7

Approaching the Shannon limit by turbo coding

7.1 Information Transmission Theorem

The reliable transmission of information-bearing signals over a noisy communication channel is at the heart of what we call communication. Information theory, founded by Claude E. Shannon in 1948 [Sha48], provides a mathematical framework for the theory of communication. It describes the fundamental limits to how efficiently one can encode information and still be able to recover it with negligible loss.

At its inception, the main role of information theory was to provide the engineering and scientific communities with a mathematical framework for the theory of communication by establishing the fundamental limits on the performance of various communication systems. Its birth was initiated with the publication of the works of Claude E. Shannon, who stated that it is possible to send information-bearing signals at a fixed code rate through a noisy communication channel with an arbitrarily small error probability as long as the code rate is below a certain fixed quantity that depends on the channel characteristics [Sha48]; he "baptized" this quantity with the name of *channel capacity* (see the discussion in Chapter 6). He further proclaimed that random sources – such as speech, music, or image signals – possess an irreducible complexity beyond which they cannot be compressed distortion-free. He called this complexity the *source entropy* (see the discussion in Chapter 5). He went on to assert that if a source has an entropy that is less than the capacity of a communication channel, then asymptotically error-free transmission of the source over the channel can be achieved. This result is usually referred to as the *Information Transmission Theorem* or the *Joint Source–Channel Coding Theorem*.

Theorem 7.1 (Information Transmission Theorem)
Consider the transmission of a source $\mathbf{U}^k = (U_1, U_2, \ldots, U_k)$ *through a channel with input* $\mathbf{X}^n = (X_1, X_2, \ldots, X_n)$ *and output* $\mathbf{Y}^n = (Y_1, Y_2, \ldots, Y_n)$ *as shown in Figure 7.1. Assume that both the source sequence* $U_1, U_2, \ldots, U_k$ *and the noise sequence* $N_1, N_2, \ldots, N_n$ *are independent and identically distributed. Then, subject to a fixed code rate* $\mathrm{R} = k/n$*, there exists a sequence of encoder–decoder pairs such that the decoding error, i.e.* $\Pr\left[\hat{\mathbf{U}}^k \neq \mathbf{U}^k\right]$*, can be made arbitrarily small (i.e. arbitrarily close to zero) by taking n sufficiently large if*

$$\frac{1}{\mathrm{T_s}} H(U) \text{ bits/second} < \frac{1}{\mathrm{T_c}} \max_{P_X} I(X;Y) \text{ bits/second}, \tag{7.1}$$

where the base-2 logarithm is adopted in the calculation of entropy and mutual information (so they are in units of bits), and $\mathrm{T_s}$ *and* $\mathrm{T_c}$ *are, respectively, the time (in units of second) to generate one source symbol* U_ℓ *and the time to transmit one channel symbol* X_ℓ*. On the other hand, if*

$$\frac{1}{\mathrm{T_s}} H(U) \text{ bits/second} > \frac{1}{\mathrm{T_c}} \max_{P_X} I(X;Y) \text{ bits/second}, \tag{7.2}$$

then $\Pr\left[\hat{\mathbf{U}}^k \neq \mathbf{U}^k\right]$ *has a universal positive lower bound for all coding schemes of any length k; hence, the error cannot be made arbitrarily small.*

Recall from Definition 6.20 that the rate of a code is defined as

$$\mathrm{R} = \frac{\log_2 \mathrm{M}}{n} \text{ bits per transmission.} \tag{7.3}$$

From Figure 7.1 we now see that here we have[1] $\mathrm{M} = 2^k$, i.e.

$$\mathrm{R} = \frac{\log_2(2^k)}{n} = \frac{k}{n}. \tag{7.4}$$

On the other hand, it is also apparent from Figure 7.1 that, due to timing reasons, we must have

$$k\mathrm{T_s} = n\mathrm{T_c}. \tag{7.5}$$

[1] The number of codewords M is given by the number of different source sequences $\mathbf{U}^k$ of length k. For simplicity we assume here that the source is binary, i.e. $|\mathcal{U}| = 2$. In general we have $\mathrm{M} = |\mathcal{U}|^k$ and $\mathrm{R} = (k/n)\log_2|\mathcal{U}|$.

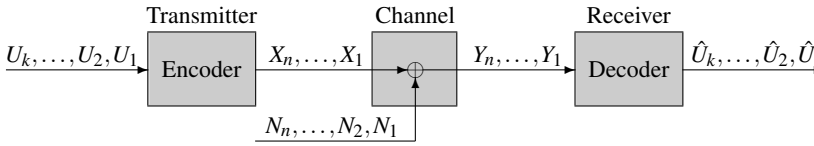

Figure 7.1 Noisy information transmission system.

Combined with (7.4) we thus see that the code rate can also be represented as

$$\mathsf{R} = \frac{\mathsf{T_c}}{\mathsf{T_s}}. \tag{7.6}$$

In Sections 7.3 and 7.4, we will further explore the two situations corresponding to whether condition (7.1) or (7.2) is valid.

7.2 The Gaussian channel

Figure 7.1 considers the situation of a binary source $U_1, U_2, \ldots, U_k$ being transmitted through a noisy channel that is characterized by

$$Y_\ell = X_\ell + N_\ell, \quad \ell = 1, 2, \ldots, n, \tag{7.7}$$

where $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_n$ are, respectively, the channel input and channel output sequences, and $N_1, N_2, \ldots, N_n$ is the noise sequence.

Assume that U_ℓ is either 0 or 1, and $\Pr[U_\ell = 0] = \Pr[U_\ell = 1] = 1/2$. Also assume that $U_1, U_2, \ldots, U_k$ are all independent.[2] Hence, its average entropy is equal to

$$\frac{1}{k} H(\mathbf{U}^k) = \frac{1}{k} \sum_{\mathbf{u}^k \in \{0,1\}^k} \Pr\left[\mathbf{U}^k = \mathbf{u}^k\right] \log_2 \left(\frac{1}{\Pr[\mathbf{U}^k = \mathbf{u}^k]} \right) \text{ bits/source symbol} \tag{7.8}$$

$$= \frac{1}{k} \sum_{\mathbf{u}^k \in \{0,1\}^k} 2^{-k} \log_2 \left(\frac{1}{2^{-k}} \right) \text{ bits/source symbol} \tag{7.9}$$

$$= 1 \text{ bit/source symbol}, \tag{7.10}$$

where we abbreviate $(U_1, U_2, \ldots, U_k)$ as $\mathbf{U}^k$.

In a practical communication system, there usually exists a certain constraint

[2] This assumption is well justified in practice: were U_ℓ not uniform and independent, then any good data compression scheme could make them so. For more details on this, we refer to Appendix 7.7.

E on the transmission power (for example, in units of joule per transmission). This power constraint can be mathematically modeled as

$$\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n} \leq \mathsf{E} \quad \text{for all } n. \tag{7.11}$$

When being transformed into an equivalent statistical constraint, one can replace (7.11) by

$$\mathsf{E}\left[\frac{X_1^2 + X_2^2 + \cdots + X_n^2}{n}\right] = \mathsf{E}\left[X_1^2\right] \leq \mathsf{E}, \tag{7.12}$$

where $\mathsf{E}[\cdot]$ denotes the expected value of the target random variable, and equality holds because we assume $\mathsf{E}\left[X_1^2\right] = \mathsf{E}\left[X_2^2\right] = \cdots = \mathsf{E}\left[X_n^2\right]$, i.e. the channel encoder is expected to assign, on average, an equal transmission power to each channel input. Note that the channel inputs are in general strongly dependent so as to combat interference; what we assume here is that they have, on average, equal marginal power. We further assume that the noise samples $N_1, N_2, \ldots, N_n$ are independent in statistics and that the probability density function[3] of each N_ℓ is given by

$$f_{N_\ell}(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{t^2}{2\sigma^2}\right), \quad t \in \Re. \tag{7.13}$$

This is usually termed the *zero-mean Gaussian distribution*, and the corresponding channel (7.7) is therefore called the *Gaussian channel*.

7.3 Transmission at a rate below capacity

It can be shown that the channel capacity (i.e. the largest code rate below which arbitrarily small error probability can be obtained) of a Gaussian channel as defined in Section 7.2 is

$$\mathsf{C}(\mathsf{E}) = \max_{f_X : \mathsf{E}\left[X^2\right] \leq \mathsf{E}} I(X;Y) \tag{7.14}$$

$$= \frac{1}{2}\log_2\left(1 + \frac{\mathsf{E}}{\sigma^2}\right) \text{ bits/channel use}, \tag{7.15}$$

[3] A *probability density function* is the *density* function for *probability*. Similar to the fact that the density of a material describes its mass per unit volume, the probability density function gives the probability of occurrence per unit point. Hence, integration of the material density over a volume leads to the mass confined within it, and integration of the probability density function over a range tells us the probability that one will observe a value in this range. The Gaussian density function in (7.13) is named after the famous mathematician Carl Friedrich Gauss, who used it to analyze astronomical data. Since it is quite commonly seen in practice, it is sometimes named the *normal* distribution.

where the details can be found in [CT06, Eq. (9.16)]. Recall from the Information Transmission Theorem (Theorem 7.1) that if the source entropy (namely, 1 bit/source symbol) is less than the capacity of a communication channel (i.e. $(1/2)\log_2(1+\mathsf{E}/\sigma^2)$ bits per channel use), then reliable transmission becomes feasible. Hence, in equation form, we can present the condition for reliable transmission as follows:

$$\frac{1}{\mathsf{T}_\mathrm{s}} \text{ bits/second} < \frac{1}{2\mathsf{T}_\mathrm{c}} \log_2\left(1+\frac{\mathsf{E}}{\sigma^2}\right) \text{ bits/second.} \tag{7.16}$$

Note that when comparing we have to represent the average source entropy and channel capacity by the same units (here, bits/second). This is the reason why we have introduced T_s and T_c.

7.4 Transmission at a rate above capacity

Now the question is what if

$$\frac{1}{\mathsf{T}_\mathrm{s}} \text{ bits/second} > \frac{1}{2\mathsf{T}_\mathrm{c}} \log_2\left(1+\frac{\mathsf{E}}{\sigma^2}\right) \text{ bits/second.} \tag{7.17}$$

In such a case, we know from the Information Transmission Theorem (Theorem 7.1) that an arbitrarily small error probability cannot be achieved. However, can we identify the smallest error rate that can be possibly obtained?

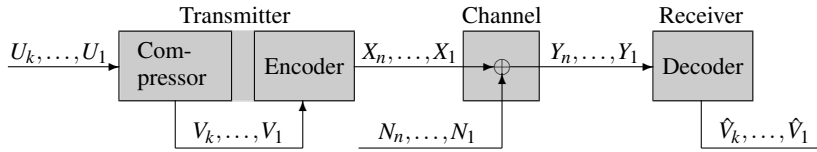

Figure 7.2 Noisy information transmission system with incorporated compressor.

A straightforward system design is to map each source sequence $u_1,\ldots,u_k$ into a compressed binary sequence $\mathbf{v}^k \triangleq g(\mathbf{u}^k)$ for transmission (see Figure 7.2), where the compressor function $g(\cdot)$ is chosen such that the resulting average entropy is less than the channel capacity, i.e.

$$\frac{1}{\mathsf{T}_\mathrm{s}}\frac{1}{k}H(\mathbf{V}^k) \text{ bits/second} < \frac{1}{2\mathsf{T}_\mathrm{c}} \log_2\left(1+\frac{\mathsf{E}}{\sigma^2}\right) \text{ bits/second.} \tag{7.18}$$

Then, from Theorem 7.1 we know that $V_1, V_2, \ldots, V_k$ can be transmitted through the additive Gaussian noise channel with arbitrarily small error. This is to say,

there exists a transmission scheme such that the decision at the channel output $\hat{v}_1, \hat{v}_2, \hat{v}_3, \ldots$ is the same as the compressed channel input $v_1, v_2, v_3, \ldots$ with probability arbitrarily close to unity.[4] As a consequence, the error that is introduced in this straightforward system design occurs only at those instances (i.e. ℓs) where the compression is not reversible, i.e. u_ℓ cannot be recovered from $\mathbf{v}^k$.

In the following, we will determine a minimum error probability that the straightforward system in Figure 7.2 cannot beat even if we optimize over all possible compressor designs subject to the average entropy of the compressor output being smaller than the channel capacity. Before we come to this analysis, we give several examples of how a compressor works and what is its error.

Example 7.2 For example, let $g(\cdot)$ map $u_1, u_2, u_3, \ldots$ into $v_1, v_2, v_3, \ldots$ in a fashion that

$$(v_1, v_2, v_3, v_4, \ldots) = g(u_1, u_2, u_3, u_4, \ldots) = (u_1, u_1, u_3, u_3, \ldots), \tag{7.19}$$

i.e. $v_{2\ell-1} = v_{2\ell} = u_{2\ell-1}$ (see Table 7.1). Since $V_{2\ell-1} = V_{2\ell}$ for every ℓ, no new information is provided by $V_{2\ell}$ given $V_{2\ell-1}$. The average entropy of $V_1, V_2, \ldots, V_k$, with $k = 2m$ even, is then given by

$$\begin{aligned}
\frac{1}{k} H(\mathbf{V}^k) &= \frac{1}{2m} H(V_1, V_2, \ldots, V_{2m}) && (7.20)\\
&= \frac{1}{2m} H(V_1, V_3, \ldots, V_{2m-1}) && (7.21)\\
&= \frac{1}{2m} H(U_1, U_3, U_5, \ldots, U_{2m-1}) && (7.22)\\
&= \frac{1}{2} \text{ bits/source symbol}, && (7.23)
\end{aligned}$$

i.e.

$$\frac{1}{\mathsf{T}_\mathrm{s}} \frac{1}{k} H(\mathbf{V}^k) = \frac{1}{2\mathsf{T}_\mathrm{s}} \text{ bits/second}. \tag{7.24}$$

[4] What Shannon targeted in his theorem is the *block error rate*, not the *bit error rate*. Hence, his theorem actually concludes that $\Pr\left[(V_1, V_2, \ldots, V_k) = (\hat{V}_1, \hat{V}_2, \ldots, \hat{V}_k)\right] \simeq 1$ when k is sufficiently large since

$$\frac{1}{\mathsf{T}_\mathrm{s}} \frac{1}{k} H(\mathbf{V}^k) \text{ bits/second}$$

is less than

$$\frac{1}{2\mathsf{T}_\mathrm{c}} \log_2 \left(1 + \frac{\mathsf{E}}{\sigma^2}\right) \text{ bits/second}.$$

This is a very strong statement because, for example, $v_\ell = \hat{v}_\ell$ for $1 \le \ell \le k-1$ and $v_k \neq \hat{v}_k$ will be counted as one block error even though there is only one bit error among these k bits. Note that $\Pr\left[(V_1, V_2, \ldots, V_k) = (\hat{V}_1, \hat{V}_2, \ldots, \hat{V}_k)\right] \simeq 1$ surely implies that $\Pr\left[V_\ell = \hat{V}_\ell\right] \simeq 1$ for most ℓ, but not vice versa.

Table 7.1 *The mapping $g(\cdot)$ from $\mathbf{u}^k$ to $\mathbf{v}^k$ defined in (7.19) in Example 7.2*

$u_1, u_2, u_3, u_4, \ldots$	$v_1, v_2, v_3, v_4, \ldots$
0000…	0000…
0001…	0000…
0010…	0011…
0011…	0011…
0100…	0000…
0101…	0000…
0110…	0011…
0111…	0011…
1000…	1100…
1001…	1100…
1010…	1111…
1011…	1111…
1000…	1100…
1101…	1100…
1110…	1111…
1111…	1111…

Under the premise that

$$\frac{1}{2\mathrm{T_s}} \text{ bits/second} < \frac{1}{2\mathrm{T_c}} \log_2\left(1 + \frac{\mathsf{E}}{\sigma^2}\right) \text{ bits/second}, \tag{7.25}$$

the average "bit" error rate of the system is given by

$$\begin{aligned}
&\frac{1}{k}\Big(\Pr[U_1 \neq V_1] + \Pr[U_2 \neq V_2] + \Pr[U_3 \neq V_3] + \Pr[U_4 \neq V_4] \\
&\quad + \cdots + \Pr[U_k \neq V_k]\Big) \\
&= \frac{1}{2m}\Big(\Pr[U_1 \neq V_1] + \Pr[U_2 \neq V_2] + \Pr[U_3 \neq V_3] + \Pr[U_4 \neq V_4] \\
&\qquad + \cdots + \Pr[U_{2m} \neq V_{2m}]\Big) && (7.26)\\
&= \frac{1}{2m}\Big(0 + \Pr[U_2 \neq V_2] + 0 + \Pr[U_4 \neq V_4] + \cdots + \Pr[U_{2m} \neq V_{2m}]\Big) && (7.27)\\
&= \frac{1}{2m}\Big(\Pr[U_2 \neq U_1] + \Pr[U_4 \neq U_3] + \cdots + \Pr[U_{2m} \neq U_{2m-1}]\Big) && (7.28)\\
&= \frac{1}{2}\Pr[U_2 \neq U_1] && (7.29)\\
&= \frac{1}{2}\Big(\Pr[(U_1, U_2) = (01)] + \Pr[(U_1, U_2) = (10)]\Big) && (7.30)
\end{aligned}$$

$$= \frac{1}{2}\left(\frac{1}{4}+\frac{1}{4}\right) = \frac{1}{4}. \tag{7.31}$$

In fact, the error rate of $1/4$ can be obtained directly without computation by observing that all the odd-indexed bits $U_1, U_3, U_5, \ldots$ can be correctly recovered from $V_1, V_3, V_5, \ldots$; however, the sequence $\mathbf{V}^k$ provides no information for all the even-indexed bits $U_2, U_2, U_6, \ldots$ Thus, we can infer that a zero error rate for odd-indexed bits and a $1/2$ error rate based on a pure guess for even-indexed bits combine to a $1/4$ error rate. ◊

Example 7.2 provides a compressor with average bit error rate (BER) of $1/4$ between input $U_1, U_2, \ldots, U_k$ and output $V_1, V_2, \ldots, V_k$ subject to its average output entropy $1/2\mathsf{T}_\mathrm{s}$ bits/second being smaller than the channel capacity $\mathsf{C}(\mathsf{E}) = (1/2\mathsf{T}_\mathrm{c})\log_2(1+\mathsf{E}/\sigma^2)$. However, dropping all even-indexed bits may not be a good compressor design because it is possible that half of the bits in $V_1, V_2, \ldots, V_k$ are different from $U_1, U_2, \ldots, U_k$; i.e., in the worst case, the difference between compressor input $U_1, U_2, \ldots, U_k$ and compressor output $V_1, V_2, \ldots, V_k$ will result in a large distortion of $k/2$ bits.

In Section 3.3.3 we saw that the $(7,4)$ Hamming code is a perfect packing of radius-one spheres in the 7-dimensional binary space. Using this property, we can provide in Example 7.3 an alternative compressor design such that the input and output are different in at most 1 bit with a (smaller) average BER of $1/8$ and a (slightly larger) average output entropy of $4/7\mathsf{T}_\mathrm{s}$ bits/second.

Example 7.3 A compressor g is defined based on the $(7,4)$ Hamming codewords listed in Table 3.2 as follows: $g(\mathbf{u}^7) = \mathbf{v}^7$ if $\mathbf{v}^7$ is a $(7,4)$ Hamming codeword and $\mathbf{u}^7$ is at Hamming distance at most one from $\mathbf{v}^7$. The perfect packing of the 16 nonoverlapping radius-one spheres centered at the codewords for the $(7,4)$ Hamming code guarantees the existence and uniqueness of such a $\mathbf{v}^7$; hence, the compressor function mapping is well defined.

The probability of each $(7,4)$ Hamming codeword appearing at the output is $8 \cdot 2^{-7} = 2^{-4}$ (since there are eight $\mathbf{u}^7$ mapped to the same $\mathbf{v}^7$). Hence,

$$\frac{1}{7}H(\mathbf{V}^7) = \frac{1}{7}\sum_{\mathbf{v}^7 \in \mathscr{C}_\mathrm{H}} 2^{-4}\log_2\left(\frac{1}{2^{-4}}\right) \tag{7.32}$$

$$= \frac{4}{7} \text{ bits/source symbol}, \tag{7.33}$$

where $\mathscr{C}_\mathrm{H}$ denotes the set of the 16 codewords of the $(7,4)$ Hamming code. Hence,

$$\frac{1}{7\mathsf{T}_\mathrm{s}}H(\mathbf{V}^7) = \frac{4}{7\mathsf{T}_\mathrm{s}} \text{ bits/second}. \tag{7.34}$$

Next, we note that $\Pr[U_1 \neq V_1] = 1/8$ because only one of the eight $\mathbf{u}^7$ that are mapped to the same $\mathbf{v}^7$ results in a different first bit. Similarly, we can obtain

$$\Pr[U_2 \neq V_2] = \Pr[U_3 \neq V_3] = \cdots = \Pr[U_7 \neq V_7] = \frac{1}{8}. \tag{7.35}$$

Hence, the average BER is given by

$$\text{BER} = \frac{1}{7}\big(\Pr[U_1 \neq V_1] + \Pr[U_2 \neq V_2] + \cdots \Pr[U_7 \neq V_7]\big) = \frac{1}{8}. \tag{7.36}$$

This example again shows that data compression can be regarded as the opposite operation of error correction coding, where the former removes the redundancy (or even some information such as in this example) while the latter adds controlled redundancy to combat the channel noise effect. ◊

Exercise 7.4 *Design a compressor mapping by reversing the roles of encoder and decoder of the three-times repetition code. Prove that the average BER is 1/4 and the average output entropy equals 1/3 bits per source symbol.* ◊

Readers may infer that one needs to know the best compressor design, which minimizes the BER subject to the average output entropy less than $\mathsf{C}(\mathsf{E})$, in order to know what will be the minimum BER attainable for a given channel (or more specifically for a given $\mathsf{C}(\mathsf{E})$). Ingeniously, Shannon identifies this minimum BER without specifying how it can be achieved. We will next describe his idea of a converse proof that shows that the minimum BER cannot be smaller than some quantity, but that does not specify $g(\cdot)$.

We can conceptually treat the compressor system as a channel with input $U_1, U_2, \ldots, U_k$ and output $V_1, V_2, \ldots, V_k$. Then, by

$$H(\mathbf{V}^k|\mathbf{U}^k) = H(g(\mathbf{U}^k)|\mathbf{U}^k) = 0, \tag{7.37}$$

we derive from (6.85) that

$$I(\mathbf{U}^k;\mathbf{V}^k) = H(\mathbf{U}^k) - H(\mathbf{U}^k|\mathbf{V}^k) = H(\mathbf{V}^k) - H(\mathbf{V}^k|\mathbf{U}^k) = H(\mathbf{V}^k). \tag{7.38}$$

This implies that the average entropy of the compressor output is equal to

$$\frac{1}{k}H(\mathbf{V}^k) = \frac{1}{k}H(\mathbf{U}^k) - \frac{1}{k}H(\mathbf{U}^k|\mathbf{V}^k) = 1 - \frac{1}{k}H(\mathbf{U}^k|\mathbf{V}^k) \text{ bits}. \tag{7.39}$$

By the chain rule for entropy,[5]

$$\frac{1}{k}H(\mathbf{V}^k) = 1 - \frac{1}{k}H(\mathbf{U}^k|\mathbf{V}^k) \tag{7.40}$$

[5] The chain rule for entropy is

$$H(\mathbf{U}^k) = H(U_1) + H(U_2|U_1) + H(U_3|U_1,U_2) + \cdots + H(U_k|U_1,\ldots,U_{k-1}).$$

This is a generalized form of Proposition 6.4 and can be proven similarly.

$$= 1 - \frac{1}{k}\big(H(U_1|\mathbf{V}^k) + H(U_2|U_1,\mathbf{V}^k) + H(U_3|U_1,U_2,\mathbf{V}^k) + \cdots + H(U_k|U_1,\ldots,U_{k-1},\mathbf{V}^k)\big) \tag{7.41}$$

$$\geq 1 - \frac{1}{k}\big(H(U_1|V_1) + H(U_2|V_2) + H(U_3|V_3) + \cdots + H(U_k|V_k)\big), \tag{7.42}$$

where (7.42) holds because additional information always helps to decrease entropy; i.e., $H(U_\ell|U_1,\ldots,U_{\ell-1},V_1,\ldots,V_k) \leq H(U_\ell|V_\ell)$ since the former has additional information $(U_1,\ldots,U_{\ell-1},V_1,\ldots,V_{\ell-1},V_{\ell+1},\ldots,V_k)$ (see Corollary 6.10).

We proceed with the derivation by pointing out that

$$H(U_\ell|V_\ell) = \Pr[V_\ell = 0]\,H(U_\ell|V_\ell = 0) + \Pr[V_\ell = 1]\,H(U_\ell|V_\ell = 1) \tag{7.43}$$

$$= \Pr[V_\ell = 0]\,H_{\mathrm{b}}\big(\Pr[U_\ell = 1 \,|\, V_\ell = 0]\big) + \Pr[V_\ell = 1]\,H_{\mathrm{b}}\big(\Pr[U_\ell = 0 \,|\, V_\ell = 1]\big) \tag{7.44}$$

$$\leq H_{\mathrm{b}}\Big(\Pr[V_\ell = 0]\Pr[U_\ell = 1 \,|\, V_\ell = 0] + \Pr[V_\ell = 1]\Pr[U_\ell = 0 \,|\, V_\ell = 1]\Big) \tag{7.45}$$

$$= H_{\mathrm{b}}(\mathrm{BER}_\ell), \tag{7.46}$$

where $\mathrm{BER}_\ell \triangleq \Pr[U_\ell \neq V_\ell]$;

$$H_{\mathrm{b}}(p) \triangleq p\log_2\frac{1}{p} + (1-p)\log_2\frac{1}{1-p}, \quad \text{for } 0 \leq p \leq 1, \tag{7.47}$$

is the so-called *binary entropy function* (see Section 5.2.2); and (7.45) follows from the concavity[6] of the function $H_{\mathrm{b}}(\cdot)$. We then obtain the final lower bound of the output average entropy:

$$\frac{1}{k}H(\mathbf{V}^k) \geq 1 - \frac{1}{k}\big(H_{\mathrm{b}}(\mathrm{BER}_1) + H_{\mathrm{b}}(\mathrm{BER}_2) + \cdots + H_{\mathrm{b}}(\mathrm{BER}_k)\big) \tag{7.48}$$

$$= 1 - \frac{1}{k}\sum_{\ell=1}^{k} H_{\mathrm{b}}(\mathrm{BER}_\ell) \tag{7.49}$$

$$\geq 1 - H_{\mathrm{b}}\left(\frac{1}{k}\sum_{\ell=1}^{k}\mathrm{BER}_\ell\right) \tag{7.50}$$

$$= 1 - H_{\mathrm{b}}(\mathrm{BER}). \tag{7.51}$$

Here, (7.50) follows again from concavity.

In conclusion, the Information Transmission Theorem (Theorem 7.1) identifies the achievable *bit error rate (BER)* for the target additive Gaussian noise

[6] For a definition of concavity see Appendix 7.8.

channel as follows:

$$\frac{1}{\mathsf{T}_\mathrm{s}}\left(1-H_\mathrm{b}(\mathrm{BER})\right) \le \frac{1}{\mathsf{T}_\mathrm{s}}\frac{1}{k}H(\mathbf{V}^k) < \frac{1}{2\mathsf{T}_\mathrm{c}}\log_2\left(1+\frac{\mathsf{E}}{\sigma^2}\right), \tag{7.52}$$

where the first inequality follows from (7.51) and the second follows from our assumption (7.18). In usual communication terminologies, people denote $\mathsf{R} = \mathsf{T}_\mathrm{c}/\mathsf{T}_\mathrm{s} = k/n$ (information bit carried per channel use) as the *channel code rate*; $\mathsf{N}_0 \triangleq 2\sigma^2$ (joule) as the *noise energy level*; $\mathsf{E}_\mathrm{b} \triangleq \mathsf{E}\mathsf{T}_\mathrm{s}/\mathsf{T}_\mathrm{c}$ (joule) as the *equivalent transmitted energy per information bit*; and $\gamma_\mathrm{b} \triangleq \mathsf{E}_\mathrm{b}/\mathsf{N}_0$ as the *signal-to-noise power ratio per information bit*. This transforms the above inequality to

$$H_\mathrm{b}(\mathrm{BER}) > 1 - \frac{1}{2\mathsf{R}}\log_2\left(1+2\mathsf{R}\gamma_\mathrm{b}\right). \tag{7.53}$$

Equation (7.53) clearly indicates that the BER cannot be made smaller than

$$H_\mathrm{b}^{-1}\left(1-\frac{1}{2\mathsf{R}}\log_2\left(1+2\mathsf{R}\gamma_\mathrm{b}\right)\right), \tag{7.54}$$

where $H_\mathrm{b}^{-1}(\cdot)$ is the inverse function of the binary entropy function $H_\mathrm{b}(\xi)$ (see Section 5.2.2) for $\xi \in [0, 1/2]$. Shannon also proved the (asymptotic) achievability of this lower bound. Hence, (7.53) provides the exact margin on what we can do and what we cannot do when the amount of information to be transmitted is above the capacity.

We plot the curves corresponding to $\mathsf{R} = 1/2$ and $\mathsf{R} = 1/3$ in Figure 7.3. The figure indicates that there exists a rate-$1/2$ system design that can achieve $\mathrm{BER} = 10^{-5}$ at $\gamma_{\mathrm{b,dB}} \triangleq 10\log_{10}(\mathsf{E}_\mathrm{b}/\mathsf{N}_0)$ close to 0 dB, i.e. for $\mathsf{E}_\mathrm{b} \simeq \mathsf{N}_0$. On the other hand, no system with a rate $\mathsf{R} = 1/2$ 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. Information theorists therefore call this threshold the *Shannon limit*.

For decades (ever since Shannon ingeniously drew such a sacred line in 1948 simply by analysis), researchers have tried to find a good design that can achieve the Shannon limit. Over the years, the gap between the real transmission scheme and this theoretical limit has been gradually closed. For example, a concatenated code [For66] proposed by David Forney can reach $\mathrm{BER} = 10^{-5}$ at about $\gamma_{\mathrm{b,dB}} \simeq 2$ dB. However, no schemes could push their performance curves within 1 dB of the Shannon limit until the invention of turbo codes in 1993 [BGT93]. Motivated by the turbo coding idea, the low-density parity-check (LDPC) codes were subsequently rediscovered[7] in 1998; these could

[7] We use "rediscover" here because the LDPC code was originally proposed by Robert G. Gallager in 1962 [Gal62]. However, due to its high complexity, computers at that time could not perform any simulations on the code; hence, nobody realized the potential of LDPC codes. It

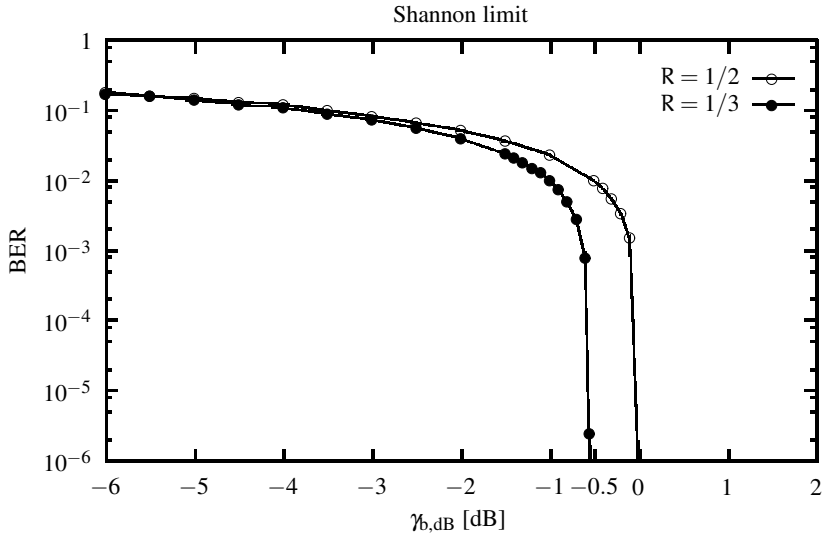

Figure 7.3 The Shannon limits for rates $1/2$ and $1/3$ codes on continuous-input AWGN channels. Decibel (abbreviated as dB) is a logarithmic scaling of a given quantity; i.e., we first take the base-10 logarithm and then multiply by 10. So, e.g., $\gamma_{b,dB} \triangleq 10\log_{10}(\gamma_b) = 10\log_{10}(E_b/N_0)$.

reduce the performance gap (between the LDPC codes and the Shannon limit) within, e.g., 0.1 dB. This counts 50 years of efforts (from 1948 to 1998) until we finally caught up with the pioneering prediction of Shannon in the classical additive Gaussian noise channel.

With excitement, we should realize that this is just the beginning of closing the gap, not the end of it. Nowadays, the channels we face in reality are much more complicated than the simple additive Gaussian noise channel. Multipath and fading effects, as well as channel nonlinearities, make the Shannon-limit approaching mission in these channels much more difficult. New ideas other than turbo and LDPC coding will perhaps be required in the future. So we are waiting for some new exciting results, similar to the discovery of turbo codes in 1993.

was Matthew Davey and David MacKay who rediscovered and examined the code, and confirmed its superb performance in 1998 [DM98].

7.5 Turbo coding: an introduction

Of all error-correcting codes to date, the turbo code was the first that could approach the Shannon limit within 1 dB, at BER $= 10^{-5}$, over the additive Gaussian noise channel. It is named the *turbo code* because the decoder functions iteratively like a turbo machine, where two turbo engines take turns to refine the previous output of the other until a certain number of iterations is reached.

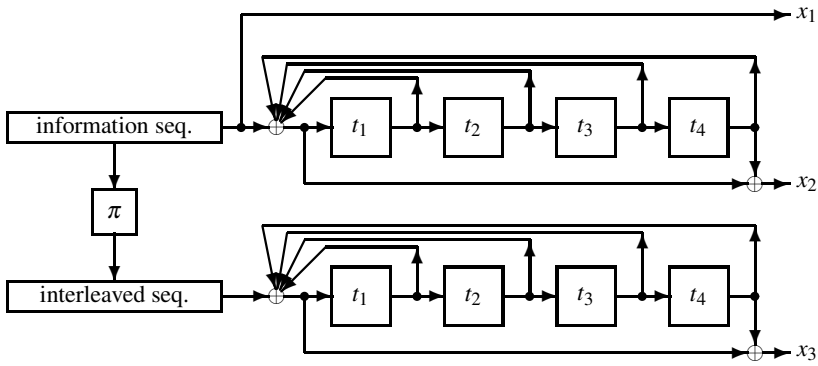

Figure 7.4 Exemplified turbo encoder from [BGT93]. An example of how a length-5 input sequence passes through this encoder is depicted in Figure 7.5. The complete list of all length-5 input sequences with their corresponding codewords is given in Table 7.2.

As an example, an information sequence $(s_1, s_2, s_3, s_4, s_5) = (10100)$ is fed into the turbo encoder shown in Figure 7.4, where s_1 is inserted first. In this figure, the squares marked with t_1, t_2, t_3, and t_4 are clocked memory elements, usually named *flip-flops* or simply *registers*, which store the coming input binary data and, at the same time, output its current content according to a clocked timer. The square marked with π is the interleaver that permutes the input sequence into its interleaved counterpart. The notation "$\oplus$" denotes the modulo-2 addition.

This figure then indicates that the output sequence from node x_1 will be the original information sequence $(s_1, s_2, s_3, s_4, s_5) = (10100)$. Since the contents of all four registers are initially zero, and since

$$t_1(\ell+1) = s_\ell \oplus t_1(\ell) \oplus t_2(\ell) \oplus t_3(\ell) \oplus t_4(\ell), \tag{7.55}$$

input sequence	register contents				output sequence
$s_5s_4s_3s_2s_1 \rightarrow$	t_1	t_2	t_3	t_4	$\rightarrow x_2(5)x_2(4)x_2(3)x_2(2)x_2(1)$
00101	0	0	0	0	
0010	1	0	0	0	1
001	1	1	0	0	11
00	1	1	1	0	111
0	1	1	1	1	1111
	0	1	1	1	11111

Figure 7.5 The snap show of the input and output sequences of the turbo encoder from Figure 7.4 at node x_2. Note that we have mirrored the sequences to match the direction of the register placement in Figure 7.4.

$$t_2(\ell+1) = t_1(\ell), \tag{7.56}$$

$$t_3(\ell+1) = t_2(\ell), \tag{7.57}$$

$$t_4(\ell+1) = t_3(\ell), \tag{7.58}$$

$$x_2(\ell) = t_1(\ell+1) \oplus t_4(\ell) = s_\ell \oplus t_1(\ell) \oplus t_2(\ell) \oplus t_3(\ell), \tag{7.59}$$

where ℓ represents the clocked time instance, we obtain the output sequence from node x_2 as 11111 (see Figure 7.5). Note that in order to start indexing all sequences from 1, we re-adjust the index at the output such that $x_2(\ell)$ is actually outputted at clocked time instance $\ell+1$.

An important feature of the turbo code design is the incorporation of an interleaver π that permutes the input sequence. For example,

$$\pi(s_1s_2s_3s_4s_5) = s_4s_2s_1s_5s_3. \tag{7.60}$$

In concept, the purpose of adding an interleaver is to introduce distant dependencies into the codewords. Notably, a strong dependency among the code bits can greatly enhance their capability against local noisy disturbance. A good example is a code of two codewords, i.e.

$$000000000000000 \quad \text{and} \quad 111111111111111,$$

where the code bits are all the same and hence are strongly dependent. Since the receiver knows all code bits should be the same, the local noisy disturbances that alter, for example, code bits 3, 7, and 10, yielding

$$001000100100000,$$

Table 7.2 *The information sequences of length 5 and their respective turbo codewords of length 15 for the turbo code in Figure 7.4 with interleaver* $\pi(s_1s_2s_3s_4s_5) = s_4s_2s_1s_5s_3$

Information sequences	Codewords		
$\mathbf{s}$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
$s_1s_2s_3s_4s_5$	$\mathbf{x}_1 = \mathbf{s}$	$x_2(1)x_2(2)x_2(3)x_2(4)x_2(5)$	$x_3(1)x_3(2)x_3(3)x_3(4)x_3(5)$
00000	—	00000	00000
00001	—	00001	00011
00010	—	00011	11001
00011	—	00010	11010
00100	—	00110	00001
00101	—	00111	00010
00110	—	00101	11000
00111	—	00100	11011
01000	—	01100	01100
01001	—	01101	01111
01010	—	01111	10101
01011	—	01110	10110
01100	—	01010	01101
01101	—	01011	01110
01110	—	01001	10100
01111	—	01000	10111
10000	—	11001	00110
10001	—	11000	00101
10010	—	11010	11111
10011	—	11011	11100
10100	—	11111	00111
10101	—	11110	00100
10110	—	11100	11110
10111	—	11101	11101
11000	—	10101	01010
11001	—	10100	01001
11010	—	10110	10011
11011	—	10111	10000
11100	—	10011	01011
11101	—	10010	01000
11110	—	10000	10010
11111	—	10001	10001

can be easily recovered back to the transmitted codeword

$$000000000000000.$$

By interleaving the information bits, s_1 may now affect distant code bits such as $x_3(\ell)$, where ℓ can now be much larger than the number of registers, 4. This is contrary to the conventional coding scheme for which the code bit is only a function of several recent information bits. For example, without interleaving, s_1 can only affect $x_2(4)$ but not any $x_2(\ell)$ with $\ell > 4$ according to (7.55)–(7.59). We can of course purposely design a code such that each code bit is a function of more distant information bits, but the main problem here is that the strong dependency of code bits on distant information bits will make the decoder infeasibly complex.

This leads to another merit of using the interleaver: it helps structure a feasible decoding scheme, i.e. turbo decoding. In short, the sub-decoder based on $\mathbf{x}_1$ and $\mathbf{x}_2$ will deal with a code that only has local dependencies as each code bit only depends on the previous four information bits. The second sub-decoder based on interleaved $\mathbf{x}_1$ and $\mathbf{x}_3$ similarly handles a code with only local dependencies. By this design, the task of decoding the code with distant dependencies can be accomplished by the cooperation of two feasible sub-decoders.

To be specific, the practice behind turbo decoding is to first decode the information sequence based on the noisy receptions due to the transmission of sequences $\mathbf{x}_1$ and $\mathbf{x}_2$ (in terms of the above example, 10100 and 11111). Since the code bits generated at node x_2 depend on previous information bits only through the contents of four registers, the sub-decoding procedure is feasible. The decoding output sequence, however, is not the final estimate about the information sequence 10100, but a sequence of real numbers that represent the probability for each bit to be, e.g. 1, calculated based on the noisy receptions due to the transmission of $\mathbf{x}_1$ and $\mathbf{x}_2$. Continuing with the example of the simple 5-bit input sequence, the decoding output sequence might be $(0.8, 0.2, 0.7, 0.1, 0.1)$. Based on these numbers, we know that with 80% probability the first bit is 1. Also, we assert with only 20% confidence that the second information bit is 1. Please note that if there is no noise during the transmission, the five real numbers in the decoding output sequence should be $(1.0, 0.0, 1.0, 0.0, 0.0)$. It is due to the noise that the receiver can only approximate the sequence that the transmitter sends. In terminology, we call these real numbers the *soft decoding outputs* in contrast to the conventional zero–one *hard decoding outputs*.

After obtaining the real-valued decoding output based on the noisy receptions due to the transmission of $\mathbf{x}_1$ and $\mathbf{x}_2$, one can proceed to refine these numbers by performing a similar decoding procedure based on the noisy re-

ceptions due to the transmission of $\mathbf{x}_1$ and $\mathbf{x}_3$ as well as the interleaved soft decoding output from the previous step, e.g. $(0.1, 0.2, 0.8, 0.1, 0.7)$ subject to the interleaver $\pi(s_1 s_2 s_3 s_4 s_5) = s_4 s_2 s_1 s_5 s_3$. With the additional knowledge from the noisy receptions due to the transmission of $\mathbf{x}_3$, these numbers may be refined to, e.g., $(0.05, 0.1, 0.9, 0.05, 0.8)$; hence, the receiver is more certain (here, 90% sure) that s_1 should be 1.

By performing the decoding procedure based on the noisy receptions due to $\mathbf{x}_1$ and $\mathbf{x}_2$ as well as the de-interleaved soft decoding output (e.g. $(0.9, 0.1, 0.8, 0.05, 0.05)$), these numbers are refined again. Then, in terms of these re-refined numbers, the decoding procedure based on the noisy receptions due to $\mathbf{x}_1$ and $\mathbf{x}_3$ is re-performed.

Because the repetitive decoding procedures are similar to running two turbo pistons alternatively, it is named *turbo coding*. Simulations show that after 18 iterations we can make the final hard decisions (i.e. 0 and 1 on each bit) based on the repeatedly refined soft decoding outputs yielding a bit error rate almost achieving the Shannon limit.

Ever since the publication of turbo coding, iterative decoding schemes have become a new research trend, and codes similar to the rediscovered LDPC code have subsequently been proposed. In this way the Shannon limit finally has become achievable after 50 years of research efforts!

7.6 Further reading

In this chapter we introduced the Information Transmission Theorem (Theorem 7.1) as a summary of what we have learned in this book. In order to appreciate the beauty of the theorem, we then examined it under a specific case when the coded information is corrupted by additive Gaussian noise. Two scenarios followed in a straightforward manner: transmission at a rate below capacity and transmission at a rate above capacity. The former directed us to reliable transmission where decoding errors can be made arbitrarily small, while the latter gave the (in principle) required $\mathsf{E}_\mathsf{b}/\mathsf{N}_0$ to achieve an acceptable BER. Information theorists have baptized this minimum $\mathsf{E}_\mathsf{b}/\mathsf{N}_0$ the *Shannon limit*. Since this is the center of information theory, most advanced textbooks in this area cover the subject extensively. For readers who are interested in learning more about the theorem, [CT06] could be a good place to start. The long-standing bible-like textbook [Gal68] by Robert G. Gallager, who was also the inventor of the Shannon-limit-achieving low-density parity-check (LDPC) codes, can also serve as a good reference. Some advanced topics in information theory, such as the channel reliability function, can be found in [Bla88].

As previously mentioned, the turbo code was the first empirically confirmed near-Shannon-limit error-correcting code. It is for this reason that the turbo code was introduced briefly at the end of this chapter. The two books [HW99] and [VY00] may be useful for those who are specifically interested in the practice and principle of turbo codes. Due to their significance, Shu Lin and Daniel Costello also devote one chapter for each of turbo coding and LDPC coding in their book in the new 2004 edition [LC04]. For general readers, the material in these two chapters should suffice for a comprehensive understanding of the related subjects.

7.7 Appendix: Why we assume uniform and independent data at the encoder

It is very common to assume that the input S_ℓ of a channel encoder comprise independent binary data bits of uniform distribution

$$\Pr[S_\ell = 0] = \Pr[S_\ell = 1] = \frac{1}{2}. \tag{7.61}$$

The reason for this lies in the observation that, under the assumption of independence with uniform marginal distribution, no data compression can be performed further on the sequence $S_1, S_2, \ldots, S_k$ since every binary combination of length k has the same probability 2^{-k} (or specifically, $(1/k)H(\mathbf{S}^k) = 1$ bit per source symbol). Note that any source that is compressed by an *optimal* data compressor should in principle produce a source output of this kind, and we can regard this assumption as that $S_1, S_2, \ldots, S_k$ are the output from an optimal data compressor.

For a better understanding of this notion, consider the following example. Assume that we wish to compress the sequence $U_1, U_2, U_3, \ldots$ to the binary sequence $S_1, S_2, S_3, \ldots$, and that each source output U_ℓ is independent of all others and has the probability distribution

$$\Pr[U_\ell = a] = \frac{1}{2}, \quad \Pr[U_\ell = b] = \Pr[U_\ell = c] = \frac{1}{4}. \tag{7.62}$$

We then use the single-letter (i.e. $\nu = 1$) Huffman code that maps as follows:

$$a \mapsto 0, \qquad b \mapsto 10, \qquad c \mapsto 11. \tag{7.63}$$

Note that (7.63) is also a Fano code (see Definition 5.17); in fact, when the source probabilities are reciprocals of integer powers of 2, both the Huffman code and the Fano code are optimal compressors with average codeword length

L_{av} equal to the source entropy, $H(U)$ bits. This then results in

$$\Pr\left[S_{\ell+1}=0 \,\middle|\, \mathbf{S}^{\ell}=\mathbf{s}^{\ell}\right] = \begin{cases} \Pr\left[U_{\ell'+1}=a \,\middle|\, \mathbf{U}^{\ell'}=\mathbf{u}^{\ell'}\right] & \text{if } \mathrm{code}(\mathbf{u}^{\ell'})=\mathbf{s}^{\ell}, \\ \Pr\left[U_{\ell'+1}=b \,\middle|\, \mathbf{U}^{\ell'}=\mathbf{u}^{\ell'} \text{ and } U_{\ell'+1}\neq a\right] & \text{if } \mathrm{code}(\mathbf{u}^{\ell'})=\mathbf{s}^{\ell-1} \\ & \text{and } s_{\ell}=1. \end{cases} \tag{7.64}$$

Here ℓ' denotes the symbol-timing at the input of the Huffman encoder (or, equivalently, the Fano encoder), and ℓ is the corresponding timing of the output of the Huffman encoder (equivalently, the Fano encoder).

We can now conclude by the independence of the sequence $U_1, U_2, U_3, \ldots$ that

$$\Pr\left[S_{\ell+1}=0 \,\middle|\, \mathbf{S}^{\ell}=\mathbf{s}^{\ell}\right] = \begin{cases} \Pr[U_{\ell'+1}=a] & \text{if } \mathrm{code}(\mathbf{u}^{\ell'})=\mathbf{s}^{\ell}, \\ \Pr[U_{\ell'+1}=b \,|\, U_{\ell'+1}\neq a] & \text{if } \mathrm{code}(\mathbf{u}^{\ell'})=\mathbf{s}^{\ell-1} \text{ and } s_{\ell}=1 \end{cases} \tag{7.65}$$

$$= \frac{1}{2}. \tag{7.66}$$

Since the resultant quantity $1/2$ is irrespective of the $\mathbf{s}^{\ell}$ given, we must have

$$\Pr\left[S_{\ell+1}=0 \,\middle|\, \mathbf{S}^{\ell}=\mathbf{s}^{\ell}\right] = \Pr[S_{\ell+1}=0] = \frac{1}{2}. \tag{7.67}$$

Hence, $S_{\ell+1}$ is independent of $\mathbf{S}^{\ell}$ and is uniform in its statistics. Since this is true for every positive integer ℓ, $S_1, S_2, S_3, \ldots$ is an independent sequence with uniform marginal distribution.

Sometimes, the output from an optimal compressor can only approach *asymptotic* independence with *asymptotic* uniform marginal distribution. This occurs when the probabilities of U are not reciprocals of powers of 2, i.e. different from what we have assumed in the previous derivation. For example, assume

$$\Pr[U=a]=\frac{2}{3} \quad \text{and} \quad \Pr[U=b]=\Pr[U=c]=\frac{1}{6}. \tag{7.68}$$

Then $S_1, S_2, S_3, \ldots$ can only be made asymptotically independent with asymptotic uniform marginal distribution in the sense that a multiple-letter code (i.e. a code that encodes several input letters at once; see Figure 5.12 in Section 5.5) needs to be used with the number of letters per compression growing to *infinity*. For example, the double-letter Huffman code in Table 7.3 gives

$$\Pr[S_1=0]=\Pr\left[\mathbf{U}^2=aa\right]=\left(\frac{2}{3}\right)^2=\frac{4}{9} \tag{7.69}$$

Table 7.3 *Double-letter and triple-letter Huffman codes for source statistics* $\Pr[U=a]=2/3$ *and* $\Pr[U=b]=\Pr[U=c]=1/6$

Letters	Code	Letters	Code	Letters	Code	Letters	Code
aa	0	*aaa*	00	*baa*	0111	*caa*	1110
ab	100	*aab*	1100	*bab*	10100	*cab*	10110
ac	110	*aac*	0100	*bac*	10101	*cac*	10111
bb	11100	*aba*	0101	*bba*	110100	*cba*	110110
ba	1010	*abb*	10000	*bbb*	1111000	*cbb*	1111100
bc	11101	*abc*	10001	*bbc*	1111001	*cbc*	1111101
ca	1011	*aca*	0110	*bca*	110101	*cca*	110111
cb	11110	*acb*	10010	*bcb*	1111010	*ccb*	1111110
cc	11111	*acc*	10011	*bcc*	1111011	*ccc*	1111111

and

$$\Pr[S_2=0]=\Pr\left[\left(\mathbf{U}^2=ab \text{ or } ba \text{ or } ca\right) \text{ or } \mathbf{U}^4=aaaa\right] \tag{7.70}$$

$$=\frac{2}{3}\cdot\frac{1}{6}+\frac{1}{6}\cdot\frac{2}{3}+\frac{1}{6}\cdot\frac{2}{3}+\left(\frac{2}{3}\right)^4 \tag{7.71}$$

$$=\frac{43}{81}. \tag{7.72}$$

These two numbers are closer to $1/2$ than those from the single-letter Huffman code that maps a,b,c to $0,10,11$, respectively, which gives

$$\Pr[S_1=0]=\Pr[U_1\neq a]=\frac{1}{3} \tag{7.73}$$

and

$$\Pr[S_2=0]=\Pr\left[(U_1=b) \text{ or } (\mathbf{U}^2=aa)\right]=\frac{33}{54}. \tag{7.74}$$

Note that the approximation from the triple-letter Huffman code may be *transiently* less "accurate" to the uniform distribution than the double-letter Huffman code (in the current example we have $\Pr[S_1=0]=16/27$, which is less close to $1/2$ than $\Pr[S_1=0]=4/9$ from (7.69)). This complements what has been pointed out in Section 5.4.2, namely that it is rather difficult to analyze (and also rather operationally intensive to examine numerically[8]) the average

[8] As an example, when ν (the number of source letters per compression) is only moderately large, such as 20, you can try to construct a Huffman code with source alphabet of size $|\{a,b,c\}|^{20}=3^{20}$ using Huffman's Algorithm from Section 4.6. Check how many iterations are required to root a tree with 3^{20} leaves.

Table 7.4 *Double-letter and triple-letter Fano codes for source statistics* $\Pr[U = a] = 2/3$ *and* $\Pr[U = b] = \Pr[U = c] = 1/6$

Letters	Code
aa	00
ab	01
ac	100
bb	11100
ba	101
bc	11101
ca	110
cb	11110
cc	11111

Letters	Code	Letters	Code	Letters	Code
aaa	00	*baa*	1001	*caa*	1010
aab	0100	*bab*	110011	*cab*	110101
aac	0111	*bac*	110100	*cac*	11011
aba	011	*bba*	111000	*cba*	11101
abb	1011	*bbb*	1111010	*cbb*	1111101
abc	110000	*bbc*	1111011	*cbc*	11111100
aca	1000	*bca*	111001	*cca*	111100
acb	110001	*bcb*	11111000	*ccb*	11111101
acc	110010	*bcc*	11111001	*ccc*	11111111

codeword length of Huffman codes. However, we can anticipate that better approximations to the uniform distribution can be achieved by Huffman codes if the number of letters per compression further increases.

In comparison with the Huffman code, the Fano code is easier in both analysis and implementation. As can be seen from Table 7.4 and Figure 7.6, $\Pr[S_1 = 0]$ quickly converges to $1/2$, and the assumption that the compressor output $S_1, S_2, S_3, \ldots$ is independent with uniform marginal distribution can be acceptable when ν is moderately large.

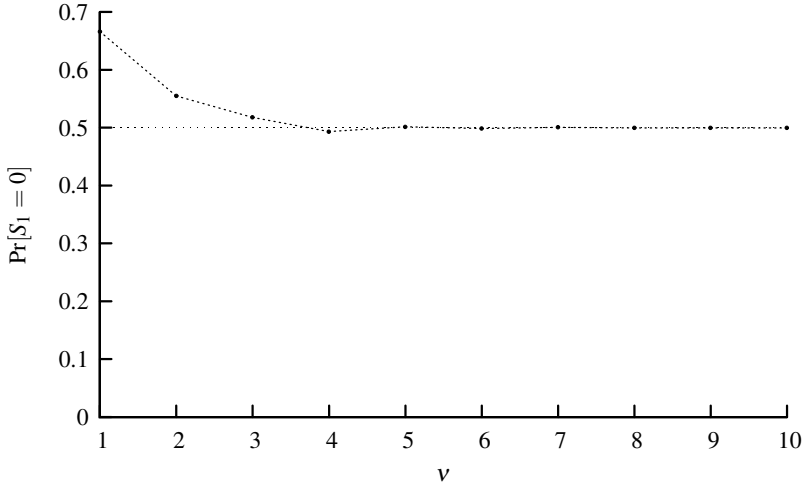

Figure 7.6 Asymptotics of ν-letter Fano codes.

7.8 Appendix: Definition of concavity

Definition 7.5 A real-valued function $h(\cdot)$ is *concave* if

$$h\big(\lambda p_1 + (1-\lambda)p_2\big) \geq \lambda h(p_1) + (1-\lambda)h(p_2) \tag{7.75}$$

for all real numbers p_1 and p_2, and all $0 \leq \lambda \leq 1$.

Geometrically, this means that the line segment that connects two points of the curve $h(\cdot)$ will always lie below the curve; see Figure 7.7 for an illustration.

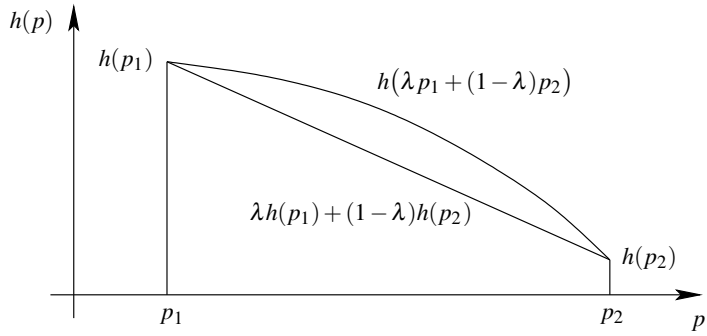

Figure 7.7 Example of a concave function.

The concavity of a function can be verified by showing that its second derivative is nonpositive. By this approach, we can prove that the binary entropy function is concave, as this can also be observed from Figure 5.2. By induction, a concave function satisfies

$$h\left(\frac{1}{k}\sum_{\ell=1}^{k} a_\ell\right) \geq \frac{1}{k}\sum_{\ell=1}^{k} h(a_\ell); \tag{7.76}$$

hence, (7.50) is also confirmed.

References

[BGT93] Claude Berrou, Alain Glavieux, and Punya Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," in *Proceedings of IEEE International Conference on Communications (ICC)*, Geneva, Switzerland, May 23–26, 1993, pp. 1064–1070.

[Bla88] Richard E. Blahut, *Principles and Practice of Information Theory*. Addison-Wesley, Reading, MA, 1988.

[CT06] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, 2nd edn. John Wiley & Sons, Hoboken, NJ, 2006.

[DM98] Matthew C. Davey and David MacKay, "Low-density parity check codes over $GF(q)$," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, June 1998.

[For66] G. David Forney, Jr., *Concatenated Codes*. MIT Press, Cambridge, MA, 1966.

[Gal62] Robert G. Gallager, *Low Density Parity Check Codes*. MIT Press, Cambridge, MA, 1962.

[Gal68] Robert G. Gallager, *Information Theory and Reliable Communication*. John Wiley & Sons, New York, 1968.

[HW99] Chris Heegard and Stephen B. Wicker, *Turbo Coding*. Kluwer Academic Publishers, Dordrecht, 1999.

[LC04] Shu Lin and Daniel J. Costello, Jr., *Error Control Coding*, 2nd edn. Prentice Hall, Upper Saddle River, NJ, 2004.

[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. Available: http://moser.cm.nctu.edu.tw/nctu/doc/shannon1948.pdf

[VY00] Branka Vucetic and Jinhong Yuan, *Turbo Codes: Principles and Applications*. Kluwer Academic Publishers, Dordrecht, 2000.