Chapter 2

Lossless Source Coding

We begin this chapter by describing the general source coding scenario in Section 2.1. Section 2.2 introduces the most rudimentary kind of source codes, called fixed-length to fixed-length block. Section 2.3 introduces "lossless" source coding, which is the focus of this chapter. ("Lossy" source coding will be the focus of Chapters 11 and 12.) The subsequent sections of the chapter investigate the limits to the performance of several different kinds of lossless source codes.

2.1 Introduction to Source Coding

Source coding is the process of representing data with binary symbols in a compact and accurate way. The scenario, illustrated in Figure 2.1.1, is the following. A *source* generates an infinite sequence of symbols $\widetilde{U} = (U_1, U_2, \ldots)$; this is the data we wish to represent. A *source encoder* produces an infinite binary *representation* $\widetilde{Z} = (Z_1, Z_2, \ldots)$ intended for transmission or storage. A *source decoder* creates a *reproduction* $\widehat{\widetilde{U}} = (\widehat{\widetilde{U}}_1, \widehat{\widetilde{U}}_2, \ldots)$ of $\widetilde{U}$ from $\widetilde{Z}$ and presents it to the *user.* Together the encoder and decoder constitute a *source code.*

The source symbols come from a set A_U called the *source alphabet*, and successive source outputs are modelled as random variables with this alphabet. In other words, the source is modelled as a random process, denoted $\{U_k\}$ or simply $\widetilde{U}$. Until otherwise stated, we will assume that $\widetilde{U}$ is stationary and memoryless; i.e., the U_k's are independent and identically

Sept. 2000

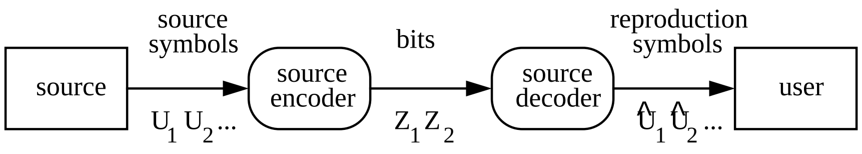

Figure 2.1.1: The source coding scenario.

distributed (IID).

We will adopt the conventions of Appendix A for characterizing the probability distributions of random variables. Accordingly, let $p_U(u)$ characterize the probability distribution of the U_k's. It is a probability mass function (pmf), when the U_k's are discrete, and a probability density function (pdf), when they are continuous.

The reproduction sequence $\hat{\tilde{U}}$ also consists of symbols from the source alphabet. The k^{th} reproduction symbol $\hat{U}_k$ is considered to be a reproduction of the k^{th} source symbol U_k.

There are two principal aspects to the performance of a source code: *compactness* and accuracy, or *fidelity*. On the one hand, a good source code produces a compact binary representation, i.e. one with few bits, for such a representation requires minimal resources for its transmission or storage. On the other hand, for obvious reasons, a good source code produces a high fidelity reproduction, i.e. each decoder output $\hat{U}_k$ is similar to the source symbol U_k for which it is a reproduction. Thus, when assessing source codes, there are two measures of performance: *rate*, which measures compactness, and *distortion*, which measures fidelity — actually the lack of fidelity. These are more carefully defined below.

There are actually two measures of rate, both defined in terms of the function $L_k(U_1, \dots, U_k)$, which denotes the number of bits produced by the encoder after it receives U_k and before it receives U_{k+1} and which may depend on the previously received symbols $U_1, \dots, U_{k-1}$. The *empirical average rate* of the code when encoding source sequence $\tilde{U}$ is

$$\langle R \rangle \triangleq \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} L_k(U_1, \dots, U_k). \tag{2.1.1}$$

When, as is usual in this book, we have a random process model for the source data, we can also compute the *statistical average rate*

$$\overline{R} \triangleq \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} EL_k(U_1, \dots, U_k), \tag{2.1.2}$$

where E denotes expected value.

There are also two measures of distortion — empirical and statistical. Both are defined in terms of a user specified *distortion measure* d, which is a function such that $d(u, \widehat{u})$ indicates the lack of fidelity, i.e. distortion, in $\widehat{u}$ when used as a reproduction of the source symbol u. Specifically, d is a non-negative, real-valued function that maps $A_U \times A_U$ into $[0, \infty)$. Small distortion indicates good fidelity and large distortion indicates poor. The *empirical average distortion* of the code when encoding source sequence $\widetilde{U}$ is

$$\langle D \rangle \triangleq= \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} d(U_k, \widehat{U}_k). \tag{2.1.3}$$

And when we have a random process model for the source data, the *statistical average distortion* is

$$\overline{D} \triangleq \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} Ed(U_k, \widehat{U}_k). \tag{2.1.4}$$

It is important to notice that the empirical average performances measures (rate and distortion) often depend on the source sequence being encoded, i.e. they can be different for different source sequences. Similarly, the statistical average performance measures often depend on the random process model for the source; i.e. they can be different for different models. In this book we are concerned mostly with statistical average performance and the terms average rate, average distortion, rate, and distortion will mean statistical averages, unless otherwise stated. However, it is important to understand that empirical average performance is what somone using the source code would actually measure, whereas the statistical average performance is what one usually computes when designing a source code. The value in computing the latter, is that it is ordinarily a good predictor of the former. In any case, a good code is one with small average rate and distortion — empirical and/or statistical.

It should come as no surprise that there is a conflict between compactness and fidelity. That is, it is hard to make one of them small without making the other large. In other words, there is a tradeoff between rate and distortion. Quantifying this tradeoff is one of the principal goals of our study of source coding.

Remarks

(1) We choose to focus on binary representations, as opposed to ternary or M-ary representations (for some integer M) because of their widespread

appearance in transmission and storage systems. It would be equally possible to work with M-ary representations, and it is easy to convert what we learn about binary representations to M-ary representations. The decision to label the two symbols "0" and "1" is entirely arbitrary, and the only justification we offer is that it is the most widely adopted convention.

(2) In any practical system, it is always possible that some of the representation bits may be modified by noise or other phenomena before presentation to the decoder. Although this could have a significant affect on the fidelity of the code, we have not included the possibility of such "transmission errors" in our source coding scenario, because we wish to focus on the fundamental limitations of the source coding process in and of itself. However, there is one place later in this chapter where we briefly discuss the effects of errors on one type of source code, and in Chapter 10 we will see that in situations where transmission errors are prevalent we may follow the source code with a *channel code* that protects the binary representation from such transmission errors.

(3) Another important measure of the goodness of a source code is its complexity or its cost of implementation. While we shall not introduce formal measures of such, we urge the reader to consider what might be involved in implementing the various codes presented in this book. For example, how many arithmetic operations are required per source symbol for encoding and decoding? And how many symbols must be saved in auxiliary storage? From time to time we shall comment on such matters.

(4) Sometimes sources emit their symbols at regular intervals of time, for example, S_U symbols per second. While this is not always the case, it can clarify the sequential nature of the source coding process to add such an assumption to the source coding scenario. With this in mind we note that when a source with *symbol rate* S_U is encoded with a code with rate R bits per symbol, the encoder produces $S_Z = S_U R$ bits per second, which we call the *code symbol rate*. Now we see that the term "rate" could mean one of three things S_Z, S_U or R, so we need to be sure to add the appropriate modifier.

(5) There are situations where the reproduction alphabet is different than the source alphabet, for example, when color images are to be displayed on a monitor that displays only sixteen shades of gray. The theory of source coding can be extended straightforwardly to this case.

However, for simplicity we have assumed that the source and reproduction alphabets are the same.

(6) There are some situations where the limits included in the definitions of rate and distortion (2.1.1)-(2.1.4) might not exist. In such cases, the conservative thing is to replace the "limit" with a "lim sup". (See any standard book on elementary analysis for a definition of a "lim sup".) In this book, we shall restrict attention to codes, source sequences and source models for which the limits exist. Thus we will not need to use lim sup's.

(7) The distortions of codes defined in (2.1.3) and (2.1.4) are called *per-letter* because they average a distortion defined individually for successive symbols. We point out here that some types of infidelity cannot be adequately measured by a per-letter type distortion, no matter how the distortion measure d is chosen. For example, a per-letter average distortion cannot measure the degree to which a reproduction preserves the edges in an image or the short-term power spectra in a speech recording. Although such infidelities may indeed be quite important, information theory is primarily oriented towards per-letter distortions.

2.2 Fixed-Length to Fixed-Length Block Source Codes

Fixed-length to fixed-length block (FFB) codes are the most rudimentary source codes. We will focus on them through Section 2.6, and again in Chapters 11 and 12. An FFB code is characterized by a positive integer K called the *source length*, another positive integer L called the *code length*, a *codebook* C containing binary sequences of length L called *codewords*, a mapping f_e called an *encoding rule* that assigns codewords to source sequences of length K, and a mapping f_d called a *decoding rule*, that assigns source sequences of length K to codewords.

The code operates in the following "block fashion". See Figure 2.2.1. The encoder waits until K symbols have arrived from the source, forming a *block* $\underline{U}_1 = (U_1, \ldots, U_K)$. It then applies the encoding rule and produces the codeword $f_e(\underline{U}_1)$, which becomes the first L representation bits, $\underline{Z}_1 = (Z_1, \ldots, Z_L)$. These bits are transmitted or stored one by one. The encoder then waits for the next block of source symbols, $\underline{U}_2 = (U_{K+1}, \ldots, U_{2K})$, applies the encoding rule and produces the next L representation bits $\underline{Z}_2 = (Z_{L+1}, \ldots, Z_{2L}) = f_e(\underline{U}_2)$, transmits them one by one, and so on. The meaning of "in block fashion" should now be evident. The decoder operates in a similar manner. It waits for the first L representation bits $\underline{Z}_1$ applies the decoding rule f_d, produces the first K reproduction symbols $\underline{\hat{U}}_1 = (U_1, \ldots, U_K) = f_d(\underline{Z}_1)$ and presents them to the user one by one. It then waits for the next L bits $\underline{Z}_2$ decodes them producing $\underline{\hat{U}}_2 = (\hat{U}_{K+1}, \ldots, \hat{U}_{2K})$ and so on.

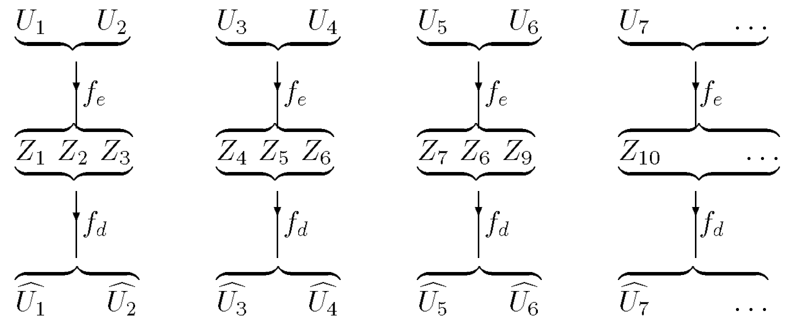
U_1 U_2 U_3 U_4 U_5 U_6 U_7 ...
f_e
Z_1 Z_2 Z_3 Z_4 Z_5 Z_6 Z_7 Z_6 Z_9 Z_{10} ...
f_d
$\widehat{U_1}$ $\widehat{U_2}$ $\widehat{U_3}$ $\widehat{U_4}$ $\widehat{U_5}$ $\widehat{U_6}$ $\widehat{U_7}$ ...

Figure 2.2.1: (a) The "block operation" of an FFB code with $K = 2,\ L = 3$

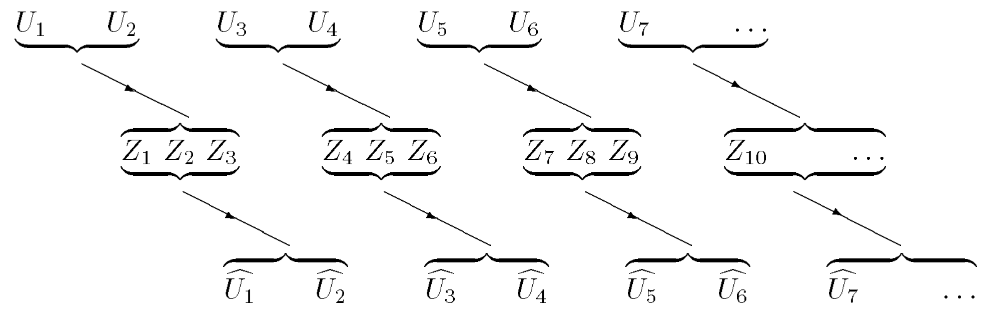
U_1 U_2 U_3 U_4 U_5 U_6 U_7 ...
Z_1 Z_2 Z_3 Z_4 Z_5 Z_6 Z_7 Z_8 Z_9 Z_{10} ...
$\widehat{U_1}$ $\widehat{U_2}$ $\widehat{U_3}$ $\widehat{U_4}$ $\widehat{U_5}$ $\widehat{U_6}$ $\widehat{U_7}$ ...

Figure 2.2.1: (b) The same code with time delays shown.

We will frequently refer to the rules f_e and f_d as if they *are* the encoder and decoder, respectively, instead of merely mappings that describe them.

When the source and reproduction alphabets are finite one may use tables to describe the encoding and decoding rules. For example, see Figures 2.2.2-2.2.4. One may visualize these rules with *point diagrams* such as that in Figure 2.2.5 for the example of Figure 2.2.2.

We now discuss the performance of an FFB source code, i.e. the rate and distortion. Recalling that the rate of a source code, defined by (2.1.2), is the average number of representation bits per source symbol, it is easy to see that the rate of an FFB code (statistical average) is

$$\overline{R} = \frac{L}{K}\ , \tag{2.2.1}$$

Encoding Rule f_e

U_1	U_2	Z_1	Z_2	Z_3	Z_4
a	a	0	0	0	0
a	b	0	0	0	1
a	c	0	0	1	0
b	a	0	0	1	1
b	b	0	1	0	0
b	c	0	1	0	1
c	a	0	1	1	0
c	b	0	1	1	1
c	c	1	0	0	0

Decoding Rule f_d

Z_1	Z_2	Z_3	Z_4	$\widehat{U}_1$	$\widehat{U}_2$
0	0	0	0	a	a
0	0	0	1	a	b
0	0	1	0	a	c
0	0	1	1	b	a
0	1	0	0	b	b
0	1	0	1	b	c
0	1	1	0	c	a
0	1	1	1	c	c
1	0	0	0	c	c

Figure 2.2.2: An FFB code with $K = 2,\ L = 4$.

Encoding Rule f_e

U_1	Z_1
a	0
b	0
c	1
d	1

Decoding Rule f_d

Z_1	$\widehat{U}_1$
0	a
1	c

Figure 2.2.3: An FFB code with $K = 1,\ L = 1$.

regardless of the source model. Because the rate never changes (indeed their empirical average rate is L/K as well), FFB codes are sometimes called *fixed-rate codes.*

Exercise 2.2.1 *(a) Find an expression for the function $L_k(u_1, \ldots, u_k)$. (b) Prove (2.2.1). (c) Prove that emprical average rate $\langle R \rangle = L/K$, as well, for any source sequence. source.* □

For an FFB Code, the distortion (statistical average) defined by (2.1.4) simplifies to

$$\overline{D} = \frac{1}{K} \sum_{k=1}^{K} E\ d(U_k, \widehat{U}_k)\ . \qquad (2.2.2)$$

Exercise 2.2.2 *(a) Prove (2.2.2). You will need to make use of the IID nature of the source. (b) Reprove (2.2.2) assuming only that the source is stationary.* □

We conclude this section by commenting on the implementation and complexity of FFB codes. One way to implement their encoding and decoding rules is simply to store and use encoding tables, such as those shown in Figures 2.2.2-2.2.4. The principal thing to notice is that the amount of storge required for a table is proportional to the number of its rows, which is Q^K for FFB encoding or decoding. This means that the storage required for table look-up encoding and decoding increases exponentially with source length K, and indicates that complexity should be viewed as growing exponentially with source length. Thus, FFB codes can be expensive to use, unless K is kept small.

				z_7	0	0	0	0	1	1	1	1
				z_6	0	0	1	1	0	0	1	1
				z_5	0	1	0	1	0	1	0	1
z_1	z_2	z_3	z_4									
0	0	0	0		NUL	DLE	SP	0	@	P	'	p
0	0	0	1		BS	CAN	(	8	H	X	h	x
0	0	1	0		BOT	DC4	$	4	D	T	d	t
0	0	1	1		FF	FS	,	<	L	/	l	∫
0	1	0	0		STX	DC2	"	2	B	R	b	r
0	0	0	1		LF	SUB	*	:	J	Z	j	z
0	1	1	0		ACK	SYN	&	6	F	V	f	v
0	1	1	1		SO	RS	.	>	N	^	n	~
1	0	0	0		SOH	DC1	!	1	A	Q	a	q
1	0	0	1		HT	EM	)	9	I	Y	i	y
1	0	1	0		ENQ	NAK	%	5	E	U	e	u
1	0	1	1		CR	GS	-	=	M	[	m	}
1	1	0	0		EXT	DC3	#	3	C	S	c	s
1	1	0	1		VT	ESC	+	;	K	]	k	{
1	1	1	0		BEL	ETB	'	7	G	W	g	w
1	1	1	1		SI	US	/	?	O	-	o	DEL

Figure 2.2.4: The decoding table of the ASCII Code, which is an FFB code with $K = 1$, $L = 7$ for an alphabet with 128 symbols.

2.3 Introduction to Lossless Source Coding

Lossless source coding (also called *noiseless source coding*) is the special case of source coding in which the user demands "essentially" no distortion and asks for as small a rate as possible.

To quantify "essentially no distortion", it is customary to adopt the *Hamming distortion measure*:

$$d_H(u, \hat{u}) \triangleq \begin{cases} 0, & u = \hat{u} \\ 1, & u \neq \hat{u} \end{cases} \quad . \tag{2.3.1}$$

In this case the average distortion between the k^{th} source symbol U_k and its

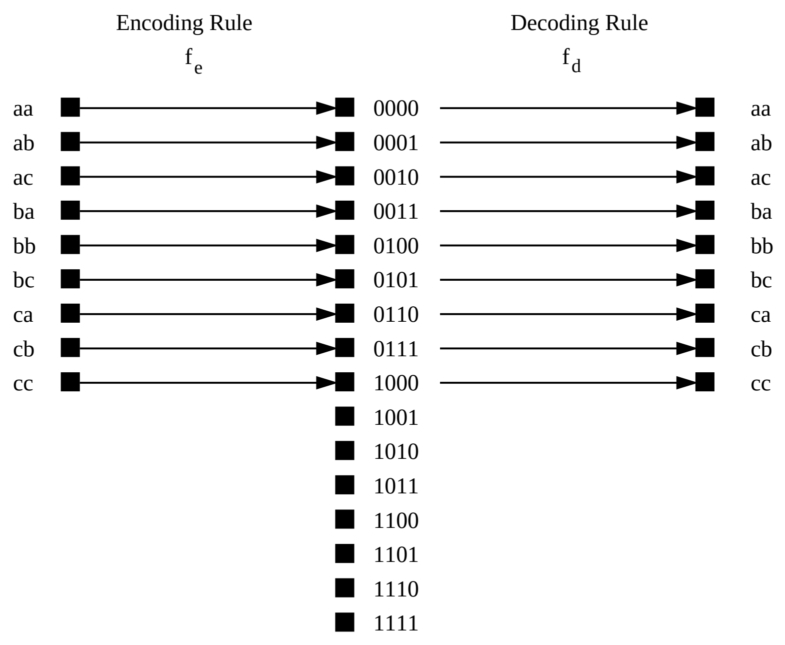

Figure 2.2.5: A point diagram for visualizing the encoding and decoding rules of Figure 2.2.2

reproduction $\hat{U}_k$ becomes the probability that they differ; i.e.,

$$E\, d_H(U_k, \widehat{U}_k) = \Pr(U_k \neq \widehat{U}_k) \ , \tag{2.3.2}$$

and distortion of the code reduces to *per-letter error probability*

$$\overline{D} = \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \Pr(U_k \neq \widehat{U}_K) \triangleq P_{LE}. \tag{2.3.3}$$

For a block code with source length K, the result of Exercise 2.2.2 implies that this further reduces to

$$\overline{D} = P_{LE} = \frac{1}{K} \sum_{k=1}^{K} \Pr(U_k \neq \widehat{U}_k). \tag{2.3.4}$$

Consequently, the demand for "essentially no distortion" translates to a demand for $P_{LE} \cong 0$.

In this chapter, the main question we answer is:

Question 2.3.1 *What is the smallest rate of codes with $P_L(\mathcal{E}) \cong 0$?*

As a start, in Section 2.4 we study FFB codes with P_{LE} exactly zero; these will be called *perfectly lossless.* Next in Sections 2.5 and 2.6, we will see that significantly smaller rates are attainable with FFB codes if P_{LE} is permitted to be a little large than zero. Such codes will be called *almost lossless*. Finally, in Section 2.7, we will investigate codes with variable-length codewords that are perfectly lossless, yet have the smaller rates just mentioned. In Chapters 11 and 12 we shall consider source coding at rates below those attainable by lossless source coding. Such codes introduce non-negligible amounts of distortion.

Remark

(1) Lossless coding with finite rates is impossible unless the source is discrete-valued. This is easy to establish for FFB codes (see the exercise below) and holds equally well for all other kinds of codes, including the variable-length codes considered later in this chapter. To simplify discussion, unless otherwise stated, we will assume that the source has a finite alphabet. Occasionally, however, we shall indicate how the results for finite alphabets extend to countably infinite alphabets.

Exercise 2.3.1 *Show that P_{LE}) $= 1$ for any FFB code with finite rate applied to any continous-valued source. Hint: Such codes can only have a finite number of codewords.* □

2.4 Perfectly Lossless FFB Source Codes

In this section we find the least rate of perfectly lossless fixed-length to fixed-length block codes. This is the "obvious" case and treating it explicitly will permit us to see clearly the gains of more serious source coding techniques to be presented later. Specifically, we will find

$$R^*_{PL}(K) \triangleq \min \left\{ r : \begin{array}{l} \text{there is a perfectly lossless FFB code} \\ \text{with source length } K \text{ and rate } r \end{array} \right\}, \quad (2.4.1)$$

which is the least rate of any perfectly lossless FFB code with source length K, and

$$\begin{aligned} R^*_{PL} &\triangleq \inf \left\{ r : \begin{array}{l} \text{there is a perfectly lossless FFB code} \\ \text{(with any source length) and rate } r \end{array} \right\} \\ &= \inf \{ R^*_{PL}(K) : K = 1, 2, \ldots \} , \end{aligned} \quad (2.4.2)$$

which is the least rate of any perfectly lossless FFB code of any blocklength.[2]

[2]We write "inf" instead of "min" because there need not actually be a smallest rate r at which there is a perfectly lossless FFB code. But there will always be a smallest number r such that there exist perfectly lossless codes with rates arbitrarily close to r, and this number is called the *infimum* and denoted *inf*. For example, $\min\{x \in (0,1]\}$ does not exist but $\inf\{x \in (0,1]\}$ equals 0.

As indicated in Remark (1) of the previous section, we will assume here and throughout the rest of this chapter that the source alphabet A_U is finite, specifically, having the Q symbols $\{a_1, a_2, \ldots, a_Q\}$.

In order for an FFB code with source length K to be perfectly lossless, it must have

$$f_d(f_e(u^K)) = u^K, \text{ for all source sequences } u^K, \tag{2.4.3}$$

which happens if and only if the encoding rule f_e is a one-to-one mapping of source sequences to codewords and the decoding rule f_d is its inverse. Consequently, a distinct binary codeword of length L must be assigned to each of the Q^K source sequences of length K. Since only 2^L distinct binary sequences are available to be used as codewords, L and K must be chosen so that $2^L \geq Q^K$, or equivalently, so that $L \geq \lceil K \log_2 Q \rceil$, where $\lceil c \rceil$ denotes the smallest integer no smaller than c. It follows that the rate of a perfectly lossless FFB code can be no smaller than $\lceil K \log_2 Q \rceil / K$.

Conversely, if K and L are any two integers such that $2^L \geq Q^K$, equivalently $L \geq \lceil K \log_2 Q \rceil$, then one may design a perfectly lossless code with rate $L/K \geq \lceil K \log_2 Q \rceil / K$ — simply let the encoding rule be any one-to-one mapping from source sequences of length K to binary sequences of length L, and let the decoding rule be its inverse.

We conclude that the least rate of perfectly lossless FFB codes with source length K is

$$R^*_{PL}(K) = \frac{\lceil K \log_2 Q \rceil}{K} \quad . \tag{2.4.4}$$

Since $K \log_2 Q \leq \lceil K \log_2 Q \rceil < K \log_2 Q + 1$, we obtain the following upper and lower bounds to $R^*_{PL}(K)$

$$\log_2 Q \leq R^*_{PL}(K) \leq \log_2 Q + \frac{1}{K} \quad . \tag{2.4.5}$$

And since R^*_{PL}, the least rate of perfectly lossless FFB codes with any source length, is just the infimum of $R^*_{PL}(K)$ over all source lengths K, we see that $R^*_{PL} = \log_2 Q$. We summarize with the following theorem, which is the first of many "coding theorems" to appear in this book.

Theorem 2.4.1 (Coding Theorem for Perfectly Lossless FFB Codes) *For any source with a Q symbol alphabet, the least rate of any perfectly lossless FFB code with source length K is*

$$R^*_{PL}(K) = \frac{\lceil K \log_2 Q \rceil}{K} \quad , \tag{2.4.6}$$

and the least rate of any perfectly lossless FFB code with any source length is

$$R^*_{PL} = \log_2 Q \quad . \tag{2.4.7}$$

Each conclusion of this theorem may be decomposed into a *positive* and a *negative* statement. The positive statement corresponding to (2.4.6) is that there exists a perfectly lossless FFB code with source length K and rate equal to $\log_2 Q$; the negative statement is that no perfectly lossless FFB code with source length K has rate less than $\log_2 Q$. The positive statement corresponding to (2.4.7) is that there exist perfectly lossless FFB codes with rates arbitrarily close to $\log_2 Q$. The negative statement is that no perfectly lossless FFB codes have rate less than $\log_2 Q$. We will see in future sections and chapters that all coding theorems have positive and negative statements — the positive specifying that a certain degree of good performance is possible, the negative specifying that no better performance is possible.

Notice that according to the upper bound to $R^*_{PL}(K)$ in (2.4.5), as K increases, $R^*_{PL}(K)$ approaches $\log_2 Q$ at least as rapidly as $1/K$. However, as the following exercise shows, the approach is not always monotonic, and the upper bound can be loose or tight.

Exercise 2.4.1 *Assuming $Q = 3$, find R^*_{PL} and $R^*_{PL}(K)$ for $K = 1$ to 6. Does $R^*_{PL}(K)$ decrease monotonically with K? How tight is the upper bound provided by (2.4.5)?* □

Exercise 2.4.2 *For what values of Q will there be perfectly lossless FFB codes with rate exactly equal to $\log_2 Q$?* □

Example 2.4.1 *When English text is to be encoded, the alphabet A_U certainly contains the 26 letters $\{a, b, \ldots, z\}$. But it must also contain the symbol "space", as this too must be encoded. In this case, $r^*_{PL} = \log_2 27 = 4.75$ bits/character. If, in addition, we wish to distinguish capital and lower case letters, then $r^*_{PL} = \log_2 53 = 5.72$. The ASCII code shown in Figure 2.2.4 uses 7 bits to represent 128 different symbols, including the lower and upper case letters, space, the ten numerals 0, 1, 2, ..., 9, the standard punctuation symbols, common symbols such as %, & and a variety of computer control characters.* □

Exercise 2.4.3 *Show that that if the source alphabet is infinite, there can be no perfectly lossless FFB codes.* □

2.5 Almost Lossless FFB Source Codes

We now consider the possibility of designing FFB codes with rate less than $\log_2 Q$. Because of Theorem 2.4.1, such codes cannot be perfectly lossless, but it turns out they have arbitrarily small error probability. In this section we will sketch the principal ideas; careful statements and proofs will be left to the next section and chapter. The main goal is to find

$$R^*_{AL} \triangleq \inf \left\{ r : \begin{array}{l} \text{for any } \delta > 0, \text{ there is an FFB code with} \\ P_{LE} \leq \delta \text{ and rate } R \text{ less than or equal} \\ \text{to } r + \delta \end{array} \right\} , \qquad (2.5.1)$$

which is the precise way of defining the smallest rate at which arbitrarily small error probability is achievable.

We begin by examining what contributes to error probability. Given an FFB code with source length K, code length L, codebook C, encoding rule f_e and decoding rule f_d, the per-letter error probability is (by the result of Exercise 2.2.2)

$$P_{LE} = \frac{1}{K} \sum_{k=1}^{K} \Pr(U_k \neq \widehat{U}_k), \qquad (2.5.2)$$

where $(\widehat{U}_1, \ldots, \widehat{U}_K) = f_d(f_e(U_1, \ldots, U_K))$. Unfortunately, it is usually rather difficult to compute P_{LE} or to make theoretical developments in terms of it. Instead, it is easier to work with the *block error probability*

$$P_{BE} \triangleq \Pr(U^K \neq \widehat{U}^K) = \Pr(U_1 \neq \widehat{U}_1 \text{ or } U_2 \neq \widehat{U}_2 \text{ or } \ldots \text{ or } U_K \neq \widehat{U}_K), \qquad (2.5.3)$$

which is closely related to P_{LE} via

$$\frac{1}{K} P_{BE} \leq P_{LE} \leq P_{BE}. \qquad (2.5.4)$$

Exercise 2.5.1 *Prove the above inequalities.* □

The upper bound $P_{LE} \leq P_{BE}$ is especially important. For if you design a system to have small P_{BE}, the user will be comfortable knowing that P_{LE}, the real concern, is no larger. From now on we shall use P_{BE} in all further discussions of lossless block coding.

Given some FFB code, let G denote the set of *correctly encoded source sequences*; i.e. the set of source sequences of length K that are encoded and decoded without error. Formally,

$$G = \{u^K : f_d\left(f_e(u^K)\right) = u^K\}. \qquad (2.5.5)$$

See Figure 2.5.1. We will show that the performance of the code is expressly related to properties of G. First, the error probability is related to the probability of G via

$$P_{BE} = \Pr(U^K \notin G) = 1 - \Pr(U^K \in G). \qquad (2.5.6)$$

Second, the rate of the code is related to the size of G by the fact that there must be a distinct codeword in the codebook for every correctly encoded sequence in G (otherwise they would not be correctly encoded and decoded). Since codewords are binary sequences of length L and since there are only 2^L such binary sequences, it must be that

$$|G| \leq 2^L \tag{2.5.7}$$

or, equivalently, that $L \geq \log_2 |G|$, where $|G|$ denotes the number of sequences in G. Consequently, the rate of the code is bounded by

$$R = \frac{L}{K} \geq \frac{\log_2 |G|}{K}. \tag{2.5.8}$$

Thus we see that if one has a good code (low rate and $P_{BE} \cong 0$), then the set G of correctly encoded sequences is a "small" set with probability close to one.

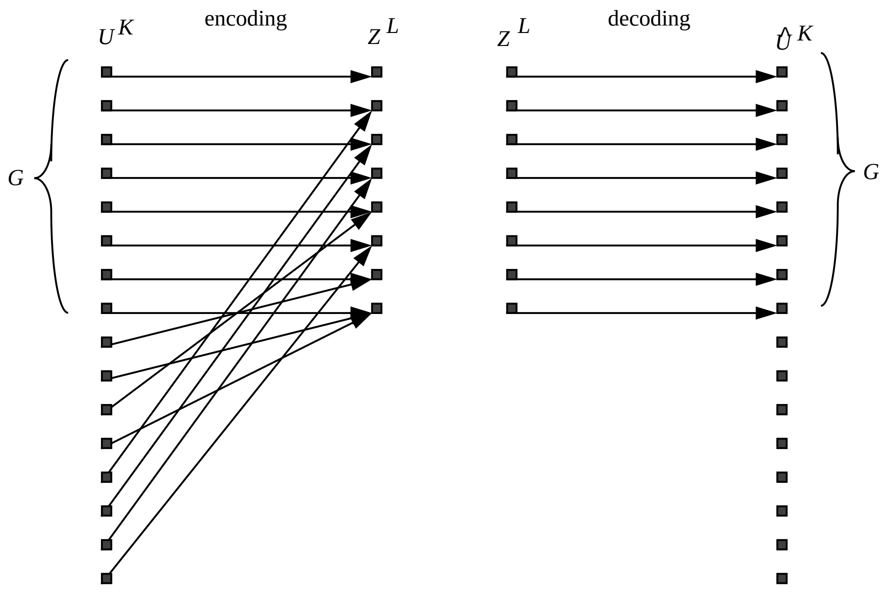

Figure 2.5.1: The set G of correctly encoded sequences. Each square represents one sequence.

Conversely, if one can find a "small" set of source sequences $\tilde{G}$ with probability close to one, then one can use it as the basis for designing a good almost lossless FFB code (low rate and $P_{BE} \cong 0$), by choosing the encoder and decoder so that $\tilde{G}$ becomes the correctly encoded set. Specifically, make f_e assign a distinct binary codeword of length $L = \lceil \log_2 |\tilde{G}| \, \rceil$ to every sequence in $\tilde{G}$, make f_e assign an already chosen codeword to every source sequence not in $\tilde{G}$, and make f_d map each codeword into the source sequence from $\tilde{G}$ that generates it. Accordingly, one obtains a code with rate $R = \lceil \log_2 |\tilde{G}| \rceil / K$ and error probability $P_{BE} = 1 - \Pr(\tilde{G}) \cong 0$.

From the above discussion we conclude that the key question in almost lossless FFB coding is:

Question 2.5.1 *How small is the smallest set of source sequences of length K with probability nearly one?*

This question can be studied apart from source coding; it is just a matter of how $p(u^K)$ distributes probability over source sequences of length K. Does it spread probability fairly uniformly, or does it mostly concentrate probability on a relatively small set, which could then be used as the basis for an almost lossless FFB code? If it concentrates probability on a set $\widetilde{G}$ whose size is significantly smaller than Q^K (the total number of source sequences of length K), then there is an almost lossless FFB code with rate $\lceil \log_2 |\widetilde{G}| \rceil / K$, which is less than $\log_2 Q$, the least rate of perfectly lossless FFB codes.

We will show that when K is large, Question 2.5.1 may be answered with the law of large numbers, for example, the weak law of large numbers. A brief discussion of this law is given in Section A.7.2 of Appendix A, and a thorough discussion is given in Chapter 3. Here, we will merely state what we need and sketch the idea for its use.

Recall that our source is an IID random process $\{U_k\}$ with finite alphabet $A_U = \{a_1, \ldots, a_Q\}$ and probability mass function $p(u)$. Let p_q be a shorthand notation for $p(a_q)$. The weak law of large numbers (WLLN) shows that when K is large, the fraction of times that a symbol a_q occurs in the K random variables $U_1, \ldots, U_K$ is, with high probability, approximately equal to p_q, for every symbol a_q in the alphabet. To make this concrete, let $n_q(U^K)$ denote the number of times that a_q appears in U^K. Then the WLLN shows that for any positive number ϵ (that we ordinarily choose to be small)

$$\Pr\left(\frac{n_q(U^K)}{K} \doteq p_q \pm \epsilon, \;\; q = 1, \ldots, Q \right) \longrightarrow 1 \text{ as } K \longrightarrow \infty, \qquad (2.5.9)$$

where $a \doteq b \pm \epsilon$ is shorthand for $|a-b| \le \epsilon$ or, equivalently, $b - \epsilon \le a \le b + \epsilon$. In other words, when K is large, it is very likely that each symbol in the alphabet occurs in U^K with a frequency close to its probability.

Like any event involving the random vector U^K, the event $\{n_q(U^K)/K \doteq p_q \pm \epsilon, \text{ for } q = 1, \ldots, Q\}$ can be expressed in the form $\{U^K \in T\}$, where T is some set of outcomes of U^K. Specifically,

$$T \triangleq \left\{ u^K : \frac{n_q(u^K)}{K} \doteq p_q \pm \epsilon, \text{ for } q = 1, \ldots, Q \right\}. \qquad (2.5.10)$$

Since every sequence in T has the property that each symbol a_q occurs with a frequency close to its probability and since this constitutes "typical" behavior, we will from now on call such sequences *typical.* In this terminology, the weak law of large numbers says that when K is large, the outcome of the random vector U^K will, with high probability, be typical; i.e., it will be one of the typical sequences in T. Equivalently,

$$\Pr(U^K \in T) \cong 1 \ . \tag{2.5.11}$$

Bearing in mind that we wish to find the smallest set with large probability and that T is, at least, a set with large probability, let us count how many sequences it contains. The feasibility of doing so derives from the key fact that all sequences in T have approximately the same probability. To demonstrate this fact, recall that the IID nature of the source implies that the probability of any sequence is a product of the probabilities of its components:

$$p(u^K) = p(u_1)p(u_2)\cdots p(u_K) \ . \tag{2.5.12}$$

Each term in this product is either p_1 or p_2 or ... or p_Q; specifically, $p(u_i) = p_q$ if $u_i = a_q$. Since $n_q(u^K)$ is the number of times a_q appears in u^K, the product may be rewritten in the form

$$p(u^K) = p_1^{n_1(u^K)} p_2^{n_2(u^K)} \cdots p_Q^{n_Q(u^K)} \ . \tag{2.5.13}$$

Now if u^K is typical (i.e., a member of T), then $n_q(u^K) \cong Kp_q$ (assuming ϵ is chosen small) and, consequently,

$$\begin{aligned} p(u^K) &\cong p_1^{Kp_1} p_2^{Kp_2} \ldots p_2^{Kp_Q} \\ &\triangleq p^*(K) \ , \end{aligned} \tag{2.5.14}$$

which shows that each sequence in T has, approximately, the same probability.

Let us now return to the counting of T. Since each sequence in T has probability approximately equal to $p^*(K)$, and since T has probability approximately equal to one, the number of sequences in T must be, approximately, $1/p^*(K)$. Thus we have determined the size of T.

Having found its size, we now argue that T is, essentially, the smallest set with probability close to one. This is because the approximately $1/p^*(K)$ (typical) sequences in T, each having probability approximately equal to $p^*(K)$, account for essentially all of the probability in the distribution of U^K. It follows that the probability of any other set is, approximately, $p^*(K)$ times the number of typical sequences that it contains. Consequently, the only way to form a set with probability close to one is to include essentially all of the sequences of T (the set might also contain other sequences with very small probability). We conclude that T, is essentially, as small as any set with probability close to one.

It is now evident that $p^*(K)$ is a very important quantity. It will be worthwhile to rewrite it as

$$\begin{aligned} p^*(K) &\triangleq p_1^{Kp_1} p_2^{Kp_2} \dots p_2^{Kp_Q} \\ &= 2^{Kp_1 \log_2 p_1} 2^{Kp_2 \log_2 p_2} \dots 2^{Kp_Q \log_2 p_Q} \\ &= 2^{K \sum_{q=1}^{Q} p_q \log_2 p_q} \\ &= 2^{-KH} \end{aligned} \tag{2.5.15}$$

where

$$H \triangleq - \sum_{q=1}^{Q} p_q \log_2 p_q \ . \tag{2.5.16}$$

This shows that $p^*(K)$ decreases exponentially with block size K. The quantity H, which is a simple function of the symbol probabilities, determines the rate of the exponential decrease. Of equal importance is the fact that in terms of H, the size of T is

$$|T| \cong 2^{KH} \ ; \tag{2.5.17}$$

i.e. it increases exponentially with K, with rate determined by H.

We now have the complete answer to Question 2.5.1. When K is large, the smallest set of length K source sequences with probability close to one contains approximately 2^{KH} sequences. Moreover, the probability distribution of U^K assigns nearly equal probability to each sequence in this set. This is often called the *asymptotic equipartition property* (AEP), because it says that asymptotically for large K the probability distribution is, essentially, equally divided among a certain set of sequences. The reader is cautioned that so far we have given only a rough statement of this result and a sketch of its derivation. Careful statements and proofs are the subject of Chapter 3, where it is formally stated and proved in the *Shannon-McMillan Theorem.*

Returning to source coding, it follows from the AEP that H is the least rate attainable with almost lossless FFB codes. (A code designed so that T is the set of correctly encoded sequences has rate $R = \lceil \log_2 |T| \rceil / K \cong H$.) A careful statement and proof of this fact is given in the next section, where it is called the *Coding Theorem for Almost Lossless FFB codes.* Among other things it is shown there that our approximate method of counting is adequate.

Clearly, H is a quantity of fundamental importance. Shannon dubbed it *entropy* because it has the same functional form as thermodynamical entropy. Although it will be thoroughly explored in Chapter 4, we would be remiss not to mention one of its principal properties here, namely,

$$0 \leq H \leq \log_2 Q, \tag{2.5.18}$$

with $H = 0$ if and only if $p_q = 1$ for some q (i.e., if and only if there is no uncertainty about the outcome of U) and with $H = \log_2 Q$ if and only if $p_q = 1/Q$ for all outcomes (i.e., if and only if there is the maximum possible uncertainty about which outcome will occur). This suggests that H can be viewed as a measure of the amount of randomness or uncertainty in the outcome of U. In any event, we see that when the outcomes of U are not equiprobable, then $H < \log_2 Q$, and consequently, almost lossless FFB codes can outperform perfectly lossless FFB codes. As entropy places limits on the rate of codes, we take its units to be those of rate, namely, bits per source symbol.

Example 2.5.1 *The entropy of a binary probability distribution* $\{p, 1-p\}$, *as a function of* p, *is*

$$H = -p \log_2 p - (1-p) \log_2 (1-p) \ , \tag{2.5.19}$$

which is plotted in Figure 2.5.2. Notice that H *is a convex* $\cap$ *function of* p *(see Appendix A) that is symmetric about* $p = 1/2$ *and that increases steeply as* p *departs from either* 0 *or* 1, *reaching a peak of* 1 *at* $p = 1/2$. *For instance, if* $p = .1$, *then* $H = .47$. *This means that the least rate of almost lossless FFB codes is .47 bits per symbol. In comparison the least rate of perfectly lossless FFB codes is* $\log_2 2 = 1$ *bit per source symbol.* □

Although we know that almost lossless FFB codes can have rate as small as H, we have had no indication of how large their source lengths K need to be. To get a feeling for this, Figure 2.5.3 plots error probability vs. rate for the best possible FFB codes with various source lengths, and for the binary source of the previous example with $p = .1$ and $H = .47$. The figure shows that very large source lengths are needed in order that the rate be close to entropy and the error probability be very small. For example, source length 200 is needed to obtain, approximately, error probability 10^{-5} and rate .7, which is 50% larger than $H = .47$. In truth, this is somewhat disappointing, because it indicates that very large (and consequently expensive) FFB codes are needed to achieve the excellent performance predicted by this theory. Fortunately, there is an alternate approach, to be discussed in Section 2.7, that yields perfectly lossless codes at rates arbitrarily close to H with far less complexity.

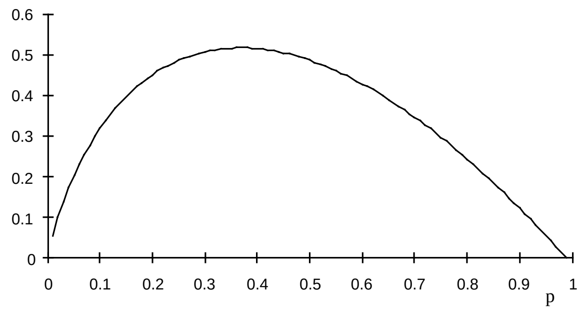

Figure 2.5.2: (a) $-p \log_2 p$

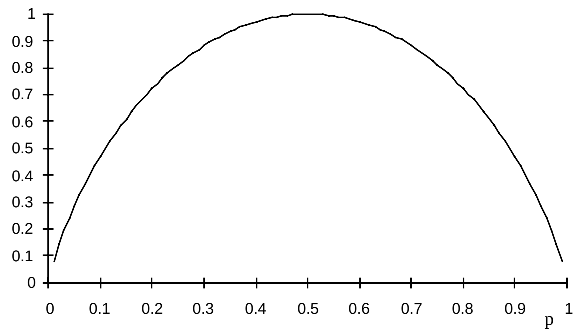

Figure 2.5.2: (b) Entropy of a binary variable: $H = -p \log_2 p - (1 - p) \log_2 (1 - p)$

Exercise 2.5.2 *For positive integers K and n, $1 \leq n \leq K$, let $G_{K,n}$ denote the set of all binary sequences of length K with n or fewer ones. Find expressions for the block error probability and rate of an FFB code having $G_{K,n}$ as its set of correctly encoded sequences. These probablities and rate are what are plotted in Figure 2.5.3.* □

Example 2.5.2 *Estimates of the probabilities of the 26 letters and "space" in the English alphabet are shown in Figure 2.5.4. The corresponding entropy is 4.08 bits per source symbol. In comparison, it would take 5 bits per source symbol to encode English text with a perfectly lossless FFB code.* □

Exercise 2.5.3 *(a) Show that $H \geq 0$; and $H = 0$ if and only if $p_q = 1$ for some q. (b) Show that $H \leq \log_2 Q$; and $H = \log_2 Q$ if and only if the a_q's are equiprobable. (Hint: Use the relation $\ln u \leq u - 1$ with equality if and only if $u = 1$ in the sum $\sum_{q=1}^{Q} p_q \log_2 \frac{1/Q}{p_q}$.)* □

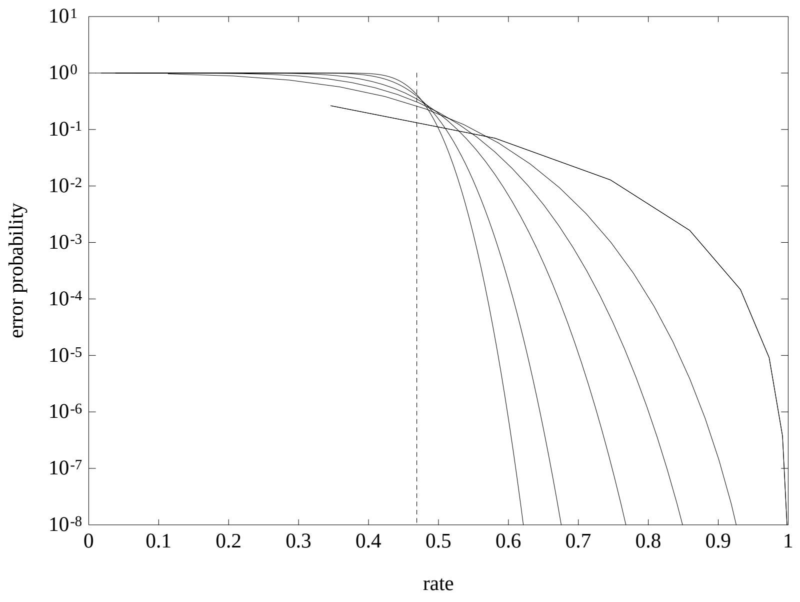

Figure 2.5.3: Block error probability vs. rate for the best FFB codes for a binary IID source with Pr(1)=.1. From right to left, the plotted curves correspond to source lengths $K = 10, 50, 100, 200, 500, 1000$. The dashed line indicates the entropy, $H = .47$ bits/symbol.

Exercise 2.5.4 *(From Gallager) An IID binary source has $p_0 = .995$ and $p_1 = .005$. An almost lossless FFB code is to be designed with source length $K = 100$ such that the set of correctly encoded sequences contains all sequences with 3 or fewer 1's. (a) Find the minimum possible rate of such a code. (b) Find the block error probability P_{BE}. (c) Use the Chebychev inequality (A.5.11) to find an upper bound to P_{BE}) and compare the result to that of part (b).* □

2.6 The Coding Theorem for Almost Lossless FFB Source Codes

In the previous section we learned from the asymptotic equipartition property that for large K the smallest set of length K source sequences with probability close to one contains approximately 2^{KH} sequences, where H is the entropy of the source. This fact was then used to argue that H is the least rate of any FFB code with small error probability. This important result about almost lossless coding is made precise in Theorem 2.6.1, whose statement and proof are the topic of this section.

Symbol	Probability	Symbol	Probability
A	.0642	O	.0632
B	.0127	P	.0152
C	.0218	Q	.0008
D	.0317	R	.0484
E	.1031	S	.0514
F	.0208	T	.0796
G	.0152	U	.0228
H	.0467	V	.0083
I	.0575	W	.0175
J	.0008	X	.0013
K	.0049	Y	.0164
L	.0321	Z	.0005
M	.0198	Space	.1859
N	.0574		

Figure 2.5.4: Frequencies of English letters: $H = -\sum_{j=1}^{27} p_j \log_2 p_j = 4.08$ bits.

In order to state the theorem, let us define $P^*_{BE}(r, K)$ to be the smallest block error probability of any FFB source code with source length K and rate less than or equal to r. That is,

$$P^*_{BE}(r, K) \triangleq \inf \left\{ p : \begin{array}{l} \text{there is an FFB code with source} \\ \text{length } K\text{, rate } r \text{ or less, and block} \\ \text{error probability } p \end{array} \right\} . \tag{2.6.1}$$

Theorem 2.6.1 (Coding Theorem for Almost Lossless FFB Source Codes) *Let $\widetilde{U}$ be an IID source with entropy H.*

(a) Positive statement: For any $r > H$,

$$P^*_{BE}(r, K) \longrightarrow 0 \text{ as } K \longrightarrow \infty \tag{2.6.2}$$

which implies

$$R^*_{AL} \leq H \tag{2.6.3}$$

(b) Negative statement (converse): For any $r < H$

$$P^*_{BE}(r, K) \longrightarrow 1 \text{ as } K \longrightarrow \infty \tag{2.6.4}$$

The positive statement says there are almost lossless FFB codes with rate as close as one could want to H. It does not, however, tell us whether there are almost lossless codes with even smaller rates. This is the role of the negative statement (or converse), which says that codes with rate less than H and large source length have large block error probability.

This theorem does not entirely answer the question of what is the least rate of almost lossless block codes, i.e. it does not completely specify R^*_{AL}, because the converse leaves open the possibility that for small source lengths, there may be almost lossless codes with rate less than H. It also leaves open the possibility that codes with rate less than H (with large or small source lengths) might have small per-letter error probability. (Recall that the latter can be less than block error probability.) A complete answer to the question must be postponed to Chapter 5, where it is shown that all codes with rate less than H (with large or small source length) have per-letter error probability bounded from below by a monotonic function of rate that is strictly greater than 0.

As previously indicated, the proof of this theorem makes use of the asymptotic equipartition property, which was sketched in the previous section and will be the principal topic of Chapter 3. For convenience, the version we need (from Chapter 3) is carefully stated below.

Theorem 2.6.2 (The Shannon-McMillan Theorem) *Let U be an IID source with entropy H.*

(a) Positive statement: For any $\epsilon > 0$ and positive integer K, there exists a set T_ϵ^K containing source sequences of length K and [3]

$$\text{(i)} \quad \Pr(U^K \in T_\epsilon^K) \longrightarrow 1 \text{ as } K \longrightarrow \infty, \tag{2.6.5}$$

$$\text{(ii)} \quad p(u^K) \doteq 2^{-K(H\pm\epsilon)}, \text{ for all } u^K \in T_\epsilon^K, \tag{2.6.6}$$

$$\text{(iii)} \quad |T_\epsilon^K| \doteq \Pr\left(U^K \in T_\epsilon^K\right) 2^{K(H\pm\epsilon)} \tag{2.6.7}$$

(b) Negative statement (converse): For any $\epsilon > 0$, there is a positively valued function $a_\epsilon(K)$ that converges to zero as $K \longrightarrow \infty$ such that for any positive integer K and any set S containing source sequences of length K,

$$|S| \geq \left(\Pr(U^K \in S) - a_\epsilon(K)\right) 2^{K(H-\epsilon)}. \tag{2.6.8}$$

[3]The notation $b \doteq f(a \pm \epsilon)$ means

$$\min_{-\epsilon \leq \delta \leq \epsilon} f(a+\delta) \leq b \leq \max_{-\epsilon \leq \delta \leq \epsilon} f(a+\delta)$$

Proof of Theorem 2.6.1

(a) Positive statement

Let us fix a number $r > H$. To show, as we must, that $P^*_{BE}(r, K) \longrightarrow 0$ as $K \longrightarrow \infty$, we will construct a sequence of FFB codes with increasing source lengths such that the code with source length K has block error probability P_{BE} going to zero as $K \longrightarrow \infty$ and rate R_K that is less than or equal to r for all sufficiently large K. Since these codes have $R_K \le r$ for all sufficiently large K, it must be that $P^*_{BE}(r, K) \le P_{BE}$ for all sufficiently large K. And since P_{BE} tends to zero as $K \longrightarrow \infty$, so must $P^*_{BE}(r, K)$ tend to zero, which will complete the proof.

To show the existence of a suitable sequence of FFB codes, let us apply the Positive Statement of the Shannon-McMillan Theorem with $\epsilon = (r - H)/2$. It shows that for every positive integer K there is a set T^K_ϵ of source sequences of length K such that (2.6.5)-(2.6.7) hold.

As in the previous section, for any K let us design an FFB code with source length K so that T^K_ϵ becomes the set of correctly encoded source sequences. That is, we make the encoder f_e assign a distinct binary codeword of length $L = \lceil \log_2 |T^K_\epsilon| \rceil$ to each sequence in T^K_ϵ, make f_e assign an already chosen codeword to each source sequence not in T^K_ϵ, and make f_d map each codeword into the source sequence from T^K_ϵ that generates it. The encoding rule is pictured in Figure 2.6.1. In this way, for all K we obtain a code with block error probability

$$P_{BE} = 1 - \Pr(U^K \in T^K_\epsilon) \ , \tag{2.6.9}$$

which goes to zero as $K \longrightarrow \infty$ by (2.6.5). The rate of this code is

$$\begin{aligned}
R_K &= \frac{L}{K} = \frac{\lceil \log_2 |T^K_\epsilon| \rceil}{K} \\
&< \frac{\log_2 |T| + 1}{K} \\
&\le \frac{K(H+\epsilon)+1}{K} \\
&= H + \epsilon + \frac{1}{K} \\
&\le H + 2\epsilon \quad \text{for all sufficiently large } K \\
&= r \ ,
\end{aligned} \tag{2.6.10}$$

where the second inequality used (2.6.7) and the fact that $\Pr(U^K \in T^K_\epsilon) \le 1$, and where the last equality used the fact that $\epsilon = (r - H)/2$. This shows what we set out to prove and, therefore, completes the proof of the positive statement.

The equivalent positive statement can be verified as follows . . . (Not written yet.)

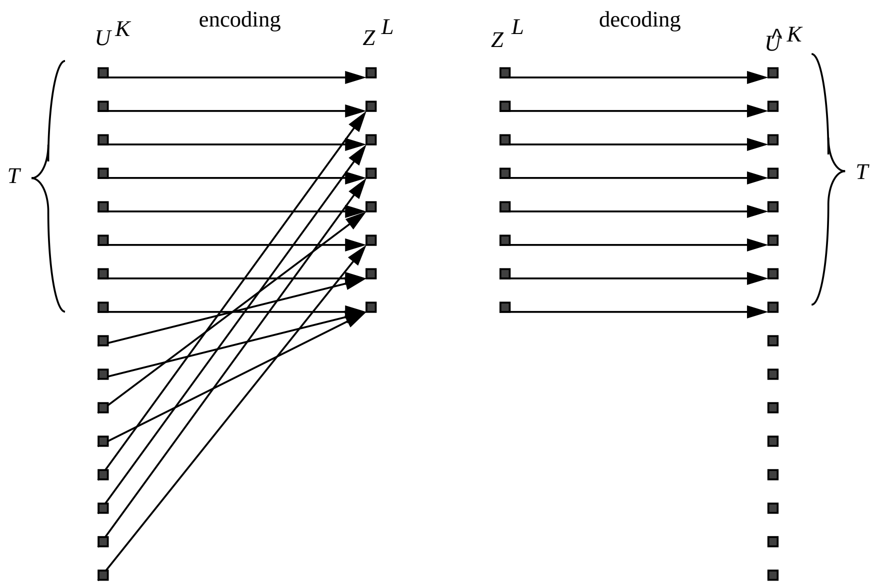

Figure 2.6.1: A code with T as the set of correctly encoded sequences. Each square represents one sequence.

(b) Negative statement

Let us fix a number $r < H$. To show, as we must, that $P^*_{BE}(r, K) \longrightarrow 1$ as $K \longrightarrow \infty$, we will find a lower bound to the block error probability of every FFB code with source length K and rate r or less, which tends to one as $K \longrightarrow \infty$.

Let us apply the Negative Statement of the Shannon-McMillan Theorem with $\epsilon = (H - r)/2$. It shows there exists a positively valued function $a_\epsilon(K)$ that converges to zero as $K \longrightarrow \infty$ such that for any positive integer K and any set S containing source sequences of length K,

$$|S| \geq \Big(\Pr(U^K \in S) - a_\epsilon(K)\Big)\, 2^{K(H-\epsilon)} \ . \tag{2.6.11}$$

Equivalently,

$$\Pr(U^K \in S) \leq |S| 2^{-K(H-\epsilon)} + a_\epsilon(K) \tag{2.6.12}$$

Now consider an arbitrary FFB code with source length K, code length L, rate $R = L/K \leq r$, codebook C, encoding rule f_e and decoding rule f_d. Let G denote the correctly encoded set of source sequences; i.e., $G = \{u^K :$

$f_d(f_e(u^K)) = u^K\}$. Then as argued in the previous section, the code's block error probability is

$$P_{BE} = 1 - \Pr(U^K \in G), \tag{2.6.13}$$

and the number of sequences in G can be no larger than the number of codewords, which in turn is no larger than 2^L. Hence,

$$|G| \le |C| \le 2^L = 2^{KR} \le 2^{Kr} \ . \tag{2.6.14}$$

Substituting G for S in (2.6.12) and using the above bound on $|G|$ gives

$$\begin{aligned} \Pr(U^K \in G) &\le 2^{Kr} 2^{K(H-\epsilon)} + a_\epsilon(K) \\ &= 2^{K(H-r-\epsilon)} + a_\epsilon(K) \\ &= 2^{K\epsilon} + a_\epsilon(K) \ , \end{aligned} \tag{2.6.15}$$

where the last equality used $\epsilon = (H-r)/2$. Finally, using $P_{BE} = 1 - \Pr(G)$ in the above yields

$$P_{BE} \ge 1 - 2^{\epsilon} - a_\epsilon(K) \ . \tag{2.6.16}$$

Notice that the right hand side of the above converges to one as K goes to ∞. Thus, as we set out to do, we have found a lower bound to the block error probability of every FFB code with source length K and rate r or less, which converges to one. This completes the proof of the negative statement and the entire theorem. □

Remarks

(1) Notice that the approximations for $p(u^K)$ and $|T_\epsilon^K|$ given in the Shannon-McMillan theorem are really quite loose because $2^{K\epsilon}$ grows to infinity as K increases. However, since the rate of the code based on T_ϵ^K is the logarithm of $|T_\epsilon^K|$ divided by K, these loose bounds were sufficient to prove the important result contained in the coding theorem.

(2) A simpler and in some respects stronger negative statement, called the *per-letter converse to the lossless source coding theorem,* will be given in Chapter 5.

(3) Although the results of this section show that almost lossless FFB codes can reduce the rate to, approximately, H (which in some cases is a big reduction), unfortunately the source lengths required to achieve this reduction are not small. For example, they may be on the order of 50 to 100. Since an FFB code needs to store the 2^{Kr} correctly encoded sequences, we see that this method is too complex for practical implementation, when for example $K = 50$ and $r = 1$.