\left(\frac{n}{2}\right), \tag{14.289a}$$

$$\chi_{\mathsf{G}}(2) = 1. \tag{14.289b}$$

Since $n = 2^\ell$, we can also reformulate this as

$$\bar{\chi}_{\mathsf{G}}(\ell) = 2^{\ell-1} + 2\bar{\chi}_{\mathsf{G}}(\ell-1), \tag{14.290a}$$

$$\bar{\chi}_{\mathsf{G}}(1) = 1. \tag{14.290b}$$

It thus follows that

$$\bar{\chi}_{\mathsf{G}}(\ell) = 2^{\ell-1} + 2\Big(2^{\ell-2} + 2(2^{\ell-3} + 2(\cdots \tag{14.291}$$

or

$$\bar{\chi}_{\mathsf{G}}(\ell) = \sum_{j=0}^{\ell-1} 2^{\ell-1-j}\cdot 2^j = \sum_{j=0}^{\ell-1} 2^{\ell-1} = \ell\, 2^{\ell-1} = \frac{1}{2}\, 2^\ell\, \ell. \tag{14.292}$$

[7] See Appendix 14.A for the definition of the big-O notation.

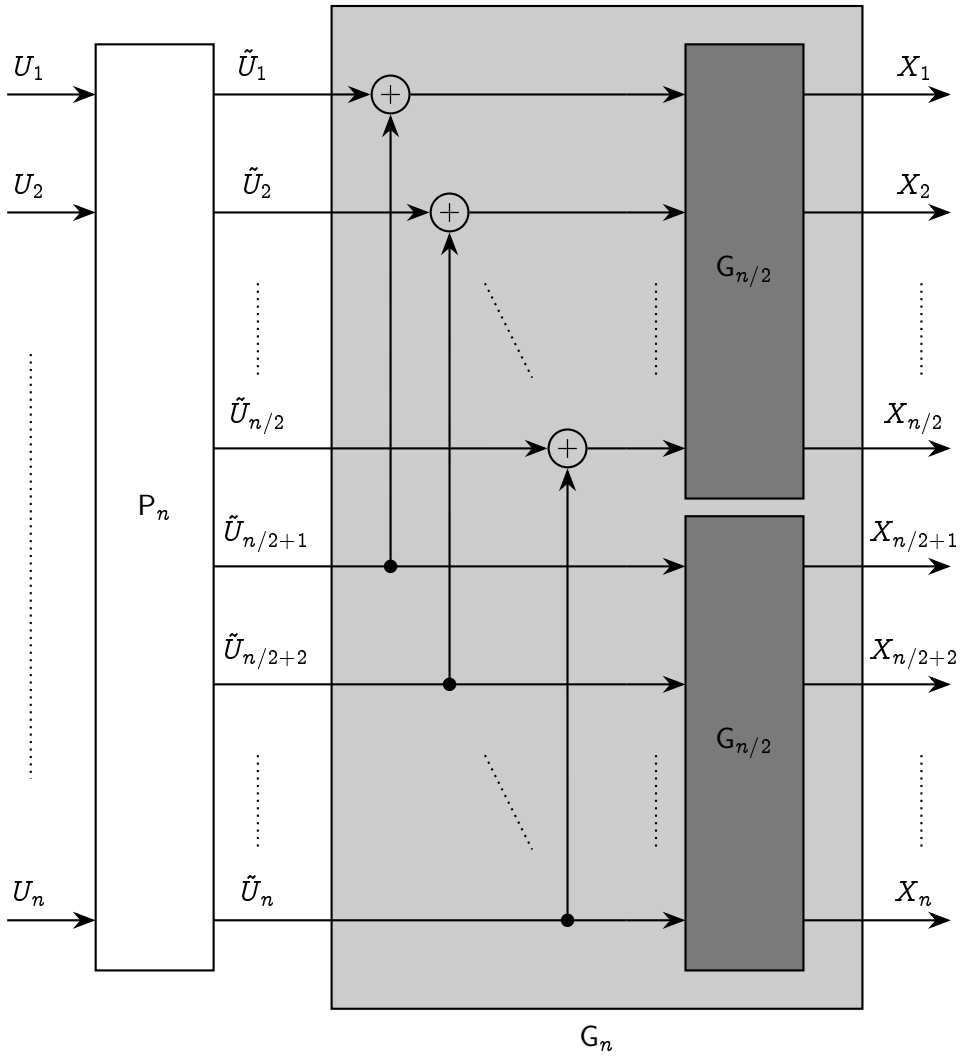


Figure 14.12: Recursive construction of polar encoder.

Rewriting this again with $n = 2^\ell$ and plugging it into (14.288), we obtain

$$\chi_{\text{encoder}}(n) = n + \frac{1}{2}\, n \log_2 n, \tag{14.293}$$

i.e.,

$$\chi_{\text{encoder}}(n) \in \mathcal{O}(n \log_2 n). \tag{14.294}$$

Note that the structure of the encoder allows parallel computing, i.e., one can find algorithms using n parallel processors that each then only needs to perform $\mathcal{O}(\log_2 n)$ operations.

14.6.2 Decoder

For an arbitrary coset coding scheme, the decoder needs to evaluate for each $k \in \mathcal{F}^{\mathrm{c}}$ the likelihood ratio

$$\mathrm{L}_n^{(k)}(y_1^n, \hat{u}_1^{k-1}) \triangleq \frac{W_n^{(k)}(y_1^n, \hat{u}_1^{k-1}|0)}{W_n^{(k)}(y_1^n, \hat{u}_1^{k-1}|1)}. \tag{14.295}$$

The question therefore arises how one can compute this efficiently. We again aim for a recursive algorithm, and we would like to make use of the formulas from Corollary 14.10. To this end, we firstly drop the constraint that $k \in \mathcal{F}^{\mathrm{c}}$, but decide to compute $\mathrm{L}_n^{(k)}$ for all $k = 1, \ldots, n$. Of course, in the end, the decoder will not use the likelihood ratios belonging to frozen bits, however, this overhead is more than compensated because many computations can be reused several times.

So, in order to be able to use the expressions of Corollary 14.10 we group the likelihood ratios into two classes: one for even k and one for odd k:

$$\begin{aligned}&\Big\{\mathrm{L}_n^{(k)}(y_1^n, \hat{u}_1^{k-1})\colon k = 1, \ldots, n\Big\}\\ &\quad = \Big\{\mathrm{L}_n^{(2j-1)}(y_1^n, \hat{u}_1^{2j-2})\colon j = 1, \ldots, \frac{n}{2}\Big\} \cup \Big\{\mathrm{L}_n^{(2j)}(y_1^n, \hat{u}_1^{2j-1})\colon j = 1, \ldots, \frac{n}{2}\Big\}.\end{aligned} \tag{14.296}$$

Now, using (14.63) we compute

$$\begin{aligned}&\mathrm{L}_n^{(2j-1)}(y_1^n, \hat{u}_1^{2j-2})\\ &= \mathrm{L}_n^{(2j-1)}(\mathbf{y}, \hat{\mathbf{u}}) && (14.297)\\ &= \frac{W_n^{(2j-1)}(\mathbf{y}, \hat{\mathbf{u}}|0)}{W_n^{(2j-1)}(\mathbf{y}, \hat{\mathbf{u}}|1)} && (14.298)\\ &= \frac{W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}|0)\, W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}}|0) + W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}|1)\, W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}}|1)}{W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}|1)\, W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}}|0) + W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}|0)\, W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}}|1)} && (14.299)\\ &= \frac{\mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}) + \frac{1}{\mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}})}}{1 + \frac{\mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}})}{\mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}})}} && (14.300)\\ &= \frac{1 + \mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}})\, \mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}})}{\mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}) + \mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}})} && (14.301)\\ &= \frac{1 + \mathrm{L}_{n/2}^{(j)}\big(y_1^{n/2}, \hat{u}_{1,\mathrm{odd}}^{2j-2} \oplus \hat{u}_{1,\mathrm{even}}^{2j-2}\big)\, \mathrm{L}_{n/2}^{(j)}\big(y_{n/2+1}^{n}, \hat{u}_{1,\mathrm{even}}^{2j-2}\big)}{\mathrm{L}_{n/2}^{(j)}\big(y_1^{n/2}, \hat{u}_{1,\mathrm{odd}}^{2j-2} \oplus \hat{u}_{1,\mathrm{even}}^{2j-2}\big) + \mathrm{L}_{n/2}^{(j)}\big(y_{n/2+1}^{n}, \hat{u}_{1,\mathrm{even}}^{2j-2}\big)}. && (14.302)\end{aligned}$$

Similarly, using (14.64),

$$\begin{aligned}
&\mathrm{L}_n^{(2j)}(y_1^n, \hat{u}_1^{2j-1}) \\
&\quad = \mathrm{L}_n^{(2j)}(\mathbf{y}, \hat{\mathbf{u}}, \hat{u}_{2j-1}) && (14.303)\\
&\quad = \frac{W_n^{(2j)}(\mathbf{y}, \hat{\mathbf{u}}, \hat{u}_{2j-1}|0)}{W_n^{(2j)}(\mathbf{y}, \hat{\mathbf{u}}, \hat{u}_{2j-1}|1)} && (14.304)\\
&\quad = \frac{W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}|\hat{u}_{2j-1})\, W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}}|0)}{W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}}|\hat{u}_{2j-1} \oplus 1)\, W_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}}|1)} && (14.305)\\
&\quad = \Big(\mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{l}}, \hat{\mathbf{u}}_{\mathrm{o}} \oplus \hat{\mathbf{u}}_{\mathrm{e}})\Big)^{1-2\hat{u}_{2j-1}}\, \mathrm{L}_{n/2}^{(j)}(\mathbf{y}_{\mathrm{u}}, \hat{\mathbf{u}}_{\mathrm{e}}) && (14.306)\\
&\quad = \Big(\mathrm{L}_{n/2}^{(j)}\Big(y_1^{n/2}, \hat{u}_{1,\mathrm{odd}}^{2j-2} \oplus \hat{u}_{1,\mathrm{even}}^{2j-2}\Big)\Big)^{1-2\hat{u}_{2j-1}}\, \mathrm{L}_{n/2}^{(j)}\Big(y_{n/2+1}^{n}, \hat{u}_{1,\mathrm{even}}^{2j-2}\Big). && (14.307)
\end{aligned}$$

Interestingly, the required likelihood ratios for the evaluation of (14.302) and (14.307) are the same! Thus, for any $j \in \{1, \ldots, n/2\}$, to compute

$$\Big\{\mathrm{L}_n^{(2j-1)}(y_1^n, \hat{u}_1^{2j-2}), \mathrm{L}_n^{(2j)}(y_1^n, \hat{u}_1^{2j-1})\Big\} \tag{14.308}$$

it is sufficient to know

$$\Big\{\mathrm{L}_{n/2}^{(j)}\Big(y_1^{n/2}, \hat{u}_{1,\mathrm{odd}}^{2j-2} \oplus \hat{u}_{1,\mathrm{even}}^{2j-2}\Big), \mathrm{L}_{n/2}^{(j)}\Big(y_{n/2+1}^{n}, \hat{u}_{1,\mathrm{even}}^{2j-2}\Big)\Big\}. \tag{14.309}$$

Hence, in order to determine the n likelihood ratios (14.295), we need to first find the $\frac{n}{2} \cdot 2 = n$ likelihood ratios (14.309). The latter can again be done recursively!

We first consider the likelihood ratios on the left in (14.309):

$$\Big\{\mathrm{L}_{n/2}^{(k)}\Big(y_1^{n/2}, \hat{u}_{1,\mathrm{odd}}^{2k-2} \oplus \hat{u}_{1,\mathrm{even}}^{2k-2}\Big) \colon k = 1, \ldots, \frac{n}{2}\Big\}. \tag{14.310}$$

We introduce the shorthand

$$\hat{v}_1^{n/2} \triangleq \hat{u}_{1,\mathrm{odd}}^{n} \oplus \hat{u}_{1,\mathrm{even}}^{n} \tag{14.311}$$

and split (14.310) up into even and odd:

$$\begin{aligned}
&\Big\{\mathrm{L}_{n/2}^{(k)}(y_1^{n/2}, \hat{v}_1^{k-1}) \colon k = 1, \ldots, \frac{n}{2}\Big\} \\
&= \Big\{\mathrm{L}_{n/2}^{(2j-1)}(y_1^{n/2}, \hat{v}_1^{2j-2}) \colon j = 1, \ldots, \frac{n}{4}\Big\} \cup \Big\{\mathrm{L}_{n/2}^{(2j)}(y_1^{n/2}, \hat{v}_1^{2j-1}) \colon j = 1, \ldots, \frac{n}{4}\Big\}.
\end{aligned} \tag{14.312}$$

Again, by (14.302) and (14.307), these classes can be computed once we know the values of

$$\begin{aligned}
&\Big\{\mathrm{L}_{n/4}^{(j)}\Big(y_1^{n/4}, \hat{v}_{1,\mathrm{odd}}^{2j-2} \oplus \hat{v}_{1,\mathrm{even}}^{2j-2}\Big) \colon j = 1, \ldots, \frac{n}{4}\Big\} \\
&\qquad \cup \Big\{\mathrm{L}_{n/4}^{(j)}\Big(y_{n/4+1}^{n/2}, \hat{v}_{1,\mathrm{even}}^{2j-2}\Big) \colon j = 1, \ldots, \frac{n}{4}\Big\}.
\end{aligned} \tag{14.313}$$

To evaluate the likelihood ratios on the right in (14.309):

$$\left\{\mathrm{L}_{n/2}^{(k)}(y_{n/2+1}^n, \hat{w}_1^{k-1})\colon k=1,\dots,\frac{n}{2}\right\}, \tag{14.314}$$

where we use the shorthand

$$\hat{w}_1^{n/2} \triangleq \hat{u}_{1,\text{even}}^n, \tag{14.315}$$

we need to determine

$$\begin{aligned}&\left\{\mathrm{L}_{n/4}^{(j)}\left(y_{n/2+1}^{3n/4}, \hat{w}_{1,\text{odd}}^{2j-2} \oplus \hat{w}_{1,\text{even}}^{2j-2}\right)\colon j=1,\dots,\frac{n}{4}\right\}\\ &\quad\cup\left\{\mathrm{L}_{n/4}^{(j)}\left(y_{3n/4+1}^n, \hat{w}_{1,\text{even}}^{2j-2}\right)\colon j=1,\dots,\frac{n}{4}\right\}.\end{aligned} \tag{14.316}$$

Hence, we need to compute $\frac{n}{4}\cdot 4 = n$ likelihood ratios.

We continue in this fashion until we reach

$$\mathrm{L}_1(y) = \frac{W(y|0)}{W(y|1)}. \tag{14.317}$$

In this way, in total we evaluate

$$(\ell+1)\cdot n = (\log_2 n + 1)\cdot n \tag{14.318}$$

times the expressions (14.302) or (14.307) for various different likelihood ratios.

Assuming that the evaluation of (14.302) or (14.307) has a complexity of at most some constant α, we see that the complexity of the decoder is

$$\chi_{\text{decoder}}(n) \le \alpha\cdot(\log_2 n + 1)\cdot n \tag{14.319}$$

i.e.,

$$\chi_{\text{decoder}}(n) \in \mathcal{O}(n\log_2 n). \tag{14.320}$$

Remark 14.36. Note that if we compute the likelihood ratios (14.295) for all $k\in\mathcal{F}^{\text{c}}$ separately using (14.302) and (14.307), but without sharing the intermediate results, then

$$\chi_k(n) \le 2\chi_k\left(\frac{n}{2}\right) + \alpha. \tag{14.321}$$

Taking $\chi_k(1)=1$, this yields

$$\chi_k(n) \le 2^\ell + (2^\ell-1)\alpha \le (1+\alpha)2^\ell = (1+\alpha)n, \tag{14.322}$$

and thus

$$\chi_{\text{decoder}}(n) \le \mathrm{K}(1+\alpha)n = \mathrm{R}(1+\alpha)n^2 \in \mathcal{O}(n^2). \tag{14.323}$$

△

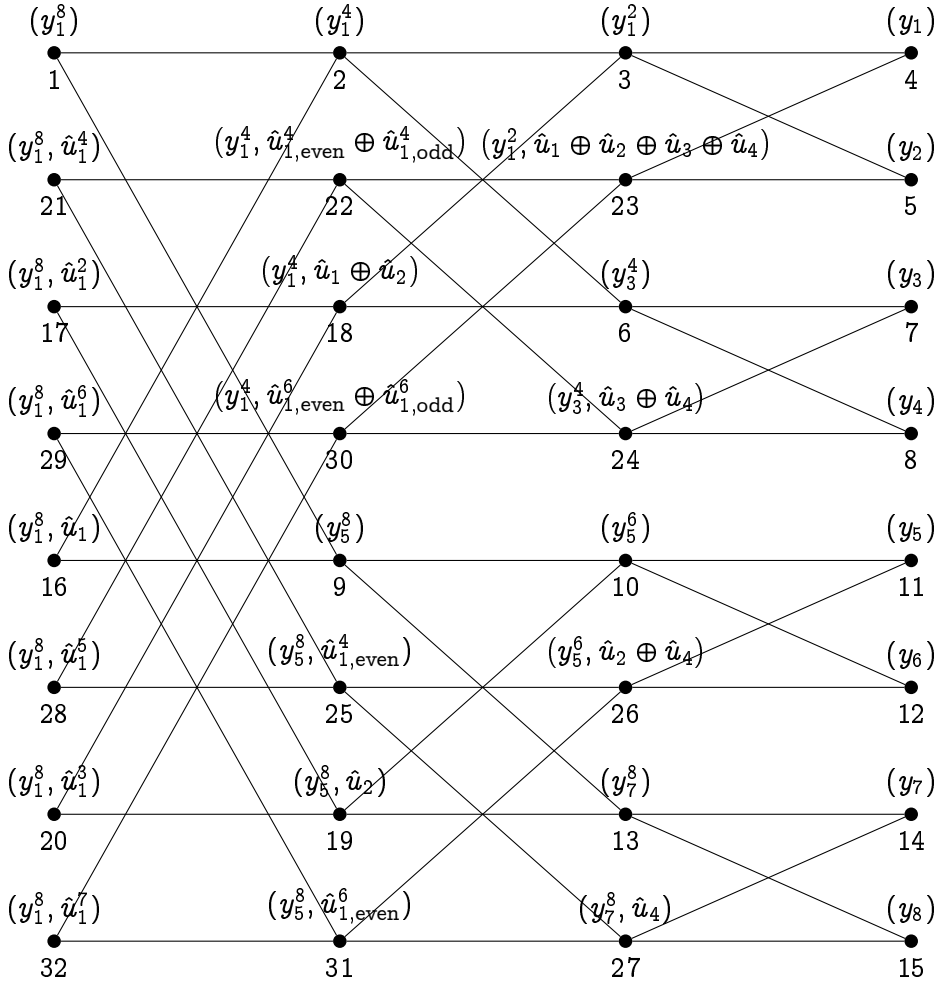


Figure 14.13: Recursive computation of likelihood ratios for successive cancellation decoding for a polar coding scheme with blocklength $n = 8$ (yielding $8(\log_2 8 + 1) = 32$ terms). The variables above the nodes show the required inputs of the corresponding likelihood ratio and the number below represents the order of computation when following a depth-first approach. At each node the corresponding likelihood ratio can only be computed once the two connected nodes on its right feed back their corresponding likelihood ratio.

Note that we have not yet discussed the order in which the $(\log_2 n + 1)n$ likelihood ratios should be evaluated. Recall that we are doing successive cancellation decoding, i.e., for the decoding of U_k, $\hat{u}_1^{k-1}$ need to be decided already. An intuitive procedure is to follow a *depth-first* algorithm.

Figure 14.13 shows an example for a blocklength $n = 8$ polar coding scheme: We start at node 1, which can only compute its likelihood ratio (LR) if it obtains the LRs from its right neighbors (node 2 and 9). This is not the case yet, thus we proceed to node 2 (which needs to wait for 3 and 6); to node 3 (waiting for 4 and 5). Node 4 then actually can compute its LR (based on the output y_1 only), and similarly node 5 (based on y_2). This now allows node 3 to compute its LR and feed it back to node 2. After processing node 6 (based on node 7 and 8), node 2 then feeds back its LR to node 1. Now we follow the second branch starting from node 1 to pass through nodes 9 to 15. Finally, node 1 can compute its LR and we can decode for $\hat{u}_1$, which is now available for the remainder of the decoding.

We proceed to node 16, that needs as its input y_1^8 and also the freshly decoded $\hat{u}_1$. Note that node 16 has already access to the LRs of node 2 and 9 (as these nodes have already finished computing their LR!), thus we can immediately decode for $\hat{u}_2$.

We proceed to node 17, etc.

14.6.3 Code Creation

Obviously, code creation complexity is less crucial than encoding and decoding complexity because a code needs to be created once only and usually in an offline fashion, while the encoding and decoding are repeated tasks and have to be executed online.

Nevertheless, the code creation should be achievable in reasonable time and effort as we cannot afford to spend months of computation time on a high-speed computer system just for the design of one particular polar code.

The polar code creation basically consists of finding the frozen bits set $\mathcal{F}$ that satisfies (14.241) and the value of the frozen bits vector $\mathbf{u}_{\mathrm{F}}$.

To choose $\mathcal{F}$, we need to evaluate the channel reliability parameter $\mathsf{Z}(\mathsf{W}_n^{(k)})$ for all $k = 1, \ldots, n$ and then sort them. Unfortunately, the computation of $\mathsf{Z}(\mathsf{W})$ grows with the alphabet size of the channel output, and the channel output alphabets grow with each application of the polar transform. So, in general no efficient algorithm is known to fulfill this task.

There is the exception of the BEC. We already have seen in Example 14.3 that applying the polar transform to a BEC will result in two new BECs with different erasure probabilities:

$$\mathrm{BEC}(\delta)^+ = \mathrm{BEC}(\delta^2), \tag{14.324}$$

$$\mathrm{BEC}(\delta)^- = \mathrm{BEC}(2\delta - \delta^2). \tag{14.325}$$

We also know from Theorem 14.20 that for a BEC,

$$\mathsf{Z}(\mathrm{BEC}^+) = \mathsf{Z}^2(\mathrm{BEC}), \tag{14.326}$$

$$\mathsf{Z}(\mathrm{BEC}^-) = 2\mathsf{Z}(\mathrm{BEC}) - \mathsf{Z}^2(\mathrm{BEC}). \tag{14.327}$$

Combined with the fact that

$$\mathsf{Z}(\mathrm{BEC}(\delta)) = \delta \tag{14.328}$$

this allows us to compute all channel reliability parameters recursively in $2(n-1)$ steps, i.e.,

$$\chi_{\mathcal{F},\mathrm{BEC}}(n) \in \mathcal{O}(n). \tag{14.329}$$

For a general binary channel, this is not possible. There is a quickly growing list of publications addressing this topic. Currently, the most efficient systems use tricks to *approximate* $\mathsf{Z}(\mathsf{W}_n^{(k)})$ in clever ways. In particular, one would like to have at the same time upper and lower bounds to the exact values that can be computed with much less complexity. This way one might, strictly speaking, miss out on the polar coding scheme, but one will find a coset coding scheme with (almost) identical performance in much shorter time. We omit the details and refer the interested reader to the literature [TV13].

The search for the optimal values of $\mathbf{u}_{\mathrm{F}}$ brings additional difficulty. Again, it is practically impossible to check through all possible choices. Luckily, it turns out that the performance of a polar coding scheme normally is not very sensitive to the exact choice of $\mathbf{u}_{\mathrm{F}}$.

There are further reasons why the frozen bits vector receives less attention: Firstly, we know that in the case of symmetric channels the value of the frozen bits vector is completely irrelevant. Secondly, recall that in the derivation of the average error probability of a coset coding scheme when we average over all frozen bits vectors, we have made the assumption that the frozen bits vectors are picked uniformly. Hence, a possible option to avoid the search for a good frozen bits vector is to use pseudo-random vectors that are deterministic and therefore known at both transmitter and receiver, but that mimic a uniform distribution.

14.7 Discussion

The discovery of a first deterministic construction of a capacity-achieving coding scheme is one of the chief breakthroughs in information theory of the first decade of the 21$^{\mathrm{st}}$ century. Polar coding still suffers from some drawbacks that at the moment still hampers its widespread use, but there is a lot of research effort put into these issues.

One of the main point of criticism is the slow error decay rate (zero error exponent [Mos22, Chapter 17], compare with Theorem 14.29) in comparison to

known LDPC or turbo codes. So, given that one chooses a coding rate $R < C$ small enough such that both LDPC and polar coding schemes exist (LDPC codes are not capacity achieving!), the polar coding scheme will require longer blocklengths than LDPC codes to attain the same average error rate.

An important point to make here, however, is that the performance of polar coding schemes is theoretically known (we have proven bounds!), while no such bounds are known for LDPC codes. There the usual approach is to simulate the coding scheme in order to get a good estimate of its performance. Unfortunately, for situations when one requires extremely small error probabilities like 10^{-17} (e.g., for a harddrive), simulation is not possible. Therefore, no one can *guarantee* that the LDPC scheme will perform as promised and required! LDPC and turbo coding schemes can show effects like an *error floor* at very low values of the error probability and there exist no known ways of proving where exactly such an error floor occurs. For such cases polar coding can offer certainty with a provably sufficient performance.

Moreover, it has been shown that by replacing the suboptimal successive cancellation decoding by other decoding schemes of higher complexity like, e.g., *successive cancellation list decoding* [TV15] or soft-output decoding schemes, [Arı08], [FB14] it is possible to significantly improve the performance of polar codes.

The main reason for the poorer performance of polar coding, however, lies hidden in the frozen bits that are — by definition — fixed and therefore irrevocably lost for communication. Since polarization happens relatively slowly, many of the frozen bits are used on channels that actually would have some small capacity left. In his Shannon Lecture, Arıkan presented a new coding scheme called *polarization-adjusted convolutional (PAC) codes* that tries to fix this. The idea of PAC codes is to use a simple (linear!) convolutional code around the polar coding scheme. This code will make sure that none of the bit channels of the polar code is used *only* for for frozen bits, so the remaining capacity within those almost useless channels can still be used. The convolutional code can be decoded with low complexity based on sequential decoding in a tree [Gal68, pp. 263–286]. The performance of such a combined convolutional/polar coding scheme is very close to the theoretical limit for any fixed blocklength! For more details we refer the interested reader to [Arı19b], [Arı19a].

We have discussed here only the simplest setup the way it was introduced in the seminal paper of Arıkan [Arı09]. There do exist ways of generalizing the concept of polarization to DMCs that are not restricted to have binary input alphabets [ŞTA09] [Şaş11a]. Even generalizations to multiple-user setups [TA12] or to channels with memory have been found [Şaş11b]. The principle of polarization can also applied to sources, leading to new algorithms for source compression [Arı10].

14.A Appendix: Landau Symbols

The Landau symbols are used to describe the asymptotic behavior of functions and series. They show up particularly in computer science to describe the complexity of algorithms as a function of the size of the input. For example, a binary search algorithm to search for a particular element in n elements takes a number of steps that lies in $\mathcal{O}(\log n)$, while the bubble-sort algorithm takes for the same task a number of steps in $\mathcal{O}(n^2)$.

Definition 14.37. Let $f, g\colon \mathbb{N} \to \mathbb{R}$ be given functions. There are five different Landau symbols:

- **Big-O notation:** We say that
$$f(n) \in \mathcal{O}(g(n)) \tag{14.330}$$
if there exists some constant $\mathrm{M} > 0$ and some $n_0 \in \mathbb{N}$ such that
$$|f(n)| \le \mathrm{M}|g(n)|, \quad \forall\, n \ge n_0. \tag{14.331}$$
In words: "f does not grow substantially faster than g."

- **Little-o notation:** We say that
$$f(n) \in \mathrm{o}(g(n)) \tag{14.332}$$
if for *all arbitrary* constants $\mathrm{M} > 0$ there exists some $n_0 \in \mathbb{N}$ such that
$$|f(n)| < \mathrm{M}|g(n)|, \quad \forall\, n \ge n_0. \tag{14.333}$$
In words: "f grows substantially slower than g."

- **Big-Omega notation:** We say that
$$f(n) \in \Omega(g(n)) \tag{14.334}$$
if there exists some constant $\mathrm{M} > 0$ and some $n_0 \in \mathbb{N}$ such that
$$|f(n)| \ge \mathrm{M}|g(n)|, \quad \forall\, n \ge n_0. \tag{14.335}$$
In words: "f does not grow substantially slower than g."

- **Little-omega notation:** We say that
$$f(n) \in \omega(g(n)) \tag{14.336}$$
if for *all arbitrary* constants $\mathrm{M} > 0$ there exists some $n_0 \in \mathbb{N}$ such that
$$|f(n)| > \mathrm{M}|g(n)|, \quad \forall\, n \ge n_0. \tag{14.337}$$
In words: "f grows substantially faster than g."

- **Theta notation:** We say that

$$f(n) \in \Theta(g(n)) \tag{14.338}$$

if

$$f(n) \in \mathcal{O}(g(n)) \text{ and } f(n) \in \Omega(g(n)). \tag{14.339}$$

In words: "f grows essentially as fast as g."

Note that $\mathcal{O}(g(n))$ is a *set* of functions that contains all functions satisfying (14.331). However, one often sees expressions like $f(n) = \mathcal{O}(g(n))$, which is supposed to mean the same thing, but is formally not correct.

In the definitions above we have assumed that the functions map natural numbers to reals. We can easily generalize this definition for arbitrary functions $f, g\colon \mathbb{R}^d \to \mathbb{R}$ and an arbitrary limit $\mathbf{x} \to \mathbf{x}_\infty$ by choosing a sequence $\{\mathbf{x}_n\}$ with $\lim_{n\to\infty} \mathbf{x}_n = \mathbf{x}_\infty$, defining

$$\tilde{f}(n) \triangleq f(\mathbf{x}_n), \qquad \tilde{g}(n) \triangleq g(\mathbf{x}_n), \quad n \in \mathbb{N}, \tag{14.340}$$

and then applying Definition 14.37 to $\tilde{f}$ and $\tilde{g}$.

While Definition 14.37 is quite easy to understand, it is often not very convenient. The following proposition simplifies life in that respect.

Proposition 14.38. Let $f, g\colon \mathbb{N} \to \mathbb{R}$ be given functions. Then the following relations hold:

$$f(n) \in \mathcal{o}(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0; \tag{14.341}$$

$$f(n) \in \omega(g(n)) \iff \lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = \infty; \tag{14.342}$$

$$f(n) \in \mathcal{O}(g(n)) \iff \overline{\lim_{n\to\infty}} \left|\frac{f(n)}{g(n)}\right| < \infty; \tag{14.343}$$

$$f(n) \in \Omega(g(n)) \iff \underline{\lim_{n\to\infty}} \left|\frac{f(n)}{g(n)}\right| > 0; \tag{14.344}$$

$$f(n) \in \Theta(g(n)) \iff 0 < \underline{\lim_{n\to\infty}} \left|\frac{f(n)}{g(n)}\right| \le \overline{\lim_{n\to\infty}} \left|\frac{f(n)}{g(n)}\right| < \infty; \tag{14.345}$$

$$f(n) \in \mathcal{O}(g(n)) \iff g(n) \in \Omega(f(n)); \tag{14.346}$$

$$f(n) \in \mathcal{o}(g(n)) \iff g(n) \in \omega(f(n)); \tag{14.347}$$

and

$$\mathcal{o}(g(n)) \subseteq \mathcal{O}(g(n)); \tag{14.348}$$

$$\omega(g(n)) \subseteq \Omega(g(n)); \tag{14.349}$$

$$\Theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n)). \tag{14.350}$$

Proof: The derivation of these relations is straightforward. We only show it for (14.341), but omit the rest. If $f(n) \in \mathcal{O}(g(n))$, then by definition for n large enough

$$|f(n)| < \mathrm{M}|g(n)|. \tag{14.351}$$

Hence,

$$\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \mathrm{M}. \tag{14.352}$$

Since this must hold for any $\mathrm{M} > 0$, we can choose M arbitrarily close to 0, thereby proving one direction.

On the other hand, assume that the right-hand side of (14.341) holds. Then for any $\epsilon > 0$ we can find some n_0 such that for all $n \geq n_0$

$$\left|\frac{f(n)}{g(n)}\right| < \epsilon, \tag{14.353}$$

or

$$|f(n)| < \epsilon|g(n)|. \tag{14.354}$$

Hence, setting $\mathrm{M} = \epsilon$ and applying Definition 14.37 we see that $f(n) \in \mathcal{O}(g(n))$. □

14.B Appendix: Concavity of Z(W) and Proof of (14.152) in Theorem 14.20

In the following proof we will make use of the Minkowski Inequality for sums.

Lemma 14.39 (Minkowski Inequality for Sums). Let $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ be nonnegative real numbers. Then for $p \geq 1$,

$$\left(\sum_{k=1}^{n} (\alpha_k + \beta_k)^p\right)^{\frac{1}{p}} \leq \left(\sum_{k=1}^{n} \alpha_k^p\right)^{\frac{1}{p}} + \left(\sum_{k=1}^{n} \beta_k^p\right)^{\frac{1}{p}}, \tag{14.355}$$

and for $p \leq 1$,

$$\left(\sum_{k=1}^{n} (\alpha_k + \beta_k)^p\right)^{\frac{1}{p}} \geq \left(\sum_{k=1}^{n} \alpha_k^p\right)^{\frac{1}{p}} + \left(\sum_{k=1}^{n} \beta_k^p\right)^{\frac{1}{p}}. \tag{14.356}$$

Proof: Assume $p \leq 1$ and define

$$f(x, y) \triangleq \left(x^{1/p} + y^{1/p}\right)^p, \quad x, y \geq 0. \tag{14.357}$$

It can be verified that $f(\cdot, \cdot)$ is convex over all pairs (x, y) in the first quarter of the plane, i.e., for all $\theta \in [0, 1]$ and any choice $x_0, y_0, x_1, y_1 \geq 0$,

$$f\big(\theta x_0 + (1-\theta)x_1, \theta y_0 + (1-\theta)y_1\big) \leq \theta f(x_0, y_0) + (1-\theta) f(x_1, y_1). \tag{14.358}$$

Hence, by adapting the Jensen Inequality (Theorem 2.1) to this two-dimensional setup, we obtain for a pair of nonnegative RVs $(\tilde{X}, \tilde{Y})$:

$$\mathsf{E}\left[\left(\tilde{X}^{1/p} + \tilde{Y}^{1/p}\right)^p\right] = \mathsf{E}\big[f(\tilde{X}, \tilde{Y})\big] \tag{14.359}$$

$$\geq f\big(\mathsf{E}[\tilde{X}], \mathsf{E}[\tilde{Y}]\big) \tag{14.360}$$

$$= \Big(\mathsf{E}\big[\tilde{X}\big]^{1/p} + \mathsf{E}\big[\tilde{Y}\big]^{1/p}\Big)^p. \tag{14.361}$$

Taking both sides to the power of $1/p$ and defining $X \triangleq \tilde{X}^{1/p}$ and $Y \triangleq \tilde{Y}^{1/p}$ yields

$$\Big(\mathsf{E}[(X+Y)^p]\Big)^{1/p} \geq \mathsf{E}\big[X^p\big]^{1/p} + \mathsf{E}\big[Y^p\big]^{1/p}. \tag{14.362}$$

Finally, by choosing X and Y to be uniform over $\{\alpha_1, \ldots, \alpha_n\}$ and $\{\beta_1, \ldots, \beta_n\}$, respectively, we obtain

$$\left(\frac{1}{n}\sum_{k=1}^{n}(\alpha_k + \beta_k)^p\right)^{\frac{1}{p}} \geq \left(\frac{1}{n}\sum_{k=1}^{n}\alpha_k^p\right)^{\frac{1}{p}} + \left(\frac{1}{n}\sum_{k=1}^{n}\beta_k^p\right)^{\frac{1}{p}}, \tag{14.363}$$

from which (14.356) follows by multiplying both sides by $n^{1/p}$.

The proof of (14.355) is identical apart from the fact that for $p \geq 1$ the function (14.357) is concave instead of convex and that therefore the direction of the inequality in the Jensen Inequality needs to be reversed. □

We are now ready to prove that the channel reliability parameter $\mathsf{Z}(\mathsf{W})$ is concave in W. To this end, let W_0 and W_1 be two binary-input DMCs and define for $\theta \in [0,1]$

$$\mathsf{W} \triangleq \theta\mathsf{W}_0 + (1-\theta)\mathsf{W}_1. \tag{14.364}$$

Then,

$$\mathsf{Z}(\mathsf{W}) = \sum_y \sqrt{W(y|0)\,W(y|1)} \tag{14.365}$$

$$= \sum_y \sqrt{W(y|0)\,W(y|1)} + \frac{1}{2}\sum_y W(y|0) + \frac{1}{2}\sum_y W(y|1) - 1 \tag{14.366}$$

$$= \frac{1}{2}\sum_y \left(\sqrt{W(y|0)} + \sqrt{W(y|1)}\right)^2 - 1 \tag{14.367}$$

$$= \frac{1}{2}\sum_y \left(\sum_x \sqrt{W(y|x)}\right)^2 - 1 \tag{14.368}$$

$$= \frac{1}{2}\sum_y \left(\sum_x \sqrt{\theta W_0(y|x) + (1-\theta)W_1(y|x)}\right)^2 - 1 \tag{14.369}$$

$$\geq \frac{1}{2}\sum_y \left(\sum_x \sqrt{\theta W_0(y|x)}\right)^2 + \frac{1}{2}\sum_y \left(\sum_x \sqrt{(1-\theta)W_1(y|x)}\right)^2 - 1 \tag{14.370}$$

$$= \theta\left(\frac{1}{2}\sum_y\left(\sum_x\sqrt{W_0(y|x)}\right)^2 - 1\right) + (1-\theta)\left(\frac{1}{2}\sum_y\left(\sum_x\sqrt{W_1(y|x)}\right)^2 - 1\right) \tag{14.371}$$

$$= \theta\mathsf{Z}(\mathsf{W}_0) + (1-\theta)\mathsf{Z}(\mathsf{W}_1). \tag{14.372}$$

Here the second equality holds because we add terms that add to zero; the inequality follows from Lemma 14.39 with $p = 1/2$, $\alpha_x = \theta W_0(y|x)$, and $\beta_x = (1-\theta)W_1(y|x)$; and for the last equality we follow the steps (14.365)–(14.368) in backward order.

We turn to the proof of the LHS in (14.152). To this end, we use Corollary 14.2 to write W^- as

$$W^-(y_1,y_2|u_1) = \frac{1}{2}\underbrace{W(y_1|u_1)\,W(y_2|0)}_{\triangleq W_0(y_1,y_2|u_1)} + \frac{1}{2}\underbrace{W(y_1|u_1\oplus 1)\,W(y_2|1)}_{\triangleq W_1(y_1,y_2|u_1)}. \tag{14.373}$$

This can be interpreted as a mixture (14.364) with $\theta = \frac{1}{2}$. Thus, we can use (14.372):

$$\mathsf{Z}(\mathsf{W}^-) \ge \frac{1}{2}\mathsf{Z}(\mathsf{W}_0) + \frac{1}{2}\mathsf{Z}(\mathsf{W}_1) \tag{14.374}$$

$$= \frac{1}{2}\sum_{y_1,y_2}\sqrt{W_0(y_1,y_2|0)\,W_0(y_1,y_2|1)} + \frac{1}{2}\sum_{y_1,y_2}\sqrt{W_1(y_1,y_2|0)\,W_1(y_1,y_2|1)} \tag{14.375}$$

$$= \frac{1}{2}\sum_{y_1,y_2}\sqrt{W(y_1|0)\,W(y_2|0)\,W(y_1|1)\,W(y_2|0)} + \frac{1}{2}\sum_{y_1,y_2}\sqrt{W(y_1|0\oplus 1)\,W(y_2|1)\,W(y_1|1\oplus 1)\,W(y_2|1)} \tag{14.376}$$

$$= \frac{1}{2}\sum_{y_2}W(y_2|0)\sum_{y_1}\sqrt{W(y_1|0)\,W(y_1|1)} + \frac{1}{2}\sum_{y_2}W(y_2|1)\sum_{y_1}\sqrt{W(y_1|0)\,W(y_1|1)} \tag{14.377}$$

$$= \frac{1}{2}\,\mathsf{Z}(\mathsf{W}) + \frac{1}{2}\,\mathsf{Z}(\mathsf{W}) \tag{14.378}$$

$$= \mathsf{Z}(\mathsf{W}). \tag{14.379}$$

This completes the proof of Theorem 14.20.

14.C Appendix: Proof of Theorem 14.24

Using the random channel selection process defined in (14.109)–(14.111) we define

$$Z_\ell \triangleq \mathsf{Z}(\mathbb{W}_\ell). \tag{14.380}$$

Together with Theorem 14.20 this leads to a class $\mathcal{Z}$ of random processes $\{Z_\ell\}_{\ell\in\mathbb{N}_0}$ that satisfy

$$Z_0 = z_0, \qquad \text{for some } z_0 \in [0,1], \tag{14.381}$$

$$\begin{cases} Z_\ell = Z_{\ell-1}^2 & \text{if } S_\ell = +, \\ Z_\ell \in \big[Z_{\ell-1}, 2Z_{\ell-1} - Z_{\ell-1}^2\big] & \text{if } S_\ell = -, \end{cases} \quad \ell = 1,2,\dots \tag{14.382}$$

We know from Corollary 14.23 that $\{Z_\ell\}$ converges (in probability) to a binary random variable Z_∞ with

$$\Pr[Z_\infty = 0] = 1 - \Pr[Z_\infty = 1] = \mathrm{I}(\mathsf{W}). \tag{14.383}$$

14.C.1 Converse Part

Fix some process $\{Z_\ell\} \in \mathcal{Z}$ and some $\beta > \frac{1}{2}$. We define another random process $\{\tilde{Z}_\ell\}$:

$$\tilde{Z}_0 \triangleq Z_0 = z_0, \tag{14.384}$$

$$\tilde{Z}_\ell = \begin{cases} \tilde{Z}_{\ell-1}^2 & \text{if } S_\ell = +, \\ \tilde{Z}_{\ell-1} & \text{if } S_\ell = -, \end{cases} \quad \ell = 1,2,\dots \tag{14.385}$$

Note that because of the assumption $\mathrm{I}(\mathsf{W}) < 1$ it follows that $0 < z_0 \le 1$. Comparing with (14.382) shows that $\{\tilde{Z}_\ell\}$ is dominated by $\{Z_\ell\}$, i.e., $\tilde{Z}_\ell \le Z_\ell$ and therefore

$$\Pr\Big[Z_\ell \ge 2^{-2^{\beta\ell}}\Big] \ge \Pr\Big[\tilde{Z}_\ell \ge 2^{-2^{\beta\ell}}\Big]. \tag{14.386}$$

Now we note that

$$\tilde{Z}_\ell = z_0^{2^L} \tag{14.387}$$

with

$$L = \sum_{j=1}^{\ell} \mathbb{1}\{S_j = +\}. \tag{14.388}$$

Hence,

$$\begin{aligned} \Pr\Big[\tilde{Z}_\ell \ge 2^{-2^{\beta\ell}}\Big] &= \Pr\Big[z_0^{2^L} \ge 2^{-2^{\beta\ell}}\Big] && (14.389)\\ &= \Pr\Big[2^L \log_2 z_0 \ge -2^{\beta\ell}\Big] && (14.390)\\ &= \Pr\Big[2^L \log_2(1/z_0) \le 2^{\beta\ell}\Big] && (14.391)\\ &= \Pr[L \le \beta\ell - \log_2\log_2(1/z_0)]. && (14.392) \end{aligned}$$

Since $\mathsf{E}[L] = \frac{1}{2}\ell$, $\beta > \frac{1}{2}$, and $z_0 > 0$, it follows from the law of large numbers that this probability tends to 1 as $\ell \to \infty$. This proves (14.209).

14.C.2 Direct Part

The proof of the direct part that we present here stems from [Tal17] and is much more direct than the original proof of Arıkan and Telatar [AT09].

We want to show that for any fixed $0 < \beta < \frac{1}{2}$,

$$\lim_{\ell\to\infty} \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta}}\Big] = \Pr[Z_\infty = 0]. \tag{14.393}$$

We are actually going to prove a stronger statement:

Lemma 14.40. Take the setup of Theorem 14.24. Then, for any $0 < \beta < \frac{1}{2}$,

$$\lim_{\ell_0\to\infty} \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta}}\,\forall\,\ell \ge \ell_0\Big] = \Pr[Z_\infty = 0]. \tag{14.394}$$

Note that this lemma has an "almost sure"-flavor[8] in contrast to (14.393) that follows the style of "convergence in probability".

We start by showing that (14.394) indeed implies (14.393). First, note that

$$\Pr\Big[Z_{\ell_0} \le 2^{-2^{\ell_0\beta}}\Big] \ge \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta}}\,\forall\,\ell \ge \ell_0\Big] \tag{14.395}$$

because the argument on the RHS contains in addition to the argument of the LHS some more RVs and therefore the probability of the RHS cannot be larger than the probability on the LHS. Thus, if (14.394) holds, then

$$\underline{\lim}_{\ell_0\to\infty} \Pr\Big[Z_{\ell_0} \le 2^{-2^{\ell_0\beta}}\Big] \ge \Pr[Z_\infty = 0]. \tag{14.396}$$

On the other hand, by contradiction, assume that there exists some $0 < \beta < \frac{1}{2}$ such that

$$\overline{\lim_{\ell\to\infty}} \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta}}\Big] > \Pr[Z_\infty = 0]. \tag{14.397}$$

Then, for an arbitrary $\epsilon > 0$,

$$\begin{aligned}
\Pr[Z_\infty = 0] &< \overline{\lim_{\ell\to\infty}} \Pr\Big[Z_\ell \le 2^{-2^{\ell\beta}}\Big] && \text{(by (14.397))} && (14.398)\\
&\le \overline{\lim_{\ell\to\infty}} \Pr[Z_\ell \le \epsilon] && (2^{-2^{\ell\beta}} \le \epsilon \text{ for } \ell \gg 1) && (14.399)
\end{aligned}$$

which means that Z_ℓ cannot converge to Z_∞, which is a contradiction to our setup. Thus, if (14.394) holds, also (14.393) must hold.

Proof of Lemma 14.40: Note that by (14.382) any process $\{Z_\ell\} \in \mathcal{Z}$ satisfies

$$Z_{\ell+1} \le 2Z_\ell^{D_\ell}, \quad \ell = 0, 1, \ldots, \tag{14.400}$$

[8] See Section 20.2 for a discussion about random convergence.

where

$$D_\ell = \begin{cases} 2 & \text{if } S_{\ell+1} = +, \\ 1 & \text{if } S_{\ell+1} = -, \end{cases} \quad \ell = 0, 1, \dots \tag{14.401}$$

Let $\epsilon_{\mathrm{a}}, \epsilon_{\mathrm{b}} \in \left(0, \frac{1}{2}\right)$ and $\ell_{\mathrm{a}}, \ell_{\mathrm{b}} \in \mathbb{N}$, $\ell_{\mathrm{a}} < \ell_{\mathrm{b}}$ be parameters. We now define three events,

$$\mathcal{E}_{\mathrm{a}} \triangleq \{|Z_\ell - Z_\infty| \le \epsilon_{\mathrm{a}} \,\forall\, \ell \ge \ell_{\mathrm{a}}\}, \tag{14.402}$$

$$\mathcal{E}_{\mathrm{b}} \triangleq \left\{ \left| \frac{|\{\ell_{\mathrm{a}} \le j < \ell \colon D_j = 2\}|}{\ell - \ell_{\mathrm{a}}} - \frac{1}{2} \right| \le \epsilon_{\mathrm{b}} \;\&\; \left| \frac{|\{\ell_{\mathrm{a}} \le j < \ell \colon D_j = 1\}|}{\ell - \ell_{\mathrm{a}}} - \frac{1}{2} \right| \le \epsilon_{\mathrm{b}} \,\forall\, \ell \ge \ell_{\mathrm{b}} \right\}, \tag{14.403}$$

$$\mathcal{E}_{\mathrm{c}} \triangleq \{Z_\infty = 0\}, \tag{14.404}$$

and a parameter θ such that

$$\epsilon_{\mathrm{a}}^{\theta} = \frac{1}{2}, \tag{14.405}$$

i.e.,

$$\theta = \frac{\log(1/2)}{\log(\epsilon_{\mathrm{a}})}. \tag{14.406}$$

Note that $\theta \in (0, 1)$ and that $\theta \to 0$ as $\epsilon_{\mathrm{a}} \to 0$.

In the situation when event $\mathcal{E}_{\mathrm{a}}$ and event $\mathcal{E}_{\mathrm{c}}$ occur together, i.e., under $\mathcal{E}_{\mathrm{a}} \cap \mathcal{E}_{\mathrm{c}}$, we have for all $\ell \ge \ell_{\mathrm{a}}$

$$Z_\ell \le \epsilon_{\mathrm{a}} \tag{14.407}$$

and thus

$$Z_\ell^{\theta} \le \epsilon_{\mathrm{a}}^{\theta} = \frac{1}{2} \implies 2 \le Z_\ell^{-\theta}. \tag{14.408}$$

So, under $\mathcal{E}_{\mathrm{a}} \cap \mathcal{E}_{\mathrm{c}}$ and for $\ell \ge \ell_{\mathrm{a}}$, (14.400) can be rewritten as

$$Z_{\ell+1} \le Z_\ell^{D_\ell - \theta}, \quad \ell \ge \ell_{\mathrm{a}}. \tag{14.409}$$

Next, we also include event $\mathcal{E}_{\mathrm{b}}$, i.e., we now consider the situation when $\mathcal{E}_{\mathrm{a}} \cap \mathcal{E}_{\mathrm{b}} \cap \mathcal{E}_{\mathrm{c}}$. We start with $Z_{\ell_{\mathrm{a}}}$ and repeatedly apply (14.409). We know that roughly half of the time Z_ℓ will be taken to the power of $1 - \theta$ and half of the time to the power of $2 - \theta$. Using $\mathcal{E}_{\mathrm{b}}$ we can bound these numbers, so that for all $\ell \ge \ell_{\mathrm{b}}$ we obtain

$$Z_\ell \le Z_{\ell_{\mathrm{a}}}^{(1-\theta)^{(\ell-\ell_{\mathrm{a}})\left(\frac{1}{2}+\epsilon_{\mathrm{b}}\right)} \cdot (2-\theta)^{(\ell-\ell_{\mathrm{a}})\left(\frac{1}{2}-\epsilon_{\mathrm{b}}\right)}}, \quad \ell \ge \ell_{\mathrm{b}}. \tag{14.410}$$

Using (14.407) and the assumption that $\epsilon_{\mathrm{a}} < \frac{1}{2}$, we can upper-bound $Z_{\ell_{\mathrm{a}}}$ by 2^{-1}:

$$Z_\ell \le 2^{-(1-\theta)^{(\ell-\ell_{\mathrm{a}})\left(\frac{1}{2}+\epsilon_{\mathrm{b}}\right)}\cdot(2-\theta)^{(\ell-\ell_{\mathrm{a}})\left(\frac{1}{2}-\epsilon_{\mathrm{b}}\right)}} \tag{14.411}$$

$$= 2^{-2^{\left(\frac{1}{2}-\Delta\right)\ell}} \tag{14.412}$$

where the last equality should be read as definition of Δ, i.e.,

$$\Delta = \frac{1}{2} - \frac{\ell-\ell_{\mathrm{a}}}{\ell}\left(\frac{1}{2}+\epsilon_{\mathrm{b}}\right)\log_2(1-\theta) - \frac{\ell-\ell_{\mathrm{a}}}{\ell}\left(\frac{1}{2}-\epsilon_{\mathrm{b}}\right)\log_2(2-\theta) \tag{14.413}$$

$$\begin{aligned} &= \frac{1}{2} - \frac{1}{2}\log_2(1-\theta) - \frac{1}{2}\log_2(2-\theta) \\ &\quad - \epsilon_{\mathrm{b}}\log_2(1-\theta) + \epsilon_{\mathrm{b}}\log_2(2-\theta) \\ &\quad + \frac{\ell_{\mathrm{a}}}{\ell}\left(\frac{1}{2}+\epsilon_{\mathrm{b}}\right)\log_2(1-\theta) + \frac{\ell_{\mathrm{a}}}{\ell}\left(\frac{1}{2}-\epsilon_{\mathrm{b}}\right)\log_2(2-\theta). \end{aligned} \tag{14.414}$$

Note that by choosing ϵ_{a} sufficiently small such that θ becomes sufficiently small, by choosing ϵ_{b} sufficiently small, and by choosing ℓ_{b} sufficiently large such that for all $\ell \ge \ell_{\mathrm{b}}$ the term $\frac{\ell_{\mathrm{a}}}{\ell}$ becomes sufficiently small, Δ can be made arbitrarily close to zero.

Thus, we see that for any $0 < \beta < \frac{1}{2}$, we can choose ϵ_{a}, ϵ_{b} small enough and ℓ_{b} large enough such that the event $\mathcal{E}_{\mathrm{a}} \cap \mathcal{E}_{\mathrm{b}} \cap \mathcal{E}_{\mathrm{c}}$ induces the event

$$\Big\{ Z_\ell \le 2^{-2^{\beta\ell}} \,\forall\, \ell \ge \ell_{\mathrm{b}} \Big\}. \tag{14.415}$$

Therefore,

$$\Pr\Big[Z_\ell \le 2^{-2^{\beta\ell}} \,\forall\, \ell \ge \ell_{\mathrm{b}} \Big] \ge \Pr(\mathcal{E}_{\mathrm{a}} \cap \mathcal{E}_{\mathrm{b}} \cap \mathcal{E}_{\mathrm{c}}). \tag{14.416}$$

So it remains to investigate $\Pr(\mathcal{E}_{\mathrm{a}} \cap \mathcal{E}_{\mathrm{b}} \cap \mathcal{E}_{\mathrm{c}})$.

To this end, we note that since Z_ℓ converges to Z_∞ in probability, i.e.,

$$\lim_{\ell\to\infty} \Pr[|Z_\ell - Z_\infty| \le \epsilon_{\mathrm{a}}] = 1, \tag{14.417}$$

it also must hold that

$$\lim_{\ell_{\mathrm{a}}\to\infty} \Pr(\mathcal{E}_{\mathrm{a}}) = 1. \tag{14.418}$$

Thus, for an arbitrary $\delta_{\mathrm{a}} > 0$, we can find an ℓ_{a} large enough such that

$$\Pr(\mathcal{E}_{\mathrm{a}}) \ge 1 - \delta_{\mathrm{a}}. \tag{14.419}$$

Next, note that since $\{S_\ell\}$ is IID uniform, also $\{D_\ell\}$ is IID uniform, and thus by the strong law of large numbers,

$$\lim_{\ell_{\mathrm{b}}\to\infty} \Pr(\mathcal{E}_{\mathrm{b}}) = 1. \tag{14.420}$$

Thus, for an arbitrary $\delta_{\mathrm{b}} > 0$, we can find an ℓ_{b} large enough such that

$$\Pr(\mathcal{E}_{\mathrm{b}}) \geq 1 - \delta_{\mathrm{b}}. \tag{14.421}$$

Hence, for arbitrary $\delta_{\mathrm{a}}, \delta_{\mathrm{b}} > 0$ we can find $\ell_{\mathrm{a}} < \ell_{\mathrm{b}}$ large enough such that

$$\begin{aligned}
\Pr(\mathcal{E}_{\mathrm{a}} \cap \mathcal{E}_{\mathrm{b}} \cap \mathcal{E}_{\mathrm{c}}) &= 1 - \Pr(\mathcal{E}_{\mathrm{a}}^{\mathrm{c}} \cup \mathcal{E}_{\mathrm{b}}^{\mathrm{c}} \cup \mathcal{E}_{\mathrm{c}}^{\mathrm{c}}) && (14.422)\\
&\geq 1 - \big(\Pr(\mathcal{E}_{\mathrm{a}}^{\mathrm{c}}) + \Pr(\mathcal{E}_{\mathrm{b}}^{\mathrm{c}}) + \Pr(\mathcal{E}_{\mathrm{c}}^{\mathrm{c}})\big) && (14.423)\\
&= 1 - \big(1 - \Pr(\mathcal{E}_{\mathrm{a}}) + 1 - \Pr(\mathcal{E}_{\mathrm{b}}) + 1 - \Pr(\mathcal{E}_{\mathrm{c}})\big) && (14.424)\\
&= \Pr(\mathcal{E}_{\mathrm{a}}) + \Pr(\mathcal{E}_{\mathrm{b}}) + \Pr(\mathcal{E}_{\mathrm{c}}) - 2 && (14.425)\\
&\geq 1 - \delta_{\mathrm{a}} + 1 - \delta_{\mathrm{b}} + \Pr(\mathcal{E}_{\mathrm{c}}) - 2 && (14.426)\\
&= \Pr[Z_\infty = 0] - \delta_{\mathrm{a}} - \delta_{\mathrm{b}}, && (14.427)
\end{aligned}$$

where the first inequality follows from the Union Bound, and the second from (14.419) and (14.421). In combination with (14.416) this proves

$$\Pr\Big[Z_\ell \leq 2^{-2^{\beta\ell}} \,\forall\, \ell \geq \ell_{\mathrm{b}}\Big] \geq \Pr[Z_\infty = 0] - \delta_{\mathrm{a}} - \delta_{\mathrm{b}}. \tag{14.428}$$

Since $\delta_{\mathrm{a}}, \delta_{\mathrm{b}}$ are arbitrary and since the probability expression on the LHS increases with ℓ_{b} (because the number of involved RVs decreases!), this proves

$$\lim_{\ell_{\mathrm{b}} \to \infty} \Pr\Big[Z_\ell \leq 2^{-2^{\beta\ell}} \,\forall\, \ell \geq \ell_{\mathrm{b}}\Big] \geq \Pr[Z_\infty = 0]. \tag{14.429}$$

The other direction

$$\lim_{\ell_{\mathrm{b}} \to \infty} \Pr\Big[Z_\ell \leq 2^{-2^{\beta\ell}} \,\forall\, \ell \geq \ell_{\mathrm{b}}\Big] \leq \Pr[Z_\infty = 0] \tag{14.430}$$

also holds because for an arbitrary ℓ_{b},

$$\begin{aligned}
\Pr\Big[Z_\ell \leq 2^{-2^{\beta\ell}} \,\forall\, \ell \geq \ell_{\mathrm{b}}\Big] &\leq \Pr\Big[\lim_{\ell \to \infty} Z_\ell = 0\Big] && (14.431)\\
&= \Pr[Z_\infty = 0], && (14.432)
\end{aligned}$$

where the inequality can be argued in a analogous fashion as for the inequality in (14.395).

This completes the proof. □

Chapter 15

Joint Source and Channel Coding

In Chapter 11 we have focused exclusively on the data transmission part of the general system shown in Figure 11.1, and we have assumed that the random message M is taken uniformly from a large set of possible messages $\mathcal{M}$. In reality, we will rarely see such a random message, but we will rather encounter a source of limited alphabet size emitting a *sequence of source symbols* that shall be transmitted. Obviously, such a sequence can be taken as the random message, i.e., the system developed in Chapter 11 will work here, too. The question, however, is whether we can do better.

If, for example, we transmit a sequence of a million bits and one single bit is wrongly decoded, then in Chapter 11's terminology we have failed to transmit the message — in spite of the fact that 999'999 bits have arrived correctly. So, from a practical point of view, we are much more interested in the average bit error probability P_{b} rather than the block error probability $P_{\mathrm{e}}^{(n)}$ (compare also with Section 13.3.3).

In this chapter, we ask the question whether we can increase our transmission rate if we only require that P_{b} be small, rather than $P_{\mathrm{e}}^{(n)}$. We shall see that we cannot.

15.1 Information Transmission System

We would like to combine our knowledge of source compression and data transmission for the design of an *information transmission system*. To this end, consider Figure 15.1: Given is a general discrete stationary source (DSS) with r-ary alphabet $\mathcal{U}$ that we would like to transmit over a discrete memoryless channel (DMC) with input and output alphabets $\mathcal{X}$ and $\mathcal{Y}$, respectively. We assume that the source generates a symbol every T_{s} seconds, i.e., the source generates uncertainty at a rate of

$$\frac{\mathrm{H}(\{U_k\})}{\mathsf{T}_{\mathrm{s}}} \text{ bits/s.} \tag{15.1}$$

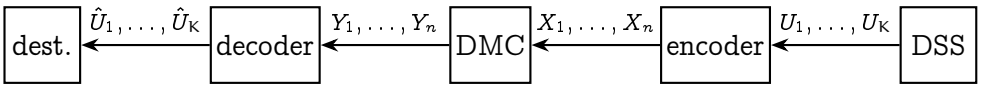


Figure 15.1: Information transmission system using a joint source channel coding scheme.

In our system we choose to jointly process K source symbols that are transformed into a channel codeword of length n and then sent over the channel.

Definition 15.1. A *joint source channel coding scheme* consists of an encoder that maps every source sequence $(U_1, \dots, U_K)$ into a channel input sequence $(X_1, \dots, X_n)$ and of a decoder that makes for every received channel output sequence $(Y_1, \dots, Y_n)$ a guess $(\hat{U}_1, \dots, \hat{U}_K)$ of the transmitted source sequences. We define the probability of a decoding error by

$$P_e^{(K)} \triangleq \Pr\Big[U_1^K \neq \hat{U}_1^K\Big]. \tag{15.2}$$

Obviously, for this system to work in the long run, the channel transmission must be clocked in the correct speed, i.e., if we denote the duration of a channel symbol by T_c, then we must have

$$KT_s = nT_c. \tag{15.3}$$

We will now derive a very fundamental result that shows when it is possible to transmit the source's random messages over the given channel depending on the source entropy and the channel capacity.

As so often we start with the converse result, i.e., we will prove an upper limitation on the source entropy, for which any transmission system will not work.

15.2 Converse to the Information Transmission Theorem

Recall the Fano Inequality (Lemma 11.25 or Corollary 11.27) and apply it to the random vectors U_1^K and $\hat{U}_1^K$:

$$\begin{aligned} \mathrm{H}(U_1^K|\hat{U}_1^K) &\le \underbrace{\mathrm{H_b}\Big(P_e^{(K)}\Big)}_{\le \log 2} + P_e^{(K)} \log\Big(\underbrace{|\mathcal{U}|^K - 1}_{\le |\mathcal{U}|^K}\Big) && (15.4)\\ &\le \log 2 + P_e^{(K)} K \log|\mathcal{U}|. && (15.5) \end{aligned}$$

Hence,

$$\begin{aligned} \frac{\mathrm{H}(\{U_k\})}{T_s} &= \frac{1}{T_s} \lim_{K\to\infty} \frac{\mathrm{H}(U_1^K)}{K} && (15.6)\\ &\le \frac{1}{KT_s} \mathrm{H}(U_1^K) && (15.7) \end{aligned}$$

$$= \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{H}(U_1^{\mathsf{K}}|\hat{U}_1^{\mathsf{K}}) + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{I}(U_1^{\mathsf{K}}; \hat{U}_1^{\mathsf{K}}) \tag{15.8}$$

$$\le \frac{\log 2}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} + \frac{1}{\mathsf{T}_{\mathrm{s}}} P_{\mathrm{e}}^{(\mathsf{K})} \log |\mathcal{U}| + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{I}(U_1^{\mathsf{K}}; \hat{U}_1^{\mathsf{K}}) \tag{15.9}$$

$$\le \frac{\log 2}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} + \frac{1}{\mathsf{T}_{\mathrm{s}}} P_{\mathrm{e}}^{(\mathsf{K})} \log |\mathcal{U}| + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{I}(X_1^n; Y_1^n) \tag{15.10}$$

$$\le \frac{\log 2}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} + \frac{1}{\mathsf{T}_{\mathrm{s}}} P_{\mathrm{e}}^{(\mathsf{K})} \log |\mathcal{U}| + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} n\mathsf{C} \tag{15.11}$$

$$= \frac{\log 2}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} + \frac{1}{\mathsf{T}_{\mathrm{s}}} P_{\mathrm{e}}^{(\mathsf{K})} \log |\mathcal{U}| + \frac{\mathsf{C}}{\mathsf{T}_{\mathrm{c}}}. \tag{15.12}$$

Here, (15.6) follows by the definition of the entropy rate; (15.7) follows because $\mathrm{H}(U_1^{\mathsf{K}})/\mathsf{K}$ is decreasing in K (see Theorem 6.24); in the subsequent equality (15.8) we use the definition of mutual information; in (15.9) we use (15.5) (Fano); the subsequent inequality (15.10) follows from the Data Processing Inequality (Lemma 11.30); (15.11) relies on our assumption of the channel being a DMC without feedback and the definition of capacity such that we can use (11.101); and the finally equality (15.12) is due to (15.3).

Hence, any joint source channel coding scheme with $P_{\mathrm{e}}^{(\mathsf{K})} \to 0$ for $\mathsf{K} \to \infty$ must satisfy

$$\boxed{\frac{\mathrm{H}(\{U_k\})}{\mathsf{T}_{\mathrm{s}}} \le \frac{\mathsf{C}}{\mathsf{T}_{\mathrm{c}}}.} \tag{15.13}$$

In other words, if

$$\frac{\mathrm{H}(\{U_k\})}{\mathsf{T}_{\mathrm{s}}} > \frac{\mathsf{C}}{\mathsf{T}_{\mathrm{c}}}, \tag{15.14}$$

then the error probability is bounded from below and cannot tend to zero even if we allow for infinite delay $\mathsf{K} \to \infty$.

15.3 Achievability of the Information Transmission Theorem

Next we will turn to the positive side of the theorem that tells us what we can do. As a matter of fact, it will turn out that we can prove an even better result: Not only is it possible to transmit the stationary source $\{U_k\}$ reliably over the DMC as long as (15.13) is satisfied, but we can do this in a way that separates the source from the channel. We can firstly compress the source using some data compression scheme that is completely ignorant of the DMC, and then apply a standard channel code for the given DMC that assumes a uniform message source and is completely ignorant of the given stationary source.

15.3.1 Ergodicity

The derivation of the achievability part makes the assumption of *ergodicity*. While we do not investigate whether this assumption holds or not, we try to motivate why it actually is only a very weak restriction that should be satisfied for most practical situations: We once again point out that the output of an optimal source compression scheme is (almost) IID and therefore "automatically" ergodic (compare with Remark 11.1).

Proposition 15.2. The codeword digits $C_{k'}$ put out by an optimal source coding scheme are all (almost) equally likely and (almost) independent of the past.

Proof: By contradiction, assume that the codeword digits are not equally likely. In this case we could find a Huffman code for the output of the ideal source coding scheme that combines the least likely digits together, resulting in a new sequence that is more efficiently representing the source than the output of the ideal source coding scheme. This is a contradiction to our optimality assumption of the ideal source coding scheme. Similarly we can argue if the current codeword was dependent on the past: In this case we could find a new adaptive Huffman code that can represent the current sequence of codeword digits using less digits based on the knowledge of the past. Again this is a contradiction to the optimality of our source coding scheme. □

Note that this proposition and its proof are not rigorous. However, this is not crucial, because we do not directly need the result. The only property that we need is that the sequence $\{C_{k'}\}$ satisfies the law of large numbers. Since an IID sequence definitely does do that, it is hopefully convincing that an almost IID sequence also does...

We also would like to point out that ergodicity is closely related with the AEP (see Theorem 20.4). For more details we refer to Chapter 20.

15.3.2 Achievable Joint Source Channel Coding Scheme

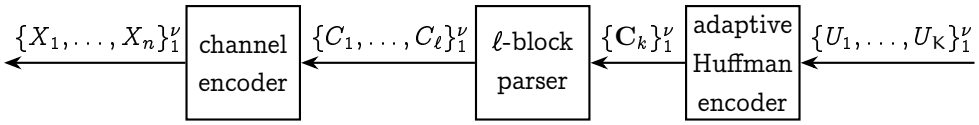


Figure 15.2: An encoder for the information transmission system of Figure 15.1.

We fix some $\epsilon > 0$ and then design an encoder as shown in Figure 15.2. In a first step, this encoder applies an optimal block–to–variable-length source encoder (e.g., an adaptive Huffman encoder) that maps a length-K source sequence into a D-ary codeword $\mathbf{C}_k$. We know from our analysis of source

compression that for any stationary source $\{U_k\}$ and for the given $\epsilon > 0$, we can choose a parser length K large enough such that the average source codeword length $\mathsf{E}[L_k]$ of the optimal D-ary codewords $\mathbf{C}_k$ satisfies

$$\frac{\mathsf{H}(\{U_k\})}{\log \mathsf{D}} \le \frac{\mathsf{E}[L_k]}{\mathsf{K}} \le \frac{\mathsf{H}(\{U_k\})}{\log \mathsf{D}} + \epsilon \quad \text{(for K large enough)} \tag{15.15}$$

(see Theorem 7.3, and compare with Theorem 7.12 and 8.11). As mentioned, K denotes the parser length (of the source coding scheme), i.e., each D-ary source codeword $\mathbf{C}_k$ describes K source symbols.

We now use this optimal source coding scheme many times in sequence, say ν times ($\nu \gg 1$), such that the total resulting D-ary output sequence $(\mathbf{C}_1, \dots, \mathbf{C}_\nu)$ has a total (random) length $\sum_{k=1}^{\nu} L_k$.

We next apply an ℓ-block parser to this D-ary output sequence $\{C_{k'}\}$ and split it up into ν equally long strings. In order to find out how we should choose ℓ, we need to consider the (random) length of the output sequence $\{C_{k'}\}$ of the adaptive Huffman encoder. To this end, as already discussed in Section 15.3.1, we assume that this output sequence satisfies the weak law of large numbers.

If the weak law of large numbers can be applied, then the probability of $\frac{1}{\nu}\sum_{k=1}^{\nu} L_k$ being close to $\mathsf{E}[L_k]$ tends to 1 for $\nu \uparrow \infty$, i.e., we can choose ν large enough such that

$$\Pr\left[\frac{1}{\nu}\sum_{k=1}^{\nu} L_k \le \mathsf{E}[L_k] + \epsilon\right] \ge 1 - \epsilon \quad \text{(for } \nu \text{ large enough)}. \tag{15.16}$$

Based on this observation, we now choose

$$\ell \triangleq \lceil \mathsf{E}[L_k] + \epsilon \rceil. \tag{15.17}$$

If the (random) length of $\{C_{k'}\}$ is less than $\nu\ell$, then we simply fill in random D-ary digits at the end.[1] On the other hand, it is also possible that the random length of $\{C_{k'}\}$ is too long:

$$\sum_{k=1}^{\nu} L_k > \nu\ell. \tag{15.18}$$

In this case the block-parser will discard some code digits, which will then result in a decoding error. However, luckily, the probability of this event is very small: From (15.16) we know that

$$\Pr\left[\sum_{k=1}^{\nu} L_k > \nu\ell\right] = \Pr\left[\frac{1}{\nu}\sum_{k=1}^{\nu} L_k > \lceil \mathsf{E}[L_k] + \epsilon \rceil\right] \tag{15.19}$$

$$\le \Pr\left[\frac{1}{\nu}\sum_{k=1}^{\nu} L_k > \mathsf{E}[L_k] + \epsilon\right] \tag{15.20}$$

$$\le \epsilon \qquad \text{(for } \nu \text{ large enough)}. \tag{15.21}$$

[1] Note that since the codewords $\mathbf{C}_k$ are prefix-free and we know their number (i.e., ν), it is easily possible to discard these randomly generated filling digits at the decoder.

Finally, the length-ℓ D-ary sequences are used as input for a usual channel coding scheme for the given DMC. This coding scheme will work reliably as long as its rate is less than the DMC's capacity.

Hence, we have two types of errors: Either the source compression scheme generates a too long compressed sequence (see (15.18) and (15.21)) or the channel introduces a too strong error for our channel coding scheme:

$$\begin{aligned} P_{\mathrm{e}}^{(\nu\mathrm{K})} &= \Pr[U_1^{\nu\mathrm{K}} \neq \hat{U}_1^{\nu\mathrm{K}}] && (15.22)\\ &\le \underbrace{\Pr(\text{compression error})}_{\substack{\le \epsilon \text{ by (15.21) if } \nu \\ \text{is large enough}}} + \underbrace{\Pr(\text{channel error})}_{\substack{\le \epsilon \text{ if } \mathrm{R} < \mathrm{C} \text{ and} \\ n \text{ is large enough}}} && (15.23)\\ &\le \epsilon + \epsilon = 2\epsilon \qquad (\text{for } \mathrm{R} < \mathrm{C} \text{ and for } \nu \text{ and } n \text{ large enough}). && (15.24) \end{aligned}$$

It only remains to make sure that the code rate R of our channel coding scheme really is below capacity. For an arbitrary $\epsilon' > 0$, we have

$$\begin{aligned} \mathrm{R} &= \frac{\log \mathrm{D}^{\ell}}{n} && (15.25)\\ &= \frac{\ell \log \mathrm{D}}{n} && (15.26)\\ &= \frac{\lceil \mathrm{E}[L_k] + \epsilon \rceil \log \mathrm{D}}{n} \qquad (\text{by (15.17)}) && (15.27)\\ &< \frac{1}{n}(\mathrm{E}[L_k] + \epsilon + 1) \log \mathrm{D} && (15.28)\\ &\le \frac{1}{n}\left(\mathrm{K} \cdot \left(\frac{\mathrm{H}(\{U_k\})}{\log \mathrm{D}} + \epsilon\right) + \epsilon + 1\right) \log \mathrm{D} \qquad (\text{by (15.15), } \mathrm{K} \gg 1) && (15.29)\\ &= \frac{\mathrm{K}}{n}\,\mathrm{H}(\{U_k\}) + \frac{\mathrm{K}\epsilon \log \mathrm{D}}{n} + \frac{(\epsilon+1)\log \mathrm{D}}{n} && (15.30)\\ &= \frac{\mathrm{T_c}}{\mathrm{T_s}}\,\mathrm{H}(\{U_k\}) + \frac{\epsilon \mathrm{T_c} \log \mathrm{D}}{\mathrm{T_s}} + \frac{(\epsilon+1)\mathrm{T_c} \log \mathrm{D}}{\mathrm{T_s K}} \qquad (\text{by (15.3)}) && (15.31)\\ &\le \frac{\mathrm{T_c}}{\mathrm{T_s}}\,\mathrm{H}(\{U_k\}) + \epsilon' \quad (\text{for } \epsilon \text{ small and K large enough}). && (15.32) \end{aligned}$$

Note that, because of (15.3), if K gets large, also n gets large.

Hence, we can guarantee that $\mathrm{R} < \mathrm{C}$ (and thereby that our system works) as long as

$$\boxed{\frac{\mathrm{H}(\{U_k\})}{\mathrm{T_s}} < \frac{\mathrm{C}}{\mathrm{T_c}}.} \qquad (15.33)$$

15.4 Joint Source and Channel Coding

We have proven the following fundamental result.

Theorem 15.3 (Information Transmission Theorem).
Assume a DMC of channel capacity C and a finite-alphabet stationary and ergodic stochastic source $\{U_k\}$. Then, if

$$\frac{\mathrm{H}(\{U_k\})}{\mathsf{T}_\mathrm{s}} \text{ bits/s} < \frac{\mathrm{C}}{\mathsf{T}_\mathrm{c}} \text{ bits/s} \tag{15.34}$$

(where T_s and T_c are the symbol durations of the source and the channel, respectively), there exists a joint source channel coding scheme with a probability of a decoding error $P_\mathrm{e}^{(\mathrm{K})} \to 0$ as $\mathrm{K} \to \infty$.

Conversely, for any stationary stochastic source $\{U_k\}$ with

$$\frac{\mathrm{H}(\{U_k\})}{\mathsf{T}_\mathrm{s}} \text{ bits/s} > \frac{\mathrm{C}}{\mathsf{T}_\mathrm{c}} \text{ bits/s}, \tag{15.35}$$

the probability of a decoding error $P_\mathrm{e}^{(\mathrm{K})}$ cannot tend to zero, i.e., it is not possible to send the source data over the DMC with arbitrarily low error probability.

Note that in the proofs given above we actually used two different approaches in the converse and in the achievability part:

Approach 1: General Design: We design an encoder that directly maps the source output sequence into a channel input sequence. The decoder then receives a channel output sequence and needs to guess which source sequence has been sent. This approach has been used in the converse.

Approach 2: Source Channel Separation Coding Scheme: We compress the source into its most efficient representation and then use a standard channel code (fit for the given channel and assuming a uniform message) to transmit the compressed source codewords. The receiver firstly guesses which codeword has been sent to recover the source codeword and then decompresses this compressed source description. This approach has been used in the achievability proof.

A couple of remarks:

- Approach 2 is a special case of Approach 1.
- In Approach 2 the standard channel code is designed assuming that the input is uniformly distributed. This works even if the source is far from being uniform because a source compression scheme produces an output that is (almost) uniform.
- Approach 2 is much easier than Approach 1 because we decouple the channel from the source: If we exchange the source then in Approach 1

we need to redesign the whole system, while in Approach 2 we only need to find a new source coding scheme. As a matter of fact, if we use a universal source coding scheme, we do not need to change anything at all! Similarly, if we exchange the DMC, then we only need to change the channel code design.

- Note that it is **not** *a priori* clear that Approach 2 is optimal because the data compression scheme completely ignores the channel and the channel coding scheme completely ignores the source. Luckily, we have been able to show that Approach 2 is as good as Approach 1.

Hence, we see that we have not only proven the Information Transmission Theorem, but we have actually shown that Approach 2 works! I.e., we can separate source from channel and design a system consisting of two completely separate parts.

Corollary 15.4 (The Source Channel Coding Separation Theorem).
Consider a finite-alphabet stationary and ergodic stochastic source $\{U_k\}$ and a DMC of capacity C such that (15.34) is satisfied, i.e.,

$$\frac{\mathrm{H}(\{U_k\})}{\mathrm{T_s}} \text{ bits/s} < \frac{\mathrm{C}}{\mathrm{T_c}} \text{ bits/s}. \tag{15.36}$$

An optimal way of transmitting the source sequence reliably over the DMC is to firstly apply an optimal data compression scheme (like Lempel–Ziv or adaptive Huffman) to the source output sequence, and then to transmit the compressed source codewords using an optimal channel coding scheme designed for the DMC. Note that the design of the source coding scheme is independent of the channel and the design of the channel coding scheme is independent of the source.

We would like to remark that nature does usually not follow the two-stage design of Approach 2, but uses a direct source-channel code. Examples:

- Human oral languages, transmitted through air. Even with a huge amount of noise and distortion (e.g., party with loud music and many people talking at the same time) we are able to understand our conversation partner!
- English text that is sent over an erasure channel: Even if about 50 percent of the letters are erased, we can still decipher the text!

Also note that our proof only holds for a DMC! There are examples of channels like, e.g., some multiple-user channels, where the two-stage approach is strictly suboptimal.

15.5 Rate of a Joint Source Channel Coding Scheme

The perhaps most fundamental difference between the Information Transmission Theorem (Theorem 15.3) and the channel coding theorem in Chapter 11 (Theorem 11.34) is that in the latter we have a free parameter R that needs to be chosen appropriately (i.e., $R < C$) in order to make sure that the system can work. Here in Theorem 15.3, however, we are given both the DMC with capacity C and clock T_c and the source with entropy rate $H(\{U_k\})$ and clock T_s. We then cannot anymore adapt these parameters to make them match (unless we start to change the channel or the source!), but have to face either the situation (15.34), which works, or (15.35), which does not work.

Nevertheless, as we have seen in the derivation of Section 15.3, we still implicitly have a rate R internally in the encoder: The second half of the encoder of the source channel separation coding scheme consists of a standard channel code as discussed in Chapter 11 and has a rate that is, as usual, defined as

$$R = \frac{\log(\#\text{ of codewords})}{n}. \tag{15.37}$$

Note that in this case the number of codewords depends on how strongly the K source digits can be compressed, i.e., it depends on the entropy rate of the source! Recalling the derivation of the rate in (15.25)–(15.32) we see that, in an ideal system and asymptotically for $K \to \infty$, we have

$$R = \frac{T_c}{T_s} H(\{U_k\}). \tag{15.38}$$

Note that this matches perfectly to the requirement that $R < C$, i.e., that (15.33) must be satisfied.

Also note that if the source happens to be a perfectly compressed binary source in the first place, i.e., if $\{U_k\}$ is an IID uniform binary sequence with $H(\{U_k\}) = 1$ bit, then we can omit the source compression scheme and we have[2]

$$R = \frac{K}{n}, \tag{15.39}$$

which by (15.3) also matches to (15.38).

So, whenever we consider the situation of a joint source and channel coding scheme, we define the implicit rate as

$$R \triangleq \frac{T_c}{T_s} H(\{U_k\}). \tag{15.40}$$

[2]Strictly speaking, we should not write R as in (15.39) because there we do not see the correct units of *bits*. The correct form actually is $R = \frac{K}{n} \log D$, which yields (15.39) if $D = 2$ and the logarithm is to the base 2.

15.6 Transmission above Capacity and Minimum Bit Error Rate

We have understood that it is possible to transmit a source over a DMC with arbitrarily small error probability only if (15.34) is satisfied. The natural next question to ask therefore is whether we can say something about the minimal error probability if we try to transmit a source with an entropy rate above capacity.

To simplify this investigation we will assume in this section that $\{U_k\}$ is an IID uniform binary source,[3] i.e., $\mathrm{H}(\{U_k\}) = 1$ bit. Moreover, we assume that the parameters of the system and the DMC are such that (15.35) holds, i.e.,

$$\frac{1}{\mathsf{T}_\mathrm{s}} \text{ bits/s} > \frac{\mathsf{C}}{\mathsf{T}_\mathrm{c}} \text{ bits/s}. \tag{15.41}$$

We know that the probability of a message (or block) error

$$P_\mathrm{e}^{(\mathsf{K})} \triangleq \Pr\Big[U_1^{\mathsf{K}} \neq \hat{U}_1^{\mathsf{K}}\Big] \tag{15.42}$$

is strictly bounded away from zero (compare again with Figure 15.1). Actually, it can be shown that $P_\mathrm{e}^{(\mathsf{K})}$ tends to 1 exponentially fast in K. However, this is not really an interesting statement. Since already a single bit error will cause the total length-K message to be wrong, we realize that some bits still could be transmitted correctly in spite of a wrong message $\hat{U}_1^{\mathsf{K}}$. Hence, we are much more interested in the *average bit error probability* or, as it is usually called, the *average bit error rate (BER)*

$$P_\mathrm{b} \triangleq \frac{1}{\mathsf{K}} \sum_{k=1}^{\mathsf{K}} P_{\mathrm{b},k} \tag{15.43}$$

where

$$P_{\mathrm{b},k} \triangleq \Pr\big[U_k \neq \hat{U}_k\big]. \tag{15.44}$$

Unfortunately, as we will show below, the converse also holds when replacing $P_\mathrm{e}^{(\mathsf{K})}$ by P_b. I.e., if (15.41) holds, then also the BER is strictly bounded away from zero, even if we let $\mathsf{K} \to \infty$. Can we say more about how far away?

A typical engineering way of trying to deal with the problem that reliable transmission is impossible is to try to take control by adding the errors purposefully ourselves! So we add an additional building block between source

[3] Again, this is not a big restriction as we can transform any source into such a memoryless uniform binary source by applying an optimal compression scheme to it.

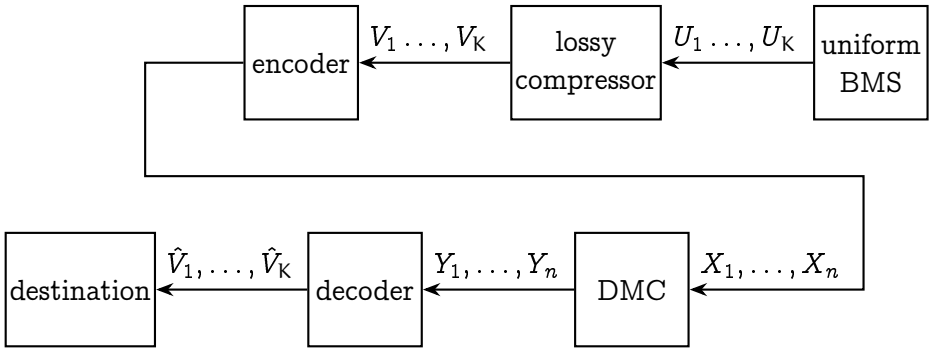


Figure 15.3: Lossy compression added to joint source channel coding: We insert a lossy mapping $g(\cdot)$ between source and channel encoder to control the introduced errors.

and encoder (see Figure 15.3) that compresses the source sequence U_1^K in a *lossy* way such that

$$\frac{1}{T_s}\frac{H(V_1^K)}{K} \text{ bits/s} < \frac{C}{T_c} \text{ bits/s}. \tag{15.45}$$

Then we know that V_1^K can be transmitted reliably over the channel and the BER is under our full control by choosing the compressor mapping $g(\cdot)$:

$$V_1^K = g(U_1^K). \tag{15.46}$$

Example 15.5. As an example consider a compressor scheme $g(\cdot)$ as follows: for K even

$$\begin{aligned}(v_1, v_2, \ldots, v_{K-1}, v_K) &= g(u_1, u_2, \ldots, u_{K-1}, u_K) && (15.47)\\ &\triangleq (u_1, u_1, u_3, u_3, \ldots, u_{K-1}, u_{K-1}), && (15.48)\end{aligned}$$

i.e., every second information bit is canceled. We then obviously have

$$\frac{1}{K} H(V_1^K) = \frac{1}{2} \text{ bits} \tag{15.49}$$

and, assuming that

$$\frac{1}{2T_s} \text{ bits/s} < \frac{C}{T_c} \text{ bits/s}, \tag{15.50}$$

we can reliably transmit V_1^K, i.e., for any $\epsilon > 0$ we have

$$\Pr[\hat{V}_k \neq V_k] \leq \epsilon \tag{15.51}$$

for K large enough. To simplify our notation we therefore now write[4] $\hat{V}_k = V_k$ and get

$$P_{b,k} = \Pr[\hat{V}_k \neq U_k] = \Pr[V_k \neq U_k] = \begin{cases} 0 & \text{for } k \text{ odd},\\ \frac{1}{2} & \text{for } k \text{ even} \end{cases} \tag{15.52}$$

[4]Strictly speaking we should say that $\Pr[\hat{V}_k = V_k] > 1 - \epsilon$.

and hence

$$P_{\mathrm{b}} = \frac{1}{2} \cdot P_{\mathrm{b,odd}} + \frac{1}{2} \cdot P_{\mathrm{b,even}} = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}. \tag{15.53}$$

So we see that we have managed to design a system with average BER $P_{\mathrm{b}} = \frac{1}{4}$ subject to $\frac{1}{2\mathsf{T}_{\mathrm{s}}}$ bits/s being smaller than the channel capacity $\frac{\mathrm{C}}{\mathsf{T}_{\mathrm{c}}}$. ◊

Example 15.6. Another, more elegant compressor design is based on the $(7,4)$ *Hamming code*. This code is a length-7 binary channel code with the following 16 codewords:

$$\begin{aligned}\mathscr{C}_{\mathrm{H}} = \{&0000000,\ 1010001,\ 1110010,\ 0100011,\\ &0110100,\ 1100101,\ 1000110,\ 0010111,\\ &1101000,\ 0111001,\ 0011010,\ 1001011,\\ &1011100,\ 0001101,\ 0101110,\ 1111111\}.\end{aligned} \tag{15.54}$$

It can be shown that this code provides a *perfect packing* of the 7-dimensional binary space: If we define a "sphere" around every codeword consisting of the codeword itself plus all length-7 binary sequences with exactly one position being different from the codeword, then firstly all these spheres will be disjoint and secondly all spheres together will contain all possible length-7 binary sequences. Note that each sphere contains eight length-7 sequences.

Hence, we can now design a compressor using "inverse Hamming coding": $g(u_1^7) = v_1^7$ if v_1^7 is a $(7,4)$ Hamming codeword and u_1^7 has at most one component different from v_1^7. In other words, every sequence u_1^7 is mapped to the sphere center of its corresponding sphere. Due to the perfect packing property, we know that every possible sequence will belong to exactly one sphere so that this mapping is well-defined.

Since the source produces all $2^7 = 128$ binary sequences of length 7 with equal probability and since always eight sequences are mapped to the same codeword, we see that the 16 codewords will show up with equal probability. Hence,

$$\frac{1}{\mathsf{K}}\,\mathrm{H}(V_1^{\mathsf{K}}) = \frac{\log_2 16}{7} = \frac{4}{7} \text{ bits}, \tag{15.55}$$

i.e., slightly larger than the rate $1/2$ bits of Example 15.5. On the other hand, we see that

$$P_{\mathrm{b},k} = \Pr[U_k \neq V_k] = \frac{1}{8} \tag{15.56}$$

because only one of the 8 sequences in each sphere will differ in the kth position from the correct sequence, and hence

$$P_{\mathrm{b}} = \frac{1}{8}. \tag{15.57}$$

This is only half the value of the compressor design given in Example 15.5! ◊

So, we come back to our original question: What is the optimal compressor design, i.e., the design that minimizes the BER subject to the average output entropy being less than the channel capacity?

This question is similarly hard to answer as to find the optimum design of a channel code. Shannon, however, again found a way to answer part of this question: He managed to identify the minimum BER that is possible without specifying how it can be achieved!

In the following we will show his proof. It is again based on the system design of Figure 15.3, i.e., we split the encoder into two parts: a lossy compressor and a standard channel encoder.

So we assume that (15.41) holds, but that a compressor is given such that

$$\frac{1}{T_s}\frac{1}{K}\,\mathrm{H}(V_1^K) < \frac{C}{T_c}. \tag{15.58}$$

We then have:

$$\begin{aligned}
\frac{1}{K}\,\mathrm{H}(V_1^K) &= \frac{1}{K}\,\mathrm{H}(V_1^K) - \frac{1}{K}\underbrace{\mathrm{H}(V_1^K|U_1^K)}_{=0} && (15.59)\\
&= \frac{1}{K}\,\mathrm{I}(V_1^K;U_1^K) && (15.60)\\
&= \frac{1}{K}\,\mathrm{H}(U_1^K) - \frac{1}{K}\,\mathrm{H}(U_1^K|V_1^K) && (15.61)\\
&= 1 - \frac{1}{K}\,\mathrm{H}(U_1^K|V_1^K) && (15.62)\\
&= 1 - \frac{1}{K}\sum_{k=1}^{K}\mathrm{H}(U_k|U_1^{k-1},V_1^K) && (15.63)\\
&\geq 1 - \frac{1}{K}\sum_{k=1}^{K}\mathrm{H}(U_k|V_k)\text{ bits}, && (15.64)
\end{aligned}$$

where the first equality follows because $V_1^K = g(U_1^K)$; (15.62) holds because $\{U_k\}$ is IID uniform binary; the subsequent equality follows from the chain rule; and in the final step we use that conditioning cannot increase entropy.

Note that

$$\begin{aligned}
\mathrm{H}(U_k|V_k) &= \Pr[V_k=0]\,\mathrm{H}(U_k|V_k=0) + \Pr[V_k=1]\;\mathrm{H}(U_k|V_k=1) && (15.65)\\
&= \Pr[V_k=0]\,\mathrm{H_b}\big(\Pr[U_k=1\,|\,V_k=0]\big)\\
&\quad + \Pr[V_k=1]\,\mathrm{H_b}\big(\Pr[U_k=0\,|\,V_k=1]\big) && (15.66)\\
&\leq \mathrm{H_b}\Big(\Pr[V_k=0]\,\Pr[U_k=1\,|\,V_k=0]\\
&\qquad + \Pr[V_k=1]\,\Pr[U_k=0\,|\,V_k=1]\Big) && (15.67)\\
&= \mathrm{H_b}\big(\Pr[U_k=1,V_k=0] + \Pr[U_k=0,V_k=1]\big) && (15.68)\\
&= \mathrm{H_b}\big(\Pr[U_k\neq V_k]\big) && (15.69)\\
&= \mathrm{H_b}(P_{\mathrm{b},k}). && (15.70)
\end{aligned}$$

Here, the first equality (15.65) follows by definition of conditional entropy; the subsequent because U_k is binary; and the inequality (15.67) follows from the concavity of $\mathrm{H_b}(\cdot)$:

$$\theta\,\mathrm{H_b}(\xi_1) + (1-\theta)\,\mathrm{H_b}(\xi_2) \le \mathrm{H_b}\big(\theta\xi_1 + (1-\theta)\xi_2\big). \tag{15.71}$$

Plugging (15.70) into (15.64) and using concavity once more, we get

$$\begin{aligned}\frac{1}{\mathsf{K}}\,\mathrm{H}(V_1^{\mathsf{K}}) &\ge 1 - \frac{1}{\mathsf{K}}\sum_{k=1}^{\mathsf{K}} \mathrm{H_b}(P_{\mathrm{b},k}) && (15.72)\\ &\ge 1 - \mathrm{H_b}\left(\frac{1}{\mathsf{K}}\sum_{k=1}^{\mathsf{K}} P_{\mathrm{b},k}\right) && (15.73)\\ &= 1 - \mathrm{H_b}(P_{\mathrm{b}}) \text{ bits}, && (15.74)\end{aligned}$$

where the last equality follows from (15.43).

Hence, from our assumption (15.58) we therefore see that

$$\frac{1}{\mathsf{T_s}}\big(1 - \mathrm{H_b}(P_{\mathrm{b}})\big) \le \frac{1}{\mathsf{T_s}}\frac{1}{\mathsf{K}}\,\mathrm{H}(V_1^{\mathsf{K}}) < \frac{\mathsf{C}}{\mathsf{T_c}}, \tag{15.75}$$

which, using the definition (15.40) and the assumption that $\mathrm{H}(\{U_k\}) = 1$ bit, transforms into

$$\mathrm{H_b}(P_{\mathrm{b}}) > 1 - \frac{\mathsf{C}}{\mathsf{R}}. \tag{15.76}$$

In order to include the cases when $\mathsf{R} \le \mathsf{C}$, we weaken the inequality to

$$\mathrm{H_b}(P_{\mathrm{b}}) \ge 1 - \frac{\mathsf{C}}{\mathsf{R}} \tag{15.77}$$

(which holds trivially when $\mathsf{R} \le \mathsf{C}$). Thus

$$P_{\mathrm{b}} \ge \mathrm{H_b^{-1}}\left(1 - \frac{\mathsf{C}}{\mathsf{R}}\right). \tag{15.78}$$

Here $\mathrm{H_b^{-1}}(\cdot)$ is the inverse function of the binary entropy function $\mathrm{H_b}(\xi)$ for $\xi \in [0, 1/2]$ (and we define $\mathrm{H_b^{-1}}(\zeta) \triangleq 0$ for $\zeta < 0$).

We see that if $\mathsf{R} > \mathsf{C}$, then the average bit error rate is bounded away from zero.

There are two questions that remain:

1. Can we find a lossy compressor that actually achieves the lower bound (15.78)?
2. Is it possible that another system design that is not based on the idea of Figure 15.3 (but on the more general system given in Figure 15.1) could achieve a bit error rate that is smaller than (15.78)?

The answer to the first question is yes, however, we will not prove it in this class.[5] The proof is based on the concept of *rate distortion theory*, another of the big inventions of Shannon [Sha48]. He asked the following question: Given a certain source and given a certain acceptable distortion, how much can we compress the source without causing more than the given acceptable distortion? Actually, we can turn the question also around: Given a certain source and given desired rate, what is the smallest possible distortion? If we now choose as a measure of distortion the bit error probability, this question is exactly our problem described here. It turns out that the smallest achievable distortion looks exactly like the right-hand side of (15.78).

The second question can be answered negatively. The proof is straightforward and is based on the Fano Inequality. Consider again the general system of Figure 15.1. For such a system we have the following:

$$\begin{aligned}
&\frac{1}{K}\,\mathrm{H}(U_1,\ldots,U_K|\hat{U}_1,\ldots,\hat{U}_K)\\
&\quad=\frac{1}{K}\sum_{k=1}^{K}\mathrm{H}(U_k|U_1,\ldots,U_{k-1},\hat{U}_1,\ldots,\hat{U}_K) && (15.79)\\
&\quad\le\frac{1}{K}\sum_{k=1}^{K}\mathrm{H}(U_k|\hat{U}_k) && (15.80)\\
&\quad\le\frac{1}{K}\sum_{k=1}^{K}\mathrm{H_b}(P_{\mathrm{b},k})+\frac{1}{K}\sum_{k=1}^{K}P_{\mathrm{b},k}\log(|\mathcal{U}|-1) && (15.81)\\
&\quad\le\mathrm{H_b}\left(\frac{1}{K}\sum_{k=1}^{K}P_{\mathrm{b},k}\right)+\log(|\mathcal{U}|-1)\frac{1}{K}\sum_{k=1}^{K}P_{\mathrm{b},k} && (15.82)\\
&\quad=\mathrm{H_b}(P_{\mathrm{b}})+P_{\mathrm{b}}\log(|\mathcal{U}|-1), && (15.83)
\end{aligned}$$

where (15.79) follows from the chain rule; (15.80) follows because conditioning reduces entropy; (15.81) follows by the Fano Inequality (17.105); (15.82) follows from concavity; and the last equality (15.83) from the definition of the bit error rate (15.43). Actually, note that we have just derived the Fano Inequality for the situation of bit errors:

$$\frac{1}{K}\,\mathrm{H}(U_1,\ldots,U_K|\hat{U}_1,\ldots,\hat{U}_K)\le\mathrm{H_b}(P_{\mathrm{b}})+P_{\mathrm{b}}\log(|\mathcal{U}|-1). \tag{15.84}$$

We can now re-derive the converse to the Information Transmission Theorem as follows (compare with (15.6)–(15.12)):

$$\begin{aligned}
\frac{\mathrm{H}(\{U_k\})}{T_{\mathrm{s}}}&=\frac{1}{T_{\mathrm{s}}}\lim_{K\to\infty}\frac{\mathrm{H}(U_1^K)}{K} && (15.85)\\
&\le\frac{1}{KT_{\mathrm{s}}}\,\mathrm{H}(U_1^K) && (15.86)
\end{aligned}$$

[5] A proof is given in the advanced course [Mos22, Section 11.7].

$$= \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{H}(U_1^{\mathsf{K}}|\hat{U}_1^{\mathsf{K}}) + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{I}(U_1^{\mathsf{K}}; \hat{U}_1^{\mathsf{K}}) \tag{15.87}$$

$$\le \frac{\mathrm{H}_{\mathrm{b}}(P_{\mathrm{b}})}{\mathsf{T}_{\mathrm{s}}} + \frac{P_{\mathrm{b}} \log(|\mathcal{U}| - 1)}{\mathsf{T}_{\mathrm{s}}} + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{I}(U_1^{\mathsf{K}}; \hat{U}_1^{\mathsf{K}}) \tag{15.88}$$

$$\le \frac{\mathrm{H}_{\mathrm{b}}(P_{\mathrm{b}})}{\mathsf{T}_{\mathrm{s}}} + \frac{P_{\mathrm{b}} \log(|\mathcal{U}| - 1)}{\mathsf{T}_{\mathrm{s}}} + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} \mathrm{I}(X_1^n; Y_1^n) \tag{15.89}$$

$$\le \frac{\mathrm{H}_{\mathrm{b}}(P_{\mathrm{b}})}{\mathsf{T}_{\mathrm{s}}} + \frac{P_{\mathrm{b}} \log(|\mathcal{U}| - 1)}{\mathsf{T}_{\mathrm{s}}} + \frac{1}{\mathsf{K}\mathsf{T}_{\mathrm{s}}} n\mathsf{C} \tag{15.90}$$

$$= \frac{\mathrm{H}_{\mathrm{b}}(P_{\mathrm{b}})}{\mathsf{T}_{\mathrm{s}}} + \frac{P_{\mathrm{b}} \log(|\mathcal{U}| - 1)}{\mathsf{T}_{\mathrm{s}}} + \frac{\mathsf{C}}{\mathsf{T}_{\mathrm{c}}}, \tag{15.91}$$

where (15.85) follows by the definition of the entropy rate; (15.86) because $\mathrm{H}(U_1^{\mathsf{K}})/\mathsf{K}$ is decreasing in K; (15.87) by definition of mutual information; the subsequent two inequalities (15.88) and (15.89) by Fano (15.83) and the Data Processing Inequality (Lemma 11.30); (15.90) by (17.15) and the memorylessness of the DMC; and the final equality holds because $\mathsf{K}\mathsf{T}_{\mathrm{s}} = n\mathsf{T}_{\mathrm{c}}$.

If we now again assume that the source is IID uniform binary, we have $\mathrm{H}(\{U_k\}) = 1$ bit and $|\mathcal{U}| = 2$, and therefore we get from (15.91):

$$\frac{1}{\mathsf{T}_{\mathrm{s}}} - \frac{\mathrm{H}_{\mathrm{b}}(P_{\mathrm{b}})}{\mathsf{T}_{\mathrm{s}}} \le \frac{\mathsf{C}}{\mathsf{T}_{\mathrm{c}}}. \tag{15.92}$$

Hence, we see that (15.77) — and therefore also (15.78) — is satisfied for any system.

We have once again seen Shannon's incredible capability of finding fundamental insights by sacrificing some answers (in particular the question of how to design a system). We will come back to (15.78) in Chapter 17.

Chapter 16

Continuous Random Variables and Differential Entropy

So far we have restricted ourselves to discrete alphabets, i.e., all random variables did only take a finite or at most countably infinite number of values. In reality, however, there are also often cases where the alphabets are uncountably infinite. We will next introduce the nice special case of *continuous* RVs where we can describe the probability distribution with a *probability density function (PDF)* (see also the discussion in Section 2.3).

16.1 Entropy of Continuous Random Variables

Let X be a continuous random variable with continuous alphabet $\mathcal{X}$ and with probability density function (PDF) $f_X(\cdot)$, i.e.,

$$\Pr[X \le x] = \int_{-\infty}^{x} f_X(t)\,\mathrm{d}t. \tag{16.1}$$

What is its entropy $\mathrm{H}(X)$? To see this, we need to find the probability mass function (PMF) $P_X(\cdot)$ for the continuous random variable X, or at least — since the PMF is not defined for a continuous random variable — an approximate value of the PMF. Fix some positive value $\Delta \ll 1$. Then for every $x \in \mathcal{X}$,

$$P_X(x) \approx \int_{x-\frac{\Delta}{2}}^{x+\frac{\Delta}{2}} f_X(t)\,\mathrm{d}t \approx f_X(x) \int_{x-\frac{\Delta}{2}}^{x+\frac{\Delta}{2}} \mathrm{d}t = \Delta \cdot f_X(x). \tag{16.2}$$

This approximation will become better if we let Δ become smaller. Using this approximation in the definition of entropy:

$$\begin{aligned} \mathrm{H}(X) &\triangleq -\sum_x P_X(x) \log P_X(x) && (16.3)\\ &\approx -\sum_x \Delta \cdot f_X(x) \log(\Delta \cdot f_X(x)) && (16.4) \end{aligned}$$

$$= -\sum_x \Delta \cdot f_X(x) \log \Delta - \sum_x \Delta \cdot f_X(x) \log f_X(x) \tag{16.5}$$

$$= -\log \Delta \sum_x f_X(x) \cdot \Delta - \sum_x f_X(x) \log f_X(x) \cdot \Delta \tag{16.6}$$

$$\overset{\Delta \downarrow 0}{\to} \lim_{\Delta \downarrow 0}\{-\log \Delta\} \underbrace{\int_{\mathcal{X}} f_X(x)\,\mathrm{d}x}_{=1} - \int_{\mathcal{X}} f_X(x) \log f_X(x)\,\mathrm{d}x \tag{16.7}$$

$$= \underbrace{\lim_{\Delta \downarrow 0}\{-\log \Delta\}}_{=\infty} - \int_{\mathcal{X}} f_X(x) \log f_X(x)\,\mathrm{d}x, \tag{16.8}$$

we realize that

$$\mathrm{H}(X) = \infty! \tag{16.9}$$

Actually, this is quite obvious if you think about it: X can take on infinitely many different values!

So, the original definition of uncertainty or entropy does not make sense for continuous random variables. How can we save ourselves from this rather unfortunate situation? Well, we simply define a new quantity...

Definition 16.1. Let $X \in \mathcal{X}$ be a continuous random variable with PDF $f_X(\cdot)$. Then the *differential entropy* $\mathrm{h}(X)$ is defined as follows:

$$\mathrm{h}(X) \triangleq -\int_{\mathcal{X}} f_X(x) \log f_X(x)\,\mathrm{d}x = \mathrm{E}[-\log f_X(X)]. \tag{16.10}$$

Note that as for the normal entropy, the basis of the logarithm defines the *unit* of differential entropy. In particular, the choice of the binary logarithm gives a differential entropy in *bits* and the natural logarithm gives a differential entropy in *nats*.

From (16.8) we know that

$$\mathrm{h}(X) = \mathrm{H}(X) - \lim_{\Delta \downarrow 0}\{-\log \Delta\}, \tag{16.11}$$

i.e., $\mathrm{h}(X)$ is a "shift" of $\mathrm{H}(X)$ "by negative infinity". Unfortunately, we will see that

$\mathrm{h}(X)$ can be negative!

A negative uncertainty?!? Very embarrassing...

Example 16.2. Let X be uniformly distributed on the interval $[0,a]$ for some value $a>0$, $X\sim\mathcal{U}([0,a])$:

$$f_X(x)=\begin{cases}\frac{1}{a} & \text{for } x\in[0,a],\\ 0 & \text{otherwise.}\end{cases} \tag{16.12}$$

We now compute the differential entropy for X:

$$\mathrm{h}(X)=-\int_0^a \frac{1}{a}\log\frac{1}{a}\,\mathrm{d}x=\frac{1}{a}\log a\int_0^a 1\,\mathrm{d}x=\log a. \tag{16.13}$$

Hence,

$$\text{if } a>1 \implies \mathrm{h}(X)>0; \tag{16.14a}$$
$$\text{if } a=1 \implies \mathrm{h}(X)=0; \tag{16.14b}$$
$$\text{if } 0<a<1 \implies \mathrm{h}(X)<0. \tag{16.14c}$$

Note that if we let $a\to 0$, we get $\mathrm{h}(X)\to-\infty$. This can be understood: if $a=0$, then X only takes on *one* value with probability 1. It is thus not a continuous RV, but discrete. Its normal uncertainty is zero ($\mathrm{H}(X)=0$), which according to (16.11) corresponds to a negative infinite value of the differential entropy. So we see that $\mathrm{h}(X)=-\infty$ basically means that X is not "random enough". Note that, as seen in (16.11), the differential entropy will be negative infinite for any *finite* value of the corresponding normal entropy. ◊

Remark 16.3. Even if X is continuous but contains some discrete random variable parts, i.e., if X is a mixture between continuous and discrete random variable, then $\mathrm{h}(X)=-\infty$. This can be seen directly from the PDF $f_X(\cdot)$, which will contain some Dirac deltas in such a situation; see the following example. △

Example 16.4. As an example, consider an random variable Y with cumulative distribution function (CDF)

$$F_Y(y)\triangleq\Pr[Y\le y]=\begin{cases}0 & y<0,\\ \frac{y+1}{4} & 0\le y<2,\\ 1 & y\ge 2\end{cases} \tag{16.15}$$

(see Figure 16.1). Note that Y is not continuous, because it has two values with positive probability:

$$\Pr[Y=a]=\begin{cases}\frac{1}{4} & \text{for } a=0 \text{ or } a=2,\\ 0 & \text{otherwise.}\end{cases} \tag{16.16}$$

But between 0 and 2, Y is uniformly distributed with a constant density. Strictly speaking, the PDF is not defined, but we engineers sometimes like to

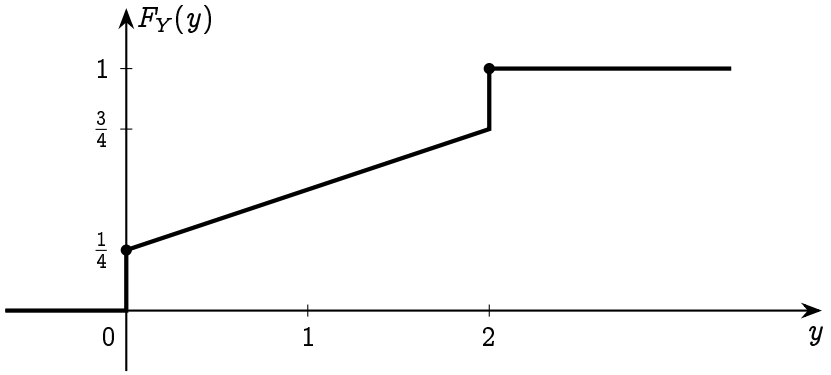


Figure 16.1: CDF of a random variable that is a mixture of discrete and continuous.

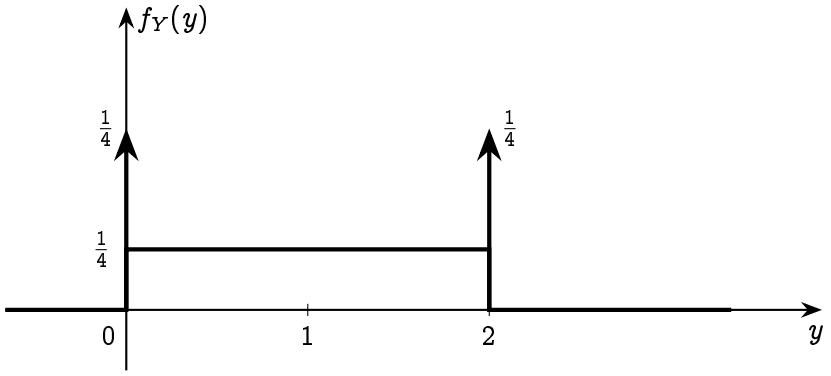


Figure 16.2: PDF of the random variable of Figure 16.1: it is a mixture between two Dirac deltas and a uniform distribution between 0 and 2.

use Dirac deltas:

$$\delta_{\text{Dirac}}(t) \triangleq \begin{cases} 0 & \text{for all } t \neq 0, \\ \text{“}\infty\text{''} & \text{for } t = 0, \end{cases} \tag{16.17}$$

where this strange value “∞” has to be thought of being such that if integrated over, it evaluates to 1:

$$\int_{-\infty}^{\infty} \delta_{\text{Dirac}}(t)\,\mathrm{d}t \triangleq 1. \tag{16.18}$$

Hence, we can write the PDF of Y as follows:

$$f_Y(y) = \frac{1}{4}\,\mathbb{1}\{y \in (0,2)\} + \frac{1}{4}\,\delta_{\text{Dirac}}(y) + \frac{1}{4}\,\delta_{\text{Dirac}}(y-2) \tag{16.19}$$

(see Figure 16.2). Note that this PDF indeed integrates to 1:

$$\begin{aligned}
&\int_{-\infty}^{\infty} f_Y(y)\,\mathrm{d}y \\
&\quad= \int_{-\infty}^{\infty} \left(\frac{1}{4}\,\mathbb{1}\{y\in(0,2)\} + \frac{1}{4}\,\delta_{\text{Dirac}}(y) + \frac{1}{4}\,\delta_{\text{Dirac}}(y-2)\right)\mathrm{d}y && (16.20)\\
&\quad= \int_0^2 \frac{1}{4}\,\mathrm{d}y + \frac{1}{4}\underbrace{\int_{-\infty}^{\infty} \delta_{\text{Dirac}}(y)\,\mathrm{d}y}_{=1} + \frac{1}{4}\underbrace{\int_{-\infty}^{\infty} \delta_{\text{Dirac}}(y-2)\,\mathrm{d}y}_{=1} && (16.21)\\
&\quad= \frac{2}{4}+\frac{1}{4}+\frac{1}{4} = 1. && (16.22)
\end{aligned}$$

If we now daringly apply the definition of differential entropy to this (weird) PDF, we get

$$\begin{aligned}
\mathrm{h}(Y) &= -\int_{-\infty}^{\infty} f_Y(y)\log f_Y(y)\,\mathrm{d}y && (16.23)\\
&= \mathsf{E}[-\log f_Y(Y)] && (16.24)\\
&= \mathsf{E}\left[-\log\left(\frac{1}{4}\,\mathbb{1}\{Y\in(0,2)\} + \frac{1}{4}\,\delta_{\text{Dirac}}(Y) + \frac{1}{4}\,\delta_{\text{Dirac}}(Y-2)\right)\right], && (16.25)
\end{aligned}$$

i.e., we see that we have a Dirac delta inside of the logarithm. Again, we have to be careful of what the meaning of such an expression is, but considering the fact, that $\delta_{\text{Dirac}}(0)=\infty$, it does not really surprise that we get $\mathsf{E}[\log\delta_{\text{Dirac}}(Y)]=\infty$ and therefore $\mathrm{h}(Y)=-\infty$. ◊

Example 16.5. Let X be zero-mean Gaussian distributed with variance σ^2, $X\sim\mathcal{N}(0,\sigma^2)$. Then

$$\begin{aligned}
\mathrm{h}(X) &= -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\,e^{-\frac{x^2}{2\sigma^2}}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\,e^{-\frac{x^2}{2\sigma^2}}\right)\mathrm{d}x && (16.26)\\
&= -\,\mathsf{E}\left[\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\,e^{-\frac{X^2}{2\sigma^2}}\right)\right] && (16.27)\\
&= \frac{1}{2}\log 2\pi\sigma^2 + \frac{\mathsf{E}[X^2]}{2\sigma^2}\log e && (16.28)\\
&= \frac{1}{2}\log 2\pi\sigma^2 + \frac{\sigma^2}{2\sigma^2}\log e && (16.29)\\
&= \frac{1}{2}\log 2\pi e\sigma^2. && (16.30)
\end{aligned}$$

Here we have demonstrated that often the calculations become much simpler when relying on the expectation-form of the definition of differential entropy. ◊

16.2 Properties of Differential Entropy

So how does $\mathrm{h}(X)$ behave? We have already seen that $\mathrm{h}(X)$ can be negative, but that a negative value does not imply anything special apart from the fact that $\mathrm{h}(X)<0$ means X has less uncertainty than if $\mathrm{h}(X)$ were positive. Now we will explore and find some more (strange) properties.

We start with scaling.

Exercise 16.6. *Let X_{d} be a discrete random variable with entropy $\mathrm{H}(X_{\mathrm{d}})$. What is the entropy of $2X_{\mathrm{d}}$?*

$$\mathrm{H}(2X_{\mathrm{d}}) = \_\_\_\_\_ \tag{16.31}$$

Why?[1] ◊

So what about the relation between $\mathrm{h}(X)$ and $\mathrm{h}(2X)$ for a continuous random variable X? To compute this, we define a new random variable $Y \triangleq aX$ for some $a > 0$. We know that the PDF of Y is

$$f_Y(y) = \frac{1}{a} f_X\left(\frac{y}{a}\right). \tag{16.32}$$

Hence,

$$\begin{aligned}
\mathrm{h}(aX) &= \mathrm{h}(Y) && (16.33)\\
&= -\int_{\mathcal{Y}} f_Y(y) \log f_Y(y)\,\mathrm{d}y && (16.34)\\
&= -\int_{\mathcal{Y}} \frac{1}{a} f_X\left(\frac{y}{a}\right) \log\left(\frac{1}{a} f_X\left(\frac{y}{a}\right)\right) \mathrm{d}y && (16.35)\\
&= \log a \cdot \int_{\mathcal{Y}} \frac{1}{a} f_X\left(\frac{y}{a}\right) \mathrm{d}y - \int_{\mathcal{Y}} \frac{1}{a} f_X\left(\frac{y}{a}\right) \log f_X\left(\frac{y}{a}\right) \mathrm{d}y && (16.36)\\
&= \log a \cdot \underbrace{\int_{\mathcal{X}} f_X(x)\,\mathrm{d}x}_{=1} - \int_{\mathcal{X}} f_X(x) \log f_X(x)\,\mathrm{d}x && (16.37)\\
&= \log a + \mathrm{h}(X), && (16.38)
\end{aligned}$$

where in the second last step we have changed integration variables and defined $x \triangleq \frac{y}{a}$. Again, quite embarrassing: The uncertainty changes depending on the scaling of the random variable!

Lemma 16.7. Let X be a continuous random variable with differential entropy $\mathrm{h}(X)$. Let $a \in \mathbb{R} \setminus \{0\}$ be a real nonzero constant. Then

$$\mathrm{h}(aX) = \mathrm{h}(X) + \log|a|. \tag{16.39}$$

Let $c \in \mathbb{R}$ be a real constant. Then

$$\mathrm{h}(X + c) = \mathrm{h}(X). \tag{16.40}$$

Proof: For $a > 0$, we have proven (16.39) above. The derivation for $a < 0$ is analogous. The proof of (16.40) is left to the reader as an exercise. □

[1] The solution can be found at the end of this chapter.

Remark 16.8. Many people do not like differential entropy because of its weird behavior. They claim that it does not make sense and is against all "engineering feeling". However, the differential entropy has its right of existence. So, e.g., it can very precisely classify different types of fading channels depending on the uncertainty in the fading process [Mos05]. △

16.3 Generalizations and Further Definitions

Similarly to normal entropy, we can define a joint and a conditional differential entropy.

Definition 16.9. The *joint differential entropy* of several continuous random variables $X_1, \dots, X_n$ with joint PDF $f_{X_1,\dots,X_n}$ is defined as

$$\begin{aligned}
&\mathrm{h}(X_1, \dots, X_n) \\
&\triangleq -\int_{\mathcal{X}_n} \cdots \int_{\mathcal{X}_1} f_{X_1,\dots,X_n}(x_1, \dots, x_n) \log f_{X_1,\dots,X_n}(x_1, \dots, x_n)\, \mathrm{d}x_1 \cdots \mathrm{d}x_n && (16.41)\\
&= \mathsf{E}[-\log f_{X_1,\dots,X_n}(X_1, \dots, X_n)]. && (16.42)
\end{aligned}$$

Definition 16.10. Let X and Y be two continuous random variables with joint PDF $f_{X,Y}(\cdot,\cdot)$. Then the *conditional differential entropy* of X conditional on Y is defined as follows:

$$\begin{aligned}
\mathrm{h}(X|Y) &\triangleq -\int_{\mathcal{Y}} \int_{\mathcal{X}} f_{X,Y}(x,y) \log f_{X|Y}(x|y)\, \mathrm{d}x\, \mathrm{d}y && (16.43)\\
&= \mathsf{E}\Big[-\log f_{X|Y}(X|Y)\Big]. && (16.44)
\end{aligned}$$

From this immediately follows that similarly to the conditional entropy, it is also holds for the conditional differential entropy that

$$\begin{aligned}
\mathrm{h}(X|Y) &= \mathsf{E}_Y[\mathrm{h}(X|Y=y)] && (16.45)\\
&= \int_{\mathcal{Y}} f_Y(y)\, \mathrm{h}(X|Y=y)\, \mathrm{d}y. && (16.46)
\end{aligned}$$

Besides its many strange properties, differential entropy also shows good behavior: Since the chain rule for PDFs work the same way as for PMFs, i.e.,

$$f_{X,Y}(x,y) = f_Y(y) \cdot f_{X|Y}(x|y), \tag{16.47}$$

the chain rule also holds for differential entropy:

$$\mathrm{h}(X,Y) = \mathrm{h}(Y) + \mathrm{h}(X|Y). \tag{16.48}$$

Next we turn to mutual information. Do we have a similar embarrassing situation as in the case of entropy vs. differential entropy? Luckily, the answer here is no. You can realize this as follows:

$$\begin{aligned} \mathrm{I}(X;Y) &= \mathrm{H}(Y) - \mathrm{H}(Y|X) && (16.49)\\ &\approx \mathrm{h}(Y) - \log\Delta - \mathrm{h}(Y|X) + \log\Delta && (16.50)\\ &= \mathrm{h}(Y) - \mathrm{h}(Y|X), && (16.51) \end{aligned}$$

i.e., the strange "shift by negative infinity" disappears because mutual information is a *difference* of entropies. However, note that since mathematically it is not possible to define a quantity as a difference of two infinite values, we use the alternative form (1.121) of mutual information as the basis of our definition.

Definition 16.11. The *mutual information* between two continuous random variables X and Y with joint PDF $f_{X,Y}(\cdot,\cdot)$ is defined as follows:

$$\mathrm{I}(X;Y) \triangleq \int_{\mathcal{Y}}\int_{\mathcal{X}} f_{X,Y}(x,y)\log\frac{f_{X,Y}(x,y)}{f_X(x)\,f_Y(y)}\,\mathrm{d}x\,\mathrm{d}y. \tag{16.52}$$

From this form, it is straightforward to derive the following natural alternative forms:

$$\begin{aligned} \mathrm{I}(X;Y) &= \int_{\mathcal{Y}}\int_{\mathcal{X}} f_{X,Y}(x,y)\log\frac{f_{Y|X}(y|x)}{f_Y(y)}\,\mathrm{d}x\,\mathrm{d}y && (16.53)\\ &= \mathrm{h}(Y) - \mathrm{h}(Y|X) && (16.54) \end{aligned}$$

and similarly

$$\mathrm{I}(X;Y) = \mathrm{h}(X) - \mathrm{h}(X|Y). \tag{16.55}$$

Accordingly, we extend the definition of relative entropy (Definition 3.1) to continuous RVs.

Definition 16.12. Let $f(\cdot)$ and $g(\cdot)$ be two PDFs over the same continuous alphabet $\mathcal{X}$. The *relative entropy* between $f(\cdot)$ and $g(\cdot)$ is defined as

$$\mathscr{D}(f\,\|\,g) \triangleq \int_{\mathcal{X}} f(x)\log\frac{f(x)}{g(x)}\,\mathrm{d}x. \tag{16.56}$$

Hence, as before, we can write

$$\mathrm{I}(X;Y) = \mathscr{D}(f_{X,Y}\,\|\,f_X\cdot f_Y). \tag{16.57}$$

We have more good news: Mutual information and relative entropy behave exactly as expected also for continuous random variables! I.e.,

$$\mathrm{I}(X;Y) \geq 0 \tag{16.58}$$

with equality if, and only if, $X \perp\!\!\!\perp Y$, and

$$\mathscr{D}(f \,\|\, g) \geq 0 \tag{16.59}$$

with equality[2] if, and only if, $f(\cdot) = g(\cdot)$. Here, (16.58) follows directly from (16.57) and (16.59), and the latter is proven analogously to Theorem 3.3 using the IT Inequality (Proposition 1.12).

Because

$$0 \leq \mathrm{I}(X;Y) = \mathrm{h}(Y) - \mathrm{h}(Y|X) \tag{16.60}$$

it then also immediately follows that

$$\mathrm{h}(Y) \geq \mathrm{h}(Y|X), \tag{16.61}$$

i.e., *conditioning reduces differential entropy*.

Moreover,

$$\mathrm{I}(2X;Y) = \mathrm{I}(X;Y) \tag{16.62}$$

as can be seen from

$$\begin{aligned} \mathrm{h}(2X) - \mathrm{h}(2X|Y) &= \mathrm{h}(X) + \log 2 - \mathrm{h}(X|Y) - \log 2 && (16.63)\\ &= \mathrm{h}(X) - \mathrm{h}(X|Y). && (16.64) \end{aligned}$$

Note that from this we also learn that

$$\mathrm{h}(Y) - \mathrm{h}(Y|2X) = \mathrm{h}(Y) - \mathrm{h}(Y|X), \tag{16.65}$$

i.e., the differential entropy behaves normally in its conditioning arguments:

$$\mathrm{h}(Y|2X) = \mathrm{h}(Y|X). \tag{16.66}$$

16.4 Mixed Continuous and Discrete Random Variables

We have now understood that entropy $\mathrm{H}(\cdot)$ only handles discrete random variables, while differential entropy $\mathrm{h}(\cdot)$ only handles continuous random variables. Mutual information $\mathrm{I}(\cdot;\cdot)$, on the other hand, is happy with both types

[2] Strictly speaking, we should add "almost everywhere". But note that if two PDFs are the same almost everywhere, they describe the same probability distribution anyway.

of random variables. So the alert reader might ask what happens if we start to mix both types of random variables in mutual information? For example, letting M be a RV taking value in a finite alphabet $\mathcal{M}$ and $Y \in \mathbb{R}$ be a continuous RV, is it meaningful to consider $\mathrm{I}(M;Y)$?

From an engineering point of view this makes perfect sense. For example, consider a message M that is translated into a channel input X and then sent over a channel that adds Gaussian noise (see Chapter 17). Indeed, our engineering intuition does not cheat us, and $\mathrm{I}(M;Y)$ is well-behaved. Nevertheless there are some quite delicate mathematical issues here. For example, from

$$\mathrm{I}(M;Y) = \mathrm{h}(Y) - \mathrm{h}(Y|M) \tag{16.67}$$

we see that we need to be able to condition differential entropy on a discrete random variable. This is alright because we know that we can condition a PDF on a discrete event of positive probability:

$$\begin{aligned}\mathrm{h}(Y|M) &= \mathsf{E}_M[\mathrm{h}(Y|M=m)] && (16.68)\\ &= \sum_{m\in\mathcal{M}} \Pr[M=m]\,\mathrm{h}(Y|M=m) && (16.69)\\ &= -\sum_{m\in\mathcal{M}} \Pr[M=m] \int_{-\infty}^{\infty} f_{Y|M=m}(y)\log f_{Y|M=m}(y)\,\mathrm{d}y. && (16.70)\end{aligned}$$

Things are more murky when we expand mutual information the other way around:

$$\mathrm{I}(M;Y) = \mathrm{H}(M) - \mathrm{H}(M|Y). \tag{16.71}$$

How shall we define $\mathrm{H}(M|Y=y)$? Note that the event $\{Y=y\}$ has zero probability! So we see that we cannot avoid defining a joint distribution of M and Y. To do this properly, one should actually go into measure theory. We will not do this here, but instead use a heuristic argument that will lead to a definition (compare also with the discussion in [Lap17, Section 20.4]). So, we would like to think of $\Pr[M=m\,|\,Y=y]$ as

$$\Pr[M=m\,|\,Y=y] \overset{??}{=} \lim_{\delta\downarrow 0} \frac{\Pr[M=m, Y\in(y-\delta,y+\delta)]}{\Pr[Y\in(y-\delta,y+\delta)]}. \tag{16.72}$$

Since

$$\Pr[Y\in(y-\delta,y+\delta)] = \int_{y-\delta}^{y+\delta} f_Y(\eta)\,\mathrm{d}\eta \approx f_Y(y)\,2\delta \tag{16.73}$$

and

$$\begin{aligned}&\Pr[M=m, Y\in(y-\delta,y+\delta)]\\ &\quad= \Pr[M=m]\,\Pr[Y\in(y-\delta,y+\delta)\,|\,M=m] && (16.74)\\ &\quad= \Pr[M=m]\int_{y-\delta}^{y+\delta} f_{Y|M=m}(\eta)\,\mathrm{d}\eta && (16.75)\\ &\quad\approx \Pr[M=m]\,f_{Y|M=m}(y)\,2\delta, && (16.76)\end{aligned}$$

where the approximations become more accurate for smaller δ, this would give

$$\Pr[M=m\,|\,Y=y] \overset{??}{=} \lim_{\delta\downarrow 0} \frac{\Pr[M=m]\, f_{Y|M=m}(y)\, 2\delta}{f_Y(y)\, 2\delta} \tag{16.77}$$

$$= \frac{\Pr[M=m]\, f_{Y|M=m}(y)}{f_Y(y)}. \tag{16.78}$$

This is intuitively quite pleasing! There is only one problem left: What happens if $f_Y(y)=0$? Luckily, we do not really need to bother about this because the probability of Y taking on any value y for which $f_Y(y)=0$ is zero:

$$\Pr[Y\in\{y\colon f_Y(y)=0\}]=0. \tag{16.79}$$

So we assign an arbitrary value for these cases:

$$\Pr[M=m\,|\,Y=y] \triangleq \begin{cases} \frac{\Pr[M=m]\, f_{Y|M=m}(y)}{f_Y(y)} & \text{if } f_Y(y)>0,\\ \frac{1}{|\mathcal{M}|} & \text{otherwise.}\end{cases} \tag{16.80}$$

Now, we are perfectly able to handle mixed expressions like $\mathrm{I}(M;Y)$, using either expansion (16.67) or (16.71) (both give the same result!).

In this way, one can even handle mutual information expressions with a mixture of discrete and continuous RVs on the same side of the semicolon. Take as example the situation of Figure 11.4 and assume in contrast to the setup of Chapter 11 that the channel has a continuous-alphabet input and output, i.e., X_1^n and Y_1^n are continuous random vectors. We can now prove that the Data Processing Inequality (Lemma 11.30) still holds in this situation:

$$\mathrm{I}(M;\hat{M}) \le \mathrm{I}(M;\hat{M}) + \underbrace{\mathrm{I}(X_1^n;\hat{M}|M)}_{\ge 0} \tag{16.81}$$

$$= \mathrm{I}(M,X_1^n;\hat{M}) \tag{16.82}$$

$$= \mathrm{I}(X_1^n;\hat{M}) + \underbrace{\mathrm{I}(M;\hat{M}|X_1^n)}_{=0} \tag{16.83}$$

$$= \mathrm{I}(X_1^n;\hat{M}) \tag{16.84}$$

$$\le \mathrm{I}(X_1^n;\hat{M}) + \underbrace{\mathrm{I}(X_1^n;Y_1^n|\hat{M})}_{\ge 0} \tag{16.85}$$

$$= \mathrm{I}(X_1^n;\hat{M},Y_1^n) \tag{16.86}$$

$$= \mathrm{I}(X_1^n;Y_1^n) + \underbrace{\mathrm{I}(X_1^n;\hat{M}|Y_1^n)}_{=0} \tag{16.87}$$

$$= \mathrm{I}(X_1^n;Y_1^n) \tag{16.88}$$

where (16.83) and (16.87) hold because of Markovity. Note that this proof in this form actually looks completely identical to the situation where all involved random variables are discrete. The reason why this derivation in this form still holds is because every single mutual information expression by itself is

well-defined using expansions like (16.67) or (16.71). For example, (16.82) can be written as

$$\mathrm{I}(M, X_1^n; \hat{M}) = \mathrm{H}(\hat{M}) - \mathrm{H}(\hat{M}|M, X_1^n) \tag{16.89}$$

and (16.86) as

$$\mathrm{I}(X_1^n; \hat{M}, Y_1^n) = \mathrm{h}(X_1^n) - \mathrm{h}(X_1^n|\hat{M}, Y_1^n). \tag{16.90}$$

Note, however, that is not possible to expand the mutual information in (16.89) or (16.90) in the other direction as neither $\mathrm{H}(M, X_1^n)$ nor $\mathrm{h}(\hat{M}, Y_1^n)$ is defined.[3] In such a case one must first use the chain rule for mutual information before the other expansion becomes possible.

16.5 Multivariate Gaussian

The multivariate Gaussian distribution is one of the most important (and easiest!) distributions.[4] Therefore we spend some extra time on it.

Theorem 16.13. Let $\mathbf{X} \in \mathbb{R}^n$ be a multivariate Gaussian random vector $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathsf{K})$ where $\boldsymbol{\mu}$ denotes the mean vector and K is the $n \times n$ covariance matrix

$$\mathsf{K} = \mathsf{E}\big[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^{\mathsf{T}}\big], \tag{16.91}$$

where we assume that K is positive definite (see Appendix B.1). Then

$$\mathrm{h}(\mathbf{X}) = \frac{1}{2}\log\big((2\pi e)^n \det \mathsf{K}\big). \tag{16.92}$$

Proof: The PDF of $\mathbf{X}$ is given as follows:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathsf{K}}}\, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\mathsf{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{16.93}$$

(see Appendix B.9). Hence, we have

$$\begin{aligned}
\mathrm{h}(\mathbf{X}) &= \mathsf{E}[-\log f_{\mathbf{X}}(\mathbf{X})] && (16.94)\\
&= \mathsf{E}\left[\frac{1}{2}\log\big((2\pi)^n \det \mathsf{K}\big) + \frac{1}{2}(\mathbf{X}-\boldsymbol{\mu})^{\mathsf{T}}\mathsf{K}^{-1}(\mathbf{X}-\boldsymbol{\mu})\log e\right] && (16.95)\\
&= \frac{1}{2}\log\big((2\pi)^n \det \mathsf{K}\big)
\end{aligned}$$

[3] Note that $\mathrm{H}(M) + \mathrm{h}(X_1^n|M)$ and $\mathrm{h}(Y_1^n) + \mathrm{H}(\hat{M}|Y_1^n)$, respectively, are defined, though.

[4] If you do not agree that the Gaussian distribution is easy, have a look at Appendices A and B! Many problems involving Gaussian RVs are solvable, while for general distribution often this is not the case. Gaussians are your friends!

$$+\frac{1}{2}\mathsf{E}\left[\sum_i\sum_j(X_i-\mu_i)(\mathsf{K}^{-1})_{i,j}(X_j-\mu_j)\right]\log e \tag{16.96}$$

$$=\frac{1}{2}\log\bigl((2\pi)^n\det\mathsf{K}\bigr)$$
$$+\frac{1}{2}\sum_i\sum_j\mathsf{E}[(X_i-\mu_i)(X_j-\mu_j)](\mathsf{K}^{-1})_{i,j}\log e \tag{16.97}$$

$$=\frac{1}{2}\log\bigl((2\pi)^n\det\mathsf{K}\bigr)+\frac{1}{2}\sum_j\sum_i(\mathsf{K})_{j,i}(\mathsf{K}^{-1})_{i,j}\log e \tag{16.98}$$

$$=\frac{1}{2}\log\bigl((2\pi)^n\det\mathsf{K}\bigr)+\frac{1}{2}\sum_j(\mathsf{K}\mathsf{K}^{-1})_{j,j}\log e \tag{16.99}$$

$$=\frac{1}{2}\log\bigl((2\pi)^n\det\mathsf{K}\bigr)+\frac{1}{2}\sum_j(\mathsf{I})_{j,j}\log e \tag{16.100}$$

$$=\frac{1}{2}\log\bigl((2\pi)^n\det\mathsf{K}\bigr)+\frac{n}{2}\log e \tag{16.101}$$

$$=\frac{1}{2}\log\bigl((2\pi)^n\det\mathsf{K}\bigr)+\frac{1}{2}\log e^n \tag{16.102}$$

$$=\frac{1}{2}\log\bigl((2\pi e)^n\det\mathsf{K}\bigr). \tag{16.103}$$

□

Theorem 16.14. Let the random vector $\mathbf{X}\in\mathbb{R}^n$ have zero mean and some fixed (positive definite) covariance matrix K, i.e., $\mathsf{E}\bigl[\mathbf{X}\mathbf{X}^{\mathsf{T}}\bigr]=\mathsf{K}$. Then

$$\mathrm{h}(\mathbf{X})\le\frac{1}{2}\log\bigl((2\pi e)^n\det\mathsf{K}\bigr) \tag{16.104}$$

with equality if, and only if, $\mathbf{X}\sim\mathcal{N}(\mathbf{0},\mathsf{K})$.

Proof: This theorem follows directly from a generalization of Theorem 3.13 to continuous RVs: Let X be a continuous RV with PDF $f(\cdot)$ that satisfies the following J constraints:

$$\mathsf{E}[r_j(X)]=\int_{-\infty}^{\infty}f(x)r_j(x)\,\mathrm{d}x=\alpha_j,\quad\text{for } j=1,\ldots,\mathrm{J}, \tag{16.105}$$

for some given functions $r_1(\cdot),\ldots,r_{\mathrm{J}}(\cdot)$ and some given values $\alpha_1,\ldots,\alpha_{\mathrm{J}}$. Then $\mathrm{h}(X)$ is maximized if, and only if,

$$f(x)=f^*(x)\triangleq e^{\lambda_0+\sum_{j=1}^{\mathrm{J}}\lambda_j r_j(x)} \tag{16.106}$$

assuming that $\lambda_0,\ldots,\lambda_{\mathrm{J}}$ can be chosen such that (16.105) is satisfied and (16.106) is a PDF.

The proof is analogous to the proof of Theorem 3.13, so we omit it.

Applying this result to our situation, we have the following: Under the constraints that $\mathsf{E}[\mathbf{X}\mathbf{X}^{\mathsf{T}}] = \mathsf{K}$, i.e., $\mathsf{E}[X_i X_j] = \mathsf{K}_{i,j}$ for all i and j, and that $\mathsf{E}[\mathbf{X}] = \mathbf{0}$, i.e., $\mathsf{E}[X_i] = 0$ for all i, we see that the maximizing distribution must have a PDF of the form

$$f_{\mathbf{X}}(x_1, \dots, x_n) = e^{\mu_0 + \sum_i \mu_i x_i + \sum_i \sum_j \lambda_{i,j} x_i x_j}, \tag{16.107}$$

where the parameters μ_0, μ_i, and $\lambda_{i,j}$ must be chosen such that the PDF integrates to 1 and such that the constraints are satisfied. Note that this is not hard to do, because it is obvious that this maximizing distribution is multivariate Gaussian. The theorem then follows from Theorem 16.13.

This result can also be shown using the fact that the relative entropy is nonnegative. To this end, let $\boldsymbol{\phi}$ be the density of a zero-mean, covariance-K multivariate Gaussian random vector. Then, for a general PDF $f_{\mathbf{X}}(\cdot)$,

$$\begin{aligned}
0 \le \mathscr{D}(f_{\mathbf{X}} \| \boldsymbol{\phi}) &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log \frac{f_{\mathbf{X}}(\mathbf{x})}{\boldsymbol{\phi}(\mathbf{x})} \, \mathrm{d}\mathbf{x} && (16.108)\\
&= -\mathsf{h}(\mathbf{X}) - \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log \boldsymbol{\phi}(\mathbf{x}) \, \mathrm{d}\mathbf{x} && (16.109)\\
&= -\mathsf{h}(\mathbf{X}) - \mathsf{E}\left[\log\left(\frac{1}{\sqrt{(2\pi)^n \det \mathsf{K}}} \, e^{-\frac{1}{2}\mathbf{X}^{\mathsf{T}} \mathsf{K}^{-1} \mathbf{X}} \right) \right] && (16.110)\\
&= -\mathsf{h}(\mathbf{X}) + \frac{1}{2} \log\bigl((2\pi)^n \det \mathsf{K}\bigr) + \frac{1}{2} \mathsf{E}\left[\mathbf{X}^{\mathsf{T}} \mathsf{K}^{-1} \mathbf{X} \right] \log e && (16.111)\\
&= -\mathsf{h}(\mathbf{X}) + \frac{1}{2} \log\bigl((2\pi)^n \det \mathsf{K}\bigr) + \frac{n}{2} \log e && (16.112)\\
&= -\mathsf{h}(\mathbf{X}) + \frac{1}{2} \log\bigl((2\pi e)^n \det \mathsf{K}\bigr), && (16.113)
\end{aligned}$$

where (16.112) is shown in (16.94)–(16.101). Hence,

$$\mathsf{h}(\mathbf{X}) \le \frac{1}{2} \log\bigl((2\pi e)^n \det \mathsf{K}\bigr). \tag{16.114}$$

Equality is only possible if we have $f_{\mathbf{X}} = \boldsymbol{\phi}$, i.e., if $\mathbf{X}$ is multivariate Gaussian. □

We can specialize Theorem 16.14 from a vector to a single RV.

Corollary 16.15. Let X be a zero-mean continuous random variable with variance σ^2. Then

$$\mathsf{h}(X) \le \frac{1}{2} \log(2\pi e \sigma^2) \tag{16.115}$$

with equality if, and only if, $X \sim \mathcal{N}(0, \sigma^2)$.

Solution to Exercise 16.6: $\mathsf{H}(2X_{\mathrm{d}}) = \mathsf{H}(X_{\mathrm{d}})$ because the probabilities of $2X_{\mathrm{d}}$ and of X_{d} are identical.

Chapter 17

Gaussian Channel

17.1 Introduction

After our preparation in Chapter 16, we are now ready to change to a situation where the channel alphabets are continuous (i.e., the alphabet sizes are infinite). However, note that at the moment we still stick to a discrete time k.

The most important such continuous-alphabet discrete-time channel is the Gaussian channel.

Definition 17.1. The *Gaussian channel* has a channel output Y_k at time k given by

$$Y_k = x_k + Z_k, \tag{17.1}$$

where x_k denotes the input of the channel at time k and is assumed to be a real number $x_k \in \mathbb{R}$, and where $\{Z_k\}$ is IID $\sim \mathcal{N}(0, \sigma^2)$ denoting additive Gaussian noise. Note that like for a DMC there is no memory in the channel. It is therefore quite usual for the channel definition to drop the time index k:

$$Y = x + Z. \tag{17.2}$$

Following our notation for the DMC we describe the Gaussian channel by the input and output alphabets

$$\mathcal{X} = \mathcal{Y} = \mathbb{R} \tag{17.3}$$

and the channel law (now a conditional PDF instead of a conditional PMF!)

$$f_{Y|X}(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(y-x)^2}{2\sigma^2}}. \tag{17.4}$$

In spite of the seemingly quite different nature of this channel compared to a DMC, it turns out that it behaves very similarly in many respects. There

are only a few changes. To notice the most important difference, consider for a moment the channel capacity from an intuitive point of view.

Assume first that $\sigma^2 = 0$. Then $Z_k = 0$ deterministically and $Y_k = x_k$ without any noise. How big is the capacity in this situation? Note that x_k can take on infinitely many different values and Y_k can perfectly "reconstruct" x_k. So, it is not hard to see that $\mathrm{C} = \infty$!

So we turn to the more realistic situation $\sigma^2 > 0$. How big is capacity now? Well, note that we have no restriction on x_k, i.e., we can choose x_k as large as we want. Therefore it is still possible to choose infinitely many different values for x_k that are infinitely far apart from each other! In particular, choose

$$x_k \in \{0, \pm a, \pm 2a, \pm 3a, \pm 4a, \ldots\} \tag{17.5}$$

and let $a \to \infty$. Since the different values are infinitely far apart from each other, we will always be able to reconstruct x_k from the noisy observation Y_k. And since x_k can take on infinitely many different values, we again end up with $\mathrm{C} = \infty$!

However, note that again this is not realistic: We will not be able to choose such large values for x_k because in any practical system the available power is limited. We see that we need to introduce one or several constraints on the input. The most common such constraint is an *average-power constraint*.

Definition 17.2. We say that the input of a channel is subject to an *average-power constraint* $\mathrm{E_s}$ if we require that every codeword $\mathbf{x} = (x_1, \ldots, x_n)$ satisfy

$$\frac{1}{n}\sum_{k=1}^{n} x_k^2 \le \mathrm{E_s} \tag{17.6}$$

for some given power $\mathrm{E_s} \ge 0$.

Actually, $\mathrm{E_s}$ is not really "power", but *symbol energy*.[1] This is because we are in a discrete-time setup and power (energy per time) is not defined.

Example 17.3. We consider $x_k \in \{+\sqrt{\mathrm{E_s}}, -\sqrt{\mathrm{E_s}}\}$, which satisfies the average-power constraint:

$$\frac{1}{n}\sum_{k=1}^{n} x_k^2 = \frac{1}{n}\sum_{k=1}^{n} \mathrm{E_s} = \mathrm{E_s}. \tag{17.7}$$

Moreover, we restrict ourselves to the case $n = 1$. Then the decoder looks at Y_1 and tries to guess whether $+\sqrt{\mathrm{E_s}}$ or $-\sqrt{\mathrm{E_s}}$ has been sent. What is the optimal decision rule?

Recall our discussion of decoding in Section 11.3: The best strategy is to design an ML decoder (assuming that both messages $x_1 = \sqrt{\mathrm{E_s}}$ or $x_1 = -\sqrt{\mathrm{E_s}}$

[1] Therefore the name $\mathrm{E_s}$: E stands for *energy* and the subscript s stands for *symbol*.

are equally likely). Due to the symmetric noise (a Gaussian distribution is symmetric around 0), we immediately see that

$$\text{if } Y_k \geq 0 \implies \text{decide } +\sqrt{\mathsf{E}_\mathsf{s}}; \tag{17.8a}$$
$$\text{if } Y_k < 0 \implies \text{decide } -\sqrt{\mathsf{E}_\mathsf{s}}. \tag{17.8b}$$

What is the probability of error? Again assuming that both values of X_1 are equally likely, we get the following:

$$\begin{aligned}
\Pr(\text{error}) &= \Pr\Big(\text{error}\Big|X_1=+\sqrt{\mathsf{E}_\mathsf{s}}\Big)\Pr\Big[X_1=+\sqrt{\mathsf{E}_\mathsf{s}}\Big] \\
&\quad + \Pr\Big(\text{error}\Big|X_1=-\sqrt{\mathsf{E}_\mathsf{s}}\Big)\Pr\Big[X_1=-\sqrt{\mathsf{E}_\mathsf{s}}\Big] && (17.9)\\
&= \frac{1}{2}\Pr\Big(\text{error}\Big|X_1=+\sqrt{\mathsf{E}_\mathsf{s}}\Big) + \frac{1}{2}\Pr\Big(\text{error}\Big|X_1=-\sqrt{\mathsf{E}_\mathsf{s}}\Big) && (17.10)\\
&= \Pr\Big(\text{error}\Big|X_1=+\sqrt{\mathsf{E}_\mathsf{s}}\Big) && (17.11)\\
&= \Pr\Big[Y_1<0\Big|X_1=+\sqrt{\mathsf{E}_\mathsf{s}}\Big] && (17.12)\\
&= \mathcal{Q}\left(\frac{\sqrt{\mathsf{E}_\mathsf{s}}}{\sigma}\right) && (17.13)
\end{aligned}$$

where the $\mathcal{Q}$-function is defined as

$$\mathcal{Q}(x) \triangleq \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{\xi^2}{2}}\,\mathrm{d}\xi = \Pr[G>x] \tag{17.14}$$

for $G\sim\mathcal{N}(0,1)$ (see also Appendix A). Hence, we have transformed the Gaussian channel into a BSC with cross-over probability $\epsilon=\mathcal{Q}\Big(\frac{\sqrt{\mathsf{E}_\mathsf{s}}}{\sigma}\Big)$.

Obviously, this is not optimal. So the question is how to do it better. ◊

17.2 Information Capacity

Analogously to our discussion of channel coding and capacity for a DMC we start by defining a purely mathematical quantity: the information capacity.

Definition 17.4. The *information capacity* of a Gaussian channel with average-power constraint E_s is defined as

$$\mathsf{C}_{\inf}(\mathsf{E}_\mathsf{s}) \triangleq \max_{f_X(\cdot):\ \mathsf{E}[X^2]\leq\mathsf{E}_\mathsf{s}} \mathrm{I}(X;Y), \tag{17.15}$$

where $f_X(\cdot)$ denotes the PDF of the input X.

Note that the constraint $\mathsf{E}\big[X^2\big]\leq\mathsf{E}_\mathsf{s}$ is **not** the average-power constraint, but simply a (second-moment) constraint on the maximization defined here.

We compute the value of $C_{\text{inf}}(\mathsf{E}_{\mathrm{s}})$:

$$\begin{aligned}
\mathrm{I}(X;Y) &= \mathrm{h}(Y) - \mathrm{h}(Y|X) && (17.16)\\
&= \mathrm{h}(Y) - \mathrm{h}(X+Z|X) && (17.17)\\
&= \mathrm{h}(Y) - \mathrm{h}(Z|X) && (17.18)\\
&= \mathrm{h}(Y) - \mathrm{h}(Z) && (17.19)\\
&= \mathrm{h}(Y) - \frac{1}{2}\log(2\pi e\sigma^2) && (17.20)\\
&\le \frac{1}{2}\log(2\pi e\,\mathsf{Var}[Y]) - \frac{1}{2}\log(2\pi e\sigma^2), && (17.21)
\end{aligned}$$

where the last inequality follows from the fact that a Gaussian RV of given variance maximizes differential entropy (Corollary 16.15). This step can be achieved with equality if (and only if) Y is Gaussian distributed with variance $\mathsf{Var}[Y]$. This is actually possible if we choose the input to be Gaussian as well.

So we have realized that the best thing is to choose the input to be Gaussian distributed. It only remains to figure out what variance to choose. Since our expression depends on the variance of Y, we compute its value:

$$\begin{aligned}
\mathsf{Var}[Y] &= \mathsf{E}\Big[\big(Y-\mathsf{E}[Y]\big)^2\Big] && (17.22)\\
&= \mathsf{E}\Big[\big(X+Z-\mathsf{E}[X]-\mathsf{E}[Z]\big)^2\Big] && (17.23)\\
&= \mathsf{E}\Big[\big(X+Z-\mathsf{E}[X]\big)^2\Big] && (17.24)\\
&= \mathsf{E}\Big[\big(X-\mathsf{E}[X]\big)^2\Big] - 2\,\mathsf{E}[Z(X-\mathsf{E}[X])] + \mathsf{E}\big[Z^2\big] && (17.25)\\
&= \mathsf{E}\Big[\big(X-\mathsf{E}[X]\big)^2\Big] - 2\underbrace{\mathsf{E}[Z]}_{=0}\,\mathsf{E}[X-\mathsf{E}[X]] + \underbrace{\mathsf{E}\big[Z^2\big]}_{=\sigma^2} && (17.26)\\
&= \mathsf{E}\Big[\big(X-\mathsf{E}[X]\big)^2\Big] + \sigma^2 && (17.27)\\
&= \mathsf{E}\big[X^2\big] - \underbrace{\big(\mathsf{E}[X]\big)^2}_{\ge 0} + \sigma^2 && (17.28)\\
&\le \mathsf{E}\big[X^2\big] + \sigma^2 && (17.29)\\
&\le \mathsf{E}_{\mathrm{s}} + \sigma^2. && (17.30)
\end{aligned}$$

Here, in (17.24) we use that Z has zero mean; (17.26) follows because Z and X are independent; and the last step (17.30) follows by the constraint on the maximization. Actually, we could have arrived at (17.28) from (17.22) directly if we had remembered that for $X \perp\!\!\!\perp Z$

$$\mathsf{Var}[X+Z] = \mathsf{Var}[X] + \mathsf{Var}[Z]. \tag{17.31}$$

Note that the upper bound (17.30) can actually be achieved if we choose X to have zero mean and variance E_{s}.

Hence,

$$\mathrm{I}(X;Y) \le \frac{1}{2}\log 2\pi e\big(\mathsf{E}_{\mathrm{s}}+\sigma^2\big) - \frac{1}{2}\log 2\pi e\sigma^2 = \frac{1}{2}\log\Big(1+\frac{\mathsf{E}_{\mathrm{s}}}{\sigma^2}\Big), \tag{17.32}$$

and this upper bound can be achieved if (and only if) we choose $X \sim \mathcal{N}(0, \mathsf{E}_\mathsf{s})$. So we have shown the following.

Proposition 17.5. The information capacity of a Gaussian channel is

$$C_{\inf}(\mathsf{E}_\mathsf{s}) = \frac{1}{2}\log\left(1 + \frac{\mathsf{E}_\mathsf{s}}{\sigma^2}\right) \tag{17.33}$$

and is achieved by a Gaussian input $X \sim \mathcal{N}(0, \mathsf{E}_\mathsf{s})$.

Remark 17.6. Note that from an energy point of view it is silly to transmit a mean $\mathsf{E}[X] \neq 0$ because the mean is deterministic and does therefore not contain any information (the decoder can always simply subtract it from the received signal without changing the mutual information) and because transmitting a mean uses (wastes!) power:

$$\mathsf{E}\big[X^2\big] - (\mathsf{E}[X])^2 \le \mathsf{E}\big[X^2\big]. \tag{17.34}$$

△

17.3 Channel Coding Theorem

Next, we will prove a coding theorem for the Gaussian channel. The proof will be very similar to the proof of the coding theorem for a DMC given in Chapter 11. The main difference is that we have to take into account the average-power constraint (17.6).

We quickly repeat the definition of a coding scheme and its most fundamental parameters.

Definition 17.7. An (M, n) *coding scheme* for the Gaussian channel with average-power constraint E_s consists of

- a message set $\mathcal{M} = \{1, 2, \dots, \mathsf{M}\}$;
- a codebook of M length-n codewords $\mathbf{x}(m) = (x_1(m), \dots, x_n(m))$, where every codeword needs to satisfy the average-power constraint

$$\frac{1}{n}\sum_{k=1}^{n} x_k^2(m) \le \mathsf{E}_\mathsf{s}, \quad m = 1, \dots, \mathsf{M}; \tag{17.35}$$

- an encoding function $\phi\colon \mathcal{M} \to \mathbb{R}^n$ that maps the message m into the codeword $\mathbf{x}(m)$; and
- a decoding function $\psi\colon \mathbb{R}^n \to \hat{\mathcal{M}}$ that maps the received sequence $\mathbf{y}$ into a guess $\hat{m}$ of which message has been transmitted, where usually $\hat{\mathcal{M}} \triangleq \mathcal{M} \cup \{0\}$ (with 0 denoting "error").

Definition 17.8. The *rate* R of an (M, n) coding scheme is defined as

$$\mathrm{R} \triangleq \frac{\log_2 \mathrm{M}}{n} \tag{17.36}$$

and is said to be *achievable* if there exists a sequence of $(\lceil 2^{n\mathrm{R}} \rceil, n)$ coding schemes (with codewords satisfying the power constraint (17.35)) such that the maximal probability of error $\lambda^{(n)}$ tends to zero. The *operational capacity* is the supremum of all achievable rates.

We now have the following fundamental result.

Theorem 17.9 (Coding Theorem for the Gaussian Channel).
The operational capacity of the Gaussian channel with average-power constraint $\mathrm{E_s}$ and noise variance σ^2 is

$$\mathrm{C}(\mathrm{E_s}) = \frac{1}{2}\log_2\left(1+\frac{\mathrm{E_s}}{\sigma^2}\right) \text{ bits per transmission.} \tag{17.37}$$

Hence, operational capacity and information capacity are identical, and we simply speak of **capacity**.

17.3.1 Plausibility

Before we properly prove the coding theorem, we give a plausibility argument of why it holds.

So assume we transmit a codeword $\mathbf{x}$ over the channel and receive a vector $\mathbf{Y} = \mathbf{x}+\mathbf{Z}$. Hence, $\mathbf{Y}$ lies with high probability in a sphere of radius r around $\mathbf{x}$, where

$$r = \sqrt{\mathsf{E}[\|\mathbf{Z}\|^2]} = \sqrt{\mathsf{E}\left[Z_1^2+\cdots+Z_n^2\right]} = \sqrt{n\sigma^2}. \tag{17.38}$$

So, consider such a sphere around each of the possible codewords. As long as there exist no codewords whose spheres overlap, we will be able to guess the right message with high probability.

Due to the power constraint on the input, our available energy for each component of the codewords is limited to $\mathrm{E_s}$ on average. Therefore, the received sequences are likely to be in a sphere of radius

$$r_{\text{tot}} = \sqrt{\mathsf{E}[\|\mathbf{Y}\|^2]} = \sqrt{\mathsf{E}\left[Y_1^2+\cdots+Y_n^2\right]} \tag{17.39}$$

$$= \sqrt{\sum_{k=1}^{n}\mathsf{E}\left[Y_k^2\right]} \tag{17.40}$$

$$= \sqrt{\sum_{k=1}^{n}\left(\mathsf{E}\left[X_k^2\right]+\mathsf{E}\left[Z_k^2\right]\right)} \tag{17.41}$$

$$= \sqrt{\mathsf{E}\left[\sum_{k=1}^{n} X_k^2\right] + n\sigma^2} \tag{17.42}$$

$$\leq \sqrt{n\mathrm{E_s} + n\sigma^2} \tag{17.43}$$

$$= \sqrt{n(\mathrm{E_s} + \sigma^2)}, \tag{17.44}$$

where in (17.43) we have used (17.35).

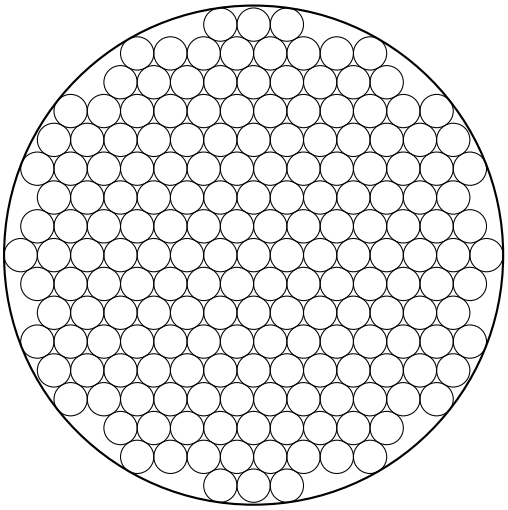


Figure 17.1: The large sphere depicts the n-dimensional space of all possible received vectors. Each small sphere depicts a codeword with some noise around it. To maximize the number of possible messages, we try to put as many small spheres into the large one as possible, but without having overlap such as to make sure that the receiver will not confuse two messages due to the noise.

Hence, to make sure that we can transmit as much information as possible, we need to try to use as many codewords as possible, but the small spheres of the codewords should not overlap. So, the question is: How many small spheres of radius r can be squeezed into the large sphere of radius r_{tot} without having any overlap (see Figure 17.1)?

Note that a sphere of radius r in an n-dimensional space has a volume of $a_n r^n$ where a_n is a constant that depends on n, but not on r. So, if we do not worry about the shape of the spheres, then at most we can squeeze the following number of small spheres into the big sphere:

$$\mathrm{M} = \frac{\mathrm{Vol(large\ sphere)}}{\mathrm{Vol(small\ sphere)}} = \frac{a_n r_{\mathrm{tot}}^n}{a_n r^n} = \frac{\left(\sqrt{n(\mathrm{E_s}+\sigma^2)}\right)^n}{\left(\sqrt{n\sigma^2}\right)^n} = \left(\frac{\mathrm{E_s}+\sigma^2}{\sigma^2}\right)^{\frac{n}{2}}. \tag{17.45}$$

Hence, we get a rate

$$\mathrm{R} = \frac{\log \mathrm{M}}{n} = \frac{1}{n} \cdot \frac{n}{2} \log\left(\frac{\mathrm{E_s} + \sigma^2}{\sigma^2}\right) = \frac{1}{2} \log\left(1 + \frac{\mathrm{E_s}}{\sigma^2}\right) = \mathrm{C_{inf}}(\mathrm{E_s}). \tag{17.46}$$

This is exactly what Theorem 17.9 claims.

17.3.2 Achievability

Next we have a look at a rigorous proof. We start with a lower bound on the rate, i.e., the achievability part of the proof. The idea is very similar to the proof given in Section 11.8. The only difference is the additional difficulty of having an average-power constraint to be taken care of.

We go through the following steps:

1: **Codebook Design:** Identically to the coding theorem for a DMC we generate a codebook at random. To this end, we fix a rate R, a codeword length n, and a PDF $f_X(\cdot)$. From the information capacity calculations above we know that the capacity-achieving input distribution is Gaussian, hence our first attempt is to choose

$$f_X(x) = \frac{1}{\sqrt{2\pi \mathrm{E_s}}} e^{-\frac{x^2}{2\mathrm{E_s}}}. \tag{17.47}$$

But if we generate all codewords like that, what happens with the average-power constraint? There is hope: By the weak law of large numbers we know that

$$\frac{1}{n} \sum_{k=1}^{n} X_k^2 \overset{n\to\infty}{\longrightarrow} \mathsf{E}[X^2] = \mathrm{E_s} \qquad \text{in probability.} \tag{17.48}$$

Hence, it looks like that with a high chance the randomly generated codewords will satisfy the power constraint.

But we have to be careful with the details! What we actually need in our analysis of the error probability is the following property: For any given $\epsilon > 0$ we must have that

$$\Pr\left[\frac{1}{n} \sum_{k=1}^{n} X_k^2 > \mathrm{E_s}\right] \le \epsilon \tag{17.49}$$

for some n large enough. Unfortunately, (17.48) only guarantees that for given $\epsilon > 0$ and $\delta > 0$ there exists an n_0 such that for all $n \ge n_0$

$$\Pr\left[\left|\frac{1}{n} \sum_{k=1}^{n} X_k^2 - \mathsf{E}[X^2]\right| > \epsilon\right] \le \delta. \tag{17.50}$$

Choosing $\delta = \epsilon$, we obtain from (17.50):

$$\epsilon \geq \Pr\left[\left|\frac{1}{n}\sum_{k=1}^{n} X_k^2 - \mathsf{E}\big[X^2\big]\right| > \epsilon\right] \tag{17.51}$$

$$= \Pr\left(\left\{\frac{1}{n}\sum_{k=1}^{n} X_k^2 - \mathsf{E}\big[X^2\big] > \epsilon\right\} \cup \left\{\frac{1}{n}\sum_{k=1}^{n} X_k^2 - \mathsf{E}\big[X^2\big] < -\epsilon\right\}\right) \tag{17.52}$$

$$\geq \Pr\left[\frac{1}{n}\sum_{k=1}^{n} X_k^2 - \mathsf{E}\big[X^2\big] > \epsilon\right] \tag{17.53}$$

$$= \Pr\left[\frac{1}{n}\sum_{k=1}^{n} X_k^2 > \mathsf{E}\big[X^2\big] + \epsilon\right]. \tag{17.54}$$

This differs from (17.49) by a small ϵ added to the expected energy $\mathsf{E}\big[X^2\big]$. So, to make sure that (17.54) agrees with the needed (17.49), we must choose

$$\mathsf{E}\big[X^2\big] + \epsilon = \mathsf{E}_{\mathrm{s}}, \tag{17.55}$$

i.e., for the random generation of the codewords we do not use the distribution in (17.47), but instead we reduce the variance slightly: We generate every letter of all codewords IID $\sim \mathcal{N}(0, \mathsf{E}_{\mathrm{s}} - \epsilon)$:

$$f_X(x) = \frac{1}{\sqrt{2\pi(\mathsf{E}_{\mathrm{s}} - \epsilon)}}\, e^{-\frac{x^2}{2(\mathsf{E}_{\mathrm{s}}-\epsilon)}}. \tag{17.56}$$

Note that since ϵ can be chosen arbitrarily small, we are not really concerned about this small change.

So we randomly generate the codebook IID according to (17.56), and then show the codebook to both the encoder and the decoder.

Remark 17.10. Note that in our random design of the codebook we do not have the guarantee that all codewords will satisfy the power constraint (17.35)! (We only have the property that a violation of the power constraint is not very likely to happen too often.) However, in spite of this, we do **not** delete a codeword that does not satisfy the power constraint!

From a practical point of view this makes no sense at all, but it simplifies our analysis considerably: If we checked every codeword and regenerated some of them if they did not satisfy the power constraint, then we would introduce a dependency between the different letters in each codeword and our analysis would break down. Instead we will take care of the "illegal" codewords at the receiver side, see below. △

2: **Encoder Design:** Given some message $m \in \{1, \dots, \lceil 2^{nR} \rceil\}$ that is chosen by the source according to a uniform distribution

$$\Pr[M = m] = \frac{1}{\lceil 2^{nR} \rceil}, \tag{17.57}$$

the encoder picks the mth codeword $\mathbf{X}(m)$ and sends it over the channel.

3: **Decoder Design:** We again use a threshold decoder based on the instantaneous mutual information

$$\mathrm{i}(\mathbf{x};\mathbf{y}) \triangleq \log \frac{f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}{f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{Y}}(\mathbf{y})}, \tag{17.58}$$

where

$$f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) = \prod_{k=1}^{n} f_X(x_k)\, f_{Y|X}(y_k|x_k) \tag{17.59}$$

with $f_X(\cdot)$ defined in (17.56) and $f_{Y|X}(\cdot|\cdot)$ denoting the channel law, i.e., the conditional PDF of the channel output given the channel input given in (17.4).

The decoder receives a sequence $\mathbf{Y}$ and searches for an $\hat{m}$ such that for some given threshold $\beta > 0$,

- for $\hat{m}$

$$\mathrm{i}(\mathbf{X}(\hat{m});\mathbf{y}) > \log_2 \beta; \tag{17.60}$$

- for all $\tilde{m} \neq \hat{m}$

$$\mathrm{i}(\mathbf{X}(\tilde{m});\mathbf{y}) \le \log_2 \beta; \tag{17.61}$$

- the codeword $\mathbf{X}(\hat{m})$ satisfies the average-power constraint

$$\frac{1}{n}\sum_{k=1}^{n} X_k^2(\hat{m}) \le \mathsf{E}_{\mathrm{s}}. \tag{17.62}$$

If the decoder can find such an $\hat{m}$, then it will put out this $\hat{m}$, otherwise it puts out $\hat{m} = 0$, i.e., it declares an error.

We choose the threshold β as in Section 11.8 to be

$$\beta \triangleq 2^{n(\mathrm{I}(X;Y)-\epsilon)} \tag{17.63}$$

for some fixed $\epsilon > 0$. Here $\mathrm{I}(X;Y)$ is the mutual information between a channel input symbol X and its corresponding output symbol Y when the input has the distribution $f_X(\cdot)$ given in (17.56), i.e.,

$$\begin{aligned}
\mathrm{I}(X;Y) &= \mathrm{h}(Y) - \mathrm{h}(Y|X) && (17.64)\\
&= \mathrm{h}(X+Z) - \mathrm{h}(Z) && (17.65)\\
&= \frac{1}{2}\log 2\pi e\big(\mathsf{E}_{\mathrm{s}} - \epsilon + \sigma^2\big) - \frac{1}{2}\log 2\pi e\sigma^2 && (17.66)\\
&= \frac{1}{2}\log\left(1 + \frac{\mathsf{E}_{\mathrm{s}} - \epsilon}{\sigma^2}\right). && (17.67)
\end{aligned}$$

Note that if the power constraint is not satisfied by the used codeword, the decoder will for sure declare an error, even if it actually could have decoded correctly. With this (from a practical point of view very strange) decoding rule we actually fix our issue of using codewords that violate the average-power constraint. For more details, see *Part 5: Strengthening* below.

4: **Performance Analysis**: Identically to the argument given in the proof of the coding theorem of a DMC, without loss of generality we can assume that $M = 1$ is transmitted (recall that each codeword is generated randomly and independently of all other codewords anyway, so it does not matter which message is chosen; see (11.124)–(11.130)). We now define the following events:

$$\mathcal{F}_0 \triangleq \left\{ \frac{1}{n} \sum_{k=1}^{n} X_k^2(1) > \mathrm{E_s} \right\}, \tag{17.68}$$

$$\mathcal{F}_m \triangleq \Big\{ \mathrm{i}(\mathbf{X}(m); \mathbf{Y}) > \log_2 \beta \Big\}, \quad m = 1, \ldots, \lceil 2^{n\mathrm{R}} \rceil, \tag{17.69}$$

i.e., $\mathcal{F}_0$ is the event that the transmitted codeword violates the power constraint, and $\mathcal{F}_m$ is the event that the mth codeword and the received sequence $\mathbf{Y}$ have an instantaneous mutual information above the given threshold. Hence an error occurs if $\mathcal{F}_0$ occurs (i.e., the power constraint is violated), if $\mathcal{F}_1^{\mathrm{c}}$ occurs (i.e., if the transmitted codeword and the received sequence have a too small instantaneous mutual information), or if $\mathcal{F}_2 \cup \mathcal{F}_3 \cup \cdots \cup \mathcal{F}_{\lceil 2^{n\mathrm{R}} \rceil}$ occurs (i.e., if one or more wrong codewords have an instantaneous mutual information with the received sequence $\mathbf{Y}$ that is too big). Using the Union Bound, we obtain:

$$\begin{aligned} \Pr(\text{error}) &= \Pr(\text{error} \,|\, M = 1) && (17.70)\\ &= \Pr\Big(\mathcal{F}_0 \cup \mathcal{F}_1^{\mathrm{c}} \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \cdots \cup \mathcal{F}_{\lceil 2^{n\mathrm{R}} \rceil} \,\Big|\, M = 1 \Big) && (17.71)\\ &\le \Pr(\mathcal{F}_0 \,|\, M = 1) + \Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M = 1) \\ &\quad + \sum_{m=2}^{\lceil 2^{n\mathrm{R}} \rceil} \Pr(\mathcal{F}_m \,|\, M = 1). && (17.72) \end{aligned}$$

From (17.54) and (17.55) we know that

$$\Pr(\mathcal{F}_0 \,|\, M = 1) \le \epsilon. \tag{17.73}$$

Then from our choice (17.63) and from the weak law of large numbers it follows that

$$\begin{aligned} \Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M = 1) &= \Pr\big[\mathrm{i}(\mathbf{X}(1); \mathbf{Y}) \le \log_2 \beta \,\big|\, M = 1 \big] && (17.74)\\ &= \Pr\left[\sum_{k=1}^{n} \mathrm{i}(X_k; Y_k) \le n\big(\mathrm{I}(X;Y) - \epsilon\big) \right] && (17.75) \end{aligned}$$

$$\le \Pr\left[\left|\frac{1}{n}\sum_{k=1}^{n} \imath(X_k;Y_k) - \mathrm{I}(X;Y)\right| \ge \epsilon\right] \tag{17.76}$$

$$\le \epsilon \qquad \text{(for } n \text{ large enough).} \tag{17.77}$$

And for $m \ge 2$, we have

$$\Pr(\mathcal{F}_m \,|\, M=1) = \Pr\big[\imath(\mathbf{X}(m);\mathbf{Y}) > \log_2 \beta \,\big|\, M=1\big] \tag{17.78}$$

$$= \Pr\left[\log_2 \frac{f_{\mathbf{X},\mathbf{Y}}(\mathbf{X}(m),\mathbf{Y})}{f_{\mathbf{X}}(\mathbf{X}(m)) f_{\mathbf{Y}}(\mathbf{Y})} > \log_2 \beta \,\middle|\, M=1\right] \tag{17.79}$$

$$= \Pr\left[\frac{f_{\mathbf{X},\mathbf{Y}}(\mathbf{X}(m),\mathbf{Y})}{f_{\mathbf{X}}(\mathbf{X}(m)) f_{\mathbf{Y}}(\mathbf{Y})} > \beta \,\middle|\, M=1\right] \tag{17.80}$$

$$= \int\cdots\int_{\substack{(\mathbf{x},\mathbf{y}) \text{ such that}\\ f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) > \beta f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{Y}}(\mathbf{y})}} f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{Y}}(\mathbf{y})\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y} \tag{17.81}$$

$$= \int\cdots\int_{\substack{(\mathbf{x},\mathbf{y}) \text{ such that}\\ f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{Y}}(\mathbf{y}) < \frac{1}{\beta} f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}} \underbrace{f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{Y}}(\mathbf{y})}_{< \frac{1}{\beta} f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y} \tag{17.82}$$

$$< \int\cdots\int_{\substack{(\mathbf{x},\mathbf{y}) \text{ such that}\\ f_{\mathbf{X}}(\mathbf{x}) f_{\mathbf{Y}}(\mathbf{y}) < \frac{1}{\beta} f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})}} \frac{1}{\beta} f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y} \tag{17.83}$$

$$\le \frac{1}{\beta} \underbrace{\int\cdots\int f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y}}_{=1} = \frac{1}{\beta}. \tag{17.84}$$

Here, again, we make use of the fact that $\mathbf{X}(m)$ is independent of $\mathbf{Y}$ such that their joint density is in product form.

Combining these results with (17.72) now yields

$$\Pr(\text{error}) < \epsilon + \epsilon + \sum_{m=2}^{\lceil 2^{n\mathrm{R}} \rceil} \frac{1}{\beta} \tag{17.85}$$

$$= 2\epsilon + \underbrace{\left(\lceil 2^{n\mathrm{R}} \rceil - 1\right)}_{\le 2^{n\mathrm{R}}} 2^{-n(\mathrm{I}(X;Y)-\epsilon)} \tag{17.86}$$

$$\le 2\epsilon + 2^{-n(\mathrm{I}(X;Y)-\mathrm{R}-\epsilon)} \tag{17.87}$$

$$\le 3\epsilon, \tag{17.88}$$

if n is sufficiently large and if $\mathrm{I}(X;Y) - \mathrm{R} - \epsilon > 0$ so that the exponent is negative. Hence we see that for any $\epsilon > 0$ we can choose n such that the average error probability, averaged over all codewords and all codebooks, is less than 3ϵ as long as

$$\mathrm{R} < \mathrm{I}(X;Y) - \epsilon = \frac{1}{2}\log\left(1 + \frac{\mathrm{E_s} - \epsilon}{\sigma^2}\right) - \epsilon, \tag{17.89}$$

where we have used (17.67). Note that this is arbitrarily close to the information capacity.

5: Strengthening: Finally, we strengthen the proof:

- Firstly, we would like to get rid of the average over the codebooks. Since our average error probability is small ($\le 3\epsilon$), there must exist at least one codebook $\mathscr{C}^*$ that has an equally small error probability, i.e., we can find a codebook $\mathscr{C}^*$ satisfying
$$P_{\mathrm{e}}^{(n)}(\mathscr{C}^*) \le 3\epsilon. \tag{17.90}$$
- Secondly, we strengthen the proof to obtain a result with respect to the maximum error probability $\lambda^{(n)}$ instead of the average error probability $P_{\mathrm{e}}^{(n)}$. To this end, we consider the good codebook $\mathscr{C}^*$, for which we know that (17.90) holds, and order the codewords according to their error probabilities λ_m, $m = 1, \dots, \lceil 2^{n\mathrm{R}} \rceil$. Let $\tilde{\lambda} \triangleq \lambda_{\lfloor 2^{n\mathrm{R}}/2 \rfloor}$ be the error probability of the worst codeword of the better half. Then
$$\begin{aligned}
3\epsilon &\ge P_{\mathrm{e}}^{(n)}(\mathscr{C}^*) && (17.91)\\
&= \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{m=1}^{\lceil 2^{n\mathrm{R}} \rceil} \lambda_m && (17.92)\\
&= \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{\substack{m=1\\ \text{(better half)}}}^{\lfloor 2^{n\mathrm{R}}/2 \rfloor} \underbrace{\lambda_m}_{\ge 0} + \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{\substack{m=\lfloor 2^{n\mathrm{R}}/2 \rfloor + 1\\ \text{(worse half)}}}^{\lceil 2^{n\mathrm{R}} \rceil} \underbrace{\lambda_m}_{\ge \tilde{\lambda}} && (17.93)\\
&\ge \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{m=\lfloor 2^{n\mathrm{R}}/2 \rfloor + 1}^{\lceil 2^{n\mathrm{R}} \rceil} \tilde{\lambda} && (17.94)\\
&= \frac{\lceil 2^{n\mathrm{R}} \rceil - \lfloor 2^{n\mathrm{R}}/2 \rfloor}{\lceil 2^{n\mathrm{R}} \rceil} \cdot \tilde{\lambda} && (17.95)\\
&\ge \frac{\tilde{\lambda}}{2}. && (17.96)
\end{aligned}$$
Thus, we have shown $\tilde{\lambda} \le 6\epsilon$. So if we design a new codebook with only half the amount of codewords by taking the good codebook $\mathscr{C}^*$ and throwing away the worse half of the codewords, we get a new codebook $\tilde{\mathscr{C}}^*$ with a maximum error probability being very small:
$$\tilde{\lambda}^{(n)} \le 6\epsilon. \tag{17.97}$$
Note that this new codebook $\tilde{\mathscr{C}}^*$ is also guaranteed to have all codewords satisfying the average-power constraint (17.35) because the error probability of any codeword that does not satisfy the power constraint is 1 and thus could not satisfy (17.97).
But by throwing away half of the codewords, we have also changed the rate: The rate of $\tilde{\mathscr{C}}^*$ satisfies
$$\tilde{\mathrm{R}} = \frac{\log_2\bigl(\lfloor \frac{2^{n\mathrm{R}}}{2} \rfloor\bigr)}{n} = \frac{\log_2\bigl(\lfloor 2^{n\mathrm{R}-1} \rfloor\bigr)}{n} \le \frac{n\mathrm{R}-1}{n} = \mathrm{R} - \frac{1}{n} \tag{17.98}$$

and condition (17.89) becomes

$$\tilde{\mathsf{R}} \le \mathsf{R} - \frac{1}{n} \tag{17.99}$$
$$< \frac{1}{2}\log\left(1+\frac{\mathsf{E}_{\mathsf{s}}-\epsilon}{\sigma^2}\right) - \epsilon - \frac{1}{n} \tag{17.100}$$
$$\le \frac{1}{2}\log\left(1+\frac{\mathsf{E}_{\mathsf{s}}-\epsilon}{\sigma^2}\right) - 2\epsilon, \tag{17.101}$$

where in (17.101) we choose $n \ge \frac{1}{\epsilon}$.

- Finally, note that ϵ is a parameter that we can choose freely. Hence, we can make it as small as we wish, so that condition (17.101) can be replaced by

$$\tilde{\mathsf{R}} < \frac{1}{2}\log\left(1+\frac{\mathsf{E}_{\mathsf{s}}}{\sigma^2}\right). \tag{17.102}$$

Thus, we have shown that it is possible to design a coding scheme with arbitrarily small error probability if n is chosen large enough and if the rate is smaller than the information capacity.

17.3.3 Converse

Consider any $(\lceil 2^{n\mathsf{R}}\rceil, n)$ coding scheme that satisfies the average-power constraint (17.35) for all $m = 1,\dots,\lceil 2^{n\mathsf{R}}\rceil$. Using the Fano Inequality (Corollary 11.27) and defining $P_{\mathsf{e}}^{(n)} = \Pr[M \neq \hat{M}]$, we have

$$\mathrm{H}(M|\hat{M}) \le \log_2 2 + P_{\mathsf{e}}^{(n)} \log_2(\lceil 2^{n\mathsf{R}}\rceil) \tag{17.103}$$
$$\le 1 + P_{\mathsf{e}}^{(n)} \log_2(2^{n\mathsf{R}+1}) \tag{17.104}$$
$$= 1 + P_{\mathsf{e}}^{(n)} \cdot (n\mathsf{R}+1) \text{ bits}, \tag{17.105}$$

where in (17.104) we bounded $\lceil 2^{n\mathsf{R}}\rceil \le 2^{n\mathsf{R}} + 1 \le 2\cdot 2^{n\mathsf{R}} = 2^{n\mathsf{R}+1}$. Recalling that M is uniformly distributed over $\{1,\dots,\lceil 2^{n\mathsf{R}}\rceil\}$, i.e.,

$$\mathrm{H}(M) = \log_2(\lceil 2^{n\mathsf{R}}\rceil) \ge n\mathsf{R} \text{ bits}, \tag{17.106}$$

we bound as follows:

$$n\mathsf{R} \le \mathrm{H}(M) \tag{17.107}$$
$$= \mathrm{I}(M;\hat{M}) + \mathrm{H}(M|\hat{M}) \tag{17.108}$$
$$\le \mathrm{I}(M;\hat{M}) + 1 + P_{\mathsf{e}}^{(n)} \cdot (n\mathsf{R}+1) \quad \text{(by (17.105))} \tag{17.109}$$
$$\le \mathrm{I}(X_1^n;Y_1^n) + 1 + P_{\mathsf{e}}^{(n)} \cdot (n\mathsf{R}+1) \quad \text{(DPI)} \tag{17.110}$$
$$= \mathrm{h}(Y_1^n) - \mathrm{h}(Y_1^n|X_1^n) + 1 + P_{\mathsf{e}}^{(n)} \cdot (n\mathsf{R}+1) \tag{17.111}$$
$$= \mathrm{h}(Y_1^n) - \mathrm{h}(Z_1^n) + 1 + P_{\mathsf{e}}^{(n)} \cdot (n\mathsf{R}+1) \tag{17.112}$$

$$
\begin{aligned}
&= \sum_{k=1}^{n} \Big(\mathrm{h}(Y_k|Y_1^{k-1}) - \mathrm{h}(Z_k|Z_1^{k-1}) \Big) \\
&\quad + 1 + P_{\mathrm{e}}^{(n)} \cdot (n\mathrm{R} + 1) && \text{(chain rule)} && (17.113)\\
&= \sum_{k=1}^{n} \Big(\mathrm{h}(Y_k|Y_1^{k-1}) - \mathrm{h}(Z_k) \Big) \\
&\quad + 1 + P_{\mathrm{e}}^{(n)} \cdot (n\mathrm{R} + 1) && (\{Z_k\} \text{ IID}) && (17.114)\\
&\le \sum_{k=1}^{n} \big(\mathrm{h}(Y_k) - \mathrm{h}(Z_k) \big) + 1 + P_{\mathrm{e}}^{(n)} \cdot (n\mathrm{R} + 1) \text{ bits.} && && (17.115)
\end{aligned}
$$

Here, (17.110) follows from the Data Processing Inequality[2] (Lemma 11.30) based on the Markov chain $M \multimap \mathbf{X} \multimap \mathbf{Y} \multimap \hat{M}$ (see Definition 11.28), and the last inequality follows because conditioning cannot increase entropy.

Now let

$$\mathsf{E}_k \triangleq \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \sum_{m=1}^{\lceil 2^{n\mathrm{R}} \rceil} x_k^2(m) \tag{17.116}$$

be the average energy of the kth component of the codewords when averaged uniformly over all codewords. Note that due to the average-power constraint (17.35) we have

$$\frac{1}{n} \sum_{k=1}^{n} \mathsf{E}_k \le \mathsf{E}_{\mathrm{s}}. \tag{17.117}$$

Moreover note that

$$
\begin{aligned}
\mathsf{E}[Y_k^2] &= \mathsf{E}\Big[(X_k + Z_k)^2\Big] && && (17.118)\\
&= \mathsf{E}[X_k^2] + \mathsf{E}[Z_k^2] && (X_k \perp\!\!\!\perp Z_k \text{ and } \mathsf{E}[Z_k] = 0) && (17.119)\\
&= \mathsf{E}\Big[x_k^2(M)\Big] + \sigma^2 && (X_k \text{ is random because of } M) && (17.120)\\
&= \sum_{m=1}^{\lceil 2^{n\mathrm{R}} \rceil} \frac{1}{\lceil 2^{n\mathrm{R}} \rceil} \cdot x_k^2(m) + \sigma^2 && (M \text{ is uniformly distributed}) && (17.121)\\
&= \mathsf{E}_k + \sigma^2 && \text{(by definition (17.116)).} && (17.122)
\end{aligned}
$$

Hence, we continue with (17.115) as follows:

$$
\begin{aligned}
\mathrm{R} &\le \frac{1}{n} \sum_{k=1}^{n} \big(\mathrm{h}(Y_k) - \mathrm{h}(Z_k) \big) + \frac{1}{n} + P_{\mathrm{e}}^{(n)} \cdot \left(\mathrm{R} + \frac{1}{n} \right) && (17.123)\\
&\le \frac{1}{n} \sum_{k=1}^{n} \left(\frac{1}{2} \log_2\big(2\pi e \operatorname{Var}[Y_k]\big) - \mathrm{h}(Z_k) \right) + \frac{1}{n} + P_{\mathrm{e}}^{(n)} \cdot \left(\mathrm{R} + \frac{1}{n} \right) && (17.124)
\end{aligned}
$$

[2] Recall from the discussion of (16.81)–(16.88) that the Data Processing Inequality continues to hold even if we mix continuous with discrete RVs!

$$\le \frac{1}{n}\sum_{k=1}^{n}\left(\frac{1}{2}\log_2\left(2\pi e\,\mathsf{E}[Y_k^2]\right)-\mathrm{h}(Z_k)\right)+\frac{1}{n}+P_{\mathrm{e}}^{(n)}\cdot\left(\mathsf{R}+\frac{1}{n}\right) \tag{17.125}$$

$$= \frac{1}{n}\sum_{k=1}^{n}\left(\frac{1}{2}\log_2\left(2\pi e\left(\mathsf{E}_k+\sigma^2\right)\right)-\frac{1}{2}\log_2\left(2\pi e\sigma^2\right)\right)+\frac{1}{n}+P_{\mathrm{e}}^{(n)}\cdot\left(\mathsf{R}+\frac{1}{n}\right) \tag{17.126}$$

$$= \frac{1}{n}\sum_{k=1}^{n}\frac{1}{2}\log_2\left(1+\frac{\mathsf{E}_k}{\sigma^2}\right)+\frac{1}{n}+P_{\mathrm{e}}^{(n)}\cdot\left(\mathsf{R}+\frac{1}{n}\right) \tag{17.127}$$

$$\le \frac{1}{2}\log_2\left(1+\frac{1}{n}\sum_{k=1}^{n}\frac{\mathsf{E}_k}{\sigma^2}\right)+\frac{1}{n}+P_{\mathrm{e}}^{(n)}\cdot\left(\mathsf{R}+\frac{1}{n}\right) \tag{17.128}$$

$$\le \frac{1}{2}\log_2\left(1+\frac{\mathsf{E}_{\mathrm{s}}}{\sigma^2}\right)+\frac{1}{n}+P_{\mathrm{e}}^{(n)}\cdot\left(\mathsf{R}+\frac{1}{n}\right)\ \text{bits}. \tag{17.129}$$

Here, (17.124) follows from Theorem 16.14 that shows that for given variance and mean the differential entropy is maximized by a Gaussian RV; the subsequent inequality (17.125) follows from

$$\mathsf{Var}[Y_k]=\mathsf{E}\left[Y_k^2\right]-\left(\mathsf{E}[Y_k]\right)^2\le\mathsf{E}\left[Y_k^2\right]; \tag{17.130}$$

(17.126) follows from (17.122); in (17.128) we apply the Jensen Inequality (Theorem 2.1); and the final step (17.129) is due to (17.117) and the monotonicity of $\log(\cdot)$.

Hence, if $P_{\mathrm{e}}^{(n)}\to 0$ as $n\to\infty$, we must have

$$\mathsf{R}\le\frac{1}{2}\log\left(1+\frac{\mathsf{E}_{\mathrm{s}}}{\sigma^2}\right). \tag{17.131}$$

This concludes the proof of the converse and thereby the proof of the coding theorem.

We add a small remark. The derivation (17.110)–(17.129) actually shows that (11.101) still holds in the context of a Gaussian channel and in presence of an average-power constraint:

$$\frac{1}{n}\,\mathrm{I}(X_1^n;Y_1^n)\le\mathsf{C}_{\mathrm{inf}}(\mathsf{E}_{\mathrm{s}}). \tag{17.132}$$

17.4 Joint Source and Channel Coding Theorem

Similarly to the discrete-alphabet case, next we would like to combine data compression and channel transmission. I.e., we will look at the combined information transmission system shown in Figure 17.2, where a general discrete stationary source shall be transmitted over the Gaussian channel.

If we recall our derivations of the Information Transmission Theorem (Theorem 15.3) and the Source Channel Coding Separation Theorem (Corollary 15.4) for a DMC, we will quickly note that the derivations can be taken over one-to-one to the situation of a Gaussian channel. So we will not bother to re-derive the results again, but simply state it. Note that instead of C we will directly use the capacity expression of the Gaussian channel (17.37).

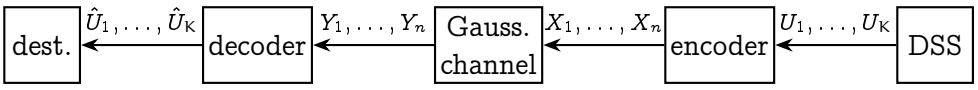


Figure 17.2: Information transmission system using a joint source channel coding scheme.

Theorem 17.11 (Information Transmission Theorem and Source Channel Coding Separation Theorem for the Gaussian Channel).
Consider a Gaussian channel with average-power constraint E_s and noise variance σ^2, and a finite-alphabet stationary and ergodic source $\{U_k\}$. Then, if

$$\frac{H(\{U_k\})}{T_s} \text{ bits/s} < \frac{1}{2T_c} \log_2\left(1 + \frac{E_s}{\sigma^2}\right) \text{ bits/s} \tag{17.133}$$

(where T_s and T_c are the symbol durations of the source and the channel, respectively), there exists a joint source channel coding scheme with a probability of a decoding error $P_e^{(K)} \to 0$ as $K \to \infty$.

Moreover, it is possible to design the transmission system such as to firstly compress the source data in a fashion that is independent of the Gaussian channel and then use a channel code for the Gaussian channel that is independent of the source.

Conversely, for any stationary source $\{U_k\}$ with

$$\frac{H(\{U_k\})}{T_s} \text{ bits/s} > \frac{1}{2T_c} \log_2\left(1 + \frac{E_s}{\sigma^2}\right) \text{ bits/s}, \tag{17.134}$$

the probability of a decoding error $P_e^{(K)}$ cannot tend to zero, i.e., it is not possible to send the source data over the Gaussian channel with arbitrarily low error probability.

We can also take over our discussion from Section 15.6 about the limitations of communication when we transmit above capacity: The lower bound to the average bit error rate (BER) given in expression (15.78) also holds for the Gaussian channel.

So, we assume an IID uniform binary source and, in (15.78), we replace the capacity C by the capacity expression of the Gaussian channel (17.33) and define

$$N_0 \triangleq 2\sigma^2 \qquad \text{(noise energy level in Joule}^3\text{)}; \tag{17.135}$$

$$E_b \triangleq E_s \cdot \frac{T_s}{T_c} = E_s \cdot \frac{1}{R} \qquad \text{(energy per information bit in Joule)}; \tag{17.136}$$

[3] For a justification for this definition, see (18.46).

$$\gamma_{\mathrm{b}} \triangleq \frac{\mathsf{E}_{\mathrm{b}}}{\mathsf{N}_0} \qquad \text{(signal-to-noise ratio).} \tag{17.137}$$

Here we again have used (compare with (15.38))

$$\mathsf{R} = \frac{\mathsf{T}_{\mathrm{c}}}{\mathsf{T}_{\mathrm{s}}} \mathrm{H}(\{U_k\}) = \frac{\mathsf{T}_{\mathrm{c}}}{\mathsf{T}_{\mathrm{s}}} \text{ bits} \tag{17.138}$$

for $\{U_k\}$ IID uniform binary.

Using these definitions, we transform (15.77) into

$$\mathrm{H}_{\mathrm{b}}(P_{\mathrm{b}}) \geq 1 - \frac{1}{2\mathsf{R}} \log_2(1 + 2\mathsf{R}\gamma_{\mathrm{b}}) \tag{17.139}$$

or

$$P_{\mathrm{b}} \geq \mathrm{H}_{\mathrm{b}}^{-1}\Big(1 - \frac{1}{2\mathsf{R}} \log_2(1 + 2\mathsf{R}\gamma_{\mathrm{b}})\Big). \tag{17.140}$$

Here, again, $\mathrm{H}_{\mathrm{b}}^{-1}(\cdot)$ is the inverse function of the binary entropy function $\mathrm{H}_{\mathrm{b}}(\xi)$ for $\xi \in \left[0, \frac{1}{2}\right]$ (and we define $\mathrm{H}_{\mathrm{b}}^{-1}(\zeta) \triangleq 0$ for $\zeta < 0$). As we have discussed in Section 15.6 already, this lower bound is tight, i.e., there exist schemes that can approach it arbitrarily closely.

The curve (17.140) is shown in Figure 17.3 as a function of the signal-to-noise ratio (SNR) for $\mathsf{R} = \frac{1}{2}$ and $\mathsf{R} = \frac{1}{3}$. We see that there exists a rate-$\frac{1}{2}$ system design that can achieve an average bit error rate $P_{\mathrm{b}} = 10^{-5}$ at an SNR γ_{b} close to 0 dB, i.e., for $\mathsf{E}_{\mathrm{b}} \approx \mathsf{N}_0$.

On the other hand, we are sure that no system with a rate $\mathsf{R} = \frac{1}{2}$ can yield a BER less than 10^{-5} if the signal energy per information bit E_{b} is less than the noise energy level N_0.

Due to its fundamental importance, the lower bound (17.140) is called *Shannon limit*. For many years after Shannon has drawn this line in 1948 [Sha48], researchers have tried very hard to find a design that can actually achieve this lower bound. Over the years and with great effort, they managed to gradually close the gap between the existing transmission schemes and the theoretical limit. For a long period between the 1960s and the early 1990s, the Shannon limit could only be approached to around 2 dB. However, in 1993 a big breakthrough was presented at a conference in Geneva: The *turbo code* was the first design that managed to get within 1 dB of the limit [BGT93].

A couple of years later, between 1996–1998, the *low-density parity-check (LDPC) codes* were rediscovered [MN96], [DM98] and shown to reduce the performance gap to even smaller values, e.g., 0.1 dB. We use "rediscover" here because LDPC codes were originally proposed by Robert G. Gallager in 1962 [Gal62]. Yet, at that time, the research community did not realize their potential. One might argue that due to their high complexity, the computers

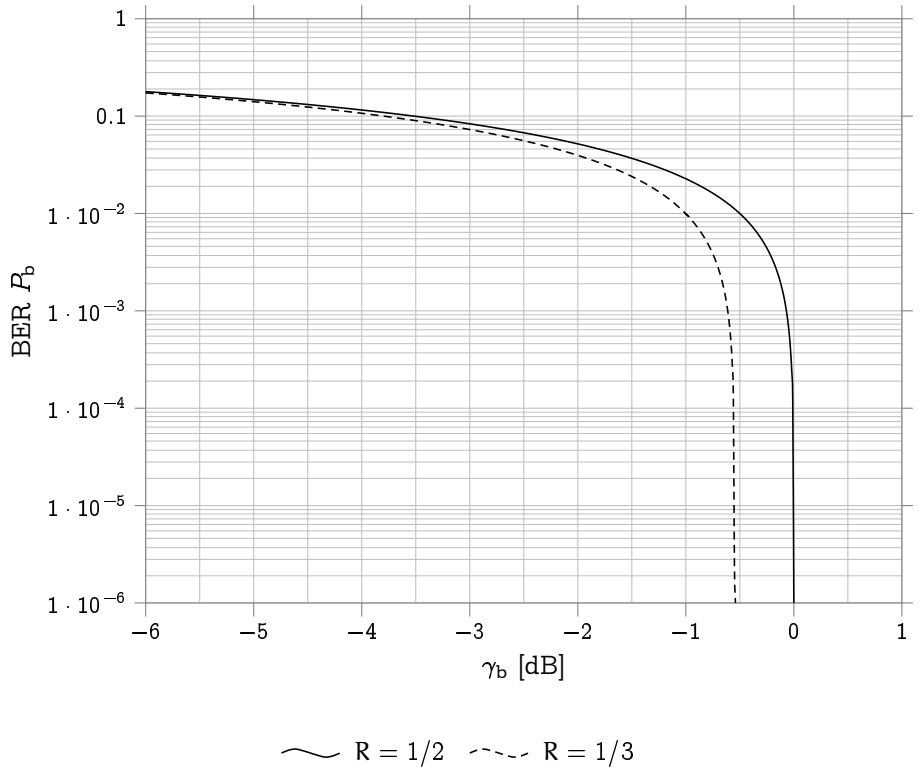


Figure 17.3: Shannon limit for the Gaussian channel.

at that time could not perform any simulations on LDPC codes and that therefore this discovery went unnoticed for so long. However, interestingly enough, Gallager actually ran quite a lot of numerical computations in the early 60s already and demonstrated performance values of LDPC codes up to an error probability of 10^{-4}. They were completely ignored for 35 years...

So it took the engineers about 50 years (from 1948 to 1998) until they finally managed to catch up with the pioneering prediction of Shannon for a DMC and for the Gaussian channel.

Chapter 18

Bandlimited Channels

In Chapter 17 we have dropped the assumption of discrete alphabets and allowed more general continuous alphabets. Now we will also drop the simplification of discrete time and consider a continuous-time channel.

As a matter of fact, in the physical world almost every signal and system is of continuous-time nature. Still, we engineers prefer to work with discrete-time problems because they are so much easier to handle and control. Therefore textbooks often start directly with a discrete-time model and simply assume that the sampling and digitalization has been taken care of. This script is no exception!

In this chapter, however, we will try to get a flavor of the real physical world and how one can try to convert it into a discrete-time setup. We have to warn the reader that this conversion from continuous time to discrete time is mathematically very tricky. We will therefore try to give a rough overview only and skip over quite a few mathematical details. For those readers who are interested in a clean treatment, the book by Amos Lapidoth [Lap17] is highly recommended.

Note that in this chapter we will be using the Fourier transform. It is assumed that the reader has a good basic knowledge of the Fourier transform and its properties.

18.1 Additive White Gaussian Noise Channel

In contrast to the regularly clocked discrete-time systems, in real continuous-time systems, the shape of the signal is free to change at any time, but we usually have a constraint concerning the used frequencies: Only a limited frequency band is available, either because of technical limitations (any filter or amplifier has implicit band limitations) or, even more importantly, because the frequency bands are licensed. Hence, the input signal to our channel is not only constrained with respect to its average power, but also with respect to its frequencies.

We consider the following channel model, that is the simplest, but at the same time also the most important continuous-time channel model in engineering. Its name is *additive white Gaussian noise (AWGN) channel*. For a given input signal at time $t \in \mathbb{R}$, the channel output is

$$Y(t) = x(t) + Z(t). \tag{18.1}$$

As already mentioned, the input $x(\cdot)$ has to satisfy an average-power constraint P and is bandlimited to W Hz, i.e., it only may contain frequencies in the interval $[-W, W]$.

The term $Z(t)$ denotes additive noise. We will assume that $Z(\cdot)$ is a *white Gaussian noise process* with a double-sided[1] *power spectral density (PSD)* that is flat inside of the band $[-W, W]$ and has a value $\frac{N_0}{2}$. Such a PSD is called *white* in an analogy to white light that contains all frequencies equally strongly. Since the input of the channel is restricted to lie inside the band $[-W, W]$, it seems reasonable that we do not need to care what the PSD of $Z(\cdot)$ looks like outside of this band. This intuition can be proven formally.

Claim 18.1. Without loss of generality we may bandlimit $Y(\cdot)$ to the band $[-W, W]$.

Proof: We omit the proof and refer the interested reader to [Lap17]. This result is intuitively pleasing because all components of $Y(\cdot)$ outside the frequency band only contain noise and no signal components, so it indeed seems clear that we can ignore them. □

Hence, we do not bother about specifying the PSD completely, but simply leave its value outside of $[-W, W]$ unspecified. Note, however, that the PSD certainly cannot be white for all frequencies because such a process would have infinite power!

Beside the already mentioned constraints on the used power and frequencies, from a practical point of view, the input signal also needs some kind of time limitation because any practical system will only send finite-duration signals. But here our first mathematical problem occurs: By a fundamental property of the Fourier transform no signal can simultaneously be strictly timelimited and bandlimited!

To deal with this issue we set up the system as follows. We design an encoder that receives a certain number of IID uniform binary information bits as input. Depending on the value of these information bits, it then chooses from a given set of strictly bandlimited signals one signal that it transmits

[1] We call the PSD *double-sided* because we define it both for positive and negative frequencies, $-\infty < f < \infty$. Some people prefer to only consider the (physically possible) positive frequencies $f \geq 0$. The PSD is then called *single-sided* and is simply twice the value of the double-sided PSD, such that when integrated over its range, both version yield the same value for the total power.

over the channel. We could call these signals *codeword-signals*, since they corresponds to the codewords used in the discrete-time setup. Recall that beside the bandwidth restrictions all codeword-signals also need to satisfy the average-power constraint (where the average is to be understood to be over the time-duration of the signal).

The decoder will receive a distorted version of the transmitted signal. It will observe this received signal for a certain limited time duration of T seconds and then make a guess of which codeword signal has been transmitted. Note that we can now define the *rate* of our system in a similar fashion to the discrete-time setup: The rate is defined as the logarithmic number of possible codeword-signals, normalized to the duration of the signal T (or rather the duration of the observation of the signal).

Due to the noise in the channel and because the decoder only observes during T seconds, there will be a certain strictly positive probability of making an error. However, we hope that as long as the rate is small enough, we can make this error arbitrarily small by making T large. So, we see that the signal duration (observation duration) T in the continuous-time coding system corresponds to the blocklength n for a discrete-time coding system.

We describe this setup mathematically.

Definition 18.2. A $(2^{\mathsf{RT}}, \mathsf{T})$ *coding scheme* for the AWGN channel with average-power constraint P consists of

- 2^{RT} codeword-signals $x^{(m)}(\cdot)$ that are bandlimited to W and that satisfy the average-power constraint

$$\frac{1}{\mathsf{T}}\int_0^{\mathsf{T}} \bigl(x^{(m)}(t)\bigr)^2 \,\mathrm{d}t \le \mathsf{P}, \quad m = 1, \ldots, 2^{\mathsf{RT}}; \tag{18.2}$$

- an encoder ϕ that for a message $M = m$ transmits the codeword-signal $x^{(m)}(\cdot)$; and
- a decoder ψ that makes a decision $\hat{M}$ about which message has been sent based on the received signal $Y(t)$ for $t \in [0, \mathsf{T}]$.

Here, R denotes the *rate* and is defined as (compare to Definition 11.16):

$$\mathsf{R} \triangleq \frac{\log_2(\#\text{ of codeword-signals})}{\mathsf{T}}, \tag{18.3}$$

measured in bits per second. A rate is said to be *achievable* if there exists a sequence of $(2^{\mathsf{RT}}, \mathsf{T})$ coding schemes such that the error probability $\Pr[\hat{M} \neq M]$ tends to zero as $\mathsf{T} \to \infty$.

The alert reader will note that we are cheating here: By limiting only the receiver to the time interval $[0, \mathsf{T}]$, but not the encoder, we try to cover up the problem that a timelimited signal at the transmitter for sure is not

bandlimited. How we are supposed to transmit an infinite-length signal in the first place is completely ignored... However, this issue is then "resolved" in our derivation once we let T tend to infinity.

18.2 Sampling Theorem

The basic idea of our treatment of bandlimited time-continuous signals is to rely on so-called *complete orthonormal systems (CONS)*,[2] i.e., on basis functions that can be used to describe these functions. We are sure that you already know a CONS that holds for all energy-limited signals: the signal description based on the signal's samples as given by the Sampling Theorem.

Theorem 18.3 (Sampling Theorem (by Nyquist & Shannon)).
Let $s(\cdot)$ be a finite-energy function that is bandlimited to $[-W, W]$ Hz. Then $s(\cdot)$ is completely specified by the samples of the function spaced $\frac{1}{2W}$ seconds apart and can be written as

$$s(t) = \sum_{k=-\infty}^{\infty} s\left(\frac{k}{2W}\right) \operatorname{sinc}(2Wt - k) \tag{18.4}$$

where

$$\operatorname{sinc}(t) \triangleq \begin{cases} \frac{\sin(\pi t)}{\pi t} & t \neq 0, \\ 1 & t = 0. \end{cases} \tag{18.5}$$

We note that

$$\xi_k(t) \triangleq \frac{1}{\sqrt{2W}} \operatorname{sinc}(2Wt - k) \tag{18.6}$$

are the basis functions used to describe $s(\cdot)$ and that

$$\sqrt{2W} \cdot s\left(\frac{k}{2W}\right) \tag{18.7}$$

denote the coordinates of $s(\cdot)$ in this basis.

Proof: The proof relies on the Fourier series of a periodic function $g(\cdot)$: Suppose $g(x)$ is periodic with period X, i.e.,

$$g(x + \ell \mathrm{X}) = g(x), \quad x \in \mathbb{R},\ \ell \in \mathbb{Z}. \tag{18.8}$$

Then we know from the theory of Fourier series that $g(\cdot)$ can be written as

$$g(x) = \frac{1}{\mathrm{X}} \sum_{k=-\infty}^{\infty} c_k \, e^{-\mathrm{i}2\pi k \frac{x}{\mathrm{X}}} \tag{18.9}$$

[2] For more details, see Appendix C.8.

with coefficients

$$c_k \triangleq \int_0^X g(x)\, e^{\mathrm{i}2\pi k \frac{x}{X}}\, \mathrm{d}x. \tag{18.10}$$

We now apply this idea to a periodic function $\hat{s}_{\mathrm{p}}(f)$ with period $2W$. I.e.,

$$\hat{s}_{\mathrm{p}}(f) = \frac{1}{2W} \sum_{k=-\infty}^{\infty} c_k\, e^{-\mathrm{i}2\pi k \frac{f}{2W}} \tag{18.11}$$

where

$$c_k = \int_{-W}^{W} \hat{s}_{\mathrm{p}}(f)\, e^{\mathrm{i}2\pi f \frac{k}{2W}}\, \mathrm{d}f. \tag{18.12}$$

Note that we have not yet specified what $\hat{s}_{\mathrm{p}}(f)$ actually is. We correct this lapse by choosing

$$c_k \triangleq s\left(\frac{k}{2W}\right). \tag{18.13}$$

Now $\hat{s}_{\mathrm{p}}(f)$ is well defined. But what is it? To find out we next compute the inverse Fourier transform[3] of the bandlimited signal $s(t)$ using its Fourier transform $\hat{s}(f)$:

$$s(t) = \int_{-\infty}^{\infty} \hat{s}(f)\, e^{\mathrm{i}2\pi f t}\, \mathrm{d}f = \int_{-W}^{W} \hat{s}(f)\, e^{\mathrm{i}2\pi f t}\, \mathrm{d}f. \tag{18.14}$$

Note that we can restrict the integration limits because we have assumed that $s(\cdot)$ is bandlimited in the first place.

Now consider the points $t = \frac{k}{2W}$:

$$s\left(\frac{k}{2W}\right) = \int_{-W}^{W} \hat{s}(f)\, e^{\mathrm{i}2\pi f \frac{k}{2W}}\, \mathrm{d}f. \tag{18.15}$$

Comparing (18.15) with (18.12), we see that we have

$$\int_{-W}^{W} \hat{s}_{\mathrm{p}}(f)\, e^{\mathrm{i}2\pi f \frac{k}{2W}}\, \mathrm{d}f = \int_{-W}^{W} \hat{s}(f)\, e^{\mathrm{i}2\pi f \frac{k}{2W}}\, \mathrm{d}f \tag{18.16}$$

for all $k \in \mathbb{Z}$, i.e., we must have that

$$\hat{s}_{\mathrm{p}}(f) = \hat{s}(f), \quad \forall\, f \in [-W, W]. \tag{18.17}$$

[3] Since capital letters are already reserved for random quantities, in this script we use a hat to denote the Fourier transform of a continuous signal. Also note that our definition of Fourier transform does not make use of the radial frequency ω. The advantage is that we do not need to bother about many factors of 2π. Moreover, f has a very fundamental physical meaning, while I do not really understand the physical meaning of ω. For more about this choice, I highly recommend to read the first couple of chapters (in particular Chapter 6) of [Lap17].

Since $\hat{s}_{\mathrm{p}}(f)$ is a periodic function with period $2W$, it also follows that outside the band $[-W,W]$, the function $\hat{s}_{\mathrm{p}}(f)$ will simply repeat itself periodically. So, finally, we understand that by our definition (18.13) we have chosen $\hat{s}_{\mathrm{p}}(f)$ to be the periodic extension of $\hat{s}(f)$ with period $2W$!

We now also see a way of getting back our original function $s(\cdot)$ from its samples $\{s(k/2W)\}$:

- Using the samples $s(\frac{k}{2W})$ as coefficients c_k in (18.11) we get $\hat{s}_{\mathrm{p}}(f)$.
- From $\hat{s}_{\mathrm{p}}(f)$ we can then get $\hat{s}(f)$ back using an ideal lowpass filter $\hat{h}_{\mathrm{LP}}(f)$:
$$\hat{s}(f) = \hat{h}_{\mathrm{LP}}(f) \cdot \hat{s}_{\mathrm{p}}(f) = \hat{h}_{\mathrm{LP}}(f) \cdot \frac{1}{2W} \sum_{k=-\infty}^{\infty} s\left(\frac{k}{2W}\right) e^{-\mathrm{i}2\pi k \frac{f}{2W}}. \quad (18.18)$$
- Finally we get $s(t)$ back by the inverse Fourier transform.

This proves the first part of the theorem.

To derive the expression (18.4), we use the engineering trick of Dirac Delta "functions" $\delta(\cdot)$. Note that $\delta(\cdot)$ is not a proper function in the mathematical sense. But we sloppily describe it as a function

$$\delta(t) \triangleq \begin{cases} 0 & \text{for all } t \neq 0, \\ \infty & t = 0, \end{cases} \quad (18.19)$$

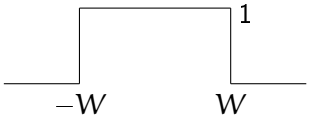
where the "value" ∞ is chosen such that

$$\int_{-\infty}^{\infty} \delta(t)\,\mathrm{d}t = 1. \quad (18.20)$$

See also Example 16.4.

So consider (18.18) and make the following observations:

- The Fourier transform of a Dirac Delta $\delta(t-k\mathsf{T})$ is
$$\delta(t-k\mathsf{T}) \circ\!\!-\!\!\bullet \int_{-\infty}^{\infty} \delta(t-k\mathsf{T})\, e^{-\mathrm{i}2\pi f t}\,\mathrm{d}t = e^{-\mathrm{i}2\pi f k\mathsf{T}}, \quad (18.21)$$
so that
$$\hat{s}_{\mathrm{p}}(f) = \frac{1}{2W} \sum_{k=-\infty}^{\infty} s\left(\frac{k}{2W}\right) e^{-\mathrm{i}2\pi k \frac{f}{2W}}$$
$$\bullet\!\!-\!\!\circ\; s_{\mathrm{p}}(t) = \frac{1}{2W} \sum_{k=-\infty}^{\infty} s\left(\frac{k}{2W}\right) \delta\left(t - \frac{k}{2W}\right). \quad (18.22)$$
- The inverse Fourier transform of a lowpass filter $\hat{h}_{\mathrm{LP}}(\cdot)$ with bandwidth W is



$$= \hat{h}_{\mathrm{LP}}(f) \bullet\!\!-\!\!\circ\; h_{\mathrm{LP}}(t) = 2W \operatorname{sinc}(2Wt). \quad (18.23)$$

Hence we get

$$\begin{aligned}
\hat{s}(f) &= \hat{h}_{\mathrm{LP}}(f) \cdot \hat{s}_{\mathrm{p}}(f) \\
\bullet\!\!-\!\!\circ\; s(t) &= \mathrm{h}_{\mathrm{LP}}(t) \star s_{\mathrm{p}}(t) && (18.24)\\
&= 2W \operatorname{sinc}(2Wt) \star \frac{1}{2W} \sum_{k=-\infty}^{\infty} s\left(\frac{k}{2W}\right) \delta\left(t - \frac{k}{2W}\right) && (18.25)\\
&= \sum_{k=-\infty}^{\infty} s\left(\frac{k}{2W}\right) \operatorname{sinc}\left(2W\left(t - \frac{k}{2W}\right)\right) && (18.26)\\
&= \sum_{k=-\infty}^{\infty} s\left(\frac{k}{2W}\right) \operatorname{sinc}(2Wt - k). && (18.27)
\end{aligned}$$

This proves the second part of the theorem. □

Remark 18.4 (Sampling via Shah-Function). There is a nice engineering way of remembering and understanding the Sampling Theorem. We define the *Shah-function* as

$$\text{⅄}_{\mathsf{T}}(t) \triangleq \sum_{k=-\infty}^{\infty} \delta(t - k\mathsf{T}). \tag{18.28}$$

This is a strange "function", but it has some cool properties. The coolest is the following:

$$\text{⅄}_{\mathsf{T}}(t) \circ\!\!-\!\!\bullet\; \frac{1}{\mathsf{T}} \text{⅄}_{\frac{1}{\mathsf{T}}}(f), \tag{18.29}$$

i.e., the Fourier transform of a Shah-function is again a Shah-function! Moreover, observe the following:

- Multiplying a function $s(t)$ by a Shah-function $\text{⅄}_{\mathsf{T}}(t)$ is equivalent to sampling the function $s(\cdot)$ in intervals of T:

  $$s_{\mathrm{p}}(t) = s(t) \cdot \mathsf{T}\, \text{⅄}_{\mathsf{T}}(t) \tag{18.30}$$

  (the additional factor T is introduced here to make sure that the energy of $s_{\mathrm{p}}(\cdot)$ is identical to the energy of $s(\cdot)$).

- Convolving a function $\hat{s}(f)$ with a Shah-function $\text{⅄}_{W}(f)$ is equivalent to periodically repeating the function $\hat{s}(\cdot)$ with period W:

  $$\hat{s}_{\mathrm{p}}(f) = \hat{s}(f) \star \text{⅄}_{W}(f). \tag{18.31}$$

Now, these two operations are coupled by the Fourier transform! So we understand the following beautiful relation:

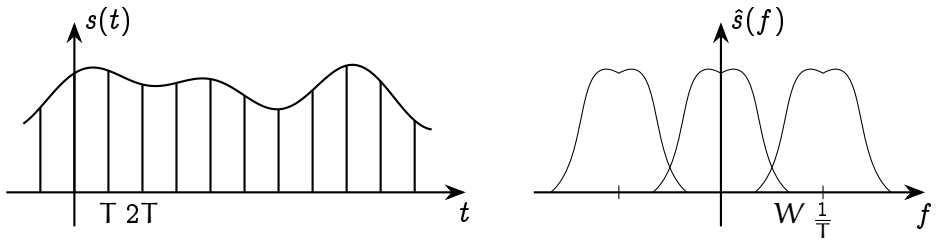


Figure 18.1: Sampling and replicating frequency spectrum: This figure depicts the relationship between sampling a bandlimited time signal and periodically repeating the corresponding spectrum.

$$s_{\mathrm{p}}(t) = s(t) \cdot \mathsf{T} \amalg_{\mathsf{T}}(t) \quad \circ\!\!-\!\!\bullet \quad \hat{s}_{\mathrm{p}}(f) = \hat{s}(f) \star \amalg_{\frac{1}{\mathsf{T}}}(f). \tag{18.32}$$

The relationship is also depicted in Figure 18.1.

We see that if $\frac{1}{\mathsf{T}} \leq 2W$ we will have overlap in the spectrum (so-called *aliasing*) and therefore we will not be able to gain the original spectrum back.

On the other hand, if $\mathsf{T} = \frac{1}{2W}$, then there is no overlap and we can gain the original function back using an ideal lowpass filter.

If we sample at a slightly higher rate than $2W$, i.e., $\mathsf{T} < \frac{1}{2W}$, then we can even relax our requirement on the lowpass filter, i.e., it does not need to be ideal anymore, but we can actually use a filter that can be realized in practice. △

18.3 From Continuous To Discrete Time

So the basic idea of the Sampling Theorem (Theorem 18.3) or any other CONS is that any signal $s(\cdot)$ can be described using some basis functions:

$$s(t) = \sum_k s_k \, \xi_k(t). \tag{18.33}$$

The basis function have the property of being *orthonormal*. This means that they are orthogonal to each other and that they are normalized to have unit-energy:

$$\int_{-\infty}^{\infty} \xi_k(t) \, \xi_{k'}(t) \, \mathrm{d}t = \begin{cases} 1 & \text{if } k = k', \\ 0 & \text{if } k \neq k'. \end{cases} \tag{18.34}$$

Due to this property it is easy to derive the coordinates s_k of $s(\cdot)$:

$$s_k = \int_{-\infty}^{\infty} s(t) \, \xi_k(t) \, \mathrm{d}t \tag{18.35}$$

as can be seen when plugging (18.33) into (18.35):

$$\int_{-\infty}^{\infty} s(t)\,\xi_k(t)\,\mathrm{d}t = \int_{-\infty}^{\infty} \left(\sum_{\ell} s_\ell\,\xi_\ell(t)\right) \xi_k(t)\,\mathrm{d}t \tag{18.36}$$

$$= \sum_{\ell} s_\ell \underbrace{\int_{-\infty}^{\infty} \xi_\ell(t)\,\xi_k(t)\,\mathrm{d}t}_{\substack{=1 \text{ for } \ell=k,\\ =0 \text{ otherwise}}} \tag{18.37}$$

$$= s_k. \tag{18.38}$$

The problem for us now is that we actually do not want to describe $Y(t)$ for all $t \in \mathbb{R}$ using infinitely many samples, but actually would like to constrain $Y(t)$ to the interval $t \in [0, \mathsf{T}]$. According to the Sampling Theorem we sample at a rate of $\frac{1}{2W}$, i.e., we have $2W\mathsf{T}$ samples in this interval. So, we would simply love to consider only these samples and write

$$Y(t) \approx \sum_{k=1}^{2W\mathsf{T}} Y_k\,\xi_k(t), \quad t \in [0, \mathsf{T}]. \tag{18.39}$$

This, however, does not really work because $\xi_k(t)$ are sinc-functions and therefore decay very slowly resulting in large contributions outside of the required time-interval $[0, \mathsf{T}]$. (We keep ignoring the additional problem that cutting any function to $[0, \mathsf{T}]$ will destroy its band limitations...)

However, one can save oneself from this problem if rather than relying on the CONS based on the Sampling Theorem, we use another CONS based on the family of the so-called *prolate spheroidal functions* $\zeta_k(t)$. These orthonormal basis functions have some very nice properties:

- They are strictly bandlimited.
- Even though they are infinite-time, they concentrate most of their energy inside the time interval $[0, \mathsf{T}]$.
- They retain their orthogonality even when they are restricted to $[0, \mathsf{T}]$.

Hence, a signal that is bandlimited to $[-W, W]$ can described in the time-interval $[0, \mathsf{T}]$ using only those $2W\mathsf{T}$ orthonormal basis functions $\zeta_k(\cdot)$ that mainly live inside $[0, \mathsf{T}]$. The remaining basis functions only give a very small contribution that can be neglected. For more details, see Appendix C.8 and [Lap17, Chapter 8].

To cut these long explanations short (and ignoring many details), it is basically possible to write

$$Y(t) = Y_1 \cdot \zeta_1(t) + Y_2 \cdot \zeta_2(t) + \cdots + Y_{2W\mathsf{T}} \cdot \zeta_{2W\mathsf{T}}(t), \tag{18.40}$$

$$x^{(m)}(t) = x_1^{(m)} \cdot \zeta_1(t) + x_2^{(m)} \cdot \zeta_2(t) + \cdots + x_{2W\mathsf{T}}^{(m)} \cdot \zeta_{2W\mathsf{T}}(t), \quad m = 1, \ldots, 2^{\mathsf{RT}}, \tag{18.41}$$

$$Z(t) = Z_1 \cdot \zeta_1(t) + Z_2 \cdot \zeta_2(t) + \cdots + Z_{2W\mathsf{T}} \cdot \zeta_{2W\mathsf{T}}(t). \tag{18.42}$$

Hence, we can describe the channel model (18.1) using the following $2WT$ equations:

$$Y_k = x_k + Z_k, \quad k = 1, \dots, 2WT. \tag{18.43}$$

Here the coordinates of the white Gaussian noise process $Z(t)$

$$Z_k \triangleq \int_0^{\mathsf{T}} Z(t) \cdot \zeta_k(t)\,\mathrm{d}t \tag{18.44}$$

are random variables that turn out to be IID $\sim \mathcal{N}(0, \frac{\mathsf{N}_0}{2})$ (for more details see Appendix C.5). So we realize that (18.43) actually corresponds to a Gaussian channel model

$$Y = x + Z \tag{18.45}$$

as discussed in Chapter 17, with a noise variance

$$\sigma^2 = \frac{\mathsf{N}_0}{2} \tag{18.46}$$

and with a blocklength $n = 2WT$.

By the way, note that the energy of $Z(t)$ inside the band $[-W, W]$ and inside the time-interval $t \in [0, \mathsf{T}]$ is identical to the total energy of all $2WT$ samples Z_k:

$$\underbrace{\underbrace{2W}_{\text{bandwidth}} \cdot \underbrace{\frac{\mathsf{N}_0}{2}}_{\text{PSD}}}_{\text{power in band } [-W,W]} \cdot \underbrace{\mathsf{T}}_{\text{duration}} \stackrel{!}{=} \underbrace{2WT}_{\#\text{ of samples}} \cdot \underbrace{\frac{\mathsf{N}_0}{2}}_{\text{variance}}. \tag{18.47}$$

It only remains to derive how the average-power constraint (18.2) translates into an average-power constraint E_{s} for this Gaussian channel. Recall that for the discrete-time model (18.45) the average-power constraint is given as

$$\frac{1}{n}\sum_{k=1}^{n} x_k^2 \le \mathsf{E}_{\mathrm{s}}. \tag{18.48}$$

Plugging the description (18.41) into the average-power constraint (18.2) we get

$$\begin{aligned}
\mathsf{P} &\ge \frac{1}{\mathsf{T}} \int_0^{\mathsf{T}} x^2(t)\,\mathrm{d}t && (18.49)\\
&= \frac{1}{\mathsf{T}} \int_0^{\mathsf{T}} \big(x_1 \cdot \zeta_1(t) + x_2 \cdot \zeta_2(t) + \cdots + x_{2WT} \cdot \zeta_{2WT}(t)\big)^2\,\mathrm{d}t && (18.50)\\
&= \frac{1}{\mathsf{T}} \Big(x_1^2 + x_2^2 + \cdots + x_{2WT}^2\Big) && (18.51)\\
&= 2W \cdot \frac{1}{2WT} \sum_{k=1}^{2WT} x_k^2 && (18.52)\\
&= 2W \cdot \frac{1}{n} \sum_{k=1}^{n} x_k^2, && (18.53)
\end{aligned}$$

where in (18.51) we have used the orthonormality of $\zeta_k(\cdot)$. Hence, we see that

$$\frac{1}{n}\sum_{k=1}^{n} x_k^2 \le \frac{\mathsf{P}}{2W} \tag{18.54}$$

i.e., in (18.48) we have

$$\mathsf{E}_{\mathrm{s}} = \frac{\mathsf{P}}{2W}. \tag{18.55}$$

Now, we recall from Section 17.3 that the (discrete-time) capacity per channel use of the Gaussian channel (18.45) under an average-power constraint E_{s} is

$$\mathsf{C}_{\mathrm{d}} = \frac{1}{2}\log_2\left(1+\frac{\mathsf{E}_{\mathrm{s}}}{\sigma^2}\right) \text{ bits per channel use,} \tag{18.56}$$

which in our situation, using (18.46) and (18.55), translates to

$$\mathsf{C}_{\mathrm{d}} = \frac{1}{2}\log_2\left(1+\frac{\frac{\mathsf{P}}{2W}}{\frac{\mathsf{N}_0}{2}}\right) = \frac{1}{2}\log_2\left(1+\frac{\mathsf{P}}{\mathsf{N}_0 W}\right) \text{ bits per sample.} \tag{18.57}$$

Since in total we have $2W\mathsf{T}$ samples, the total maximum amount of information that can be transmitted during the interval $[0,\mathsf{T}]$ is

$$2W\mathsf{T}\cdot\mathsf{C}_{\mathrm{d}} = W\mathsf{T}\log_2\left(1+\frac{\mathsf{P}}{\mathsf{N}_0 W}\right) \text{ bits.} \tag{18.58}$$

When normalized to T to describe the amount of information that can be transmitted per second, this finally gives

$$\mathsf{C} = W\log_2\left(1+\frac{\mathsf{P}}{\mathsf{N}_0 W}\right) \text{ bits/s.} \tag{18.59}$$

So we see that as long as the rate R of our coding scheme (Definition 18.2) is below the value C given in (18.59), we can choose the signal duration (observation duration) T long enough to make the error probability as small as wished. Once again we see the duality of the blocklength n in the discrete-time system and the signal duration T in the continuous-time system.

Note that by letting T tend to infinity, we also get rid of all the approximations and imprecisions in our derivation that we have so assiduously swept under the rug...

Hence, we have shown the following, of all his theorems, Shannon's most famous theorem.

Theorem 18.5 (Coding Theorem for the AWGN Channel).
Consider a continuous-time channel where the output $Y(\cdot)$ is given by

$$Y(t) = x(t) + Z(t) \tag{18.60}$$

with an input $x(\cdot)$ that is bandlimited to W Hz and that is constrained to have an average power of at most P, and with AWGN $Z(\cdot)$ of double-sided

spectral density $\frac{N_0}{2}$. The capacity of this channel is given by

$$C = W \log_2\left(1 + \frac{P}{N_0 W}\right) \text{ bits/s}. \tag{18.61}$$

We would like to briefly discuss this exciting result. Note that even though W shows up in the denominator inside the logarithm, it also shows up as a factor outside of the log. This factor outside is dominant for values of $W \ll P$. Moreover, the capacity expression is monotonically increasing in W. Its maximum, for $W \to \infty$, is

$$C = \frac{P}{N_0} \log_2 e \text{ bits/s}, \tag{18.62}$$

i.e., for infinite bandwidth the capacity grows linearly with the power. Hence, we see that usually it is more attractive to increase bandwidth rather than to increase power. Unfortunately, in practice bandwidth normally is far more expensive than power.

For this very same reason, the engineers are not so much interested in how much they can transmit per second, but rather how much they can transmit per *unit band*. The figure of interest then becomes $\frac{C}{W}$, describing how much rate (in bits/s) one achieves per Hz bandwidth. The capacity formula can then be written as

$$\frac{C}{W} = \log_2\left(1 + \frac{P}{N_0 W}\right) \frac{\text{bits}}{\text{s} \cdot \text{Hz}}. \tag{18.63}$$

The term $\frac{C}{W}$ is known as *maximum bandwidth efficiency*.

So consider some coding scheme using a bandwidth W that transmits at a rate of $R \le C$ bits/s. This scheme has a bandwidth efficiency of $\frac{R}{W}$. Suppose further that this coding scheme needs an energy E_b for every information bit it transmits, i.e., the total power P of this system is

$$P = R \cdot E_b. \tag{18.64}$$

Using this in (18.63) leads to

$$\frac{R}{W} \le \frac{C}{W} = \log_2\left(1 + \frac{R E_b}{N_0 W}\right) = \log_2\left(1 + \frac{E_b}{N_0} \cdot \frac{R}{W}\right), \tag{18.65}$$

i.e., this gives us a lower bound on the required *signal-to-noise ratio*[4] $\frac{E_b}{N_0}$ for the given bandwidth efficiency:

$$\frac{E_b}{N_0} \ge \frac{2^{\frac{R}{W}} - 1}{\frac{R}{W}}. \tag{18.66}$$

[4]The ratio $\frac{E_b}{N_0}$ is often referred to as *"ebno"* and is the most common definition for a signal-to-noise ratio (SNR); see (17.137).

Finally, returning once again to the maximum (at infinite bandwidth) capacity expression (18.62) and using (18.64), we obtain

$$\frac{E_b}{N_0} = \frac{C}{R} \ln 2. \tag{18.67}$$

Since any reliable system must have $R \leq C$, we see that

$$\gamma_b = \frac{E_b}{N_0} \geq \ln 2 = 0.693 = -1.6 \text{ dB}, \tag{18.68}$$

i.e., it is not possible to transmit reliably over the AWGN channel at an SNR smaller than -1.6 dB.

Chapter 19

Parallel Gaussian Channels

In this chapter we leave the continuous-time domain again and return to our discrete-time Gaussian channel model. In Chapter 17 we have already studied this model in detail. Now we would like to add one new ingredient: What happens if we have several Gaussian channels available at the same time? Of course, we can use them all at the same time, and we can simply divide our available power equally among all of them. But this might not be the best thing to do, in particular, if some of the Gaussian channels are better than other (i.e., they have less noise) or if the noise processes that impair the different signals on the different channels are actually correlated.

19.1 Channel Model

We consider J parallel Gaussian channels:

$$Y^{(j)} = x^{(j)} + Z^{(j)}, \quad j = 1, \ldots, \mathsf{J}, \tag{19.1}$$

where $x^{(j)}$ is the input to the jth channel and where $Z^{(j)}$ is additive Gaussian noise in the jth channel. The channel can actually be nicely written in vector form:

$$\mathbf{Y} = \mathbf{x} + \mathbf{Z} \tag{19.2}$$

with a vector-input $\mathbf{x} \in \mathbb{R}^{\mathsf{J}}$ and with the noise random vector $\mathbf{Z}$ being zero-mean Gaussian with some given nonsingular covariance matrix[1] $\mathsf{K}_{\mathbf{ZZ}}$:

$$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathsf{K}_{\mathbf{ZZ}}). \tag{19.3}$$

Note that in general we allow the noise vector to be correlated.

For every random message U, the transmitter now does not choose one, but J codewords. Hence, the codebook does not consist of 2^{nR} length-n codewords, but rather of 2^{nR} size-$(\mathsf{J} \times n)$ *code-matrices* $\mathsf{X}(m)$, $m = 1, \ldots, 2^{nR}$.

[1] For those readers who are worried about dependent Gaussian random vectors, we highly recommend to have a look at Appendices A and B.

We assume a total average-power constraint E, i.e., every code-matrix must satisfy

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{\mathsf{J}}\left(x_k^{(j)}(m)\right)^2 \le \mathsf{E}, \quad m=1,\dots,2^{n\mathsf{R}}, \tag{19.4}$$

or, in vector-form,

$$\frac{1}{n}\sum_{k=1}^{n}\left\|\mathbf{x}_k(m)\right\|^2 \le \mathsf{E}, \quad m=1,\dots,2^{n\mathsf{R}}. \tag{19.5}$$

The engineering meaning of this total average-power constraint is that the transmitter has (on average over time) a power E available that it is free to distribute among the J different channels.

We now ask the following question: What is the maximum total amount of information we can transmit reliably through these J parallel channels?

We will answer this question in a first step for the easier situation when the components of the noise vector are independent. The general case will then be treated in Section 19.4.

19.2 Independent Parallel Gaussian Channels

So, in this and the next section, we assume that the parallel channels are independent of each other, i.e., $\mathbf{Z}\sim\mathcal{N}\Big(\mathbf{0},\operatorname{diag}\Big(\sigma_1^2,\dots,\sigma_{\mathsf{J}}^2\Big)\Big)$. We can think of this setup as a transmitter that has J independent antennas or lines available that can be used at the same time. Note that the channels need not all have the same noise variance, i.e., some channels can be better (smaller variances) than others.

Since the channels are independent, a system that will just use these channels independently is obviously achievable. We only need to make sure that the average-power constraint is not violated. So, we split the available power E up into J parts E_j:

$$\mathsf{E}_j \ge 0, \quad j=1,\dots,\mathsf{J}, \tag{19.6a}$$

$$\sum_{j=1}^{\mathsf{J}}\mathsf{E}_j = \mathsf{E}, \tag{19.6b}$$

and then we use an independent system for each channel with the corresponding average power E_j. Since each of these channels is an individual Gaussian channel, we know from Theorem 17.9 that as long as the jth rate is below the jth channel's capacity, the probability of making an error on that channel can be made arbitrarily small. This is true for all channels, so by the Union Bound it holds that the total error probability can be made arbitrarily small.[2]

[2] In accordance with splitting up the power, we also need to split up the message into J submessages that are then transmitted over the corresponding channel.

Thus we have shown that a total rate satisfying

$$\mathsf{R} < \sum_{j=1}^{\mathsf{J}} \frac{1}{2} \log\Big(1 + \frac{\mathsf{E}_j}{\sigma_j^2}\Big) \tag{19.7}$$

is achievable.

To prove that this is indeed the best we can do, we rederive the converse of Section 17.3.3. The rate of any achievable coding scheme must satisfy

$$\begin{aligned}
\mathsf{R} &= \frac{1}{n}\,\mathrm{I}(M;\hat{M}) + \frac{1}{n}\,\mathrm{H}(M|\hat{M}) && (19.8)\\
&\le \frac{1}{n}\,\mathrm{I}(M;\hat{M}) + \frac{1}{n} + P_{\mathrm{e}}^{(n)}\mathsf{R} && (19.9)\\
&\le \frac{1}{n}\,\mathrm{I}(\mathbf{X}_1^n;\mathbf{Y}_1^n) + \frac{1}{n} + P_{\mathrm{e}}^{(n)}\mathsf{R}, && (19.10)
\end{aligned}$$

where the first inequality follows from the Fano Inequality and second inequality from the Data Processing Inequality.

We continue to bound the first term of (19.10) as follows:

$$\begin{aligned}
\frac{1}{n}\,\mathrm{I}(\mathbf{X}_1^n;\mathbf{Y}_1^n) &= \frac{1}{n}\,\mathrm{h}(\mathbf{Y}_1^n) - \frac{1}{n}\,\mathrm{h}(\mathbf{Y}_1^n|\mathbf{X}_1^n) && (19.11)\\
&= \frac{1}{n}\,\mathrm{h}(\mathbf{Y}_1^n) - \frac{1}{n}\,\mathrm{h}(\mathbf{Z}_1^n) && (19.12)\\
&= \frac{1}{n}\sum_{k=1}^{n}\Big(\mathrm{h}(\mathbf{Y}_k|\mathbf{Y}_1^{k-1}) - \mathrm{h}(\mathbf{Z}_k)\Big) && (19.13)\\
&\le \frac{1}{n}\sum_{k=1}^{n}\big(\mathrm{h}(\mathbf{Y}_k) - \mathrm{h}(\mathbf{Z}_k)\big) && (19.14)\\
&= \frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{\mathsf{J}}\Big(\mathrm{h}\Big(Y_k^{(j)}\Big|Y_k^{(1)},\dots,Y_k^{(j-1)}\Big) - \mathrm{h}\Big(Z_k^{(j)}\Big)\Big) && (19.15)\\
&\le \frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{\mathsf{J}}\Big(\mathrm{h}\Big(Y_k^{(j)}\Big) - \mathrm{h}\Big(Z_k^{(j)}\Big)\Big) && (19.16)\\
&= \sum_{j=1}^{\mathsf{J}}\frac{1}{n}\sum_{k=1}^{n}\Big(\mathrm{h}\Big(Y_k^{(j)}\Big) - \frac{1}{2}\log(2\pi e\sigma_j^2)\Big). && (19.17)
\end{aligned}$$

Here in (19.13) we use the chain rule and the fact that the Gaussian channel is memoryless; (19.14) follows because conditioning reduces entropy; the subsequent equality (19.15) follows again from the chain rule and the fact that the parallel Gaussian channels are independent; and the inequality in (19.16) holds again because conditioning reduces entropy.

Now, using the same ideas as in (17.116), we define

$$\mathsf{E}_{j,k} \triangleq \frac{1}{2^{n\mathsf{R}}}\sum_{m=1}^{2^{n\mathsf{R}}}\Big(x_k^{(j)}(m)\Big)^2 \tag{19.18}$$

to be the average energy of the (j,k)th component of the code-matrix when averaged uniformly over all code-matrices. Note that due to the average-power constraint (19.4) we have

$$\sum_{j=1}^{\mathrm{J}} \underbrace{\frac{1}{n}\sum_{k=1}^{n} \mathrm{E}_{j,k}}_{\triangleq \mathrm{E}_j} \le \mathrm{E}, \tag{19.19}$$

where the parameters E_j $(j=1,\ldots,\mathrm{J})$ have the same meaning as in (19.6). Moreover, identically to (17.118)–(17.122)

$$\begin{aligned}
\mathsf{Var}\big[Y_k^{(j)}\big] &= \mathsf{E}\left[\left(Y_k^{(j)}\right)^2\right] - \left(\mathsf{E}\left[Y_k^{(j)}\right]\right)^2 && (19.20)\\
&\le \mathsf{E}\left[\left(Y_k^{(j)}\right)^2\right] && (19.21)\\
&= \mathsf{E}\left[\left(X_k^{(j)} + Z_k^{(j)}\right)^2\right] && (19.22)\\
&= \mathsf{E}\left[\left(X_k^{(j)}\right)^2\right] + \mathsf{E}\left[\left(Z_k^{(j)}\right)^2\right] && (19.23)\\
&= \mathsf{E}\left[\left(x_k^{(j)}(M)\right)^2\right] + \sigma^2 && (19.24)\\
&= \sum_{m=1}^{2^{n\mathrm{R}}} \frac{1}{2^{n\mathrm{R}}}\left(x_k^{(j)}(m)\right)^2 + \sigma^2 && (19.25)\\
&= \mathrm{E}_{j,k} + \sigma^2. && (19.26)
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{1}{n}\,\mathrm{I}(\mathbf{X}_1^n;\mathbf{Y}_1^n) &\le \sum_{j=1}^{\mathrm{J}} \frac{1}{n}\sum_{k=1}^{n}\left(\mathrm{h}\big(Y_k^{(j)}\big) - \frac{1}{2}\log(2\pi e\sigma_j^2)\right) && (19.27)\\
&\le \sum_{j=1}^{\mathrm{J}} \frac{1}{n}\sum_{k=1}^{n}\left(\frac{1}{2}\log\Big(2\pi e\,\mathsf{Var}\Big[Y_k^{(j)}\Big]\Big) - \frac{1}{2}\log(2\pi e\sigma_j^2)\right) && (19.28)\\
&\le \sum_{j=1}^{\mathrm{J}} \frac{1}{n}\sum_{k=1}^{n}\left(\frac{1}{2}\log\big(2\pi e\,(\mathrm{E}_{j,k}+\sigma_j^2)\big) - \frac{1}{2}\log(2\pi e\sigma_j^2)\right) && (19.29)\\
&= \sum_{j=1}^{\mathrm{J}} \frac{1}{n}\sum_{k=1}^{n}\frac{1}{2}\log\left(1+\frac{\mathrm{E}_{j,k}}{\sigma_j^2}\right) && (19.30)\\
&\le \sum_{j=1}^{\mathrm{J}} \frac{1}{2}\log\left(1+\frac{1}{n}\sum_{k=1}^{n}\frac{\mathrm{E}_{j,k}}{\sigma_j^2}\right) && (19.31)\\
&= \sum_{j=1}^{\mathrm{J}} \frac{1}{2}\log\left(1+\frac{\mathrm{E}_j}{\sigma_j^2}\right), && (19.32)
\end{aligned}$$

where (19.31) follows from the Jensen Inequality because the logarithm is a concave function, and where in the last equality we have used the definition of E_j in (19.19).

Combining (19.32) with (19.10) and letting n tend to infinity, we see that indeed no coding scheme with $P_{\mathrm{e}}^{(n)}$ going to zero as $n \to \infty$ can have a rate larger than the right-hand side of (19.7).

So far, we have derived the following capacity formula for independent parallel Gaussian channels:

$$\mathrm{C}(\mathrm{E}) = \max_{\substack{\mathrm{E}_1,\ldots,\mathrm{E}_\mathrm{J}\\ \mathrm{E}_j\ge 0,\, j=1,\ldots,\mathrm{J}\\ \sum_{j=1}^{\mathrm{J}} \mathrm{E}_j \le \mathrm{E}}} \sum_{j=1}^{\mathrm{J}} \frac{1}{2}\log\left(1+\frac{\mathrm{E}_j}{\sigma_j^2}\right). \tag{19.33}$$

It remains to evaluate this expression, i.e., we need to figure out how to choose the power allocation E_j to the J different channels. This will be done in the next section.

19.3 Optimal Power Allocation: Waterfilling

While still considering the simpler case of independent channels, we now turn to the question on how to allocate the power E_j to the different channels, i.e., we need to evaluate the maximization in (19.33).

We start by pointing out that $\frac{1}{2}\log\left(1+\frac{\mathrm{E}_j}{\sigma_j^2}\right)$ is monotonically increasing in E_j. Therefore, we see that the maximum will be achieved for a choice of $(\mathrm{E}_1,\ldots,\mathrm{E}_\mathrm{J})$ such that

$$\sum_{j=1}^{\mathrm{J}} \mathrm{E}_j = \mathrm{E} \tag{19.34}$$

with equality.

Note further that $\frac{1}{2}\log\left(1+\frac{\mathrm{E}_j}{\sigma_j^2}\right)$ is a concave function and therefore that

$$g(\mathrm{E}_1,\ldots,\mathrm{E}_\mathrm{J}) \triangleq \sum_{j=1}^{\mathrm{J}} \frac{1}{2}\log\left(1+\frac{\mathrm{E}_j}{\sigma_j^2}\right) \tag{19.35}$$

is concave, too. We therefore see that (19.33) matches exactly the situation of the Karush–Kuhn–Tucker conditions[3] (Theorem 9.11). An optimal choice

[3] Strictly speaking we do not have an exact match: In Theorem 9.11 we optimize over a probability vector that sums to 1, while here we optimize over a vector $\mathbf{E} = (\mathrm{E}_1,\ldots,\mathrm{E}_\mathrm{J})^{\mathsf{T}}$ that sums to E. This, however, is only a question of normalization. We can divide $\mathbf{E}$ by E to make it look like a probability vector, and the normalization factor can then be incorporated into the parameter λ of (19.36).

of $(\mathsf{E}_1,\ldots,\mathsf{E}_\mathsf{J})$ must satisfy:

$$\frac{\partial g(\mathsf{E}_1,\ldots,\mathsf{E}_\mathsf{J})}{\partial \mathsf{E}_\ell} \stackrel{!}{=} \lambda \qquad \forall\, \ell \text{ with } \mathsf{E}_\ell > 0, \tag{19.36a}$$

$$\frac{\partial g(\mathsf{E}_1,\ldots,\mathsf{E}_\mathsf{J})}{\partial \mathsf{E}_\ell} \stackrel{!}{\le} \lambda \qquad \forall\, \ell \text{ with } \mathsf{E}_\ell = 0. \tag{19.36b}$$

So, we evaluate the Karush–Kuhn–Tucker conditions:

$$\frac{\partial}{\partial \mathsf{E}_\ell} \sum_{j=1}^{\mathsf{J}} \frac{1}{2} \log\left(1+\frac{\mathsf{E}_j}{\sigma_j^2}\right) = \frac{1}{2} \cdot \frac{1}{1+\frac{\mathsf{E}_\ell}{\sigma_\ell^2}} \cdot \frac{1}{\sigma_\ell^2} \cdot \log e = \frac{1}{2} \frac{\log e}{\mathsf{E}_\ell + \sigma_\ell^2}, \tag{19.37}$$

i.e.,

$$\frac{1}{2} \frac{\log e}{\mathsf{E}_\ell + \sigma_\ell^2} = \lambda \qquad \text{if } \mathsf{E}_\ell > 0, \tag{19.38a}$$

$$\frac{1}{2} \frac{\log e}{\mathsf{E}_\ell + \sigma_\ell^2} \le \lambda \qquad \text{if } \mathsf{E}_\ell = 0. \tag{19.38b}$$

By introducing a new parameter $\nu \triangleq \frac{\log e}{2\lambda}$, we can write this more simply as

$$\mathsf{E}_\ell = \nu - \sigma_\ell^2 \qquad \text{if } \mathsf{E}_\ell > 0, \tag{19.39a}$$

$$\mathsf{E}_\ell \ge \nu - \sigma_\ell^2 \qquad \text{if } \mathsf{E}_\ell = 0, \tag{19.39b}$$

which is equivalent to

$$\mathsf{E}_\ell = \nu - \sigma_\ell^2 \qquad \text{if } \nu - \sigma_\ell^2 > 0, \tag{19.40a}$$

$$\mathsf{E}_\ell = 0 \qquad \text{if } \nu - \sigma_\ell^2 \le 0, \tag{19.40b}$$

or, even simpler,

$$\boxed{\mathsf{E}_\ell = (\nu - \sigma_\ell^2)^+.} \tag{19.41}$$

Here ν must be chosen such that (19.34) is satisfied, i.e.,

$$\sum_{j=1}^{\mathsf{J}} (\nu - \sigma_j^2)^+ = \mathsf{E}. \tag{19.42}$$

This solution can be very nicely interpreted graphically (see Figure 19.1): we depict the various channels with bars representing the value of the noise variance, and we "pour" the available power into the graphic like water. The parameter ν then represents the water level. Hence, if we start with $\mathsf{E} = 0$ and slowly increase the available power, we will firstly only use the best channel (the one with the smallest noise variance). All other channels are switched off. However, because the logarithm is a concave function, the slope is decreasing with increasing power. Therefore, once we have increased E enough, the second best channel will yield a slope that is identical to the first and hence we start also pouring power into the second one. And so on.

This solution is therefore called the *waterfilling solution*.

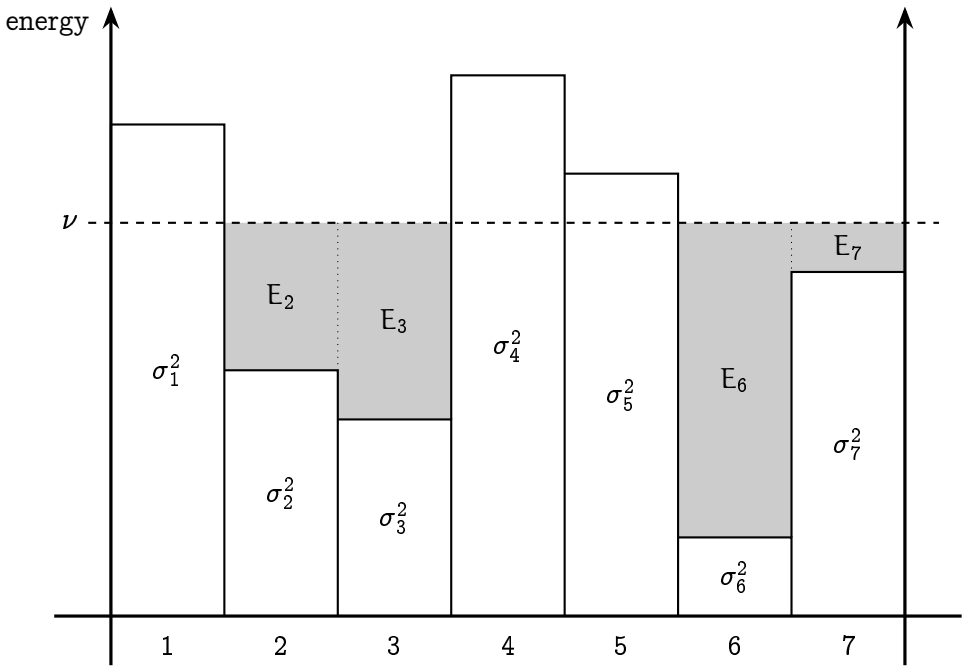


Figure 19.1: Waterfilling solution for the power allocation in J = 7 parallel independent Gaussian channels.

19.4 Dependent Parallel Gaussian Channels

We now turn to the general situation when the channels are dependent. It is actually possible to work out the optimal input analytically like we have done for the independent case in Sections 19.2 and 19.3, but the derivation is rather lengthy and complicated. We will follow here another approach that is simpler and that in the end also gives a better explanation of the meaning of the solution.

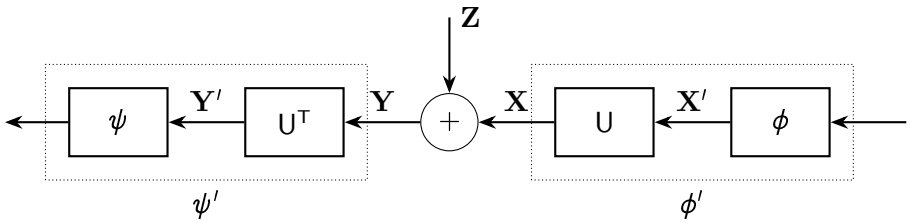


Figure 19.2: Parallel Gaussian channel with additional rotations.

Consider the parallel Gaussian channel shown in Figure 19.2 where the input vector $\mathbf{x}$ is a rotated version of the output of the encoder:

$$\mathbf{x} = \mathsf{U}\mathbf{x}' \tag{19.43}$$

for some orthogonal matrix U, and where the output vector $\mathbf{Y}$ is rotated back before it is fed to the decoder:

$$\mathbf{Y}' = \mathsf{U}^{\mathsf{T}}\mathbf{Y}. \tag{19.44}$$

First we note that because

$$\|\mathbf{x}\|^2 = \|\mathsf{U}\mathbf{x}'\|^2 = \|\mathbf{x}'\|^2 \tag{19.45}$$

(which holds because U is orthogonal, i.e., it corresponds to a rotation!), $\mathbf{x}$ satisfies the average-power constraint (19.5) if, and only if, $\mathbf{x}'$ satisfies the constraint.

Next, we observe that because $\mathsf{U}^{\mathsf{T}}\mathsf{U} = \mathsf{I}_{\mathsf{J}}$,

$$\begin{aligned}
\mathbf{Y}' &= \mathsf{U}^{\mathsf{T}}\mathbf{Y} && (19.46)\\
&= \mathsf{U}^{\mathsf{T}}(\mathbf{x}+\mathbf{Z}) && (19.47)\\
&= \mathsf{U}^{\mathsf{T}}(\mathsf{U}\mathbf{x}'+\mathbf{Z}) && (19.48)\\
&= \mathsf{U}^{\mathsf{T}}\mathsf{U}\mathbf{x}' + \mathsf{U}^{\mathsf{T}}\mathbf{Z} && (19.49)\\
&= \mathbf{x}' + \mathsf{U}^{\mathsf{T}}\mathbf{Z}. && (19.50)
\end{aligned}$$

Since according to Appendix B.5 and Lemma B.11,

$$\mathsf{U}^{\mathsf{T}}\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathsf{U}^{\mathsf{T}}\mathsf{K}_{\mathbf{ZZ}}\mathsf{U}), \tag{19.51}$$

it thus follows that the channel from $\mathbf{X}'$ to $\mathbf{Y}'$ is also a parallel Gaussian channel with an identical total average-power constraint, but with the noise having a different covariance matrix $\mathsf{U}^{\mathsf{T}}\mathsf{K}_{\mathbf{ZZ}}\mathsf{U}$.

Now note that any rate that is achievable from the channel $\mathbf{X}' \to \mathbf{Y}'$ is also achievable on the channel $\mathbf{X} \to \mathbf{Y}$ (simply define a new encoder ϕ' consisting of the original encoder ϕ with attached rotation U and a new decoder ψ' consisting of a rotation U^{T} before the original decoder ψ, and note that the performance of (ϕ', ψ') on $\mathbf{X} \to \mathbf{Y}$ is obviously identical to the performance of (ϕ, ψ) on $\mathbf{X}' \to \mathbf{Y}'$). Thus,

$$\mathsf{C}(\mathbf{X} \to \mathbf{Y}) \geq \mathsf{C}(\mathbf{X}' \to \mathbf{Y}'). \tag{19.52}$$

But because ϕ' and ψ' are not necessarily optimal for the channel $\mathbf{X} \to \mathbf{Y}$, we do not know if equality holds.

Let $\mathsf{C}(\mathsf{E}, \mathsf{K})$ denote the capacity of a parallel Gaussian channel with total average power E and with noise of covariance matrix K. From the discussion above (19.45)–(19.51) we know that

$$\mathsf{C}(\mathbf{X} \to \mathbf{Y}) = \mathsf{C}(\mathsf{E}, \mathsf{K}_{\mathbf{ZZ}}), \tag{19.53}$$

$$\mathsf{C}(\mathbf{X}' \to \mathbf{Y}') = \mathsf{C}(\mathsf{E}, \mathsf{U}^{\mathsf{T}}\mathsf{K}_{\mathbf{ZZ}}\mathsf{U}), \tag{19.54}$$

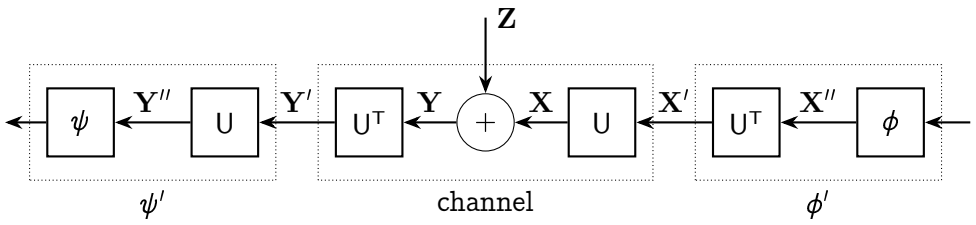


Figure 19.3: Setup of Figure 19.2 with one more rotation added. We now count the inner rotations as being part of a parallel Gaussian channel. Then the same arguments as before show that any rate achievable on $\mathbf{X}'' \to \mathbf{Y}''$ is also achievable on $\mathbf{X}' \to \mathbf{Y}'$.

i.e., (19.52) transforms into

$$\mathsf{C}(\mathsf{E}, \mathsf{K}_{\mathbf{ZZ}}) \geq \mathsf{C}(\mathsf{E}, \mathsf{U}^{\mathsf{T}} \mathsf{K}_{\mathbf{ZZ}} \mathsf{U}). \tag{19.55}$$

Repeating the exact same argument for the situation when a second rotation U^{T} is added between ϕ and U and the corresponding back rotation is added between U^{T} and ψ, i.e., $\mathbf{X}' = \mathsf{U}^{\mathsf{T}}\mathbf{X}''$ and $\mathbf{Y}'' = \mathsf{U}\mathbf{Y}'$ (see Figure 19.3), we obtain analogously to (19.52)

$$\mathsf{C}(\mathbf{X}' \to \mathbf{Y}') \geq \mathsf{C}(\mathbf{X}'' \to \mathbf{Y}''), \tag{19.56}$$

i.e.,

$$\begin{aligned} \mathsf{C}(\mathsf{E}, \mathsf{U}^{\mathsf{T}} \mathsf{K}_{\mathbf{ZZ}} \mathsf{U}) &\geq \mathsf{C}(\mathsf{E}, \mathsf{U}(\mathsf{U}^{\mathsf{T}} \mathsf{K}_{\mathbf{ZZ}} \mathsf{U})\mathsf{U}^{\mathsf{T}}) && (19.57)\\ &= \mathsf{C}(\mathsf{E}, \mathsf{K}_{\mathbf{ZZ}}). && (19.58) \end{aligned}$$

Thus, (19.58) and (19.55) combine to show that

$$\mathsf{C}(\mathsf{E}, \mathsf{K}_{\mathbf{ZZ}}) = \mathsf{C}(\mathsf{E}, \mathsf{U}^{\mathsf{T}} \mathsf{K}_{\mathbf{ZZ}} \mathsf{U}), \tag{19.59}$$

i.e., we indeed have equality and adding the rotation into the system does not affect its capacity!

This insight allows us now to quickly find the solution for the capacity of the dependent parallel Gaussian channel: The trick is to transform the channel into a different channel with independent noise (but with the same capacity!) and then to use the solution from Sections 19.2 and 19.3.

To this end, we note that for any nonsingular covariance matrix $\mathsf{K}_{\mathbf{ZZ}}$ there exists an orthogonal matrix U such that

$$\mathsf{K}_{\mathbf{ZZ}} = \mathsf{U}\Lambda\mathsf{U}^{\mathsf{T}} \tag{19.60}$$

where Λ is a diagonal matrix with positive entries on the diagonal (see Proposition B.4). By choosing U in Figure 19.2 as this particular matrix, we see

that (19.59) transforms to

$$\begin{aligned} C(E, K_{\mathbf{ZZ}}) &= C(E, U^{\mathsf{T}} K_{\mathbf{ZZ}} U) && (19.61)\\ &= C(E, U^{\mathsf{T}}(U\Lambda U^{\mathsf{T}})U) && (19.62)\\ &= C(E, \Lambda), && (19.63) \end{aligned}$$

thereby proving that the capacity of the dependent parallel Gaussian channel is identical to the capacity of a corresponding independent parallel Gaussian channel. An optimal solution is thus to first rotate the input so as to "whiten" the noise, then apply waterfilling on these now independent new channels, and finally rotate the solution back. Or to express it in terms of Figure 19.2: The optimal encoder consists of an optimal encoder for the corresponding independent parallel Gaussian channel in combination with the rotation U.

In mathematical terms, the optimal input is Gaussian

$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, K_{\mathbf{XX}}) \tag{19.64}$$

with a covariance matrix

$$K_{\mathbf{XX}} = U \operatorname{diag}(\tilde{E}_1, \ldots, \tilde{E}_J) U^{\mathsf{T}}, \tag{19.65}$$

where $\tilde{E}_1, \ldots, \tilde{E}_J$ is the optimal waterfilling solution to the corresponding independent parallel Gaussian channel with covariance matrix Λ:

$$\tilde{E}_j = (\nu - \lambda_j)^+ \tag{19.66}$$

such that

$$\sum_{j=1}^{J} \tilde{E}_j = E. \tag{19.67}$$

We end this section by pointing out that in a more general context the capacity of such vector channels under an average-power constraint (19.5) actually has a nice general form.

Theorem 19.1 (Coding Theorem for Vector Channels). The capacity of a memoryless vector channel under the total average-power constraint (19.5) (and under some technical constraint regarding the channel law) is given as

$$C(E) = \max_{f_{\mathbf{X}}:\ \mathsf{E}[\|\mathbf{X}\|^2] \le E} I(\mathbf{X}; \mathbf{Y}). \tag{19.68}$$

The proof follows actually similar lines as shown in Section 19.2, but one needs to be careful, e.g., when trying to apply the Jensen Inequality. In the case of a Gaussian channel we know the form of the capacity cost function $C(E)$, which simplifies things quite a bit. In general, the proof works out for sure if the capacity function given in (19.68) happens to be concave in the average power.

19.5 Colored Gaussian Noise

Instead of thinking of several parallel Gaussian channels that are dependent, we can use the same model (19.2), but interpret it differently as a single Gaussian channel that is dependent over time, i.e., that has memory. So this is a first step away from our memoryless models to a more practical situation where the noise depends on its past.

We consider the following version of a Gaussian channel

$$Y_k = x_k + Z_k, \tag{19.69}$$

where $\{Z_k\}$ is a Gaussian process with memory, i.e., with a certain given autocovariance function.[4] In the frequency domain this dependence is expressed by the fact that the *power spectral density (PSD)*[5] is not flat. Such a noise process is usually called *colored noise* in contrast to *white noise*.

We omit the mathematical details, but only quickly describe the result. The optimal solution again will apply waterfilling, however, this time not over different channels, but instead in the frequency domain into the PSD (see Figure 19.4).

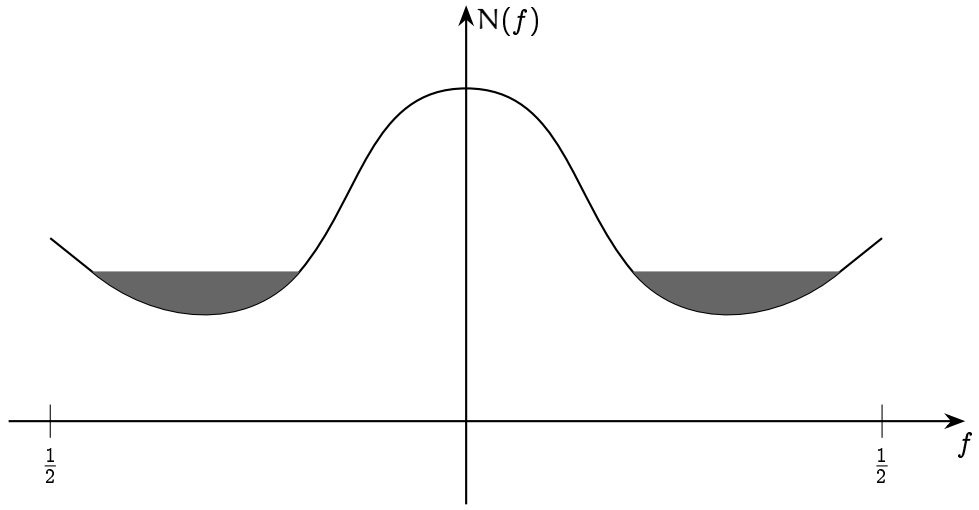


Figure 19.4: Power spectral density of colored noise with corresponding waterfilling solution.

[4] For a discussion about the differences between the autocovariance function and the autocorrelation function see Appendix C.

[5] Actually, as we are in a discrete-time setting, we should speak of *energy spectral density* as power is not even defined for discrete-time signals. However, it is customary to call this function PSD in spite of the fact that when integrating over it, one obtains the average energy of the signal, not its power. See also [Lap17, Chapter 13].

The capacity of such a colored Gaussian channel with PSD $\mathrm{N}(f)$ is given by

$$\mathrm{C}(\mathrm{E}) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{1}{2} \log\left(1 + \frac{\big(\nu - \mathrm{N}(f)\big)^+}{\mathrm{N}(f)}\right) \mathrm{d}f \tag{19.70}$$

with ν such that

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} \big(\nu - \mathrm{N}(f)\big)^+ \, \mathrm{d}f = \mathrm{E}. \tag{19.71}$$

In practice that would mean that we actually modulate our information in the frequency domain of the transmitted signal and choose a power distribution according to the waterfilling solution. Unfortunately, this then results in an input signal with very strong dependence over time, which is not very convenient for demodulation and decoding.

Instead, in practical systems a different approach is taken that is also used in the context of *fading channels*. In fading channels, the channel not only disturbs the input signal by adding additive noise (usually actually *white* Gaussian noise, not colored!), but it also filters the input by a linear filter, i.e., the input is convolved with some impulse response function $h(\cdot)$. In the frequency domain this means that input is multiplied by the corresponding transfer function, resulting in a situation similar to the case of colored noise of Figure 19.4: for different frequencies the input sees a channel of different quality. Compare with the discussion in Appendix C.6.

In order to handle this, the channel is now split up into many independent nonoverlapping narrow-band channels, which in a first approximation can be assumed to be white. This procedure is called *discrete multitone* and is the basic working principle of ADSL[6] modems: For practical purposes we also discretize the PSD, i.e., instead of the z-transform, we rely on the discrete Fourier transform (or rather its fast implementation *fast Fourier transform (FFT)*). The information is modulated onto each frequency sample where the used energy is determined by waterfilling.[7] This modulated discrete spectrum is then converted by the inverse discrete Fourier transform into a discrete-time signal, which goes — converted by a D/A-converter to a continuous-time signal — through the channel. On the receiver side, the received signal (after the A/D-converter) is transformed back into the frequency regime by the discrete Fourier transform. Each modulated frequency sample suffers from noise, however, because of the FFT, this noise is independent between each frequency sample, so in effect, we have J parallel independent channels (where

[6] ADSL stands for *asymmetric digital subscriber line*.

[7] In a practical system, actually not only the energy is adapted, but also the bit rate. This way the system tries to make sure that all transmitted symbols have a similar error probability.

J is the blocksize of the discrete Fourier transform) as discussed in Sections 19.2 and 19.3, and we can therefore decode each frequency sample independently.

So we see the similarities between these fading channels and the correlated parallel channels of Section 19.4: In Section 19.4, we encode the information onto $\tilde{\mathbf{X}}$ and use the matrix U to rotate it to get $\mathbf{X}$, which is then transmitted. The matrix U basically decouples the channels and makes the noise independent.

In the case of fading channels (or also colored noise channels), the fast Fourier transform takes over the role of decoupling: The FFT matrix makes sure that we get many independent parallel channels. Note that while U depends on the channel, the FFT works independently of the channel, thereby simplifying the system considerably.

We end this discussion by mentioning that *orthogonal frequency division multiplexing (OFDM)* — at the moment one of the most important modulation schemes used for wireless communication systems — is basically the same thing as discrete multitone, but applied to passband instead of baseband. So, it achieves the same goal: it creates many independent parallel channels. However, since in a wireless channel the transfer function changes often and quickly, one usually does not bother with waterfilling, but simply assigns the same energy to all subchannels.

Chapter 20

Asymptotic Equipartition Property and Weak Typicality

In this chapter we describe a very powerful tool that is used very often in information theory: *typicality*. The basis of typicality is the law of large numbers.

Since the concept is rather abstract, we have not used it so far, but actually, several of our main results in this class could have been proven using this tool. As a matter of fact, in the advanced information theory class [Mos22], we will make typicality its main tool.

20.1 Motivation

To explain the basic idea of typicality, we use a simple example. Consider an unfair coin X with PMF

$$P_X(x) = \begin{cases} 0.7 & \text{if } x = 1, \\ 0.3 & \text{if } x = 0, \end{cases} \tag{20.1}$$

such that $\mathrm{H}(X) \approx 0.881$ bits.

Now Mr. Unreliable claims that he has twice observed a length-10 sequence of this unfair coin, i.e., $(X_1, \ldots, X_{10})$ where X_i is IID $\sim P_X(\cdot)$. He asserts that he got the following two outputs:

$$1. \qquad (X_1, \ldots, X_{10}) = 1101011110; \tag{20.2}$$

$$2. \qquad (X_1, \ldots, X_{10}) = 1111111111. \tag{20.3}$$

Do we believe him?

Well, probably you will agree that the second event is somehow unexpected. Our life experience tells us that such a special outcome is not very likely and we rather expect to see a sequence like the first. This feeling gets even much stronger if the sequence is longer, e.g., has length 900 instead of only 10.

So far so good, but the whole thing becomes strange when we compute the probabilities of the two sequences:

$$1. \quad \Pr[(X_1,\ldots,X_{10}) = 1101011110] = 0.7^7 \cdot 0.3^3 \approx 0.22\%, \tag{20.4}$$

$$2. \quad \Pr[(X_1,\ldots,X_{10}) = 1111111111] = 0.7^{10} \approx 2.82\%. \tag{20.5}$$

Hence, the second sequence is about 13 times more likely than the first! So how could we think that the first is "more likely"?!?

To resolve this paradox, firstly note that we should not say that the first sequence is "more likely" (since it is not!), but rather that it is "more ordinary" or "more *typical*".

Secondly — and this is the clue! — we need to realize that even though the second sequence is much more likely than the first, the probability of observing a sequence *of the same type* as the first sequence (20.2) is far larger than the probability of the second sequence, indeed, it is close to 1. By the "the same type" we mean a sequence whose occurrence frequencies correspond to the PMF, i.e., in our case a sequence with about 70% ones and 30% zeros.

On the other hand, the probability of observing some special (nontypical) sequences like the all-one sequence or the all-zero sequence is close to 0. This becomes more and more pronounced if the length of the sequences is increased.

Since we usually see sequences like (20.2), we call these sequences *typical sequences*.

We can summarize this observation roughly as follows:

> When generating an IID length-n sequence, the probability of obtaining a typical sequence with approximately $nP_X(0)$ zeros and $nP_X(1)$ ones is almost equal to 1 for large n.

This fits well with our own experience in real life.

We will show below that we can state this observation differently in a slightly more abstract way as follows:[1]

$$\Pr\Big(\Big\{(x_1,\ldots,x_n)\colon P_{X_1,\ldots,X_n}(x_1,\ldots,x_n) \approx 2^{-n\,\mathrm{H}(X)}\Big\}\Big) \approx 1. \tag{20.6}$$

This means that we do not only almost for sure get a typical sequence, but that actually every typical sequence has almost the same probability!

> Almost certainly, we get a typical sequence. Among all typical sequences, the probability of one particular typical sequence is almost uniform.

[1] In this chapter we will follow the engineering notation and always assume that entropy is measured in bits. Hence we write $2^{-n\,\mathrm{H}(X)}$ instead of, e.g., $e^{-n\,\mathrm{H}(X)}$.

In the example above we have $2^{-10\,\mathrm{H}(X)} \approx 0.22\%$.

In the following sections we will make these ideas a bit more concrete.

20.2 Random Convergence

The definition of convergence of a deterministic sequence $a_1, a_2, \ldots$ to some value a is quite straightforward. Basically it says that for some n big enough, $a_n, a_{n+1}, \ldots$ will be so close to the convergence point that it will not leave a given small ϵ-environment around the convergence point. In more mathematical wording we say that a sequence $a_1, a_2, \ldots$ converges to a and write

$$\lim_{n\to\infty} a_n = a \tag{20.7}$$

if for any $\epsilon > 0$ there exists a constant N_ϵ such that for all $n \geq \mathrm{N}_\epsilon$

$$|a_n - a| \leq \epsilon. \tag{20.8}$$

Unfortunately, things are a bit more complicated if we look at the convergence of *random* sequences. Firstly, we might suspect that in general a random sequence will converge not to a deterministic value, but to a random variable.[2] Secondly, there are several different types of convergence.

Definition 20.1. We say that a sequence of random variables $X_1, X_2, \ldots$ *converges* to a random variable X

- *with probability 1* (also called *almost surely*), if
$$\Pr\Big[\lim_{n\to\infty} X_n = X\Big] = 1; \tag{20.9}$$
- *in probability*, if for any $\epsilon > 0$ we have $\Pr[|X_n - X| > \epsilon] \to 0$ as $n \to \infty$, i.e.,
$$\lim_{n\to\infty} \Pr[|X_n - X| \leq \epsilon] = 1; \tag{20.10}$$
- *in distribution*, if $P_{X_n}(x) \to P_X(x)$ for all x as $n \to \infty$, i.e.,
$$\lim_{n\to\infty} P_{X_n}(x) = P_X(x), \quad \forall\, x. \tag{20.11}$$

One can prove that these three types of convergence are in order: If a sequence converges with probability 1, then it will also converge in probability and in distribution; and if a sequence converges in probability, it also converges in distribution. The other way round is not always true. For more details we refer to probability theory textbooks.

Closely related to random convergence are the following two important inequalities.

[2] Luckily there is a large class of random sequences for which the converging random variable turns out to be deterministic. In particular this holds true in any situation where the weak law of large numbers can be applied.

Lemma 20.2 (Markov Inequality). A nonnegative random variable X of finite mean $\mathsf{E}[X] < \infty$ satisfies for any $a > 0$

$$\Pr[X \geq a] \leq \frac{\mathsf{E}[X]}{a}, \quad a > 0. \tag{20.12}$$

Proof: Fix some $a > 0$ and define

$$Y_a \triangleq \begin{cases} 0 & \text{if } X < a, \\ a & \text{if } X \geq a. \end{cases} \tag{20.13}$$

Since X is nonnegative by assumption, it follows that

$$Y_a \leq X \tag{20.14}$$

and therefore

$$\mathsf{E}[Y_a] \leq \mathsf{E}[X]. \tag{20.15}$$

On the other hand, we have

$$\mathsf{E}[Y_a] = \Pr[X < a] \cdot 0 + \Pr[X \geq a] \cdot a = a \Pr[X \geq a] \tag{20.16}$$

and therefore

$$a \Pr[X \geq a] \leq \mathsf{E}[X]. \tag{20.17}$$

The result now follows by dividing both sides by a. □

Lemma 20.3 (Chebyshev Inequality). A random variable X with finite mean μ and finite variance σ^2 satisfies for any $\epsilon > 0$

$$\Pr[|X - \mu| \geq \epsilon] \leq \frac{\sigma^2}{\epsilon^2}, \quad \epsilon > 0. \tag{20.18}$$

Proof: This follows directly from applying the Markov Inequality (Lemma 20.2) to $(X - \mu)^2$ with $a = \epsilon^2$. □

20.3 AEP

Equipped with this basic understanding of random convergence, we are now ready to state a very simple, but very powerful result, which is a direct consequence of the weak law of large numbers.

Theorem 20.4 (Asymptotic Equipartition Property (AEP)).
If $X_1, X_2, \ldots$ are IID $\sim P_X(\cdot)$, then

$$-\frac{1}{n}\log P_{X_1,\ldots,X_n}(X_1,\ldots,X_n) \overset{n\to\infty}{\longrightarrow} \mathrm{H}(X) \quad \text{in probability.} \tag{20.19}$$

We remind the reader that $P_{X_1,\ldots,X_n}(x_1,\ldots,x_n)$ denotes the joint probability distribution of $X_1,\ldots,X_n$.

Proof: Since functions of independent random variables are also independent random variables, and since all X_k are IID, we realize that $Y_k \triangleq \log P_X(X_k)$ are also IID. Moreover, we note that

$$P_{X_1,\ldots,X_n}(x_1,\ldots,x_n) = \prod_{k=1}^{n} P_X(x_k) \tag{20.20}$$

because all X_k are IID. Now recall the weak law of large numbers, which says that for a sequence $Z_1,\ldots,Z_n$ of IID random variables of mean μ and finite variance and for any $\epsilon > 0$,

$$\lim_{n\to\infty} \Pr\left[\left|\frac{Z_1+\cdots+Z_n}{n} - \mu\right| \geq \epsilon\right] = 0. \tag{20.21}$$

Hence,

$$\begin{aligned}
&\lim_{n\to\infty} -\frac{1}{n}\log P_{X_1,\ldots,X_n}(X_1,\ldots,X_n)\\
&\quad= \lim_{n\to\infty} -\frac{1}{n}\log \prod_{k=1}^{n} P_X(X_k) && (20.22)\\
&\quad= \lim_{n\to\infty} -\frac{1}{n}\sum_{k=1}^{n} \log P_X(X_k) && (20.23)\\
&\quad= \mathsf{E}[-\log P_X(X)] \quad \text{in probability} && (20.24)\\
&\quad= \mathrm{H}(X). && (20.25)
\end{aligned}$$

□

Remark 20.5. The AEP says that with high probability

$$P_{X_1,\ldots,X_n}(X_1,\ldots,X_n) \approx 2^{-n\,\mathrm{H}(X)} \tag{20.26}$$

(simply exponentiate both sides of (20.19)). Hence, for large n almost all sequences are about equally likely! This explains the name *asymptotic equipartition property*.

The AEP holds of course for any base of the logarithm. We will concentrate on $\log_2 \xi$, such that the inverse function is 2^{ξ}. △

20.4 Typical Set

Based on (20.26) we now give the following definition, that turns out to be extremely useful in proofs.

Definition 20.6. Fix $\epsilon > 0$, a length n, and a PMF $P_X(\cdot)$. Then the *typical set* $\mathcal{A}_\epsilon^{(n)}(P_X)$ with respect to $P_X(\cdot)$ is defined as the set of all those length-n sequences $(x_1, \dots, x_n) \in \mathcal{X}^n$ that have a probability close to $2^{-n\,\mathrm{H}(X)}$:

$$\mathcal{A}_\epsilon^{(n)}(P_X) \triangleq \Big\{(x_1, \dots, x_n) \in \mathcal{X}^n \colon 2^{-n(\mathrm{H}(X)+\epsilon)} \le P_{X_1,\dots,X_n}(x_1, \dots, x_n) \le 2^{-n(\mathrm{H}(X)-\epsilon)}\Big\}. \tag{20.27}$$

Example 20.7. For $n = 10$, $\epsilon = 0.01$, and $P_X(\cdot)$ as in (20.1), we get

$$\begin{aligned}(X_1, \dots, X_{10}) &\in \mathcal{A}_{0.01}^{(10)}(P_X)\\ &\iff 2.1 \cdot 10^{-3} \le P_{X_1,\dots,X_{10}}(X_1, \dots, X_{10}) \le 2.4 \cdot 10^{-3}.\end{aligned} \tag{20.28}$$

Hence, the first sequence (20.2) with probability

$$P_{X_1,\dots,X_{10}}(1101011110) \approx 2.22 \cdot 10^{-3} \tag{20.29}$$

is typical, while the second sequence (20.3) with probability

$$P_{X_1,\dots,X_{10}}(1111111111) \approx 28.2 \cdot 10^{-3} \tag{20.30}$$

is not typical. ◊

Theorem 20.8 (Properties of $\mathcal{A}_\epsilon^{(n)}(P_X)$).

1. If $(x_1, \dots, x_n) \in \mathcal{A}_\epsilon^{(n)}(P_X)$, then
$$\mathrm{H}(X) - \epsilon \le -\frac{1}{n} \log P_{X_1,\dots,X_n}(x_1, \dots, x_n) \le \mathrm{H}(X) + \epsilon. \tag{20.31}$$
2. If $X_1, \dots, X_n$ are IID $\sim P_X(\cdot)$, then
$$\Pr\big[(X_1, \dots, X_n) \in \mathcal{A}_\epsilon^{(n)}(P_X)\big] > 1 - \epsilon, \tag{20.32}$$
for n sufficiently large.
3. For all n, the size of the typical set is upper-bounded as follows:
$$\big|\mathcal{A}_\epsilon^{(n)}(P_X)\big| \le 2^{n(\mathrm{H}(X)+\epsilon)}. \tag{20.33}$$

4. For n sufficiently large, the size of the typical set is lower-bounded as follows:

$$\left|\mathcal{A}_\epsilon^{(n)}(P_X)\right| > (1-\epsilon)\,2^{n(\mathrm{H}(X)-\epsilon)}, \quad n \text{ sufficiently large.} \tag{20.34}$$

Proof: Part 1 follows trivially from the definition of $\mathcal{A}_\epsilon^{(n)}(P_X)$.

Part 2 follows directly from the AEP applied to the definition of $\mathcal{A}_\epsilon^{(n)}(P_X)$. To see this, we first rewrite the AEP in mathematical form. The statement (20.19) says that $\forall\,\delta > 0$ and $\forall\,\epsilon > 0$ there exists a threshold $\mathrm{N}_{\delta,\epsilon}$ such that for $n \geq \mathrm{N}_{\delta,\epsilon}$ we have

$$\Pr\Bigg[\underbrace{\left|-\frac{1}{n}\log P_{X_1,\ldots,X_n}(X_1,\ldots,X_n) - \mathrm{H}(X)\right| \leq \epsilon}_{\text{Note that by Part 1 any sequence} \in \mathcal{A}_\epsilon^{(n)}(P_X) \text{ satisfies this!}}\Bigg] > 1-\delta. \tag{20.35}$$

Now note that the expression inside of the probability in (20.35) actually corresponds exactly to the definition of the typical set $\mathcal{A}_\epsilon^{(n)}(P_X)$. Hence, written in a new form, the AEP says nothing else than

$$\Pr[(X_1,\ldots,X_n) \in \mathcal{A}_\epsilon^{(n)}(P_X)] > 1-\delta. \tag{20.36}$$

Since δ and ϵ are arbitrary, we can choose them freely. We choose $\delta \triangleq \epsilon$ and are done.

To derive Part 3 we simplify our notation and use a vector notation for sequences $\mathbf{X} = (X_1,\ldots,X_n)$.

$$\begin{aligned}
1 &= \sum_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X}}(\mathbf{x}) && && (20.37)\\
&\geq \sum_{\mathbf{x}\in\mathcal{A}_\epsilon^{(n)}(P_X)} P_{\mathbf{X}}(\mathbf{x}) && \text{(drop some terms)} && (20.38)\\
&\geq \sum_{\mathbf{x}\in\mathcal{A}_\epsilon^{(n)}(P_X)} 2^{-n(\mathrm{H}(X)+\epsilon)} && \text{(by definition of } \mathcal{A}_\epsilon^{(n)}(P_X)\text{)} && (20.39)\\
&= 2^{-n(\mathrm{H}(X)+\epsilon)} \cdot \left(\sum_{\mathbf{x}\in\mathcal{A}_\epsilon^{(n)}(P_X)} 1\right) && && (20.40)\\
&= 2^{-n(\mathrm{H}(X)+\epsilon)} \cdot \left|\mathcal{A}_\epsilon^{(n)}(P_X)\right|. && && (20.41)
\end{aligned}$$

Hence,

$$\left|\mathcal{A}_\epsilon^{(n)}(P_X)\right| \leq 2^{n(\mathrm{H}(X)+\epsilon)}. \tag{20.42}$$

Finally, Part 4 can be derived as follows. For n sufficiently large we have

$$\begin{aligned}
1-\epsilon &< \Pr[\mathbf{X}\in\mathcal{A}_\epsilon^{(n)}(P_X)] && \text{(by Part 2, for } n \text{ suff. large)} && (20.43)\\
&= \sum_{\mathbf{x}\in\mathcal{A}_\epsilon^{(n)}(P_X)} P_{\mathbf{X}}(\mathbf{x}) && && (20.44)\\
&\le \sum_{\mathbf{x}\in\mathcal{A}_\epsilon^{(n)}(P_X)} 2^{-n(\mathrm{H}(X)-\epsilon)} && \text{(by definition of } \mathcal{A}_\epsilon^{(n)}(P_X)) && (20.45)\\
&= \bigl|\mathcal{A}_\epsilon^{(n)}(P_X)\bigr|\cdot 2^{-n(\mathrm{H}(X)-\epsilon)}. && && (20.46)
\end{aligned}$$

Hence,

$$\bigl|\mathcal{A}_\epsilon^{(n)}(P_X)\bigr| > (1-\epsilon)\,2^{n(\mathrm{H}(X)-\epsilon)} \tag{20.47}$$

for n sufficiently large. □

Hence, we see that the typical set is a relatively small set (its size is only about $2^{n\,\mathrm{H}(X)}$, which is in contrast to the total of 2^n length-n sequences), but it contains almost all probability[3] $\Pr(\mathcal{A}_\epsilon^{(n)}(P_X)) > 1-\epsilon$.

20.5 High-Probability Sets and the Typical Set

We know that $\mathcal{A}_\epsilon^{(n)}(P_X)$ is small, but contains most of the probability. So the question arises if there exists some other set that is even smaller than $\mathcal{A}_\epsilon^{(n)}(P_X)$, but still contains most of the probability? In other words, does a set $\mathcal{B}_\delta^{(n)}$ exist such that

$$\bigl|\mathcal{B}_\delta^{(n)}\bigr| \ll \bigl|\mathcal{A}_\epsilon^{(n)}(P_X)\bigr| \tag{20.48}$$

and

$$\Pr[\mathbf{X}\in\mathcal{B}_\delta^{(n)}] > 1-\epsilon? \tag{20.49}$$

Note that $\mathcal{A}_\epsilon^{(n)}(P_X)$ is not the smallest such set. This can be seen, for example, by recalling that $\mathcal{A}_\epsilon^{(n)}(P_X)$ usually does not contain the most probable sequence. However, as we will see, the smallest set has essentially the same size as $\mathcal{A}_\epsilon^{(n)}(P_X)$.

In the following we will use the slightly sloppy, but common notation

$$\Pr(\mathcal{B}) \triangleq \Pr[(X_1,\dots,X_n)\in\mathcal{B}], \tag{20.50}$$

where we assume that $(X_1,\dots,X_n)$ are IID $\sim P_X(\cdot)$.

[3] In literature, one often finds the slightly sloppy notation $\Pr\bigl(\mathcal{A}_\epsilon^{(n)}(P_X)\bigr)$ instead of the more precise $\Pr\bigl[\mathbf{X}\in\mathcal{A}_\epsilon^{(n)}(P_X)\bigr]$.

Definition 20.9 (High-Probability Set). For any r-ary alphabet $\mathcal{X}$, any $\delta > 0$, and any positive integer n, let $\mathcal{B}_\delta^{(n)} \subset \mathcal{X}^n$ be some set of length-n sequences with

$$\Pr(\mathcal{B}_\delta^{(n)}) > 1 - \delta. \tag{20.51}$$

Particularly, we could choose $\mathcal{B}_\delta^{(n)}$ to be the smallest set such that (20.51) holds.

Theorem 20.10. Let $(X_1, X_2, \dots, X_n) \in \mathcal{X}^n$ be a random sequence chosen IID $\sim P_X(\cdot)$. For $0 < \delta < \frac{1}{2}$ and for any $\delta' > 0$ we then have

$$\frac{1}{n} \log \bigl|\mathcal{B}_\delta^{(n)}\bigr| > \mathrm{H}(X) - \delta' \tag{20.52}$$

for n sufficiently large.

Proof: Fix some $0 < \epsilon < \frac{1}{2}$ and consider the typical set $\mathcal{A}_\epsilon^{(n)}(P_X)$. We know that for n sufficiently large, we have

$$\Pr(\mathcal{A}_\epsilon^{(n)}(P_X)) > 1 - \epsilon. \tag{20.53}$$

Moreover, by assumption we have

$$\Pr(\mathcal{B}_\delta^{(n)}) > 1 - \delta. \tag{20.54}$$

Using

$$\Pr(\mathcal{A} \cup \mathcal{B}) = \Pr(\mathcal{A}) + \Pr(\mathcal{B}) - \Pr(\mathcal{A} \cap \mathcal{B}) \tag{20.55}$$

we obtain

$$\begin{aligned}
&\Pr\Bigl(\mathcal{A}_\epsilon^{(n)}(P_X) \cap \mathcal{B}_\delta^{(n)}\Bigr) \\
&\quad = \underbrace{\Pr(\mathcal{A}_\epsilon^{(n)}(P_X))}_{>1-\epsilon} + \underbrace{\Pr(\mathcal{B}_\delta^{(n)})}_{>1-\delta} - \underbrace{\Pr\Bigl(\mathcal{A}_\epsilon^{(n)}(P_X) \cup \mathcal{B}_\delta^{(n)}\Bigr)}_{\le 1} && (20.56)\\
&\quad > 1 - \epsilon + 1 - \delta - 1 && (20.57)\\
&\quad = 1 - \epsilon - \delta && (20.58)
\end{aligned}$$

for n sufficiently large. Hence,

$$\begin{aligned}
1 - \epsilon - \delta &< \Pr\Bigl(\mathcal{A}_\epsilon^{(n)}(P_X) \cap \mathcal{B}_\delta^{(n)}\Bigr) && (20.59)\\
&= \sum_{\mathbf{x} \in \mathcal{A}_\epsilon^{(n)}(P_X) \cap \mathcal{B}_\delta^{(n)}} P_{\mathbf{X}}(\mathbf{x}) && (20.60)\\
&\le \sum_{\mathbf{x} \in \mathcal{A}_\epsilon^{(n)}(P_X) \cap \mathcal{B}_\delta^{(n)}} 2^{-n(\mathrm{H}(X)-\epsilon)} && (20.61)\\
&= \bigl|\mathcal{A}_\epsilon^{(n)}(P_X) \cap \mathcal{B}_\delta^{(n)}\bigr| \cdot 2^{-n(\mathrm{H}(X)-\epsilon)} && (20.62)\\
&\le \bigl|\mathcal{B}_\delta^{(n)}\bigr| \cdot 2^{-n(\mathrm{H}(X)-\epsilon)}, && (20.63)
\end{aligned}$$

where the fist inequality (20.59) follows from (20.58); the subsequent equality (20.60) from (20.50); the subsequent inequality (20.61) from the fact that since $\mathbf{x} \in \mathcal{A}_\epsilon^{(n)}(P_X) \cap \mathcal{B}_\delta^{(n)}$ the sequence $\mathbf{x}$ must be element of $\mathcal{A}_\epsilon^{(n)}(P_X)$ and that therefore according to the definition of $\mathcal{A}_\epsilon^{(n)}(P_X)$

$$P_{\mathbf{X}}(\mathbf{x}) \le 2^{-n(\mathrm{H}(X)-\epsilon)}; \tag{20.64}$$

the subsequent equality (20.62) follows from counting the number of summands in the sum; and the final inequality (20.63) follows because the number of elements in $\mathcal{A}_\epsilon^{(n)}(P_X) \cap \mathcal{B}_\delta^{(n)}$ cannot be larger than the number of elements in $\mathcal{B}_\delta^{(n)}$.

Since it was assumed that $\epsilon < \frac{1}{2}$ and $\delta < \frac{1}{2}$ it follows that $1-\epsilon-\delta > 0$ and we can take the logarithm on both sides of (20.63):

$$\log_2\bigl|\mathcal{B}_\delta^{(n)}\bigr| > \log_2(1-\epsilon-\delta) + n(\mathrm{H}(X)-\epsilon), \tag{20.65}$$

from which follows that

$$\frac{1}{n}\log_2\bigl|\mathcal{B}_\delta^{(n)}\bigr| > \mathrm{H}(X) - \delta' \tag{20.66}$$

where $\delta' = \epsilon - \frac{1}{n}\log_2(1-\epsilon-\delta)$. Note that δ' can be made arbitrarily small by an appropriate choice of ϵ and of n large enough. □

Hence $\mathcal{B}_\delta^{(n)}$ has at least $2^{n(\mathrm{H}(X)-\delta')}$ elements. This means that $\mathcal{A}_\epsilon^{(n)}(P_X)$ has about the same size as $\mathcal{B}_\delta^{(n)}$, and hence also the same size as the smallest high-probability set.

20.6 Data Compression Revisited

(Reminder: In the following all logarithms have base 2, particularly $\mathrm{H}(U)$ is in bits!)

We now will see an example of very "typical" information theory: We use typical sets to prove our fundamental result about maximum lossless compression (that we know already). The trick is to design a special compression scheme that is not really practical, but very convenient for a proof.

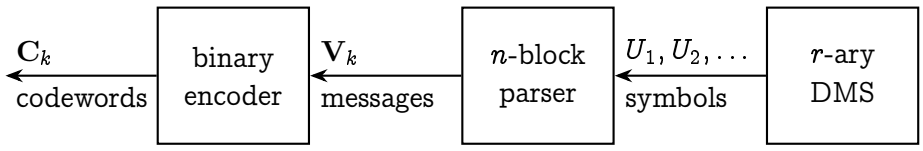


Figure 20.1: A binary block–to–variable-length source compression scheme.

We would like to compress the output sequence generated by an r-ary DMS with distribution P_U. We use an n-block parser, so that each message $\mathbf{V}$ is a sequence in $\mathcal{U}^n$. See Figure 20.1.

We now design a simple coding scheme. We start by grouping all possible message sequences into two sets: the typical messages that are member of $\mathcal{A}_\epsilon^{(n)}(P_U)$ and the nontypical messages that are not member of $\mathcal{A}_\epsilon^{(n)}(P_U)$. The coding for the two groups of messages will then be quite different:

- For each typical message $\mathbf{v} \in \mathcal{A}_\epsilon^{(n)}(P_U)$, we assign a distinct fixed-length binary codeword of a certain length l_1. Since

$$\left|\mathcal{A}_\epsilon^{(n)}(P_U)\right| \le 2^{n(\mathrm{H}(U)+\epsilon)}, \tag{20.67}$$

  it follows that a possible choice of l_1 is

$$\begin{aligned} l_1 &\triangleq \left\lceil \log_2 \left|\mathcal{A}_\epsilon^{(n)}(P_U)\right| \right\rceil && (20.68)\\ &\le \log_2 \left|\mathcal{A}_\epsilon^{(n)}(P_U)\right| + 1 && (20.69)\\ &\le n\big(\mathrm{H}(U)+\epsilon\big) + 1. && (20.70) \end{aligned}$$

- For each nontypical message $\mathbf{v} \notin \mathcal{A}_\epsilon^{(n)}(P_U)$, we also assign a distinct fixed-length binary codeword, but of a different length l_2. Since

$$\left|\mathcal{U}^n\right| = r^n, \tag{20.71}$$

  it follows that a possible choice of l_2 is

$$l_2 \triangleq \lceil \log_2 r^n \rceil \le n \log_2 r + 1. \tag{20.72}$$

Now we have two types of codewords, some are of length l_1 and some are of length l_2. In order to make sure that we can distinguish between them (i.e., to make sure that the code is prefix-free), we now put a 0 in front of all codewords of length l_1, and a 1 in front of all codewords of length l_2. So the first bit serves as flag that determines the length of the codeword. Hence, the length of the codeword for a message $\mathbf{v} \in \mathcal{U}^n$ satisfies

$$l(\mathbf{v}) \le \begin{cases} n\big(\mathrm{H}(U)+\epsilon\big)+2 & \text{if } \mathbf{v} \in \mathcal{A}_\epsilon^{(n)}(P_U),\\ n\log_2 r + 2 & \text{otherwise.} \end{cases} \tag{20.73}$$

Note that the typical messages are mapped to a short codeword, while the nontypical messages have long codewords. This seems to be a good compression scheme because we expect to see typical message most of the time. Moreover, from Theorem 20.8 we also know that all typical sequences have approximately the same probability, so it seems reasonable to assign codewords of identical length to them.

We next compute the expected codeword length:

$$\begin{aligned}
\mathsf{E}[l(\mathbf{V})] &= \sum_{\mathbf{v}\in\mathcal{U}^n} P_{\mathbf{V}}(\mathbf{v})\, l(\mathbf{v}) && (20.74)\\
&= \sum_{\mathbf{v}\in\mathcal{A}_\epsilon^{(n)}(P_U)} P_{\mathbf{V}}(\mathbf{v})\, l(\mathbf{v}) + \sum_{\mathbf{v}\notin\mathcal{A}_\epsilon^{(n)}(P_U)} P_{\mathbf{V}}(\mathbf{v})\, l(\mathbf{v}) && (20.75)\\
&\le \sum_{\mathbf{v}\in\mathcal{A}_\epsilon^{(n)}(P_U)} P_{\mathbf{V}}(\mathbf{v}) \Big(n\big(\mathrm{H}(U)+\epsilon\big)+2\Big)\\
&\quad + \sum_{\mathbf{v}\notin\mathcal{A}_\epsilon^{(n)}(P_U)} P_{\mathbf{V}}(\mathbf{v}) \left(n\log_2 r + 2\right) && (20.76)\\
&= \Big(n\big(\mathrm{H}(U)+\epsilon\big)+2\Big) \underbrace{\Pr\Big(\mathcal{A}_\epsilon^{(n)}(P_U)\Big)}_{\le 1}\\
&\quad + \left(n\log_2 r + 2\right) \underbrace{\Big(1-\Pr\Big(\mathcal{A}_\epsilon^{(n)}(P_U)\Big)\Big)}_{\substack{<\epsilon \text{ for } n \text{ large enough}\\ \text{by Theorem 20.8-2}}} && (20.77)\\
&< n\big(\mathrm{H}(U)+\epsilon\big)+2+\epsilon\cdot\left(n\log_2 r+2\right) && (20.78)\\
&= n\big(\mathrm{H}(U)+\epsilon'\big) && (20.79)
\end{aligned}$$

for n large enough, where

$$\epsilon' \triangleq \epsilon + \epsilon\log_2 r + \frac{2\epsilon+2}{n}. \tag{20.80}$$

Note that ϵ' can be made arbitrarily small by choosing ϵ small and n large enough.

So once again we have proven the following result that we know already from Chapter 5.

Theorem 20.11. For any $\epsilon > 0$, we can find a prefix-free binary code for an r-ary DMS U such that the average codeword length per source symbol satisfies

$$\frac{\mathsf{E}[L]}{n} \le \mathrm{H}(X) + \epsilon \tag{20.81}$$

for n sufficiently large. Here n is the message length of the n-block parser.

Moreover, also note that from the properties of typical sets we once again see that the output of a good data compression scheme will be almost uniformly distributed (compare with Remark 11.1): In order to make our scheme almost perfect, we choose an extremely small ϵ, e.g., $\epsilon = 10^{-1'000'000}$, and make the blocklength (parser length) n very large such that $\Pr(\mathcal{A}_\epsilon^{(n)}) > 1-\epsilon$ (Theorem 20.8 proves that this is possible!).

Now, basically all sequences that show up are typical and therefore all get assigned a codeword of the same length l_1. Moreover, we also know from

Theorem 20.8 that all typical sequences have (almost) the same probability $2^{-n\,\mathrm{H}(X)}$. Hence, we see that the output codewords of this compressing scheme are all equally likely and have the same length. Hence, the output digits indeed appear IID and uniformly distributed.

20.7 AEP for General Sources with Memory

So far the AEP and its consequences were based on the assumption that the letters of the sequences are chosen IID. This turns out to be too restrictive. There are many sources with memory that also satisfy the AEP.

Definition 20.12 (Source satisfying the AEP). A general source $\{U_k\}$ taking value in some discrete alphabet $\mathcal{U}$ is said to *satisfy the AEP* if the entropy rate exists,

$$\mathrm{H}(\{U_k\}) = \lim_{n\to\infty} \frac{1}{n}\,\mathrm{H}(U_1,\dots,U_n), \tag{20.82}$$

and if

$$-\frac{1}{n}\log P_{U_1,\dots,U_n}(U_1,\dots,U_n) \overset{n\to\infty}{\longrightarrow} \mathrm{H}(\{U_k\}) \quad \text{in probability.} \tag{20.83}$$

The class of sources that satisfy the AEP is rather large. Indeed the following holds.

Theorem 20.13 (Shannon–McMillan–Breiman Theorem). Any stationary and ergodic source taking value in a finite alphabet satisfies the AEP.

Proof: The proof is quite elaborate and therefore omitted. The interested reader is referred to, e.g., [CT06, pp. 644] or [AC88]. □

We can now generalize the definition of typical sets as follows.

Definition 20.14. For a given $\epsilon > 0$, a length n, and a source that satisfies the AEP $\{U_k\}$, the *typical set* $\mathcal{A}_\epsilon^{(n)}(\{U_k\})$ is defined as the set of all those length-n source output sequences $(u_1,\dots,u_n)$ that have a probability close to $2^{-n\,\mathrm{H}(\{U_k\})}$:

$$\mathcal{A}_\epsilon^{(n)}(\{U_k\}) \triangleq \Big\{(u_1,\dots,u_n)\colon 2^{-n(\mathrm{H}(\{U_k\})+\epsilon)} \le P_{U_1,\dots,U_n}(u_1,\dots,u_n) \le 2^{-n(\mathrm{H}(\{U_k\})-\epsilon)}\Big\}. \tag{20.84}$$

All the properties of Theorem 20.8 continue to hold. This is obvious because for the derivation of Theorem 20.8 only the AEP has been used. But in Definition 20.14, we have made it the main assumption that the AEP holds. We therefore omit the details of the proof, but only repeat the properties here.

Theorem 20.15.

1. If $(u_1, \dots, u_n) \in \mathcal{A}_\epsilon^{(n)}(\{U_k\})$, then

$$\mathrm{H}(\{U_k\}) - \epsilon \le -\frac{1}{n} \log P_{U_1,\dots,U_n}(u_1, \dots, u_n) \le \mathrm{H}(\{U_k\}) + \epsilon. \tag{20.85}$$

2. If $\{U_k\}$ satisfies the AEP, then

$$\Pr[(U_1, \dots, U_n) \in \mathcal{A}_\epsilon^{(n)}(\{U_k\})] > 1 - \epsilon, \tag{20.86}$$

   for n sufficiently large.

3. $\left|\mathcal{A}_\epsilon^{(n)}(\{U_k\})\right| \le 2^{n(\mathrm{H}(\{U_k\})+\epsilon)}$, for all n.

4. $\left|\mathcal{A}_\epsilon^{(n)}(\{U_k\})\right| > (1-\epsilon)\, 2^{n(\mathrm{H}(\{U_k\})-\epsilon)}$, for n sufficiently large.

20.8 General Source Coding Theorem

We next use our general definition of typical sets (Definition 20.14) to generalize the source coding theorem to general sources that satisfy the AEP. For simplicity, we only state it for the case of binary ($\mathrm{D} = 2$) codes. The extension to general D is straightforward.

We weaken our requirements for source coding by allowing the coding scheme to be *unsuccessful* for certain source output sequences, i.e., in certain cases the source sequence cannot be recovered from the codeword. The *success probability* is then defined as the probability that source generates a sequence that can be recovered successfully from its compressed version.

Specifically, we consider an encoding function

$$\phi_n \colon \mathcal{U}^n \to \{0,1\}^{\mathrm{K}} \tag{20.87}$$

that maps a source sequences $(U_1, \dots, U_n)$ into a binary codeword $\mathbf{C}$ of length K, and the corresponding decoding function

$$\psi_n \colon \{0,1\}^{\mathrm{K}} \to \mathcal{U}^n \tag{20.88}$$

that tries to recover the source sequence by mapping the codeword $\mathbf{C}$ into $(\hat{U}_1, \dots, \hat{U}_n)$. The efficiency of the coding scheme is given by its *coding rate*

$$\mathrm{R}_n \triangleq \frac{\mathrm{K}}{n} \text{ bits per source symbol.} \tag{20.89}$$

We can now prove the following general coding theorem.

Theorem 20.16 (General Source Coding Theorem for Sources satisfying the AEP). Consider a source $\{U_k\}$ that satisfies the AEP. Then for every $\epsilon > 0$, there exists a sequence of coding schemes (ϕ_n, ψ_n) with coding rate R_n satisfying

$$\lim_{n\to\infty} \mathrm{R}_n \le H(\{U_k\}) + \epsilon \tag{20.90}$$

such that the success probability tends to 1 as n tends to infinity.

Conversely, for any source $\{U_k\}$ that satisfies the AEP and any coding scheme (ϕ_n, ψ_n) with

$$\lim_{n\to\infty} \mathrm{R}_n < H(\{U_k\}), \tag{20.91}$$

the success probability must tend to 0 as n tends to infinity.

Proof: The direct part can been proven completely analogously to Section 20.6. For any source sequence that is typical, we assign a codeword of length l_1, while for any nontypical sequence we allow the system to be unsuccessful. We omit the details.

For the converse, suppose that we have a coding scheme with $\lim_{n\to\infty} \mathrm{R}_n < H(\{U_k\})$. Then there exists some $\epsilon > 0$ small enough and some n_0 large enough such that

$$\mathrm{R}_n < H(\{U_k\}) - 2\epsilon, \quad \forall\, n \ge n_0. \tag{20.92}$$

Using $\mathbf{U} \triangleq (U_1, \dots, U_n)$, we can now bound the performance of this coding scheme as follows:

$$\begin{aligned}
\Pr(\text{success}) &= \Pr\Big(\text{success and } \mathbf{U} \in \mathcal{A}_\epsilon^{(n)}(\{U_k\})\Big) \\
&\quad + \Pr\Big(\text{success and } \mathbf{U} \notin \mathcal{A}_\epsilon^{(n)}(\{U_k\})\Big) && (20.93)\\
&\le \Pr\Big(\text{success and } \mathbf{U} \in \mathcal{A}_\epsilon^{(n)}(\{U_k\})\Big) + \Pr\Big[\mathbf{U} \notin \mathcal{A}_\epsilon^{(n)}(\{U_k\})\Big] && (20.94)\\
&= \sum_{\substack{\mathbf{u}\in\mathcal{A}_\epsilon^{(n)}(\{U_k\})\\ \mathbf{u} \text{ is successful}}} \Pr[\mathbf{U}=\mathbf{u}] + \Pr\Big[\mathbf{U} \notin \mathcal{A}_\epsilon^{(n)}(\{U_k\})\Big] && (20.95)\\
&< \sum_{\substack{\mathbf{u}\in\mathcal{A}_\epsilon^{(n)}(\{U_k\})\\ \mathbf{u} \text{ is successful}}} 2^{-n(\mathrm{H}(\{U_k\})-\epsilon)} + \epsilon && (20.96)\\
&\le 2^{\mathrm{K}} \cdot 2^{-n(\mathrm{H}(\{U_k\})-\epsilon)} + \epsilon && (20.97)\\
&= 2^{n\mathrm{R}_n} \cdot 2^{-n(\mathrm{H}(\{U_k\})-\epsilon)} + \epsilon && (20.98)\\
&= 2^{-n(\mathrm{H}(\{U_k\})-\mathrm{R}_n-\epsilon)} + \epsilon && (20.99)\\
&< 2^{-n\epsilon} + \epsilon. && (20.100)
\end{aligned}$$

Here the inequality in (20.96) follows from (20.84) and from Theorem 20.15-2; the subsequent inequality (20.97) holds because under the condition that all sequences $\mathbf{u}$ are successful, each needs a distinct codeword, but there are only at most 2^K such binary sequences; and the last inequality (20.100) follows from (20.92). □

20.9 Joint AEP

We now return to the simpler situation of IID distributions. We next generalize the AEP (Theorem 20.4) to pairs of sequences $(\mathbf{X}, \mathbf{Y})$ where each letter pair (X_k, Y_k) is dependent, but the sequences are still IID over k.

Theorem 20.17 (Joint AEP).
If the sequence of pairs of RVs $(X_1, Y_1), (X_2, Y_2), \ldots$ is IID $\sim P_{X,Y}(\cdot,\cdot)$, then

$$-\frac{1}{n}\log P_{X_1,Y_1,\ldots,X_n,Y_n}(X_1, Y_1, \ldots, X_n, Y_n) \overset{n\to\infty}{\longrightarrow} \mathrm{H}(X,Y) \quad \text{in probability.} \tag{20.101}$$

Proof: We note that

$$P_{X_1,Y_1,\ldots,X_n,Y_n}(x_1, y_1, \ldots, x_n, y_n) = \prod_{k=1}^{n} P_{X,Y}(x_k, y_k). \tag{20.102}$$

Hence, by the weak law of large numbers,

$$\begin{aligned}
&\lim_{n\to\infty} -\frac{1}{n}\log P_{X_1,Y_1,\ldots,X_n,Y_n}(X_1, Y_1, \ldots, X_n, Y_n) \\
&\quad= \lim_{n\to\infty} -\frac{1}{n}\log \prod_{k=1}^{n} P_{X,Y}(X_k, Y_k) && (20.103)\\
&\quad= \lim_{n\to\infty} -\frac{1}{n}\sum_{k=1}^{n} \log P_{X,Y}(X_k, Y_k) && (20.104)\\
&\quad= \mathrm{E}[-\log P_{X,Y}(X,Y)] \quad \text{in probability} && (20.105)\\
&\quad= \mathrm{H}(X,Y). && (20.106)
\end{aligned}$$

□

20.10 Jointly Typical Sequences

Based on the joint AEP, we will also generalize the idea of typical sets to sets of pairs of sequences that are *jointly typical*.

Definition 20.18. The set $\mathcal{A}_\epsilon^{(n)}(P_{X,Y})$ of *jointly typical sequences* $(\mathbf{x},\mathbf{y})$ with respect to the joint distribution $P_{X,Y}(\cdot,\cdot)$ is defined as

$$\begin{aligned}\mathcal{A}_\epsilon^{(n)}(P_{X,Y}) \triangleq \Big\{&(\mathbf{x},\mathbf{y}) \in \mathcal{X}^n \times \mathcal{Y}^n\colon \\ &\left|-\frac{1}{n}\log P(\mathbf{x}) - \mathrm{H}(X)\right| \le \epsilon, \\ &\left|-\frac{1}{n}\log P(\mathbf{y}) - \mathrm{H}(Y)\right| \le \epsilon, \\ &\left|-\frac{1}{n}\log P(\mathbf{x},\mathbf{y}) - \mathrm{H}(X,Y)\right| \le \epsilon\Big\} \end{aligned} \tag{20.107}$$

where

$$P(\mathbf{x},\mathbf{y}) \triangleq \prod_{k=1}^{n} P_{X,Y}(x_k,y_k) \tag{20.108}$$

and where $P(\mathbf{x})$ and $P(\mathbf{y})$ are the corresponding marginal distributions.

Remark 20.19. Note that we require that if $\mathbf{x}$ and $\mathbf{y}$ are jointly typical, then by definition they are also typical by themselves. Unfortunately, this does not automatically[4] follow from the third condition in (20.107), which explains why we have to add the two first conditions to the definition of $\mathcal{A}_\epsilon^{(n)}(P_{X,Y})$.

Also note that if $\mathbf{x}$ and $\mathbf{y}$ both are typical by themselves, then this does not necessarily mean that they are also jointly typical. For this the additional condition

$$-\frac{1}{n}\log P(\mathbf{x},\mathbf{y}) \approx \mathrm{H}(X,Y) \tag{20.109}$$

has to be satisfied. △

Now we can derive similar properties of the jointly typical set $\mathcal{A}_\epsilon^{(n)}(P_{X,Y})$ as we did for the typical set $\mathcal{A}_\epsilon^{(n)}(P_X)$. Like for Theorem 20.4 and Theorem 20.8, the basic ingredient is again the weak law of large numbers.

Theorem 20.20 (Properties of $\mathcal{A}_\epsilon^{(n)}(P_{X,Y})$).
Let $(\mathbf{X},\mathbf{Y})$ be random vectors of length n that are drawn according to the distribution $P(\mathbf{x},\mathbf{y}) \triangleq \prod_{k=1}^{n} P_{X,Y}(x_k,y_k)$ (i.e., the component pairs are IID over k). Then we have the following:

1. $\Pr\Big[(\mathbf{X},\mathbf{Y}) \in \mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Big] = \Pr\Big(\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Big) > 1-\epsilon$, for n sufficiently large.
2. $\big|\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\big| \le 2^{n(\mathrm{H}(X,Y)+\epsilon)}$, for all n.

[4] In the situation of strong typicality (see [Mos22, Chapter 4],) this problem does not occur. There the condition for joint typicality directly implies that the sequences are also typical by themselves.

3. $\bigl|\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\bigr| > (1-\epsilon)\,2^{n(\mathrm{H}(X,Y)-\epsilon)}$, for n sufficiently large.
4. Assume $(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) \sim P_{\mathbf{X}} \cdot P_{\mathbf{Y}}$, i.e., the sequences $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ are independently drawn according to the marginals of (20.108). Then

$$\Pr\Bigl[(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) \in \mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Bigr] \le 2^{-n(\mathrm{I}(X;Y)-3\epsilon)}, \qquad \text{for all } n, \tag{20.110}$$

$$\Pr\Bigl[(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) \in \mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Bigr] > (1-\epsilon)\,2^{-n(\mathrm{I}(X;Y)+3\epsilon)}, \qquad \text{for } n \text{ sufficiently large.} \tag{20.111}$$

Proof: The proof is very similar to the proof of Theorem 20.8. The first part follows from the weak law of large numbers: Given $\epsilon > 0$, we know that

$$\exists\,\mathrm{N}_1 \text{ s.t. } \forall n \ge \mathrm{N}_1\colon \Pr\left[\left|-\frac{1}{n}\log P(\mathbf{X}) - \mathrm{H}(X)\right| > \epsilon\right] < \frac{\epsilon}{3}, \tag{20.112}$$

$$\exists\,\mathrm{N}_2 \text{ s.t. } \forall n \ge \mathrm{N}_2\colon \Pr\left[\left|-\frac{1}{n}\log P(\mathbf{Y}) - \mathrm{H}(Y)\right| > \epsilon\right] < \frac{\epsilon}{3}, \tag{20.113}$$

$$\exists\,\mathrm{N}_3 \text{ s.t. } \forall n \ge \mathrm{N}_3\colon \Pr\left[\left|-\frac{1}{n}\log P(\mathbf{X},\mathbf{Y}) - \mathrm{H}(X,Y)\right| > \epsilon\right] < \frac{\epsilon}{3}. \tag{20.114}$$

By the Union Bound, we hence see that for all $n \ge \max\{\mathrm{N}_1,\mathrm{N}_2,\mathrm{N}_3\}$,

$$\begin{aligned}
&1 - \Pr\Bigl(\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Bigr)\\
&\quad= \Pr\Biggl(\Bigl\{\mathbf{x}\colon \Bigl|-\frac{1}{n}\log P(\mathbf{x}) - \mathrm{H}(X)\Bigr| > \epsilon\Bigr\}\\
&\qquad\qquad \cup \Bigl\{\mathbf{y}\colon \Bigl|-\frac{1}{n}\log P(\mathbf{y}) - \mathrm{H}(Y)\Bigr| > \epsilon\Bigr\}\\
&\qquad\qquad \cup \Bigl\{(\mathbf{x},\mathbf{y})\colon \Bigl|-\frac{1}{n}\log P(\mathbf{x},\mathbf{y}) - \mathrm{H}(X,Y)\Bigr| > \epsilon\Bigr\}\Biggr) \qquad (20.115)\\
&\quad\le \Pr\left[\left|-\frac{1}{n}\log P(\mathbf{X}) - \mathrm{H}(X)\right| > \epsilon\right]\\
&\qquad + \Pr\left[\left|-\frac{1}{n}\log P(\mathbf{Y}) - \mathrm{H}(Y)\right| > \epsilon\right]\\
&\qquad + \Pr\left[\left|-\frac{1}{n}\log P(\mathbf{X},\mathbf{Y}) - \mathrm{H}(X,Y)\right| > \epsilon\right] \qquad (20.116)\\
&\quad< \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon. \qquad (20.117)
\end{aligned}$$

The second part can be derived as follows:

$$\begin{aligned}
1 &= \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{X}^n\times\mathcal{Y}^n} P(\mathbf{x},\mathbf{y}) \qquad (20.118)\\
&\ge \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{A}_\epsilon^{(n)}(P_{X,Y})} P(\mathbf{x},\mathbf{y}) \qquad (20.119)
\end{aligned}$$

$$\geq \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{A}_\epsilon^{(n)}(P_{X,Y})} 2^{-n(\mathrm{H}(X,Y)+\epsilon)} \tag{20.120}$$

$$= 2^{-n(\mathrm{H}(X,Y)+\epsilon)} \left|\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\right|, \tag{20.121}$$

where (20.120) follows from the third condition in the definition of the jointly typical set (Definition 20.18). Hence,

$$\left|\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\right| \leq 2^{n(\mathrm{H}(X,Y)+\epsilon)}. \tag{20.122}$$

For the third part, note that for n sufficiently large we have from Part 1

$$1-\epsilon < \Pr\Bigl(\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Bigr) \qquad \text{(for } n \text{ suff. large)} \tag{20.123}$$

$$= \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{A}_\epsilon^{(n)}(P_{X,Y})} P(\mathbf{x},\mathbf{y}) \tag{20.124}$$

$$\leq \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{A}_\epsilon^{(n)}(P_{X,Y})} 2^{-n(\mathrm{H}(X,Y)-\epsilon)} \tag{20.125}$$

$$= \left|\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\right| \cdot 2^{-n(\mathrm{H}(X,Y)-\epsilon)}, \tag{20.126}$$

where (20.125) follows again from the definition of $\mathcal{A}_\epsilon^{(n)}(P_{X,Y})$. Hence,

$$\left|\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\right| > (1-\epsilon)\, 2^{n(\mathrm{H}(X,Y)-\epsilon)} \tag{20.127}$$

for n sufficiently large.

Finally, the fourth part is the only new statement. It basically says that if we generate $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ independently, then the chance that they accidentally look like being jointly typical is very small.

To prove this, assume that $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ are independent, but have the same marginals as $\mathbf{X}$ and $\mathbf{Y}$, respectively. Then

$$\Pr\Bigl[(\tilde{\mathbf{X}},\tilde{\mathbf{Y}})\in\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Bigr] = \sum_{(\tilde{\mathbf{x}},\tilde{\mathbf{y}})\in\mathcal{A}_\epsilon^{(n)}(P_{X,Y})} P(\tilde{\mathbf{x}})\cdot P(\tilde{\mathbf{y}}) \tag{20.128}$$

$$\leq \sum_{(\tilde{\mathbf{x}},\tilde{\mathbf{y}})\in\mathcal{A}_\epsilon^{(n)}(P_{X,Y})} 2^{-n(\mathrm{H}(X)-\epsilon)}\cdot 2^{-n(\mathrm{H}(Y)-\epsilon)} \tag{20.129}$$

$$= \left|\mathcal{A}_\epsilon^{(n)}(P_{X,Y})\right| \cdot 2^{-n(\mathrm{H}(X)-\epsilon)-n(\mathrm{H}(Y)-\epsilon)} \tag{20.130}$$

$$\leq 2^{n(\mathrm{H}(X,Y)+\epsilon)}\cdot 2^{-n(\mathrm{H}(X)-\epsilon+\mathrm{H}(Y)-\epsilon)} \tag{20.131}$$

$$= 2^{-n(\mathrm{H}(X)+\mathrm{H}(Y)-\mathrm{H}(X,Y)-3\epsilon)} \tag{20.132}$$

$$= 2^{-n(\mathrm{I}(X;Y)-3\epsilon)}, \tag{20.133}$$

where (20.129) follows from the first two conditions of the definition of the jointly typical set (Definition 20.18), and where (20.131) follows from Part 2.

The lower bound can be derived in a similar fashion. □

Table 20.2: An example of a joint probability distribution.

$P_{X,Y}$	$Y=0$	$Y=1$	$Y=2$	P_X
$X=0$	$\frac{1}{2}$	0	$\frac{1}{4}$	$\frac{3}{4}$
$X=1$	0	$\frac{1}{4}$	0	$\frac{1}{4}$
P_Y	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	

Example 20.21. Consider a joint probability distribution $P_{X,Y}$ given in Table 20.2.

Consider now two sequences of length $n=10$:

$$\mathbf{x}=(0,1,0,0,0,1,1,0,0,0) \tag{20.134}$$

$$\mathbf{y}=(0,0,2,1,0,2,0,0,1,2). \tag{20.135}$$

We see immediately that the sequences look typical with respect to P_X and P_Y, respectively, however, they cannot have been generated jointly according to $P_{X,Y}$ because we have various position that are not possible: e.g., $(x,y)=(1,0)$ in the second position, or $(x,y)=(1,2)$ in position 6. So this pair of sequences is not jointly typical. ◊

To summarize we can say the following: There are about $2^{n\,\mathrm{H}(X)}$ typical X-sequences, $2^{n\,\mathrm{H}(Y)}$ typical Y-sequences, but only $2^{n\,\mathrm{H}(X,Y)}$ jointly typical sequences. Hence, the probability that a randomly chosen, but independent pair of an X- and a Y-sequence happens to look jointly typical is only about $2^{-n\,\mathrm{I}(X;Y)}$.

20.11 Data Transmission Revisited

The jointly typical set can be used to find a very elegant proof of the achievability part of the coding theorem (Theorem 11.34). To demonstrate this, we will now give a new version of the proof shown in Section 11.8.

Recall our IID random codebook construction based on a distribution $P_X(\cdot)$ (see (11.111)) and our encoder that, given the message m, simply transmits the mth codeword $\mathbf{X}(m)$ of $\mathscr{C}$ over the channel.

We will now design a new (also suboptimal) decoder: a so-called *typicality decoder*. This decoder is quite useless for a practical system, however, it allows a very elegant and simple analysis. It works as follows. After having received some $\mathbf{y}$, a typicality decoder looks for an $\tilde{m}\in\mathcal{M}$ such that

- $(\mathbf{X}(\tilde{m}),\mathbf{y})$ is jointly typical, and
- there is no other message $\tilde{m}'\neq\tilde{m}$ such that $(\mathbf{X}(\tilde{m}'),\mathbf{y})$ is jointly typical.

If the decoder can find such an $\tilde{m}$, then it will make the decision $\hat{m} = \tilde{m}$, otherwise it decides $\hat{m} = 0$ (which is equivalent to declaring an error).

We will now analyze the performance of our scheme. We can still (without loss of generality) assume that $M = 1$. Let

$$\mathcal{F}_m \triangleq \Big\{(\mathbf{X}(m), \mathbf{Y}) \in \mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Big\}, \quad m = 1, \ldots, 2^{n\mathrm{R}}, \tag{20.136}$$

be the event that the mth codeword and the received sequence $\mathbf{Y}$ are jointly typical. Hence an error occurs if $\mathcal{F}_1^{\mathrm{c}}$ occurs (i.e., if the transmitted codeword and the received sequence are not jointly typical) or if $\mathcal{F}_2 \cup \mathcal{F}_3 \cup \cdots \cup \mathcal{F}_{2^{n\mathrm{R}}}$ occurs (i.e., if one or more wrong codewords are jointly typical with the received sequence $\mathbf{Y}$). Hence, we can write

$$\begin{aligned}
\Pr(\text{error}) &= \Pr(\text{error} \,|\, M = 1) && (20.137)\\
&= \Pr(\mathcal{F}_1^{\mathrm{c}} \cup \mathcal{F}_2 \cup \mathcal{F}_3 \cup \cdots \cup \mathcal{F}_{2^{n\mathrm{R}}} \,|\, M = 1) && (20.138)\\
&\le \Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M = 1) + \sum_{m=2}^{2^{n\mathrm{R}}} \Pr(\mathcal{F}_m \,|\, M = 1) && (20.139)
\end{aligned}$$

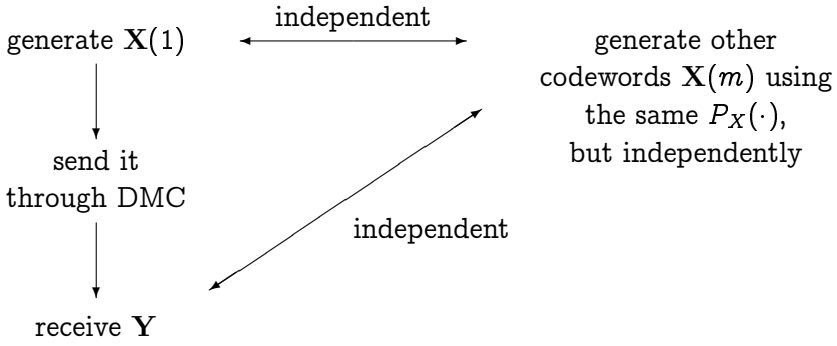
where the inequality follows from the Union Bound. From Theorem 20.17-1 we know that

$$\Pr\Big[(\mathbf{X}, \mathbf{Y}) \in \mathcal{A}_\epsilon^{(n)}(P_{X,Y})\Big] > 1 - \epsilon \tag{20.140}$$

i.e.,

$$\Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M = 1) < \epsilon \tag{20.141}$$

for n sufficiently large. Furthermore, we know that $\mathbf{Y}$ is completely independent of $\mathbf{X}(m)$ for all $m \ge 2$:



Hence, by Theorem 20.17-4 it now follows that for $m \ge 2$,

$$\begin{aligned}
\Pr(\mathcal{F}_m \,|\, M = 1) &= \Pr\Big((\mathbf{X}(m), \mathbf{Y}) \in \mathcal{A}_\epsilon^{(n)}(P_{X,Y}) \,\Big|\, M = 1\Big) && (20.142)\\
&\le 2^{-n(\mathrm{I}(X;Y) - 3\epsilon)} \qquad \forall\, n. && (20.143)
\end{aligned}$$

Combining these results with (20.139) then yields

$$\begin{aligned}
\Pr(\text{error}) &\le \Pr(\mathcal{F}_1^{\mathrm{c}} \,|\, M=1) + \sum_{m=2}^{2^{n\mathrm{R}}} \Pr(\mathcal{F}_m \,|\, M=1) && (20.144)\\
&< \epsilon + \sum_{m=2}^{2^{n\mathrm{R}}} 2^{-n(\mathrm{I}(X;Y)-3\epsilon)} && (20.145)\\
&= \epsilon + \underbrace{\bigl(2^{n\mathrm{R}}-1\bigr)}_{<2^{n\mathrm{R}}} 2^{-n(\mathrm{I}(X;Y)-3\epsilon)} && (20.146)\\
&< \epsilon + 2^{n\mathrm{R}} \cdot 2^{-n(\mathrm{I}(X;Y)-3\epsilon)} && (20.147)\\
&= \epsilon + 2^{-n(\mathrm{I}(X;Y)-\mathrm{R}-3\epsilon)} && (20.148)\\
&\le 2\epsilon && (20.149)
\end{aligned}$$

if n is sufficiently large and if $\mathrm{I}(X;Y)-\mathrm{R}-3\epsilon > 0$ so that the exponent is negative. Hence we see that as long as

$$\mathrm{R} < \mathrm{I}(X;Y) - 3\epsilon \tag{20.150}$$

for any $\epsilon > 0$ we can choose n such that the average error probability, averaged over all codewords and all codebooks, is less than 2ϵ:

$$\Pr(\text{error}) < 2\epsilon. \tag{20.151}$$

Note that while in Section 11.8 we relied on the weak law of large numbers to prove our result, here we used typicality. But since typicality is based on the AEP, which is simply another form of the weak law of large numbers, there is no fundamental difference between these two approaches.

20.12 Joint Source and Channel Coding Revisited

Finally, we also go back to the proof given in Section 15.3 and show another version that is based on typicality. So, we assume that the source $\{U_k\}$ satisfies the AEP. We fix an $\epsilon > 0$ and consider the set of typical sequences $\mathcal{A}_\epsilon^{(\mathrm{K})}(\{U_k\})$. Our system now works in two stages. In a first stage the encoder will look at a sequence of K source symbols and check whether it is typical or not. If it is typical, it assigns to it a unique number between 1 and M. If it is not typical, it does not really matter what the encoder does because the decoder will fail anyway. So we can assign any number in this case, e.g., we might choose to assign the number 1.

Note that according to Theorem 20.15-3 the total number of typical sequences is upper-bounded by

$$\mathrm{M} \le 2^{\mathrm{K}(\mathrm{H}(\{U_k\})+\epsilon)}. \tag{20.152}$$

Moreover, also note that all typical sequences are by definition (almost) equally likely: For every $\mathbf{u} \in \mathcal{A}_\epsilon^{(\mathsf{K})}(\{U_k\})$ we have

$$2^{-\mathsf{K}(\mathsf{H}(\{U_k\})+\epsilon)} \le \Pr[U_1^{\mathsf{K}} = \mathbf{u}] \le 2^{-\mathsf{K}(\mathsf{H}(\{U_k\})-\epsilon)}. \tag{20.153}$$

Hence, we can use a standard channel code (designed for a uniform source) to transmit these messages over the channel. So in the second stage, the encoder will assign a unique codeword of length n to each of the M possible messages. At the receiver side we use a joint-typicality decoder: Given the received sequence Y_1^n, the decoder looks for a unique codeword $\mathbf{x}$ that is jointly typical with Y_1^n. If it finds exactly one such codeword, it regenerates the corresponding (typical) source sequence u_1^{K}. If it finds none or more than one, it declares an error.

This system will make an error if either the source sequence happens to be nontypical or if the channel introduces a too strong error:

$$\begin{aligned}
P_{\mathsf{e}}^{(\mathsf{K})} &= \Pr[U_1^{\mathsf{K}} \neq \hat{U}_1^{\mathsf{K}}] && (20.154)\\
&= \Pr\Big[U_1^{\mathsf{K}} \notin \mathcal{A}_\epsilon^{(\mathsf{K})}\Big] \cdot \underbrace{\Pr\Big[U_1^{\mathsf{K}} \neq \hat{U}_1^{\mathsf{K}} \,\Big|\, U_1^{\mathsf{K}} \notin \mathcal{A}_\epsilon^{(\mathsf{K})}\Big]}_{=1} \\
&\quad + \underbrace{\Pr\Big[U_1^{\mathsf{K}} \in \mathcal{A}_\epsilon^{(\mathsf{K})}\Big]}_{\le 1} \cdot \Pr\Big[U_1^{\mathsf{K}} \neq \hat{U}_1^{\mathsf{K}} \,\Big|\, U_1^{\mathsf{K}} \in \mathcal{A}_\epsilon^{(\mathsf{K})}\Big] && (20.155)\\
&\le \Pr\Big[U_1^{\mathsf{K}} \notin \mathcal{A}_\epsilon^{(\mathsf{K})}\Big] + \Pr\Big[U_1^{\mathsf{K}} \neq \hat{U}_1^{\mathsf{K}} \,\Big|\, U_1^{\mathsf{K}} \in \mathcal{A}_\epsilon^{(\mathsf{K})}\Big] && (20.156)\\
&= \underbrace{\Pr\Big[U_1^{\mathsf{K}} \notin \mathcal{A}_\epsilon^{(\mathsf{K})}\Big]}_{\substack{\le \epsilon \text{ if } \mathsf{K} \text{ is}\\ \text{large enough}}} \\
&\quad + \underbrace{\Pr\Big[(X_1^n, Y_1^n) \notin \mathcal{A}_\epsilon^{(n)} \text{ or } (\tilde{X}_1^n, Y_1^n) \in \mathcal{A}_\epsilon^{(n)} \,\Big|\, U_1^{\mathsf{K}} \in \mathcal{A}_\epsilon^{(\mathsf{K})}\Big]}_{\le \epsilon \text{ if } \mathsf{R}<\mathsf{C} \text{ and } \mathsf{K} \text{ is large enough}} && (20.157)\\
&\le \epsilon + \epsilon = 2\epsilon \qquad (\text{if } \mathsf{R} < \mathsf{C} \text{ and } \mathsf{K} \text{ is large enough}). && (20.158)
\end{aligned}$$

Here in (20.158) the first upper bound follows from the properties of typical sequences, and the second upper bound follows from the channel coding theorem and holds as long as the rate of the channel code is less than the channel capacity.

The rate of our channel code can be bounded as follows:

$$\begin{aligned}
\mathsf{R} = \frac{\log_2 \mathsf{M}}{n} &\le \frac{\log_2 2^{\mathsf{K}(\mathsf{H}(\{U_k\})+\epsilon)}}{n} \qquad (\text{by (20.152)}) && (20.159)\\
&= \frac{\mathsf{K}}{n}(\mathsf{H}(\{U_k\}) + \epsilon) && (20.160)\\
&= \frac{\mathsf{T_c}}{\mathsf{T_s}}\,\mathsf{H}(\{U_k\}) + \frac{\mathsf{T_c}}{\mathsf{T_s}}\epsilon \qquad (\text{by (15.3)}) && (20.161)\\
&\le \frac{\mathsf{T_c}}{\mathsf{T_s}}\,\mathsf{H}(\{U_k\}) + \epsilon', && (20.162)
\end{aligned}$$

where ϵ' can be made arbitrarily small by making ϵ small.

Hence, we can guarantee that $\mathrm{R} < \mathrm{C}$ as long as

$$\frac{\mathrm{T_c}}{\mathrm{T_s}} \mathrm{H}(\{U_k\}) + \epsilon' < \mathrm{C}. \tag{20.163}$$

Since ϵ, and therefore ϵ', are arbitrary, our scheme will work if

$$\frac{\mathrm{H}(\{U_k\})}{\mathrm{T_s}} < \frac{\mathrm{C}}{\mathrm{T_c}}. \tag{20.164}$$

20.13 Typicality for Continuous Random Variables

The AEP generalizes directly to the situation of continuous random variables. Therefore, also all results concerning the Gaussian channel can be derived based on typicality.

Theorem 20.22 (AEP for Continuous Random Variables).
Let $X_1, \dots, X_n$ be a sequence of continuous random variables that are drawn IID $\sim f_X(\cdot)$. Then

$$-\frac{1}{n} \log f_{X_1,\dots,X_n}(X_1, \dots, X_n) \overset{n\to\infty}{\longrightarrow} \mathrm{E}[-\log f_X(X)] = \mathrm{h}(X) \quad \text{in probability.} \tag{20.165}$$

Proof: The proof relies as before on the weak law of large numbers:

$$\begin{aligned}
&\lim_{n\to\infty} -\frac{1}{n} \log f_{X_1,\dots,X_n}(X_1, \dots, X_n) \\
&\quad = \lim_{n\to\infty} -\frac{1}{n} \log \prod_{k=1}^{n} f_X(X_k) && (20.166)\\
&\quad = \lim_{n\to\infty} -\frac{1}{n} \sum_{k=1}^{n} \log f_X(X_k) && (20.167)\\
&\quad = \mathrm{E}[-\log f_X(X)] \quad \text{in probability} && (20.168)\\
&\quad = \mathrm{h}(X). && (20.169)
\end{aligned}$$

□

Definition 20.23. For $\epsilon > 0$ and any $n \in \mathbb{N}$ we define the *typical set* $\mathcal{A}_\epsilon^{(n)}(f_X)$ with respect to f_X as follows:

$$\mathcal{A}_\epsilon^{(n)}(f_X) \triangleq \left\{(x_1, \dots, x_n) \in \mathcal{X}^n \colon \left| -\frac{1}{n} \log f(x_1, \dots, x_n) - \mathrm{h}(X) \right| \le \epsilon \right\} \tag{20.170}$$

where

$$f(x_1, \dots, x_n) \triangleq \prod_{k=1}^{n} f_X(x_k). \tag{20.171}$$

Note that $\mathcal{A}_\epsilon^{(n)}(f_X)$ has in general (uncountably) infinitely many members because $\mathcal{X}$ is continuous. So it does not make sense to talk about the size of the set, but instead we define its volume.

Definition 20.24. The *volume* $\operatorname{Vol}(\mathcal{A})$ of a set $\mathcal{A} \subset \mathbb{R}^n$ is defined as

$$\operatorname{Vol}(\mathcal{A}) \triangleq \int\cdots\int_{\mathcal{A}} \mathrm{d}x_1 \cdots \mathrm{d}x_n. \tag{20.172}$$

Theorem 20.25. The typical set $\mathcal{A}_\epsilon^{(n)}(f_X)$ has the following properties:

1. $\Pr\Big(\mathcal{A}_\epsilon^{(n)}(f_X)\Big) > 1-\epsilon$, for n sufficiently large;
2. $\operatorname{Vol}\Big(\mathcal{A}_\epsilon^{(n)}(f_X)\Big) \le 2^{n(\mathrm{h}(X)+\epsilon)}$, for all n;
3. $\operatorname{Vol}\Big(\mathcal{A}_\epsilon^{(n)}(f_X)\Big) > (1-\epsilon)\,2^{n(\mathrm{h}(X)-\epsilon)}$, for n sufficiently large.

Proof: The proof is completely analogous to the proof of Theorem 20.8. Part 1 follows from the AEP: $\forall\,\epsilon,\delta>0$, $\exists\,\mathrm{N}_\delta$ such that for $n \ge \mathrm{N}_\delta$ we have

$$\Pr\left[\left|-\frac{1}{n}\log f(X_1,\dots,X_n) - \mathrm{h}(X)\right| \le \epsilon\right] > 1-\delta. \tag{20.173}$$

Now choose $\epsilon \triangleq \delta$.

Part 2 can be derived as follows:

$$\begin{aligned}
1 &= \int\cdots\int_{\mathcal{X}^n} f(x_1,\dots,x_n)\,\mathrm{d}x_1\cdots\mathrm{d}x_n && (20.174)\\
&\ge \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}} f(x_1,\dots,x_n)\,\mathrm{d}x_1\cdots\mathrm{d}x_n && (20.175)\\
&\ge \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}} 2^{-n(\mathrm{h}(X)+\epsilon)}\,\mathrm{d}x_1\cdots\mathrm{d}x_n && (20.176)\\
&= 2^{-n(\mathrm{h}(X)+\epsilon)} \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}} \mathrm{d}x_1\cdots\mathrm{d}x_n && (20.177)\\
&= 2^{-n(\mathrm{h}(X)+\epsilon)} \cdot \operatorname{Vol}\Big(\mathcal{A}_\epsilon^{(n)}\Big). && (20.178)
\end{aligned}$$

Here, in (20.175) we reduce the integration area (which reduces the value of the integral because the integrand is nonnegative); and (20.176) follows because by definition, every sequence in the typical set $\mathcal{A}_\epsilon^{(n)}$ satisfies

$$2^{-n(\mathrm{h}(X)+\epsilon)} \le f(x_1,\dots,x_n) \le 2^{-n(\mathrm{h}(X)-\epsilon)}. \tag{20.179}$$

Hence,

$$\operatorname{Vol}\Big(\mathcal{A}_\epsilon^{(n)}\Big) \le 2^{n(\mathrm{h}(X)+\epsilon)}. \tag{20.180}$$

Part 3 follows from Part 1: For n sufficiently large we have

$$\begin{aligned} 1-\epsilon &< \Pr\Big(\mathcal{A}_\epsilon^{(n)}\Big) && (20.181)\\ &= \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}} f(x_1,\dots,x_n)\,\mathrm{d}x_1\cdots\mathrm{d}x_n && (20.182)\\ &\le \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}} 2^{-n(\mathrm{h}(X)-\epsilon)}\,\mathrm{d}x_1\cdots\mathrm{d}x_n && (20.183)\\ &= 2^{-n(\mathrm{h}(X)-\epsilon)}\cdot \mathrm{Vol}\Big(\mathcal{A}_\epsilon^{(n)}\Big). && (20.184) \end{aligned}$$

Hence,

$$\mathrm{Vol}\Big(\mathcal{A}_\epsilon^{(n)}\Big) > (1-\epsilon)\,2^{n(\mathrm{h}(X)-\epsilon)} \tag{20.185}$$

for n sufficiently large. □

The generalization to the joint AEP for continuous RVs, jointly typical sets $\mathcal{A}_\epsilon^{(n)}(f_{X,Y})$, and their properties is completely analogous to Theorem 20.17, Definition 20.18, and Theorem 20.20, respectively.

Theorem 20.26 (Joint AEP for Continuous Random Variables). If the sequence of pairs of continuous RVs $(X_1,Y_1),(X_2,Y_2),\dots$ is IID $\sim f_{X,Y}(\cdot,\cdot)$, then

$$-\frac{1}{n}\log f_{X_1,Y_1,\dots,X_n,Y_n}(X_1,Y_1,\dots,X_n,Y_n) \overset{n\to\infty}{\longrightarrow} \mathrm{h}(X,Y) \quad \text{in probability.} \tag{20.186}$$

Proof: Since (X_k,Y_k) are IID and by the weak law of large numbers,

$$\begin{aligned} &\lim_{n\to\infty} -\frac{1}{n}\log f(X_1,Y_1,\dots,X_n,Y_n)\\ &\quad= \lim_{n\to\infty} -\frac{1}{n}\log\prod_{k=1}^n f_{X,Y}(X_k,Y_k) && (20.187)\\ &\quad= \lim_{n\to\infty} -\frac{1}{n}\sum_{k=1}^n \log f_{X,Y}(X_k,Y_k) && (20.188)\\ &\quad= \mathsf{E}[-\log f_{X,Y}(X,Y)] \quad \text{in probability} && (20.189)\\ &\quad= \mathrm{h}(X,Y). && (20.190) \end{aligned}$$

□

Definition 20.27. The set $\mathcal{A}_\epsilon^{(n)}(f_{X,Y})$ of *jointly typical sequences* $(\mathbf{x},\mathbf{y})$ with respect to the joint PDF $f_{X,Y}(\cdot,\cdot)$ is defined as

$$\begin{aligned} \mathcal{A}_\epsilon^{(n)}(f_{X,Y}) \triangleq \Bigg\{(\mathbf{x},\mathbf{y})\in\mathcal{X}^n\times\mathcal{Y}^n\colon &\left|-\frac{1}{n}\log f(\mathbf{x})-\mathrm{h}(X)\right|\le\epsilon,\\ &\left|-\frac{1}{n}\log f(\mathbf{y})-\mathrm{h}(Y)\right|\le\epsilon,\\ &\left|-\frac{1}{n}\log f(\mathbf{x},\mathbf{y})-\mathrm{h}(X,Y)\right|\le\epsilon\Bigg\} \end{aligned} \tag{20.191}$$

where

$$f(\mathbf{x},\mathbf{y}) \triangleq \prod_{k=1}^{n} f_{X,Y}(x_k,y_k) \tag{20.192}$$

and where $f(\mathbf{x})$ and $f(\mathbf{y})$ are the corresponding marginal PDFs.

Theorem 20.28 (Properties of $\mathcal{A}_\epsilon^{(n)}(f_{X,Y})$). Let $(\mathbf{X},\mathbf{Y})$ be random vectors of length n with joint PDF

$$f(\mathbf{x},\mathbf{y}) \triangleq \prod_{k=1}^{n} f_{X,Y}(x_k,y_k) \tag{20.193}$$

(i.e., $(\mathbf{X},\mathbf{Y})$ are IID over k). Then we have the following:

1. $\Pr\Big[(\mathbf{X},\mathbf{Y}) \in \mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big] = \Pr\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big) > 1-\epsilon$, for n sufficiently large.
2. $\mathrm{Vol}\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big) \le 2^{n(\mathrm{h}(X,Y)+\epsilon)}$, for all n.
3. $\mathrm{Vol}\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big) > (1-\epsilon)\,2^{n(\mathrm{h}(X,Y)-\epsilon)}$, for n sufficiently large.
4. Assume $(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) \sim f_{\mathbf{X}}\cdot f_{\mathbf{Y}}$, i.e., the sequences $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ are independently drawn according to the marginals of $f_{\mathbf{X},\mathbf{Y}}$. Then

$$\Pr\Big[(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) \in \mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big] \le 2^{-n(\mathrm{I}(X;Y)-3\epsilon)}, \qquad \text{for all } n, \tag{20.194}$$

$$\Pr\Big[(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) \in \mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big] > (1-\epsilon)\,2^{-n(\mathrm{I}(X;Y)+3\epsilon)}, \qquad \text{for } n \text{ sufficiently large.} \tag{20.195}$$

Proof: By the weak law of large numbers and for a given $\epsilon > 0$, we know that

$$\exists\,\mathrm{N}_1 \text{ s.t. } \forall n \ge \mathrm{N}_1\colon \Pr\left[\left|-\frac{1}{n}\log f(\mathbf{X}) - \mathrm{h}(X)\right| > \epsilon\right] < \frac{\epsilon}{3}, \tag{20.196}$$

$$\exists\,\mathrm{N}_2 \text{ s.t. } \forall n \ge \mathrm{N}_2\colon \Pr\left[\left|-\frac{1}{n}\log f(\mathbf{Y}) - \mathrm{h}(Y)\right| > \epsilon\right] < \frac{\epsilon}{3}, \tag{20.197}$$

$$\exists\,\mathrm{N}_3 \text{ s.t. } \forall n \ge \mathrm{N}_3\colon \Pr\left[\left|-\frac{1}{n}\log f(\mathbf{X},\mathbf{Y}) - \mathrm{h}(X,Y)\right| > \epsilon\right] < \frac{\epsilon}{3}. \tag{20.198}$$

Thus, by the Union Bound, we have for all $n \ge \max\{\mathrm{N}_1,\mathrm{N}_2,\mathrm{N}_3\}$,

$$\begin{aligned}&1-\Pr\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big)\\ &\quad= \Pr\left(\left\{\mathbf{x}\colon \left|-\frac{1}{n}\log f(\mathbf{x}) - \mathrm{h}(X)\right| > \epsilon\right\}\right.\end{aligned}$$

$$\cup \left\{\mathbf{y}\colon \left|-\frac{1}{n}\log f(\mathbf{y}) - \mathrm{h}(Y)\right| > \epsilon\right\}$$
$$\cup \left\{(\mathbf{x},\mathbf{y})\colon \left|-\frac{1}{n}\log f(\mathbf{x},\mathbf{y}) - \mathrm{h}(X,Y)\right| > \epsilon\right\}\Bigg) \tag{20.199}$$
$$\le \Pr\left[\left|-\frac{1}{n}\log f(\mathbf{X}) - \mathrm{h}(X)\right| > \epsilon\right]$$
$$+ \Pr\left[\left|-\frac{1}{n}\log f(\mathbf{Y}) - \mathrm{h}(Y)\right| > \epsilon\right]$$
$$+ \Pr\left[\left|-\frac{1}{n}\log f(\mathbf{X},\mathbf{Y}) - \mathrm{h}(X,Y)\right| > \epsilon\right] \tag{20.200}$$
$$< \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon. \tag{20.201}$$

The second part follows because

$$1 = \int\cdots\int_{\mathcal{X}^n\times\mathcal{Y}^n} f(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \tag{20.202}$$
$$\ge \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}(f_{X,Y})} f(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \tag{20.203}$$
$$\ge \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}(f_{X,Y})} 2^{-n(\mathrm{h}(X,Y)+\epsilon)}\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \tag{20.204}$$
$$= 2^{-n(\mathrm{h}(X,Y)+\epsilon)}\,\mathrm{Vol}\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big), \tag{20.205}$$

where (20.204) follows from the third condition in (20.191).

The third part holds because for n sufficiently large we have from Part 1

$$1-\epsilon < \Pr\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big) \qquad \text{(for } n \text{ suff. large)} \tag{20.206}$$
$$= \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}(f_{X,Y})} f(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \tag{20.207}$$
$$\le \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}(f_{X,Y})} 2^{-n(\mathrm{h}(X,Y)-\epsilon)}\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \tag{20.208}$$
$$= \mathrm{Vol}\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big)\cdot 2^{-n(\mathrm{h}(X,Y)-\epsilon)}. \tag{20.209}$$

Finally, to prove the fourth part, assume that $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ are independent, but have the same marginals as $\mathbf{X}$ and $\mathbf{Y}$, respectively. Then

$$\Pr\Big[(\tilde{\mathbf{X}},\tilde{\mathbf{Y}}) \in \mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big]$$
$$= \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}(f_{X,Y})} f(\tilde{\mathbf{x}})\cdot f(\tilde{\mathbf{y}})\,\mathrm{d}\tilde{\mathbf{y}}\,\mathrm{d}\tilde{\mathbf{x}} \tag{20.210}$$
$$\le \int\cdots\int_{\mathcal{A}_\epsilon^{(n)}(f_{X,Y})} 2^{-n(\mathrm{h}(X)-\epsilon)}\cdot 2^{-n(\mathrm{h}(Y)-\epsilon)}\,\mathrm{d}\tilde{\mathbf{y}}\,\mathrm{d}\tilde{\mathbf{x}} \tag{20.211}$$
$$= \mathrm{Vol}\Big(\mathcal{A}_\epsilon^{(n)}(f_{X,Y})\Big)\cdot 2^{-n(\mathrm{h}(X)-\epsilon)-n(\mathrm{h}(Y)-\epsilon)} \tag{20.212}$$
$$\le 2^{n(\mathrm{h}(X,Y)+\epsilon)}\cdot 2^{-n(\mathrm{h}(X)-\epsilon+\mathrm{h}(Y)-\epsilon)} \tag{20.213}$$
$$= 2^{-n(\mathrm{h}(X)+\mathrm{h}(Y)-\mathrm{h}(X,Y)-3\epsilon)} \tag{20.214}$$
$$= 2^{-n(\mathrm{I}(X;Y)-3\epsilon)}, \tag{20.215}$$

where (20.211) follows from the first two conditions in (20.191), and where (20.213) follows from Part 2.

The lower bound can be derived accordingly. □

20.14 Summary

Data Compression (based on AEP): There exists a small subset of all possible source sequences of size $2^{n\,\mathrm{H}}$ that contains almost all probability. Hence, we can represent a source with H bits per symbol with a very small probability of error.

Data Transmission (based on joint AEP): The output sequence is very likely to be jointly typical with the transmitted codeword, and the probability that it is jointly typical with any other codeword is $2^{-n\,\mathrm{I}}$. Hence we can use about $2^{n\,\mathrm{I}}$ codewords and still have a very small probability of error.

Source Channel Separation Theorem: We can design a source code and a channel code separately and still be optimal.

Chapter 21

Cryptography

21.1 Introduction to Cryptography

Cryptography is Greek and means *hidden writing*. It is the art of transmitting a message in hidden form such that only the intended recipient will be able to read it. There are many ways of trying to achieve this goal. A message could be hidden such that only a person who knows where to look for it will find it. Or the recipient needs to know a certain decoding scheme in order to make sense into a scrambled text. Or the recipient needs to know or possess a key to unlock an encrypted message.

Cryptography has been very important for a long time in history, simply because information and knowledge often means power and money. Some typical examples of how cryptography used to work before it became a science are as follows:

- The message is written in invisible ink that can be made visible by adding a certain chemical or heat.
- The message is tattooed into the skin on the head of a man, then the man waits until his hair has grown back before he travels to the required destination. There he shaves the hair off again to reveal the message.
- The message is written row-wise into a square grid and then read column-wise to scramble it.

Cryptography has two very different goals that are often mixed up:

- A cryptographic system provides *secrecy* if it determines who can *receive* a message.
- A cryptographic system provides *authenticity* if it determines who can have *sent* a message.

As a matter of fact, it was only recently in the long history of cryptography that people have started to appreciate this fundamental difference.

In the following we will give a very brief overview of some main ideas behind cryptographic systems. We start with Shannon who was the first person to look at cryptography in a proper scientific way. Basically, he changed cryptography from being an art to become a "scientific art".

21.2 Cryptographic System Model

In 1949, Claude E. Shannon published a paper called "Communication Theory of Secrecy Systems" [Sha49], which gave the first truly scientific treatment of cryptography. Shannon had been working on this topic due to the second world war and the insights he gained caused him later on to develop information theory. However, after the war his work on cryptography was still declared confident, and so he first published his work on data compression and transmission [Sha48]. So usually people firstly refer to the latter and only then to cryptography, but it actually should be the other way around: Cryptography really started the whole area of information theory.

In his 1949 paper, Shannon started by providing the fundamental way of how one should look at a cryptographic system. His system is shown in Figure 21.1. Actually, Shannon did not yet consider the public and private

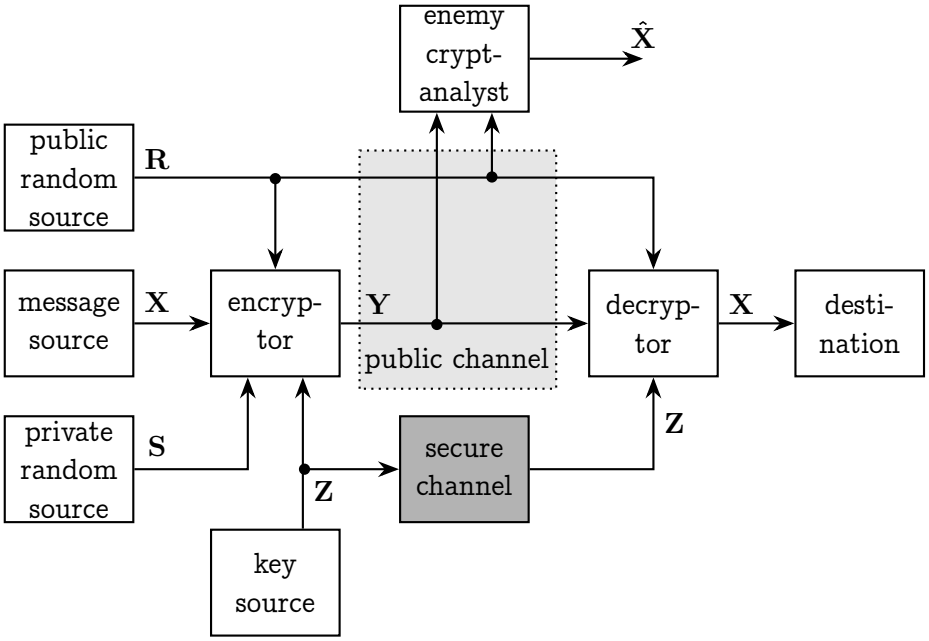


Figure 21.1: System model of a cryptographic system the way Shannon defined it.

random source, but their inclusion is straightforward. In this system model,

we have the following quantities:

$$\text{plaintext:} \quad \mathbf{X} = (X_1, X_2, \ldots, X_{\mathrm{J}}) \tag{21.1}$$
$$\text{ciphertext:} \quad \mathbf{Y} = (Y_1, Y_2, \ldots, Y_n) \tag{21.2}$$
$$\text{secret key:} \quad \mathbf{Z} = (Z_1, Z_2, \ldots, Z_{\ell_z}) \tag{21.3}$$
$$\text{public randomization:} \quad \mathbf{R} = (R_1, R_2, \ldots, R_{\ell_r}) \tag{21.4}$$
$$\text{private randomization:} \quad \mathbf{S} = (S_1, S_2, \ldots, S_{\ell_s}) \tag{21.5}$$

Note that the length of these different sequences do not need to be the same.

The aim of the system is that the message $\mathbf{X}$ arrives at the destination in such a way that the enemy cryptanalyst cannot decipher it. To this end, the encryptor will use the private key $\mathbf{Z}$, the private randomizer $\mathbf{S}$ and the public randomizer $\mathbf{R}$ to transform the message or *plaintext* $\mathbf{X}$ into an encrypted message $\mathbf{Y}$, called *ciphertext*. It is then assumed that the enemy cryptanalyst can only observe $\mathbf{Y}$ and $\mathbf{R}$, but does not know the realization of $\mathbf{S}$ and $\mathbf{Z}$. Based on this observation, he will then try to reconstruct the original message $\mathbf{X}$. The intended receiver, on the other hand, additionally knows the secret key $\mathbf{Z}$, which must allow him to perfectly recover $\mathbf{X}$ from $\mathbf{Y}$. Note, however, that $\mathbf{S}$ is not known to the decryptor, so it must be possible to get $\mathbf{X}$ back without it.

In this most common setup, we assume that the enemy cryptanalyst has access only to $\mathbf{Y}$ and $\mathbf{R}$: This is called *ciphertext-only attack*. There exist also more problematic situations where the enemy cryptanalyst also knows the original message $\mathbf{X}$ (and, e.g., tries to figure out the key) — this is known as *known-plaintext attack* — or, even worse, can actually choose the plaintext $\mathbf{X}$ himself — denoted *chosen-plaintext attack* — and thereby look at optimally bad cases for the encryption that will simplify the breaking.

21.3 Kerckhoff Hypothesis

There is one very important assumption implicit to our system model of Figure 21.1: We always assume that the enemy cryptanalyst has *full knowledge* of how our cryptosystem works. The only parts of the system that remain secret are the values of key, message, and private randomizer.

This assumption cannot be proven and has been (or even sometimes still is) questioned. However, there are very good reasons for it: History has shown that any secret design will eventually be discovered. Either an insider changes sides, the enemy cryptanalyst is able to steal the plans or a machine, or he is simply able to reverse-engineer the system's design. Hence, security based on the secret design is not fail-proof at all.

This has been properly recognized towards the end of the 19th century by a man called Auguste Kerckhoff, who made the following postulate.

Postulate 21.1 (Kerckhoff Hypothesis (1883)). The only part of a cryptographic system that should be kept secret is the current value of the secret key.

21.4 Perfect Secrecy

It is now time to investigate the system of Figure 21.1 more in detail. We see that the ciphertext $\mathbf{Y}$ is a function of $\mathbf{X}$, $\mathbf{Z}$, $\mathbf{R}$, and $\mathbf{S}$. Hence we have

$$\mathrm{H}(\mathbf{Y}|\mathbf{X},\mathbf{Z},\mathbf{R},\mathbf{S}) = 0. \tag{21.6}$$

The decryptor has no access to $\mathbf{S}$, but must be able to recover the plaintext from the ciphertext using the secret key and the public randomizer. Hence, a requirement for the system to work is that

$$\mathrm{H}(\mathbf{X}|\mathbf{Y},\mathbf{Z},\mathbf{R}) = 0. \tag{21.7}$$

Implicit in the figure there are two more important assumptions:

- $\mathbf{X}$, $\mathbf{Z}$, $\mathbf{R}$, and $\mathbf{S}$ are all statistically independent; and
- the secret key $\mathbf{Z}$ and the public randomizer $\mathbf{R}$ are to be used once only!

Based on these definitions and assumptions, Shannon now gave a simple, but very powerful definition of what a perfect cryptographic system should do.

Definition 21.2. A cryptographic system provides *perfect secrecy* if the enemy cryptanalyst's observation $(\mathbf{Y},\mathbf{R})$ is statistically independent of the plaintext $\mathbf{X}$:

$$(\mathbf{Y},\mathbf{R}) \perp\!\!\!\perp \mathbf{X}. \tag{21.8}$$

In information theoretic notation, this is equivalent to saying that

$$\mathrm{H}(\mathbf{X}|\mathbf{Y},\mathbf{R}) = \mathrm{H}(\mathbf{X}). \tag{21.9}$$

It is good to know that it is actually possible to design such perfect-secrecy systems!

Example 21.3 (One-Time Pad). One possible design of a system that provides perfect secrecy is shown in Figure 21.2. It is called *one-time pad*.

The main idea is to use modulo-2 adders that scramble the plaintext with the key bits. As long as the key bits are IID uniform, the system provides perfect secrecy, independently of the distribution of the plaintext, as can be shown as follows:

$$\begin{aligned}
\Pr[\mathbf{Y}=\mathbf{y}\,|\,\mathbf{X}=\mathbf{x}] &= \Pr[\mathbf{X}\oplus\mathbf{Z}=\mathbf{y}\,|\,\mathbf{X}=\mathbf{x}] && (21.10)\\
&= \Pr[\mathbf{x}\oplus\mathbf{Z}=\mathbf{y}\,|\,\mathbf{X}=\mathbf{x}] && (21.11)\\
&= \Pr[\mathbf{Z}=\mathbf{y}\oplus\mathbf{x}\,|\,\mathbf{X}=\mathbf{x}] && (21.12)\\
&= \left(\frac{1}{2}\right)^n, && (21.13)
\end{aligned}$$

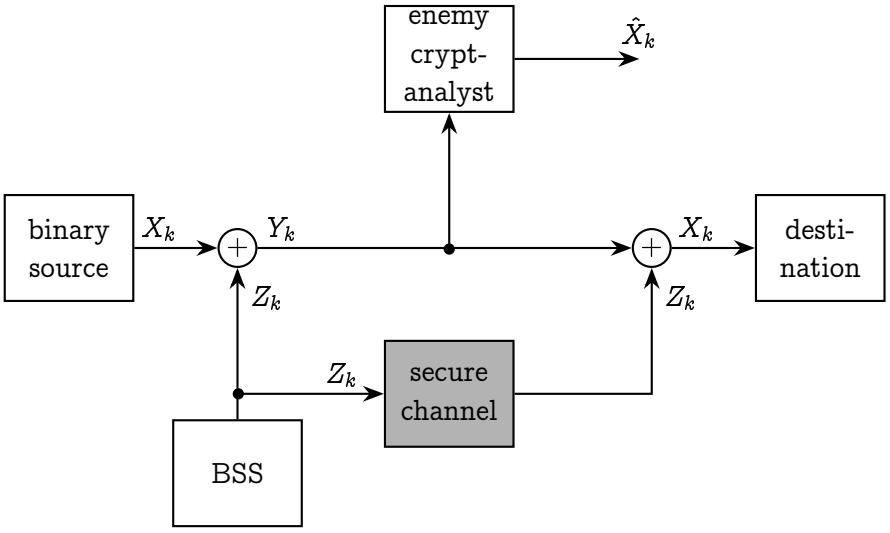


Figure 21.2: A cryptosystem that provides perfect secrecy: one-time pad.

where the last step follows because we have assumed that $\{Z_k\}$ is IID uniform. Therefore we see that $\Pr[\mathbf{Y}=\mathbf{y}\,|\,\mathbf{X}=\mathbf{x}]$ does not depend on $\mathbf{x}$, i.e., $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}$. ◊

Shannon did not only come up with the fundamental definition of perfect secrecy, but he was also able to prove an important property of it.

Theorem 21.4 (Bound on Key Size).
In any system that provides perfect secrecy, we must have

$$\mathrm{H}(\mathbf{X}) \le \mathrm{H}(\mathbf{Z}). \tag{21.14}$$

Proof: In spite of the theorem's importance, its proof is astonishingly simple. It is based on basic properties of entropy:

$$\begin{aligned}
\mathrm{H}(\mathbf{X}|\mathbf{Y},\mathbf{R}) &\le \mathrm{H}(\mathbf{X},\mathbf{Z}|\mathbf{Y},\mathbf{R}) && (21.15)\\
&= \mathrm{H}(\mathbf{Z}|\mathbf{Y},\mathbf{R}) + \mathrm{H}(\mathbf{X}|\mathbf{Z},\mathbf{Y},\mathbf{R}) && (21.16)\\
&= \mathrm{H}(\mathbf{Z}|\mathbf{Y},\mathbf{R}) && (21.17)\\
&\le \mathrm{H}(\mathbf{Z}). && (21.18)
\end{aligned}$$

Here, (21.15) follows because adding random variables cannot decrease uncertainty; (21.16) follows from the chain rule; in (21.17) we use our assumption of perfect recovery (21.7); and the final inequality (21.18) is due to conditioning that cannot increase entropy.

Hence, if we want perfect secrecy, i.e., $\mathrm{H}(\mathbf{X}|\mathbf{Y},\mathbf{R}) = \mathrm{H}(\mathbf{X})$, then we must have that $\mathrm{H}(\mathbf{X}) \le \mathrm{H}(\mathbf{Z})$. □

In other words, Theorem 21.4 says that the used key must be at least as long[1] as the message!

The alert reader will therefore now complain that instead of going through all the trouble of encrypting and decrypting the message and transmitting the secret key (which is longer than the message!) over the secure channel, we should instead simply use the secure channel to transmit the message in the first place!

This is not a completely fair observation. For example, there might be a time difference between the transmission of the secret key over the secure channel and the transmission of the ciphertext over the public channel. Consider, e.g., the situation of a ship: The key could be generated and brought aboard before a ship departs, but the secret messages are then sent via radio transmission once the ship is far out at sea and no secure channel exists anymore.

Nevertheless, the alert reader has a point. His observation very clearly puts its finger on the two most fundamental problems of the system shown in Figure 21.1:

- In a practical situation, we would like to have the key size much smaller than the plaintext. In other words, we would like to reuse the same key $\mathbf{Z}$ for different messages $\mathbf{X}$.[2]
- In a practical situation we often do not have a secure channel available.

We will try to address these two issues in the following sections. We start with an investigation of what happens if one shortens the key length in such a way that (21.14) is violated.

21.5 Imperfect Secrecy

We have learned that perfect secrecy is possible, but that it is rather expensive because it needs a large amount of secret key. In many practical situations perfect secrecy seems too complicated. So the question is what we can do if we are happy with imperfect secrecy. To analyze this, Shannon again introduced a quantity that very accurately describes almost every imperfect cryptographic system.

So we consider again Figure 21.1, but assume for the moment that there are no public and private randomizers $\mathbf{R}$ and $\mathbf{S}$. Moreover, we assume that

[1] Strictly speaking this is only true with the additional assumption that all message bits and all key bits are independent and uniform, such that they maximize entropy. If the message bits have big dependencies, i.e., $\mathrm{H}(\mathbf{X})$ is not very large, we might have a shorter key than the message and still get perfect secrecy as long as (21.14) is satisfied.

[2] Note that reusing the same key for different messages can be seen as a short-length key $\mathbf{Z}$ being used for a very long message $\mathbf{X}$ that actually consists of the concatenation of many messages.

we use a system that uses a relatively short finite-length secret key $\mathbf{Z}$ for a long message (or equivalently, many concatenated short messages) and that therefore cannot provide perfect secrecy.

How can we characterize our system? Obviously, since the secret key $\mathbf{Z}$ is of finite length, if we keep using it to encode a sequence of message digits $\{X_k\}$, eventually it must be possible to compute the value of $\mathbf{Z}$ simply from the observation of $\{Y_k\}$. So the quality of our system will be determined by

- the minimum number of ciphertext digits $Y_1, \ldots, Y_{k_\mathrm{u}}$ that are needed to determine $\mathbf{Z}$ uniquely; and
- the difficulty of actually computing $\mathbf{Z}$ from such a sequence $Y_1, \ldots, Y_{k_\mathrm{u}}$.

The latter depends very strongly on the system and on our ingenuity of attacking the cryptographic system. We will discuss more about that in Section 21.6.

The former, however, can be described in quite general form for any reasonably good system. To this end, Shannon gave the following definitions.

Definition 21.5. The *key equivocation function* $f_{\mathrm{eq}}(\cdot)$ is defined as follows:

$$f_{\mathrm{eq}}(k) \triangleq \begin{cases} \mathrm{H}(\mathbf{Z}) & k = 0, \\ \mathrm{H}(\mathbf{Z}|Y_1, \ldots, Y_k) & k = 1, 2, 3, \ldots \end{cases} \tag{21.19}$$

Furthermore, the *unicity distance* k_u is defined as the smallest k such that $f_{\mathrm{eq}}(k) \approx 0$. Hence k_u is the smallest amount of ciphertext necessary to determine the key uniquely.[3]

Ingeniously, Shannon was able to derive a general form of $f_{\mathrm{eq}}(\cdot)$ that holds basically for all reasonably good cryptographic systems. He only made two reasonable assumptions about a cryptosystem:

1. A good cryptosystem will try to make $Y_1, \ldots, Y_k$ look totally random for as large k as possible. Hence, for as large k as possible,
$$\mathrm{H}(Y_1, \ldots, Y_k) \approx k \cdot \log |\mathcal{Y}|. \tag{21.20}$$
2. By definition, when the secret key $\mathbf{Z}$ is known, $\mathbf{Y} = (Y_1, \ldots, Y_n)$ determines $\mathbf{X}$ completely:
$$\mathrm{H}(Y_1, \ldots, Y_n|\mathbf{Z}) = \mathrm{H}(\mathbf{X}). \tag{21.21}$$
For most ciphers and sources, it then holds that $Y_1, \ldots, Y_k$ will determine a certain percentage of $\mathbf{X}$. Usually,
$$\mathrm{H}(Y_1, \ldots, Y_k|\mathbf{Z}) \approx \frac{k}{n} \mathrm{H}(\mathbf{X}) \tag{21.22}$$
is a good approximation.

[3] However, as mentioned above already, note that even though theoretically it must be possible to determine the key $\mathbf{Z}$ from $Y_1, \ldots, Y_{k_\mathrm{u}}$ because $\mathrm{H}(\mathbf{Z}|Y_1, \ldots, Y_{k_\mathrm{u}}) \approx 0$, in practice this should still be a very difficult task!

Based on these two assumptions we now get

$$\begin{aligned} \mathrm{H}(Y_1,\ldots,Y_k,\mathbf{Z}) &= \mathrm{H}(\mathbf{Z}) + \mathrm{H}(Y_1,\ldots,Y_k|\mathbf{Z}) && (21.23)\\ &\approx \mathrm{H}(\mathbf{Z}) + \frac{k}{n}\,\mathrm{H}(\mathbf{X}) && (21.24) \end{aligned}$$

and

$$\begin{aligned} \mathrm{H}(Y_1,\ldots,Y_k,\mathbf{Z}) &= \mathrm{H}(Y_1,\ldots,Y_k) + \mathrm{H}(\mathbf{Z}|Y_1,\ldots,Y_k) && (21.25)\\ &\approx k\cdot\log|\mathcal{Y}| + f_{\mathrm{eq}}(k). && (21.26) \end{aligned}$$

Hence,

$$\begin{aligned} f_{\mathrm{eq}}(k) &\approx \mathrm{H}(\mathbf{Z}) - k\log|\mathcal{Y}| + \frac{k}{n}\,\mathrm{H}(\mathbf{X}) && (21.27)\\ &= \mathrm{H}(\mathbf{Z}) - k\log|\mathcal{Y}|\left(1-\frac{\mathrm{H}(\mathbf{X})}{n\log|\mathcal{Y}|}\right) && (21.28)\\ &\triangleq \mathrm{H}(\mathbf{Z}) - k\rho\log|\mathcal{Y}|, && (21.29) \end{aligned}$$

where the last equality should be read as definition for ρ.

Proposition 21.6. For most reasonable ciphers, the key equivocation function is linear in k:

$$f_{\mathrm{eq}}(k) \approx \mathrm{H}(\mathbf{Z}) - k\rho\log|\mathcal{Y}| \tag{21.30}$$

with a slope $-\rho\log|\mathcal{Y}|$ where

$$\rho \triangleq 1-\frac{\mathrm{H}(\mathbf{X})}{n\log|\mathcal{Y}|}. \tag{21.31}$$

The basic behavior of the key equivocation function is shown in Figure 21.3.

From Proposition 21.6 now follows that the unicity distance is approximately given as

$$k_{\mathrm{u}} \approx \frac{\mathrm{H}(\mathbf{Z})}{\rho\log|\mathcal{Y}|}. \tag{21.32}$$

Therefore, since we would like k_{u} to be as big as possible, we need ρ to be as small as possible, i.e., $\mathrm{H}(\mathbf{X})$ should be as large as possible! For exactly this goal, the private randomizer $\mathbf{S}$ is useful: It helps to increase the entropy!

Example 21.7 (Usage of Private Randomizer). In English text (26 letters + space), the most likely letter is "space" (probability around 0.1859), then "e"

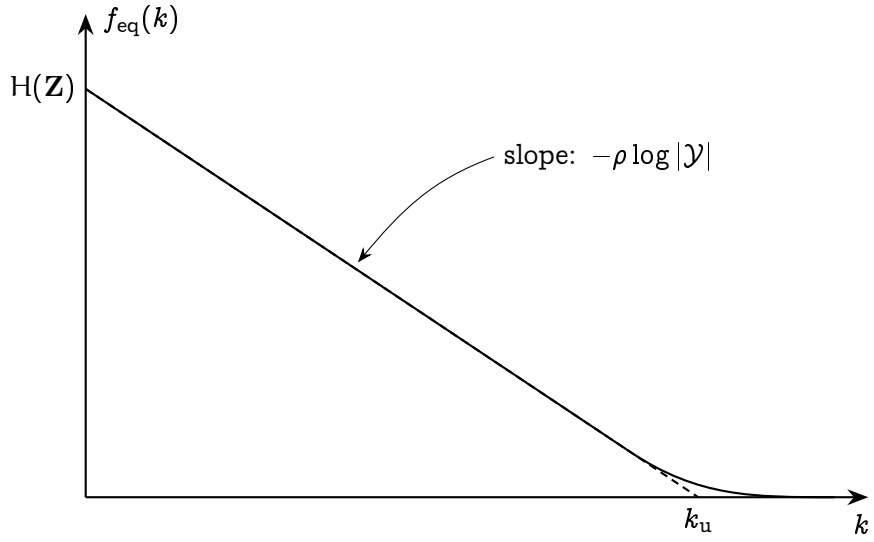


Figure 21.3: The key equivocation function.

(probability around 0.1031), etc. If we convert such text into 11-bit symbols ($2^{11} = 2048$) in the manner that 381 ($= 2048 \cdot 0.1859$) of the 2048 symbols represent "space", 211 ($2048 \cdot 0.1031$) symbols represent "e", etc., and then choose any of the possible representations uniformly at random using the private randomizer $\mathbf{S}$, then the output looks uniformly distributed! Note that the receiver does not need to know $\mathbf{S}$ because he can simply replace any of the 381 space-symbols by "space", etc. ◊

So we see that if we use a private randomizer $\mathbf{S}$, in (21.31) we have to replace $\mathrm{H}(\mathbf{X})$ by $\mathrm{H}(\tilde{\mathbf{X}}) = \mathrm{H}(\mathbf{X}) + \mathrm{H}(\mathbf{S})$. Therefore the private randomizer helps increasing the unicity distance, i.e., we will need to know more ciphertext until $\mathbf{Z}$ can (theoretically) be computed from it.

The above analysis is not affected by the use of a public randomizer.

Note that if instead of a ciphertext-only attack we allow a known-plaintext attack (enemy cryptanalyst also knows $\mathbf{X}$) or, even worse, a chosen-plaintext attack (enemy cryptanalyst can choose $\mathbf{X}$ himself), then we need to replace $\mathrm{H}(\mathbf{X})$ by 0, i.e., $\rho = 1$.

21.6 Computational vs. Unconditional Security

Since perfect secrecy is difficult to achieve, most practical systems rely on "practical" security, which means that we do not rely on the *impossibility* of breaking a code, but on the *difficulty* in breaking it.

If the best super-computer needs 500 years on average to break a certain cipher, then this is good enough for us (as in 500 years the encrypted message

is probably useless anyway). We say that such a system is *computationally secure*.

However, note that we have the problem that we are not able to prove the security! There still is the possibility that someone has a brilliant idea and suddenly finds a way of cracking a code within two hours. So this means that computational security is not a guaranteed security.

So, by relying on computational security, we try to deal with the unpractical requirement that the secret key has to have more entropy than the plaintext (see Theorem 21.4). Another way to put this is that if we rely on computational security, then we can reuse the same key many times. An attacker will not be able to break our code even if according to the equivocation function f_{eq} the secret key is already fully determined by $\{Y_k\}$.

Unfortunately, in Figure 21.1 we still have another obstacle that hinders us in practical implementations: the secure channel. We have already mentioned that sometimes there might be a secure channel for a certain time period (e.g., during the time a ship is in the harbor), but not anymore afterwards. Unfortunately, in many practical situations there simply is no secure channel. Consider, e.g., the Internet: Nobody is willing to fly to the United States before being able to establish a secure connection with the amazon.com servers...

Incredibly, there is a solution to this problem: *Public-key cryptography* gets rid of the need of a secure channel!

21.7 Public-Key Cryptography

In 1976, Diffie and Hellman [DH76] (and independently Merkle [Mer78]) published a paper that gave a fundamentally new idea: They realized that if we allow to rely on computationally secure systems only, then we can do *without a secure channel*!

So, for the remainder of this chapter we leave Figure 21.1 behind, but consider a new system. We will depict the exact shape of this system later in Section 21.7.2, see Figures 21.5 and 21.6.

The basic idea could be explained by an analogy shown in Figure 21.4: Mr. A would like to send a locked suitcase to his friend B without having to send the key. How can he do it? The idea is simple. Firstly he sends the suitcase locked. The friend B cannot open it as he does not have the key. But he can add an additional lock and send the suitcase back to Mr. A. Mr. A now removes his lock (the lock of his friend B remains there and Mr. A cannot open it because only B has the key) and then sends the suitcase again to B. B can now remove his own lock and open the suitcase. We realize: The suitcase has never been traveling without lock, but there was no need of sending a key!

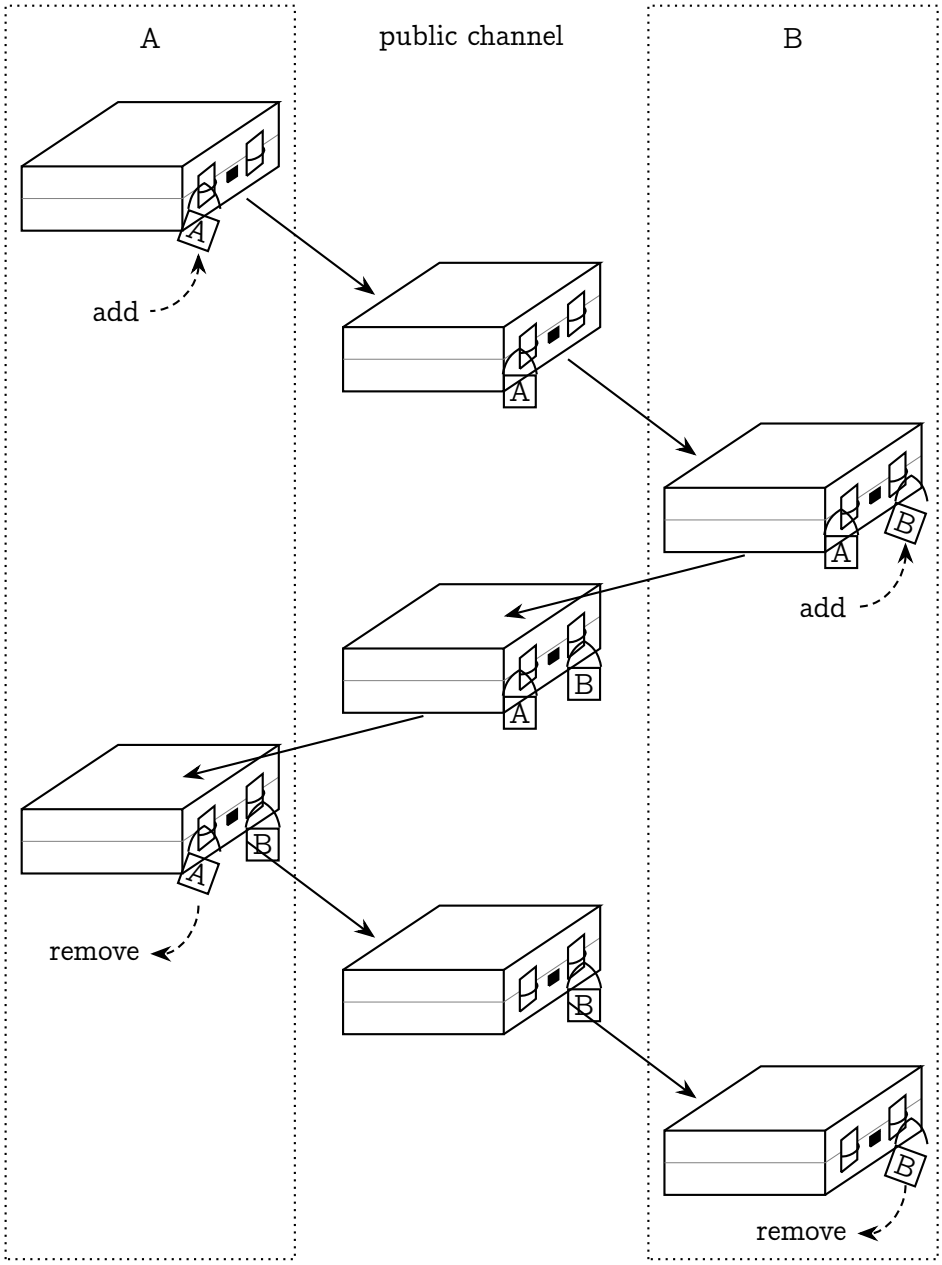


Figure 21.4: Analogy on how one can send secret messages without exchanging keys.

To explain how this idea can be translated into the world of digital communication we need to discuss two crucial definitions introduced by Diffie, Hellman, and Merkle: the *one-way function* and the *trapdoor one-way function*.

21.7.1 One-Way Function

Definition 21.8. A one-to-one function $f(\cdot)$ is a *one-way function* if

- it is easy to compute $\mathbf{y} = f(\mathbf{x})$ for every $\mathbf{x}$, but
- it is computationally impossible to compute $\mathbf{x} = f^{-1}(\mathbf{y})$ for virtually all $\mathbf{y}$.

The reader may note the unclear language we have been using here: "easy", "computationally impossible", "virtually all", ... These are all terms that mathematically do not make much sense, however, it turns out that for engineers they are quite clear and very useful.

As a matter of fact, one-way functions were not inventions of Diffie and Hellman, but they have been used in different context before. Best known is their use in login-systems: A user i chooses a password $\mathbf{x}_i$ that he needs to present to the login-screen in order to be allowed to log into a computer system. The computer, however, does not store this password, but computes $\mathbf{y}_i = f(\mathbf{x}_i)$ and stores $(i, \mathbf{y}_i)$. When logging in, user i gives $\mathbf{x}_i$, the computer computes $\mathbf{y}_i$ and compares it with the stored value. If they are identical, the login is approved, otherwise rejected. The advantage of this system is that if someone somehow gets hold of $(i, \mathbf{y}_i)$, this will not help him to break into the system because he cannot compute $\mathbf{x}_i = f^{-1}(\mathbf{y}_i)$.

We will now give an example of a (conjectured) one-way function.

Example 21.9. The function

$$y = f(x) = \alpha^x \pmod{p}, \tag{21.33}$$

where p is a large prime number such that $n = p - 1$ has a large prime factor (ideally $n = p-1 = 2p'$ for p' also prime), is a (conjectured[4]) one-way function:

- $x = \log_\alpha y \pmod{p}$ is very hard to compute;

[4] Again, this has not been proven. There is strong evidence, but theoretically there exists the possibility that one day someone has a brilliant idea on how to quickly compute discrete logarithms, changing the one-way function to a normal "two-way" function.

- $\alpha^x \pmod p$ is easy to compute by squaring: For example,

$$\begin{aligned}\alpha^{1300} &= \alpha^{1024}\cdot\alpha^{256}\cdot\alpha^{16}\cdot\alpha^4 && (21.34)\\ &= \alpha^{2^{10}}\cdot\alpha^{2^8}\cdot\alpha^{2^4}\cdot\alpha^{2^2} && (21.35)\\ &= \underbrace{(((\alpha^2)^2)^{\cdots})^2}_{\substack{10 \text{ squaring}\\ \text{operations}}}\cdot\underbrace{(((\alpha^2)^2)^{\cdots})^2}_{\substack{8 \text{ squaring}\\ \text{operations}}}\cdot(((\alpha^2)^2)^2)^2\cdot(\alpha^2)^2 && (21.36)\end{aligned}$$

  needs only 10 squaring operations (if we store the intermediate values for 2, 4, and 8 squaring operations) and 3 multiplications. Hence, it is very fast. ♢

How can we now use one-way functions in the context of cryptography? Consider the following system of generating a common key between two users: Fix an α and a p in advance. User i chooses a *private key* $x_i \pmod p$ and computes a *public key* $y_i = \alpha^{x_i} \pmod p$. User j does the same thing. If user i and j want to communicate, then user i fetches y_j and user j fetches y_i from a *trusted public database* that contains the public keys of many users.

User i now computes

$$(y_j)^{x_i} = (\alpha^{x_j})^{x_i} = \alpha^{x_j x_i} \pmod p \tag{21.37}$$

and user j computes

$$(y_i)^{x_j} = (\alpha^{x_i})^{x_j} = \alpha^{x_i x_j} \pmod p. \tag{21.38}$$

Both get the same value! This common value $\alpha^{x_i x_j} \pmod p$ can now be used as common secret key on a standard cryptographic system.

An enemy cryptanalyst can also fetch y_i and y_j, but he can only compute the common secret key if he manages to compute $x_i = \log_\alpha y_i \pmod p$ or $x_j = \log_\alpha y_j \pmod p$, which, as said, is very difficult.

The problem of this system is that both users will always use the same secret key, which is not a good idea. So Diffie and Hellman expanded their idea to not only find a common key, but directly create a complete *public-key cryptosystem*.

21.7.2 Trapdoor One-Way Function

Definition 21.10. A *trapdoor one-way function* $f_z(\cdot)$ is a family of one-to-one functions with parameter z (the *trapdoor*) satisfying the following properties:

1. If z is known, it is easy to find algorithms $\mathscr{E}_z$ and $\mathscr{D}_z$ that easily compute f_z and f_z^{-1}, respectively.
2. For virtually all z and $\mathbf{y}$, it is computationally infeasible to compute $\mathbf{x} = f_z^{-1}(\mathbf{y})$ even if $\mathscr{E}_z$ is known.

3. It is easy to pick a value of z at random.

Again note the mathematically unclear terms "easy", "computationally infeasible", and "virtually all".

Let us first assume that such a trapdoor one-way function actually exists and show how we can use it.

We can create a *public-key cryptosystem* in the following way. Every user i chooses a trapdoor z_i. (This is easy due to Property 3.) User i then finds his encryption algorithm $\mathscr{E}_{z_i}$ and his decryption algorithm $\mathscr{D}_{z_i}$. He publishes $\mathscr{E}_{z_i}$ in a *trusted public database (TPD)*. Note that usually $\mathscr{E}_{z_i}$ and $\mathscr{D}_{z_i}$ are known in advance apart from some parameters, i.e., we only need to publish these parameters. The decryption algorithm $\mathscr{D}_{z_i}$ is kept secret. We say that

- $\mathscr{E}_{z_i}$ is the *public key*, and
- $\mathscr{D}_{z_i}$ is the *private key*.

This system now works both for secrecy and authenticity:

Secrecy: User i wants to send a secret message $\mathbf{X}$ to user j. He fetches $\mathscr{E}_{z_j}$ of user j from the TPD, computes $\mathbf{Y} = f_{z_j}(\mathbf{X})$, and transmits $\mathbf{Y}$. The ciphertext $\mathbf{Y}$ can only be decrypted by user j because only he knows $\mathscr{D}_{z_j}$ to compute $\mathbf{X} = f_{z_j}^{-1}(\mathbf{Y})$.

The corresponding system is shown in Figure 21.5.

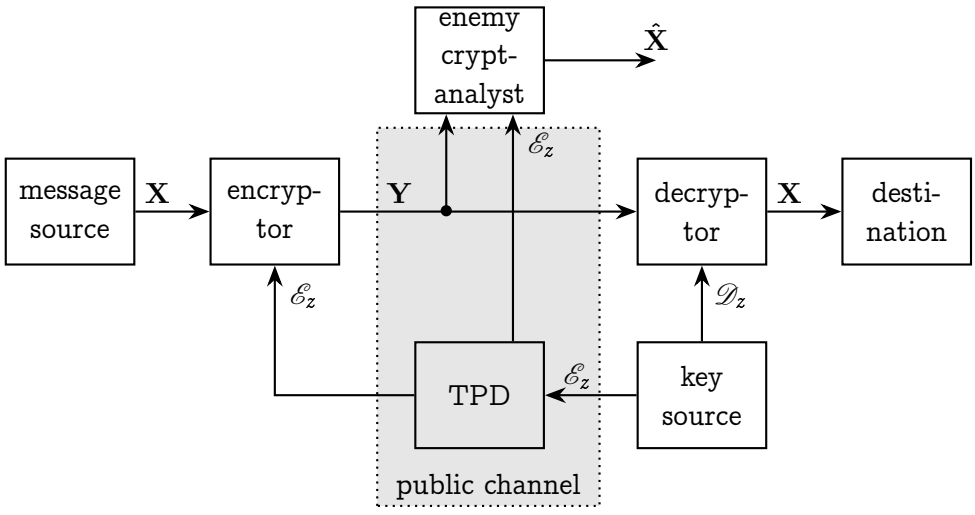


Figure 21.5: System model of a public-key cryptosystem that provides secrecy.

Authenticity (digital signature): User i applies his private decryption algorithm $\mathscr{D}_{z_i}$ on $\mathbf{X}$: $\mathbf{X}' = f_{z_i}^{-1}(\mathbf{X})$, then transmits $(\mathbf{X}, \mathbf{X}')$ to user j. User j can get $\mathscr{E}_{z_i}$ from the TPD and use it to compute $f_{z_i}(\mathbf{X}')$. If $\mathbf{X} = f_{z_i}(\mathbf{X}')$,

then this proves that $\mathbf{X}$ must come from user i since only user i knows $\mathscr{D}_{z_i}$.

The enemy cryptanalyst tries to send a fake message $\tilde{\mathbf{X}}$ to user j and to forge a wrong signature $\tilde{\mathbf{X}}'$ pretending that the message came from the legitimate user i. But even though the enemy cryptanalyst has access to $\mathscr{E}_{z_i}$, he cannot create a proper $\tilde{\mathbf{X}}'$ because for that he would need to know $\mathscr{D}_{z_i}$.

The corresponding system is shown in Figure 21.6.

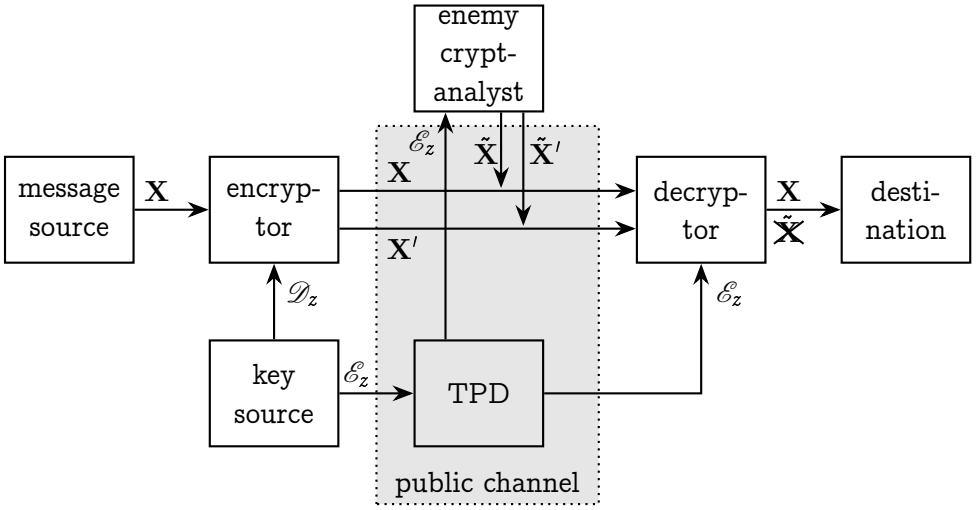


Figure 21.6: System model of a public-key cryptosystem that provides authenticity.

Note that this idea of a digital signature was the most important contribution of public-key cryptography!

We finish our short insight into public cryptography by stating one of the best known (conjectured) trapdoor one-way functions. We will only give a rough big picture and will omit any mathematical details.

Conjecture 21.11 (RSA Trapdoor One-Way Function). The **River–Shamir–Adleman (RSA) trapdoor** is given by

$$z = (p_1, p_2, e), \tag{21.39}$$

where p_1, p_2 are large distinct prime numbers such that $p_1 - 1$ and $p_2 - 1$ have large prime factors and where e is a unit in $\mathbb{Z}_{(p_1-1)(p_2-1)}$ (i.e., an integer for which there exists a multiplicative inverse in $\mathbb{Z}_{(p_1-1)(p_2-1)}$). This is easy to pick at random because powerful algorithms are available to check whether a number is prime or not.

Using

$$m \triangleq p_1 p_2 \tag{21.40}$$

we now define the trapdoor one-way function as follows:

$$f_z(x) = x^e \pmod{m}. \tag{21.41}$$

The **public key** $\mathscr{E}_z$ is

$$\mathscr{E}_z = (m, e). \tag{21.42}$$

Note that with the knowledge of (m, e), $f_z(x)$ can be easily computed by the squaring approach shown in Section 21.7.1.

The **private key** $\mathscr{D}_z$ is

$$\mathscr{D}_z = (m, d), \tag{21.43}$$

where

$$d \triangleq e^{-1} \pmod{(p_1 - 1)(p_2 - 1)}. \tag{21.44}$$

It can be shown that

$$(x^e)^d \pmod{m} = x, \tag{21.45}$$

i.e., it is easy to compute $x = f_z^{-1}(y)$ by the squaring approach. However, even if we know (m, e), to compute d we need to factorize $m = p_1 p_2$, which is very hard!

It has been proven that breaking the RSA trapdoor one-way function is equivalent to factorizing m. So as long as no one can find a way of quickly factorizing large numbers into its primes, our cryptosystems used throughout the world for banking, controlling, communicating, etc., are safe.

Appendix A

Gaussian Random Variables

In many communication scenarios the noise is modeled as a Gaussian stochastic process. This is sometimes justified by invoking the Central Limit Theorem that says that many small independent disturbances add up to a random process that is approximately Gaussian. But even if no such theorem can be used, the engineers like to use Gaussian random processes because — even though Gaussian processes might seem quite scary at first — they are actually well understood and often allow closed-form solutions.

Rather than starting immediately with the definition and analysis of Gaussian processes, we shall take the more moderate approach and start by first discussing Gaussian random variables. Building on that we shall discuss Gaussian vectors in Appendix B, and only then introduce Gaussian processes in Appendix C. Our aim is to convince you that

Gaussians are your friends!

A.1 Standard Gaussian Random Variables

We begin with the definition of a standard Gaussian random variable.

Definition A.1. We say that the random variable W has a *standard Gaussian distribution* if its probability density function (PDF) $f_W(\cdot)$ is given by

$$f_W(w) = \frac{1}{\sqrt{2\pi}} e^{-\frac{w^2}{2}}, \quad w \in \mathbb{R}. \tag{A.1}$$

See Figure A.1 for a plot of this density.

For this definition to be meaningful it had better be the case that the right-hand side of the above is a density function, i.e., that it is nonnegative

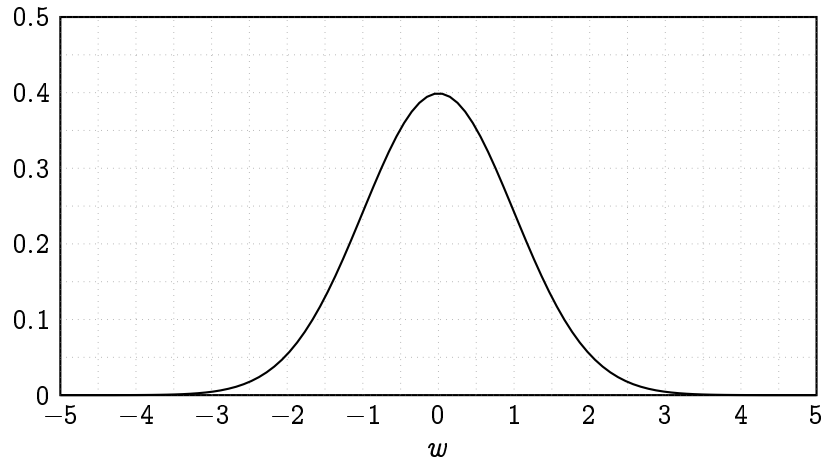


Figure A.1: The standard Gaussian probability density function.

and that it integrates to one. This is indeed the case. In fact, the density is positive, and it integrates to one because

$$\int_{-\infty}^{\infty} e^{-\frac{w^2}{2}}\,\mathrm{d}w = \sqrt{2\pi}. \tag{A.2}$$

This latter integral can be verified by computing its square as follows:

$$\begin{aligned}
\left(\int_{-\infty}^{\infty} e^{-\frac{w^2}{2}}\,\mathrm{d}w\right)^2 &= \int_{-\infty}^{\infty} e^{-\frac{w^2}{2}}\,\mathrm{d}w \int_{-\infty}^{\infty} e^{-\frac{v^2}{2}}\,\mathrm{d}v && \text{(A.3)}\\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\frac{w^2+v^2}{2}}\,\mathrm{d}w\,\mathrm{d}v && \text{(A.4)}\\
&= \int_{-\pi}^{\pi}\int_{0}^{\infty} r\,e^{-\frac{r^2}{2}}\,\mathrm{d}r\,\mathrm{d}\theta && \text{(A.5)}\\
&= 2\pi\int_{0}^{\infty} e^{-u}\,\mathrm{d}u && \text{(A.6)}\\
&= 2\pi. && \text{(A.7)}
\end{aligned}$$

Here we used the change of variables from Cartesian (w, v) to polar (r, θ):

$$\left.\begin{aligned} w &= r\cos\theta\\ v &= r\sin\theta \end{aligned}\right\} \quad r \geq 0,\ -\pi \leq \theta < \pi,\ \mathrm{d}w\,\mathrm{d}v = r\,\mathrm{d}r\,\mathrm{d}\theta. \tag{A.8}$$

Note that the density of a standard Gaussian random variable (A.1) is symmetric about the origin, so that if W is a standard Gaussian, then so is $-W$. This symmetry also establishes that the expectation of a standard Gaussian is zero:

$$\mathsf{E}[W] = 0. \tag{A.9}$$

The variance of a standard Gaussian can be computed using integration by parts:

$$\int_{-\infty}^{\infty} w^2 \frac{1}{\sqrt{2\pi}} e^{-\frac{w^2}{2}} \,\mathrm{d}w = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} w \left(-\frac{\mathrm{d}}{\mathrm{d}w} e^{-\frac{w^2}{2}} \right) \mathrm{d}w \tag{A.10}$$

$$= \frac{1}{\sqrt{2\pi}} \left(-w\, e^{-\frac{w^2}{2}} \Big|_{-\infty}^{\infty} + \int_{-\infty}^{\infty} e^{-\frac{w^2}{2}} \,\mathrm{d}w \right) \tag{A.11}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{w^2}{2}} \,\mathrm{d}w \tag{A.12}$$

$$= 1. \tag{A.13}$$

Here we have used the fact that the limit of $w \exp(-w^2/2)$ as $|w|$ tends to infinity is zero, and in the last line we used (A.2).

A.2 Gaussian Random Variables

We next define a Gaussian (not necessarily standard) random variable as the result of applying an affine transformation to a standard Gaussian.

Definition A.2. We say that a random variable X is *Gaussian* if it can be written in the form

$$X = aW + b \tag{A.14}$$

for some deterministic real numbers $a, b \in \mathbb{R}$ and for a standard Gaussian W.

If $b = 0$, i.e., if X can be written as $X = aW$ for some $a \in \mathbb{R}$, then we call X to be *centered Gaussian*.

Remark A.3. We do not preclude a from being zero. The case $a = 0$ leads to X being deterministically equal to b. We thus include the deterministic random variables in the family of Gaussian random variables. △

Remark A.4. From Definition A.2 it follows that the family of Gaussian random variables is closed with respect to affine transformations: If X is Gaussian and $\alpha, \beta \in \mathbb{R}$ are deterministic, then $\alpha X + \beta$ is also Gaussian. △

Proof: Since X is Gaussian, it can be written as

$$X = aW + b, \tag{A.15}$$

where W is a standard Gaussian. Consequently,

$$\alpha X + \beta = \alpha(aW + b) + \beta \tag{A.16}$$

$$= (\alpha a)W + (\alpha b + \beta), \tag{A.17}$$

which demonstrates that $\alpha X + \beta$ is Gaussian, because αa is a deterministic real number, $\alpha b + \beta$ is a deterministic number, and W is a standard Gaussian. □

If (A.14) holds then the random variables on the right-hand and the left-hand sides of (A.14) must be of equal mean. But the mean of a standard Gaussian is zero so that the mean of the right-hand side of (A.14) is b. The left-hand side is of mean $\mathsf{E}[X]$, and we thus conclude that in the representation (A.14) the deterministic constant b is uniquely determined by the mean of X, and in fact,

$$b = \mathsf{E}[X]. \tag{A.18}$$

Similarly, since the variance of a standard Gaussian is one, the variance of the right-hand side of (A.14) is a^2. So because the variance of the left-hand side is $\mathsf{Var}[X]$, we conclude that

$$a^2 = \mathsf{Var}[X]. \tag{A.19}$$

Up to its sign, the deterministic constant a in the representation (A.14) is thus also unique.

Based on the above, one might mistakenly think that for any given mean μ and variance σ^2 there are two different Gaussian distributions corresponding to

$$\sigma W + \mu \quad \text{and} \quad -\sigma W + \mu \tag{A.20}$$

where W is a standard Gaussian. This, however, is not the case, as we show in the following lemma.

Lemma A.5. There is only one Gaussian distribution of a given mean μ and variance σ^2.

Proof: This can be seen in two different ways. The first way is to note that both representations in (A.20) actually lead to the same distribution, because the standard Gaussian W has a symmetric distribution, so that σW and $-\sigma W$ have the same law. The other approach is based on computing the density of $\sigma W + \mu$ and showing that it is a symmetric function of σ, see (A.22). □

Having established that there is only one Gaussian distribution of a given mean μ and variance σ^2, we denote it by

$$\mathcal{N}(\mu, \sigma^2) \tag{A.21}$$

and set out to study its PDF. If $\sigma^2 = 0$, then the Gaussian distribution is deterministic with mean μ and has no density[1]. If $\sigma^2 > 0$, then the density can be computed from the density of the standard Gaussian distribution: If X is $\mathcal{N}(\mu, \sigma^2)$ distributed, then since $\mu + \sigma W$ is Gaussian of mean μ and variance

[1] Some would say that the density of a deterministic random variable is given by a Dirac Delta, but we prefer not to use Dirac Deltas in this discussion as they are not functions.

σ^2 where W is a standard Gaussian, it follows from the fact that there is only one Gaussian distribution of given mean and variance, that the distribution of X is the same as the distribution of $\mu+\sigma W$. The density of the latter can be computed from (A.1) to yield that the density f_X of a $\mathcal{N}(\mu,\sigma^2)$-distributed Gaussian random variable X is

$$f_X(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\,e^{-\frac{(x-\mu)^2}{2\sigma^2}},\quad x\in\mathbb{R}, \tag{A.22}$$

which is depicted in Figure A.2.

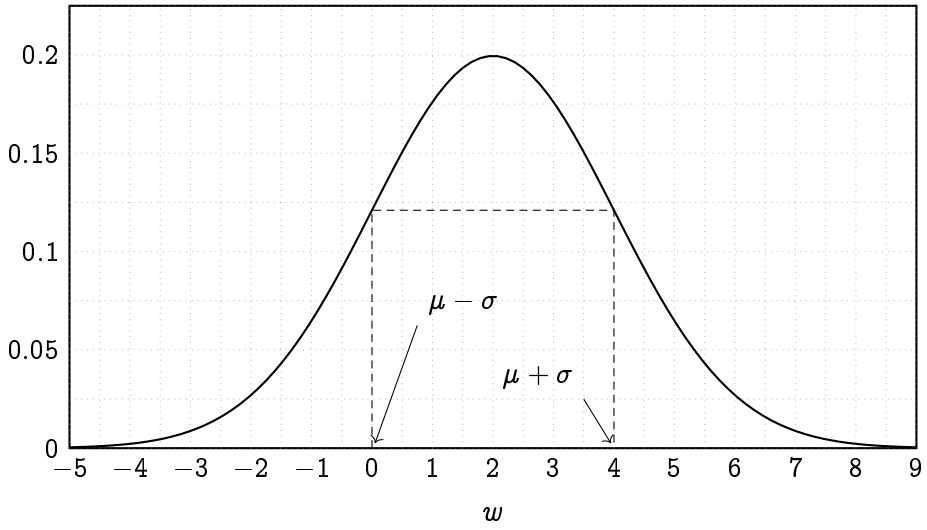


Figure A.2: The Gaussian probability density function with mean $\mu=2$ and variance $\sigma^2=4$.

Here we have used the fact that if $X=g(W)$ where $g(\cdot)$ is a deterministic one-to-one function (in our case $g(w)=\mu+\sigma\cdot w$) and where W is of density $f_W(\cdot)$ (in our case (A.1)), then the density $f_X(\cdot)$ of X is given by

$$f_X(x)=\begin{cases}0 & \text{if for no } \xi \text{ with } f_W(\xi)>0 \text{ is } x=g(\xi),\\ \frac{1}{\left|\frac{\mathrm{d}g(w)}{\mathrm{d}w}\big|_{w=\xi}\right|}f_W(\xi) & \text{if } \xi \text{ satisfies } x=g(\xi) \text{ and } f_W(\xi)>0.\end{cases} \tag{A.23}$$

From the fact that Gaussian random variables are closed under deterministic affine transformations, it follows that if $X\sim\mathcal{N}(\mu,\sigma^2)$ with $\sigma^2>0$ then $(X-\mu)/\sigma$ is also a Gaussian random variable. Since it is of zero mean and of unit variance, it follows that it must be a standard Gaussian, because there is only one Gaussian distribution of zero mean and unit variance. We thus conclude:

$$X\sim\mathcal{N}(\mu,\sigma^2)\;\Longrightarrow\;\frac{X-\mu}{\sigma}\sim\mathcal{N}(0,1),\quad \sigma^2>0. \tag{A.24}$$

Recall that the cumulative distribution function (CDF) $F_X(\cdot)$ of a random variable X is defined as

$$F_X(x) \triangleq \Pr[X \le x] \tag{A.25}$$
$$= \int_{-\infty}^{x} f_X(\xi)\,\mathrm{d}\xi, \quad x \in \mathbb{R}, \tag{A.26}$$

where the second equality holds if X has a density function $f_X(\cdot)$. If W is a standard Gaussian then its CDF is thus given by

$$F_W(w) = \int_{-\infty}^{w} \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}}\,\mathrm{d}\xi, \quad w \in \mathbb{R}. \tag{A.27}$$

There is, alas, no closed-form expression for this integral. To handle such expressions we next introduce the Q-function.

A.3 Q-Function

Definition A.6. The *Q-function* is defined by

$$Q(\alpha) \triangleq \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-\frac{\xi^2}{2}}\,\mathrm{d}\xi. \tag{A.28}$$

We thus see that $Q(\alpha)$ is the probability that a standard Gaussian random variable will exceed the value α. For a graphical interpretation of this integral see Figure A.3.

Note the relationship of the Q-function to the *error function*:

$$Q(\alpha) = \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{\alpha}{\sqrt{2}}\right)\right). \tag{A.29}$$

The Q-function is a well-tabulated function, and we are therefore usually happy when we can express answers to various questions using this function. The CDF of a standard Gaussian W can be expressed using the Q-function as follows:

$$F_W(w) = \Pr[W \le w] \tag{A.30}$$
$$= 1 - \Pr[W \ge w] \tag{A.31}$$
$$= 1 - Q(w). \tag{A.32}$$

Similarly, with the aid of the Q-function we can express the probability that a standard Gaussian random variable W will take a value in some given interval $[a, b]$:

$$\Pr[a \le W \le b] = \Pr[W \ge a] - \Pr[W \ge b] \tag{A.33}$$
$$= Q(a) - Q(b), \qquad a \le b. \tag{A.34}$$

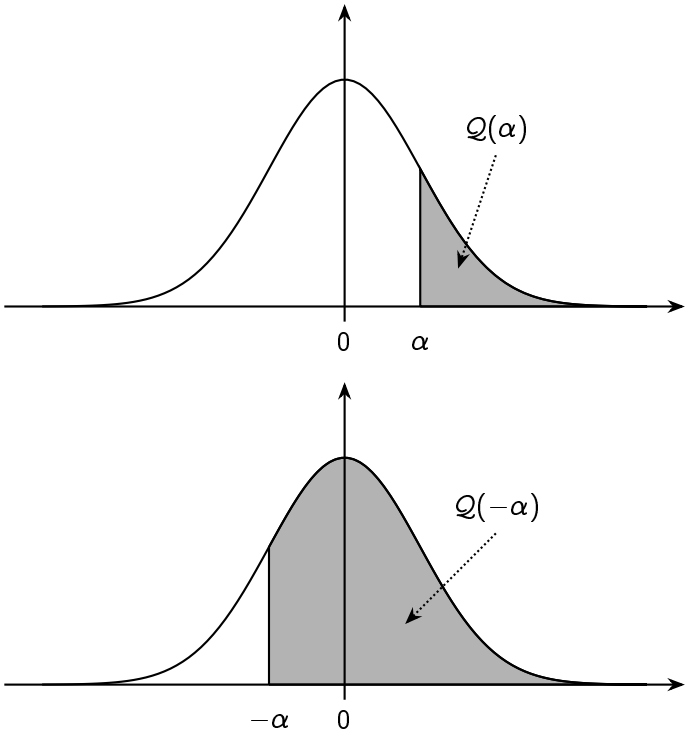


Figure A.3: Graphical interpretation of the Q-function.

More generally, if $X \sim \mathcal{N}(\mu, \sigma^2)$ with $\sigma > 0$ then

$$\begin{aligned}
\Pr[a \le X \le b] &= \Pr[X \ge a] - \Pr[X \ge b] && \text{(A.35)}\\
&= \Pr\left[\frac{X-\mu}{\sigma} \ge \frac{a-\mu}{\sigma}\right] - \Pr\left[\frac{X-\mu}{\sigma} \ge \frac{b-\mu}{\sigma}\right] && \text{(A.36)}\\
&= Q\left(\frac{a-\mu}{\sigma}\right) - Q\left(\frac{b-\mu}{\sigma}\right), \quad a \le b. && \text{(A.37)}
\end{aligned}$$

Here the last equality follows because $(X-\mu)/\sigma$ is a standard Gaussian random variable, see (A.24).

The Q-function is usually only tabulated for nonnegative arguments. The reason has to do with the symmetry of the standard Gaussian density (A.1). If $W \sim \mathcal{N}(0,1)$ then, by the symmetry of its density,

$$\begin{aligned}
\Pr[W \ge -\alpha] &= \Pr[W \le \alpha] && \text{(A.38)}\\
&= 1 - \Pr[W > \alpha] && \text{(A.39)}\\
&= 1 - \Pr[W \ge \alpha]; && \text{(A.40)}
\end{aligned}$$

see Figure A.3. Consequently,

$$Q(\alpha) + Q(-\alpha) = 1, \quad \alpha \in \mathbb{R}, \tag{A.41}$$

and it suffices to tabulate the Q-function for nonnegative arguments. Note also that from the above it follows that $Q(0) = 1/2$.

There is an alternative expression for the Q-function, that can be quite useful. It is an integral expression with fixed integration limits:

$$Q(\alpha) = \frac{1}{\pi} \int_0^{\pi/2} e^{-\frac{\alpha^2}{2\sin^2\theta}} \,\mathrm{d}\theta, \quad \alpha \geq 0. \tag{A.42}$$

This expression can be derived as follows. One computes a two dimensional integral in two different ways. Let $X \sim \mathcal{N}(0,1)$ and $Y \sim \mathcal{N}(0,1)$ be independent random variables. Consider the probability of the event $X \geq 0$ and $Y \geq \alpha$ where $\alpha > 0$. Since the two random variables are independent, it follows

$$\Pr[X \geq 0 \text{ and } Y \geq \alpha] = \Pr[X \geq 0] \cdot \Pr[Y \geq \alpha] \tag{A.43}$$

$$= \frac{1}{2} Q(\alpha). \tag{A.44}$$

We shall now proceed to compute the left-hand side of the above in polar coordinates centered at the origin (see Figure A.4). This yields

$$\frac{1}{2} Q(\alpha) = \int_0^{\infty} \int_{\alpha}^{\infty} \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}} \,\mathrm{d}y \,\mathrm{d}x \tag{A.45}$$

$$= \int_0^{\pi/2} \int_{\frac{\alpha}{\sin\theta}}^{\infty} \frac{1}{2\pi} e^{-\frac{r^2}{2}} \, r \,\mathrm{d}r \,\mathrm{d}\theta \tag{A.46}$$

$$= \frac{1}{2\pi} \int_0^{\pi/2} \int_{\frac{\alpha^2}{2\sin^2\theta}}^{\infty} e^{-t} \,\mathrm{d}t \,\mathrm{d}\theta \tag{A.47}$$

$$= \frac{1}{2\pi} \int_0^{\pi/2} e^{-\frac{\alpha^2}{2\sin^2\theta}} \,\mathrm{d}\theta, \tag{A.48}$$

where we have performed the change of variable $t = r^2/2$.

We collect some of the properties of the Q-function in the following proposition.

Proposition A.7. Let $W \sim \mathcal{N}(0,1)$ and let $X \sim \mathcal{N}(\mu, \sigma^2)$ for $\sigma > 0$.

1. The CDF of W is given in terms of the Q-function as

$$F_W(w) = 1 - Q(w). \tag{A.49}$$

2. The probability of a half-ray can be expressed as

$$\Pr[X > \alpha] = Q\left(\frac{\alpha - \mu}{\sigma}\right). \tag{A.50}$$

3. The symmetry of the standard Gaussian distribution implies that it suffices to tabulate the Q-function for nonnegative arguments:

$$Q(-\alpha) + Q(\alpha) = 1. \tag{A.51}$$

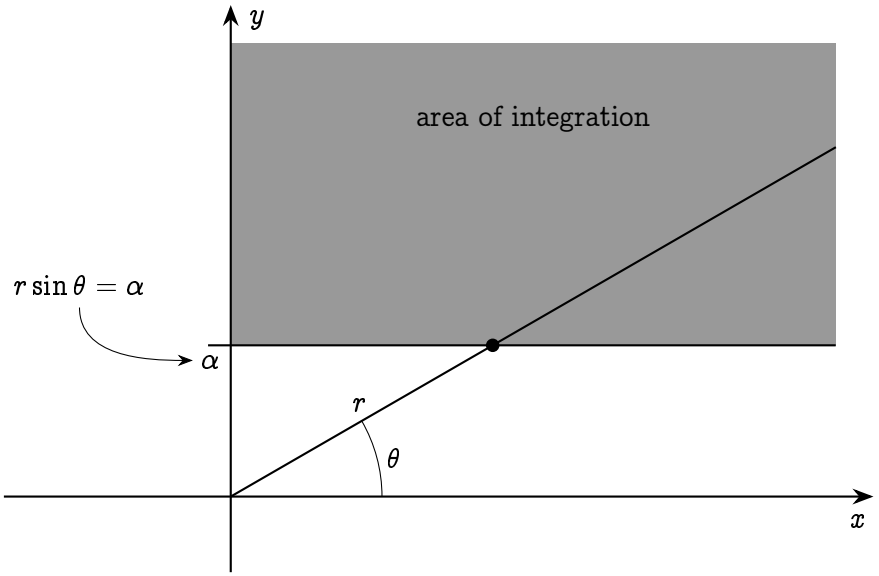


Figure A.4: Computing $\frac{1}{2}\,Q(\alpha)$ by the use of polar coordinates.

4. In particular, the probability that W is positive is given by

$$Q(0)=\frac{1}{2}. \tag{A.52}$$

We next describe various approximations for the Q-function. We are particularly interested in its value for large arguments[2]. Since $Q(\alpha)$ is the probability that a standard Gaussian random variable W exceeds α, it follows that

$$\lim_{\alpha\to\infty} Q(\alpha)=0. \tag{A.53}$$

Thus, large arguments to the Q-function correspond to small values of the Q-function. The following bound will justify the approximation

$$Q(\alpha)\approx e^{-\frac{\alpha^2}{2}},\quad \alpha\gg 1. \tag{A.54}$$

Proposition A.8. The Q-function can be bounded as follows:

$$\frac{1}{\sqrt{2\pi}\alpha}\,e^{-\frac{\alpha^2}{2}}\left(1-\frac{1}{\alpha^2}\right)<Q(\alpha)<\frac{1}{\sqrt{2\pi}\alpha}\,e^{-\frac{\alpha^2}{2}},\quad \alpha>0, \tag{A.55}$$

$$\frac{1}{\sqrt{2\pi}\alpha}\,e^{-\frac{\alpha^2}{2}}\left(\frac{\alpha^2}{1+\alpha^2}\right)<Q(\alpha),\quad \alpha\in\mathbb{R}, \tag{A.56}$$

and

$$Q(\alpha)\le\frac{1}{2}\,e^{-\frac{\alpha^2}{2}},\quad \alpha\ge 0. \tag{A.57}$$

[2]This corresponds in many digital communication applications to low probabilities of bit errors.

Moreover, the following bounds are less tight for $|\alpha| \gg 1$, but tighter around $\alpha = 0$ (and also valid for negative α):

$$\max\left\{1 - \frac{1}{2} e^{\sqrt{\frac{2}{\pi}}\alpha}, 0\right\} \le \mathcal{Q}(\alpha) \le \min\left\{\frac{1}{2} e^{-\sqrt{\frac{2}{\pi}}\alpha}, 1\right\}, \quad \alpha \in \mathbb{R}. \tag{A.58}$$

The bounds of Proposition A.8 are depicted in Figure A.5.

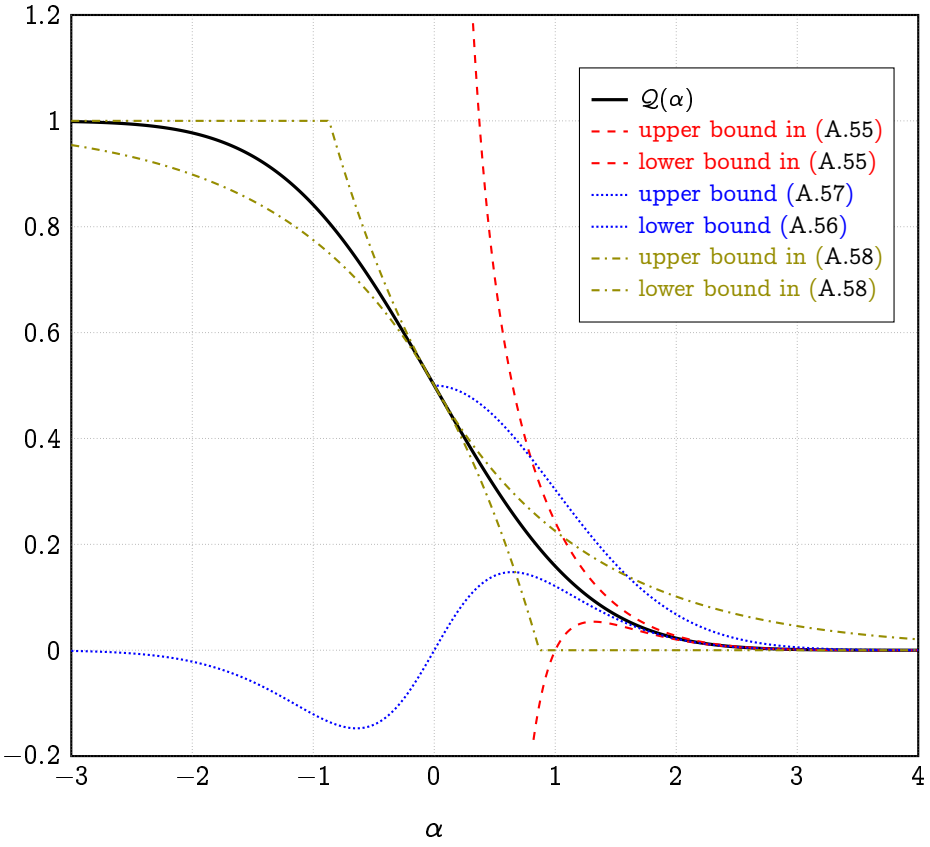


Figure A.5: Various upper and lower bounds on the $\mathcal{Q}$-function according to Proposition A.8.

Proof: Using the substitution $t = x^2/2$, the upper bound in (A.55) can be derived as follows:

$$\begin{aligned}
\mathcal{Q}(\alpha) &= \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \,\mathrm{d}x && \text{(A.59)}\\
&< \int_{\alpha}^{\infty} \frac{x}{\alpha} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \,\mathrm{d}x \qquad (\alpha > 0) && \text{(A.60)}\\
&= \int_{\alpha^2/2}^{\infty} \frac{1}{\sqrt{2\pi}\alpha} e^{-t} \,\mathrm{d}t && \text{(A.61)}\\
&= \frac{1}{\sqrt{2\pi}\alpha} e^{-\frac{\alpha^2}{2}}. && \text{(A.62)}
\end{aligned}$$

For the lower bound (A.56), firstly define

$$\varphi(x) \triangleq \frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}} \tag{A.63}$$

and note that

$$\frac{\mathrm{d}}{\mathrm{d}x}\varphi(x) = -x\frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}} = -x\varphi(x) \tag{A.64}$$

and therefore

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\varphi(x)}{x}\right) = \frac{x\frac{\mathrm{d}}{\mathrm{d}x}\varphi(x) - \varphi(x)}{x^2} \tag{A.65}$$

$$= \frac{-x^2\varphi(x) - \varphi(x)}{x^2} \tag{A.66}$$

$$= -\frac{x^2+1}{x^2}\varphi(x). \tag{A.67}$$

So, for $\alpha > 0$, we get

$$\left(1+\frac{1}{\alpha^2}\right) Q(\alpha) = \int_\alpha^\infty \left(1+\frac{1}{\alpha^2}\right)\frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}}\,\mathrm{d}x \tag{A.68}$$

$$> \int_\alpha^\infty \left(1+\frac{1}{x^2}\right)\frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}}\,\mathrm{d}x \tag{A.69}$$

$$= \int_\alpha^\infty \frac{x^2+1}{x^2}\varphi(x)\,\mathrm{d}x \tag{A.70}$$

$$= -\frac{\varphi(x)}{x}\bigg|_\alpha^\infty \tag{A.71}$$

$$= \frac{\varphi(\alpha)}{\alpha} = \frac{1}{\sqrt{2\pi}\alpha}\, e^{-\frac{\alpha^2}{2}}. \tag{A.72}$$

For $\alpha \le 0$, we note that the left-hand side of (A.56) is nonpositive and therefore trivially a (alas quite loose) lower bound.

The lower bound in (A.55) is strictly smaller than the lower bound (A.56) and therefore implicitly satisfied.

The upper bound (A.57) follows by replacing the integrand in (A.42) with its maximal value, namely, its value at $\theta = \pi/2$.

The proof of (A.58) is omitted. □

A.4 Characteristic Function of a Gaussian

Definition A.9. The *characteristic function* $\Phi_X(\omega)$ of a random variable X is defined for every $\omega \in \mathbb{R}$ by

$$\Phi_X(\omega) \triangleq \mathsf{E}\left[e^{\mathrm{i}\omega X}\right] = \int_{-\infty}^{\infty} f_X(x)\, e^{\mathrm{i}\omega x}\,\mathrm{d}x, \tag{A.73}$$

where the second equality holds if X has a density.

The characteristic function is intimately related to the Fourier transform of the density function. Some things to recall about that characteristic function are given in the following proposition.

Proposition A.10. Let X be a random variable of characteristic function $\Phi_X(\omega)$.

1. If $\mathsf{E}[X^n] < \infty$ for some $n \geq 1$, then the rth order derivative $\Phi_X^{(r)}(\omega)$ exists for every $r \leq n$ and
$$\mathsf{E}[X^r] = \frac{\Phi_X^{(r)}(0)}{\mathrm{i}^r}. \tag{A.74}$$
2. If X and Y are two random variables of identical characteristic functions, then they have the same distribution.
3. If X and Y are two independent random variables of characteristic functions $\Phi_X(\omega)$ and $\Phi_Y(\omega)$ respectively, then the characteristic function $\Phi_{X+Y}(\omega)$ of the sum $X+Y$ is given by the product of the individual characteristic functions:
$$\Phi_{X+Y}(\omega) = \Phi_X(\omega) \cdot \Phi_Y(\omega), \quad X \perp\!\!\!\perp Y. \tag{A.75}$$

If $X \sim \mathcal{N}(\mu, \sigma^2)$, then it can be verified by direct computation that

$$\Phi_X(\omega) = e^{\mathrm{i}\omega\mu - \frac{1}{2}\omega^2\sigma^2}. \tag{A.76}$$

Consequently, by repeatedly taking the derivatives of the above, we obtain from (A.74) the moments of a standard Gaussian distribution:

$$\mathsf{E}[W^\nu] = \begin{cases} 1 \cdot 3 \cdots (\nu-1) & \text{if } \nu \text{ is even,} \\ 0 & \text{if } \nu \text{ is odd,} \end{cases} \quad W \sim \mathcal{N}(0,1). \tag{A.77}$$

We mention here also that

$$\mathsf{E}[|W|^\nu] = \begin{cases} 1 \cdot 3 \cdots (\nu-1) & \text{if } \nu \text{ is even,} \\ \sqrt{\frac{2}{\pi}} \cdot 2^{(\nu-1)/2} \cdot \left(\frac{\nu-1}{2}\right)! & \text{if } \nu \text{ is odd,} \end{cases} \quad W \sim \mathcal{N}(0,1). \tag{A.78}$$

Closely related to the characteristic function is the *moment generating function* $M_X(\theta)$ of a random variable X, which is defined as

$$M_X(\theta) \triangleq \mathsf{E}\left[e^{\theta X}\right] = \int_{-\infty}^{\infty} f_X(x)\, e^{\theta x}\, \mathrm{d}x \tag{A.79}$$

for those values of θ for which the expectation is finite. Here the second equality holds if X has a density. The moment generating function is intimately related to the double-sided Laplace transform of the density function.

Using the characteristic function we can readily show the following.

Proposition A.11. If $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$ are two independent Gaussian random variables, then also their sum $X + Y$ is Gaussian:

$$X + Y \sim \mathcal{N}\Big(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2\Big). \tag{A.80}$$

Proof: Since $X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$ it follows from (A.76) that

$$\Phi_X(\omega) = e^{\mathrm{i}\omega\mu_x - \frac{1}{2}\omega^2\sigma_x^2}, \tag{A.81}$$
$$\Phi_Y(\omega) = e^{\mathrm{i}\omega\mu_y - \frac{1}{2}\omega^2\sigma_y^2}. \tag{A.82}$$

Since the characteristic function of the sum of two independent random variables is equal to the product of the individual characteristic functions, we have

$$\Phi_{X+Y}(\omega) = \Phi_X(\omega) \cdot \Phi_Y(\omega) \tag{A.83}$$
$$= e^{\mathrm{i}\omega\mu_x - \frac{1}{2}\omega^2\sigma_x^2} \cdot e^{\mathrm{i}\omega\mu_y - \frac{1}{2}\omega^2\sigma_y^2} \tag{A.84}$$
$$= e^{\mathrm{i}\omega(\mu_x+\mu_y) - \frac{1}{2}\omega^2(\sigma_x^2+\sigma_y^2)}. \tag{A.85}$$

But this is also the characteristic function of a $\mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$ random variable. Since two random variables can have the same characteristic function only if they have the same distribution, we conclude that $X + Y \sim \mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$. □

Using the fact that a deterministic affine transformation of a Gaussian random variable results in a Gaussian random variable and the above proposition, we obtain the following.

Proposition A.12. If $X_1, \ldots, X_n$ are independent Gaussian random variables, and $\alpha_1, \ldots, \alpha_n$ are deterministic real numbers, then the random variable

$$Y \triangleq \sum_{\ell=1}^{n} \alpha_\ell X_\ell \tag{A.86}$$

is a Gaussian random variable with mean

$$\mathsf{E}[Y] = \sum_{\ell=1}^{n} \alpha_\ell \, \mathsf{E}[X_\ell] \tag{A.87}$$

and variance

$$\mathsf{Var}[Y] = \sum_{\ell=1}^{n} \alpha_\ell^2 \, \mathsf{Var}[X_\ell]. \tag{A.88}$$

A.5 Summary

Top things to remember about Gaussian random variables are as follows:

- There is only one Gaussian distribution of a given mean μ and variance σ^2. It is denoted $\mathcal{N}(\mu, \sigma^2)$ and it is either deterministic (if $\sigma^2 = 0$) or of density

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbb{R}. \tag{A.89}$$

- If X has a Gaussian distribution, then any deterministic affine transformation of X is also Gaussian. That is, if X is Gaussian and $\alpha, \beta \in \mathbb{R}$ are deterministic, then

$$\alpha X + \beta \tag{A.90}$$

  is also Gaussian.

- If $X \sim \mathcal{N}(\mu, \sigma^2)$ and $\sigma^2 > 0$, then $(X - \mu)/\sigma$ has a standard Gaussian distribution, i.e.,

$$\frac{X - \mu}{\sigma} \sim \mathcal{N}(0, 1), \quad \sigma^2 > 0. \tag{A.91}$$

- The Q-function is defined as

$$Q(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-\frac{\xi^2}{2}} \, \mathrm{d}\xi. \tag{A.92}$$

  It can be used to compute the probability that a Gaussian random variable lies within an interval.

- The sum of any finite number of independent Gaussian random variables is also a Gaussian random variable. The sum's mean is the sum of the means, and the sum's variance is the sum of the variances.

Appendix B

Gaussian Vectors

Before we can introduce the *multivariate Gaussian distribution*[1] or *Gaussian vectors*, we need to make a short detour and discuss some special types of matrices, so-called *positive semidefinite matrices*, and random vectors in general.

B.1 Positive Semidefinite Matrices

We will see that positive semidefinite matrices are a special kind of symmetric matrices. Therefore, before we state their definition, we quickly review some properties of symmetric matrices.

Recall that a real matrix A is said to be *symmetric* if

$$\mathsf{A} = \mathsf{A}^{\mathsf{T}}. \tag{B.1}$$

Obviously, a symmetric matrix must be square. Next, we remind the reader that square matrices can be characterized using the concept of *eigenvectors* and *eigenvalues*. An *eigenvector* is a vector $\mathbf{x}$ with a special direction particular to the given square matrix such that when it is multiplied with the matrix, it again results in a vector of the same direction (but possibly of different length):

$$\mathsf{A}\mathbf{x} = \lambda\mathbf{x}. \tag{B.2}$$

The scaling factor λ is called *eigenvalue*.

We also remind the reader that even a *real* matrix can have *complex* eigenvalues.[2] For symmetric matrices, however, things are much nicer.

[1] *Multivariate distribution* is just a fancy word for *joint distribution* of several random variables. It contrasts *univariate distribution*, which means the distribution of a single random variable.

[2] At least it is not difficult to show that the eigenvalues of real matrices always come in complex conjugate pairs.

Lemma B.1. All eigenvalues of a (real) symmetric matrix are real (and therefore the corresponding eigenvectors can be chosen to be real, too). Moreover, the eigenvectors of a symmetric matrix that belong to different eigenvalues are always perpendicular.

Proof: Suppose that for a symmetric A, (B.2) holds where for the moment we need to allow λ and $\mathbf{x}$ to be complex. Now multiply (B.2) by $\mathbf{x}^\dagger$ from the left:[3]

$$\mathbf{x}^\dagger A\mathbf{x} = \mathbf{x}^\dagger \lambda \mathbf{x} = \lambda \mathbf{x}^\dagger \mathbf{x} = \lambda \|\mathbf{x}\|^2. \tag{B.3}$$

Here $\|\mathbf{x}\|^2 > 0$ because an eigenvector cannot be the zero vector by definition. On the other hand, taking the Hermitian conjugate of (B.2) and multiplying both sides by $\mathbf{x}$ from the right yields:

$$\mathbf{x}^\dagger A^\dagger \mathbf{x} = \mathbf{x}^\dagger A^{\mathsf{T}} \mathbf{x} = \mathbf{x}^\dagger A\mathbf{x} \stackrel{!}{=} \mathbf{x}^\dagger \lambda^\dagger \mathbf{x} = \mathbf{x}^\dagger \lambda^* \mathbf{x} = \lambda^* \mathbf{x}^\dagger \mathbf{x} = \lambda^* \|\mathbf{x}\|^2, \tag{B.4}$$

where we have used that A is real and symmetric. Comparing (B.3) and (B.4) now shows that

$$\lambda = \lambda^*, \tag{B.5}$$

i.e., $\lambda \in \mathbb{R}$. This proves the first claim.

To prove the second claim, suppose that $\mathbf{x}_1$ and $\mathbf{x}_2$ are two eigenvectors of A belonging to two eigenvalues λ_1 and λ_2, respectively, where we assume that $\lambda_1 \neq \lambda_2$. Since A and λ_i are real, we can assume that $\mathbf{x}_i$ are also real. We now have the following:

$$\begin{aligned}
\lambda_1 \mathbf{x}_1^{\mathsf{T}} \mathbf{x}_2 &= (\lambda_1 \mathbf{x}_1)^{\mathsf{T}} \mathbf{x}_2 && \text{(B.6)}\\
&= (A\mathbf{x}_1)^{\mathsf{T}} \mathbf{x}_2 && \text{(B.7)}\\
&= \mathbf{x}_1^{\mathsf{T}} A^{\mathsf{T}} \mathbf{x}_2 && \text{(B.8)}\\
&= \mathbf{x}_1^{\mathsf{T}} A\mathbf{x}_2 && \text{(B.9)}\\
&= \mathbf{x}_1^{\mathsf{T}} (A\mathbf{x}_2) && \text{(B.10)}\\
&= \mathbf{x}_1^{\mathsf{T}} \lambda_2 \mathbf{x}_2 && \text{(B.11)}\\
&= \lambda_2 \mathbf{x}_1^{\mathsf{T}} \mathbf{x}_2, && \text{(B.12)}
\end{aligned}$$

where we again have used the symmetry $A = A^{\mathsf{T}}$. Since $\lambda_1 \neq \lambda_2$ by assumption, we must have that

$$\mathbf{x}_1^{\mathsf{T}} \mathbf{x}_2 = 0, \tag{B.13}$$

proving that the eigenvectors are orthogonal. □

[3] Note that by the Hermitian conjugation $\mathbf{x}^\dagger$ we denote the transpose and complex conjugate version of the vector $\mathbf{x}$, i.e., $\mathbf{x}^\dagger = (\mathbf{x}^*)^{\mathsf{T}}$.

Note that Lemma B.1 can be generalized further to prove that a real symmetric $n \times n$ matrix always has n eigenvectors that can be chosen to be real and orthogonal (even if several eigenvectors belong to the same eigenvalue). For more details see, e.g., [Str09].

We are now ready for the definition of positive (semi-)definite matrices.

Definition B.2. We say that an $n \times n$ real matrix K is *positive semidefinite* and write

$$\mathsf{K} \succeq 0 \tag{B.14}$$

if it is symmetric and if

$$\boldsymbol{\alpha}^{\mathsf{T}} \mathsf{K} \boldsymbol{\alpha} \geq 0, \quad \forall\, \boldsymbol{\alpha} \in \mathbb{R}^n. \tag{B.15}$$

We say that a matrix K is *positive definite* and write

$$\mathsf{K} \succ 0 \tag{B.16}$$

if it is symmetric and if

$$\boldsymbol{\alpha}^{\mathsf{T}} \mathsf{K} \boldsymbol{\alpha} > 0, \quad \forall\, \boldsymbol{\alpha} \neq \mathbf{0}. \tag{B.17}$$

Positive semidefinite matrices have nice properties as is shown in the following proposition.

Proposition B.3 (Properties of Positive Semidefinite Matrices). A symmetric $n \times n$ real matrix K is positive semidefinite if, and only if, one (and hence all) of the following equivalent conditions holds:

1. All the eigenvalues of K are nonnegative.
2. The matrix K can be written in the form $\mathsf{K} = \mathsf{U}\Lambda\mathsf{U}^{\mathsf{T}}$ where the matrix Λ is diagonal with nonnegative entries on the diagonal and where the matrix U is orthogonal, i.e., it satisfies $\mathsf{U}\mathsf{U}^{\mathsf{T}} = \mathsf{I}_n$.
3. The matrix K can be written in the form $\mathsf{K} = \mathsf{S}^{\mathsf{T}}\mathsf{S}$ for some $n \times n$ matrix S.
4. One possible choice of S is
$$\mathsf{S} = \Lambda^{1/2}\mathsf{U}^{\mathsf{T}}. \tag{B.18}$$

The analogous proposition of positive definite matrices is as follows.

Proposition B.4 (Properties of Positive Definite Matrices). A symmetric $n \times n$ real matrix K is positive definite if, and only if, one (and hence all) of the following equivalent conditions holds:

1. All the eigenvalues of K are positive.

2. The matrix K can be written in the form $\mathsf{K} = \mathsf{U}\Lambda\mathsf{U}^{\mathsf{T}}$ where the matrix Λ is diagonal with positive diagonal entries and where the matrix U is orthogonal, i.e., it satisfies $\mathsf{U}\mathsf{U}^{\mathsf{T}} = \mathsf{I}_n$.

3. The matrix K can be written in the form $\mathsf{K} = \mathsf{S}^{\mathsf{T}}\mathsf{S}$ for some nonsingular $n \times n$ matrix S.

4. One possible choice of S is

$$\mathsf{S} = \Lambda^{1/2}\mathsf{U}^{\mathsf{T}}. \tag{B.19}$$

Proof: The second statement follows directly from the fact that positive (semi-)definite matrices are symmetric and hence have n real and orthogonal eigenvectors $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n$: we define

$$\mathsf{U} \triangleq \begin{pmatrix} \uparrow & \cdots & \uparrow \\ & \cdots & \\ \boldsymbol{\psi}_1 & \cdots & \boldsymbol{\psi}_n \\ & \cdots & \\ \downarrow & \cdots & \downarrow \end{pmatrix} \tag{B.20}$$

and note that it is orthogonal:

$$\mathsf{U}\mathsf{U}^{\mathsf{T}} = \mathsf{U}^{\mathsf{T}}\mathsf{U} = \mathsf{I}_n. \tag{B.21}$$

Further, we define the diagonal matrix Λ

$$\Lambda \triangleq \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{pmatrix} \tag{B.22}$$

where λ_i denote the n eigenvalues (possibly some of them are identical). Then we have

$$\mathsf{K}\mathsf{U} = \mathsf{K}\begin{pmatrix} \uparrow & \cdots & \uparrow \\ & \cdots & \\ \boldsymbol{\psi}_1 & \cdots & \boldsymbol{\psi}_n \\ & \cdots & \\ \downarrow & \cdots & \downarrow \end{pmatrix} \tag{B.23}$$

$$= \begin{pmatrix} \uparrow & \cdots & \uparrow \\ & \cdots & \\ \mathsf{K}\boldsymbol{\psi}_1 & \cdots & \mathsf{K}\boldsymbol{\psi}_n \\ & \cdots & \\ \downarrow & \cdots & \downarrow \end{pmatrix} \tag{B.24}$$

$$= \begin{pmatrix} \uparrow & \cdots & \uparrow \\ & \cdots & \\ \lambda_1\boldsymbol{\psi}_1 & \cdots & \lambda_n\boldsymbol{\psi}_n \\ & \cdots & \\ \downarrow & \cdots & \downarrow \end{pmatrix} \tag{B.25}$$

$$= \begin{pmatrix} \uparrow & \cdots & \uparrow \\ & \cdots & \\ \boldsymbol{\psi}_1 & \cdots & \boldsymbol{\psi}_n \\ & \cdots & \\ \downarrow & \cdots & \downarrow \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{pmatrix} \tag{B.26}$$

$$= \mathsf{U}\Lambda, \tag{B.27}$$

and hence, using (B.21),

$$\mathsf{K}\mathsf{U}\mathsf{U}^{\mathsf{T}} = \mathsf{K} = \mathsf{U}\Lambda\mathsf{U}^{\mathsf{T}}. \tag{B.28}$$

To prove the first statement note that by definition (B.15), for the choice $\boldsymbol{\alpha} = \boldsymbol{\psi}_\nu$,

$$0 \le \boldsymbol{\psi}_\nu^{\mathsf{T}}\mathsf{K}\boldsymbol{\psi}_\nu = \boldsymbol{\psi}_\nu^{\mathsf{T}}(\mathsf{K}\boldsymbol{\psi}_\nu) = \boldsymbol{\psi}_\nu^{\mathsf{T}}(\lambda_\nu\boldsymbol{\psi}_\nu) = \lambda_\nu \underbrace{\boldsymbol{\psi}_\nu^{\mathsf{T}}\boldsymbol{\psi}_\nu}_{=1} = \lambda_\nu. \tag{B.29}$$

For positive definite matrices the inequality is strict.

To prove the remaining two statements, note that because of the nonnegativity of the eigenvalues, we can define

$$\Lambda^{1/2} \triangleq \begin{pmatrix} \sqrt{\lambda_1} & 0 & \cdots & 0 \\ 0 & \sqrt{\lambda_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sqrt{\lambda_n} \end{pmatrix} \tag{B.30}$$

and

$$\mathsf{S} \triangleq \Lambda^{1/2}\mathsf{U}^{\mathsf{T}}. \tag{B.31}$$

Note that for positive definite matrices, all eigenvalues are strictly positive and therefore S is nonsingular. Then

$$\begin{aligned}\mathsf{S}^{\mathsf{T}}\mathsf{S} &= \left(\Lambda^{1/2}\mathsf{U}^{\mathsf{T}}\right)^{\mathsf{T}}\Lambda^{1/2}\mathsf{U}^{\mathsf{T}} && \text{(B.32)}\\ &= \mathsf{U}\Lambda^{1/2}\Lambda^{1/2}\mathsf{U}^{\mathsf{T}} && \text{(B.33)}\\ &= \mathsf{U}\Lambda\mathsf{U}^{\mathsf{T}} && \text{(B.34)}\\ &= \mathsf{K}. && \text{(B.35)}\end{aligned}$$

Here the last equality follows from (B.28). □

As a consequence of Propositions B.3 and B.4 (and of Definition B.2) we have the following two corollaries.

Corollary B.5. If K is a real $n \times n$ positive semidefinite matrix and if $\boldsymbol{\alpha} \in \mathbb{R}^n$ is arbitrary, then $\boldsymbol{\alpha}^{\mathsf{T}}\mathsf{K}\boldsymbol{\alpha} = 0$ if, and only if, $\mathsf{K}\boldsymbol{\alpha} = \mathbf{0}$.

Proof: One direction is trivial and does not require that K be positive semidefinite: If $\mathsf{K}\boldsymbol{\alpha} = \mathbf{0}$ then $\boldsymbol{\alpha}^{\mathsf{T}}\mathsf{K}\boldsymbol{\alpha}$ must also be equal to zero. Indeed, in this case we have by the associativity of matrix multiplication $\boldsymbol{\alpha}^{\mathsf{T}}\mathsf{K}\boldsymbol{\alpha} = \boldsymbol{\alpha}^{\mathsf{T}}(\mathsf{K}\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{0} = 0$.

As to the other direction we note that since $\mathsf{K} \succeq 0$ it follows that there exists some $n \times n$ matrix S such that $\mathsf{K} = \mathsf{S}^{\mathsf{T}}\mathsf{S}$. Hence,

$$\begin{aligned}\boldsymbol{\alpha}^{\mathsf{T}}\mathsf{K}\boldsymbol{\alpha} &= \boldsymbol{\alpha}^{\mathsf{T}}\mathsf{S}^{\mathsf{T}}\mathsf{S}\boldsymbol{\alpha} && \text{(B.36)}\\ &= (\mathsf{S}\boldsymbol{\alpha})^{\mathsf{T}}(\mathsf{S}\boldsymbol{\alpha}) && \text{(B.37)}\\ &= \|\mathsf{S}\boldsymbol{\alpha}\|^2. && \text{(B.38)}\end{aligned}$$

Consequently, $\boldsymbol{\alpha}^{\mathsf{T}}\mathsf{K}\boldsymbol{\alpha} = 0$ implies that $\mathsf{S}\boldsymbol{\alpha} = \mathbf{0}$, which implies that $\mathsf{S}^{\mathsf{T}}\mathsf{S}\boldsymbol{\alpha} = \mathbf{0}$, i.e., that $\mathsf{K}\boldsymbol{\alpha} = \mathbf{0}$. □

Corollary B.6. If K is a real $n \times n$ positive definite matrix then $\boldsymbol{\alpha}^{\mathsf{T}}\mathsf{K}\boldsymbol{\alpha} = 0$ if, and only if, $\boldsymbol{\alpha} = \mathbf{0}$.

Proof: This follows immediately from Definition B.2. □

B.2 Random Vectors and Covariance Matrices

A random vector is a quite straightforward generalization of a random variable.

Definition B.7. An *n-dimensional random vector* (or also called a *random n-vector*) $\mathbf{X}$ is a (measurable) mapping from the set of experiment outcomes Ω to the n-dimensional Euclidean space $\mathbb{R}^n$:

$$\mathbf{X}\colon \Omega \to \mathbb{R}^n,\ \omega \mapsto \mathbf{x}. \tag{B.39}$$

Thus, a random vector $\mathbf{X}$ is very much like a random variable, except that rather than taking value in the real line $\mathbb{R}$, it takes value in $\mathbb{R}^n$.

The density of a random vector is the joint density of its components. The density of a random n-vector is thus a real valued function from $\mathbb{R}^n$ to the nonnegative reals that integrates (over $\mathbb{R}^n$) to one.

Definition B.8. The *expectation* $\mathsf{E}[\mathbf{X}]$ of a random n-vector

$$\mathbf{X} = (X_1, \ldots, X_n)^{\mathsf{T}} \tag{B.40}$$

is a vector whose components are the expectations of the corresponding elements of $\mathbf{X}$:

$$\mathsf{E}[\mathbf{X}] \triangleq \begin{pmatrix} \mathsf{E}[X_1] \\ \vdots \\ \mathsf{E}[X_n] \end{pmatrix}. \tag{B.41}$$

Hence, we see that the expectation of a random vector is only defined if the expectations of all of the components of the vector are defined. The jth element of $\mathsf{E}[\mathbf{X}]$ is thus the expectation of the jth component of $\mathbf{X}$, namely, $\mathsf{E}[X_j]$. Similarly, the expectation of a random matrix is the matrix of expectations.

If all the components of a random n-vector $\mathbf{X}$ are of finite variance, then we can define its $n \times n$ covariance matrix $\mathsf{K}_{\mathbf{XX}}$ as follows.

Definition B.9. The $n \times n$ *covariance matrix* $\mathsf{K}_{\mathbf{XX}}$ of $\mathbf{X}$ is defined as

$$\mathsf{K}_{\mathbf{XX}} \triangleq \mathsf{E}\Big[(\mathbf{X} - \mathsf{E}[\mathbf{X}])(\mathbf{X} - \mathsf{E}[\mathbf{X}])^{\mathsf{T}}\Big]. \tag{B.42}$$

That is,

$$\mathsf{K}_{\mathbf{XX}} = \mathsf{E}\left[\begin{pmatrix} X_1 - \mathsf{E}[X_1] \\ \vdots \\ X_n - \mathsf{E}[X_n] \end{pmatrix} \begin{pmatrix} X_1 - \mathsf{E}[X_1] & \cdots & X_n - \mathsf{E}[X_n] \end{pmatrix}\right] \tag{B.43}$$

$$= \begin{pmatrix} \mathsf{Var}[X_1] & \mathsf{Cov}[X_1, X_2] & \cdots & \mathsf{Cov}[X_1, X_n] \\ \mathsf{Cov}[X_2, X_1] & \mathsf{Var}[X_2] & \cdots & \mathsf{Cov}[X_2, X_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{Cov}[X_n, X_1] & \mathsf{Cov}[X_n, X_2] & \cdots & \mathsf{Var}[X_n] \end{pmatrix}. \tag{B.44}$$

If $n = 1$ so that the n-dimensional random vector $\mathbf{X}$ is actually a scalar, then the covariance matrix K_{XX} is a 1×1 matrix whose sole component is the variance of the sole component of $\mathbf{X}$.

Note that given the $n \times n$ covariance matrix $\mathsf{K}_{\mathbf{XX}}$ of a random n-vector $\mathbf{X}$, it is easy to compute the covariance matrix of a subset of its components. For

example, if we are only interested in the 2×2 covariance matrix of $(X_1, X_2)^\mathsf{T}$, all we have to do is pick the first two columns and two rows of $\mathsf{K}_{\mathbf{XX}}$. More generally, the $r \times r$ covariance matrix of $(X_{j_1}, X_{j_2}, \dots, X_{j_r})^\mathsf{T}$ for $1 \le j_1 < j_2 < \cdots < j_r \le n$ is obtained from $\mathsf{K}_{\mathbf{XX}}$ by picking rows and columns $j_1, \dots, j_r$. For example,[4] if

$$\mathsf{K}_{\mathbf{XX}} = \begin{pmatrix} 30 & 31 & 9 & 7 \\ 31 & 39 & 11 & 13 \\ 9 & 11 & 9 & 12 \\ 7 & 13 & 12 & 26 \end{pmatrix}, \tag{B.45}$$

then the covariance matrix of $(X_2, X_4)^\mathsf{T}$ is

$$\begin{pmatrix} 39 & 13 \\ 13 & 26 \end{pmatrix}. \tag{B.46}$$

Remark B.10. At this point we would like to remind the reader that matrix multiplication is a linear transformation and therefore commutes with the expectation operation. Thus if $\mathbb{H}$ is a random $n \times m$ matrix and A is a deterministic $\ell \times n$ matrix, then

$$\mathsf{E}[\mathsf{A}\mathbb{H}] = \mathsf{A}\,\mathsf{E}[\mathbb{H}], \tag{B.47}$$

and similarly if B is a deterministic $m \times \nu$ matrix then

$$\mathsf{E}[\mathbb{H}\mathsf{B}] = \mathsf{E}[\mathbb{H}]\,\mathsf{B}. \tag{B.48}$$

Also the transpose operation commutes with expectation: If $\mathbb{H}$ is a random matrix then

$$\mathsf{E}\big[\mathbb{H}^\mathsf{T}\big] = \big(\mathsf{E}[\mathbb{H}]\big)^\mathsf{T}. \tag{B.49}$$

△

We next prove the following important lemma.

[4] The alert reader might notice that this matrix is positive semidefinite: It can be written as $\mathsf{S}^\mathsf{T}\mathsf{S}$ where

$$\mathsf{S} = \begin{pmatrix} 1 & 2 & 2 & 5 \\ 2 & 3 & 2 & 1 \\ 3 & 1 & 1 & 0 \\ 4 & 5 & 0 & 0 \end{pmatrix}.$$

This is no accident as we will see soon.

Lemma B.11. Let $\mathbf{X}$ be a random n-vector with covariance matrix $\mathsf{K}_{\mathbf{XX}}$ and let A be some (deterministic) $m \times n$ matrix. Define the random m-vector $\mathbf{Y}$ as

$$\mathbf{Y} \triangleq \mathsf{A}\mathbf{X}. \tag{B.50}$$

Then the $m \times m$ covariance matrix $\mathsf{K}_{\mathbf{YY}}$ is given by

$$\mathsf{K}_{\mathbf{YY}} = \mathsf{A}\,\mathsf{K}_{\mathbf{XX}}\,\mathsf{A}^{\mathsf{T}}. \tag{B.51}$$

Proof: This follows from

$$\begin{aligned}
\mathsf{K}_{\mathbf{YY}} &\triangleq \mathsf{E}\big[(\mathbf{Y}-\mathsf{E}[\mathbf{Y}])(\mathbf{Y}-\mathsf{E}[\mathbf{Y}])^{\mathsf{T}}\big] && \text{(B.52)}\\
&= \mathsf{E}\big[(\mathsf{A}\mathbf{X}-\mathsf{E}[\mathsf{A}\mathbf{X}])(\mathsf{A}\mathbf{X}-\mathsf{E}[\mathsf{A}\mathbf{X}])^{\mathsf{T}}\big] && \text{(B.53)}\\
&= \mathsf{E}\big[\mathsf{A}(\mathbf{X}-\mathsf{E}[\mathbf{X}])(\mathsf{A}(\mathbf{X}-\mathsf{E}[\mathbf{X}]))^{\mathsf{T}}\big] && \text{(B.54)}\\
&= \mathsf{E}\big[\mathsf{A}(\mathbf{X}-\mathsf{E}[\mathbf{X}])(\mathbf{X}-\mathsf{E}[\mathbf{X}])^{\mathsf{T}}\mathsf{A}^{\mathsf{T}}\big] && \text{(B.55)}\\
&= \mathsf{A}\,\mathsf{E}\big[(\mathbf{X}-\mathsf{E}[\mathbf{X}])(\mathbf{X}-\mathsf{E}[\mathbf{X}])^{\mathsf{T}}\mathsf{A}^{\mathsf{T}}\big] && \text{(B.56)}\\
&= \mathsf{A}\,\mathsf{E}\big[(\mathbf{X}-\mathsf{E}[\mathbf{X}])(\mathbf{X}-\mathsf{E}[\mathbf{X}])^{\mathsf{T}}\big]\mathsf{A}^{\mathsf{T}} && \text{(B.57)}\\
&= \mathsf{A}\,\mathsf{K}_{\mathbf{XX}}\,\mathsf{A}^{\mathsf{T}}. && \text{(B.58)}
\end{aligned}$$

□

One of the most important properties of covariance matrices is that they are all positive semidefinite.

Theorem B.12. Covariance matrices satisfy the following three properties.

1. All covariance matrices are positive semidefinite.
2. Any positive semidefinite matrix is the covariance matrix of some random vector.[5]
3. Covariance matrices are symmetric.

Proof: Part 3 follows both from the definition of covariance matrices and from Part 1, once Part 1 is proven.

To prove Part 1, let $\mathbf{X}$ be a random n-vector with covariance matrix $\mathsf{K}_{\mathbf{XX}}$ and define $Y \triangleq \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{X}$ for an arbitrary deterministic vector $\boldsymbol{\alpha}$. Then

$$0 \le \mathsf{Var}[Y] = \mathsf{E}\Big[(Y-\mathsf{E}[Y])^2\Big] = \mathsf{K}_{YY} = \boldsymbol{\alpha}^{\mathsf{T}}\,\mathsf{K}_{\mathbf{XX}}\,\boldsymbol{\alpha}, \tag{B.59}$$

where the last equality follows from (B.51). Since this holds for arbitrary $\boldsymbol{\alpha}$, it follows by definition that $\mathsf{K}_{\mathbf{XX}} \succeq 0$.

[5] Actually, it is always possible to choose this random vector to be jointly Gaussian. See Definition B.19 of Gaussian random vectors below.

To prove Part 2, let $\mathsf{K} \succeq 0$. Then we know that there exists a matrix S such that $\mathsf{K} = \mathsf{S}^{\mathsf{T}}\mathsf{S}$. Now let $\mathbf{W}$ be a random n-vector with independent components of variance 1, i.e., $\mathsf{K}_{\mathbf{WW}} = \mathsf{I}_n$, and define $\mathbf{X} \triangleq \mathsf{S}^{\mathsf{T}}\mathbf{W}$. By (B.51) we then have

$$\mathsf{K}_{\mathbf{XX}} = \mathsf{S}^{\mathsf{T}}\mathsf{I}_n\mathsf{S} = \mathsf{S}^{\mathsf{T}}\mathsf{S} = \mathsf{K}. \tag{B.60}$$

□

Proposition B.13. Let $\mathbf{Z}$ be a random n-vector of zero mean and of covariance matrix $\mathsf{K}_{\mathbf{ZZ}}$. Then one of the components of $\mathbf{Z}$ is a linear combination of the other components with probability 1 if, and only if, $\mathsf{K}_{\mathbf{ZZ}}$ is singular.

Proof: One of the components of $\mathbf{Z}$ is a linear combination of the other components with probability 1 if, and only if, there exists some nonzero vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ such that $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{Z}$ is zero with probability 1. This, on the other hand, is equivalent to $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{Z}$ having a zero variance. By Lemma B.11 this variance is $\boldsymbol{\alpha}^{\mathsf{T}}\mathsf{K}_{\mathbf{ZZ}}\boldsymbol{\alpha}$, which by Corollary B.5 is zero if, and only if, $\mathsf{K}_{\mathbf{ZZ}}\boldsymbol{\alpha} = \mathbf{0}$, i.e., $\mathsf{K}_{\mathbf{ZZ}}$ is singular. □

The above proposition can be extended to take account of the rank of $\mathsf{K}_{\mathbf{ZZ}}$.

Proposition B.14. Let $\mathbf{Z}$ be a zero-mean random vector of covariance matrix $\mathsf{K}_{\mathbf{ZZ}}$. The columns $\ell_1, \ldots, \ell_r$ of the $n \times n$ covariance matrix $\mathsf{K}_{\mathbf{ZZ}}$ are linearly independent and span all the columns of $\mathsf{K}_{\mathbf{ZZ}}$ if, and only if, the random r-vector $(Z_{\ell_1}, \ldots, Z_{\ell_r})^{\mathsf{T}}$ has a nonsingular covariance matrix and with probability 1 each component of $\mathbf{Z}$ can be written as a linear combination of $Z_{\ell_1}, \ldots, Z_{\ell_r}$.

Example B.15. For example, suppose that

$$\mathsf{K}_{\mathbf{ZZ}} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 1 \\ 3 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 5 & 7 \\ 5 & 9 & 13 \\ 7 & 13 & 19 \end{pmatrix}. \tag{B.61}$$

We note that the three columns of $\mathsf{K}_{\mathbf{ZZ}}$ satisfy the linear relationship:

$$(-1)\begin{pmatrix} 3 \\ 5 \\ 7 \end{pmatrix} + 2\begin{pmatrix} 5 \\ 9 \\ 13 \end{pmatrix} - 1\begin{pmatrix} 7 \\ 13 \\ 19 \end{pmatrix} = \mathbf{0}. \tag{B.62}$$

This linear relationship implies the linear relationship

$$-Z_1 + 2Z_2 - Z_3 = 0 \tag{B.63}$$

with probability 1. ◊

B.3 Characteristic Function

Definition B.16. Let $\mathbf{X}$ be a random n-vector taking value in $\mathbb{R}^n$. Then its *characteristic function* $\Phi_{\mathbf{X}}(\cdot)$ is a mapping from $\mathbb{R}^n$ to $\mathbb{C}$. It maps a vector $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)^{\mathsf{T}}$ to $\Phi_{\mathbf{X}}(\boldsymbol{\omega})$ where

$$\Phi_{\mathbf{X}}(\boldsymbol{\omega}) \triangleq \mathsf{E}\left[e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\mathbf{X}}\right] \tag{B.64}$$

$$= \mathsf{E}\left[e^{\mathrm{i}\sum_{\ell=1}^{n}\omega_\ell X_\ell}\right], \quad \boldsymbol{\omega} \in \mathbb{R}^n. \tag{B.65}$$

If $\mathbf{X}$ has the density $f_{\mathbf{X}}(\cdot)$, then

$$\Phi_{\mathbf{X}}(\boldsymbol{\omega}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x})\, e^{\mathrm{i}\sum_{\ell=1}^{n}\omega_\ell x_\ell}\, \mathrm{d}x_1 \cdots \mathrm{d}x_n. \tag{B.66}$$

Note that (B.66) is a variation of the multi-dimensional Fourier transform of $f_{\mathbf{X}}(\cdot)$. It is thus not surprising that two random n-vectors $\mathbf{X}$, $\mathbf{Y}$ are of the same distribution if, and only if, they have identical characteristic functions:

$$\mathbf{X} \overset{\mathscr{L}}{=} \mathbf{Y} \iff \Phi_{\mathbf{X}}(\boldsymbol{\omega}) = \Phi_{\mathbf{Y}}(\boldsymbol{\omega}),\ \forall\, \boldsymbol{\omega} \in \mathbb{R}^n. \tag{B.67}$$

From this one can show the following lemma.

Lemma B.17. Two random variables X and Y are independent if, and only if,

$$\mathsf{E}\left[e^{\mathrm{i}(\omega_1 X + \omega_2 Y)}\right] = \mathsf{E}\left[e^{\mathrm{i}\omega_1 X}\right] \cdot \mathsf{E}\left[e^{\mathrm{i}\omega_2 Y}\right], \quad \omega_1, \omega_2 \in \mathbb{R}. \tag{B.68}$$

Proof: One direction is straightforward: If X and Y are independent, then so are $e^{\mathrm{i}(\omega_1 X)}$ and $e^{\mathrm{i}(\omega_2 Y)}$, so that the expectation of their product is the product of their expectations:

$$\mathsf{E}\left[e^{\mathrm{i}(\omega_1 X + \omega_2 Y)}\right] = \mathsf{E}\left[e^{\mathrm{i}\omega_1 X}\, e^{\mathrm{i}\omega_2 Y}\right] \tag{B.69}$$

$$= \mathsf{E}\left[e^{\mathrm{i}\omega_1 X}\right] \mathsf{E}\left[e^{\mathrm{i}\omega_2 Y}\right], \quad \omega_1, \omega_2 \in \mathbb{R}. \tag{B.70}$$

As to the other direction, suppose that X' has the same law as X, that Y' has the same law as Y, and that X' and Y' are independent. Since X' has the same law as X, it follows that

$$\mathsf{E}\left[e^{\mathrm{i}\omega_1 X'}\right] = \mathsf{E}\left[e^{\mathrm{i}\omega_1 X}\right], \quad \omega_1 \in \mathbb{R}, \tag{B.71}$$

and similarly for Y'

$$\mathsf{E}\left[e^{\mathrm{i}\omega_1 Y'}\right] = \mathsf{E}\left[e^{\mathrm{i}\omega_1 Y}\right], \quad \omega_2 \in \mathbb{R}. \tag{B.72}$$

Consequently, since X' and Y' are independent,

$$\mathsf{E}\left[e^{\mathrm{i}(\omega_1 X' + \omega_2 Y')}\right] = \mathsf{E}\left[e^{\mathrm{i}\omega_1 X'}\right] \cdot \mathsf{E}\left[e^{\mathrm{i}\omega_2 Y'}\right] \tag{B.73}$$

$$= \mathsf{E}\left[e^{\mathrm{i}\omega_1 X}\right] \cdot \mathsf{E}\left[e^{\mathrm{i}\omega_2 Y}\right], \quad \omega_1, \omega_2 \in \mathbb{R}. \tag{B.74}$$

where the second equality follows from (B.71) and (B.72).

We thus see that if (B.68) holds, then the characteristic function of the vector $(X,Y)^{\mathsf{T}}$ is identical to the characteristic function of the vector $(X',Y')^{\mathsf{T}}$. Consequently the joint distribution of (X,Y) must be the same as the joint distribution of (X',Y'). But according to the latter distribution the two components (X' and Y') are independent, hence the same must be true for the joint distribution of the two components of the former. Thus, X and Y must be independent. □

B.4 Standard Gaussian Vector

We are now ready to introduce Gaussian random vectors. Similar to the Gaussian random variable situation, we start with the *standard Gaussian vector*.

Definition B.18. We shall say that a random n-vector $\mathbf{W}$ is an *n-dimensional standard Gaussian vector* if its n components are independent scalar standard Gaussians. In that case its density $f_{\mathbf{W}}(\cdot)$ is given by

$$f_{\mathbf{W}}(\mathbf{w}) = \prod_{\ell=1}^{n} \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}w_\ell^2} \right) \tag{B.75}$$

$$= \frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2}\sum_{\ell=1}^{n} w_\ell^2} \tag{B.76}$$

$$= (2\pi)^{-\frac{n}{2}} e^{-\frac{1}{2}\|\mathbf{w}\|^2}, \quad \mathbf{w} \in \mathbb{R}^n. \tag{B.77}$$

Here we use $\mathbf{w}$ to denote the argument of the density function $f_{\mathbf{W}}(\cdot)$ of $\mathbf{W}$, and we denote by $w_1, \ldots, w_n$ the n components of $\mathbf{w}$.

Note that the definition of standard Gaussian random variables is an extension of the standard scalar Gaussian distribution. The sole component of a standard one-dimensional Gaussian vector is a scalar of a $\mathcal{N}(0,1)$ distribution. Conversely, every scalar that has a $\mathcal{N}(0,1)$ distribution can be viewed as a one-dimensional random vector of the standard Gaussian distribution.

A standard Gaussian random n-vector $\mathbf{W}$ has the following mean and covariance matrix:

$$\mathsf{E}[\mathbf{W}] = \mathbf{0} \quad \text{and} \quad \mathsf{K}_{\mathbf{W}\mathbf{W}} = \mathsf{I}_n. \tag{B.78}$$

This follows because the components of $\mathbf{W}$ are all of zero mean; they are independent (and hence uncorrelated); and of unit variance.

The characteristic function of a standard Gaussian n-vector $\mathbf{W}$ can be easily computed from the characteristic function of a $\mathcal{N}(0,1)$ scalar random variable, because the components of a standard Gaussian vector are IID $\sim$

$\mathcal{N}(0,1)$:

$$\Phi_{\mathbf{W}}(\boldsymbol{\omega}) = \mathsf{E}\Big[e^{\mathrm{i}\sum_{\ell=1}^{n}\omega_\ell W_\ell}\Big] \tag{B.79}$$

$$= \mathsf{E}\left[\prod_{\ell=1}^{n} e^{\mathrm{i}\omega_\ell W_\ell}\right] \tag{B.80}$$

$$= \prod_{\ell=1}^{n} \mathsf{E}\Big[e^{\mathrm{i}\omega_\ell W_\ell}\Big] \tag{B.81}$$

$$= \prod_{\ell=1}^{n} e^{-\frac{1}{2}\omega_\ell^2} \tag{B.82}$$

$$= e^{-\frac{1}{2}\|\boldsymbol{\omega}\|^2}, \quad \boldsymbol{\omega} \in \mathbb{R}^n. \tag{B.83}$$

Here in (B.82) we have used the characteristic function of a standard Gaussian RV.

B.5 Gaussian Vectors

Definition B.19. A random n-vector $\mathbf{X}$ is said to be a *Gaussian vector* if for some positive integer m there exists a deterministic $n \times m$ matrix A; a standard Gaussian random m-vector $\mathbf{W}$; and a deterministic n-vector $\mathbf{b}$ such that

$$\mathbf{X} = \mathsf{A}\mathbf{W} + \mathbf{b}. \tag{B.84}$$

A Gaussian vector is said to be a *centered Gaussian vector* if $\mathbf{b} = \mathbf{0}$, i.e., if we can write

$$\mathbf{X} = \mathsf{A}\mathbf{W}. \tag{B.85}$$

Note that any scalar $\mathcal{N}(\mu, \sigma^2)$ random variable, when viewed as a one-dimensional random vector, is a Gaussian vector. Indeed, it can be written as $\sigma W + \mu$. Somewhat less obvious is the fact that the (sole) component of any one-dimensional Gaussian vector has a scalar Gaussian distribution. To see that note that if X is a one-dimensional Gaussian vector then $X = \mathbf{a}^{\mathsf{T}}\mathbf{W} + b$, where $\mathbf{a}$ is an m-vector, $\mathbf{W}$ is a standard Gaussian m-vector, and b is a deterministic scalar. Thus,

$$X = \sum_{\ell=1}^{m} a_\ell W_\ell + b, \tag{B.86}$$

and the result follows because we have seen that the sum of independent Gaussian scalars is a Gaussian scalar (Proposition A.12).

The following are immediate consequences of the definition:

- Any standard Gaussian n-vector is a Gaussian vector.

  Choose in the above definition $m = n$; $\mathsf{A} = \mathsf{I}_n$; and $\mathbf{b} = \mathbf{0}$.

- Any deterministic vector is Gaussian.

  Choose the matrix A as the all-zero matrix.

- If the components of $\mathbf{X}$ are independent scalar Gaussians (not necessarily of equal variance), then $\mathbf{X}$ is a Gaussian vector.

  Choose A to be an appropriate diagonal matrix.

- If

$$\mathbf{X}_1 = (X_{1,1}, \dots, X_{1,n_1})^{\mathsf{T}} \tag{B.87}$$

  is a Gaussian n_1-vector and

$$\mathbf{X}_2 = (X_{2,1}, \dots, X_{2,n_2})^{\mathsf{T}} \tag{B.88}$$

  is a Gaussian n_2-vector, which is independent of $\mathbf{X}_1$, then the random (n_1+n_2)-vector

$$(X_{1,1}, \dots, X_{1,n_1}, X_{2,1}, \dots, X_{2,n_2})^{\mathsf{T}} \tag{B.89}$$

  is Gaussian.

  Suppose that the pair $(\mathsf{A}_1, \mathbf{b}_1)$ represent $\mathbf{X}_1$ where A_1 is of size $n_1 \times m_1$ and $\mathbf{b}_1 \in \mathbb{R}^{n_1}$. Similarly let the pair $(\mathsf{A}_2, \mathbf{b}_2)$ represent $\mathbf{X}_2$ where A_2 is of size $n_2 \times m_2$ and $\mathbf{b}_2 \in \mathbb{R}^{n_2}$. Then the vector (B.89) can be represented using the $(n_1+n_2) \times (m_1+m_2)$ block-diagonal matrix A of diagonal components A_1 and A_2, and using the vector $\mathbf{b} \in \mathbb{R}^{n_1+n_2}$ that results when the vector $\mathbf{b}_1$ is stacked on top of the vector $\mathbf{b}_2$.

- Assume that $\mathbf{X}$ is a Gaussian n-vector. If C is an arbitrary deterministic $\nu \times n$ matrix and $\mathbf{d}$ is a deterministic ν-vector, then the random ν-vector $\mathsf{C}\mathbf{X} + \mathbf{d}$ is a Gaussian ν-vector.

  Indeed, if $\mathbf{X} = \mathsf{A}\mathbf{W} + \mathbf{b}$ where A is a deterministic $n \times m$ matrix, $\mathbf{W}$ is a standard Gaussian m-vector, and $\mathbf{b}$ is a deterministic m-vector, then

$$\begin{aligned} \mathsf{C}\mathbf{X} + \mathbf{d} &= \mathsf{C}(\mathsf{A}\mathbf{W} + \mathbf{b}) + \mathbf{d} && \text{(B.90)} \\ &= (\mathsf{C}\mathsf{A})\mathbf{W} + (\mathsf{C}\mathbf{b} + \mathbf{d}), && \text{(B.91)} \end{aligned}$$

  which demonstrates that $\mathsf{C}\mathbf{X} + \mathbf{d}$ is Gaussian, because $(\mathsf{C} \cdot \mathsf{A})$ is a deterministic $\nu \times m$ matrix, $\mathbf{W}$ is a standard Gaussian m-vector; and $\mathsf{C}\mathbf{b} + \mathbf{d}$ is a deterministic ν-vector.

- If the Gaussian vector $\mathbf{X}$ has the representation (B.84), then

$$\mathsf{E}[\mathbf{X}] = \mathbf{b} \quad \text{and} \quad \mathsf{K}_{\mathbf{X}\mathbf{X}} = \mathsf{A}\mathsf{A}^{\mathsf{T}}. \tag{B.92}$$

The behavior of the mean follows from (B.47) by viewing $\mathbf{W}$ as a random matrix and noting that its expectation is zero. The covariance follows from Lemma B.11 and (B.78).

By (B.92) we conclude that an alternative definition for a Gaussian random vector is to say that $\mathbf{X}$ is a Gaussian random vector if $\mathbf{X} - \mathsf{E}[\mathbf{X}]$ is a zero-mean or *centered* Gaussian vector. We shall thus focus our attention now on centered Gaussian vectors. The results we obtain will be easily extendable to noncentered Gaussian vectors.

Specializing (B.91) to the zero-mean case we obtain that if $\mathbf{X}$ is a centered Gaussian vector, then $\mathsf{C}\mathbf{X}$ is also a centered Gaussian vector. This has the following consequences:

- If $\mathbf{X}$ is a centered Gaussian n-vector, then any subset of its components is also a centered Gaussian vector.

  The vector $(X_{j_1}, \ldots, X_{j_p})^{\mathsf{T}}$ where $1 \le j_1, \ldots, j_p \le n$ results from applying $\mathsf{C}\mathbf{X}$ where C is a $p \times n$ matrix of entries

  $$(\mathsf{C})_{\nu,\ell} = \mathbb{1}\{\ell = j_\nu\}. \tag{B.93}$$

  For example,

  $$\begin{pmatrix} X_3 \\ X_1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}. \tag{B.94}$$

- Similarly, if $\mathbf{X}$ is centered Gaussian, then any permutation of its elements is centered Gaussian.

  The proof is identical. For example,

  $$\begin{pmatrix} X_3 \\ X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}. \tag{B.95}$$

By (B.92) we see that in the representation (B.84) the vector $\mathbf{b}$ is fully determined by the mean of the random vector. In fact, the two are the same. The matrix A, however, is not determined by the covariance matrix $\mathsf{K}_{\mathbf{XX}}$ of the random vector. Indeed, $\mathsf{K}_{\mathbf{XX}}$ may be written in many different ways as the product of a matrix by its transpose. One would therefore be mistakenly tempted to think that there are many different Gaussian distributions of a given mean and covariance matrix, but this is not the case. We shall see that there is only one.

B.6 Mean and Covariance Determine the Law of a Gaussian

One way to see that the mean and covariance matrix of a Gaussian vector fully determine its law is via the characteristic function: If $\mathbf{X} = \mathsf{A}\mathbf{W}$ where A is an $n \times m$ deterministic matrix and the m-vector $\mathbf{W}$ is a standard Gaussian vector, then the characteristic function $\Phi_{\mathbf{X}}(\cdot)$ of $\mathbf{X}$ can be calculated as follows:

$$\begin{aligned}
\Phi_{\mathbf{X}}(\boldsymbol{\omega}) &= \mathsf{E}\left[e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\mathbf{X}}\right] && \text{(B.96)}\\
&= \mathsf{E}\left[e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\mathsf{A}\mathbf{W}}\right] && \text{(B.97)}\\
&= \mathsf{E}\left[e^{\mathrm{i}(\mathsf{A}^{\mathsf{T}}\boldsymbol{\omega})^{\mathsf{T}}\mathbf{W}}\right] && \text{(B.98)}\\
&= e^{-\frac{1}{2}\|\mathsf{A}^{\mathsf{T}}\boldsymbol{\omega}\|^2} && \text{(B.99)}\\
&= e^{-\frac{1}{2}(\mathsf{A}^{\mathsf{T}}\boldsymbol{\omega})^{\mathsf{T}}\mathsf{A}^{\mathsf{T}}\boldsymbol{\omega}} && \text{(B.100)}\\
&= e^{-\frac{1}{2}\boldsymbol{\omega}^{\mathsf{T}}\mathsf{A}\mathsf{A}^{\mathsf{T}}\boldsymbol{\omega}} && \text{(B.101)}\\
&= e^{-\frac{1}{2}\boldsymbol{\omega}^{\mathsf{T}}\mathsf{K}_{\mathbf{X}\mathbf{X}}\boldsymbol{\omega}}, \quad \boldsymbol{\omega} \in \mathbb{R}^n. && \text{(B.102)}
\end{aligned}$$

The characteristic function of a centered Gaussian vector is thus fully determined by its covariance matrix. Since the characteristic function of a random variable fully specifies its law (see (B.67)), we conclude that any two centered Gaussians of equal covariance matrices must have identical laws, thus proving the following theorem.

Theorem B.20. There is only one multivariate centered Gaussian distribution of a given covariance matrix. Accounting for the mean we have that there is only one multivariate Gaussian distribution of a given mean vector and a given covariance matrix.

We denote the multivariate Gaussian distribution of mean $\boldsymbol{\mu}$ and covariance K by $\mathcal{N}(\boldsymbol{\mu}, \mathsf{K})$.

This theorem has extremely important consequences. One of which has to do with the properties of independence and uncorrelatedness. Recall that any two independent random variables are also uncorrelated. The reverse is, in general, not true. It is, however, true for jointly Gaussian random variables. If X and Y are jointly Gaussian, i.e., $(X, Y)^{\mathsf{T}}$ is a Gaussian 2-vector, then X and Y are independent if, and only if, they are uncorrelated. More generally we have the following corollary.

Corollary B.21. Let $\mathbf{X}$ be a centered Gaussian $(n_1 + n_2)$-vector. Let $\mathbf{X}_1 = (X_1, \ldots, X_{n_1})^{\mathsf{T}}$ correspond to its first n_1 components, and let $\mathbf{X}_2 = (X_{n_1+1}, \ldots, X_{n_1+n_2})^{\mathsf{T}}$ correspond to the rest of its components.[6] Then the vectors $\mathbf{X}_1$

[6] Note that we are slightly overstretching our notation here: X_1 denotes the first component of $\mathbf{X}$, while $\mathbf{X}_1$ denotes a vector (whose first component by definition also happens to be X_1).

and $\mathbf{X}_2$ are independent if, and only if, they are uncorrelated, i.e., if, and only if,

$$\mathsf{E}\big[\mathbf{X}_1\mathbf{X}_2^{\mathsf{T}}\big] = 0. \tag{B.103}$$

Proof: The easy direction (and the direction that has nothing to do with Gaussianity) is to show that if $\mathbf{X}_1$ and $\mathbf{X}_2$ are independent, then (B.103) holds. Indeed, by the independence and the fact that the vectors are of zero mean we have

$$\begin{aligned}
\mathsf{E}\big[\mathbf{X}_1\mathbf{X}_2^{\mathsf{T}}\big] &= \mathsf{E}[\mathbf{X}_1]\,\mathsf{E}\big[\mathbf{X}_2^{\mathsf{T}}\big] && \text{(B.104)}\\
&= \mathsf{E}[\mathbf{X}_1](\mathsf{E}[\mathbf{X}_2])^{\mathsf{T}} && \text{(B.105)}\\
&= \mathbf{0}\mathbf{0}^{\mathsf{T}} && \text{(B.106)}\\
&= 0. && \text{(B.107)}
\end{aligned}$$

We now prove the reverse using the Gaussianity. Let $\mathbf{X}$ be the (n_1+n_2)-vector that results from stacking $\mathbf{X}_1$ on top of $\mathbf{X}_2$. Then

$$\begin{aligned}
\mathsf{K}_{\mathbf{X}\mathbf{X}} &= \begin{pmatrix} \mathsf{E}\big[\mathbf{X}_1\mathbf{X}_1^{\mathsf{T}}\big] & \mathsf{E}\big[\mathbf{X}_1\mathbf{X}_2^{\mathsf{T}}\big] \\ \mathsf{E}\big[\mathbf{X}_1\mathbf{X}_2^{\mathsf{T}}\big] & \mathsf{E}\big[\mathbf{X}_2\mathbf{X}_2^{\mathsf{T}}\big] \end{pmatrix} && \text{(B.108)}\\
&= \begin{pmatrix} \mathsf{K}_{\mathbf{X}_1\mathbf{X}_1} & 0 \\ 0 & \mathsf{K}_{\mathbf{X}_2\mathbf{X}_2} \end{pmatrix}, && \text{(B.109)}
\end{aligned}$$

where we have used (B.103).

Let $\mathbf{X}_1'$ and $\mathbf{X}_2'$ be independent vectors such that the law of $\mathbf{X}_1'$ is identical to the law of $\mathbf{X}_1$ and such that the law of $\mathbf{X}_2'$ is identical to the law of $\mathbf{X}_2$. Let $\mathbf{X}'$ be the (n_1+n_2)-vector that results from stacking $\mathbf{X}_1'$ on top of $\mathbf{X}_2'$. Note that $\mathbf{X}'$ is a centered Gaussian vector. This can be seen because, being the components of a Gaussian vector, both $\mathbf{X}_1'$ and $\mathbf{X}_2'$ must be Gaussian, and since they are independent, the vector that is formed by stacking them on top of each other must also be Gaussian. Since $\mathbf{X}_1'$ and $\mathbf{X}_2'$ are zero-mean and independent, and since the law (and hence covariance matrix) of $\mathbf{X}_1'$ is the same as of $\mathbf{X}_1$ and similarly for $\mathbf{X}_2'$ we have that the covariance matrix of the vector $\mathbf{X}'$ is given by

$$\begin{aligned}
\mathsf{K}_{\mathbf{X}'\mathbf{X}'} &= \begin{pmatrix} \mathsf{K}_{\mathbf{X}_1'\mathbf{X}_1'} & 0 \\ 0 & \mathsf{K}_{\mathbf{X}_2'\mathbf{X}_2'} \end{pmatrix} && \text{(B.110)}\\
&= \begin{pmatrix} \mathsf{K}_{\mathbf{X}_1\mathbf{X}_1} & 0 \\ 0 & \mathsf{K}_{\mathbf{X}_2\mathbf{X}_2} \end{pmatrix}. && \text{(B.111)}
\end{aligned}$$

But this is identical to the covariance matrix (B.109) of the zero-mean Gaussian $\mathbf{X}$. Consequently, since zero-mean Gaussian vector of the same covariance matrix must have the same law, it follows that the law of $\mathbf{X}$ is identical to the

law of $\mathbf{X}'$. But under the law of $\mathbf{X}'$ the first n_1 components are independent of the last n_2 components, so that the same must be true of $\mathbf{X}$. □

A variation on the same theme:

Corollary B.22. If the components of the Gaussian vector $\mathbf{X}$ are uncorrelated so that the matrix $\mathsf{K}_{\mathbf{XX}}$ is diagonal, then the components of $\mathbf{X}$ are independent.

Proof: This follows by a repeated application of the previous corollary. □

Another consequence of the fact that there is only one multivariate Gaussian distribution with a given mean and covariance is the following corollary.

Corollary B.23. If the components of a Gaussian vector are independent in pairs, then they are independent. (Recall that in general X, Y, Z can be such that X, Y are independent; Y, Z are independent; X, Z are independent; and yet X, Y, Z are not independent. This cannot happen if X, Y, Z are the components of a Gaussian vector.)

Proof: To see this consider the covariance matrix. If the components are independent in pairs the covariance matrix must be diagonal. But this is also the structure of the covariance matrix of a Gaussian random vector whose components are independent. Consequently, since the covariance matrix of a Gaussian vector fully determines the vector's law, it follows that the two vectors have the same law, and the result follows. □

The following corollary is extremely useful in the study of signal detection.

Corollary B.24. Suppose that $\mathbf{Z}$ is a Gaussian vector of covariance $\sigma^2\mathsf{I}_n$, where I_n is the $n \times n$ identity matrix. This is just another way of saying that $\mathbf{Z}$ has n components that are IID, each distributed $\sim \mathcal{N}(0, \sigma^2)$.

Let $\{\boldsymbol{\phi}_\ell\}_{\ell=1}^n$ be n orthonormal deterministic vectors so that

$$\langle \boldsymbol{\phi}_\ell, \boldsymbol{\phi}_{\ell'} \rangle \triangleq \boldsymbol{\phi}_\ell^{\mathsf{T}} \boldsymbol{\phi}_{\ell'} = \begin{cases} 1 & \text{if } \ell = \ell', \\ 0 & \text{if } \ell \neq \ell'. \end{cases} \tag{B.112}$$

Then the random variables $\langle \mathbf{Z}, \boldsymbol{\phi}_1 \rangle, \dots, \langle \mathbf{Z}, \boldsymbol{\phi}_n \rangle$ are IID $\sim \mathcal{N}(0, \sigma^2)$.

Proof: Let $\tilde{\mathbf{Z}}$ denote the vector

$$\tilde{\mathbf{Z}} = \begin{pmatrix} \langle \mathbf{Z}, \boldsymbol{\phi}_1 \rangle \\ \vdots \\ \langle \mathbf{Z}, \boldsymbol{\phi}_n \rangle \end{pmatrix}. \tag{B.113}$$

Note that $\tilde{\mathbf{Z}}$ can be represented as

$$\tilde{\mathbf{Z}} = \begin{pmatrix} \leftarrow & \boldsymbol{\phi}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{\phi}_n^{\mathsf{T}} & \rightarrow \end{pmatrix} \mathbf{Z}, \tag{B.114}$$

thus demonstrating that $\tilde{\mathbf{Z}}$ is a Gaussian vector. Its covariance can be computed according to Lemma B.11 to yield

$$\mathsf{K}_{\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}} = \begin{pmatrix} \leftarrow & \boldsymbol{\phi}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{\phi}_n^{\mathsf{T}} & \rightarrow \end{pmatrix} \mathsf{E}\big[\mathbf{Z}\mathbf{Z}^{\mathsf{T}}\big] \begin{pmatrix} \uparrow & \cdots & \uparrow \\ \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_n \\ \downarrow & \cdots & \downarrow \end{pmatrix} \tag{B.115}$$

$$= \begin{pmatrix} \leftarrow & \boldsymbol{\phi}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{\phi}_n^{\mathsf{T}} & \rightarrow \end{pmatrix} \sigma^2 \mathsf{I}_n \begin{pmatrix} \uparrow & \cdots & \uparrow \\ \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_n \\ \downarrow & \cdots & \downarrow \end{pmatrix} \tag{B.116}$$

$$= \sigma^2 \begin{pmatrix} \leftarrow & \boldsymbol{\phi}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{\phi}_n^{\mathsf{T}} & \rightarrow \end{pmatrix} \begin{pmatrix} \uparrow & \cdots & \uparrow \\ \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_n \\ \downarrow & \cdots & \downarrow \end{pmatrix} \tag{B.117}$$

$$= \sigma^2 \mathsf{I}_n. \tag{B.118}$$

Here the last equality follows by (B.112). □

This same corollary can be also stated a little differently as follows.

Corollary B.25. If $\mathbf{W}$ is a standard Gaussian n-vector and U is orthogonal, i.e.,

$$\mathsf{U}\mathsf{U}^{\mathsf{T}} = \mathsf{U}^{\mathsf{T}}\mathsf{U} = \mathsf{I}_n, \tag{B.119}$$

then $\mathsf{U}\mathbf{W}$ is also a standard Gaussian vector.

Proof: By the definition of a Gaussian vector we conclude that $\mathsf{U}\mathbf{W}$ is Gaussian. Its covariance matrix is

$$\mathsf{U}\,\mathsf{K}_{\mathbf{W}\mathbf{W}}\,\mathsf{U}^{\mathsf{T}} = \mathsf{U}\mathsf{I}_n\mathsf{U}^{\mathsf{T}} = \mathsf{U}\mathsf{U}^{\mathsf{T}} = \mathsf{I}_n. \tag{B.120}$$

Thus, $\mathsf{U}\mathbf{W}$ has the same mean (zero) and the same covariance matrix (I_n) as a standard Gaussian. Since the mean and covariance of a Gaussian fully specify its law, $\mathsf{U}\mathbf{W}$ must be standard. □

The next lemma shows that in the definition of Gaussian vectors $\mathbf{x} = \mathsf{A}\mathbf{W} + \mathbf{b}$ we can restrict ourselves to *square* $n \times n$ matrices A.

Lemma B.26. If $\mathbf{X}$ is an n-dimensional centered Gaussian vector, then there exists a *square* deterministic $n \times n$ matrix A such that the law of $\mathbf{X}$ is the same as the law of $A\mathbf{W}$ where $\mathbf{W}$ is an n-dimensional standard Gaussian.

Proof: Let $K_{\mathbf{XX}}$ denote the $n \times n$ covariance matrix of $\mathbf{X}$. Being a covariance matrix, it must be positive semidefinite, and there therefore exists some $n \times n$ matrix S such that $K_{\mathbf{XX}} = S^T S$. Consider now the random vector $S^T\mathbf{W}$ where $\mathbf{W}$ is standard n-dimensional Gaussian. Being of the form of a matrix multiplying a Gaussian vector, $S^T\mathbf{W}$ is a Gaussian vector, and since $\mathbf{W}$ is centered, so is $S^T\mathbf{W}$. By Lemma B.11 and by the fact that the covariance matrix of $\mathbf{W}$ is the identity, it follows that the covariance matrix of $S^T\mathbf{W}$ is $S^T S$, i.e., $K_{\mathbf{XX}}$. Thus $\mathbf{X}$ and $S^T\mathbf{W}$ are centered Gaussians of the same covariance, so that they must be of the same law. The law of $\mathbf{X}$ is thus the same as the law of the product of a square matrix by a standard Gaussian. □

This idea will be extended further in the following section.

B.7 Canonical Representation of Centered Gaussian Vectors

As we have seen, the representation of a centered Gaussian vector as the result of the multiplication of a deterministic matrix by a standard Gaussian vector is not unique. Indeed, whenever the $n \times m$ deterministic matrix A satisfies $AA^T = K$ it follows that $A\mathbf{W}$ has a $\mathcal{N}(\mathbf{0}, K)$ distribution. (Here $\mathbf{W}$ is a standard m-vector). This follows because $A\mathbf{W}$ is a random vector of covariance matrix AA^T; it is — by the definition of the multivariate centered Gaussian distribution — a centered Gaussian; and all centered Gaussians of a given covariance matrix have the same law. In this section we shall focus on a particular choice of the matrix A that is particularly useful in the analysis of Gaussians. In this representation A is a square matrix and can be written as the product of an orthogonal matrix by a diagonal matrix. The diagonal matrix acts on $\mathbf{W}$ by stretching and shrinking its components, and the orthogonal matrix then rotates the result.

Theorem B.27. Let $\mathbf{X}$ be a centered Gaussian n-dimensional vector of covariance matrix $K_{\mathbf{XX}}$. Then $\mathbf{X}$ has the same law as the centered Gaussian

$$U\Lambda^{1/2}\mathbf{W} \tag{B.121}$$

where $\mathbf{W}$ is a standard n-dimensional Gaussian vector, the columns of the orthogonal matrix U are orthonormal eigenvectors of the covariance matrix $K_{\mathbf{XX}}$, and the matrix Λ is a diagonal matrix whose diagonal entries correspond to the eigenvalues of $K_{\mathbf{XX}}$ with the νth diagonal element corresponding to the eigenvector in the νth column of U.

Remark B.28. Because Λ is diagonal we can explicitly write $\Lambda^{1/2}\mathbf{W}$ as

$$\Lambda^{1/2}\mathbf{W} = \begin{pmatrix} \sqrt{\lambda_1} W_1 \\ \vdots \\ \sqrt{\lambda_n} W_n \end{pmatrix}. \tag{B.122}$$

Thus, in the above theorem the random vector $\Lambda^{1/2}\mathbf{W}$ has independent components with the νth component being $\mathcal{N}(0, \lambda_\nu)$ distributed. This demonstrates that a centered Gaussian vector is no other than the rotation (multiplication by an orthogonal matrix) of a random vector whose components are independent centered univariate Gaussians of different variances. △

Proof of Theorem B.27: The matrix $\mathsf{K}_{\mathbf{XX}}$ is positive semidefinite so that it can be written as $\mathsf{K}_{\mathbf{XX}} = \mathsf{S}^\mathsf{T}\mathsf{S}$ where $\mathsf{S} = \Lambda^{1/2}\mathsf{U}^\mathsf{T}$ where $\mathsf{K}_{\mathbf{XX}}\,\mathsf{U} = \mathsf{U}\Lambda$; U is orthogonal; and Λ is a diagonal matrix whose diagonal elements are the (nonnegative) eigenvalues of $\mathsf{K}_{\mathbf{XX}}$. If $\mathbf{W}$ is a standard Gaussian then $\mathsf{S}^\mathsf{T}\mathbf{W}$ is of zero mean and covariance $\mathsf{S}^\mathsf{T}\mathsf{S}$, i.e., the same covariance as $\mathbf{X}$. Since there is only one centered multivariate Gaussian distribution of a given covariance matrix, it follows that the law of $\mathsf{S}^\mathsf{T}\mathbf{W}$ is the same as the law of $\mathbf{X}$. □

Example B.29. Figures B.1 and B.2 demonstrate this canonical representation. There the contour and mesh plots of four two-dimensional Gaussian vectors are shown:

$$\mathbf{X}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\mathbf{W}, \qquad \mathsf{K}_{\mathbf{X}_1\mathbf{X}_1} = \mathsf{I}_2, \tag{B.123}$$

$$\mathbf{X}_2 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}\mathbf{W}, \qquad \mathsf{K}_{\mathbf{X}_2\mathbf{X}_2} = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}, \tag{B.124}$$

$$\mathbf{X}_3 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}\mathbf{W}, \qquad \mathsf{K}_{\mathbf{X}_3\mathbf{X}_3} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}, \tag{B.125}$$

$$\mathbf{X}_4 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}\mathbf{W}, \qquad \mathsf{K}_{\mathbf{X}_4\mathbf{X}_4} = \frac{1}{2}\begin{pmatrix} 5 & 3 \\ 3 & 5 \end{pmatrix}, \tag{B.126}$$

where $\mathbf{W}$ is a standard Gaussian 2-vector

$$\mathbf{W} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right). \tag{B.127}$$

◊

The above proposition can also be used in order to demonstrate the linear transformation one can perform on a Gaussian to obtain a standard Gaussian. Thus, it is the multivariate version of the univariate result showing that if $X \sim \mathcal{N}(\mu, \sigma^2)$, where $\sigma^2 > 0$ then $(X - \mu)/\sigma$ has a $\mathcal{N}(0, 1)$ distribution.

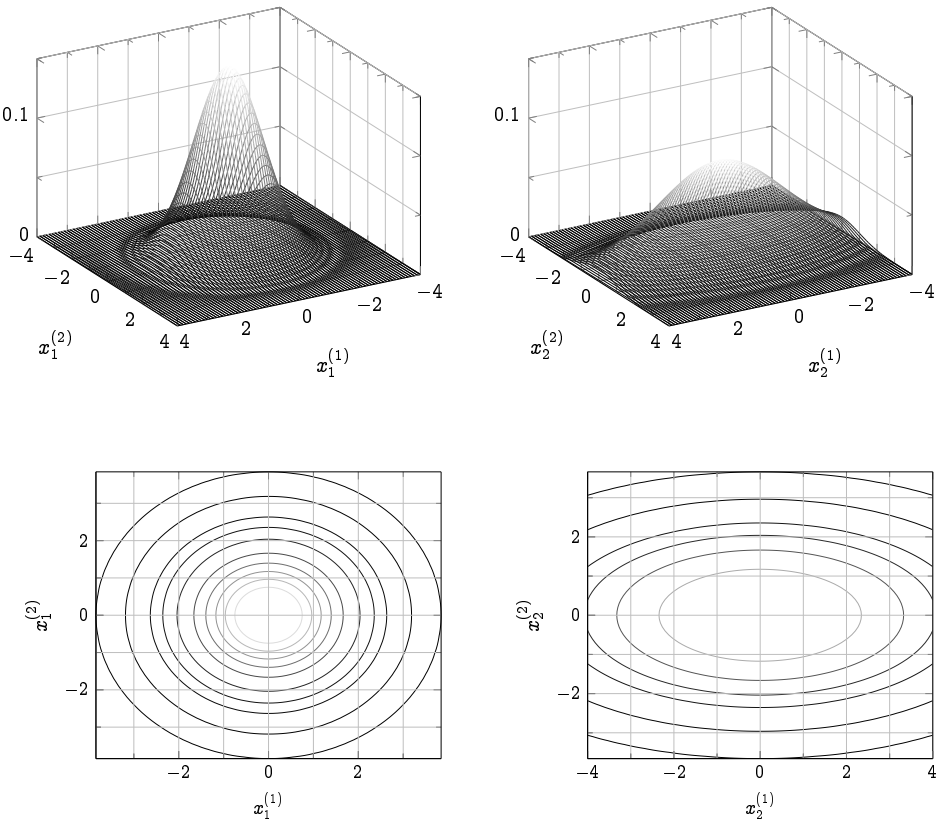


Figure B.1: Mesh and contour plot of the densities of the joint two-dimensional Gaussian vectors $\mathbf{X}_1$ and $\mathbf{X}_2$ given in Example B.29.

Proposition B.30. Let $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathsf{K})$ and let U and Λ be as above. Assume that K is positive definite so that its eigenvalues are strictly positive. Then

$$\Lambda^{-1/2}\mathsf{U}^{\mathsf{T}}(\mathbf{X} - \boldsymbol{\mu}) \sim \mathcal{N}(\mathbf{0}, \mathsf{I}) \tag{B.128}$$

where $\Lambda^{-1/2}$ is the matrix inverse of $\Lambda^{1/2}$, i.e., it is a diagonal matrix whose diagonal entries are the reciprocals of the square-roots of the eigenvalues of K.

Proof: Since the product of a deterministic matrix by a Gaussian vector results in a Gaussian vector and since the mean and covariance of a Gaussian vector fully specify its law, it suffices to check that the left-hand side of (B.128) is of the specified mean (namely, zero) and of the specified covariance matrix (namely, the identity matrix). □

B.8 Characteristic Function of a Gaussian Vector

Let $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathsf{K})$ where $\mathsf{K} = \mathsf{S}^{\mathsf{T}}\mathsf{S}$. Then $\mathbf{X}$ can be written as

$$\mathbf{X} = \mathsf{S}^{\mathsf{T}}\mathbf{W} + \boldsymbol{\mu}. \tag{B.129}$$

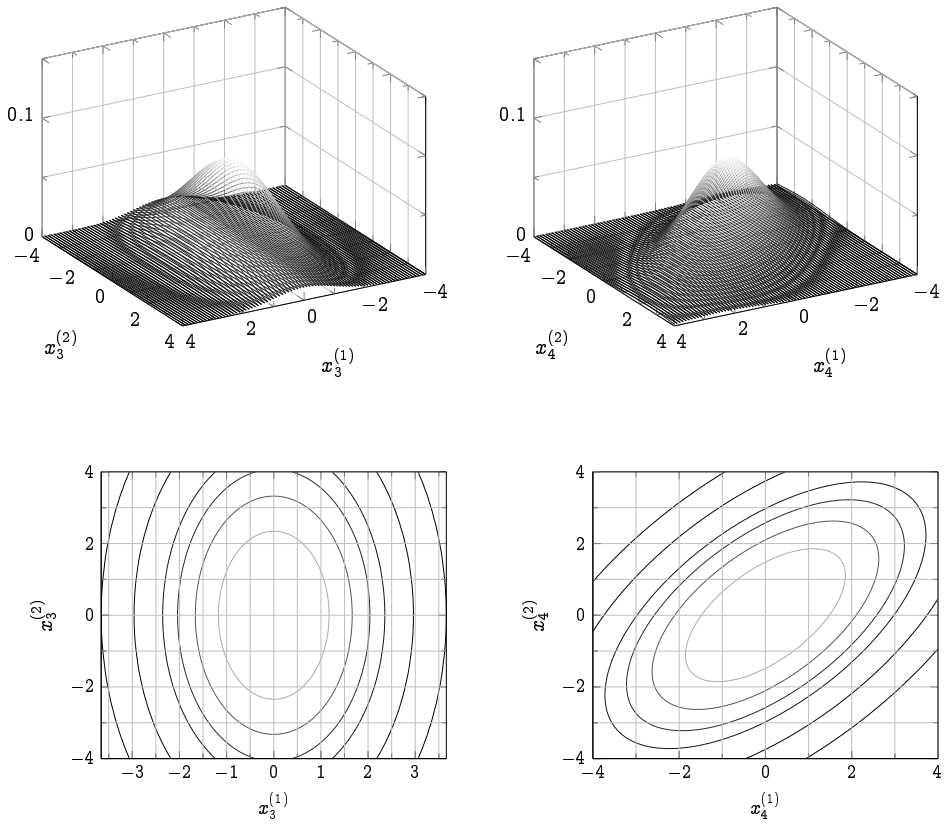


Figure B.2: Mesh and contour plot of the densities of the joint two-dimensional Gaussian vectors $\mathbf{X}_3$ and $\mathbf{X}_4$ given in Example B.29.

Using the characteristic function of a standard Gaussian vector (B.83), we can now easily derive the characteristic function of $\mathbf{X}$:

$$\begin{aligned}
\Phi_{\mathbf{X}}(\boldsymbol{\omega}) &= \mathsf{E}\left[e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\mathbf{X}}\right] && \text{(B.130)}\\
&= \mathsf{E}\left[e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}(\mathsf{S}^{\mathsf{T}}\mathbf{W}+\boldsymbol{\mu})}\right] && \text{(B.131)}\\
&= \mathsf{E}\left[e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\mathsf{S}^{\mathsf{T}}\mathbf{W}+\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}}\right] && \text{(B.132)}\\
&= \mathsf{E}\left[e^{\mathrm{i}(\mathsf{S}\boldsymbol{\omega})^{\mathsf{T}}\mathbf{W}}\right] e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}} && \text{(B.133)}\\
&= \Phi_{\mathbf{W}}(\mathsf{S}\boldsymbol{\omega}) \cdot e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}} && \text{(B.134)}\\
&= e^{-\frac{1}{2}\|\mathsf{S}\boldsymbol{\omega}\|^2} \cdot e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}} && \text{(B.135)}\\
&= e^{-\frac{1}{2}(\mathsf{S}\boldsymbol{\omega})^{\mathsf{T}}(\mathsf{S}\boldsymbol{\omega})+\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}} && \text{(B.136)}\\
&= e^{-\frac{1}{2}\boldsymbol{\omega}^{\mathsf{T}}\mathsf{S}^{\mathsf{T}}\mathsf{S}\boldsymbol{\omega}+\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}} && \text{(B.137)}\\
&= e^{-\frac{1}{2}\boldsymbol{\omega}^{\mathsf{T}}\mathsf{K}\boldsymbol{\omega}+\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}}. && \text{(B.138)}
\end{aligned}$$

B.9 Density of a Gaussian Vector

As we have seen, if the covariance matrix of a centered random vector is singular, then with probability 1 at least one of the components of the vector can be expressed as a linear combination of the other components. Consequently, random vectors with singular covariance matrices cannot have a density.

We shall thus only discuss the density of the multivariate Gaussian distribution for the case where the covariance matrix K is nonsingular, i.e., is positive definite. In this case all its eigenvalues are strictly positive. To derive this density we shall use Theorem B.27 to represent the $\mathcal{N}(\mathbf{0}, \mathsf{K})$ distribution as the distribution of the result of

$$\mathsf{U}\Lambda^{1/2}\mathbf{W} \tag{B.139}$$

where U is an orthogonal matrix and Λ is a diagonal matrix satisfying $\mathsf{K}\mathsf{U} = \mathsf{U}\Lambda$. Note that Λ is nonsingular because its diagonal elements are the eigenvalues of the positive definite matrix K.

The following example motivates the discussion about the rank of the matrix A in the representation (B.84).

Example B.31. Suppose that the random 3-vector

$$\mathbf{X} = (X_1, X_2, X_3)^{\mathsf{T}} \tag{B.140}$$

can be represented in terms of the standard Gaussian 4-vector $\mathbf{W}$ as

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 & 3 \\ -1 & 5 & 3 & 1 \\ 1 & 7 & 7 & 7 \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{pmatrix}. \tag{B.141}$$

Hence, $\mathbf{X}$ is centered Gaussian. Since

$$\begin{pmatrix} 1 & 7 & 7 & 7 \end{pmatrix} = 2\begin{pmatrix} 1 & 1 & 2 & 3 \end{pmatrix} + \begin{pmatrix} -1 & 5 & 3 & 1 \end{pmatrix} \tag{B.142}$$

it follows that

$$X_3 = \begin{pmatrix} 1 & 7 & 7 & 7 \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{pmatrix} \tag{B.143}$$

$$= \left(2\begin{pmatrix} 1 & 1 & 2 & 3 \end{pmatrix} + \begin{pmatrix} -1 & 5 & 3 & 1 \end{pmatrix}\right) \begin{pmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{pmatrix} \tag{B.144}$$

$$= 2\begin{pmatrix}1 & 1 & 2 & 3\end{pmatrix}\begin{pmatrix}W_1\\W_2\\W_3\\W_4\end{pmatrix} + \begin{pmatrix}-1 & 5 & 3 & 1\end{pmatrix}\begin{pmatrix}W_1\\W_2\\W_3\\W_4\end{pmatrix} \tag{B.145}$$

$$= 2X_1 + X_2, \tag{B.146}$$

irrespective of the realization of $\mathbf{W}$! Consequently, the random vector $\mathbf{X}$ can also be written as:

$$\begin{pmatrix}X_1\\X_2\end{pmatrix} = \begin{pmatrix}1 & 1 & 2 & 3\\-1 & 5 & 3 & 1\end{pmatrix}\begin{pmatrix}W_1\\W_2\\W_3\\W_4\end{pmatrix}; \quad \text{and} \tag{B.147}$$

$$X_3 = 2X_1 + X_2. \tag{B.148}$$

The reader is encouraged to verify that $\mathbf{X}$ can also be written as:

$$\begin{pmatrix}X_2\\X_3\end{pmatrix} = \begin{pmatrix}-1 & 5 & 3 & 1\\1 & 7 & 7 & 7\end{pmatrix}\begin{pmatrix}W_1\\W_2\\W_3\\W_4\end{pmatrix}; \quad \text{and} \tag{B.149}$$

$$X_1 = -\frac{1}{2}X_2 + \frac{1}{2}X_3. \tag{B.150}$$

This example is closely related to Example B.15: Also here we have a singular covariance matrix

$$\mathsf{K}_{\mathbf{XX}} = \begin{pmatrix}15 & 13 & 43\\13 & 36 & 62\\43 & 62 & 148\end{pmatrix}, \tag{B.151}$$

and therefore at least (in this case actually exactly) one component of $\mathbf{X}$ can be written as a linear combination of the others with probability 1. However, note the subtle difference here: Since the Gaussian vector is fully specified by mean and covariance, the fact that the covariance matrix shows a relation between components will not only imply a corresponding relation between the components *with probability 1*, but actually *deterministically*! So, we emphasize that (B.148) holds deterministically and not only with probability 1. ♢

The above example generalizes to the following proposition.

Proposition B.32. If the zero-mean Gaussian n-vector $\mathbf{X}$ has the representation (B.84) with an $n \times m$ matrix A of rank r, then it can also be represented in the

following way. Choose r linearly independent rows of A, say $1 \le j_1, \dots, j_r \le n$. Express the vector $\tilde{\mathbf{X}} = (X_{j_1}, \dots, X_{j_r})^{\mathsf{T}}$ as $\tilde{\mathsf{A}}\mathbf{W}$ where $\tilde{\mathsf{A}}$ is an $r \times m$ matrix consisting of the r rows numbered $j_1, \dots, j_r$ of A. The remaining components of $\mathbf{X}$ can be described as deterministic linear combinations of $X_{j_1}, \dots, X_{j_r}$.

Note that it is possible to reduce m to r using the canonical representation of $\tilde{\mathbf{X}}$.

If one of the components of a random vector is a linear combination of the others, then the random vector has no density function. Thus, if one wishes to work with densities, we need to keep the linear dependency "on the side".

We are now ready to use the change of density formula to compute the density of

$$\mathbf{X} = \mathsf{U}\Lambda^{1/2}\mathbf{W}, \tag{B.152}$$

where as discussed we assume that the diagonal matrix Λ has only positive entries. Let

$$\mathsf{A} \triangleq \mathsf{U}\Lambda^{1/2} \tag{B.153}$$

so that the density we are after is the density of $\mathsf{A}\mathbf{W}$. Using the formula for computing the density of $\mathsf{A}\mathbf{W}$ from that of $\mathbf{W}$ and noting that

$$\mathsf{A}\mathsf{A}^{\mathsf{T}} = \mathsf{U}\Lambda^{1/2}\big(\mathsf{U}\Lambda^{1/2}\big)^{\mathsf{T}} = \mathsf{U}\Lambda^{1/2}\Lambda^{1/2}\mathsf{U}^{\mathsf{T}} = \mathsf{U}\Lambda\mathsf{U}^{\mathsf{T}} = \mathsf{K} \tag{B.154}$$

we have

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{f_{\mathbf{W}}(\mathsf{A}^{-1}\mathbf{x})}{|\det(\mathsf{A})|} \tag{B.155}$$

$$= \frac{\exp\Big(-\frac{1}{2}(\mathsf{A}^{-1}\mathbf{x})^{\mathsf{T}}(\mathsf{A}^{-1}\mathbf{x})\Big)}{(2\pi)^{n/2}|\det(\mathsf{A})|} \tag{B.156}$$

$$= \frac{\exp\Big(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}(\mathsf{A}^{-1})^{\mathsf{T}}(\mathsf{A}^{-1}\mathbf{x})\Big)}{(2\pi)^{n/2}|\det(\mathsf{A})|} \tag{B.157}$$

$$= \frac{\exp\Big(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}(\mathsf{A}\mathsf{A}^{\mathsf{T}})^{-1}\mathbf{x}\Big)}{(2\pi)^{n/2}|\det(\mathsf{A})|} \tag{B.158}$$

$$= \frac{\exp\Big(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathsf{K}^{-1}\mathbf{x}\Big)}{(2\pi)^{n/2}|\det(\mathsf{A})|}. \tag{B.159}$$

We now write $|\det(\mathsf{A})|$ in terms of the determinant of the covariance matrix K as

$$|\det(\mathsf{A})| = \sqrt{\det(\mathsf{A}) \cdot \det(\mathsf{A})} \tag{B.160}$$

$$= \sqrt{\det(\mathsf{A}) \cdot \det(\mathsf{A}^{\mathsf{T}})} \tag{B.161}$$

$$= \sqrt{\det(\mathsf{A}\mathsf{A}^{\mathsf{T}})} \tag{B.162}$$

$$= \sqrt{\det(\mathsf{K})} \tag{B.163}$$

to obtain that if $X \sim \mathcal{N}(\boldsymbol{\mu}, \mathsf{K})$ where K is nonsingular, then

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{\exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\mathsf{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big)}{\sqrt{(2\pi)^n \det(\mathsf{K})}}, \quad \mathbf{x} \in \mathbb{R}^n. \tag{B.164}$$

B.10 Linear Functions of Gaussian Vectors

From the definition of the multivariate distribution, it follows quite easily that if $\mathbf{X}$ is an n-dimensional Gaussian vector and if C is an $m \times n$ deterministic matrix, then the m-dimensional vector $\mathsf{C}\mathbf{X}$ is also a Gaussian vector (see also (B.91)). By considering the case $m = 1$ we obtain that if $\boldsymbol{\alpha}$ is any deterministic n-dimensional vector then $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{X}$ is a one-dimensional Gaussian vector. But the components of a Gaussian vector are univariate Gaussians, and we thus conclude that the sole component of $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{X}$ is a univariate Gaussian, i.e., that the random variable $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{X}$ has a univariate Gaussian distribution. Note that the mapping $\mathbf{x} \mapsto \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{x}$ is a linear mapping from the d-dimensional Euclidean space to the real line. Such a mapping is called a *linear function*.

We shall next show that the reverse is also true. If $\mathbf{X}$ is an n-dimensional random vector such that for every deterministic $\boldsymbol{\alpha} \in \mathbb{R}^n$ the random variable $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{X}$ has a univariate Gaussian distribution, then $\mathbf{X}$ must be a Gaussian vector.

We have the following theorem.

Theorem B.33. The n-dimensional random vector $\mathbf{X}$ is Gaussian if, and only if, for every deterministic n-vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ the random variable $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{X}$ has a univariate Gaussian distribution.

Proof: We thus assume that $\mathbf{X}$ is an n-dimensional random vector such that for every deterministic $\boldsymbol{\alpha} \in \mathbb{R}^n$ the random variable $\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{X}$ has a univariate Gaussian distribution and proceed to prove that $\mathbf{X}$ must have a multivariate Gaussian distribution. Let $\mathsf{K}_{\mathbf{XX}}$ be the covariance matrix of $\mathbf{X}$, and let $\boldsymbol{\mu}_{\mathbf{X}}$ denote its mean. Let $\boldsymbol{\omega} \in \mathbb{R}^n$ be arbitrary, and consider the random variable $\boldsymbol{\omega}^{\mathsf{T}}\mathbf{X}$. This is a random variable of mean $\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}_{\mathbf{X}}$ and of covariance $\boldsymbol{\omega}^{\mathsf{T}}\mathsf{K}_{\mathbf{XX}}\boldsymbol{\omega}$; see Lemma B.11. But we assumed that any linear functional of $\mathbf{X}$ has a Gaussian distribution, so

$$\boldsymbol{\omega}^{\mathsf{T}}\mathbf{X} \sim \mathcal{N}(\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}_{\mathbf{X}}, \boldsymbol{\omega}^{\mathsf{T}}\mathsf{K}_{\mathbf{XX}}\boldsymbol{\omega}). \tag{B.165}$$

Consequently, by the characteristic function of the univariate Gaussian distribution (A.76) we conclude that

$$\mathsf{E}\Big[e^{\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\mathbf{X}}\Big] = e^{-\frac{1}{2}\boldsymbol{\omega}^{\mathsf{T}}\mathsf{K}_{\mathbf{XX}}\boldsymbol{\omega}+\mathrm{i}\boldsymbol{\omega}^{\mathsf{T}}\boldsymbol{\mu}_{\mathbf{X}}}, \quad \boldsymbol{\omega} \in \mathbb{R}^n. \tag{B.166}$$

But this also represents the characteristic function of $\mathbf{X}$. According to (B.138) this is identical to the characteristic function of a multivariate Gaussian of mean $\boldsymbol{\mu}_{\mathbf{X}}$ and covariance matrix $\mathsf{K}_{\mathbf{XX}}$. Consequently, since two random vectors that have the same characteristic function must have the same law (B.67), it follows that $\mathbf{X}$ must be a Gaussian vector. □

B.11 Summary

Top things to remember about the multivariate centered Gaussian distribution are as follows:

1. The multivariate centered Gaussian distribution is a generalization of the univariate centered Gaussian distribution in the following sense. A random vector that has only one component, i.e., that takes value in $\mathbb{R}$, has a centered multivariate Gaussian distribution if, and only if, its sole component — when viewed as a scalar random variable — has a univariate $\mathcal{N}(0,\sigma^2)$ distribution for some $\sigma^2 \geq 0$.

2. Moreover, every component of a multivariate centered Gaussian vector has a centered univariate Gaussian distribution.

3. The reverse, however, is not true. Just because each component of a random vector has a univariate centered Gaussian distribution, it does not follow that the vector has a multivariate centered Gaussian distribution. (**ATTENTION!**). Compare also with Points 11 and 12 below.

4. Given any $n \times n$ positive semidefinite matrix $\mathsf{K} \succeq 0$, there exists a *unique* multivariate centered Gaussian distribution of the prescribed covariance matrix K. This distribution is denoted $\mathcal{N}(\mathbf{0},\mathsf{K})$.

5. To generate a random n-vector $\mathbf{X}$ of the law $\mathcal{N}(\mathbf{0},\mathsf{K})$ one can start with a random n-vector $\mathbf{W}$ whose components are IID $\sim \mathcal{N}(0,1)$ and set

$$\mathbf{X} = \mathsf{U}\Lambda^{1/2}\mathbf{W}, \tag{B.167}$$

   where the matrix U is orthogonal; the matrix Λ is diagonal and

$$\mathsf{K}\mathsf{U} = \mathsf{U}\Lambda. \tag{B.168}$$

   Here $\Lambda^{1/2}$ is a diagonal matrix whose diagonal elements are the nonnegative square-roots of the corresponding diagonal entries of Λ.

6. If $\mathbf{X} \sim \mathcal{N}(\mathbf{0},\mathsf{K})$ where K is positive definite ($\mathsf{K} \succ 0$) and U and Λ are as above, then

$$\Lambda^{-1/2}\mathsf{U}^{\mathsf{T}}\mathbf{X} \sim \mathcal{N}(\mathbf{0},\mathsf{I}_n). \tag{B.169}$$

7. If $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathsf{K})$ where K is positive definite ($\mathsf{K} \succ 0$), then $\mathbf{X}$ has the density

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\mathsf{K})}}\, e^{-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathsf{K}^{-1}\mathbf{x}}. \tag{B.170}$$

   This can be interpreted graphically as follows. Let $\mathbf{W}$, U and Λ be as above. Note that the condition $\mathsf{K} \succ 0$ is equivalent to the condition that the diagonal entries of Λ and $\Lambda^{1/2}$ are strictly positive. The level lines of the density of $\mathbf{W}$ are centered circles; those of the density of $\Lambda^{1/2}\mathbf{W}$ are ellipsoids of principal axes parallel to the Cartesian axes; and those of $\mathsf{U}\Lambda^{1/2}\mathbf{W}$ are rotated versions of those of $\Lambda^{1/2}\mathbf{W}$. Compare with Figures B.1 and B.2.

8. If the covariance matrix $\mathsf{K} \succeq 0$ is not positive definite (but only positive semidefinite), then the $\mathcal{N}(\mathbf{0}, \mathsf{K})$ distribution does not have a density. However in this case, if $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathsf{K})$, then a subset of the components of $\mathbf{X}$ have a jointly Gaussian distribution with a density and the rest of the components of $\mathbf{X}$ can be expressed as deterministic linear combinations of the components in the subset. Indeed, if columns $\ell_1, \dots, \ell_d$ are linearly independent and span all the columns of K, then every component of $\mathbf{X}$ can be expressed as a deterministic linear combination of $X_{\ell_1}, \dots, X_{\ell_d}$ and the random d-vector $\tilde{\mathbf{X}} = (X_{\ell_1}, \dots, X_{\ell_d})^{\mathsf{T}}$ has a centered Gaussian distribution with a density. In fact, $\tilde{\mathbf{X}}$ has a $\mathcal{N}(\mathbf{0}, \tilde{\mathsf{K}})$ distribution where the $d \times d$ covariance matrix $\tilde{\mathsf{K}}$ is positive definite and can be computed from K by picking from it rows $\ell_1, \dots, \ell_d$ and columns $\ell_1, \dots, \ell_d$.

9. If C is a deterministic $m \times n$ matrix and the random n-vector $\mathbf{X}$ has a $\mathcal{N}(\mathbf{0}, \mathsf{K})$ distribution, then the random m-vector $\mathsf{C}\mathbf{X}$ is also a centered Gaussian vector. Its $m \times m$ covariance matrix is $\mathsf{C}\mathsf{K}\mathsf{C}^{\mathsf{T}}$.

10. Consequently:

    - Any random vector that results from permuting the components of a centered Gaussian vector is also a centered Gaussian vector.
    - Picking a subset of the components of a Gaussian vector and stacking them into a vector results in a centered Gaussian vector.
    - If the covariance matrix of a centered Gaussian $\mathbf{X}$ is a scalar multiple of the identity matrix I, then the distribution of $\mathbf{X}$ is invariant under multiplication by a deterministic orthogonal matrix U.

11. Let the random n_1-vector $\mathbf{X}_1$ be $\mathcal{N}(\mathbf{0}, \mathsf{K}_1)$ distributed and let the random n_2-vector $\mathbf{X}_2$ be $\mathcal{N}(\mathbf{0}, \mathsf{K}_2)$ distributed. Then if $\mathbf{X}_1$ and $\mathbf{X}_2$ are independent, the $(n_1 + n_2)$-vector that results when the vector $\mathbf{X}_1$ is

stacked on top of the vector $\mathbf{X}_2$ has a multivariate centered Gaussian distribution:

$$\begin{pmatrix}\mathbf{X}_1\\ \mathbf{X}_2\end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \mathsf{K}), \qquad \mathsf{K} = \begin{pmatrix}\mathsf{K}_1 & 0\\ 0 & \mathsf{K}_2\end{pmatrix}. \tag{B.171}$$

(Here K is an $(n_1+n_2)\times(n_1+n_2)$ matrix.)

12. The above extends to the case where more than two independent Gaussian vectors are stacked into a longer vector. In particular, if the components of a random vector are independent and they each have a univariate Gaussian distribution, then the random vector has a multivariate Gaussian distribution with a diagonal covariance matrix.

13. If a subset of the components of a Gaussian vector are uncorrelated with a different subset of the components of the same vector, then the two subsets are statistically independent.

14. If the components of a Gaussian vector are independent in pairs, then they are independent.

Appendix C

Stochastic Processes

Stochastic processes are not really difficult to understand when viewed as a family of random variables that is parametrized by the time index t. However, there are a few mathematical subtleties concerning measurability, integrability, existence, and certain nasty events of zero Lebesgue measure. Our treatment here will mostly ignore these issues and try to concentrate on the main engineering picture. We highly recommend, however, to have a closer look at the exact treatment in [Lap17, Chapter 25].

C.1 Stochastic Processes & Stationarity

A *stochastic process* $X(t,\omega)$ is a generalization of random variables: It is a a collection of random variables that is indexed by the parameter t. So, for every fixed value t, $X(t,\cdot)$ is a random variable, i.e., a function that maps the outcomes of a random experiment to a real number:

$$X(t,\cdot)\colon \Omega \to \mathbb{R},\ \omega \mapsto X(t,\omega). \tag{C.1}$$

On the other hand, for a fixed $\omega \in \Omega$, $X(\cdot,\omega)$ denotes the realizations of all these random variables, i.e., we have now a function of time:

$$X(\cdot,\omega)\colon \mathbb{R} \to \mathbb{R},\ t \mapsto X(t,\omega). \tag{C.2}$$

This function is usually called *sample-path*, *sample-function*, or *realization*. Similar to the case of random variables, it is usual to drop ω and simply write $X(t)$ for $t \in \mathbb{R}$. Here the capitalization of X points to the fact that a random experiment is behind this function.

More formally, we have the following definition.

Definition C.1. A *stochastic process* $\{X(t)\}_{t\in\mathcal{T}}$ is an indexed family of random variables that are defined on a common probability space $(\Omega,\mathcal{F},P)$. Here $\mathcal{T}$ denotes the indexing set that usually will be $\mathcal{T}=\mathbb{R}$, denoting continuous time, or $\mathcal{T}=\mathbb{Z}$, denoting discrete time.

In the following we will concentrate on continuous-time stochastic processes, i.e., we will assume that $\mathcal{T} = \mathbb{R}$.

In order to make sure that a stochastic process $\{X(t)\}$ is clearly defined, we need to specify a rule from which all joint distribution functions can, at least in principle, be calculated. That is, for all positive integers n, and all choices of n epochs $t_1, t_2, \dots, t_n$, it must be possible to find the joint cumulative distribution function (CDF),

$$F_{X(t_1),\dots,X(t_n)}(x_1,\dots,x_n) \triangleq \Pr[X(t_1) \le x_1, \dots, X(t_n) \le x_n]. \tag{C.3}$$

Once such a rule is given, any other property of the stochastic process can be computed. For example, the mean $\mathsf{E}[X(t)] = \overline{X}(t)$ and the covariance $\mathsf{Cov}[X(t), X(u)]$ of the process are specified by the joint distribution functions. In particular,

$$\mathsf{Cov}[X(t), X(u)] \triangleq \mathsf{E}\Big[\Big(X(t) - \overline{X}(t)\Big)\Big(X(u) - \overline{X}(u)\Big)\Big]. \tag{C.4}$$

Representing $\{X(t)\}$ in terms of its mean $\overline{X}(\cdot)$, and its fluctuation

$$\tilde{X}(t) \triangleq X(t) - \overline{X}(t), \quad t \in \mathbb{R}, \tag{C.5}$$

the covariance simplifies to

$$\mathsf{Cov}[X(t), X(u)] = \mathsf{E}[\tilde{X}(t)\,\tilde{X}(u)]. \tag{C.6}$$

Most noise functions have zero mean, in which case $\{\tilde{X}(t)\} = \{X(t)\}$ and the covariance therefore $\mathsf{Cov}[X(t), X(u)] = \mathsf{E}[X(t)\,X(u)]$.

Example C.2. Let $\dots, S_{-1}, S_0, S_1, \dots$ be a sequence of IID random variables and define the pulse $h(t) \triangleq \mathbb{1}\{0 \le t < \mathsf{T}\}$. Then

$$X(t) = \sum_{\ell=-\infty}^{\infty} S_\ell\, h(t - \ell\mathsf{T}) \tag{C.7}$$

is a stochastic process where the sample functions are piecewise constant over intervals of size T. The mean is $\mathsf{E}[X(t)] = \mathsf{E}[S_\ell]$ for $t \in [\ell\mathsf{T}, (\ell+1)\mathsf{T})$. Since $\{S_\ell\}$ is IID, this means that

$$\mathsf{E}[X(t)] = \mathsf{E}[S], \quad \forall t. \tag{C.8}$$

The covariance is nonzero only for t and u in the same interval, i.e.,

$$\mathsf{Cov}[X(t), X(u)] = \mathsf{Var}[S_\ell] \tag{C.9}$$

if, for some integer ℓ, both t and u lie in the interval $[\ell\mathsf{T}, (\ell+1)\mathsf{T})$. Otherwise $\mathsf{Cov}[X(t), X(u)] = 0$:

$$\mathsf{Cov}[X(t), X(u)] = \begin{cases} \mathsf{Var}[S] & \text{if } \exists\, \ell \in \mathbb{Z} \text{ s.t. } t, u \in [\ell\mathsf{T}, (\ell+1)\mathsf{T}), \\ 0 & \text{otherwise.} \end{cases} \tag{C.10}$$

$\Diamond$

We see from Definition C.1 that it is not a simple matter to properly define a general stochastic process: We need to specify infinitely many different joint distributions! To ease our life, we will have to make some more assumptions that will allow to simplify the specification and analysis of a stochastic process.

Definition C.3. A stochastic process $\{X(t)\}$ is called *stationary* or *strict-sense stationary (SSS)* if for every $n \in \mathbb{N}$, any epochs $t_1, \ldots, t_n \in \mathbb{R}$, and every $\tau \in \mathbb{R}$,

$$(X(t_1+\tau), \ldots, X(t_n+\tau)) \overset{\mathscr{L}}{=} (X(t_1), \ldots, X(t_n)), \tag{C.11}$$

where "$\overset{\mathscr{L}}{=}$" stands for "equal in law", i.e., the random vectors on both sides have the same probability distribution.

In words this means that a stationary stochastic process will not change its probability distribution over time. In particular this means that, e.g., the mean $\overline{X}(t)$ must be constant (i.e., not depend on t) and the covariance $\mathsf{Cov}[X(t), X(u)]$ must be a function of the difference $t-u$ only.

A weaker condition that nevertheless proves useful in many situations is *wide-sense stationarity*.

Definition C.4. A stochastic process $\{X(t)\}$ is called *weakly stationary* or *wide-sense stationary (WSS)* if the following conditions hold:

1. The variance of $X(t)$ is finite.
2. The mean of $X(t)$ is constant for all $t \in \mathbb{R}$:
$$\mathsf{E}[X(t)] = \mathsf{E}[X(t+\tau)], \quad \forall t, \tau \in \mathbb{R}. \tag{C.12}$$
3. The covariance of $X(t)$ satisfies
$$\mathsf{Cov}[X(t), X(u)] = \mathsf{Cov}[X(t+\tau), X(u+\tau)], \quad \forall t, u, \tau \in \mathbb{R}. \tag{C.13}$$

In the following we will usually use *stationary* to refer to SSS processes, but *WSS* to refer to weakly stationary processes.

From our discussion above we immediately get the following proposition.

Proposition C.5. Every finite-variance stationary stochastic process is WSS.

However, note that the reverse is not true: Some WSS processes are not stationary.

C.2 Autocovariance Function

We next restrict ourselves to WSS processes. For such processes we can define the autocovariance function.

Definition C.6. The *autocovariance function* $\mathsf{K}_{XX}\colon \mathbb{R} \to \mathbb{R}$ of a WSS stochastic process $\{X(t)\}$ is defined for every $\tau \in \mathbb{R}$ by

$$\mathsf{K}_{XX}(\tau) \triangleq \mathsf{Cov}[X(t), X(t+\tau)], \tag{C.14}$$

where the right-hand side does not depend on t because $\{X(t)\}$ is assumed to be WSS.

Remark C.7. Many people like to call $\mathsf{K}_{XX}(\cdot)$ *autocorrelation function*. This, however, is strictly speaking not correct because correlation is normalized to have values between -1 and 1, i.e., the autocorrelation function is the autocovariance function divided by the variance of the process:

$$\rho_{XX}(\tau) \triangleq \frac{\mathsf{Cov}[X(t), X(t+\tau)]}{\mathsf{Var}[X(t)]}, \quad \tau \in \mathbb{R}. \tag{C.15}$$

(Note again that the right-hand side does not depend on t because $\{X(t)\}$ is assumed to be WSS.) To increase the confusion even further, the term "autocorrelation function" is also used to describe the function

$$\mathsf{R}_{ss} : \tau \mapsto \int_{-\infty}^{\infty} s(t+\tau)s(t)\,\mathrm{d}t \tag{C.16}$$

for some *deterministic* function $s(\cdot)$. I prefer to call the function (C.16) *self-similarity function* (following the suggestion of Prof. Amos Lapidoth and Prof. James L. Massey). △

The following properties of the autocovariance function are analogous to Theorem B.12.

Theorem C.8 (Properties of the Autocovariance Function).

1. The autocovariance function $\mathsf{K}_{XX}(\cdot)$ of a continuous-time WSS process $\{X(t)\}$ is a **positive definite function**, i.e., for every $n \in \mathbb{N}$, and for every choice of coefficients $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ and epochs $t_1, \ldots, t_n \in \mathbb{R}$:
$$\sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i \alpha_{i'}\, \mathsf{K}_{XX}(t_i - t_{i'}) \geq 0. \tag{C.17}$$
2. The autocovariance function $\mathsf{K}_{XX}(\cdot)$ of a continuous-time WSS process $\{X(t)\}$ is a **symmetric function**:
$$\mathsf{K}_{XX}(\tau) = \mathsf{K}_{XX}(-\tau), \quad \tau \in \mathbb{R}. \tag{C.18}$$

3. Every symmetric positive definite function is the autocovariance function of some WSS stochastic process.[1]

Proof: To prove Part 2 we calculate

$$\begin{aligned} \mathsf{K}_{XX}(\tau) &= \mathsf{Cov}[X(t+\tau), X(t)] && \text{(C.19)}\\ &= \mathsf{Cov}\,[X(t'), X(t'-\tau)] && \text{(C.20)}\\ &= \mathsf{Cov}\,[X(t'-\tau), X(t')] && \text{(C.21)}\\ &= \mathsf{K}_{XX}(-\tau), && \text{(C.22)} \end{aligned}$$

where we have defined $t' = t + \tau$ and used that $\mathsf{Cov}[X, Y] = \mathsf{Cov}[Y, X]$.

To prove Part 1 we compute

$$\begin{aligned} \sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i \alpha_{i'}\, \mathsf{K}_{XX}(t_i - t_{i'}) &= \sum_{i=1}^{n}\sum_{i'=1}^{n} \alpha_i \alpha_{i'}\, \mathsf{Cov}[X(t_i), X(t_{i'})] && \text{(C.23)}\\ &= \mathsf{Cov}\left[\sum_{i=1}^{n} \alpha_i X(t_i), \sum_{i'=1}^{n} \alpha_{i'} X(t_{i'})\right] && \text{(C.24)}\\ &= \mathsf{Var}\left[\sum_{i=1}^{n} \alpha_i X(t_i)\right] && \text{(C.25)}\\ &\geq 0. && \text{(C.26)} \end{aligned}$$

We omit the proof of Part 3 and refer to [Lap17, Chapter 25] for more details. □

C.3 Gaussian Processes

The most useful stochastic processes for representing noise are the Gaussian processes.

Definition C.9. A stochastic process is *Gaussian* if, for every $n \in \mathbb{N}$ and all choices of epochs $t_1, \ldots, t_n \in \mathbb{R}$, the random vector $(X(t_1), \ldots, X(t_n))^{\mathsf{T}}$ is jointly Gaussian.

From our discussion about Gaussian vectors in Appendix B we know that a Gaussian vector has a finite variance and is completely specified by its mean and covariance matrix. More precisely, we have the following.

[1] Actually, it is always possible to choose this WSS process to be stationary Gaussian. See Definition C.9 of a Gaussian process below.

Corollary C.10 (Properties of Gaussian Processes).

1. A Gaussian process has a finite variance.
2. A Gaussian process is stationary if, and only if, it is WSS.
3. A stationary Gaussian process is fully specified by its autocovariance function and its mean.

Proof: The proof follows in a straightforward way from our results in Appendix B. We omit the details. □

Example C.11. The process in Example C.2 is a Gaussian process if the underlying variables $\{S_\ell\}$ are Gaussian. However, note that this process is neither stationary nor WSS. This can be seen if we realize that from (C.10) we have

$$\mathsf{Cov}\left[X\left(\frac{\mathsf{T}}{4}\right), X\left(\frac{3\mathsf{T}}{4}\right)\right] = \mathsf{Var}[S], \tag{C.27}$$

but

$$\mathsf{Cov}\left[X\left(\frac{3\mathsf{T}}{4}\right), X\left(\frac{5\mathsf{T}}{4}\right)\right] = 0, \tag{C.28}$$

even though for both cases

$$\frac{3\mathsf{T}}{4} - \frac{\mathsf{T}}{4} = \frac{5\mathsf{T}}{4} - \frac{3\mathsf{T}}{4} = \frac{\mathsf{T}}{2}. \tag{C.29}$$

Thus, Part 3 of Definition C.4 is violated. ◊

C.4 Power Spectral Density

It is quite common under engineers to define the *power spectral density (PSD)* of a WSS stochastic process as the Fourier transform of its autocovariance function. There is nothing wrong with this, however, it does not really explain why it should be so. Note that the PSD has a clear engineering meaning: It is a power *density* in the frequency domain! This means that when integrated over a frequency interval, it describes the amount of power within this interval. So a practical, operational definition of the PSD should state that the PSD is a function that describes the amount of power contained in the signal in various frequency bands.

Luckily, it can be shown that for WSS processes this (operational) power spectral density coincides with the Fourier transform of the autocovariance function.

Theorem C.12. Let $\{X(t)\}$ be a zero-mean WSS stochastic process with continuous autocovariance function $\mathsf{K}_{XX}(\tau)$, $\tau \in \mathbb{R}$. Let $\mathsf{S}_{XX}(f)$, $f \in \mathbb{R}$, denote the double-sided power spectral density function of $\{X(t)\}$. Then:

1. $\mathsf{S}_{XX}(\cdot)$ is the Fourier transform of $\mathsf{K}_{XX}(\cdot)$.
2. For every linear filter with integrable impulse response $h(\cdot)$, the power of the filter output $\{Y(t) = (\mathbf{X} \star \mathbf{h})(t)\}$ when fed at the input by $X(t)$ is

$$\text{Power of } (\mathbf{X} \star \mathbf{h})(\cdot) = \int_{-\infty}^{\infty} \mathsf{S}_{XX}(f) |\hat{h}(f)|^2 \, \mathrm{d}f, \tag{C.30}$$

   where $\hat{h}(\cdot)$ denotes the Fourier transform transform[2] of $h(\cdot)$.

Proof: We ignore the mathematical details and refer to [Lap17, Chapters 15 & 25] again. The proof of Part 2 follows from Theorem C.16 below. □

Note that in (C.30) we can choose the filter to be a bandpass filter of very narrow band and thereby measure the power contents of $\{X(t)\}$ inside this band: For

$$\hat{h}(f) = \begin{cases} 1 & \text{if } f \in [f_0, f_1], \\ 0 & \text{otherwise} \end{cases} \tag{C.31}$$

we have that the total power of the output signal $\{Y(t) = (\mathbf{X} \star \mathbf{h})(t)\}$ is

$$\int_{f_0}^{f_1} \mathsf{S}_{XX}(f) \, \mathrm{d}f. \tag{C.32}$$

Hence, $\mathsf{S}_{XX}(f)$ represents the density of power inside a frequency band. For a more detailed description of linear filtering of stochastic processes we refer to Section C.6.

C.5 Linear Functionals of WSS Stochastic Processes

Most of the processing that must be done on noise waveforms involves either linear filters or linear functionals. We start by looking at the latter. Concretely, we are interested in integrals of the form

$$\int_{-\infty}^{\infty} X(t) s(t) \, \mathrm{d}t \tag{C.33}$$

for some WSS process $\{X(t)\}$ and for a (properly well-behaved) deterministic function $s(\cdot)$. Ignoring some mathematical subtleties,[3] we think of (C.33) as

[2] Since capital letters are already reserved for random quantities, in this script we use a hat to denote the Fourier transform.

[3] For example, we have to worry about the measurability of the stochastic process and the integrability of the deterministic function. Moreover, even for such nice assumptions, there might be some realizations of the stochastic process for which the integral is unbounded. However, one can show that such events have zero probability. For more see [Lap17, Chapter 25].

a random variable, i.e., we define

$$Y \triangleq \int_{-\infty}^{\infty} X(t)\, s(t)\, \mathrm{d}t. \tag{C.34}$$

Heuristically, we can derive the mean of Y as follows:

$$\begin{aligned}
\mathsf{E}[Y] &= \mathsf{E}\left[\int_{-\infty}^{\infty} X(t)\, s(t)\, \mathrm{d}t\right] && \text{(C.35)}\\
&= \int_{-\infty}^{\infty} \mathsf{E}[X(t)]\, s(t)\, \mathrm{d}t && \text{(C.36)}\\
&= \int_{-\infty}^{\infty} \mathsf{E}[X(0)]\, s(t)\, \mathrm{d}t && \text{(C.37)}\\
&= \mathsf{E}[X(0)] \int_{-\infty}^{\infty} s(t)\, \mathrm{d}t, && \text{(C.38)}
\end{aligned}$$

where we have used the linearity of expectation and the WSS assumption of $\{X(t)\}$. Similarly, writing $\mu \triangleq \mathsf{E}[X(0)]$ and $\tilde{X}(t) = X(t) - \mu$, we get

$$\begin{aligned}
\mathsf{Var}[Y] &= \mathsf{Var}\left[\int_{-\infty}^{\infty} X(t)\, s(t)\, \mathrm{d}t\right] && \text{(C.39)}\\
&= \mathsf{Var}\left[\int_{-\infty}^{\infty} (\tilde{X}(t) + \mu)\, s(t)\, \mathrm{d}t\right] && \text{(C.40)}\\
&= \mathsf{Var}\left[\int_{-\infty}^{\infty} \tilde{X}(t)\, s(t)\, \mathrm{d}t + \mu \int_{-\infty}^{\infty} s(t)\, \mathrm{d}t\right] && \text{(C.41)}\\
&= \mathsf{Var}\left[\int_{-\infty}^{\infty} \tilde{X}(t)\, s(t)\, \mathrm{d}t\right] && \text{(C.42)}\\
&= \mathsf{E}\left[\left(\int_{-\infty}^{\infty} \tilde{X}(t)\, s(t)\, \mathrm{d}t\right)^2\right] && \text{(C.43)}\\
&= \mathsf{E}\left[\left(\int_{-\infty}^{\infty} \tilde{X}(t)\, s(t)\, \mathrm{d}t\right)\left(\int_{-\infty}^{\infty} \tilde{X}(\tau)\, s(\tau)\, \mathrm{d}\tau\right)\right] && \text{(C.44)}\\
&= \mathsf{E}\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{X}(t)\, s(t)\, \tilde{X}(\tau)\, s(\tau)\, \mathrm{d}t\, \mathrm{d}\tau\right] && \text{(C.45)}\\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathsf{E}[\tilde{X}(t)\, \tilde{X}(\tau)]\, s(t)\, s(\tau)\, \mathrm{d}t\, \mathrm{d}\tau && \text{(C.46)}\\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(t)\ \mathsf{K}_{XX}(t-\tau)\, s(\tau)\, \mathrm{d}t\, \mathrm{d}\tau. && \text{(C.47)}
\end{aligned}$$

Here in (C.42) we use the fact that adding a constant to a random variable does not change its variance, and the last equality follows from the definition of the autocovariance function.

Note that using the definition of the self-similarity function (C.16), we can write this as follows:

$$\begin{aligned}
\mathsf{Var}[Y] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(t)\ \mathsf{K}_{XX}(t-\tau)\, s(\tau)\, \mathrm{d}t\, \mathrm{d}\tau && \text{(C.48)}\\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(\sigma+\tau)\ \mathsf{K}_{XX}(\sigma)\, s(\tau)\, \mathrm{d}\sigma\, \mathrm{d}\tau && \text{(C.49)}
\end{aligned}$$

$$= \int_{-\infty}^{\infty} \mathsf{K}_{XX}(\sigma) \int_{-\infty}^{\infty} s(\sigma+\tau)\, s(\tau)\, \mathrm{d}\tau\, \mathrm{d}\sigma \tag{C.50}$$

$$= \int_{-\infty}^{\infty} \mathsf{K}_{XX}(\sigma)\ \mathsf{R}_{ss}\, \sigma\, \mathrm{d}\sigma. \tag{C.51}$$

Recalling Parseval's Theorem,

$$\int_{-\infty}^{\infty} g(t)\, h(t)\, \mathrm{d}t = \int_{-\infty}^{\infty} \hat{g}(f)\, \hat{h}^*(f)\, \mathrm{d}f, \tag{C.52}$$

we can express this in the frequency domain as

$$\mathsf{Var}[Y] = \int_{-\infty}^{\infty} \mathsf{S}_{XX}(f)\, |\hat{s}(f)|^2\, \mathrm{d}f. \tag{C.53}$$

Even though these derivations are heuristic and actually need some justification, they are basically correct. We have the following theorem.

Theorem C.13. Let $\{X(t)\}$ be a WSS stochastic process with autocovariance function $\mathsf{K}_{XX}(\cdot)$. Let $s(\cdot)$ be some decent deterministic function. Then the random variable Y in (C.34) is well-defined and has a mean given in (C.38) and a variance given in (C.51) or (C.53).

We continue with integrals of the form $\int_{-\infty}^{\infty} X(t)s(t)\,\mathrm{d}t$, but with the additional assumption that $\{X(t)\}$ is Gaussian. Actually, we even consider a slightly more general form of a linear functional:

$$Y \triangleq \int_{-\infty}^{\infty} X(t)s(t)\,\mathrm{d}t + \sum_{i=1}^{n} \alpha_i X(t_i), \tag{C.54}$$

for some stationary Gaussian process $\{X(t)\}$, a decent deterministic function $s(\cdot)$, some $n \in \mathbb{N}$, and arbitrary coefficients $\alpha_1, \dots, \alpha_n \in \mathbb{R}$ and epochs $t_1, \dots, t_n \in \mathbb{R}$. Following a similar derivation as shown above, one can prove the following theorem.

Theorem C.14. Consider the setup of (C.54) with $\{X(t)\}$ being a stationary Gaussian stochastic process. Then Y is a Gaussian random variable with mean

$$\mathsf{E}[Y] = \mathsf{E}[X(0)] \left(\int_{-\infty}^{\infty} s(t)\,\mathrm{d}t + \sum_{i=1}^{n} \alpha_i \right) \tag{C.55}$$

and variance

$$\begin{aligned}\mathsf{Var}[Y] = {} & \int_{-\infty}^{\infty} \mathsf{K}_{XX}(\sigma)\ \mathsf{R}_{ss}\, \sigma\, \mathrm{d}\sigma + \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'}\, \mathsf{K}_{XX}(t_i - t_{i'}) \\ & + 2 \sum_{i=1}^{n} \alpha_i \int_{-\infty}^{\infty} s(t)\, \mathsf{K}_{XX}(t - t_i)\,\mathrm{d}t.\end{aligned} \tag{C.56}$$

Here, $\mathsf{R}_{ss}\,\cdot$ denotes the self-similarity function of $s(\cdot)$, see (C.16).

Proof: This theorem is a special case of Theorem C.15. □

As a matter of fact, this theorem can be extended as follows.

Theorem C.15. Let the random vector $\mathbf{Y} = (Y_1, \dots, Y_m)^{\mathsf{T}}$ have components

$$Y_j \triangleq \int_{-\infty}^{\infty} X(t)\, s_j(t)\, \mathrm{d}t + \sum_{i=1}^{n_j} \alpha_{j,i} X(t_{j,i}), \quad j = 1, \dots, m, \tag{C.57}$$

where $\{X(t)\}$ is a stationary Gaussian process, $\{s_j(\cdot)\}_{j=1}^m$ are m (decent) deterministic functions, $\{n_j\}_{j=1}^m$ are nonnegative integers, and the coefficients $\{\alpha_{j,i}\}$ and the epochs $\{t_{i,j}\}$ are deterministic real numbers for all $j \in \{1, \dots, m\}$ and all $i \in \{1, \dots, n_j\}$. Then $\mathbf{Y}$ is a Gaussian vector with a covariance matrix whose components can be computed as follows:

$$\begin{aligned}\mathsf{Cov}[Y_j, Y_k] = {} & \int_{-\infty}^{\infty} \mathsf{K}_{XX}(\sigma) \int_{-\infty}^{\infty} s_j(t) s_k(t-\sigma)\, \mathrm{d}t\, \mathrm{d}\sigma \\ & + \sum_{i=1}^{n_j} \alpha_{j,i} (\mathbf{s}_k \star \mathsf{K}_{XX})(t_{j,i}) \\ & + \sum_{i'=1}^{n_k} \alpha_{k,i'} (\mathbf{s}_j \star \mathsf{K}_{XX})(t_{k,i'}) \\ & + \sum_{i=1}^{n_j} \sum_{i'=1}^{n_k} \alpha_{j,i} \alpha_{k,i'}\, \mathsf{K}_{XX}(t_{j,i} - t_{k,i'})\end{aligned} \tag{C.58}$$

or, equivalently,

$$\begin{aligned}\mathsf{Cov}[Y_j, Y_k] = {} & \int_{-\infty}^{\infty} \mathsf{S}_{XX}(f)\, \hat{s}_j(f)\, \hat{s}_k^*(f)\, \mathrm{d}f \\ & + \sum_{i=1}^{n_j} \alpha_{j,i} \int_{-\infty}^{\infty} \mathsf{S}_{XX}(f)\, \hat{s}_k(f)\, e^{\mathrm{i}2\pi f t_{j,i}}\, \mathrm{d}f \\ & + \sum_{i'=1}^{n_k} \alpha_{k,i'} \int_{-\infty}^{\infty} \mathsf{S}_{XX}(f)\, \hat{s}_j(f)\, e^{\mathrm{i}2\pi f t_{k,i'}}\, \mathrm{d}f \\ & + \sum_{i=1}^{n_j} \sum_{i'=1}^{n_k} \alpha_{j,i} \alpha_{k,i'} \int_{-\infty}^{\infty} \mathsf{S}_{XX}(f)\, e^{\mathrm{i}2\pi f (t_{j,i} - t_{k,i'})}\, \mathrm{d}f.\end{aligned} \tag{C.59}$$

Proof: Since by Definition C.9 for any choice of time epochs $t_1, \dots, t_n$, the vector $(X(t_1), \dots, X(t_n))^{\mathsf{T}}$ is Gaussian and since the components Y_j all are linear functionals of such vectors, it is quite intuitive that $\mathbf{Y}$ is a Gaussian vector. We ignore the mathematical subtleties of this proof and refer to [Lap17, Chapter 25]. The expression (C.59) follows from (C.58) using Parseval's Theorem; and (C.58) can be justified as follows:

$$\mathsf{Cov}[Y_j, Y_k] = \mathsf{Cov}\Bigg[\int_{-\infty}^{\infty} X(t)\, s_j(t)\, \mathrm{d}t + \sum_{i=1}^{n_j} \alpha_{j,i} X(t_{j,i}),$$

$$\int_{-\infty}^{\infty} X(\tau)\, s_k(\tau)\,\mathrm{d}\tau + \sum_{i'=1}^{n_k} \alpha_{k,i'} X(t_{k,i'})\Bigg] \tag{C.60}$$

$$\begin{aligned}
&= \mathsf{Cov}\left[\int_{-\infty}^{\infty} X(t)\, s_j(t)\,\mathrm{d}t, \int_{-\infty}^{\infty} X(\tau)\, s_k(\tau)\,\mathrm{d}\tau\right] \\
&\quad + \sum_{i=1}^{n_j} \alpha_{j,i}\, \mathsf{Cov}\left[X(t_{j,i}), \int_{-\infty}^{\infty} X(\tau)\, s_k(\tau)\,\mathrm{d}\tau\right] \\
&\quad + \sum_{i'=1}^{n_k} \alpha_{k,i'}\, \mathsf{Cov}\left[\int_{-\infty}^{\infty} X(t)\, s_j(t)\,\mathrm{d}t, X(t_{k,i'})\right] \\
&\quad + \sum_{i=1}^{n_j}\sum_{i'=1}^{n_k} \alpha_{j,i}\alpha_{k,i'}\, \mathsf{Cov}[X(t_{j,i}), X(t_{k,i'})]
\end{aligned} \tag{C.61}$$

$$\begin{aligned}
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathsf{Cov}[X(t), X(\tau)]\, s_j(t)\, s_k(\tau)\,\mathrm{d}t\,\mathrm{d}\tau \\
&\quad + \sum_{i=1}^{n_j} \alpha_{j,i} \int_{-\infty}^{\infty} \mathsf{Cov}[X(t_{j,i}), X(\tau)]\, s_k(\tau)\,\mathrm{d}\tau \\
&\quad + \sum_{i'=1}^{n_k} \alpha_{k,i'} \int_{-\infty}^{\infty} \mathsf{Cov}[X(t), X(t_{k,i'})]\, s_j(t)\,\mathrm{d}t \\
&\quad + \sum_{i=1}^{n_j}\sum_{i'=1}^{n_k} \alpha_{j,i}\alpha_{k,i'}\, \mathsf{K}_{XX}(t_{j,i} - t_{k,i'})
\end{aligned} \tag{C.62}$$

$$\begin{aligned}
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathsf{K}_{XX}(t-\tau)\, s_j(t)\, s_k(\tau)\,\mathrm{d}t\,\mathrm{d}\tau \\
&\quad + \sum_{i=1}^{n_j} \alpha_{j,i} \int_{-\infty}^{\infty} \mathsf{K}_{XX}(t_{j,i} - \tau)\, s_k(\tau)\,\mathrm{d}\tau \\
&\quad + \sum_{i'=1}^{n_k} \alpha_{k,i'} \int_{-\infty}^{\infty} \mathsf{K}_{XX}(t_{k,i'} - t)\, s_j(t)\,\mathrm{d}t \\
&\quad + \sum_{i=1}^{n_j}\sum_{i'=1}^{n_k} \alpha_{j,i}\alpha_{k,i'}\, \mathsf{K}_{XX}(t_{j,i} - t_{k,i'}).
\end{aligned} \tag{C.63}$$

□

C.6 Filtering Stochastic Processes

Next we discuss the result of passing a WSS stochastic process through a stable[4] linear filter with impulse response $h(\cdot)$. Figure C.1 illustrates the situation.

Since for a fixed t, the output can be written as

$$Y(t) = \int_{-\infty}^{\infty} X(\sigma) h(t-\sigma)\,\mathrm{d}\sigma, \tag{C.64}$$

[4] A linear filter is called *stable* if its impulse response $h(\cdot)$ is integrable.

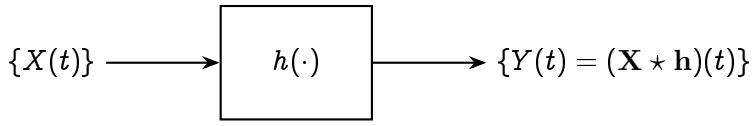


Figure C.1: Filtered stochastic process.

we can (more or less) directly apply our results from the Section C.5 about linear functionals. Again we will ignore some mathematical subtleties here and refer to [Lap17, Chapter 25] for any proofs.

Theorem C.16. Let $\{Y(t)\}$ be the result of passing the centered (zero-mean) WSS stochastic process $\{X(t)\}$ of autocovariance function $\mathsf{K}_{XX}(\cdot)$ through the linear filter with (decent) impulse response $h(\cdot)$. Then we have the following:

1. $\{Y(t)\}$ is a centered WSS stochastic process with autocovariance function
$$\mathsf{K}_{YY}(\tau) = (\mathsf{K}_{XX} \star \mathsf{R}_{hh})(\tau), \quad \tau \in \mathbb{R}. \tag{C.65}$$
where $\mathsf{R}_{hh}\,\cdot$ is the self-similarity function of $h(\cdot)$ (see (C.16)).

2. If $\{X(t)\}$ has PSD S_{XX}, then $\{Y(t)\}$ has PSD
$$\mathsf{S}_{YY}(f) = \bigl|\hat{h}(f)\bigr|^2 \cdot \mathsf{S}_{XX}(f), \quad f \in \mathbb{R}. \tag{C.66}$$

3. For every $t, \tau \in \mathbb{R}$,
$$\mathsf{E}[X(t)Y(t+\tau)] = (\mathsf{K}_{XX} \star \mathbf{h})(\tau), \tag{C.67}$$
where the right-hand side does not depend on t.

4. If $\{X(t)\}$ is Gaussian, then so is $\{Y(t)\}$. Even better, for every choice of $n, m \in \mathbb{N}$ and epochs $t_1, \dots, t_n, t_{n+1}, \dots, t_{n+m} \in \mathbb{R}$, the random vector
$$\bigl(X(t_1), \dots, X(t_n), Y(t_{n+1}), \dots, Y(t_{n+m})\bigr)^{\mathsf{T}} \tag{C.68}$$
is a centered Gaussian random vector. The covariance matrix can be computed using $\mathsf{K}_{XX}(\cdot)$, (C.65), and (C.67).

Note that the joint Gaussianity of the random variables in (C.68) follows from Theorem C.13. Indeed we can express $X(t_j)$ as
$$X(t_j) = \int_{-\infty}^{\infty} X(t) s_j(t)\,\mathrm{d}t + \alpha_j X(t_j), \quad j = 1, \dots, n, \tag{C.69}$$
for $s_j(\cdot)$ the zero function and $\alpha_j = 1$, and $Y(t_j)$ similarly as
$$Y(t_j) = \int_{-\infty}^{\infty} X(t) s_j(t)\,\mathrm{d}t + \alpha_j X(t_j), \quad j = n+1, \dots, n+m, \tag{C.70}$$
for $s_j \colon t \mapsto h(t_j - t)$ and $\alpha_j = 0$.

Also note that Theorem C.12 actually is a consequence of Theorem C.16.

C.7 White Gaussian Noise

The most important stochastic process in digital communication is so-called *white Gaussian noise*. This noise is often used to model additive noise in communication systems and in many other places, too. Sometimes, its use is actually not really justified from a practical point of view, but is motivated merely because the white Gaussian noise can be handled analytically.

Our definition differs from the usual definition found in most textbook in the sense that we specify a certain bandwidth and define the white Gaussian noise only with respect to this given band. The reason for this is that it is impossible to have white Gaussian noise over the whole infinite spectrum as such a process would have infinite power and its autocovariance function would not be defined.[5]

Definition C.17. We say that $\{N(t)\}$ is *white Gaussian noise of double-sided power spectral density* $\frac{\mathrm{N}_0}{2}$ *with respect to the bandwidth* W if $\{N(t)\}$ is a stationary centered Gaussian random process that has PSD $\mathrm{S}_{NN}(\cdot)$ satisfying

$$\mathrm{S}_{NN}(f) = \frac{\mathrm{N}_0}{2}, \quad f \in [-W, W]. \tag{C.71}$$

Note that our definition on purpose leaves the PSD unspecified outside the given band $[-W, W]$.

We can now apply our knowledge from Sections C.5 and C.6 to this definition and get the following key properties of white Gaussian noise.

Corollary C.18. Let $\{N(t)\}$ be white Gaussian noise of double-sided power spectral density $\frac{\mathrm{N}_0}{2}$ with respect to the bandwidth W.

1. If $s(\cdot)$ is a properly well-behaved function that is bandlimited to W Hz, then
$$\int_{-\infty}^{\infty} N(t)s(t)\,\mathrm{d}t \sim \mathcal{N}\left(0, \frac{\mathrm{N}_0}{2}\langle \mathbf{s}, \mathbf{s}\rangle\right), \tag{C.72}$$
where we define the **inner product** of two real functions $u(\cdot)$ and $v(\cdot)$ as
$$\langle \mathbf{u}, \mathbf{v}\rangle \triangleq \int_{-\infty}^{\infty} u(t)v(t)\,\mathrm{d}t. \tag{C.73}$$
2. If $s_1(\cdot), \ldots, s_m(\cdot)$ are properly well-behaved functions that are bandlimited to W Hz, then the m random variables
$$Y_1 \triangleq \int_{-\infty}^{\infty} N(t)s_1(t)\,\mathrm{d}t, \tag{C.74}$$
$$\vdots$$
$$Y_m \triangleq \int_{-\infty}^{\infty} N(t)s_m(t)\,\mathrm{d}t \tag{C.75}$$

[5] The common definition tries to circumvent this issue by using Dirac Deltas. While this might be convenient from an engineering point of view, it causes quite a few mathematical problems because Dirac Deltas are not functions.

are jointly centered Gaussian random variables of covariance matrix

$$\mathsf{K}_{\mathbf{YY}} = \frac{\mathsf{N}_0}{2}\begin{pmatrix} \langle \mathbf{s}_1, \mathbf{s}_1\rangle & \langle \mathbf{s}_1, \mathbf{s}_2\rangle & \cdots & \langle \mathbf{s}_1, \mathbf{s}_m\rangle \\ \langle \mathbf{s}_2, \mathbf{s}_1\rangle & \langle \mathbf{s}_2, \mathbf{s}_2\rangle & \cdots & \langle \mathbf{s}_2, \mathbf{s}_m\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{s}_m, \mathbf{s}_1\rangle & \langle \mathbf{s}_m, \mathbf{s}_2\rangle & \cdots & \langle \mathbf{s}_m, \mathbf{s}_m\rangle \end{pmatrix}. \tag{C.76}$$

3. If $\phi_1(\cdot), \dots, \phi_m(\cdot)$ are properly well-behaved functions that are bandlimited to W Hz and that are **orthonormal**

$$\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_{i'}\rangle = \begin{cases} 1 & \text{if } i = i', \\ 0 & \text{otherwise,} \end{cases} \tag{C.77}$$

then the m random variables

$$\int_{-\infty}^{\infty} N(t)\phi_1(t)\,\mathrm{d}t, \dots, \int_{-\infty}^{\infty} N(t)\phi_m(t)\,\mathrm{d}t \tag{C.78}$$

are IID $\sim \mathcal{N}(0, \mathsf{N}_0/2)$.

4. If $s(\cdot)$ is a properly well-behaved function that is bandlimited to W Hz, and if $\mathsf{K}_{NN}(\cdot)$ is the autocovariance function of $\{N(t)\}$, then

$$(\mathsf{K}_{NN} \star \mathrm{s})(t) = \frac{\mathsf{N}_0}{2} s(t), \quad t \in \mathbb{R}. \tag{C.79}$$

5. If $s(\cdot)$ is a properly well-behaved function that is bandlimited to W Hz, then for every epoch $t \in \mathbb{R}$

$$\mathsf{Cov}\left[\int_{-\infty}^{\infty} N(\tau)s(\tau)\,\mathrm{d}\tau, N(t)\right] = \frac{\mathsf{N}_0}{2} s(t). \tag{C.80}$$

Proof: Parts 1 and 3 are special cases of Part 2, so we directly prove Part 2. Using (C.59) of Theorem C.15 with $\alpha_{j,i} = 0$, we get

$$\begin{aligned} \mathsf{Cov}[Y_j, Y_k] &= \int_{-\infty}^{\infty} \mathsf{S}_{NN}(f)\,\hat{s}_j(f)\,\hat{s}_k^*(f)\,\mathrm{d}f && \text{(C.81)} \\ &= \int_{-W}^{W} \mathsf{S}_{NN}(f)\,\hat{s}_j(f)\,\hat{s}_k^*(f)\,\mathrm{d}f && \text{(C.82)} \\ &= \frac{\mathsf{N}_0}{2}\int_{-W}^{W} \hat{s}_j(f)\,\hat{s}_k^*(f)\,\mathrm{d}f && \text{(C.83)} \\ &= \frac{\mathsf{N}_0}{2}\int_{-\infty}^{\infty} \hat{s}_j(f)\,\hat{s}_k^*(f)\,\mathrm{d}f && \text{(C.84)} \\ &= \frac{\mathsf{N}_0}{2}\int_{-\infty}^{\infty} s_j(t)\,s_k(t)\,\mathrm{d}t, && \text{(C.85)} \end{aligned}$$

where in (C.82) we use that $s_1(\cdot), \dots, s_m(\cdot)$ are bandlimited.

For Part 4 we write $\mathsf{K}_{NN}(\cdot)$ as inverse Fourier transform of $\mathsf{S}_{NN}(\cdot)$:

$$
\begin{aligned}
(\mathsf{K}_{NN} \star \mathrm{s})(t) &= \int_{-\infty}^{\infty} s(\tau)\, \mathsf{K}_{NN}(t-\tau)\, \mathrm{d}\tau && \text{(C.86)}\\
&= \int_{-\infty}^{\infty} s(\tau) \int_{-\infty}^{\infty} \mathsf{S}_{NN}(f)\, e^{\mathrm{i}2\pi f(t-\tau)}\, \mathrm{d}f\, \mathrm{d}\tau && \text{(C.87)}\\
&= \int_{-\infty}^{\infty} \mathsf{S}_{NN}(f) \int_{-\infty}^{\infty} s(\tau)\, e^{-\mathrm{i}2\pi f\tau}\, \mathrm{d}\tau\, e^{\mathrm{i}2\pi ft}\, \mathrm{d}f && \text{(C.88)}\\
&= \int_{-\infty}^{\infty} \mathsf{S}_{NN}(f) \hat{s}(f)\, e^{\mathrm{i}2\pi ft}\, \mathrm{d}f && \text{(C.89)}\\
&= \int_{-W}^{W} \mathsf{S}_{NN}(f) \hat{s}(f)\, e^{\mathrm{i}2\pi ft}\, \mathrm{d}f && \text{(C.90)}\\
&= \frac{\mathsf{N}_0}{2} \int_{-W}^{W} \hat{s}(f)\, e^{\mathrm{i}2\pi ft}\, \mathrm{d}f && \text{(C.91)}\\
&= \frac{\mathsf{N}_0}{2} s(t). && \text{(C.92)}
\end{aligned}
$$

Finally, Part 5 follows from a derivation similar to (C.60)–(C.63) in combination with Part 4. □

C.8 Orthonormal and Karhunen–Loeve Expansions

The following material is pretty advanced and is presented only for those readers who want to go a little more deeply into the topic of representing stochastic processes by orthonormal expansions. Recall that a set of (complex) functions $\phi_1(\cdot), \phi_2(\cdot), \ldots$ is called *orthonormal* if

$$\int_{-\infty}^{\infty} \phi_i(t) \phi_{i'}^*(t)\, \mathrm{d}t = \mathbb{1}\{i = i'\} \tag{C.93}$$

for all integers i, i'. These functions can be complex, but we will mostly stick to real examples. The limits of integration throughout this section will be taken as $(-\infty, \infty)$, but usually the functions of interest are nonzero only over some finite interval $[\mathsf{T}_0, \mathsf{T}_1]$.

The best known such set of orthonormal functions are those in the Fourier series,

$$\phi_n(t) = \frac{1}{\sqrt{\mathsf{T}}}\, e^{\mathrm{i}2\pi n \frac{t}{\mathsf{T}}} \tag{C.94}$$

for the interval $[-\mathsf{T}/2, \mathsf{T}/2]$. We can then take any continuous square-integrable function $x(\cdot)$ over $[-\mathsf{T}/2, \mathsf{T}/2]$ and represent it by

$$x(\cdot) = \sum_i x_i \phi_i(\cdot), \quad \text{where } x_i = \int_{-\infty}^{\infty} x(t) \phi_i^*(t)\, \mathrm{d}t. \tag{C.95}$$

For real functions, one can avoid the complex orthonormal functions by using sines and cosines in place of the complex exponentials.

The nature of this transformation is not due to the special nature of sinusoids, but only due to the fact that the function is being represented as a series of orthonormal functions. To see this, let $\{\phi_i(\cdot)\}$ be any set of real orthonormal functions, and assume that a function $x(\cdot)$ can be represented as

$$x(\cdot) = \sum_i x_i \phi_i(\cdot). \tag{C.96}$$

Multiplying both sides of (C.96) by $\phi_j(\cdot)$ and integrating yields

$$\int_{-\infty}^{\infty} x(t)\phi_j(t)\,\mathrm{d}t = \int_{-\infty}^{\infty} \sum_i x_i \phi_i(t)\phi_j(t)\,\mathrm{d}t. \tag{C.97}$$

Interchanging the order of integration and summation and using (C.93), we get

$$\int_{-\infty}^{\infty} x(t)\phi_j(t)\,\mathrm{d}t = x_j. \tag{C.98}$$

Thus, if a function can be represented by orthonormal functions as in (C.96), then the coefficients $\{x_i\}$ must be determined as in (C.98), which is the same pair of relations as in (C.95). We can also represent the energy in $x(\cdot)$ in terms of the coefficients $\{x_i\}$. Since

$$x^2(t) = \left(\sum_i x_i \phi_i(t)\right) \cdot \left(\sum_j x_j \phi_j(t)\right) \tag{C.99}$$

for all t, we get

$$\int_{-\infty}^{\infty} x^2(t)\,\mathrm{d}t = \int_{-\infty}^{\infty} \sum_i \sum_j x_i x_j \phi_i(t)\phi_j(t)\,\mathrm{d}t = \sum_i x_i^2. \tag{C.100}$$

Next suppose $x(\cdot)$ is any real square-integrable function and $\{\phi_i(\cdot)\}$ is an orthonormal set. Let $x_i \triangleq \int_{-\infty}^{\infty} x(t)\phi_n(t)\,\mathrm{d}t$ and let

$$\epsilon_k(t) \triangleq x(t) - \sum_{i=1}^{k} x_i \phi_i(t) \tag{C.101}$$

be the error when $x(\cdot)$ is represented by the first k of these orthonormal functions. First we show that $\epsilon_k(\cdot)$ is orthogonal to $\phi_j(\cdot)$ for $1 \le j \le k$:

$$\int_{-\infty}^{\infty} \epsilon_k(t)\phi_j(t)\,\mathrm{d}t = \int_{-\infty}^{\infty} x(t)\phi_j(t)\,\mathrm{d}t - \int_{-\infty}^{\infty} \sum_{i=1}^{k} x_i \phi_i(t)\phi_j(t)\,\mathrm{d}t \tag{C.102}$$

$$= x_j - x_j = 0. \tag{C.103}$$

Viewing functions as vectors, $\epsilon_k(\cdot)$ is the difference between $x(\cdot)$ and its projection on the linear subspace spanned by $\{\phi_i(\cdot)\}_{1\le i\le k}$. The integral of the

squared error is given by

$$\int_{-\infty}^{\infty} x^2(t)\,\mathrm{d}t = \int_{-\infty}^{\infty} \left(\epsilon_k(t) + \sum_{i=1}^{k} x_i\phi_i(t)\right)^2 \mathrm{d}t \tag{C.104}$$

$$= \int_{-\infty}^{\infty} \epsilon_k^2(t)\,\mathrm{d}t + \int_{-\infty}^{\infty} \sum_{i=1}^{k}\sum_{j=1}^{k} x_i x_j \phi_i(t)\phi_j(t)\,\mathrm{d}t \tag{C.105}$$

$$= \int_{-\infty}^{\infty} \epsilon_k^2(t)\,\mathrm{d}t + \sum_{i=1}^{k} x_i^2. \tag{C.106}$$

Since $\epsilon_k^2(t) \geq 0$, we have the Bessel Inequality,

$$\sum_{i=1}^{k} x_i^2 \leq \int_{-\infty}^{\infty} x^2(t)\,\mathrm{d}t. \tag{C.107}$$

We see from (C.106) that $\int_{-\infty}^{\infty} \epsilon_k^2(t)\,\mathrm{d}t$ is nonincreasing with k. Thus, in the limit $k \to \infty$, either the energy in $\epsilon_k(\cdot)$ approaches 0 or it approaches some positive constant. A set of orthonormal functions is called *complete* over some class $\mathcal{C}$ of functions if this error energy approaches 0 for all $x(\cdot) \in \mathcal{C}$. For example, the Fourier series set of functions in (C.94) is complete over the set of functions that are square integrable and zero outside of $[-\mathsf{T}/2, \mathsf{T}/2]$. There are many other countable sets of functions that are complete over this class of functions. Such sets are called *complete orthonormal systems (CONS)*.

The question now arises whether a countable set of functions exists that is complete over the square-integrable functions in the interval $(-\infty, \infty)$. The answer is yes. One example starts with the Fourier series functions over $[-\mathsf{T}/2, \mathsf{T}/2]$. This set is extended by adding the time shifts of these functions, shifting by $k\mathsf{T}$ for each integer k. Note that the Fourier integral functions $e^{\mathrm{i}2\pi ft}$ over all $f \in \mathbb{R}$ do not work here, since there are an uncountably infinite number of such functions and they cannot be made orthonormal (i.e., they each have infinite energy).

Parseval's relationship is another useful relationship among functions

$$x(\cdot) = \sum_i x_i \phi(\cdot) \tag{C.108}$$

and

$$y(\cdot) = \sum y_i \phi(\cdot). \tag{C.109}$$

This relationship states that

$$\int_{-\infty}^{\infty} x(t)y(t)\,\mathrm{d}t = \sum_i x_i y_i. \tag{C.110}$$

To derive this, note that

$$\int_{-\infty}^{\infty} \big(x(t) - y(t)\big)^2\,\mathrm{d}t = \sum_i (x_i - y_i)^2. \tag{C.111}$$

Squaring the terms inside brackets and canceling out the terms in x^2 and y^2, we get (C.110).

These same relationships work for stochastic processes. That is, consider a segment of a stochastic process, $X(t)$, $\mathsf{T}_1 \le t \le \mathsf{T}_2$. Assume that the covariance function

$$\mathsf{K}_{XX}(t,\tau) \triangleq \mathsf{Cov}[X(t), X(\tau)] \tag{C.112}$$

is continuous, and represent $X(\cdot)$ by an orthonormal set that is complete over finite-energy functions in the interval $[\mathsf{T}_1, \mathsf{T}_2]$. Then we have

$$X(\cdot) = \sum_i X_i \phi_i(\cdot), \qquad X_i = \int_{-\infty}^{\infty} X(t)\phi_i(t)\,\mathrm{d}t. \tag{C.113}$$

The *Karhunen–Loeve expansion* is an orthonormal expansion using a particular set of orthonormal functions. These orthonormal functions are chosen so that for some particular zero-mean stochastic process $\{X(t)\}$, $\mathsf{T}_1 \le t \le \mathsf{T}_2$, of interest, the random variables $\{X_i\}$ in (C.113) are uncorrelated.

It will be easiest to understand this expansion if we first look at the related problem for a discrete-time, zero-mean stochastic process over a finite interval, say $\{X(1), X(2), \ldots, X(\mathsf{T})\}$. This is simply a set of T random variables with a covariance matrix $\mathsf{K}_{\mathbf{XX}}$. We have seen that $\mathsf{K}_{\mathbf{XX}}$ has a set of T orthonormal eigenvectors $\mathbf{q}_i$, $1 \le i \le \mathsf{T}$. The corresponding eigenvalues are nonnegative, and they are positive if $\mathsf{K}_{\mathbf{XX}}$ is positive definite.

If we visualize sampling the finite-duration ($\mathsf{T}_1 \le t \le \mathsf{T}_2$) continuous-time process $\{X(t)\}$ very finely, at intervals of width δ, then we have the same problem as the discrete-time case above. If we go to the continuous-time limit without worrying about mathematical details, then the eigenvector equation $\mathsf{K}_{\mathbf{XX}}\ \mathbf{q} = \lambda\mathbf{q}$ changes into the integral eigenvalue equation

$$\int_{\mathsf{T}_1}^{\mathsf{T}_2} \mathsf{K}_{XX}(t,\tau)\,\phi(\tau)\,\mathrm{d}\tau = \lambda\phi(t), \quad \mathsf{T}_1 \le t \le \mathsf{T}_2. \tag{C.114}$$

The following theorem is a central result of functional analysis, and is proved, e.g., in [RSN90, p. 242].

Theorem C.19. Assume that $\mathsf{K}_{XX}(\cdot,\cdot)$ is continuous and square integrable with a value greater than 0, i.e.,

$$0 < \int_{\mathsf{T}_1}^{\mathsf{T}_2} \int_{\mathsf{T}_1}^{\mathsf{T}_2} \mathsf{K}_{XX}^2(t,\tau)\,\mathrm{d}t\,\mathrm{d}\tau < \infty. \tag{C.115}$$

Then all solutions of (C.114) have nonnegative eigenvalues λ. There is at least one solution with eigenvalue $\lambda > 0$ and at most a finite number of orthonormal eigenvectors with the same positive eigenvalue. Also, there are

enough eigenfunctions $\{\phi_i(\cdot)\}$ with positive eigenvalues so that for any square-integrable continuous function $x(\cdot)$ over $[\mathsf{T}_1, \mathsf{T}_2]$,

$$x(\cdot) = \sum_i x_i \phi_i(\cdot) + h(\cdot), \tag{C.116}$$

where $h(\cdot)$ is an eigenfunction of eigenvalue 0.

Note that T_1 can be taken to be $-\infty$ and/or T_2 can be taken to be $+\infty$ in the above theorem. (Unfortunately, this cannot be done for a stationary process since $\mathsf{K}_{XX}(\cdot,\cdot)$ would not be square integrable then.) What this theorem says is that the continuous-time case here is almost the same as the matrix case that we studied before. The only difference is that the number of eigenvalues can be countably infinite, and the number of orthonormal eigenvectors with eigenvalue 0 can be countably infinite.

The eigenvectors of different eigenvalues are orthonormal (by essentially the same argument as in the matrix case), so that what (C.116) says is that the set of orthonormal eigenvectors is complete over the square-integrable functions in the interval $[\mathsf{T}_1, \mathsf{T}_2]$. This indicates that there are many possible choices for complete orthonormal sets of functions.

There are a few examples where (C.114) can be solved with relative ease, but usually it is quite difficult to find solutions. But this is not important because the Karhunen–Loeve expansion is usually used to solve detection and estimation problems *abstractly* in terms of this orthonormal expansion, and then convert the solution into something that is meaningful without actually knowing the concrete values of the expansion.

We now proceed to derive some of the properties of this expansion. So, we assume that $\mathsf{K}_{XX}(\cdot,\cdot)$ is continuous and satisfies

$$\int_{\mathsf{T}_1}^{\mathsf{T}_2} \mathsf{K}_{XX}(t,\tau)\, \phi_i(\tau)\, \mathrm{d}\tau = \lambda_i \phi_i(t), \quad t \in [\mathsf{T}_1, \mathsf{T}_2], \tag{C.117}$$

and define the random variables $\{X_i\}$ as

$$X_i \triangleq \int_{\mathsf{T}_1}^{\mathsf{T}_2} X(t)\, \phi_i(t)\, \mathrm{d}t. \tag{C.118}$$

To see that these random variables are uncorrelated, we note that

$$\begin{aligned}
\mathsf{E}[X_i X_j] &= \mathsf{E}\left[\int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2} X(t)\, X(\tau)\, \phi_i(t)\, \phi_j(\tau)\, \mathrm{d}t\, \mathrm{d}\tau\right] && \text{(C.119)}\\
&= \int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2} \mathsf{K}_{XX}(t,\tau)\, \phi_i(t)\, \phi_j(\tau)\, \mathrm{d}t\, \mathrm{d}\tau && \text{(C.120)}\\
&= \int_{\mathsf{T}_1}^{\mathsf{T}_2} \lambda_i \phi_i(t)\, \phi_j(t)\, \mathrm{d}t && \text{(C.121)}\\
&= \lambda_i\, \mathbb{1}\{i=j\}, && \text{(C.122)}
\end{aligned}$$

where in (C.121) we have used (C.117) and where the last equality (C.122) follows from orthogonality.

Next we want to show that

$$\mathsf{K}_{XX}(t,\tau)=\sum_i \lambda_i\phi_i(t)\phi_i(\tau), \tag{C.123}$$

where equality holds almost everywhere. This is called *Mercer's Theorem*. To show this, assume that the eigenvalues of (C.114) are arranged in descending order and consider approximating $\mathsf{K}_{XX}(t,\tau)$ by $\sum_{i=1}^k \lambda_i\phi_i(t)\phi_i(\tau)$. Letting ϵ_k be the integral of the squared error, we have

$$\epsilon_k = \int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2}\left(\mathsf{K}_{XX}(t,\tau)-\sum_{i=1}^k \lambda_i\phi_i(t)\phi_i(\tau)\right)^2 \mathrm{d}t\,\mathrm{d}\tau \tag{C.124}$$

$$= \int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2}\mathsf{K}_{XX}^2(t,\tau)\,\mathrm{d}t\,\mathrm{d}\tau - 2\sum_{i=1}^k\int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2}\lambda_i\phi_i(t)\phi_i(\tau)\,\mathsf{K}_{XX}(t,\tau)\,\mathrm{d}t\,\mathrm{d}\tau$$
$$+\int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2}\sum_{i=1}^k\lambda_i^2\phi_i^2(t)\phi_i^2(\tau)\,\mathrm{d}t\,\mathrm{d}\tau \tag{C.125}$$

$$= \int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2}\mathsf{K}_{XX}^2(t,\tau)\,\mathrm{d}t\,\mathrm{d}\tau - 2\sum_{i=1}^k\int_{\mathsf{T}_1}^{\mathsf{T}_2}\lambda_i^2\phi_i(t)\phi_i(t)\,\mathrm{d}t$$
$$+\sum_{i=1}^k\lambda_i^2\int_{\mathsf{T}_1}^{\mathsf{T}_2}\phi_i^2(t)\,\mathrm{d}t\underbrace{\int_{\mathsf{T}_1}^{\mathsf{T}_2}\phi_i^2(\tau)\,\mathrm{d}\tau}_{=1} \tag{C.126}$$

$$= \int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2}\mathsf{K}_{XX}^2(t,\tau)\,\mathrm{d}t\,\mathrm{d}\tau - \sum_{i=1}^k\lambda_i^2. \tag{C.127}$$

We see from (C.124) that ϵ_k is nonnegative, and from (C.127) that ϵ_k is nonincreasing with k. Thus ϵ_k must reach a limit. We now argue that this limit must be 0. It can be shown that $\mathsf{K}_{XX}(t,\tau)-\sum_{i=1}^k\lambda_i\phi_i(t)\phi_i(\tau)$ is a covariance function for all k, including the limit $k\to\infty$. If $\lim_{k\to\infty}\epsilon_k>0$, then this limiting covariance function has a positive eigenvalue and a corresponding eigenvector which is orthogonal to all the eigenvectors in the expansion $\sum_{i=1}^\infty\lambda_i\phi_i(t)\phi_i(\tau)$. Since we have ordered the eigenvalues in decreasing order and $\lim_{i\to\infty}\lambda_i=0$, this is a contradiction, so $\lim_{k\to\infty}\epsilon_k=0$. Since $\lim_{k\to\infty}\epsilon_k=0$, and since ϵ_k is the integral squared of the difference between $\mathsf{K}_{XX}(t,\tau)$ and $\sum_{i=1}^k\lambda_i\phi_i(t)\phi_i(\tau)$, we see that

$$\mathsf{K}_{XX}(t,\tau)=\sum_{i=1}^\infty\lambda_i\phi_i(t)\phi_i(\tau) \tag{C.128}$$

with equality almost everywhere. As an added bonus, we see from (C.127) that

$$\int_{\mathsf{T}_1}^{\mathsf{T}_2}\int_{\mathsf{T}_1}^{\mathsf{T}_2}\mathsf{K}_{XX}(t,\tau)\,\mathrm{d}t\,\mathrm{d}\tau=\sum_{i=1}^\infty\lambda_i^2. \tag{C.129}$$

Finally, we want to show that

$$X(t) = \sum_{i=1}^{\infty} X_i \phi_i(t), \quad t \in [\mathsf{T}_1, \mathsf{T}_2], \tag{C.130}$$

where the sum is over those i for which $\lambda_i > 0$ and with equality in the sense that the variance of the difference in the two sides of (C.130) is 0. To see this, note that

$$\begin{aligned}
&\mathsf{E}\left[\int_{\mathsf{T}_1}^{\mathsf{T}_2} \left(X(t) - \sum_i X_i \phi_i(t)\right)^2 \mathrm{d}t\right] \\
&\quad = \mathsf{E}\left[\int_{\mathsf{T}_1}^{\mathsf{T}_2} X^2(t)\,\mathrm{d}t\right] - 2\sum_i \mathsf{E}\Big[X_i \underbrace{\int_{\mathsf{T}_1}^{\mathsf{T}_2} X(t)\phi_i(t)\,\mathrm{d}t}_{=X_i}\Big] \\
&\qquad + \sum_i \sum_{i'} \underbrace{\mathsf{E}[X_i X_{i'}]}_{=\lambda_i \mathbb{1}\{i=i'\}} \underbrace{\int_{\mathsf{T}_1}^{\mathsf{T}_2} \phi_i(t)\phi_{i'}(t)\,\mathrm{d}t}_{=\mathbb{1}\{i=i'\}} \qquad \text{(C.131)} \\
&\quad = \mathsf{E}\left[\int_{\mathsf{T}_1}^{\mathsf{T}_2} X^2(t)\,\mathrm{d}t\right] - 2\sum_i \mathsf{E}\left[X_i^2\right] + \sum_i \lambda_i \qquad \text{(C.132)} \\
&\quad = \mathsf{E}\left[\int_{\mathsf{T}_1}^{\mathsf{T}_2} X^2(t)\,\mathrm{d}t\right] - \sum_i \lambda_i. \qquad \text{(C.133)}
\end{aligned}$$

We know that the left side of (C.131) is zero if the eigenvectors of zero eigenvalue are included, but (C.133) shows that the equality holds without the eigenvectors of zero eigenvalue. Since $\mathsf{E}\left[X^2(t)\right] = \mathsf{K}_{XX}(t,t)$, this also verifies the useful identity

$$\int_{\mathsf{T}_1}^{\mathsf{T}_2} \mathsf{K}_{XX}(t,t)\,\mathrm{d}t = \sum_{i=1}^{\infty} \lambda_i. \tag{C.134}$$

These results have all assumed that $\mathsf{K}_{XX}(\cdot,\cdot)$ is continuous. We now look at two examples, the first of which helps us to understand what happens if $\mathsf{K}_{XX}(\cdot,\cdot)$ is not continuous, and the second of which helps us to understand a common engineering notion about degrees of freedom.

Example C.20 (Pathological Nonexistent Noise). Consider a Gaussian process $\{X(t)\}$ where for each $t \in \mathbb{R}$, $X(t) \sim \mathcal{N}(0,1)$. Assume also that

$$\mathsf{E}[X(t)X(\tau)] = 0, \quad \forall\, t \neq \tau. \tag{C.135}$$

Thus $\mathsf{K}_{XX}(t,\tau)$ is 1 for $t = \tau$ and 0 otherwise. Since $\mathsf{K}_{XX}(\cdot,\cdot)$ is not continuous, Theorem C.19 does not necessarily hold, but because $\mathsf{K}_{XX}(\cdot,\cdot)$ is zero almost everywhere, any orthonormal set of functions can be used as eigenvectors of eigenvalue 0. The trouble occurs when we try to define random variables

$X_i \triangleq \int_{T_1}^{T_2} X(t)\phi_i(t)\,\mathrm{d}t$. Since the sample functions of $\{X(t)\}$ are discontinuous everywhere, it is hard to make sense out of this integral. However, if we try to approximate this integral as

$$Y_\delta = \sum_{\ell=T_1/\delta}^{T_2/\delta} X(\ell\delta)\phi_i(\ell\delta)\delta, \tag{C.136}$$

then we see that

$$Y_\delta \sim \mathcal{N}\left(0, \delta^2 \sum_{\ell=T_1/\delta}^{T_2/\delta} \phi_i^2(\ell\delta)\right). \tag{C.137}$$

This variance goes to zero as $\delta \to 0$, so it is reasonable to take $X_i = 0$. This means that we cannot represent $X(\cdot) = \sum_i X_i\phi_i(\cdot)$. It also means that if we filter $X(\cdot)$, the filter output is best regarded as 0. Thus, in a sense, this kind of process cannot be observed. We can also view this process as white Gaussian noise in the limit as the power spectral density approaches 0.

The point of this example is to show how strange a stochastic process $\{X(t)\}$ is when $K_{XX}(t,\tau)$ is discontinuous at $t = \tau$. It is not hard to show that if a WSS process has an autocovariance function $K_{XX}(\cdot)$ that is continuous at $t = 0$, then it is continuous everywhere, so that in some sense the kind of phenomena in this example is typical of discontinuous covariance functions. ◊

Example C.21 (Prolate Spheroidal Wave Functions). We usually view the set of waveforms that are essentially timelimited to $[-T/2, T/2]$ and essentially bandlimited to $[0, W]$ as having about $2WT$ degrees of freedom if WT is large. For example, there are $2WT + 1$ orthonormal sine and cosine waves with frequency $f \le W$ in the Fourier series over $[-T/2, T/2]$. This notion of degrees of freedom is inherently imprecise since strictly timelimited functions cannot be strictly bandlimited and vice versa. Our objective in this example is to make this notion as close to precise as possible.

Suppose $\{X(t)\}$ is a stationary process with power spectral density

$$S_{XX}(f) = \begin{cases} 1 & f \le W, \\ 0 & \text{elsewhere.} \end{cases} \tag{C.138}$$

The corresponding autocovariance function is

$$K_{XX}(\tau) = \frac{\sin(2\pi W\tau)}{\pi\tau}, \quad \tau \in \mathbb{R}. \tag{C.139}$$

Consider truncating the sample functions of this process to $[-T/2, T/2]$. Before being truncated, these sample functions are bandlimited, but after truncation, they are no longer bandlimited. Let $\{\lambda_i, \phi_i(\cdot)\}$ be the eigenvalues and eigenvectors of the integral equation (C.114) for $K_{XX}(\cdot)$ given in (C.139) for

$-T/2 \le \tau \le T/2$. Each eigenvalue/eigenvector pair $\lambda_i, \phi_i(\cdot)$ has the property that when $\phi_i(\cdot)$ is passed through an ideal low pass filter and then truncated again to $[-T/2, T/2]$, the same function $\phi_i(\cdot)$ appears attenuated by λ_i (this is in fact what (C.114) says). With a little extra work, it can be seen that the output $\theta_i(\cdot)$ of this ideal low pass filter before truncation has energy λ_i and after truncation has energy λ_i^2.

The eigenvectors $\phi_i(\cdot)$ have an interesting extremal property (assuming that the eigenvalues are ordered in decreasing order). In particular, $\phi_i(\cdot)$ is the unit energy function over $[-T/2, T/2]$ with the largest energy in the frequency band $[0, W]$, subject to the constraint of being orthogonal to $\phi_j(\cdot)$ for each $j < i$. In other words, the first i eigenfunctions are the orthogonal timelimited functions with the largest energies in the frequency band $[0, W]$.

The functions $\theta_i(\cdot)$ defined above are called *prolate spheroidal wave functions* and arise in many areas of mathematics, physics, and engineering. They are orthogonal, and also their truncated versions to $[-T/2, T/2]$ are orthogonal.

Finally, when WT is large, it turns out that λ_i is very close to 1 for $i \ll 2WT$, λ_i is very close to 0 for $i \gg 2WT$, and the transition region has a width proportional to $\ln(4\pi WT)$. In other words, as WT gets large, there are about $2WT$ orthogonal waveforms that are timelimited to $[-T/2, T/2]$ and approximately bandlimited to $[0, W]$. ◊

Bibliography

[AC88] Paul H. Algoet and Thomas M. Cover, "A sandwich proof of the shannon–mcmillan–breiman theorem," *The Annals of Probability*, vol. 16, no. 2, pp. 899–909, April 1988.

[Arı08] Erdal Arıkan, "A performance comparison of polar codes and Reed–Muller codes," *IEEE Communications Letters*, vol. 12, no. 6, pp. 447–449, June 2008.

[Arı09] Erdal Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

[Arı10] Erdal Arıkan, "Source polarization," in *Proceedings IEEE International Symposium on Information Theory (ISIT)*, Austin, TX, USA, June 13–18, 2010, pp. 899–901.

[Arı19a] Erdal Arıkan, "From sequential decoding to channel polarization and back again," September 2019, arXiv:1908.09594 [cs.IT]. Available: https://arxiv.org/abs/1908.09594

[Arı19b] Erdal Arıkan, "Shannon Lecture: From sequential decoding to channel polarization and back again," July 10, 2019, ISIT, Paris, France. Available: https://youtu.be/1yiSUPj42aQ

[AT09] Erdal Arıkan and İ. Emre Telatar, "On the rate of channel polarization," April 6, 2009, arXiv:0807.3806v3 [cs.IT]. Available: https://arxiv.org/abs/0807.3806v3

[BGT93] Claude Berrou, Alain Glavieux, and Punya Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proceedings IEEE International Conference on Communications (ICC)*, Geneva, Switzerland, May 23–26, 1993, pp. 1064–1070.

[BT02] Dimitri P. Bertsekas and John N. Tsitsiklis, *Introduction to Probability*. Belmont, MA, USA: Athena Scientific, 2002.

[CA05a] Po-Ning Chen and Fady Alajaji, "Lecture notes on information theory," vol. 1, Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan, and Department of Mathematics & Statistics, Queen's University, Kingston, Canada, August 2005.

[CA05b] Po-Ning Chen and Fady Alajaji, "Lecture Notes on Information Theory," vol. 2, Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan, and Department of Mathematics & Statistics, Queen's University, Kingston, Canada, August 2005.

[CT06] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, 2nd ed. New York, NY, USA: John Wiley & Sons, 2006.

[DH76] Whitfield Diffie and Martin E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, November 1976.

[DM98] Matthew C. Davey and David J. C. MacKay, "Low-density parity check codes over $GF(q)$," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, June 1998.

[Eli75] Peter Elias, "Universal codeword sets and representations of the integers," *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 194–203, March 1975.

[Fan49] Robert M. Fano, "The transmission of information," Research Laboratory of Electronics, Massachusetts Institute of Technology (MIT), Technical Report No. 65, March 17, 1949.

[FB14] Ubaid U. Fayyaz and John R. Barry, "Low-complexity soft-output decoding of polar codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 958–966, May 2014.

[Fei54] Amiel Feinstein, "A new basic theorem of information theory," *IRE Transactions on Information Theory*, vol. 4, no. 4, pp. 2–22, September 1954.

[Gal62] Robert G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA, USA: MIT Press, 1962.

[Gal68] Robert G. Gallager, *Information Theory and Reliable Communication*. New York, NY, USA: John Wiley & Sons, 1968.

[Har28] Ralph Hartley, "Transmission of information," *Bell System Technical Journal*, vol. 7, no. 3, pp. 535–563, July 1928.

[Huf52] David A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, September 1952.

[HY10] Siu-Wai Ho and Raymond W. Yeung, "The interplay between entropy and variational distance," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 5906–5929, December 2010.

[Khi56] Aleksandr Y. Khinchin, "On the fundamental theorems of information theory (Russian)," *Uspekhi Matematicheskikh Nauk XI*, vol. 1, pp. 17–75, 1956.

[Khi57] Aleksandr Y. Khinchin, *Mathematical Foundations of Information Theory*. New York, NY, USA: Dover Publications, 1957.

[Knu89] Donald E. Knuth, "Typesetting concrete mathematics," *TUGboat*, vol. 10, no. 1, pp. 31–36, April 1989. Available: https://www.tug.org/TUGboat/Contents/contents10-1.html

[Lap17] Amos Lapidoth, *A Foundation in Digital Communication*, 2nd ed. Cambridge, UK: Cambridge University Press, February 2017.

[Mas96] James L. Massey, *Applied Digital Information Theory I and II,* Lecture notes, Signal and Information Processing Laboratory, ETH Zürich, 1995/1996. Available: https://www.isiweb.ee.ethz.ch/archive/massey_scr/

[MC12] Stefan M. Moser and Po-Ning Chen, *A Student's Guide to Coding and Information Theory*. Cambridge, UK: Cambridge University Press, January 2012, ISBN: 978–1–107–01583–8 (hardcover) and 978–1–107–60196–3 (paperback).

[Mer78] Ralph Merkle, "Secure communication over insecure channels," *Commmunications of the Association for Computing Machinery (ACM)*, pp. 294–299, April 1978.

[MN96] David J. C. MacKay and Radford M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, August 1996, reprinted with printing errors corrected in vol. 33, no. 6, pp. 457–458.

[Mos05] Stefan M. Moser, *Duality-Based Bounds on Channel Capacity*, ser. ETH Series in Information Theory and its Applications. Konstanz, Germany: Hartung-Gorre Verlag, January 2005, ISBN: 3–89649–956–4, vol. 1, edited by Amos Lapidoth. ISBN 3–89649–956–4. Available: https://moser-isi.ethz.ch/publications.html

[Mos22] Stefan M. Moser, *Advanced Topics in Information Theory (Lecture Notes)*, 5th ed. Signal and Information Processing Laboratory, ETH Zürich, Switzerland, and Institute of Communications Engineering, National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan, 2022. Available: https://moser-isi.ethz.ch/scripts.html

[Pop18] "2018 World Population Data Sheet," Population Reference Bureau, 1875 Connecticut Avenue, Washington DC, USA, Tech. Rep., August 2018. Available: https://www.prb.org/

[PPV10] Yury Polyanskiy, H. Vincent Poor, and Sergio Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[RSN90] Frigyes Riesz and Béla Sz.-Nagy, *Functional Analysis*. New York, NY, USA: Dover Publications, 1990.

[Şaş11a] Eren Şaşoğlu, "Polarization and polar codes," *Foundations and Trends® in Communications and Information Theory*, vol. 8, no. 4, pp. 259–381, 2011.

[Şaş11b] Eren Şaşoğlu, "Polarization in the presence of memory," in *Proceedings IEEE International Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, July 31 – August 5, 2011, pp. 189–193.

[Say99] Jossy Sayir, "On coding by probability transformation," Ph.D. dissertation, ETH Zürich, 1999, Diss. ETH No. 13099. Available: https://doi.org/10.3929/ethz-a-002093628

[Sha48] Claude E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and October 1948.

[Sha49] Claude E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, October 1949.

[ŞTA09] Eren Şaşoğlu, İ. Emre Telatar, and Erdal Arıkan, "Polarization for arbitrary discrete memoryless channels," in *Proceedings IEEE Information Theory Workshop (ITW)*, Taormina, Sicily, October 11–16, 2009, pp. 144–148.

[Str09] Gilbert Strang, *Introduction to Linear Algebra*, 4th ed. Wellesley, MA, USA: Wellesley-Cambridge Press, 2009.

[TA12] İ. Emre Telatar and Emmanuel A. Abbe, "Polar codes for the m-user multiple access channel," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5437–5448, August 2012.

[Tal17] Ido Tal, "A simple proof of fast polarization," *IEEE Transactions on Information Theory*, vol. 63, no. 12, pp. 7617–7619, December 2017.

[Tun67] Brian P. Tunstall, "Synthesis of noiseless compression codes," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, September 1967.

[TV13] Ido Tal and Alexander Vardy, "How to construct polar codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6562–6582, October 2013.

[TV15] Ido Tal and Alexander Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

List of Figures

List of Tables

Index

Italic entries are to names.