1
Introduction

Systems dedicated to the communication or storage of information are commonplace in everyday life. Generally speaking, a communication system is a system which sends information from one place to another. Examples include telephone networks, computer networks, audio/video broadcasting, etc. Storage systems, e.g. magnetic and optical disk drives, are systems for storage and later retrieval of information. In a sense, such systems may be regarded as communication systems which transmit information from now (the present) to then (the future). Whenever or wherever problems of information processing arise, there is a need to know how to compress the textual material and how to protect it against possible corruption. This book is to cover the fundamentals of *information theory* and *coding theory*, to solve the above main problems, and to give related examples in practice. The amount of background mathematics and electrical engineering is kept to a minimum. At most, simple results of calculus and probability theory are used here, and anything beyond that is developed as needed.

1.1 Information theory versus coding theory

Information theory is a branch of probability theory with extensive applications to communication systems. Like several other branches of mathematics, information theory has a physical origin. It was initiated by communication scientists who were studying the statistical structure of electrical communication equipment and was principally founded by Claude E. Shannon through the landmark contribution [Sha48] on the mathematical theory of communications. In this paper, Shannon developed the fundamental limits on data compression and reliable transmission over noisy channels. Since its inception, information theory has attracted a tremendous amount of research effort and provided lots

of inspiring insights into many research fields, not only communication and signal processing in electrical engineering, but also statistics, physics, computer science, economics, biology, etc.

Coding theory is mainly concerned with explicit methods for efficient and reliable data transmission or storage, which can be roughly divided into data compression and error-control techniques. Of the two, the former attempts to compress the data from a source in order to transmit or store them more efficiently. This practice is found every day on the Internet where data are usually transformed into the ZIP format to make files smaller and reduce the network load.

The latter adds extra data bits to make the transmission of data more robust to channel disturbances. Although people may not be aware of its existence in many applications, its impact has been crucial to the development of the Internet, the popularity of compact discs (CD), the feasibility of mobile phones, the success of the deep space missions, etc.

Logically speaking, coding theory leads to information theory, and information theory provides the performance limits on what can be done by suitable encoding of the information. Thus the two theories are intimately related, although in the past they have been developed to a great extent quite separately. One of the main purposes of this book is to show their mutual relationships.

1.2 Model and basic operations of information processing systems

Communication and storage systems can be regarded as examples of information processing systems and may be represented abstractly by the block diagram in Figure 1.1. In all cases, there is a *source* from which the information originates. The information source may be many things; for example, a book, music, or video are all information sources in daily life.

Figure 1.1 Basic information processing system.

The source output is processed by an *encoder* to facilitate the transmission (or storage) of the information. In communication systems, this function is often called a *transmitter*, while in storage systems we usually speak of a

recorder. In general, three basic operations can be executed in the encoder: source coding, channel coding, and modulation. For source coding, the encoder maps the source output into digital format. The mapping is one-to-one, and the objective is to eliminate or reduce the redundancy, i.e. that part of the data which can be removed from the source output without harm to the information to be transmitted. So, source coding provides an efficient representation of the source output. For channel coding, the encoder introduces extra redundant data in a prescribed fashion so as to combat the noisy environment in which the information must be transmitted or stored. Discrete symbols may not be suitable for transmission over a physical channel or recording on a digital storage medium. Therefore, we need proper modulation to convert the data after source and channel coding to waveforms that are suitable for transmission or recording.

The output of the encoder is then transmitted through some physical communication channel (in the case of a communication system) or stored in some physical storage medium (in the case of a storage system). As examples of the former we mention wireless radio transmission based on electromagnetic waves, telephone communication through copper cables, and wired high-speed transmission through fiber optic cables. As examples of the latter we indicate magnetic storage media, such as those used by a magnetic tape, a hard-drive, or a floppy disk drive, and optical storage disks, such as a CD-ROM[1] or a DVD.[2] Each of these examples is subject to various types of noise disturbances. On a telephone line, the disturbance may come from thermal noise, switching noise, or crosstalk from other lines. On magnetic disks, surface defects and dust particles are regarded as noise disturbances. Regardless of the explicit form of the medium, we shall refer to it as the *channel*.

Information conveyed through (or stored in) the channel must be recovered at the destination and processed to restore its original form. This is the task of the *decoder*. In the case of a communication system, this device is often referred to as the *receiver*. In the case of a storage system, this block is often called the *playback system*. The signal processing performed by the decoder can be viewed as the inverse of the function performed by the encoder. The output of the decoder is then presented to the final user, which we call the *information sink*.

The physical channel usually produces a received signal which differs from the original input signal. This is because of signal distortion and noise introduced by the channel. Consequently, the decoder can only produce an estimate

[1] CD-ROM stands for *compact disc read-only memory*.
[2] DVD stands for *digital video disc* or *digital versatile disc*.

of the original information message. All well-designed systems aim at reproducing as reliably as possible while sending as much information as possible per unit time (for communication systems) or per unit storage (for storage systems).

1.3 Information source

Nature usually supplies information in continuous forms like, e.g., a beautiful mountain scene or the lovely chirping of birds. However, digital signals in which both amplitude and time take on discrete values are preferred in modern communication systems. Part of the reason for this use of digital signals is that they can be transmitted more reliably than analog signals. When the inevitable corruption of the transmission system begins to degrade the signal, the digital pulses can be detected, reshaped, and amplified to standard form before relaying them to their final destination. Figure 1.2 illustrates an ideal binary digital pulse propagating along a transmission line, where the pulse shape is degraded as a function of line length. At a propagation distance where the transmitted pulse can still be reliably identified (before it is degraded to an ambiguous state), the pulse is amplified by a digital amplifier that recovers its original ideal shape. The pulse is thus regenerated. On the other hand, analog signals cannot be so reshaped since they take an infinite variety of shapes. Hence the farther the signal is sent and the more it is processed, the more degradation it suffers from small errors.

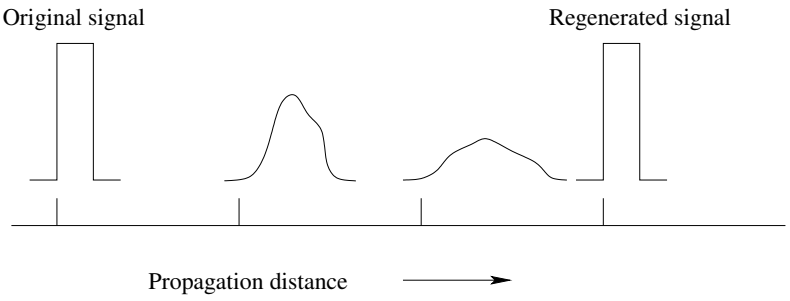

Figure 1.2 Pulse degradation and regeneration.

Modern practice for transforming analog signals into digital form is to sample the continuous signal at equally spaced intervals of time, and then to quantize the observed value, i.e. each sample value is approximated by the nearest

level in a finite set of discrete levels. By mapping each quantized sample to a codeword consisting of a prescribed number of code elements, the information is then sent as a stream of digits. The conversion process is illustrated in Figure 1.3. Figure 1.3(a) shows a segment of an analog waveform. Figure 1.3(b) shows the corresponding digital waveform based on the binary code in Table 1.1. In this example, symbols 0 and 1 of the binary code are represented by zero and one volt, respectively. Each sampled value is quantized into four binary digits (bits) with the last bit called *sign bit* indicating whether the sample value is positive or negative. The remaining three bits are chosen to represent the absolute value of a sample in accordance with Table 1.1.

Table 1.1 *Binary representation of quantized levels*

Index of quantization level	Binary representation	Index expressed as sum of powers of 2
0	000	
1	001	2^0
2	010	2^1
3	011	2^1+2^0
4	100	2^2
5	101	2^2+2^0
6	110	2^2+2^1
7	111	$2^2+2^1+2^0$

As a result of the sampling and quantizing operations, errors are introduced into the digital signal. These errors are nonreversible in that it is not possible to produce an exact replica of the original analog signal from its digital representation. However, the errors are under the designer's control. Indeed, by proper selection of the sampling rate and number of the quantization levels, the errors due to the sampling and quantizing can be made so small that the difference between the analog signal and its digital reconstruction is not discernible by a human observer.

1.4 Encoding a source alphabet

Based on the discussion in Section 1.3, we can assume without loss of generality that an information source generates a finite (but possibly large) number of messages. This is undoubtedly true for a digital source. As for an analog

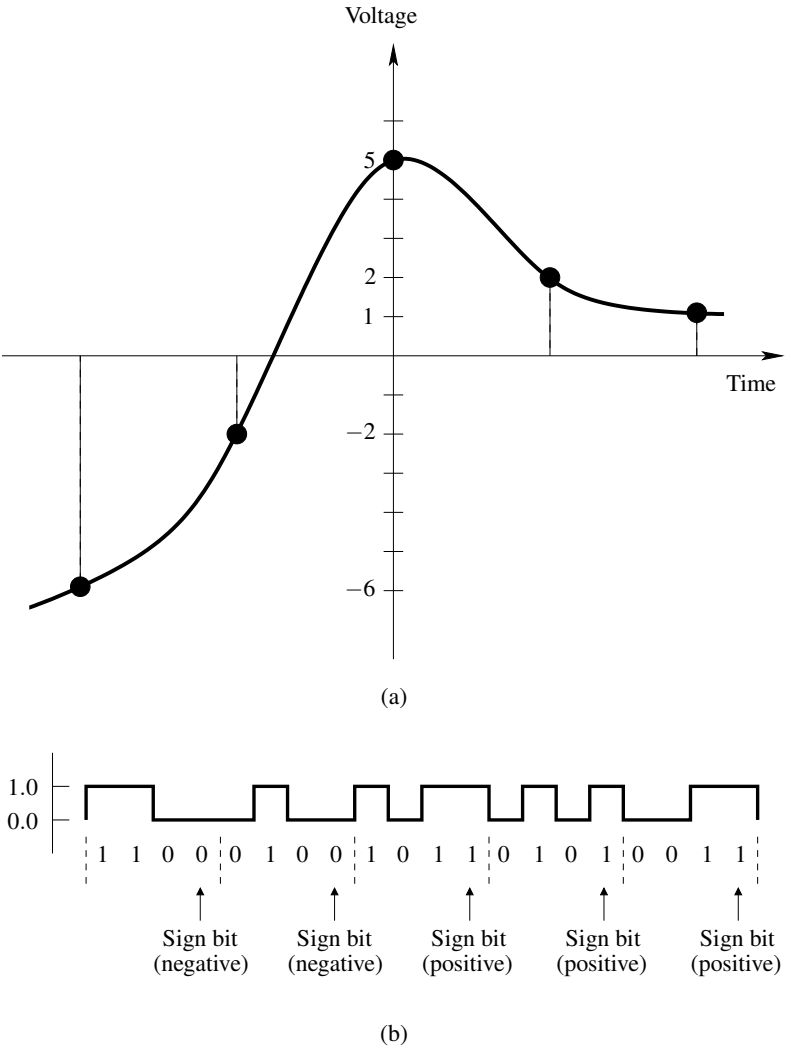

Figure 1.3 (a) Analog waveform. (b) Digital representation.

source, the analog-to-digital conversion process mentioned above also makes the assumption feasible. However, even though specific messages are actually sent, the system designer has no idea in advance which message will be chosen for transmission. We thus need to think of the source as a random (or stochastic) source of information, and ask how we may encode, transmit, and recover the original information.

An information source's output alphabet is defined as the collection of all possible messages. Denote by $\mathcal{U}$ a source alphabet which consists of r messages, say $u_1, u_2, \ldots, u_r$, with probabilities $p_1, p_2, \ldots, p_r$ satisfying

$$p_i \geq 0, \forall i, \text{ and } \sum_{i=1}^{r} p_i = 1. \tag{1.1}$$

Here the notation $\forall$ means "for all" or "for every." We can always represent each message by a sequence of bits, which provides for easier processing by computer systems. For instance, if we toss a fair dice to see which number faces up, only six possible outputs are available with $\mathcal{U} = \{1,2,3,4,5,6\}$ and $p_i = 1/6, \forall\, 1 \leq i \leq 6$. The following shows a straightforward binary description of these messages:

$$1 \leftrightarrow 001,\ 2 \leftrightarrow 010,\ 3 \leftrightarrow 011,\ 4 \leftrightarrow 100,\ 5 \leftrightarrow 101,\ 6 \leftrightarrow 110, \tag{1.2}$$

where each decimal number is encoded as its binary expression. Obviously, there exist many other ways of encoding. For example, consider the two mappings listed below:

$$1 \leftrightarrow 00,\ 2 \leftrightarrow 01,\ 3 \leftrightarrow 100,\ 4 \leftrightarrow 101,\ 5 \leftrightarrow 110,\ 6 \leftrightarrow 111 \tag{1.3}$$

and

$$1 \leftrightarrow 1100,\ 2 \leftrightarrow 1010,\ 3 \leftrightarrow 0110,\ 4 \leftrightarrow 1001,\ 5 \leftrightarrow 0101,\ 6 \leftrightarrow 0011. \tag{1.4}$$

Note that all the messages are one-to-one mapped to the binary sequences, no matter which of the above encoding methods is employed. The original message can always be recovered from the binary sequence.

Given an encoding method, let l_i denote the length of the output sequence, called the *codeword*, corresponding to u_i, $\forall\ 1 \leq i \leq r$. From the viewpoint of source coding for data compression, an optimal encoding should minimize the average length of codewords defined by

$$\mathrm{L_{av}} \triangleq \sum_{i=1}^{r} p_i l_i. \tag{1.5}$$

By (1.5), the average lengths of codewords in (1.2), (1.3), and (1.4) are, respectively,

$$L_{\text{av}}^{(1.2)} = \frac{1}{6}3+\frac{1}{6}3+\frac{1}{6}3+\frac{1}{6}3+\frac{1}{6}3+\frac{1}{6}3 = 3, \tag{1.6}$$

$$L_{\text{av}}^{(1.3)} = \frac{1}{6}2+\frac{1}{6}2+\frac{1}{6}3+\frac{1}{6}3+\frac{1}{6}3+\frac{1}{6}3 = \frac{8}{3} \simeq 2.667, \tag{1.7}$$

$$L_{\text{av}}^{(1.4)} = \frac{1}{6}4+\frac{1}{6}4+\frac{1}{6}4+\frac{1}{6}4+\frac{1}{6}4+\frac{1}{6}4 = 4. \tag{1.8}$$

The encoding method in (1.3) thus provides a more efficient way for the representation of these source messages.

As for channel coding, a good encoding method should be able to protect the source messages against the inevitable noise corruption. Suppose 3 is to be transmitted and an error occurs in the least significant bit (LSB), namely the first bit counted from the right-hand side of the associated codeword. In the case of code (1.2) we now receive 010 instead of 011, and in the case of code (1.3) we receive 101 instead of 100. In both cases, the decoder will retrieve a wrong message (2 and 4, respectively). However, 0111 will be received if 3 is encoded by (1.4). Since 0111 is different from all the codewords in (1.4), we can be aware of the occurrence of an error, i.e. the error is detected, and possible retransmission of the message can be requested. Not just the error in the LSB, but any single error can be detected by this encoding method. The code (1.4) is therefore a better choice from the viewpoint of channel coding.

Typically, for channel coding, the encoding of the message to be transmitted over the channel adds redundancy to combat channel noise. On the other hand, the source encoding usually removes redundancy contained in the message to be compressed. A more detailed discussion on channel and source coding will be shown in Chapters 2 and 3 and in Chapters 4 and 5, respectively.

1.5 Octal and hexadecimal codes

Although the messages of an information source are usually encoded as binary sequences, the binary code is sometimes inconvenient for humans to use. People usually prefer to make a single discrimination among many things. Evidence for this is the size of the common alphabets. For example, the English alphabet has 26 letters, the Chinese "alphabet" (bopomofo) has 37 letters, the Phoenician alphabet has 22 letters, the Greek alphabet has 24 letters, the Russian alphabet 33, the Cyrillic alphabet has 44 letters, etc. Thus, for human use, it is often convenient to group the bits into groups of three at a time and call them the octal code (base 8). This code is given in Table 1.2.

Table 1.2 *Octal code*

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

When using the octal representation, numbers are often enclosed in parentheses with a following subscript 8. For example, the decimal number 81 is written in octal as $(121)_8$ since $81 = \text{“1”} \times 8^2 + \text{“2”} \times 8^1 + \text{“1”} \times 8^0$. The translation from octal to binary is so immediate that there is little trouble in going either way.

The binary digits are sometimes grouped in fours to make the hexadecimal code (Table 1.3). For instance, to translate the binary sequence 101011000111 to the octal form, we first partition these bits into groups of three:

$$\underbrace{101}\,\underbrace{011}\,\underbrace{000}\,\underbrace{111}. \tag{1.9}$$

Each group of bits is then mapped to an octal number by Table 1.2, hence resulting in the octal representation $(5307)_8$. If we partition the bits into groups of four, i.e.

$$\underbrace{1010}\,\underbrace{1100}\,\underbrace{0111}, \tag{1.10}$$

we can get the hexadecimal representation $(\text{AC7})_{16}$ by Table 1.3. Since computers usually work in *bytes*, which are 8 bits each, the hexadecimal code fits into the machine architecture better than the octal code. However, the octal code seems to fit better into the human's psychology. Thus, in practice, neither code has a clear victory over the other.

1.6 Outline of the book

After the introduction of the above main topics, we now have a basis for discussing the material the book is to cover.

Table 1.3 *Hexadecimal code*

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

In general, the error-detecting capability will be accomplished by adding some digits to the message, thus making the message slightly longer. The main problem is to achieve a required protection against the inevitable channel errors without too much cost in adding extra digits. Chapter 2 will look at ways of encoding source symbols so that any errors, up to a given level, may be detected at the terminal end. For a detected error, we might call for a repeat transmission of the message, hoping to get it right the next time.

In contrast to error-detecting codes, error-correcting codes are able to correct some detected errors directly without having to retransmit the message a second time. In Chapter 3, we will discuss two kinds of error-correcting codes, the *repetition code* and the *Hamming code*, as well as their encoding and decoding methods.

In Chapter 4, we consider ways of representing information in an efficient way. The typical example will be an information source that can take on r different possible values. We will represent each of these r values by a string of 0s and 1s with varying length. The question is how to design these strings such that the average length is minimized, but such that we are still able to recover the original data from it. So, in contrast to Chapters 2 and 3, here we try to shorten the codewords.