1

Coding for Reliable Digital Information Transmission and Storage

This chapter presents a simple model of a digital communication (or storage) system and its key function units, which are relevant for reliable information transmission (or storage) over a transmission (or storage) medium subject to noise disturbance (or medium defect). The two function units that play the key role in protection of transmitted (or stored) information against noise (or medium defect) are *encoder* and *decoder*. The function of the encoder is to transform an information sequence into another sequence, called *a coded sequence*, which enables the detection and correction of transmission errors at the receiving end of the system. The transformation of an information sequence to a coded sequence is referred to as *encoding* and the coded sequence is called a *codeword*. At the receiving end of the system, the decoder reproduces the transmitted (or stored) information sequence reliably from the received (or stored) coded sequence. The information recovering process performed by the decoder is referred to as *decoding*. The combination of encoding and decoding for correcting transmission (or storage) errors is referred to as *error-control coding* or *error-correcting coding*.

The objective of this book is to present encoding and decoding methods that are effective in providing reliable information transmission (or storage) and can be practically implemented in a system. In this chapter, two categories of encoding, three decoding rules, and two error-control strategies are introduced. Their objective is to minimize the probability of decoding errors made at the decoder. Also, presented in this introduction chapter are two measurements of performances of a coded communication (or storage) system.

1.1 Introduction

In recent years, there has been an increasing demand for *efficient* and *reliable* digital data-transmission and data-storage systems. This demand has been accelerated by the emergence of large-scale and high-speed data networks for the exchange, processing, and storage of digital information. A major concern of the system designers is the control of transmission or storage errors caused by channel noise (or storage defects) so that *reliable reproduction* of transmitted (or stored) information can be obtained (retrieved).

In 1948, Shannon [1] demonstrated in a landmark paper that, by proper encoding of the information, errors induced by a noisy channel (or storage medium) can be reduced to *any desired level* without sacrificing the rate of information transmission (or storage). Since Shannon's work, a great deal of effort has been expended on the topic of devising efficient encoding and decoding methods for error control in a noisy environment. Recent developments have contributed toward achieving the reliability required by today's high-speed digital communication and high-density storage systems, and the use of error-control coding has become an integral part in the design of these systems.

The transmission and storage of digital information have much in common. They both transfer data from an information source to a destination through a channel.[1] A typical transmission (or storage) system may be represented by the block diagram shown in Fig. 1.1. The information source can be either a person or a machine (e.g., a computer). The source output, which is to be communicated to the destination, can be either a continuous waveform or a sequence of discrete symbols.

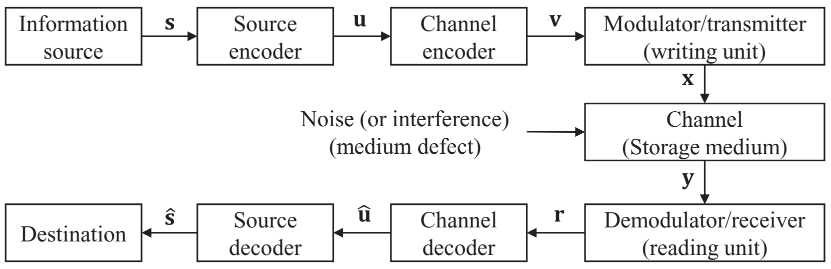

Figure 1.1 Block diagram of a typical data-transmission (or data-storage) system.

The source encoder transforms the source information **s** into a sequence **u** of binary digits (bits) called the *information sequence*. In the case of a continuous

[1] A channel is the physical medium through which the information is conveyed or by which it is stored. Microwave links, optical/coaxial cables, telephone circuits, and magnetic disks are examples of channels. Channels are subjected to various types of distortion, noise, and interference which cause the channel outputs to be different from its inputs. For our purposes, a channel is modeled as a probabilistic device and examples will be presented in later sections.

source, this involves *analog-to-digital* (A/D) conversion. The source encoder is ideally designed so that (1) the number of bits per unit time required to represent the source information $\mathbf{s}$ is minimized, and (2) the source information $\mathbf{s}$ can be reconstructed from the information sequence $\mathbf{u}$ without ambiguity. The subject of source coding is not discussed in this book. For a thorough treatment of this important topic, see References [2–6].

The *channel encoder* transforms the information sequence $\mathbf{u}$ into a discrete encoded sequence $\mathbf{v}$ called a *codeword*. The transformation of an information sequence $\mathbf{u}$ into a codeword $\mathbf{v}$ is called *encoding*. In most instances, $\mathbf{v}$ is also a binary sequence; however, in some applications, information sequences are encoded into nonbinary codewords. The design and implementation of channel encoders to combat the noisy environment in which codewords are transmitted (or stored) is one of the major topics of this book.

Discrete symbols at the output of the encoder are not suitable for transmission over a physical channel or recording on a digital storage medium. The *modulator* (or *writing unit*) transforms each output symbol of the channel encoder into a waveform of duration of T seconds, which is suitable for transmission (or recording). This waveform enters the channel (or storage medium) and is corrupted by noise (or medium defect). The *demodulator* (or *reading unit*) processes each received waveform of duration of T seconds and produces an output that may be discrete (quantized) or continuous (unquantized). The sequence $\mathbf{r}$ of demodulator outputs corresponding to the encoded sequence $\mathbf{v}$ is called the *received sequence*.

The *channel decoder* processes and decodes the received sequence $\mathbf{r}$ into a binary sequence $\hat{\mathbf{u}}$ called the *estimated information sequence*. The decoding strategy is based on the *rules* of channel encoding and the noise characteristics of the channel (or storage medium). Ideally, the estimated information sequence $\hat{\mathbf{u}}$ will be a replica of the transmitted information sequence $\mathbf{u}$, although the channel noise (or storage medium defect) may cause some errors, i.e., $\mathbf{v} \neq \mathbf{r}$. Another major topic of this book is the design and implementation of channel decoders that minimize the probability of decoding errors by detecting and correcting transmission errors in the received sequence $\mathbf{r}$.

The source decoder transforms the estimated information sequence $\hat{\mathbf{u}}$ into an estimate $\hat{\mathbf{s}}$ of the source output $\mathbf{s}$ and delivers this estimate to the destination. When the source is continuous, this involves *digital-to-analog* (D/A) conversion. In a well-designed system, the estimate will be a faithful reproduction of the source information $\mathbf{s}$ except the case when the channel (or storage medium) is very noisy.

To focus attention on the channel encoding and channel decoding, we simplify the system shown in Fig. 1.1 as follows: (1) the information source and source encoder are combined into a digital source with output $\mathbf{u}$; (2) the modulator (or writing unit), the channel (or storage medium), and the demodulator (or reading unit) are combined into a coding channel with input $\mathbf{v}$ and output $\mathbf{r}$; and (3) the source decoder and destination are combined into a digital sink with input $\hat{\mathbf{u}}$. The simplified model of a coded communication (or storage) system is shown in Fig. 1.2.

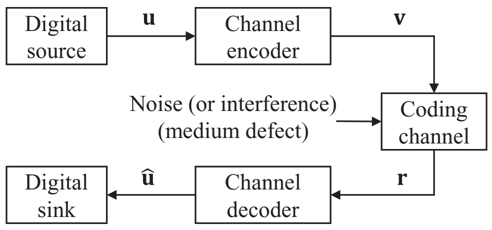

Figure 1.2 A simplified model of a coded system.

The major engineering problem addressed in this book is to design and implement the channel encoder–decoder pair such that (l) information can be transmitted (or recorded) in a noisy environment as fast as possible; (2) reliable reproduction (retrieval) of the information can be obtained at the output of the channel decoder; and (3) the cost of implementing the encoder and decoder falls within acceptable limits.

1.2 Categories of Error-Correcting Codes

The design of the channel encoder involves a mapping from an information sequence $\mathbf{u}$ to a coded sequence $\mathbf{v}$, which enables the error detection and/or correction at the channel decoder. The mapping can be achieved by so-called error-correcting codes or error-control codes.

Error-correcting codes can be classified into two categories, *block* and *convolutional* codes, which are structurally different. The encoder for a block code divides the information sequence into blocks, each consisting of k information bits. A block of k information bits is represented by a binary k-tuple $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ with $u_i = 0$ or 1, $0 \leq i < k$, called a *message*. (In block coding, the symbol $\mathbf{u}$ is used to denote a k-bit message rather than the entire information sequence.) There is a total of 2^k different messages. The encoder encodes each message $\mathbf{u}$ independently into a codeword which is an n-tuple $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ of discrete symbols from a finite alphabet, where n is called the *length* of the codeword. To avoid ambiguity, each message $\mathbf{u}$ is encoded into a *unique* codeword $\mathbf{v}$, i.e., the mapping between a message $\mathbf{u}$ and a codeword $\mathbf{v}$ is *one-to-one*. Because there are 2^k different messages, there are 2^k different codewords at the encoder output. This set of 2^k codewords of length n is said to form an (n, k) *block code*, where n is called the *code length* and k is called the *code dimension*. The ratio $R = k/n$ is called the *code rate* and can be interpreted as the number of information bits entering the channel per transmitted symbol. Because the n-symbol output codeword $\mathbf{v}$ of the encoder depends only on the corresponding k-bit input message $\mathbf{u}$, the encoder is *memoryless*, and thus can be implemented with a combinational logic circuit. (In block coding, the symbol $\mathbf{v}$ is used to denote an n-symbol block rather than the entire encoded sequence.)

In a binary code, each codeword $\mathbf{v}$ is also binary. In this case, we must have $k \leq n$ or $R \leq 1$. When $k < n$, $n - k$ bits are added to each message $\mathbf{u}$ to form a codeword $\mathbf{v}$. The $n - k$ bits added to each message $\mathbf{u}$ by the encoder are called *redundant bits*. These redundant bits carry no new information, and their major function is to provide the code with the capability of detecting and/or correcting errors introduced during the transmission of a codeword, i.e., combating the channel noise (or storage medium defect). How to form these redundant bits to achieve reliable transmission over a noisy channel is the major concern in designing the encoder (or the code). An example of a binary block code with $k = 4$ and $n = 7$ is shown in Table 1.1. This code is a $(7, 4)$ binary block code that is composed of 16 codewords of length 7 for 16 different messages. Each codeword in this code consists of four information bits and three redundant bits. In Chapter 3, it will be shown that this code is capable of correcting a single error, regardless of its location, caused by noise during the transmission of a codeword.

Table 1.1 A binary block code with $k = 4$ and $n = 7$.

Messages (u_0, u_1, u_2, u_3)	Codewords $(v_0, v_1, v_2, v_3, v_4, v_5, v_6)$	Messages (u_0, u_1, u_2, u_3)	Codewords $(v_0, v_1, v_2, v_3, v_4, v_5, v_6)$
(0 0 0 0)	(0 0 0 0 0 0 0)	(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 0)	(1 1 0 1 0 0 0)	(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 0)	(0 1 1 0 1 0 0)	(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 0)	(1 0 1 1 1 0 0)	(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 0)	(1 1 1 0 0 1 0)	(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 0)	(0 0 1 1 0 1 0)	(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 0)	(1 0 0 0 1 1 0)	(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 0)	(0 1 0 1 1 1 0)	(1 1 1 1)	(1 1 1 1 1 1 1)

The encoder for a convolutional code also accepts k-bit blocks of the information sequence $\mathbf{u}$ and produces an encoded sequence (codeword) $\mathbf{v}$ of n-symbol blocks. (In convolutional coding, the symbols $\mathbf{u}$ and $\mathbf{v}$ are used to denote sequences of blocks rather than a single block.) However, each encoded block depends not only on the corresponding k-bit message block at the same time unit, but also on the m *previous message blocks*. Hence, the encoder has a *memory order of* m. The set of encoded sequences produced by a k-bit input and n-bit output encoder of memory order m is called an (n, k, m) *convolutional code*. The ratio $R = k/n$ is called the code rate. Because the encoder contains memory, it must be implemented with a sequential logic circuit. In a binary convolutional code, redundant bits for combating the channel noise are added to the information sequence when $k < n$ or $R < 1$. Typically, k and n are small integers and more redundancy is added by increasing the memory order m of the code while holding k and n, and hence the code rate R is fixed. How to use the memory to achieve reliable transmission over a noisy channel is the major concern in designing the convolutional encoder.

An example of a binary convolutional encoder with $k = 1$, $n = 2$, and $m = 2$ is shown in Fig. 1.3. As an illustration of how codewords are generated, consider the information sequence $\mathbf{u} = (1\ 1\ 0\ 1\ 0\ 0\ 0\ \ldots)$, where the leftmost bit is assumed to enter the encoder first. Before any information bit enters the encoder, the contents of the two shift registers are initialized to zeros. Using the rules of EXCLUSIVE-OR (XOR) and assuming that the multiplexer takes the first encoded bit from the top output and the second bit from the bottom output, it is easy to see that the encoded sequence is $\mathbf{v} = (1\ 1,\ 1\ 0,\ 1\ 0,\ 0\ 0,\ 0\ 1,\ 1\ 1,\ 0\ 0,\ 0\ 0,\ 0\ 0,\ \ldots)$.

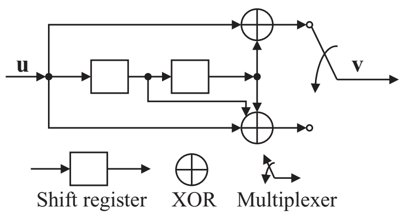

Figure 1.3 A binary convolutional encoder with $k = 1$, $n = 2$, and $m = 2$.

In this book, we only cover block codes, both *classical* and *modern*, because the most current and next generations of communication and storage systems are using and will use block codes in various forms. Readers who are interested in convolutional codes are referred to [7–10].

1.3 Modulation and Demodulation

The modulator in a communication system must select a waveform of duration of T seconds, which is suitable for transmission, for each encoder output symbol. In the case of a binary code, the modulator shall generate one of two signals, $s_0(t)$ for an encoded "0" or $s_1(t)$ for an encoded "1" which are called *carriers*. For a wideband channel, the optimum choices of signals are

$$\begin{aligned} s_0(t) &= \sqrt{\tfrac{2E_s}{T}}\cos(2\pi f_0 t),\ \ 0 \le t \le T, \\ s_1(t) &= \sqrt{\tfrac{2E_s}{T}}\cos(2\pi f_0 t + \pi) = -\sqrt{\tfrac{2E_s}{T}}\cos(2\pi f_0 t),\ \ 0 \le t \le T, \end{aligned} \tag{1.1}$$

where f_0 is a multiple of $1/T$ and E_s is the energy of each signal. This type of modulation is called a *binary phase-shift-keyed* (BPSK) *modulation*, because the phase of the carrier $\cos(2\pi f_0 t)$ is shifted between 0 and π, depending on the encoder output.

A common form of noise disturbance present in any communication system is *additive white Gaussian noise* (AWGN), which commonly includes the ambient heat in the modulator–demodulator hardware and the transmitted medium. A channel with such noise disturbance is called an *AWGN channel* and an AWGN channel with binary input is denoted by BI-AWGN channel. If the transmitted signal over an AWGN channel is $s(t)$ (either $s_0(t)$ or $s_1(t)$), then the received signal is

$$r(t) = s(t) + n(t), \tag{1.2}$$

where $n(t)$ is a *Gaussian random process* with *one-sided power spectral density* (PSD) N_0. Other forms of noise are also present in many systems. For example, in a communication system subject to multipath transmission, the received signal is observed to fade (lose strength) during certain time intervals. This fading can be modeled as a multiplicative noise component on the signal $s(t)$.

The demodulator must produce an output corresponding to the received signal in each T-second interval. This output may be a real number or one of a discrete set of preselected symbols, depending on the demodulator design. An optimum demodulator always includes a matched filter or correlation detector followed by a sampling switch. For BPSK modulation with coherent detection, the sampled output is a real number given by

$$y = \int_0^T r(t)\sqrt{\frac{2E_s}{T}}\cos(2\pi f_0 t)dt. \tag{1.3}$$

The sequence of unquantized demodulator outputs can be passed directly to the channel decoder for processing. In this case, the channel decoder must be capable of handling unquantized inputs, that is, it must be able to process real numbers. A much more common approach to decoding is to quantize the real-number detector output y into one of a finite number N of discrete output symbols. In this case, the channel decoder has discrete inputs and processes discrete values. Most coded communication (or storage) systems use some form of discrete processing.

If the detector output in a given interval depends only on the transmitted signal in that interval, and not on any previous transmission, the channel is said to be *memoryless*. In this case, the combination of an M-ary input modulator, the physical channel, and an N-ary output demodulator can be modeled as a *discrete memoryless channel* (DMC). A DMC is completely described by a set of transition probabilities, $P(j|i)$, $0 \leq i < M$, $0 \leq j < N$, where i represents a modulator input symbol, j represents a demodulator output symbol, and $P(j|i)$ is the probability of receiving j given that i was transmitted. The transition probability $P(j|i)$ can be calculated from a knowledge of the signals used, the probability distribution of the noise, and the output quantization threshold of the demodulator. If $M = 2$, the input of modular is binary, either 0 or 1. Such a DMC is called binary-input DMC (BI-DMC).

As an example, consider a communication system in which (l) binary modulation is used ($M = 2$), (2) the amplitude distribution of the noise is symmetric, and (3) the demodulator output is quantized to $N = 2$ levels. In this case, a particularly simple and practically important channel model, called the *binary symmetric channel* (BSC), results. The block diagram for a BSC is shown in Fig. 1.4(a). The transition probability p is given by

$$p = Q\left(\sqrt{\frac{2E_s}{N_0}}\right), \tag{1.4}$$

where $Q(x)$ is the *complementary error function* of Gaussian statistics given as follows:

$$Q(x) \triangleq \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-y^2/2} dy, \ \ x > 0. \tag{1.5}$$

The transition probability p is simply the BPSK bit-error probability for equally likely signals. An upper bound on $Q(x)$ that will be used later in evaluating the error performance of codes on a BSC is

$$Q(x) \leq \frac{e^{-x^2/2}}{2}, \ \ x > 0. \tag{1.6}$$

It is usually assumed that the power level of transmitter and the message symbol transmission rate are to remain constant. Applying various error-correcting codes, we introduce redundancy to the transmitted vector, which thus decreases the E_s/N_0 at the receiver side. With an error-correcting code of code rate R applied to the system, (1.4) becomes

$$p = Q\left(\sqrt{\frac{2RE_s}{N_0}}\right). \tag{1.7}$$

Figure 1.4(b) shows a transition probability diagram of a BI-DMC.

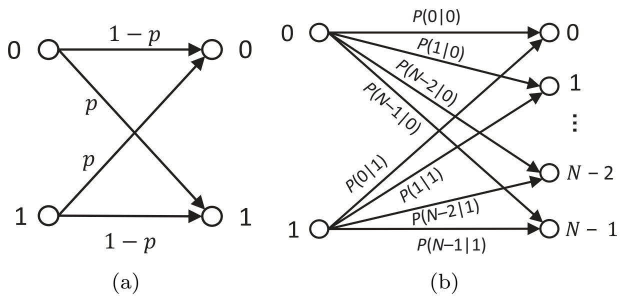

Figure 1.4 Transition probability diagrams: (a) BSC and (b) BI-DMC.

In this book, we focus only on error-correcting codes, their encoding and decoding for communication systems in which binary modulation is used, and the outputs of the demodulator are either quantized or unquantized.

1.4 Hard-Decision and Soft-Decision Decodings

Consider a binary block code of length n. Each codeword in the code is a sequence of 0s and 1s, denoted by $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$. At the transmitter end, each bit v_i of $\mathbf{v}$ is transformed into a specific signal waveform (BPSK signaling is the most widely used one). Therefore, each codeword $\mathbf{v}$ is mapped into a sequence of signal waveforms and transmitted over a noisy channel. At the receiver, the corrupted signal waveforms are demodulated and sampled into a sequence of real numbers $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$, which is the output sequence of the demodulator.

Before entering the channel decoder, if each element of $\mathbf{y}$ is quantized into two levels (0 and 1), we say that a *hard-decision* is made. Then, the input to the channel decoder is a sequence $\mathbf{r} = (r_0, r_1, \ldots, r_{n-1})$ of n bits. Based on this received sequence, the channel decoder performs error detection and/or error correction. Because decoding is based on the hard-decision output sequence of the binary demodulator, the decoding is called *hard-decision decoding*.

If each element of the output sequence $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ of the demodulator is quantized into *more than two* levels, then the input sequence $\mathbf{r} = (r_0, r_1, \ldots, r_{n-1})$ of the decoder is a sequence of discrete symbols from a finite alphabet set. In this case, the channel decoder has discrete inputs and it must process discrete values. If each element of the output sequence $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ of the demodulator is not quantized, the input sequence of the decoder is $\mathbf{y}$, i.e., $\mathbf{r} = \mathbf{y}$. In this case, the channel decoder must process real numbers. Decoding either a received sequence of discrete values or a received sequence of real numbers is referred to as *soft-decision decoding*.

Performing a hard-decision of a received signal at the demodulator, some useful information is lost. As a result, the decoding accuracy is decreased. However, the channel decoder needs only to process binary symbols, 0s and 1s, which results in a low decoding complexity and small decoding delay. Under the context of hard-decision decoding, long codes can be used. For communication systems that require low decoding complexity, low power consumption, and fast decoding process (small decoding delay), the hard-decision decoding is desired. Soft-decision decoding offers *significant* error-correcting improvement in performance compared to hard-decision decoding but requires higher decoding complexity, longer decoding delay, and larger power consumption. As the VLSI circuit design technology advances, the speed of decoding process can be increased significantly and higher decoding complexity can be handled with reasonable expense. Consequently, soft-decision decoding is used in more and more applications, such as 5G, satellite and optical communication and flash memories. Various soft-decision decoding algorithms have been developed, which offer effective tradeoffs between error-correcting performance and decoding complexity.

1.5 Maximum A Posteriori and Maximum Likelihood Decodings

Consider the block diagram of a block-coded communication system on an AWGN channel with finite output quantization as shown in Fig. 1.5. The source output $\mathbf{u}$ represents a k-bit message, the encoder output $\mathbf{v}$ represents an n-symbol codeword, the demodulator output $\mathbf{r}$ represents the corresponding

N-ary received n-tuple, and the decoder output $\hat{\mathbf{u}}$ represents the k-bit estimate of the source output message $\mathbf{u}$.

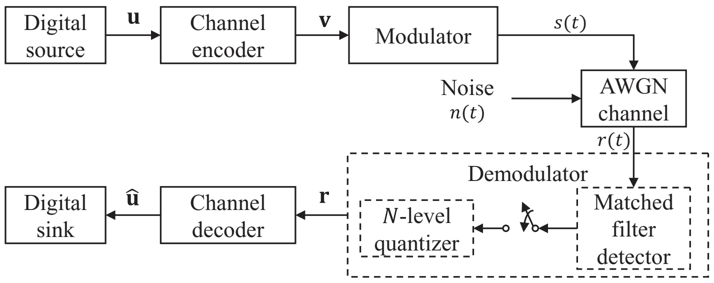

Figure 1.5 A coded communication system with binary-input and N-ary-output discrete memoryless AWGN channel.

The channel decoder must produce an estimate of the message $\mathbf{u}$ based on the received sequence $\mathbf{r}$. Equivalently, because there is a one-to-one correspondence between the message $\mathbf{u}$ and the codeword $\mathbf{v}$, the decoder can produce an estimate $\hat{\mathbf{v}}$ of the codeword $\mathbf{v}$. Clearly, $\hat{\mathbf{u}} = \mathbf{u}$ if and only if $\hat{\mathbf{v}} = \mathbf{v}$. A *decoding rule* is a strategy for choosing an estimate $\hat{\mathbf{v}}$ of the transmitted codeword $\mathbf{v}$ for each possible received sequence $\mathbf{r}$. A decoding error occurs if $\hat{\mathbf{v}} \neq \mathbf{v}$. Given that a sequence $\mathbf{r}$ is received, the error probability of the decoder is defined as

$$P(E|\mathbf{r}) \triangleq P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r}), \tag{1.8}$$

where E denotes the error event $\hat{\mathbf{v}} \neq \mathbf{v}$. The *error probability of the channel decoder* is then given by

$$P(E) = \sum_{\mathbf{r}} P(E|\mathbf{r})P(\mathbf{r}), \tag{1.9}$$

where $P(\mathbf{r})$ is the probability of receiving the sequence $\mathbf{r}$. The probability $P(\mathbf{r})$ is independent of the decoding rule, because $\mathbf{r}$ is produced prior to the decoding. Hence, *an optimum decoding rule*, that is, one that minimizes $P(E)$, must minimize $P(E|\mathbf{r})$ for all $\mathbf{r}$. As minimizing $P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})$ is equivalent to maximizing $P(\hat{\mathbf{v}} = \mathbf{v}|\mathbf{r})$, $P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})$ is minimized for a given $\mathbf{r}$ by choosing $\hat{\mathbf{v}}$ as the codeword $\mathbf{v}$ which maximizes

$$P(\mathbf{v}|\mathbf{r}) = P(\mathbf{r}|\mathbf{v})P(\mathbf{v})/P(\mathbf{r}), \tag{1.10}$$

that is, $\hat{\mathbf{v}}$ is chosen as the codeword which has the *largest a posteriori probability* (APP) $P(\hat{\mathbf{v}}|\mathbf{r})$ (or the *most likely codeword* given that $\mathbf{r}$ is received). This decoding is *optimal* and is called the *maximum a posteriori* (MAP) *decoding*. The decoding process is carried out as follows: (1) compute the APP $P(\mathbf{v}|\mathbf{r})$ for all codewords, $\mathbf{v}$, for a given received sequence $\mathbf{r}$, and (2) decode the received sequence $\mathbf{r}$ into the codeword $\hat{\mathbf{v}}$ which has the largest APP, i.e., $P(\hat{\mathbf{v}}|\mathbf{r}) \geq P(\mathbf{v}^*|\mathbf{r})$ for $\mathbf{v}^* \neq \hat{\mathbf{v}}$.