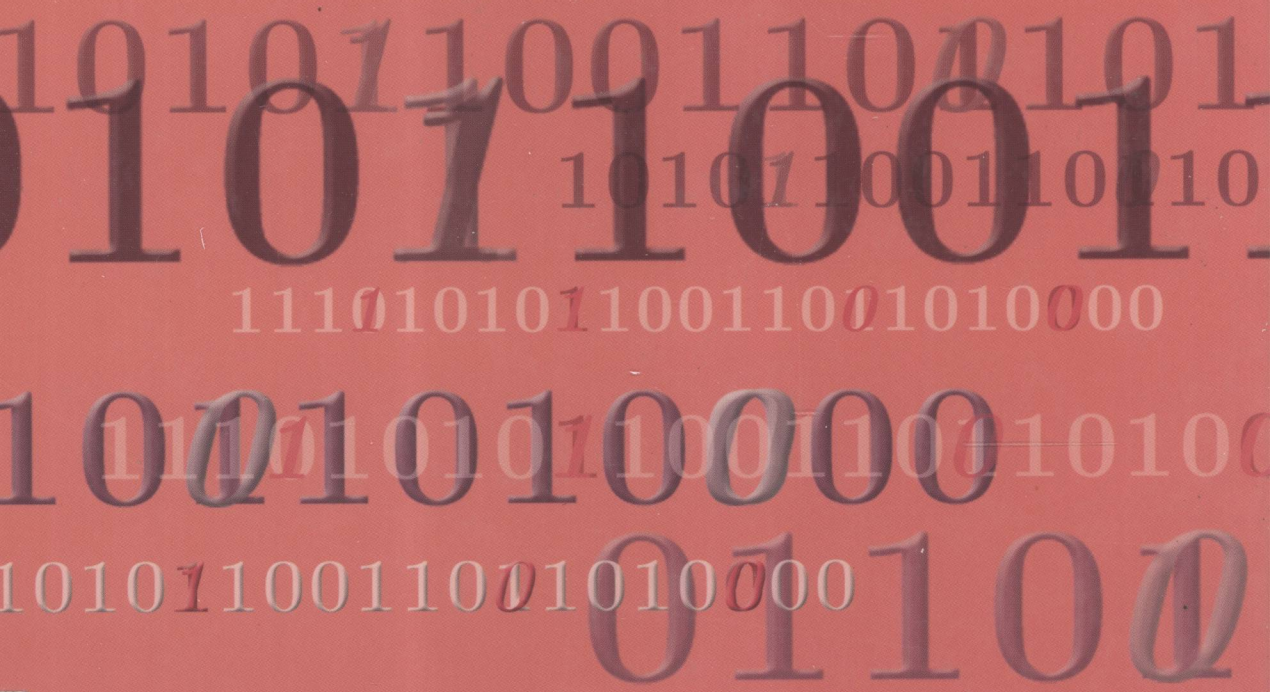
Introduction to Coding Theory

Ron M. Roth



CAMBRIDGE

Introduction to Coding Theory

Ron M. Roth

Technion—Israel Institute of Technology
Haifa, Israel

CAMBRIDGE
UNIVERSITY PRESS

Cambridge University Press
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK
Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521845045

First published 2006

Printed in the United Kingdom at the University Press, Cambridge

A catalog record for this book is available from the British Library

Library of Congress Cataloging in Publication Data

ISBN-13 978-0-521-84504-5 hardback
ISBN-10 0-521-84504-1 hardback

Introduction to Coding Theory

Error-correcting codes constitute one of the key ingredients in achieving the high degree of reliability required in modern data transmission and storage systems. This book introduces the reader to the theoretical foundations of error-correcting codes, with an emphasis on Reed–Solomon codes and their derivative codes.

After reviewing linear codes and finite fields, the author describes Reed–Solomon codes and various decoding algorithms. Cyclic codes are presented, as are MDS codes, graph codes, and codes in the Lee metric. Concatenated, trellis, and convolutional codes are also discussed in detail. Homework exercises introduce additional concepts such as Reed–Muller codes, and burst error correction. The end-of-chapter notes often deal with algorithmic issues, such as the time complexity of computational problems.

While mathematical rigor is maintained, the text is designed to be accessible to a broad readership, including students of computer science, electrical engineering, and mathematics, from senior-undergraduate to graduate level.

This book contains over 100 worked examples and over 340 exercises—many with hints.

RON M. ROTH joined the faculty of Technion—Israel Institute of Technology (Haifa, Israel) in 1988, where he is a Professor of Computer Science and holds the General Yaakov Dori Chair in Engineering. He also held visiting positions at IBM Research Division (San Jose, California) and, since 1993, at Hewlett–Packard Laboratories (Palo Alto, California). He is a Fellow of the Institute of Electrical and Electronics Engineers (IEEE).

Preface

Do ye imagine to reprove words?
Job 6:26

This book has evolved from lecture notes that I have been using for an introductory course on coding theory in the Computer Science Department at Technion. The course deals with the basics of the theory of error-correcting codes, and is intended for students in the graduate and upper-undergraduate levels from Computer Science, Electrical Engineering, and Mathematics. The material of this course is covered by the first eight chapters of this book, excluding Sections 4.4–4.7 and 6.7. Prior knowledge in probability, linear algebra, modern algebra, and discrete mathematics is assumed. On the other hand, all the required material on finite fields is an integral part of the course. The remaining parts of this book can form the basis of a second, advanced-level course.

There are many textbooks on the subject of error-correcting codes, some of which are listed next: Berlekamp [36], Blahut [46], Blake and Mullin [49], Lin and Costello [230], MacWilliams and Sloane [249], McEliece [259], Peterson and Weldon [278], and Pless [280]. These are excellent sources, which served as very useful references when compiling this book. The two volumes of the *Handbook of Coding Theory* [281] form an extensive encyclopedic collection of what is known in the area of coding theory.

One feature that probably distinguishes this book from most other classical textbooks on coding theory is that generalized Reed–Solomon (GRS) codes are treated *before* BCH codes—and even before cyclic codes. The purpose of this was to bring the reader to see, as early as possible, families of codes that cover a wide range of minimum distances. In fact, the cyclic properties of (conventional) Reed–Solomon codes are immaterial for their distance properties and may only obscure the underlying principles of the decoding algorithms of these codes. Furthermore, bit-error-correcting codes, such as binary BCH codes, are found primarily in spatial communication applications, while readers are now increasingly exposed to temporal com-

munication platforms, such as magnetic and optical storage media. And in those applications—including domestic CD and DVD—the use of GRS codes prevails.

Therefore, the treatment of finite fields in this book is split, where the first batch of properties (in Chapter 3) is aimed at laying the basic background on finite fields that is sufficient to define GRS codes and understand their decoding algorithm. A second batch of properties of finite fields is provided in Chapter 7, prior to discussing cyclic codes, and only then is the reader presented with the notions of minimal polynomials and cyclotomic cosets.

Combinatorial bounds on the parameters of codes are treated mainly in Chapter 4. In an introductory course, it would suffice to include only the Singleton and sphere-packing bounds (and possibly the non-asymptotic version of the Gilbert–Varshamov bound). The remaining parts of this chapter contain the asymptotic versions of the combinatorial bounds, yet also cover the information-theoretic bounds, namely, the Shannon Coding Theorem and Converse Coding Theorem for the q-ary symmetric channel. The latter topics may be deferred to an advanced-level course.

GRS codes and alternant codes constitute the center pillar of this book, and a great portion of the text is devoted to their study. These codes are formally introduced in Chapter 5, following brief previews in Sections 3.8 and 4.1. Classical methods for GRS decoding are described in Chapter 6, whereas Chapter 9 is devoted to the list decoding of GRS codes and alternant codes. The performance of these codes as Lee-metric codes is then the main topic of Chapter 10. GRS codes play a significant role also in Chapter 11, which deals with MDS codes.

The last three chapters of the book focus on compound constructions of codes. Concatenated codes and expander-based codes (which are, in a way, two related topics) are presented in Chapters 12 and 13, and an introduction to trellis codes and convolutional codes is given in Chapter 14. This last chapter was included in this book for the sake of an attempt for completeness: knowing that the scope of the book could not possibly allow it to touch all the aspects of trellis codes and convolutional codes, the model of state-dependent coding, which these codes represent, was still too important to be omitted.

Each chapter ends with problems and notes, which occupy on average a significant portion of the chapter. Many of the problems introduce additional concepts that are not covered in text; these include Reed–Muller codes, product codes and array codes, burst error correction, interleaving, the implementation of arithmetic in finite fields, or certain bounds—e.g., the Griesmer and Plotkin bounds. The notes provide pointers to references and further reading. Since the text is intended also for readers who are computer scientists, the notes often contain algorithmic issues, such as the time com-

plexity of certain computational problems that are related to the discussion in the text.

Finally, the Appendix (including the problems therein) contains a short summary of several terms from modern algebra and discrete mathematics, as these terms are frequently used in the book. This appendix is meant merely to recapitulate material, which the reader is assumed to be rather familiar with from prior studies.

I would like to thank the many students and colleagues, whose input on earlier versions of this book greatly helped in improving the presentation. Special thanks are due to Shirley Halevy, Ronny Lempel, Gitit Ruckenstein, and Ido Tal, who taught the course with me at Technion and offered a wide variety of useful ideas while the book was being written. Ido was particularly helpful in detecting and correcting many of the errors in earlier drafts of the text (obviously, the responsibility for all remaining errors is totally mine). I owe thanks to Brian Marcus and Gadiel Seroussi for the good advice that they provided along the way, and to Gadiel, Vitaly Skachek, and the anonymous reviewers for the constructive comments and suggestions. Part of the book was written while I was visiting the Information Theory Research Group at Hewlett–Packard Laboratories in Palo Alto, California. I wish to thank the Labs for their kind hospitality, and the group members in particular for offering a very encouraging and stimulating environment.

Contents

Chapter 1

Introduction

In this chapter, we introduce the model of a communication system, as originally proposed by Claude E. Shannon in 1948. We will then focus on the channel portion of the system and define the concept of a probabilistic channel, along with models of an encoder and a decoder for the channel. As our primary example of a probabilistic channel—here, as well as in subsequent chapters—we will introduce the memoryless q-ary symmetric channel, with the binary case as the prevailing instance used in many practical applications. For $q = 2$ (the binary case), we quote two key results in information theory. The first result is a coding theorem, which states that information through the channel can be transmitted with an arbitrarily small probability of decoding error, as long as the transmission rate is below a quantity referred to as the capacity of the channel. The second result is a converse coding theorem, which states that operating at rates above the capacity necessarily implies unreliable transmission.

In the remaining part of the chapter, we shift to a combinatorial setting and characterize error events that can occur in channels such as the q-ary symmetric channel, and can always be corrected by suitably selected encoders and decoders. We exhibit the trade-off between error correction and error detection: while an error-detecting decoder provides less information to the receiver, it allows us to handle twice as many errors. In this context, we will become acquainted with the erasure channel, in which the decoder has access to partial information about the error events, namely, the location of the symbols that might be in error. We demonstrate that—here as well—such information allows us to double the number of correctable errors.

1.1 Communication systems

Figure 1.1 shows a communication system for transmitting information from a *source* to a *destination* through a *channel*. The communication can be

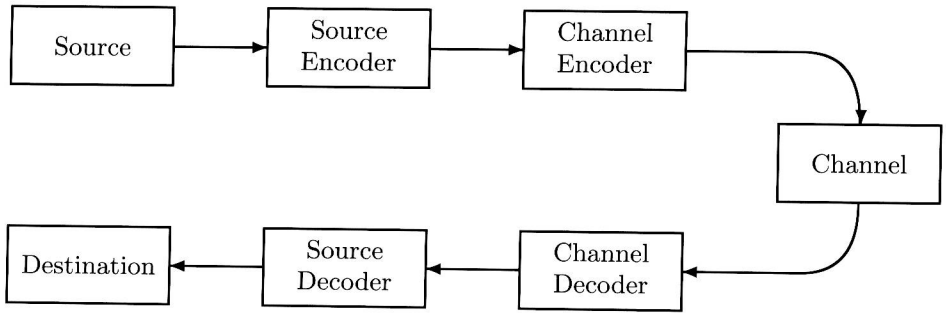


Figure 1.1. Communication system.

either in the space domain (i.e., from one location to another) or in the time domain (i.e., by storing data at one point in time and retrieving it some time later).

The role of source coding is twofold. First, it serves as a translator between the output of the source and the input to the channel. For example, the information that is transmitted from the source to the destination may consist of analog signals, while the channel may expect to receive digital input; in such a case, an analog-to-digital conversion will be required at the encoding stage, and then a back conversion is required at the decoding stage. Secondly, the source encoder may *compress* the output of the source for the purpose of economizing on the length of the transmission; at the other end, the source decoder decompresses the received signal or sequence. Some applications require that the decoder restore the data so that it is identical to the original, in which case we say that the compression is *lossless*. Other applications, such as most audio and image transmissions, allow some (controlled) difference—or distortion—between the original and the restored data, and this flexibility is exploited to achieve higher compression; the compression is then called *lossy*.

Due to physical and engineering limitations, channels are not perfect: their output may differ from their input because of noise or manufacturing defects. Furthermore, sometimes the design requires that the format of the data at the output of the channel (e.g., the set of signals that can be read at the output) should differ from the input format. In addition, there are applications, such as magnetic and optical mass storage media, where certain patterns are not allowed to appear in the recorded (i.e., transmitted) bit stream. The main role of channel coding is to overcome such limitations and to make the channel as transparent as possible from the source and destination points of view. The task of signal translation, which was mentioned earlier in the context of source coding, may be undertaken partially (or wholly) also by the channel encoder and decoder.

1.2 Channel coding

We will concentrate on the channel coding part of Figure 1.1, as shown in Figure 1.2.

Figure 1.2. Channel coding.

Our model of the channel will be that of the *(discrete) probabilistic channel*: a probabilistic channel S is defined as a triple (F, Φ, Prob), where F is a finite *input alphabet*, Φ is a finite *output alphabet*, and Prob is a conditional probability distribution

$$\mathsf{Prob}\{\,\mathbf{y} \text{ received} \mid \mathbf{x} \text{ transmitted}\,\}$$

defined for every pair $(\mathbf{x}, \mathbf{y}) \in F^m \times \Phi^m$, where m ranges over all positive integers and F^m (respectively, Φ^m) denotes the set of all words of length m over F (respectively, over Φ). (We assume here that the channel neither deletes nor inserts symbols; that is, the length of an output word $\mathbf{y}$ always equals the length of the respective input word $\mathbf{x}$.)

The input to the channel encoder is an *information word* (or *message*) $\mathbf{u}$ out of M possible information words (see Figure 1.2). The channel encoder generates a *codeword* $\mathbf{c} \in F^n$ that is input to the channel. The resulting output of the channel is a *received word* $\mathbf{y} \in \Phi^n$, which is fed into the channel decoder. The decoder, in turn, produces a *decoded codeword* $\hat{\mathbf{c}}$ and a *decoded information word* $\hat{\mathbf{u}}$, with the aim of having $\mathbf{c} = \hat{\mathbf{c}}$ and $\mathbf{u} = \hat{\mathbf{u}}$. This implies that the channel encoder needs to be such that the mapping $\mathbf{u} \mapsto \mathbf{c}$ is one-to-one.

The *rate* of the channel encoder is defined as

$$R = \frac{\log_{|F|} M}{n} .$$

If all information words have the same length over F, then this length is given by the numerator, $\log_{|F|} M$, in the expression for R (strictly speaking, we need to round up the numerator in order to obtain that length; however, this integer effect phases out once we aggregate over a sequence of $\ell \to \infty$ transmissions, in which case the number of possible information words becomes M^ℓ and the codeword length is $\ell \cdot n$). Since the mapping of the encoder is one-to-one, we have $R \leq 1$.

The encoder and decoder parts in Figure 1.2 will be the subject of Sections 1.3 and 1.4, respectively. We next present two (related) examples of

probabilistic channels, which are very frequently found in practical applications.

Example 1.1 The memoryless *binary symmetric channel* (in short, BSC) is defined as follows. The input and output alphabets are $F = \Phi = \{0, 1\}$, and for every two binary words $\mathbf{x} = x_1 x_2 \ldots x_m$ and $\mathbf{y} = y_1 y_2 \ldots y_m$ of a given length m,

$$\begin{aligned} &\mathsf{Prob}\{\, \mathbf{y} \text{ received} \mid \mathbf{x} \text{ transmitted} \,\} \\ &\qquad = \prod_{j=1}^{m} \mathsf{Prob}\{\, y_j \text{ received} \mid x_j \text{ transmitted} \,\}\,, \end{aligned} \tag{1.1}$$

where, for every $x, y \in F$,

$$\mathsf{Prob}\{\, y \text{ was received} \mid x \text{ was transmitted} \,\} = \begin{cases} 1-p & \text{if } y = x \\ p & \text{if } y \neq x \end{cases}.$$

The parameter p is a real number in the range $0 \leq p \leq 1$ and is called the *crossover probability* of the channel.

The action of the BSC can be described as flipping each input bit with probability p, independently of the past or the future (the adjective "memoryless" reflects this independence). The channel is called "symmetric" since the probability of the flip is the same regardless of whether the input is 0 or 1. The BSC is commonly represented by a diagram as shown in Figure 1.3. The possible input values appear to the left and the possible output values are shown to the right. The label of a given edge from input x to output y is the conditional probability of receiving the output y given that the input is x.

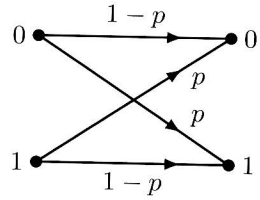


Figure 1.3. Binary symmetric channel.

The cases $p = 0$ and $p = 1$ correspond to reliable communication, whereas $p = \frac{1}{2}$ stands for the case where the output of the channel is statistically independent of its input. □

Example 1.2 The memoryless *q-ary symmetric channel* with crossover probability p is a generalization of the BSC to alphabets $F = \Phi$ of size q. The

conditional probability (1.1) now holds for every two words $\mathbf{x} = x_1 x_2 \ldots x_m$ and $\mathbf{y} = y_1 y_2 \ldots y_m$ over F, where

$$\text{Prob}\{\, y \text{ was received} \mid x \text{ was transmitted} \,\} = \begin{cases} 1-p & \text{if } y = x \\ p/(q-1) & \text{if } y \neq x \end{cases} .$$

(While the term "crossover" is fully justified only in the binary case, we will nevertheless use it for the general q-ary case as well.) □

In the case where the input alphabet F has the same (finite) size as the output alphabet Φ, it will be convenient to assume that $F = \Phi$ and that the elements of F form a finite Abelian group (indeed, for every positive integer q there is an Abelian group of size q, e.g., the ring $\mathbb{Z}_q$ of integer residues modulo q; see Problem A.21 in the Appendix). We then say that the channel is an *additive channel*. Given an additive channel, let $\mathbf{x}$ and $\mathbf{y}$ be input and output words, respectively, both in F^m. The *error word* is defined as the difference $\mathbf{y}-\mathbf{x}$, where the subtraction is taken component by component. The action of the channel can be described as adding (component by component) an error word $\mathbf{e} \in F^m$ to the input word $\mathbf{x}$ to produce the output word $\mathbf{y} = \mathbf{x} + \mathbf{e}$, as shown in Figure 1.4. In general, the distribution of the error word $\mathbf{e}$ may depend on the input $\mathbf{x}$. The q-ary symmetric channel is an example of a channel where $\mathbf{e}$ is statistically independent of $\mathbf{x}$ (in such cases, the term *additive noise* is sometimes used for the error word $\mathbf{e}$).

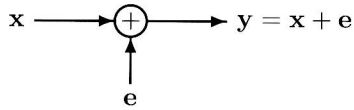


Figure 1.4. Additive channel.

When F is an Abelian group, it contains the zero (or unit) element. The *error locations* are the indexes of the nonzero entries in the error word $\mathbf{e}$. Those entries are referred to as the *error values*.

1.3 Block codes

An (n, M) *(block) code* over a finite alphabet F is a nonempty subset $\mathcal{C}$ of size M of F^n. The parameter n is called the *code length* and M is the *code size*. The *dimension* (or *information length*) of $\mathcal{C}$ is defined by $k = \log_{|F|} M$, and the *rate* of $\mathcal{C}$ is $R = k/n$. The range of the mapping defined by the channel encoder in Figure 1.2 forms an (n, M) code, and this is the context in which the term (n, M) code will be used. The elements of a code are called *codewords*.

In addition to the length and the size of a code, we will be interested in the sequel also in quantifying how much the codewords in the code differ from one another. To this end, we will make use of the following definitions.

Let F be an alphabet. The *Hamming distance* between two words $\mathbf{x}, \mathbf{y} \in F^n$ is the number of coordinates on which $\mathbf{x}$ and $\mathbf{y}$ differ. We denote the Hamming distance by $\mathsf{d}(\mathbf{x}, \mathbf{y})$.

It is easy to verify that the Hamming distance satisfies the following properties of a metric for every three words $\mathbf{x}, \mathbf{y}, \mathbf{z} \in F^n$:

- $\mathsf{d}(\mathbf{x}, \mathbf{y}) \geq 0$, with equality if and only if $\mathbf{x} = \mathbf{y}$.
- Symmetry: $\mathsf{d}(\mathbf{x}, \mathbf{y}) = \mathsf{d}(\mathbf{y}, \mathbf{x})$.
- The triangle inequality: $\mathsf{d}(\mathbf{x}, \mathbf{y}) \leq \mathsf{d}(\mathbf{x}, \mathbf{z}) + \mathsf{d}(\mathbf{z}, \mathbf{y})$.

Let F be an Abelian group. The *Hamming weight* of $\mathbf{e} \in F^n$ is the number of nonzero entries in $\mathbf{e}$. We denote the Hamming weight by $\mathsf{w}(\mathbf{e})$. Notice that for every two words $\mathbf{x}, \mathbf{y} \in F^n$,

$$\mathsf{d}(\mathbf{x}, \mathbf{y}) = \mathsf{w}(\mathbf{y} - \mathbf{x}) \ .$$

Turning now back to block codes, let $\mathcal{C}$ be an (n, M) code over F with $M > 1$. The *minimum distance* of $\mathcal{C}$ is the minimum Hamming distance between any two distinct codewords of $\mathcal{C}$; that is, the minimum distance d is given by

$$d = \min_{\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C} \,:\, \mathbf{c}_1 \neq \mathbf{c}_2} \mathsf{d}(\mathbf{c}_1, \mathbf{c}_2) \ .$$

An (n, M) code with minimum distance d is called an (n, M, d) *code* (when we specify the minimum distance d of an (n, M) code, we implicitly indicate that $M > 1$). We will sometimes use the notation $\mathsf{d}(\mathcal{C})$ for the minimum distance of a given code $\mathcal{C}$.

Example 1.3 The binary $(3, 2, 3)$ *repetition code* is the code

$$\{000, 111\}$$

over $F = \{0, 1\}$. The dimension of the code is $\log_2 2 = 1$ and its rate is $1/3$. □

Example 1.4 The binary $(3, 4, 2)$ *parity code* is the code

$$\{000, 011, 101, 110\}$$

over $F = \{0, 1\}$. The dimension is $\log_2 4 = 2$ and the code rate is $2/3$. □