MODERN CODING THEORY

Preliminary version – April 19, 2007

Modern Coding Theory

BY

T. RICHARDSON AND R. URBANKE

Cambridge University Press

Preliminary version – April 19, 2007

Modern Coding Theory

Library of Congress Catalog Card Number: 00–00000

ISBN 0-000-00000-0

Preliminary version – April 19, 2007

WARNING — USE AT OWN RISK

These notes are work in progress and change daily. The most current version can be found at `http://lthcwww.epfl.ch/mct/index.php`.

Although we are not bold (nor rich) enough to offer a monetary award for typos found, we would greatly appreciate any feedback.

PREFACE

This book is an attempt to summarize the state-of-the-art of *iterative* channel coding. Two other popular names which describe the same area are *probabilistic* coding and *codes on graphs*. Iterative coding was originally devised by Gallager in 1962 in his remarkable thesis and then long forgotten. It was rediscovered by Berrou, Glavieux, and Thitimajshima in 1993 in the form of turbo codes, and then independently in the mid 90's by MacKay and McNeal, Wiberg, as well as Luby et. al. in a form much closer to Gallager's original construction. Iterative techniques have had a strong impact on coding theory and practice and, more generally, on the whole of communications.

The title *Modern Coding Theory* is clearly a hyperbole. After all, there have been several other important recent developments in coding theory. To name just the most prominent one, Sudan's list decoding algorithm for Reed-Solomon codes and its extension to soft-decision decoding have sparked new life in what was considered a fairly mature if not retired subject. So what is our excuse? Iterative methods are influencing a wide range of applications within and beyond communications. The method and theory are highly tied to advances in current computing technology and it is therefore inherently modern. Nevertheless, the font on the book cover is supposed to express the irony that the roots of "modern" coding go back to a time when typewriters ruled the world.

This book is written with several audiences in mind. We hope that it will be a useful text for a course in coding theory. If such a course is dedicated solely to iterative techniques, most necessary material should be contained in this book. If the course covers both classical algebraic coding and iterative topics, this book can be used in conjunction with one of the many excellent books on classical coding. We have excluded virtually all classical material on purpose, except for the most basic definitions. We hope that this book will also be of use to the practitioner in the field who is trying to decide what coding scheme to employ, how a new scheme can be designed, or how an existing system can be improved.

It is important to note that the field of iterative coding has not settled in the same way as classical coding has. There are nearly as many flavors of iterative coding systems – and graphical models to denote them – as there are researchers in the field. We have therefore decided to focus more on techniques to analyze and design such systems rather than specific such instances. In order to present the theory we have chosen Gallager's original ensemble of low-density parity-check codes as a representative example. This ensemble is simple enough that the main results can be presented easily. Once the basic concepts are absorbed, their extension to more general

cases is typically routine and several (but not an exhaustive list) of such extensions are discussed. In particular, we have included a thorough investigation of turbo-codes. Another characteristic of this book is that we spend a considerable number of pages on discussing iterative coding over the binary erasure channel. Why spend so much time on a very specific and limited channel model? It is probably fair to say that what we know about iterative coding we learned first for the binary erasure channel. Due to the special properties of this channel, its basic analysis needs not much more than pen and paper and some knowledge of calculus and probability. All important concepts encountered during the study of the binary erasure channel seem to carry over to general channels, although our ability to prove some of these extensions is in some cases defeated by technical challenges.

There are many possible paths through this book. Our own personal preference is to start with the chapter on factor graphs (Chapter 2). The material covered in this chapter has the special appeal that it unifies many themes of information theory, coding, and communication. Although all three areas trace their origin to Shannon's 1948 paper, they have subsequently diverged to a point where a typical textbook in one area treats each of the other two topics at most in passing. The factor graph approach is a nice way to glue them back together. One and the same technique allows for the computation of capacity, and deals with equalization, modulation and coding on an equal footing. We then recommend to cover the core of the material in Chapter 3 (binary erasure channel) and Chapter 4 (general channels) in a linear fashion.

The remaining material can be read in almost any order according to the preferences of the reader. E.g., at this point it might be rewarding to broaden the view and to go through some of the material on more general channels (Chapter 5). Alternatively, you might be more interested in general ensembles. Chapter 6 discusses turbo codes and Chapter 7 deals with various ensembles and the issues of graph designs. We have not tried to give an exhaustive list of all known iterative codes since this list is growing daily and there is no sign that this growth will stop anytime soon. Rather, we have tried to pick some representative examples.

Chapter 8 gives a brief look at a complementary way of analyzing iterative systems in terms of the expansion of the underlying bipartite graph.

The Appendix contains various chapters on topics which either describe tools for analysis or are simply too technical to fit into the main part. Chapter A takes a look at the encoding problem. Curiously, for iterative schemes the encoding task can be of equal complexity (or even higher) than the decoding task. Appendix B discusses efficient and accurate ways of implementing density evolution. In Appendix C we describe various techniques from probability which are useful in asserting that most elements of a properly chosen ensemble behave "close" to the ensemble average. We take a close look at generating functions in Appendix D. In particular

we discuss how to accurately estimate the coefficients of powers of polynomials – a recurrent theme in this book.

Although we have tried to make the material as accessible as possible, the prerequisites for various portions of the book vary considerably. Some seemingly simple questions need quite sophisticated tools for their answer. A good example is the material related to the weight distribution of LDPC codes and their error floor behavior. In these cases, when the density of equations increases to a painful level, the casual reader is advised not to get discouraged but rather to skip the proofs. Fortunately, in all these cases the subsequent material depends very little on the mathematical details of the proof.

If you are a lecturer and you are giving a beginning graduate level course we recommend that you follow the basic course outlined above but skip some of the less accessible topics. For example, little is lost by simply stating that for "most" ensembles the design rate is equal to the real rate without going through the proof. This is a "natural" statement which is readily accepted. For general binary memoryless symmetric channels one can first focus on Gallager's decoding algorithm A. The analysis for this case is very similar to the one for the binary erasure channel. A subsequent discussion of the belief propagation decoder can skip some of proofs and so avoid a discussion of some of the technical difficulties. If your course is positioned as an advanced graduate level course then most of the material should be accessible to the students.

We started out to write a thin book containing all there is to know about iterative coding. We ended up with a rather thick one and a number of regrettable omissions. To mention just the most important ones: we do not cover the emerging theory of pseudo codewords and their connections to the error floor for general channels. We only scratched the surface of the rich area of interleaver design. Rateless codes deserve a much more prominent role, and there is no discussion of the powerful techniques borrowed from statistical mechanics which have been used successfully in the analysis. Finally, we only mention, but do not discuss source coding by iterative techniques.

But rather than ending with regrets, let us close with the following (slightly modified) quote by Descartes: "[We] hope that posterity will judge [us] kindly, not only as to the things which [we] have explained, but also as to those which [we] have intentionally omitted so as to leave to others the pleasure of discovery." ;-)

HERE GO ACKNOWLEDGMENTS.

T. Richardson
Richardson's Island

R. Urbanke
Lausanne
September 1, 2020

CONTENTS

Chapter 1

INTRODUCTION

§1.1. Why You Should Read This Book

The technology of communication and computing advanced at a breathtaking pace in the 20th century, especially in the second half. A significant part of this advance, especially in communication, began some 60 years ago when Shannon published his seminal paper "A Mathematical Theory of Communication." In that paper Shannon framed and posed a fundamental question: how can we efficiently and reliably transmit information? Shannon also gave a basic answer: coding can do it. Since that time the problem of finding practical coding systems that approach the fundamental limits established by Shannon has been at the heart of information theory and communications. Recently tremendous advances have taken place that bring us quite close to answering this question. Perhaps, at least in a practical sense, the question has been answered. This book is about that answer.

The advance came with a fundamental paradigm shift in the area of coding that took place in the early 90's. In Modern Coding Theory, codes are viewed as large complex systems described by *random sparse graphical models* and encoding as well as decoding are accomplished by efficient *local* algorithms. The local interactions of the code bits are very simple but the overall code is nevertheless complex (and so sufficiently powerful to allow reliable communication) due to the large number of interactions. The idea of random codes is very much in the spirit of Shannon's original formulation. What is new is the sparseness of the description and the local nature of the algorithms.

These are exciting times for coding theorists and practitioners. Despite all the progress made, many fundamental questions are still open. Even if you are not interested in coding itself, however, you might be motivated to read this book. Although the focus of this book is squarely on coding, the larger view holds a much bigger picture. Sparse graphical models and message-passing algorithms, to name just two of the notions that are fundamental to our treatment, play an increasingly important role in many other fields as well. This is not a coincidence. Many of the innovations were brought into the field of coding by physicists or computer scientists. Conversely, the success of modern coding has inspired work in several other fields.

Modern coding will not displace classical coding anytime soon. At any point in time there are hundreds of millions of Reed-Solomon codes working hard to make your life less error prone. This is unlikely to change substantially in the near future.

But modern coding offers an alternative way of solving the communications problem. Most current wireless communications systems have already adopted modern coding.

Technically, our aim is focused on Shannon's classical problem: we want to transmit a *message* across a *noisy channel* so that the *receiver* can determine this message with *high probability* despite the imperfections of the channel. We are interested in *low-complexity* schemes that introduce *little delay* and allow *reliable* transmission close to the ultimate limit, the *Shannon capacity*.

We start with a review of the communications problem (Section 1.2), we cover some classical notions of codes (Sections 1.3, 1.4, 1.5, 1.7, and 1.8), and we review the channel coding theorem (Section 1.6). Section 1.9 gives an outline of the modern approach to coding. Finally, we close in Section 1.10 with a review of the notational conventions and some useful facts used.

§1.2. Communications Problem

Let us review the simplest communications scenario – the *point-to-point* communications problem depicted in Figure 1.1. A *source* (speech, audio, data, . . .) transmits

Figure 1.1: The basic point-to-point communications problem.

via a *noisy channel* (phone line, optical link, wireless, storage medium, . . .) to a *sink*. We are interested in *reliable* transmission, i.e., we want to recreate the transmitted information with as little *distortion* (number of wrong bits, mean squared error distortion, . . .) as possible at the sink.

In his seminal paper in 1948, Shannon formalized the communications problem and showed that the point-to-point problem can always be decomposed into two separate problems as shown in Figure 1.2. First, the *source encoder* transforms the source into a bit stream. Ideally, the source encoder removes all redundancy from the source so that the resulting bit stream uses the smallest possible number of bits while still representing the source with sufficient accuracy. The *channel encoder* then processes the bit stream to add redundancy. This redundancy is carefully chosen so as to combat the noise that is introduced by the channel.

To be mathematically more precise: we model the output of the source as a stochastic process. For example, we might represent text as the output of a Markov chain, describing the local dependency structure of letter sequences. It is the task of the *source encoder* to represent this output as efficiently as possible (using as few bits as possible) given a desired upper bound on the distortion. The *distortion measure* is

chosen by the user and is supposed to reflect the "cost" of deviating from the original source output. If the source emits points in $\mathbb{R}^n$ it might be natural to consider the squared Euclidean distance, whereas if the source emits binary strings a more natural measure might be to count the number of positions in which the source output and the word that can be reconstructed from the encoded source differ. Shannon's *source coding theorem* asserts that, for a given source and distortion measure, there exists a minimum rate $R = R(d)$ (bits per emitted source symbol) which is necessary (and sufficient) to describe this source with distortion not exceeding d. The plot of this rate R as a function of the distortion d is usually called the *rate-distortion* curve. In the second stage an appropriate amount of *redundancy* is added to these source bits to protect them against the errors in the channel. This process is called *channel coding*. Throughout the book we model the channel as a probabilistic mapping and we are typically interested in the *average* performance, where the average is taken over all channel realizations. Shannon's *channel coding theorem* asserts the existence of a maximum rate (bits per channel use) at which information can be transmitted reliably, i.e., with vanishing probability of error, over a given channel. This maximum rate is called the *capacity* of the channel and is denoted by C. At the receiver we first decode the received bits to determine the transmitted information. We then use the decoded bits to reconstruct the source at the receiver. Shannon's *source-channel separation* theorem asserts that the source can be reconstructed with a distortion of at most d at the receiver if $R(d) < \mathrm{C}$, i.e., if the rate required to represent the given source with the allowed distortion is smaller than the capacity of the channel. Conversely, no scheme can do better. One great benefit of the separation theorem is that a communications link can be used for a large variety of sources: one good channel coding solution can be used with virtually any source. Virtually all systems in use today are based on this principle. It is important though to be

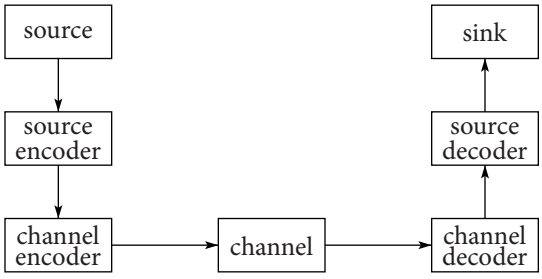

Figure 1.2: The basic point-to-point communications problem in view of the source-channel separation theorem.

aware of the limitations of the source-channel separation theorem. The optimality

is only in terms of the achievable distortion when large blocks of data are encoded together. Joint schemes might be substantially better in terms of complexity or delay. Also, the separation is no longer valid if one looks at multi-user scenarios.

We will not be concerned with the source coding problem or, equivalently, we assume that the source coding problem has been optimally solved. For us, the source emits a sequence of independent identically distributed (iid) bits which are equally likely to be zero or one. Under this assumption, we will see how to accomplish the channel coding problem in an efficient manner for a variety of scenarios.

§1.3. Coding: Trial and Error

How can we transmit information reliably over a noisy channel at a strictly positive rate? At some level we have already given the answer: add redundancy to the message that can be exploited to combat the distortion introduced by the channel. By starting with a special case we want to clarify the key concepts.

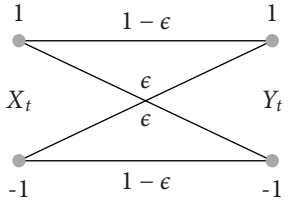

Figure 1.4: BSC(ϵ)

Example 1.3 (Binary Symmetric Channel). Let us introduce the *binary symmetric channel* with *cross-over probability* ϵ depicted in Figure 1.4. We denote it by BSC(ϵ). Both input X_t and output Y_t are elements of $\{\pm 1\}$. A transmitted bit is either received correctly or received *flipped*, the latter occurring with probability ϵ, and different bits are flipped or not flipped independently. Without loss of generality we can assume that $0 < \epsilon < \frac{1}{2}$.

The BSC is the generic model of a binary memoryless channel in which hard decisions are made at the front end of the receiver, i.e., where the received value is quantized to two values. ◇

First Trial: we start by considering *uncoded* transmission over the BSC(ϵ). Thus, we send the source bits across the channel as is, without the insertion of redundant bits. At the receiver we estimate the transmitted bit X based on the observation Y. As we will learn soon in Section 1.5, the optimal decision rule, call it $\hat{x}^{\mathrm{MAP}}(y)$, is to choose that $x \in \{\pm 1\}$ which maximizes $p_{X|Y}(x\,|\,y)$ for the given y. For the given case this estimator reduces to $\hat{x}^{\mathrm{MAP}}(y) = y$. The probability that the estimate differs from the true value, i.e., $\mathrm{P_b} \triangleq \mathrm{P}\{\hat{x}^{\mathrm{MAP}}(Y) \neq X\}$, is equal to ϵ. Since for every

information bit we want to convey we send exactly one bit over the channel we say that this scheme has *rate* 1. We conclude that with uncoded transmission we can achieve a $(\text{rate}, \mathrm{P_b})$-pair of $(1, \epsilon)$.

Second Trial: if this error probability is too high for our application, what transmission strategy can we use to lower it? The simplest strategy is *repetition-coding*. Assume we repeat each bit k times. To keep things simple, assume that k is odd. So if X, the bit to be transmitted, has value x then the input to the $\mathrm{BSC}(\epsilon)$ is the k-tuple $x, \cdots, x$. Denote the k associated observations by $Y_1 \cdots Y_k$. It is not hard to see that the optimal estimator is given by the majority rule

$$\hat{x}^{\mathrm{MAP}}(y_1, \cdots, y_k) = \text{majority of } \{y_1, \cdots, y_k\}.$$

Hence the probability of bit error is given by

$$\mathrm{P_b} = \mathrm{P}\left\{\hat{x}^{\mathrm{MAP}}(Y) \neq X\right\} \overset{k \text{ odd}}{=} \mathrm{P}\left\{\text{at least } \lceil k/2 \rceil \text{ errors occur}\right\} = \sum_{i>k/2} \binom{k}{i} \epsilon^i (1-\epsilon)^{k-i}.$$

Since for every information bit we want to convey we send k bits over the channel we say that such a scheme has rate $\frac{1}{k}$. So with repetition codes we can achieve the $(\text{rate}, \mathrm{P_b})$-pairs $(\frac{1}{k}, \sum_{i>k/2} \binom{k}{i} \epsilon^i (1-\epsilon)^{k-i})$. For $\mathrm{P_b}$ to approach zero we have to choose k larger and larger and as a consequence the rate approaches zero as well.

Can we keep the rate positive and still make the error probability go to zero?

§1.4. Codes and Ensembles

Information is inherently discrete. It is natural and convenient to use *finite* fields to represent it. The most important instance for us is the *binary field* $\mathbb{F}_2$, consisting of $\{0, 1\}$ with mod-2 addition and mod-2 multiplication ($0 + 0 = 1 + 1 = 0$; $0 + 1 = 1$; $0 \cdot 0 = 1 \cdot 0 = 0$; $1 \cdot 1 = 1$). In words, if we use $\mathbb{F}_2$ then we represent information in terms of (sequences of) *bits*, a natural representation and convenient for the purpose of processing. If you are not familiar with finite fields, very little is lost if you replace any mention of a generic finite field $\mathbb{F}$ with $\mathbb{F}_2$. We write $|\mathbb{F}|$ to indicate the number of elements of the finite field $\mathbb{F}$, e.g., $|\mathbb{F}_2| = 2$. Why do we choose finite *fields*? As we will see, this allows us to make use of algebraic operations in both the encoding as well as the decoding, significantly reducing the complexity.

Definition 1.5 (Code). A *code* C of *length* n and *cardinality* M over a field $\mathbb{F}$ is a subset of $\mathbb{F}^n$ with M elements, i.e.,

$$C(n, M) \triangleq \{x^{[1]}, \cdots, x^{[M]}\},\ x^{[m]} \in \mathbb{F}^n, 1 \leq m \leq M.$$

The elements of the code are called *codewords*. The parameter n is called the *block-length*. ▽

EXAMPLE 1.6 (REPETITION CODE). Let $\mathbb{F} = \mathbb{F}_2$. The binary *repetition code* is defined as $C(n = 3, M = 2) = \{000, 111\}$. ◇

In the above example we have introduced *binary* codes, i.e., codes whose components are elements of $\mathbb{F}_2 = \{0, 1\}$. Some times it is more convenient to think of the two field elements as $\{\pm 1\}$ instead (see, e.g., the definition of the BSC in Example 1.3). The standard mapping is $0 \leftrightarrow 1$ and $1 \leftrightarrow -1$. It is convenient to use both notatations. We will freely and frequently switch and, with some abuse of notation, we will make no distinction between these two cases and talk simply about binary codes and $\mathbb{F}_2$ even if the components take values in $\{\pm 1\}$.

DEFINITION 1.7 (RATE). The *rate* (measured as information symbols per transmitted symbol) of a code $C(n, M)$ is $r \triangleq \frac{1}{n} \log_{|\mathbb{F}|} M$. ▽

EXAMPLE 1.8 (REPETITION CODE). Let $\mathbb{F} = \mathbb{F}_2$. We have $r(C(3, 2)) = \frac{1}{3} \log_2 2 = \frac{1}{3}$. It takes 3 channel symbols to transmit one information symbol. ◇

The following two definitions will play a role only much later in the book. But it is convenient to collect them here for reference.

DEFINITION 1.9 (SUPPORT SET). The *support set* of a codeword $x \in C$ is the set of locations $i \in [n] = \{1, \cdots, n\}$ such that $x_i \neq 0$. ▽

DEFINITION 1.10 (MINIMAL CODEWORDS). Consider a *binary* code C, i.e., a code over $\mathbb{F}_2$. We say that a codeword $x \in C$ is *minimal* if its support set does not contain the support set of any other (non-zero) codeword. ▽

The Hamming distance introduced in the following definition and the derived minimum distance of a code (see Definition 1.12) are *the* central characters in all of classical coding. For us they only play a minor role. This is probably one of the most distinguishing factors between classical and modern coding.

DEFINITION 1.11 (HAMMING WEIGHT AND HAMMING DISTANCE). Let $u, v \in \mathbb{F}^n$. The *Hamming weight* of a word u, which we denote by $w(u)$, is equal to the number of non-zero symbols in u, i.e., the cardinality of the support set. The *Hamming distance* of a pair (u, v), which we denote by $d(u, v)$, is the number of positions in which u differs from v. We have $d(u, v) = d(u - v, 0) = w(u - v)$. Further, $d(u, v) = d(v, u)$ and $d(u, v) \geq 0$, with equality if and only if $u = v$. Also, $d(\cdot, \cdot)$ satisfies the *triangle inequality*

$$d(u, v) \leq d(u, t) + d(t, v),$$

for any triple $u, v, t \in \mathbb{F}^n$. In words, $d(\cdot, \cdot)$ is a true *distance* in the mathematical sense. ▽

DEFINITION 1.12 (MINIMUM DISTANCE OF A CODE). Let C be a code. Its *minimum distance* $d(C)$ is defined as

$$d(C) \triangleq \min\{d(u,v) : u, v \in C, u \neq v\}. \qquad \nabla$$

Let $x \in \mathbb{F}^n$ and $t \in \mathbb{N}$. A *sphere* of radius t centered at the point x is the set of all points in $\mathbb{F}^n$ that have distance at most t from x. If, for a code C of minimum distance d, we place spheres of radius $t \triangleq \left\lfloor \frac{d-1}{2} \right\rfloor$ around each codeword, then these spheres are disjoint. This follows from the triangle inequality. Further, by definition of d, t is the largest such radius.

The radius t has an important operational meaning that explains why much of classical coding is centered on the construction of codes with large minimum distance. To be concrete, consider the binary case. Assume we use a code $C(n, M, d)$ (i.e., a code with M codewords of length n and minimum distance d) for transmission over a BSC and assume that we employ a *bounded distance* decoder with decoding radius t, $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$. More precisely, given y the decoder chooses $\hat{x}^{\mathrm{BD}}(y)$ defined by

$$\hat{x}^{\mathrm{BD}}(y) \triangleq \begin{cases} x \in C, & \text{if } d(x,y) \leq t, \\ \text{error}, & \text{if no such } x \text{ exists}, \end{cases}$$

where by "error" the decoder declares that it is unable to decode. Note that for a given y there can be at most one $x \in C$ so that $d(x,y) \leq t$: otherwise, if for both $x \in C$ and $x' \in C$ we have $d(x,y) \leq t$ and $d(x',y) \leq t$ then by the triangle inequality $d(x,x') \leq d(x,y) + d(x',y) \leq 2t \leq d-1$, a contradiction.

Such a combination corrects all errors of weight t or less. Therefore, a large t implies a large resilience against channel errors.

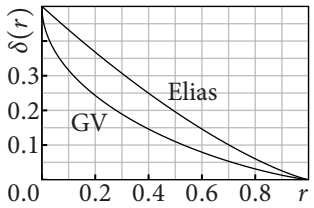

Figure 1.13: Upper and lower bound on $\delta^*(r)$.

How large can d (and hence t) be made in the binary case? Let $\delta \triangleq d/n$ denote the *normalized* distance and consider for a fixed rate r, $0 < r < 1$,

$$\delta^*(r) \triangleq \limsup_{n\to\infty} \max\left\{ \frac{d(C)}{n} : C \in \mathcal{C}\left(n, 2^{\lfloor nr \rfloor}\right) \right\},$$

where $\mathcal{C}\left(n, 2^{\lfloor nr \rfloor}\right)$ denotes the set of all binary block codes of length n containing at least $2^{\lfloor nr \rfloor}$ codewords. Problem 1.14 discusses the asymptotic *Gilbert-Varshamov* bound

$$h_2^{-1}(1-r) \leq \delta^*(r),$$

where $h_2(x) \triangleq -x\log_2 x - (1-x)\log_2(1-x)$ is the *binary entropy function* and where for $y \in [0,1]$, $h_2^{-1}(y)$ is the unique element $x \in [0, \frac{1}{2}]$ such that $h_2(x) = y$. There exists a related upper bound due to *Elias* which states that

$$\delta^*(r) \leq 2h_2^{-1}(1-r)(1-h_2^{-1}(1-r)). \tag{1.14}$$

Both bounds are illustrated in Figure 1.13.

We can now answer the question posed at the end of the previous section. For a fixed channel BSC(ϵ) pick a rate r such that $\delta^*(r) > 2\epsilon + \omega$, where ω is some arbitrarily small but strictly positive quantity. We see from the Gilbert-Varshamov bound that such a strictly positive r and ω exist as long as $\epsilon < 1/4$. Further, in this case we can find a code of rate r of arbitrarily large blocklength n which has a relative minimum distance at least $\delta = 2\epsilon + \omega$. By the Chebyshev inequality (see Lemma C.3 on page 488), for every desired strictly positive probability P there exists a positive constant c such that the number of errors which one encounters in a block of length n is at most $n\epsilon + c\sqrt{n}$ with probability P. Assume that we employ a bounded distance decoder. If we choose n sufficiently large so that $n\epsilon + c\sqrt{n} < \delta n/2 = n\epsilon + n\omega/2$, then the bounded distance decoder succeeds with probability at least P. Since P can be chosen arbitrarily close to 1 we see that there exist codes which allow transmission at a positive rate with arbitrarily small probability of error.

Constructing provably good codes is difficult. A standard approach to show the *existence* of good codes is the probabilistic method: an *ensemble* $\mathcal{C}$ of codes is constructed using some random process and one proves that good codes occur with positive probability within this ensemble. Often the probability is close to 1 – almost all codes are good. This approach, used already by Shannon in his 1948 landmark paper, simplifies the task enormously (at the cost of a less useful result).

Definition 1.15 (Shannon's Random Ensemble). Let the field $\mathbb{F}$ be fixed. Consider the following ensemble $\mathcal{C}(n, M)$ of codes of length n and cardinality M. There are nM degrees of freedom in choosing a code, one degree of freedom for each component of each codeword. The ensemble consists of all $|\mathbb{F}|^{nM}$ possible codes of length n and cardinality M. We endow this set with a uniform probability distribution. To sample from this ensemble proceed as follows. Pick the codewords $x^{[1]}, \cdots, x^{[M]}$ randomly by letting each component $x_i^{[m]}$ be an independently and uniformly chosen element of $\mathbb{F}$. ▽

We will see that such a code is likely to be "good" for many channels.

§1.5. MAP and ML Decoding

Assume we transmit over a channel with input $\mathbb{F}$ and output space $\mathcal{Y}$ using a code $C(n,M) = \{x^{[1]}, \cdots, x^{[M]}\}$. Let the channel be specified by its transition probability $p_{Y|X}(y|x)$. The transmitter chooses the codeword $X \in C(n,M)$ with probability $p_X(x)$. (In communications the idea is that the transmitter wants to transmit one of M messages and uses one codeword for each possible message.) This codeword is then transmitted over the channel. Let Y denote the observation at the output of the channel. To what codeword should Y be decoded? If we decode Y to $\hat{x}(Y) \in C$, then the probability that we have made an error is $1 - p_{X|Y}(\hat{x}(Y)|y)$. Thus, to minimize the probability of error we should choose $\hat{x}(Y)$ to maximize $p_{X|Y}(\hat{x}(Y)|y)$. The *maximum a posteriori* (MAP) decoding rule reads

$$\begin{aligned}\hat{x}^{\mathrm{MAP}}(y) &\triangleq \operatorname{argmax}_{x\in C} p_{X|Y}(x|y)\\ \text{by Bayes's rule} \qquad &= \operatorname{argmax}_{x\in C} p_{Y|X}(y|x)\frac{p_X(x)}{p_Y(y)}\\ &= \operatorname{argmax}_{x\in C} p_{Y|X}(y|x)p_X(x).\end{aligned}$$

Ties can be broken in some arbitrary manner without affecting the error probability. As we indicated, this estimator minimizes the probability of (block) error $\mathrm{P_B} \triangleq \mathrm{P}\{\hat{x}^{\mathrm{MAP}}(Y) \neq X\}$. If all codewords are equally likely, i.e., if p_X is uniform, then

$$\hat{x}^{\mathrm{MAP}}(y) = \operatorname{argmax}_{x\in C} p_{Y|X}(y|x)p_X(x) = \operatorname{argmax}_{x\in C} p_{Y|X}(y|x) \triangleq \hat{x}^{\mathrm{ML}}(y),$$

where the right hand side represents the decoding rule of the *maximum likelihood* (ML) decoder. In words, for a uniform prior p_X the MAP and the ML decoders are equivalent.

§1.6. Channel Coding Theorem

We have already seen that transmission at a strictly positive rate and an arbitrarily small probability of error is possible. What is the *largest* rate at which we can achieve a vanishing probability of error? Let us now investigate this question for transmission over the $\mathrm{BSC}(\epsilon)$.

We are interested in the scenario depicted in Figure 1.16. For a given binary code $C(n,M)$ the transmitter chooses with uniform probability a codeword $X \in C(n,M)$ and transmits this codeword over the channel $\mathrm{BSC}(\epsilon)$. The output of the channel is denoted by Y. At the receiver the decoder estimates the transmitted codeword given the observation Y using the MAP rule $\hat{x}^{\mathrm{MAP}}(y)$. How small can we make the incurred *block error probability* $\mathrm{P_B^{MAP}}(C,\epsilon) \triangleq \mathrm{P}\left\{\hat{X}^{\mathrm{MAP}} \neq X\right\}$ for given parameters n and M? Let $\hat{\mathrm{P}}_{\mathrm{B}}^{\mathrm{MAP}}(n,M,\epsilon)$ be the minimum of $\mathrm{P_B^{MAP}}(C,\epsilon)$ over all choices of $C \in \mathcal{C}(n,M)$.

Figure 1.16: Transmission over the BSC(ϵ).

THEOREM 1.17 (SHANNON'S CHANNEL CODING THEOREM). If $0 < r < 1 - h_2(\epsilon)$ then $\hat{\mathrm{P}}_{\mathrm{B}}^{\mathrm{MAP}}(n, 2^{\lfloor rn \rfloor}, \epsilon) \stackrel{n\to\infty}{\longrightarrow} 0$.

Proof. Pick a code C from $\mathcal{C}(n, 2^{\lfloor rn \rfloor})$, the random ensemble introduced in Definition 1.15. Since the MAP decoder is hard to analyze we use the following suboptimal decoder. For some fixed Δ, $\Delta > 0$, define $\rho = n\epsilon + \sqrt{2n\epsilon(1-\epsilon)/\Delta}$. If $x^{[m]}$ is the only codeword such that $d(y, x^{[m]}) \le \rho$ then decode y as $x^{[m]}$ – otherwise declare an error.

For $u, v \in \{\pm 1\}^n$ let

$$f(u,v) \triangleq \begin{cases} 0, & \text{if } d(u,v) > \rho, \\ 1, & \text{if } d(u,v) \le \rho, \end{cases}$$

and define

$$g^{[m]}(y) \triangleq 1 - f(x^{[m]}, y) + \sum_{m' \neq m} f(x^{[m']}, y).$$

Note that $g^{[m]}(y)$ equals zero if $x^{[m]}$ is the only codeword such that $d(y, x^{[m]}) \le \rho$ and that it is at least one otherwise. Let $\mathrm{P}_{\mathrm{B}}^{[m]}$ denote the conditional block error probability assuming that $X = x^{[m]}$, i.e., $\mathrm{P}_{\mathrm{B}}^{[m]} = \mathrm{P}\{\hat{x}(Y) \neq X \,|\, X = x^{[m]}\}$.

$$\begin{aligned}
\mathrm{P}_{\mathrm{B}}^{[m]}(C,\epsilon) &= \sum_{y: g^{[m]}(y) \ge 1} p_{Y|X^{[m]}}(y \,|\, x^{[m]}) \le \sum_{y \in \{\pm 1\}^n} p_{Y|X^{[m]}}(y \,|\, x^{[m]}) g^{[m]}(y) \\
&= \sum_{y \in \{\pm 1\}^n} p_{Y|X^{[m]}}(y \,|\, x^{[m]})[1 - f(x^{[m]}, y)] \\
&\quad + \sum_{y \in \{\pm 1\}^n} \sum_{m' \neq m} p_{Y|X^{[m]}}(y \,|\, x^{[m]}) f(x^{[m']}, y) \\
&= \mathrm{P}\left\{ d(Y, x^{[m]}) > \rho \,|\, X^{[m]} = x^{[m]} \right\} \\
&\quad + \sum_{y \in \{\pm 1\}^n} \sum_{m' \neq m} p_{Y|X^{[m]}}(y \,|\, x^{[m]}) f(x^{[m']}, y).
\end{aligned}$$

Note that $d(y, x^{[m]}) = w(y + x^{[m]})$, where $y + x^{[m]}$ is the vector of channel errors (recall that over $\mathbb{F}_2$ addition and subtraction are the same). It follows that $d(Y, x^{[m]})$ is the sum of n independent Bernoulli random variables, call it Z. Then

Z is a random variable with mean $n\epsilon$ and variance $n\epsilon(1-\epsilon)$. Recall from above that $\rho = n\epsilon + \sqrt{2n\epsilon(1-\epsilon)/\Delta}$. Therefore, from Chebyshev's inequality (see Lemma C.3 on page 488) we get

$$\mathrm{P}\left\{|Z - n\epsilon| \geq \sqrt{2n\epsilon(1-\epsilon)/\Delta}\right\} \leq \frac{n\epsilon(1-\epsilon)\Delta}{2n\epsilon(1-\epsilon)} = \frac{\Delta}{2}.$$

We can write $\mathrm{P_B}(C,\epsilon)$ as

$$\frac{1}{M}\sum_{m=1}^{M} \mathrm{P_B^{[m]}}(C,\epsilon) \leq \frac{\Delta}{2} + \frac{1}{M}\sum_{m=1}^{M} \sum_{y\in\{\pm 1\}^n} \sum_{m'\neq m} p_{Y\,|\,X^{[m]}}(y\,|\,x^{[m]}) f(x^{[m']}, y).$$

Let $\mathbb{E}_{\mathcal{C}(n,M)}[\cdot]$ denote the expectation with respect to the ensemble $\mathcal{C}(n,M)$. We conclude that

$$\begin{aligned}
\hat{\mathrm{P}}_{\mathrm{B}}(n,M,\epsilon) &\leq \mathbb{E}_{\mathcal{C}(n,M)}\big[\mathrm{P_B}(C,\epsilon)\big] \\
&\leq \frac{\Delta}{2} + \frac{1}{M}\sum_{m=1}^{M} \sum_{y\in\{\pm 1\}^n} \sum_{m'\neq m} \mathbb{E}\big[p_{Y\,|\,X^{[m]}}(y\,|\,X^{[m]}) f(X^{[m']}, y)\big] \\
&\overset{(a)}{=} \frac{\Delta}{2} + \frac{1}{M}\sum_{m=1}^{M} \sum_{y\in\{\pm 1\}^n} \sum_{m'\neq m} \mathbb{E}\big[p_{Y\,|\,X^{[m]}}(y\,|\,X^{[m]})\big]\,\mathbb{E}\big[f(X^{[m']}, y)\big] \\
&= \frac{\Delta}{2} + \frac{1}{M}\sum_{m=1}^{M} \sum_{y\in\{\pm 1\}^n} \sum_{m'\neq m} \mathbb{E}\big[p_{Y\,|\,X^{[m]}}(y\,|\,X^{[m]})\big]\frac{\sum_{k=0}^{\lfloor\rho\rfloor}\binom{n}{k}}{2^n} \\
&= \frac{\Delta}{2} + (M-1)\frac{\sum_{k=0}^{\lfloor\rho\rfloor}\binom{n}{k}}{2^n},
\end{aligned}$$

where in step (a) we used the fact that if we consider for $m' \neq m$ the two associated codewords $X^{[m]}$ and $X^{[m']}$ as random variables then they are by construction *(pairwise) independent*. If we now use the estimate $\sum_{k=0}^{m}\binom{n}{k} \leq 2^{nh_2(m/n)}$, which is valid for $m \leq n/2$ (see (1.59) and Problem 1.23), then as $\rho \leq n/2$ for sufficiently large n

$$\begin{aligned}
\hat{\mathrm{P}}_{\mathrm{B}}(n,M,\epsilon) &\leq \frac{\Delta}{2} + (M-1)2^{-n\left(1-h_2\left(\epsilon+\sqrt{\frac{2\epsilon(1-\epsilon)}{n\Delta}}\right)\right)} \\
&\leq \frac{\Delta}{2} + 2^{nr}2^{-n\left(1-h_2\left(\epsilon+\sqrt{\frac{2\epsilon(1-\epsilon)}{n\Delta}}\right)\right)} \\
&= \frac{\Delta}{2} + 2^{-n\left(1-h_2\left(\epsilon+\sqrt{\frac{2\epsilon(1-\epsilon)}{n\Delta}}\right)-r\right)} \\
&\leq \Delta \quad \text{for } n \text{ large enough if } r < 1 - h_2(\epsilon).
\end{aligned}$$

The proof is complete if we observe that this upper bound is valid for any $\Delta > 0$. □

The above proof shows that there exist codes in $\mathcal{C}\left(n, 2^{\lfloor nr \rfloor}\right)$ which permit reliable transmission over the BSC(ϵ) up to a rate of $1 - h_2(\epsilon)$ bits per channel use. Actually, a much stronger statement is true, namely *almost any* code in the above ensemble can be used for transmission at vanishing probabilities of error.

Although we do not prove this here, the converse is true as well: any attempt to transmit at a rate higher than $1 - h_2(\epsilon)$ must result in error probabilities bounded away from zero. Indeed, one can show that the block error probability $\mathrm{P_B}$ must tend to one for any sequence of codes of increasing blocklength and rate strictly above $1 - h_2(\epsilon)$. Therefore, $1 - h_2(\epsilon)$ is a *threshold* value, separating what is achievable from what is not. It is called the *Shannon capacity* of the BSC(ϵ) and we denote it by $\mathrm{C_{BSC}}(\epsilon) = 1 - h_2(\epsilon)$.

As mentioned several times before, the minimum distance plays a central role in all of classical coding. The paradigm of classical coding can be summarized as follows: (i) find a code with a large minimum distance and a strong algebraic structure; (ii) devise a decoding algorithm which exploits the algebraic structure in order to accomplish bounded distance decoding efficiently (see page 1.4); This philosophy works well if we transmit at a rate which is bounded away from capacity. But, as the next example shows, we can not hope to achieve capacity in this way.

Example 1.18 (Bounded Distance Decoder is Not Sufficient). Consider a code of rate r, $r \in (0, 1)$. By the Elias bound (1.14) the normalized minimum distance $\delta(r)$ is upper bounded by $\delta(r) \leq 2h_2^{-1}(1-r)(1-h_2^{-1}(1-r))$, so that

$$h_2^{-1}(1-r) \geq \frac{1}{2}\left(1 - \sqrt{1 - 2\delta(r)}\right),$$

for $\delta(r) \in (0, 1/2)$. From this we deduce the weaker bound $h_2^{-1}(1-r) > \frac{1}{2}\delta(r) + \left(\frac{1}{2}\delta(r)\right)^2$ which is easier to handle. If we transmit over the BSC(ϵ) then the expected number of errors in a block of length n is $n\epsilon$. Further, for large n with high probability the actual number of errors is within $O(\sqrt{n})$ of this expected number (see the previous proof on page 11). Therefore, if we employ a bounded distance decoder we need $\frac{1}{2}\delta(r) \geq \epsilon$. If we combine this with the previous bound we get $h_2^{-1}(1-r) > \epsilon + \epsilon^2$. This is not possible if $\epsilon > \frac{\sqrt{3}-1}{2}$ since then the right hand side exceeds $1/2$. It follows that such a bounded distance decoder cannot be used for reliable transmission over a BSC(ϵ) if $\epsilon \in (\frac{\sqrt{3}-1}{2}, \frac{1}{2})$. And for $\epsilon \in (0, \frac{\sqrt{3}-1}{2})$ we conclude that $r < 1 - h_2(\epsilon+\epsilon^2) < 1 - h_2(\epsilon) = \mathrm{C_{BSC}}(\epsilon)$, i.e., capacity cannot be achieved either. $\diamond$

§1.7. Linear Codes and Their Complexity

By our remarks above, almost any code in $\mathcal{C}\left(n, 2^{\lfloor nr \rfloor}\right)$ is suitable for reliable transmission at rates close to Shannon capacity at low error probability provided only

that the length n is sufficiently large (we have limited the proof of the channel coding theorem to the BSC but this theorem applies in a much wider setting). So why not declare the coding problem solved and stop here? The answer is that Shannon's theorem does not take into account the *description*, the *encoding* and the *decoding complexities*.

First note that, without further restriction on the structure, already the description of a particular code quickly becomes impractical as n grows, since it requires $n2^{\lfloor nr \rfloor}$ bits. Hence, as a first step towards a reduction in complexity we restrict our attention to *linear* codes.

§1.7.1. Linear Codes

We say that a code C over a field $\mathbb{F}$ is *linear* if it is closed under n-tuple addition and scalar multiplication:

$$\alpha x + \alpha' x' \in C, \quad \forall x, x' \in C \text{ and } \forall \alpha, \alpha' \in \mathbb{F}\,.$$

In fact, it suffices to check that

$$\alpha x - x' \in C, \quad \forall x, x' \in C \text{ and } \forall \alpha \in \mathbb{F}\,. \tag{1.19}$$

Choosing $\alpha = 0$ in (1.19) shows that if $x' \in C$ then so is $-x'$. Further, choosing $\alpha = 1$ and $x' = x$ shows that the all-zero word is a codeword of any (non-empty) linear code. Equivalently, since $\mathbb{F}^n$ is a *vector space*, condition (1.19) implies that a linear code is a *subspace* of $\mathbb{F}^n$.

For a linear code C the minimum distance $d(C)$ is equal to the *minimum of the weight* of all non-zero codewords,

$$\begin{aligned} d(C) &= \min\left\{d(x,x') : x, x' \in C, x \neq x'\right\} = \min\left\{d(x-x',0) : x, x' \in C, x \neq x'\right\} \\ &= \min\left\{w(x-x') : x, x' \in C, x \neq x'\right\} = \min\left\{w(x) : x \in C, x \neq 0\right\}. \end{aligned}$$

Since a linear code C of length n over $\mathbb{F}$ is a subspace of $\mathbb{F}^n$, there must exist an integer k, $0 \leq k \leq n$, so that C has a *dimension* k. This means that C contains $|\mathbb{F}|^k$ codewords. In the sequel we denote by $[n,k,d]$ the parameters of a linear code of length n, dimension k and minimum distance d. It is customary to call a $k \times n$ matrix G, whose rows form a linearly independent basis for C, a *generator* matrix for C. Conversely, given a matrix $G \in \mathbb{F}^{k\times n}$ of rank k we can associate with it the code $C(G)$,

$$C(G) \triangleq \left\{x \in \mathbb{F}^n : x = uG, u \in \mathbb{F}^k\right\}. \tag{1.20}$$

There are many generator matrices G which describe the same code (see Problem 1.5).

DEFINITION 1.21 (PROPER CODES). We say that a linear code is *proper* if its generator matrix G has no zero columns.[1] ▽

Zero columns convey zero information – no pun intended – and so we can safely restrict our attention to proper codes in the sequel.

DEFINITION 1.22 (SYSTEMATIC GENERATOR MATRIX). A generator matrix G of a linear code $C[n,k,d]$ is said to be in *systematic form* if $G = (I_k \; P)$, where I_k is a $k \times k$ identity matrix and P is an arbitrary $k \times (n-k)$ matrix with entries in $\mathbb{F}$. If G is in systematic form and $u \in \mathbb{F}^k$ is an information word, then the corresponding codeword $x = uG$ has the form (u, uP), i.e., the first k components of x are equal to the information word u. ▽

To each linear code C we associate the *dual* code $C^\perp$,

$$C^\perp \triangleq \left\{ v \in \mathbb{F}^n : xv^T = 0, \forall x \in C \right\} = \left\{ v \in \mathbb{F}^n : Gv^T = 0^T \right\}. \tag{1.23}$$

Assume that v and v' are elements of $C^\perp$ and that x is an element of C. Since $xv^T = 0 = x(v')^T$ implies $x(\alpha v - v')^T = 0$ for any $\alpha \in \mathbb{F}$, it follows that $C^\perp$ is a linear code as well. Therefore it has a basis. It is customary to denote such a basis by H. This basis H is a generator matrix of the code $C^\perp$. It is also said to be a *parity-check* matrix of the original code C. Let G be a generator matrix for a code C and let H be a corresponding parity-check matrix. By (1.23) the dual code is the set of solutions to the system of equations $Gv^T = 0^T$. Therefore, since G has k (linearly independent) rows we know, by linear algebra, that the dual code has dimension $n-k$, and therefore H has dimension $(n-k) \times n$. In fact, assume without loss of generality that G is in systematic form, $G = (I_k \; P)$. Represent v as $v = (v_s \; v_p)$, where v_s is of length k and where v_p is of length $n-k$. From $Gv^T = v_s^T + Pv_p^T$, we see that for each of the $|\mathbb{F}|^{n-k}$ distinct choices of v_p there is exactly one $v_s = -Pv_p^T$ so that $Gv^T = 0^T$.

The dual code is therefore characterized by

$$C^\perp \triangleq \left\{ v \in \mathbb{F}^n : v = vH, v \in \mathbb{F}^{n-k} \right\} = \left\{ v \in \mathbb{F}^n : Gv^T = 0^T \right\}.$$

In the same manner we have

$$C \triangleq \left\{ x \in \mathbb{F}^n : x = uG, u \in \mathbb{F}^k \right\} = \left\{ x \in \mathbb{F}^n : Hx^T = 0^T \right\}.$$

That the second description is true can be seen as follows. Clearly, for every $x \in C$, $Hx^T = 0^T$. This shows that $C \subseteq \left\{ x \in \mathbb{F}^n : Hx^T = 0^T \right\}$. But by assumption $|C| = |\mathbb{F}|^k = \left| \left\{ x \in \mathbb{F}^n : Hx^T = 0^T \right\} \right|$, since H has rank $n-k$.

[1] Note that properness is a code property, i.e., it does not depend on the generator matrix G which we choose to represent C.

As we will see, this latter description is particularly useful for our purpose. Given a generator matrix G, it is easy to find a corresponding parity-check matrix H and vice versa, see Problem 1.7.

EXAMPLE 1.24 (BINARY HAMMING CODES). Let $\mathbb{F} = \mathbb{F}_2$ and let $m \in \mathbb{N}$. Let H be a $m \times (2^m - 1)$ binary matrix whose columns are formed by all the binary m-tuples except the all-zero m-tuple. We claim that H is the parity-check matrix of a binary linear code of length $n = 2^m - 1$, dimension $2^m - m - 1$, and minimum distance 3. To see that the minimum distance is 3, note that any two columns of H are linearly independent, but that there are triples of columns which are linearly dependent. Therefore, $Hx^T = 0^T$ has no solution for $x \in \mathbb{F}_2^n$ with $1 \le w(x) \le 2$ but it has solutions with $w(x) = 3$. Clearly, C has dimension at least $2^m - m - 1$ since H has m rows. Let us now show that C must have dimension *at most* $2^m - m - 1$, i.e., we have equality. Since C has distance 3, the spheres of radius 1 centered at each codeword are disjoint. Let us count the total number of words contained in all these spheres:

$$|C|\left(\binom{n}{0} + \binom{n}{1}\right) = |C|(1+n) = |C|2^m.$$

From this it follows that $|C|2^m \le 2^n = 2^{2^m-1}$, where the right hand side represents the total number of points in the space $\mathbb{F}_2^n$. Turning this around we get $|C| \le 2^{2^m-m-1}$. This shows that C has dimension at most $2^m - m - 1$. Codes such that the spheres of radius $t \triangleq \left\lfloor \frac{d-1}{2} \right\rfloor$ centered around the codewords *cover* the whole space are called *perfect*. As a particular example consider the case $m = 3$. Then

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}, \tag{1.25}$$

and C_{Ham}, the code defined by the parity-check matrix H, is the $[7, 4, 3]$ binary Hamming code. ◇

In the above definitions we have assumed that the rows of G and H are linearly independent. For the sequel it is useful to relax this definition. We call a $k \times n$ matrix G a *generator* matrix *even* if G has rank strictly less than k. We say that k/n is the *design* rate of the code. The true rate is of course $\operatorname{rank}(G)/n$. An equivalent statement is true for a $(n-k) \times n$ parity-check matrix H.

DEFINITION 1.26 (ELIAS' GENERATOR AND GALLAGER'S PARITY-CHECK ENSEMBLE). Fix the blocklength n and the *design* dimension k. To sample from Elias' *generator* ensemble, construct a $k \times n$ generator matrix by choosing each entry iid according to a Bernoulli random variable with parameter one-half. To sample from Gallager's

parity-check ensemble, proceed in the same fashion to obtain a sample $(n-k)\times n$ parity-check matrix. Although both ensembles behave quite similarly, they are not identical. This is most easily seen by noting that every code in the generator ensemble has rate *at most* k/n and that some codes have a strictly smaller rate, whereas all codes in the parity-check ensemble have rate *at least* k/n. A closer investigation of the weight distribution of both ensembles is the topic of Problems 1.16 and 1.17. We denote these two ensembles by $\mathcal{G}(n,k)$ and $\mathcal{H}(n,k)$, respectively. ▽

For the most part we are only concerned with the binary case and therefore, unless explicitly stated otherwise, we assume in the sequel that $\mathbb{F}=\mathbb{F}_2$.

Are linear ensembles capable of achieving capacity? Consider, e.g., the generator ensemble $\mathcal{G}(n,k)$ and transmission over the BSC(ϵ). That the answer is in the affirmative can be seen as follows: consider a slight twist on the ensemble $\mathcal{G}(n,k)$. Pick a random element C from $\mathcal{G}(n,k)$ and a random *translation* vector c. The code is the set of codewords $C+c$. We translate so as to eliminate the special role that the all-zero word plays (since it is contained in any linear code). In this new ensemble the codewords are pairwise statistically independent and have the right marginals (namely the uniform distribution) as discussed in Problem 1.15. An inspection of the proof of Theorem 1.17 shows that these are the only two properties which are used in the proof. But a translation of all codewords leaves the error probability invariant so that also the ensemble $\mathcal{G}(n,k)$ itself is capable of achieving capacity. More generally, for any binary-input output-symmetric (see Definition 4.8) memoryless (see Definition 4.3) channel linear codes can achieve capacity.

§1.7.2. Description/Encoding Complexity of Linear Codes

From the generator and parity-check representation of a linear code we see that its description complexity is at most $\min\{rn^2,(1-r)n^2\}$ bits, where r is the rate of the code. Further, from (1.20) it is clear that the *encoding* task, i.e., the mapping of the information block onto the codeword can be accomplished in $O(n^2)$ operations.

§1.7.3. MAP Decoding Complexity of Linear Codes

Let us now focus on the *decoding complexity*. To keep things simple, we restrict ourselves to the case of transmission over the BSC(ϵ). Assume we have a uniform prior on the set of codewords. The optimal decoding rule then reads

$$\begin{aligned}
\hat{x}^{\mathrm{ML}}(y) &= \operatorname{argmax}_{x:Hx^T=0^T} p_{Y|X}(y\,|\,x)\\
&= \operatorname{argmax}_{x:Hx^T=0^T} \epsilon^{d(x,y)}(1-\epsilon)^{n-d(x,y)}\\
\text{since } \epsilon\le\tfrac{1}{2} \qquad &= \operatorname{argmin}_{x:Hx^T=0^T} d(x,y)\\
&= \operatorname{argmin}_{x:Hx^T=0^T} w(x+y)\\
e\triangleq x+y \qquad &= \operatorname{argmin}_{e+y:He^T=Hy^T} w(e)
\end{aligned}$$

$$s^T \triangleq Hy^T \qquad = \operatorname{argmin}_{e+y:He^T=s^T} w(e),$$

The quantity $s^T \triangleq Hy^T$ is called the *syndrome* and it is known at the receiver. Consider the following related decision problem.

PROBLEM Π: *ML Decision Problem*

INSTANCE: A binary $(n-k)\times n$ matrix H, a vector $s \in \{0,1\}^{n-k}$ and an integer $w > 0$.

QUESTION: Is there a vector $e \in \{0,1\}^n$ of weight at most w such that $He^T = s^T$?

Clearly, we can solve the above maximum likelihood decision problem once we have solved the associated maximum likelihood decoding problem: for a given s find the maximum likelihood estimate $\hat{x}^{\mathrm{ML}}$ and, therefore, the "error" vector e. By definition this is the lowest weight vector which "explains" the data and therefore the maximum likelihood decision problem has an affirmative answer for $w \geq w(e)$ and a negative answer otherwise. We conclude that the maximum likelihood decoding problem is at least as "difficult" as the above maximum likelihood decision problem.

In the theory of complexity a problem Π is said to belong to the class P if it can be solved by a deterministic Turing machine in *polynomial* time in the length of the input. Instead of a Turing machine one may think of a program written in (let's say) C running on a standard computer, except that this computer has infinite memory. Simply speaking, these are problems for which efficient algorithms are known: solving a system of linear equations and finding the minimum spanning tree or sorting are well-known examples in this class. Unfortunately, many problems which occur in practice appear not to belong to this class. A broader class is the class NP which contains all problems which can be solved by a *non-deterministic* Turing machine in *polynomial* time. A non-deterministic algorithm is one which, when confronted with a choice between two alternatives, can create two copies of itself and simultaneously follow the consequences of both courses. For our discussion it suffices to know that all problems of the form "Does there exist a subset with a specific property?", assuming that this property is easily checked for any given subset, belongs to the class NP. Of course, this may lead to an exponentially growing number of copies. The algorithm is said to solve the given problem if any one of these copies produces the correct answer. Clearly, we have P ⊂ NP and whether this inclusion is proper is an important open problem. Not necessarily all problems in NP are equally hard. Assume that a specific problem Π in NP has the property that any problem in NP can be reduced to Π in polynomial time. Then, ignoring polynomial factors, it is reasonable to say that Π is as hard as any problem in NP. We say that such a problem is NP-*complete*. We now are faced with the following discouraging result.

THEOREM 1.27. The *ML Decision Problem* for the BSC is NP-complete.

Does this mean that we should throw in the towel and declare that the efficient transmission of information at low error probability is a hopeless task? Not at all. First, the above result only says that for *some* codes the ML decoding problem is as hard as any problem in the class NP and therefore that for those codes the ML decoding problem is unlikely to have an efficient solution. All the transmission problem requires is the existence of *some* codes which allow the transmission of information close to capacity at low probability of error and which are efficiently decodable. Furthermore, as we saw already in the proof of the channel coding theorem, there exist suboptimal decoders which are powerful enough for the task.

§1.8. RATE, PROBABILITY, COMPLEXITY, AND LENGTH

The most important parameters for the transmission problem are: rate, probability of (block or bit) error, delay, and complexity (encoding and decoding). Delay is not an easy quantity to work with. It is therefore customary to consider instead the blocklength of the coding system. If you know already about convolutional codes (see Section 6.1) then you are aware that these two concepts are not necessarily the same. The situation gets even more complicated if we consider transmission over channels with feedback. But in the context of block coding schemes over channels without feedback (which is the focus of this book) we do not commit any fundamental error by equating the two.

Rate, probability of error, and blocklength are well-defined concepts. The notion of complexity on the other hand is somewhat fuzzy. As we have just discussed, there is a fairly clean separation between polynomial and exponential complexity. However, we typically discuss algorithms that have linear (in the blocklength) complexity and we are concerned with the actual constants involved. These constants of course depend on the technology we use to implement the system. In a hardware realization we typically mean the number of gates necessary or the number of connections required. If the system is implemented as a program, then we are concerned about the number of operations and the amount of memory. It is therefore only of limited value to give a formal definition of complexity and we are content with a more engineering oriented notion.

For a fixed channel, we want to transmit at large rates r with low probability of error P using simple encoding and decoding algorithms and short codes (small n). Clearly there are trade-offs: by their very nature longer codes allow more reliable transmission than shorter ones (or transmission at higher rates), and in a similar way more complex decoding algorithms (like, e.g. ML decoders) perform better than sub-optimal but lower-complexity algorithms. We want to determine all achievable

tuples

$$(r, \mathrm{P}, \chi_E, \chi_D, n)$$

and their associated coding schemes, where χ_E and χ_D denote the encoding and decoding complexity, respectively. This is a tall order and we are currently far from such a complete characterization. The task becomes much simpler if we ignore one of the quantities and investigate the trade-offs between the remaining. Clearly, the problem trivializes if we set no bounds on either the rate or the probability of error. On the other hand the problem stays non-trivial if we allow *unbounded complexity* or *unbounded delay* (unbounded blocklengths n).

A particularly useful transformation is to let $\delta \triangleq 1 - r/\mathrm{C}$, so that $r = (1-\delta)\mathrm{C}$. We call δ the *multiplicative gap*[2] to capacity and we are interested in the region of achievable tuples $(\delta, \mathrm{P}, \chi_E, \chi_D, n)$.

§1.8.1. Unbounded Complexity

If we ignore the issue of complexity, we enter the realm of classical information theory. We are interested in the behavior of the error probability as a function of the blocklength n and the gap δ.

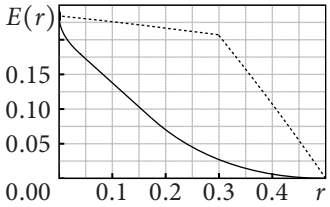

Figure 1.29: Error exponent of block codes (solid line) and of convolutional codes (dashed line) for the BSC($\epsilon \approx 0.11$).

Example 1.28 (Error Exponents for Block Codes). Consider transmission using a binary linear block code $C[n, nr]$ (a code of length n and dimension nr) via a discrete binary memoryless channel with capacity C, $\mathrm{C} > 0$, using a MAP decoder. Let $\mathrm{P}_{\mathrm{B}}^{\mathrm{MAP}}(n, r)$ be the resulting block error probability for the optimal choice of the code (the minimum of the error probability over all choices of the code with a given rate). It is a celebrated result of information theory that $\mathrm{P}_{\mathrm{B}}^{\mathrm{MAP}}(n, r)$ decreases exponentially fast in the blocklength, i.e.,

$$\mathsf{e}^{-n(E(r)+o_n(1))} \le \mathrm{P}_{\mathrm{B}}^{\mathrm{MAP}}(n, r) \le \mathsf{e}^{-nE(r)}. \tag{1.30}$$

[2] Not to be confused with the relative minimum distance.

$E(r)$ is called the *error exponent*. We have $E(r) > 0$ for all rates r in the range $r \in [0, \mathrm{C})$, where C is the (Shannon) *capacity* of the channel. For the BSC we determined its capacity in Section 1.6. (See Section 4.1.8 for the derivation of the capacity for more general channels.) Finally, $o_n(1)$ denotes a quantity which tends to zero as n tends to infinity. Figure 1.29 depicts the error exponent $E(r)$ (solid line) for the BSC($\epsilon \approx 0.11$). This error exponent is known exactly for $r \in [r_c, \mathrm{C}]$, where r_c, $0 < r_c < \mathrm{C}$, is called the *critical rate*. For rates $r \in [0, r_c)$, only a lower bound on $E(r)$ is known. At C the (left) derivative of $E(r)$ vanishes but the second (left) derivative is strictly positive, i.e., $E(r) = (\mathrm{C}-r)^2\alpha + O((\mathrm{C}-r)^3)$, where α is strictly positive. Therefore, if $r(\delta) = (1-\delta)\mathrm{C}$, $\delta \in [0,1]$, then $E(\delta) = \delta^2\mathrm{C}^2\alpha + O(\delta^3)$. More generally, for a "typical" discrete binary memoryless channels the error exponent as a function of δ, $\delta \in [0,1]$, has the form $E(\delta) = \delta^2\mathrm{C}^2\alpha + O(\delta^3)$, where α is strictly positive. We summarize: the error probability $\mathrm{P_b^{MAP}}$ behaves roughly like $\mathsf{e}^{-n\delta^2C^2\alpha}$, i.e., it decreases exponentially in n with an exponent which is proportional to δ^2. ◇

Example 1.31 (Error Exponents for Convolutional Codes). Consider transmission using a binary convolutional code (see Section 6.1) of rate r and with *memory* m via a discrete binary memoryless channel with capacity C, $\mathrm{C} > 0$, using a MAP decoder. Let $\mathrm{P_b^{MAP}}(m, r)$ be the resulting bit error probability for an optimal choice of the code. In analogy to the case of block codes, $\mathrm{P_b^{MAP}}(m, r)$ decreases exponentially fast in m, i.e,

$$\mathsf{e}^{-\frac{m}{r}(E(r)+o_m(1))} \le \mathrm{P_b^{MAP}}(m, r) \le \mathsf{e}^{-\frac{m}{r}E(r)}. \tag{1.32}$$

Similar to the block code case, this error exponent is known exactly for $r \in [\mathrm{C}_c, \mathrm{C}]$, where C_c, $0 < \mathrm{C}_c < \mathrm{C}$, is called the *critical rate*. Figure 1.29 depicts the error exponent $E(r)$ (dashed line) for the BSC($\epsilon \approx 0.11$). For rates $r \in [0, \mathrm{C}_c)$ only a lower bound on $E(r)$ is known. As the main difference to the block code case, for convolutional codes $E(r) = (\mathrm{C}-r)\alpha + O((\mathrm{C}-r)^2)$, where α is strictly positive. Therefore, if $r(\delta) = (1-\delta)\mathrm{C}$, then $E(\delta) = \delta\mathrm{C}\alpha + O(\delta^2)$. More generally, for any discrete binary memoryless channel the error exponent as a function of δ, $\delta \in [0,1]$, is given by $E(\delta) = \delta\mathrm{C}\alpha + O(\delta^2)$, where α is strictly positive. We summarize: the error probability $\mathrm{P_B^{MAP}}$ decreases exponentially in m but now the exponent is proportional to δ. ◇

§1.8.2. Unbounded Delay

For some applications fairly large blocklengths (delays) are perfectly acceptable. Therefore it is worth to investigate the behavior of coding schemes if we lift the restriction on blocklengths and only focus on rate, probability of error, and complexity. In the following we assume a somewhat naive computational model in which infinite precision arithmetic can be accomplished with unit cost.

EXAMPLE 1.33 (COMPLEXITY OF BLOCK CODES). Consider again transmission with a binary linear code $C[n, nr]$ via a discrete binary memoryless channel with capacity C, $\mathrm{C} > 0$, using a MAP decoder. Generically, the complexity of a MAP decoder, measured per information bit, is equal to $\frac{c}{nr} 2^{n \min(r, 1-r)}$, where c is a constant depending on the implementation. This can be seen as follows: there are 2^{nr} codewords. One straightforward approach is to determine the a posteriori probability for each of them given the observation and to choose the argument which maximizes this measure. Normalized per information bit, this gives rise to a complexity of $\frac{c}{nr} 2^{nr}$. On the other hand, as discussed in Problem 1.18, the MAP decoder can also be based on the parity-check matrix of the code, and this gives rise to a complexity of $\frac{c}{nr} 2^{n(1-r)}$. An alternative generic MAP decoding scheme which also gives rise to a complexity $\frac{c}{nr} 2^{n(1-r)}$ and which is based on the dual code is discussed in Problem 4.41. If in (1.30) we fix P_B and solve for n as a function of δ we get

$$n(\delta) = O(\frac{1}{\delta^2}).$$

It follows that the decoding complexity $\chi_D(\delta)$ is exponential in $1/\delta^2$. The encoding complexity normalized per information bit is equal to $\chi_E(n, r) = cn$, where c is again a small constant. We conclude that

$$\chi_E(\delta) = \frac{c}{\delta^2 \mathrm{C}^2 \alpha}. \qquad \diamond$$

EXAMPLE 1.34 (COMPLEXITY OF CONVOLUTIONAL CODES). Consider once more transmission with a binary convolutional code via a discrete binary memoryless channel with capacity C, $\mathrm{C} > 0$, using a ML decoder. ML decoding of these codes can be accomplished efficiently by means of the so-called *Viterbi algorithm* and the decoding complexity per information bit is equal to $\chi_D(r) = \frac{1}{r} 2^m$ (see Section 6.1). By going through essentially the same steps as before, we see that the decoding complexity grows exponentially in $\frac{1}{\delta}$. This is a large improvement compared to block codes but still exponential in the inverse of the gap.

The above argument might give rise to confusion. How can convolutional codes be better than block codes? After all, we can always terminate a convolutional code with negligible loss in rate and so construct an equivalent block code. The answer is that in our discussion of block codes we assumed a *generic* decoder which does not exploit any structure which might be present in the block code. If a suitable structure is present (as is the case for convolutional codes) we can do better. $\diamond$

There are many ways of combining given codes to arrive at a new code (see Problems 1.2 and 1.3). A basic and fundamental such construction is the one of *code concatenation*, an idea originally introduced by Forney. In this construction one

uses an *inner code* to convert the very noisy bits arriving from the channel into fairly reliable ones and then an *outer code* which "cleans up" any remaining errors in the block. Such code constructions can substantially decrease the decoding complexity viewed as a function of the blocklength n. But if we consider the decoding complexity as a function of the gap to capacity δ, it still increases exponentially in $1/\delta$.

Another avenue for exploration is the field of *sub-optimal* decoding algorithms. Although up to this point we have tacitly assumed that we want to perform MAP decoding, this is certainly not necessary as we can see from the proof of the channel coding theorem which itself uses a sub-optimal decoder.

Example 1.35 (Iterative Coding). Iterative coding is the focus of this book. Unfortunately the exact complexity versus performance trade-off for iterative decoding schemes is not known to date. In fact, it is not even known whether such schemes are capable of achieving the capacity for a wide range of channels. It is *conjectured* that

$$\chi_D(\delta, p) = \chi_E(\delta, p) = \frac{c}{\delta},$$

for some constant c which depends on the channel. To furnish a proof of the above conjecture is without doubt the biggest open challenge in the realm of iterative decoding. According to this conjecture the complexity (per information bit) grows *linearly* in the inverse to the gap as compared to exponentially. This is the main motivation for studying iterative decoding schemes. ◇

§1.9. First Tour of Iterative Decoding

We have seen that iterative decoding holds the promise of approaching the capacity with unprecedentedly low complexity. Before delving into the details (and possibly getting lost therein) let us give a short overview of the most important components and aspects. In all subsequent cases we opt for the simplest non-trivial scenario. There will be plenty of opportunity for generalizations later.

The first important ingredient is to represent codes in a *graphical* way. Consider again the Hamming code $C_{\text{Ham}}[7, 4, 3]$ given in terms of the parity-check matrix shown on page 15. By definition, $x = (x_1, \cdots, x_7)$, $x \in \mathbb{F}_2^7$, is a codeword of C_{Ham} if and only if

$$\begin{aligned} x_1 + x_2 + x_4 + x_5 &= 0, \\ x_1 + x_3 + x_4 + x_6 &= 0, \\ x_2 + x_3 + x_4 + x_7 &= 0. \end{aligned}$$

We associate with $C_{\text{Ham}}[7, 4, 3]$ the following graphical representation. We rewrite the three parity-check constraints in the form (recall that we work over $\mathbb{F}_2$ so that

$-x = x$)

$$x_1 + x_2 + x_4 = x_5, \tag{1.36}$$

$$x_1 + x_3 + x_4 = x_6, \tag{1.37}$$

$$x_2 + x_3 + x_4 = x_7. \tag{1.38}$$

Think of x_1, x_2, x_3, x_4 as the four independent *information* bits and x_5, x_6, x_7 as the derived *parity* bits. Let us represent the constraints via a *Venn* diagram as shown in Figure 1.39. Each circle corresponds to one parity-check constraint – the number of ones in each circle must be even. The topmost circle corresponds to the constraint

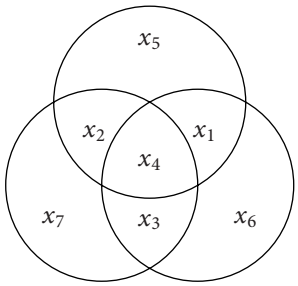

Figure 1.39: The Venn diagram representation of C.

expressed in (1.36). Consider first the *encoding* operation. Assume we are given $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$. It is then easy to fill in the missing values for (x_5, x_6, x_7) by applying one parity-check constraint at a time as shown in Figure 1.40: from (1.36) we deduce that $x_5 = 0$, from (1.37) it follows that $x_6 = 1$, and (1.38) shows that $x_7 = 0$. The complete codeword is therefore $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (0, 1, 0, 1, 0, 1, 0)$. Next, consider the *decoding* problem. Assume that transmission takes place over the binary erasure channel (BEC) with parameter ϵ (see page 69 for more details on this channel) and that the received message is $(0, ?, ?, 1, 0, ?, 0)$. We want to reconstruct the transmitted word. Figure 1.41 shows how this can be done. The decoding proceeds in a fashion similar to the encoding – we check each constraint (circle) to see if we can reconstruct a missing value from the values we already know. In the first step we recover $x_2 = 1$ using the constraint implied by the top circle. Next we determine $x_3 = 0$ by resolving the constraint given by the left circle. Finally, using the last constraint, we recover $x_6 = 1$. Unfortunately this "local" decoder – discovering one missing value at a time – does not always succeed. To see this, consider the case when the received message is $(?, ?, 0, ?, 0, 1, 0)$. A little thought (or an exhaustive check) show(s) that there is a unique codeword in C_{Ham}, namely $(0, 1, 0, 1, 0, 1, 0)$, which is compatible with this message. But the local decoding algorithm fails. As

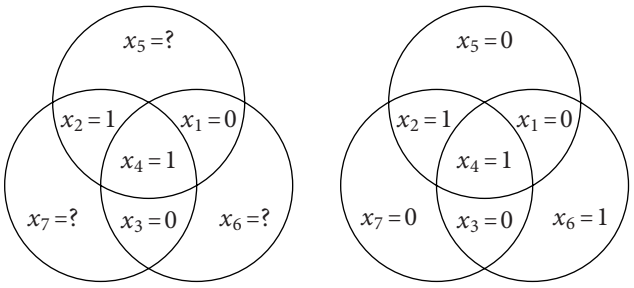

Figure 1.40: The encoding corresponding to $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$. The number of ones contained in each circle must be even. By applying one such constraint at a time the initially unknown components x_5, x_6, and x_7 can be determined. The resulting codeword is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (0, 1, 0, 1, 0, 1, 0)$.

one can see in Figure 1.42, none of the three parity-check equations by themselves can resolve any ambiguity.

In practice we encounter codes that have hundreds, thousands or even millions of nodes. In this case the Venn diagram representation is no longer convenient and we instead use the so-called *Tanner graph* description (see Sections 2.2 and 3.3). The basic principle however stays the same. Encoding and decoding is accomplished *locally*: specifically, we send messages along the edges of the Tanner graph which represents the code and process these messages locally at each node.

We summarize: we have seen that we can represent codes in a graphical manner and that we can use this graphical representation to perform both encoding and decoding. The two algorithms are very similar and they are *local*. This *local processing on a graphical model* is the main paradigm of iterative decoding. We have also seen that sometimes the iterative decoder fails despite the fact that a unique decoding is possible.

Figure 1.43 shows the block erasure probability of a so called $(3, 6)$*-regular low-density parity-check code* when transmission takes place over the binary erasure channel under iterative decoding (see Chapter 3). The channel is characterized by the parameter ϵ which denotes the *erasure* probability of each transmitted bit. This plot is representative of the typical behavior. For increasing lengths the individual curves become steeper and steeper and they converge to a limiting asymptotic curve. Of particular importance is the value ϵ^{BP} which marks the zero crossing of this asymptotic curve. For the example given we have $\epsilon^{\mathrm{BP}} \approx 0.4294$. Its operational meaning is the following: for sufficiently large blocklengths we can achieve arbitrarily small probability of error if we transmit over this channel with channel parameter strictly less than ϵ^{BP} (if the fraction of erased bits is strictly less than ϵ^{BP}); but if we

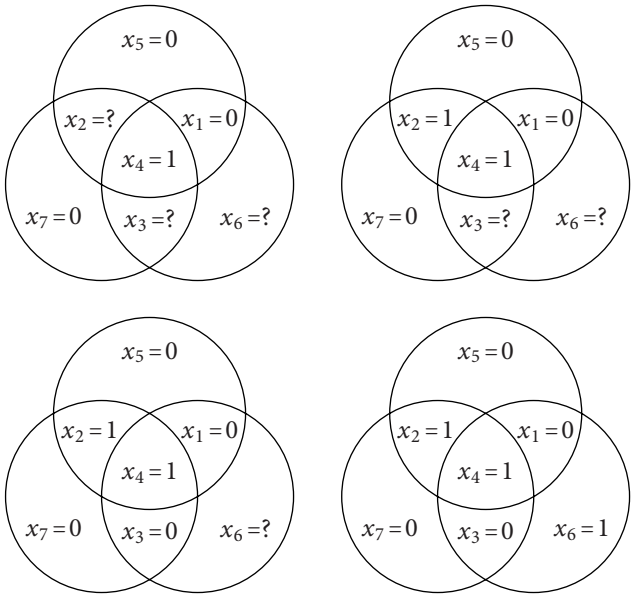

Figure 1.41: The decoding corresponding to the received message $(0, ?, ?, 1, 0, ?, 0)$. First we recover $x_2 = 1$ using the constraint implied by the top circle. Next we determine $x_3 = 0$ by resolving the constraint given by the left circle. Finally, using the last constraint, we recover $x_6 = 1$.

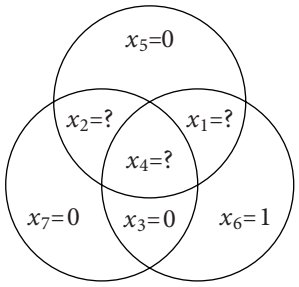

Figure 1.42: The decoding corresponding to the received message $(?, ?, 0, ?, 0, 1, 0)$. The local decoding fails since none of the three parity-check equations by themselves can resolve any ambiguity.

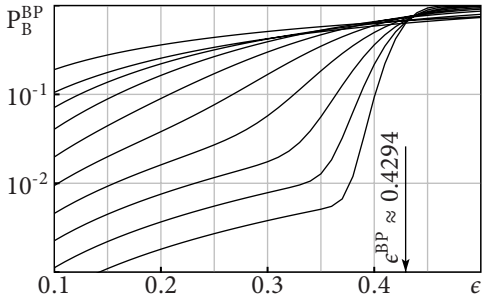

Figure 1.43: $\mathbb{E}_{\mathrm{LDPC}(nx^3, \frac{n}{2}x^6)}\left[\mathrm{P}_{\mathrm{B}}^{\mathrm{BP}}(\mathtt{G}, \epsilon)\right]$ as a function of ϵ for $n = 2^i$, $i \in [10]$.

choose the value of the channel parameter above ϵ^{BP} then the error probability is bounded away from zero. The value ϵ^{BP} acts therefore like a *capacity* for this particular coding scheme: it separates what is achievable from what is not. The rate of the code is one-half and that $\epsilon^{\mathrm{BP}} \approx 0.4294 < \frac{1}{2} = \epsilon^{\mathrm{Sha}}$, where ϵ^{Sha} denotes the Shannon threshold (the threshold which is achievable with optimal codes and under optimal decoding). This indicates that even in the limit of infinite block lengths we cannot achieve capacity with this particular coding scheme.

Is this failure to reach capacity due to the code or due to the suboptimal nature of the decoder? We will see that if our example code were decoded optimally the threshold would be $\epsilon^{\mathrm{MAP}} \approx 0.48815$, which still falls somewhat short of the Shannon threshold, namely 0.5. We conclude that both code and decoding algorithm are to blame. Why don't we just use elements from Gallager's parity-check ensemble, which we know can achieve capacity? Unfortunately, iterative decoding does not work well on elements of this ensemble. Therefore, in constructing codes that are capable of approaching Shannon capacity under iterative decoding we have to worry both about constructing good codes (which under optimal decoding could achieve capacity), and also about the performance of the suboptimal decoder compared to the optimal one.

Figure 1.43 also gives insight into the *finite-length* behavior. If we consider each finite-length curve, we see that it can be fairly cleanly separated into two regions – the so called *waterfall* region in which the error probability falls off sharply and the *error floor* region in which the curves are much more shallow. We will see that the waterfall region is due to large decoding failures and that in this region the decay of the error probability is exponential in the blocklength. On the other hand, the error floor is due to small failures and in this region the decay is only polynomial.

In Figure 1.43 we did not plot the performance of a code but the *average* performance of an ensemble. It is this ensemble approach which makes an analysis possible. What can we say about the performance of individual instances? For individual

instances the contributions stemming from large failures are sharply concentrated around this ensemble average. But the contributions due to small weaknesses in the graph are no longer concentrated. In fact, in the limit of large blocklengths, the *distribution* of these small weaknesses converges to a well-defined limit. The code design involves therefore two stages. First, we find ensembles that exhibit a large threshold (good behavior in the waterfall regime). Within this ensemble we then find elements that have few small weaknesses and, therefore, low error floors.

§1.10. Notation, Conventions, and Some Useful Facts

Large parts of this book should be accessible to a reader with a standard background in engineering mathematics but no prior background in coding. Some of the less standard techniques that are useful for our investigation are summarized in the appendices.

A familiarity with some basic notions of information theory is helpful in places. In fact not much is needed. We sometimes refer to the *entropy* of a random variable X which we denote by $H(X)$. Entropy is a measure of the "unpredictability" of a random variable. The smaller the entropy the easier it is to predict the outcome of a random experiment. We use small letters for variables and capital letters for random variables: we say that the random variable X takes on the value x. We write densities as $p_X(x)$, and sometimes we use the shorthand $p(x)$. In the case that X is discrete with probability distribution $p_X(x)$, the entropy is defined as $H(X) = -\sum_x p_X(x) \log p_X(x)$. If X has a density then the equivalent quantity is called *differential entropy* and is defined in the natural way as $h(X) = -\int_x p_X(x) \log p_X(x) \mathrm{d}x$. Entropy and differential entropy share the basic properties mentioned below and so we will not make any further notational distinctions and simply refer to entropy. At a few places we invoke the so-called *chain rule*: if X and Y are random variables then

$$H(X, Y) = H(X) + H(Y \,|\, X), \tag{1.44}$$

where $H(Y \,|\, X) \triangleq \sum_x H(Y \,|\, X = x) p_X(x)$ and where $H(Y \,|\, X = x)$ is the entropy of the random variable with probability distribution $p_{Y|X}(y \,|\, x)$, where x is fixed. This chain rule is a direct consequence of $p_{X,Y}(x, y) = p_X(x) p_{Y|X}(y \,|\, x)$ (see Problem 1.24). The rule extends in the natural way to more than two random variables: e.g., $H(X, Y, Z) = H(X) + H(Y \,|\, X) + H(Z \,|\, X, Y)$.

For the relatively simple channels we are concerned with the *Shannon capacity*, i.e., the *maximal rate* at which we can *reliably* transmit, is given by

$$\mathsf{C} = \max_{p_X(x)} I(X; Y), \tag{1.45}$$

where X denotes the input of the channel and Y the output and where the *mutual information* $I(X;Y)$ is equal to $H(X) - H(X\,|\,Y) = H(Y) - H(Y\,|\,X)$. The mutual information $I(X;Y)$ is a measure of how much information Y contains about X (or vice versa). Problem 1.27 re-derives the capacity of the BSC(ϵ) from this general formula.

A fundamental fact is that mutual information is non-negative, i.e., $I(X;Y) \geq 0$ (see Problem 1.25). Using the representation $I(X;Y) = H(X) - H(X\,|\,Y)$, we see that *conditioning does not increase entropy*, i.e.,

$$H(X) \geq H(X\,|\,Y). \tag{1.46}$$

We write

$$X \to Y \to Z \tag{1.47}$$

to indicate that the triple X, Y, and Z forms a *Markov chain*, i.e., to indicate that $p_{X,Y,Z}(x,y,z) = p_X(x)p_{Y\,|\,X}(y\,|\,x)p_{Z\,|\,Y}(z\,|\,y)$. In this case we have

$$p_{X,Z\,|\,Y}(x,z\,|\,y) = p_{X\,|\,Y}(x\,|\,y)p_{Z\,|\,X,Y}(z\,|\,x,y) = p_{X\,|\,Y}(x\,|\,y)p_{Z\,|\,Y}(z\,|\,y).$$

In words, if $X \to Y \to Z$ then X and Z are independent given Y. Conversely, $p_{X,Z\,|\,Y}(x,z\,|\,y) = p_{X\,|\,Y}(x\,|\,y)p_{Z\,|\,Y}(z\,|\,y)$ implies that $X \to Y \to Z$. By symmetry of this condition we see that $X \to Y \to Z$ implies $Z \to Y \to X$. We get a Markov chain in a natural way if, for a pair of random variables X and Y, we let $Z = f(Y)$ for some function $f(\cdot)$.

We make use of the *data processing inequality* which states that for any triple of random variables X, Y and Z such that $X \to Y \to Z$,

$$H(X\,|\,Y) \leq H(X\,|\,Z). \tag{1.48}$$

Equivalently, $I(X;Y) \geq I(X;Z)$. This is a natural statement: *processing can never increase the mutual information.*

Given a channel $p_{Y\,|\,X}(y\,|\,x)$ and a function $f(\cdot)$, we say that $Z \triangleq f(Y)$ is a *sufficient statistic* for X given Y if $X \to Z \to Y$, i.e., if X is independent of Y given Z (the relationship $X \to Y \to Z$ is always true if $Z = f(Y)$ as we have just discussed). A convenient necessary and sufficient condition that $Z = f(Y)$ constitutes a sufficient statistic is that $p_{Y\,|\,X}(y\,|\,x)$ can be written in the form $a(x,z)b(y)$ for some suitable functions $a(\cdot,\cdot)$ and $b(\cdot)$ (see Problem 1.21). For us the two most important consequences of knowing that $Z = f(Y)$ is a sufficient statistic for X given Y are that (i) the optimum decision on X can be based on Z alone, and that (ii) $H(X\,|\,Y) = H(X\,|\,Z)$. The first claim is a direct consequence of the fact that $p_{Y\,|\,X}(y\,|\,x)$ can be written in the form $a(x,z)b(y)$ and so the second term can be canceled in the

MAP rule (see Section 1.5). The second claim follows from the data processing inequality since we know that in this case we have both $X \to Y \to Z$ (which proves $H(X\,|\,Y) \le H(X\,|\,Z)$) but also $X \to Z \to Y$ (which proves $H(X\,|\,Z) \le H(X\,|\,Y)$).

It is helpful to know the *Fano* inequality: assume we want to estimate the random variable X taking values in the finite alphabet $\mathcal{X}$ knowing the (correlated) random variable Y. Let $\hat{x}(y)$ denote any estimation rule and let $\mathrm{P}\{\hat{x}(Y) \neq X\}$ denote the resulting probability of error. The Fano inequality asserts that (see Problem 1.26)

$$h(\mathrm{P}) + \mathrm{P}\log(|\mathcal{X}| - 1) \ge H(X\,|\,Y), \tag{1.49}$$

where $h(x) \triangleq -x\log x - (1-x)\log(1-x)$ is the binary entropy function. For the special case where X is a *binary* random variable the Fano inequality reads

$$h(\mathrm{P}) \ge H(X\,|\,Y). \tag{1.50}$$

Random variables X which have a Gaussian distribution appear frequently. In the Gaussian case we have $p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-(x-\mu)^2/(2\sigma^2))$, where μ is the *mean* of X, $\mu = \mathbb{E}[X]$, and σ^2 is the *variance* of X, $\sigma^2 = \mathbb{E}[(X-\mu)^2]$. We denote this density by $\mathcal{N}(\mu, \sigma^2)$.

If x is a real parameter taking values in $[0,1]$, e.g., a probability, we write $\bar{x}$ for $1-x$. If $n \in \mathbb{N}$ then the set $\{1, \cdots, n\}$ is often abbreviated as $[n]$.

Occasionally we want to bound the number of positive roots of a real valued polynomial. Let $p(x) \triangleq \sum_{i=0}^{d} p_i x^i$ be a *real-valued* polynomial of degree d. Consider the associated piecewise linear function whose graph interpolates the points (i, p_i), $i \in \{0, \ldots, d\}$. We say that $p(x)$ has *ν sign changes* if this associated function crosses the x-axis ν times.

EXAMPLE 1.51. Consider the polynomial $p(x) = 1 + x^4 - 3.4x^5 + x^{11}$. The associated piecewise linear graph passes the x-axis twice so that $\nu = 2$. ◇

THEOREM 1.52 (DESCARTE'S RULES OF SIGNS). Let $p(x)$ be a real-valued polynomial with ν sign changes and r positive real roots. Then $r \le \nu$ and $\nu - r$ is even.

EXAMPLE 1.53. Consider again the polynomial $p(x) = 1 + x^4 - 3.4x^5 + x^{11}$. Since $\nu = 2$ there exists either no positive root or there are exactly two. In fact, since $p(0) = 1 > 0$, $p(1) = -0.4 < 0$, and $p(2) = 1956.2 > 0$ we know that there are exactly two. A numerical calculation shows that the two roots are at $x \approx 0.89144$ and at $x \approx 1.11519$. ◇

We frequently need to refer to some basic relations and estimates of factorials and binomials. We summarize them here for reference.

$$n! = \int_0^\infty \mathrm{e}^{-s} s^n \mathrm{d}s, \tag{1.54}$$

$$n! = \sqrt{2\pi n}(n/\mathsf{e})^n(1 + O(1/n)), \tag{1.55}$$

$$(n/\mathsf{e})^n \leq n!, \tag{1.56}$$

$$n^k/k!\ \mathsf{e}^{-\frac{k^2}{n}} \leq \binom{n}{k} \leq n^k/k!, \tag{1.57}$$

$$\frac{1}{n+1}2^{nh_2(k/n)} \leq \binom{n}{k}, \tag{1.58}$$

$$\sum_{k=0}^{m}\binom{n}{k} \leq 2^{nh_2(m/n)}, \quad m \leq n/2, \tag{1.59}$$

$$\binom{2n}{n} = 4^n/\sqrt{n\pi}(1 - 1/(8n) + O(n^{-2})). \tag{1.60}$$

In general, we make no special distinction between scalars and vectors. In a few places, where confusion might otherwise arise, we denote a vector as $\underline{x}$. We assume that vectors are row vectors. We sometimes write x_i^j as a shorthand for the set of elements $x_i, x_{i+1}, \cdots, x_j$. Given a matrix A and a suitable index set $\mathcal{I}$, we let $A_{\mathcal{I}}$ denote the submatrix of A, restricted to the columns indexed by $\mathcal{I}$. We use the abbreviation *iid* to mean *independent and identically distributed*. If we deal with a set of (random) variables $X_1, \cdots, X_n$ it is often convenient to refer to *all but one* of them. In such a case we write $X_{\sim j}$ as a shorthand for $X_1, \cdots, X_{j-1}, X_{j+1}, \cdots, X_n$.

If $f(x)$ is a convex-$\cap$ function (we write "convex-$\cap$" for a concave function like $f(x) = \log(x)$ and "convex-$\cup$" for a convex function like x^2) and $p_X(x)$ is a density then *Jensen's* inequality asserts that

$$\int f(x)p_X(x)\mathrm{d}x \leq f\Big(\int x p_X(x)\mathrm{d}x\Big). \tag{1.61}$$

Given a polynomial $p(x)$ (or a function which is analytic around zero) we write $\mathrm{coef}\{p(x), x^k\}$ to denote the coefficient of x^k in its Taylor series expansion: $\mathrm{coef}\{(1+x)^n, x^k\} = \binom{n}{k}$.

If $n \in \mathbb{N}$ we say that a function $f(n)$ is $O(g(n))$ if there exists a constant c so that $|f(n)| \leq c|g(n)|$ for all sufficiently large $n \in \mathbb{N}$. We say that $f(n)$ is $o(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = 0$. Finally, we write $f(n) = \Theta(g(n))$ if there exist two strictly positive constants c' and c'' so that $c'|g(n)| \leq |f(n)| \leq c''|g(n)|$ for n sufficiently large. If the underlying parameter n is not clear from context we explicitly denote it by writing $O_n(\cdot)$, $o_n(\cdot)$, and $\Theta_n(\cdot)$.

Unless specifically said otherwise, all performance plots are for *ensemble averages* and not for specific instances. Expectations are denoted as $\mathbb{E}[\cdot]$. This makes the plots reproducible. Of course, typically better performance can be achieved by carefully selecting a specific instance from the ensemble.

We mark the end of a definition with $\triangledown$, the end of an example with $\diamond$, and the end of a proof with $\square$.

Theorems, lemmas, equations, figures, tables, etc. are labeled by *one sequence of consecutive* numbers within a chapter. This facilitates locating a referenced item.

Maybe slightly unusually, the main text does not contain pointers to the literature. The typical lemma and theorem is a synthesis of either a sequence of papers or has been discovered independently by several authors. Rather than interrupting the flow of ideas in the main text we have opted to collect a history of the developments of the main results at the end of each chapter.

By now, the list of papers in the area of iterative decoding measures in the (tens of) thousands. We have therefore not even attempted to give an exhaustive account of references but to give a sufficient number of pointers to the existing literature so that you can locate the relevant material quickly. We apologize to all authors whose papers we have not included in our list. For convenience of the reader we have placed a list of references at the end of each chapter. Each bibliographic entry is followed by a list of numbers within square brackets which indicate the pages on which this particular reference is cited.

The order of the exercises roughly follows the order of the topics discussed in the text. We have not ranked the exercises by difficulty. Those exercises that you can solve are easy, the other ones are hard.

Notes

Information theory and coding were born in Shannon's paper "A Mathematical Theory of Communication" [53]. It was in this paper that Shannon formulated the communications problem as we have presented it in Section 1.2 and showed that the transmission task can be broken down into a *source coding* and a *channel coding* problem. Our exposition of Shannon's *channel coding theorem* for the BSC closely follows van Lint [58].

At around the same time, and working at the same location (Bell Labs), Hamming constructed the first *class* of error-correcting codes [31]. Interestingly, Hamming's motivation was not Shannon's promised capacity but the practical concern of operating a "computing" machine consisting of many relays correctly, despite the fact that every couple of million relay operations an error would occur. A large fraction of the literature on coding for communications was thereafter concerned with finding dense packings of "Hamming"-spheres of radius half the minimum (Hamming) distance.

To date there exists an extensive literature on bounds on *weight distributions*. We refer the reader to the Handbook of Coding Theory [48] which contains several survey articles. The lower bound on the *minimum distance* which we discuss in Problem 1.12 is due to Gilbert [29]. A similar bound was independently discovered by Varshamov [60]. Asymptotically these two bounds yield the same result, which

is the statement discussed in Section 1.4 and in Problem 1.14. The upper bound on the minimum distance stated in Section 1.4 is due to Elias, who presented this result in class lectures in the early sixties. The bound was rediscovered by Bassalygo and first appeared in print in [5]. A substantially tighter upper bound was derived by McEliece, Rodemich, Rumsey, and Welch [44]. It is a tighter version of the linear *programming bound* based on Delsarte's *association schemes* [12]. For a discussion of the weight distribution of random codes (linear and nonlinear) see Barg and Forney [3] and the references therein.

The *union bound* on the performance of a code based on its weight distribution presented in Problem 1.20 is the simplest of a large class of such bounds. A considerably more sophisticated approach was discovered by Gallager in his thesis [25]. Good starting points are the article and the monograph by Sason and Shamai [52, 50] which discuss in depth the relationships between the numerous bounds proposed to date and which contain an exhaustive list of references.

In 1971 Cook proved that any problem in NP can be reduced in deterministic polynomial time to the *satisfiability* problem [10]. Later, Karp showed that the satisfiability problem can be reduced to many other problems in NP in deterministic polynomial time. The collection of all these problem is the class of NP-complete problems. The fact that the ML decision problem is NP-complete for binary linear codes was shown by Berlekamp, McEliece, and van Tilborg [7] by reducing the problem to *three-dimensional matching*. The intractability of computing the minimum distance was later shown by Vardy [59]. The classic reference relating to complexity is the book by Garey and Johnson [28]. For survey articles concerning complexity issues in coding theory see Barg [2], Sudan [56], Dumer, Micciancio, and Sudan [14] as well as Spielman [55].

Elias showed that linear codes achieve capacity for the BSC [15]. A proof that linear codes achieve capacity on *any* binary-input output-symmetric memoryless channel can be found in Gallager [27].

BCH codes were discovered by *B*ose and Ray-*C*haudhuri [9] and independently by *H*ocquenghem [36]. *Reed-Solomon* codes (see Problem 1.8), which can be seen as a special case of BCH codes, were proposed around the same time by Reed and Solomon [49]. They are part of many standards and products. Fairly recently, there have been exciting new developments concerning the decoding of Reed-Solomon codes beyond the minimum distance. This development was initiated by Sudan who introduced his list decoding algorithm [57] for Reed-Solomon codes. This was then quickly followed by an improved list decoding algorithm due to Guruswami and Sudan [30]. Further, it was shown by Kötter and Vardy how to turn the list decoding algorithm into a soft-decision decoding algorithm [38].

Convolutional codes (Example 1.31) are due to Elias [15]. The theory of convolutional codes was developed by Forney in the series of papers [20, 21, 22]. Fur-

ther excellent references on the theory of convolutional codes are the books by Johannesson and Zigangirov [37], Viterbi and Omura [62], and the survey article by McEliece, which is contained in the collection [48]. *Sequential decoding* was invented by Wozencraft [65] with important contributions due to Fano [16]. The *Viterbi algorithm* was originally introduced by Viterbi [61] as a proof technique. It was pointed out by Forney [17] and later independently by Omura [45] that the Viterbi algorithm was optimal, i.e., that it accomplished MAP decoding. Heller, who at that time worked at the Jet Propulsion Laboratory (JPL), was the first to recognize that the Viterbi algorithm was not only of theoretical interest but also immensely practical [33, 34]. The classical reference is [23].

Forney developed the theory of *concatenated codes* (see page 21) in his thesis [18, 19].

A simple derivation of the random coding exponent was given by Gallager in [26], extending Shannon's random coding technique [53]. Further bounds were developed by Shannon, Gallager, and Berlekamp [54]. The equivalent statements for convolutional codes were developed by Yudkin [66], Viterbi [61], and Forney [24]. The currently best known bounds on the error exponent of block codes are due to Barg and McGregor [4].

For a history of the development of iterative coding see the historical notes at the end of Chapter 4. The representation of Hamming codes via Venn diagrams is due to McEliece [43].

The Descartes rule of signs can be found in [13] and [35, Chapter 6]. The idea of performing bit MAP decoding of a block code via a trellis, as discussed in Problem 1.18, is due to Bahl, Cocke, Jelinek, and Raviv [1]. The trellis is typically called the *Wolf trellis* [64], in reference to Wolf's work on the related block MAP decoding problem (another name is *syndrome trellis*). The MacWilliams identities, discussed in Problem 1.19, which give a beautiful connection between the weight distribution of a code and its dual, were published by her in [40].

There is a large list of excellent books on classical coding theory. To mention just a few of the most popular ones in alphabetic order: Berlekamp [6], Blahut [8], Lin and Costello [39], MacWilliams and Sloane [41], McEliece [42], Pless [47], and van Lint [58]. A list of survey articles was collected by Pless and Huffman in [48]. For books that focus exclusively on convolutional codes see Johannesson and Zigangirov [37], Piret [46] and Viterbi and Omura [62]. More recently, some books on iterative decoding also have become available: Heegard and Wicker [32], Schlegel and Perez [51], and Vucetic and Yuan [63]. Readers interested in information theory need look no further than to Gallager [27] or Cover and Thomas [11]. A further recommended source which spans a wide range of topics is the book by MacKay.

Codes also play an important role in many areas of science. To name just a few: in *mathematics* they give rise to dense point lattices and allow the construction of

designs for use in *statistics*. In the area of *computer science*, public-key crypto systems may be based on codes, they are the crucial ingredient for some extractors – generating many weakly random bits from a few truly random ones – and they make efficient hashing functions. Codes are helpful in minimizing the communication complexity and they can be used to turn worst-case hardness into average-case hardness. In the area of *communications*, besides helping to combat the noise introduced by the communication channel, codes can help to represent sources efficiently, to avoid large peak-to-average power ratios, or to minimize interference in multi-user systems. Most of these areas are well outside the scope of this book.

Problems

1.1 (Inner Product). Let $\mathbb{F}$ be a field and consider the vector space $\mathbb{F}^n$, $n \in \mathbb{N}$. For $u, v \in \mathbb{F}^n$ define the *inner product* of u and v by $\langle u, v\rangle \triangleq \sum_{i=1}^n u_i v_i$, where all operations are performed in $\mathbb{F}$. Show that this inner product has the following properties

1. $\langle t+u, v\rangle = \langle t, v\rangle + \langle u, v\rangle, \quad t, u, v \in \mathbb{F}^n$
2. $\langle \alpha u, v\rangle = \alpha\langle u, v\rangle, \quad \alpha \in \mathbb{F}, u, v \in \mathbb{F}^n$
3. $\langle u, v\rangle = \langle v, u\rangle, \quad u, v \in \mathbb{F}^n$

Unfortunately, $\langle \cdot, \cdot \rangle$ is not necessarily an inner product in the mathematical sense: show that for $\mathbb{F} = \mathbb{F}_2$, $\langle u, u\rangle = 0$ does not imply $u = 0$ by exhibiting a counter example. Therefore, $\mathbb{F}_2^n$, equipped with $\langle \cdot, \cdot \rangle$, is not an inner-product space. As a consequence, if G is a generator matrix over $\mathbb{F}_2$ and H is a corresponding parity-check matrix, then

$$\begin{pmatrix} G \\ H \end{pmatrix}$$

does not necessarily span the whole space $\mathbb{F}_2^n$ – the row-space spanned by H is not the orthogonal complement of the row-space spanned by G. Consider, e.g., the generator matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

and the associated code $C(G)$. Determine $C^\perp$. A code C such that $C = C^\perp$ is called *self-dual*.

1.2 (Extending, Puncturing, and Shortening). In this exercise we are concerned with certain simple procedures which allow us to construct new codes from given ones. Assume we have a code $C(n, M, d)$ (not necessarily linear). If d is odd then we can *extend* the code by appending to each codeword $x = (x_1, \cdots x_n)$ the extra symbol $x_{n+1} \triangleq -\sum_{i=1}^n x_i$. The inverse operation, namely to delete a symbol, is

called *puncturing*. Finally, consider any subset of codewords which have the same last symbol. Keep only this subset and delete this common symbol. This procedure is called *shortening*. How do these procedures change the parameters of the code? Under which of these operations do linear codes stay linear?

1.3 ($u + v$ Construction). Assume we are given two binary codes $C_1(n, M_1, d_1)$ and $C_2(n, M_2, d_2)$. Define a new code C by

$$C \triangleq \{(u, u + v) | u \in C_1, v \in C_2\}.$$

Show that C is binary and has parameters $C(2n, M_1 M_2, d = \min(2d_1, d_2))$.

1.4 (Proper Codes). Let C be a proper (see Definition 1.21 on page 14) linear code of length n over the field $\mathbb{F}$. Prove that for every position i, $1 \le i \le n$, and every field element α, $\alpha \in \mathbb{F}$, the number of codewords that have an α at position i is equal to $|C|/|\mathbb{F}|$. What happens if the code is not proper?

1.5 (One C, many G). Show that a binary linear code C of length n and dimension k has $2^{\binom{k}{2}} \prod_{i=1}^{k} \left(2^i - 1\right)$ distinct $k \times n$ generator matrices.

1.6 (Conversion of G into Systematic Form). Let G be a $k \times n$ generator matrix of rank k. Show that G can be brought into systematic form by elementary row/column operations (i.e., permutations of rows/columns, multiplication of a row by a non-zero element of $\mathbb{F}$ and addition of one row to another.) Prove that these operations do not change the minimum distance of the code.

Hint: Show that G must contain a $k \times k$ rank- k submatrix, call it A, formed by k columns of G. Multiply G from the left by A^{-1} and perform column permutations if necessary to bring the result into sytematic form.

1.7 (Conversion $G \leftrightarrow H$). Show that if G is a generator matrix in systematic form, $G = (I_k P)$, then $H = \left(-P^T I_{n-k}\right)$ is a corresponding parity-check matrix. The parity-check matrix of the $[7, 4, 3]$ binary Hamming code given in (1.25) has the form $\left(-P^T I_{n-k}\right)$. Use this fact to write down a corresponding generator matrix G.

1.8 (Reed-Solomon Codes). Reed-Solomon (RS) codes are one of the most widely used codes today. In addition they possess several astounding properties (see e.g., Problem 1.10). Here is how they are defined.

Given a finite field $\mathbb{F}$, choose n and k such that $n \le |\mathbb{F}|$ and $1 \le k \le n$. To construct a Reed-Solomon (RS) code with parameters n and k over the field $\mathbb{F}$ choose n *distinct* elements from $\mathbb{F}$, call them $x_0, \cdots, x_{n-1}$. Let $\mathbb{F}[x]$ denote the ring of polynomials with coefficients in $\mathbb{F}$, i.e., the set of polynomials in the indeterminate x with

coefficients in $\mathbb{F}$ and the standard addition and multiplication of polynomials. Then C is defined as

$$C \triangleq \{(A(x_0), \cdots, A(x_{n-1})) : A(x) \in \mathbb{F}[x] \text{ s.t. } \deg(A(x)) < k\}.$$

In words, we consider the set of polynomials with coefficients in $\mathbb{F}$ and degree at most $k-1$. Each such polynomial we evaluate at the n distinct points x_i, $0 \le i < n$, and the result of these evaluations form the n components of a codeword.

Show that a RS code as defined above with parameters n and k over the field $\mathbb{F}$ has dimension k and minimum distance $n-k+1$. We summarize the parameters of such a code in the compact form $[n, k, n-k+1]$.

Hint: Over any field $\mathbb{F}$, a polynomial of degree d has at most d roots in $\mathbb{F}$. This is called the *fundamental theorem of algebra*.

1.9 (Singleton Bound). Show that for any code $C(n, M, d)$ over a field $\mathbb{F}$, $d \le n - \log_{|\mathbb{F}|}(M) + 1$. This is called the *Singleton* bound. Codes which achieve this bound are called *maximum-distance separable* (MDS).

Hint: Arrange the M codewords of length n in form of a $M \times n$ matrix and delete all but the first $\lfloor \log_{|\mathbb{F}|}(M) \rfloor$ columns. Argue that the minimum distance between the resulting rows is at most one.

1.10 (Maximum Distance Separable Codes). As discussed in Problem 1.9, a code C which fulfills the Singleton bound is called maximum distance separable (MDS). Examples of such codes are repetition codes with parameters $[n, 1, n]$ and Reed-Solomon codes with parameters $[n, k, n-k+1]$. Let C be an MDS code with parameters $[n, k, d = n-k+1]$. Show that C has the following property. Any subset of $[n]$ of cardinality k is an *information set*. More precisely, for any such subset and any choice of elements of $\mathbb{F}$ at these k positions there is exactly one codeword which agrees with this choice.

Next let C be a binary linear code with parameters $[n, k]$ and generator matrix G. Assume that its dual, $C^\perp$, is a MDS code (this dual has hence parameters $[n, n-k, k+1]$). Show that also in this case C has the property that any subset of $[n]$ of cardinality k is an information set.

1.11 (Hamming Bound). Consider a code $C(n, M, d)$ over a field $\mathbb{F}$. Prove that $M \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} (|\mathbb{F}| - 1)^i \le |\mathbb{F}|^n$.

Hint: Consider spheres of radius $t \triangleq \left\lfloor \frac{d-1}{2} \right\rfloor$ centered around each codeword.

Note: A code C which fulfills the Hamming bound with equality is said to be *perfect*. It is easy to check that binary repetition codes of odd length are perfect codes. Further we saw that all Hamming codes are perfect. Without proof we state that the only other perfect multiple-error correcting codes are the $[23, 12, 7]$ binary Golay code and the $[11, 6, 5]$ ternary Golay code.

1.12 (Gilbert-Varshamov Bound). Show that for any $d \leq n$, $n \in \mathbb{N}$, and any field $\mathbb{F}$ there exist codes $C(n, M, d)$ over $\mathbb{F}$ with $M \geq \frac{|\mathbb{F}|^n}{\sum_{i=0}^{d-1} \binom{n}{i}(|\mathbb{F}|-1)^i}$.

Hint: Consider spheres of radius $d-1$ centered around each codeword.

1.13 (Greedy Code Search Algorithm). The Gilbert-Varshamov bound which we discussed in Problem 1.12 gives rise to the following greedy code search algorithm. Start by picking a point in $\mathbb{F}^n$ where the choice is made with uniform probability. The initial code C consists of this chosen codeword. At any following step in the construction algorithm delete from $\mathbb{F}^n$ all codewords currently in C as well as all points contained in spheres of radius $(d-1)$ around these codewords, where d is the design distance. If the resulting set is non-empty pick another point and add it to C, otherwise the algorithm terminates. Show that this algorithm results in a code $C(n, M, d)$ over $\mathbb{F}$ with minimum distance at least d and cardinality at least

$$\left\lceil |\mathbb{F}|^n / \Big(\sum_{i=0}^{d-1} \binom{n}{i} (|\mathbb{F}|-1)^i \Big) \right\rceil .$$

Note: Such a greedy algorithm does in general not result in a linear code. Surprisingly, the bound is still true, and a slightly modified greedy algorithm still works, if we restrict C to be linear. Define $V(\mathbb{F}, n, t) \triangleq \sum_{i=0}^{t} \binom{n}{i}(|\mathbb{F}|-1)^i$. Let C_0 be the trivial $[n, 0, n]$ code consisting of the zero word only. In general assume that we constructed already the linear $[n, k, \geq d]$ code C_k. Such a code contains $M = q^k$ codewords. If $q^k V(\mathbb{F}, n, d) \geq q^n$ then we stop and output C. Otherwise proceed as follows. Delete from $\mathbb{F}^n$ all codewords of C_k as well as all points contained in the spheres of radius $(d-1)$ around these codewords. Since $q^k V(\mathbb{F}, n, d-1) < q^n$, the resulting set is not empty. Pick any element from this set and call it g_{k+1}. Let C_{k+1} be the linear code resulting from joining g_{k+1} to a set of generators $\{g_1, \cdots, g_k\}$ for C_k. We claim that C_{k+1} has minimum distance at least d. Every element c' in C_{k+1} has a representation of the form $c' = ag_{k+1} + c$ where $a \in \mathbb{F}$ and $c \in C_k$. If $a = 0$ then $w(c') = w(c) \geq d$ by assumption that C_k is a $[n, k, \geq d]$ code. If on the other hand $a \neq 0$ then $w(c') = w(ag_{k+1} + c) = w(g_{k+1} + a^{-1}c) = d(g_{k+1}, -a^{-1}c) \geq d$ by our choice of g_{k+1} and since by linearity $-a^{-1}c \in C_k$.

1.14 (Asymptotic Gilbert-Varshamov Bound). Fix $\mathbb{F} = \mathbb{F}_2$. Consider codes of increasing blocklength n with $2^{\lfloor nr \rfloor}$ codewords, where r, $r \in (0, 1)$, is the rate. Let $d(C)$ denote the minimum distance of a code C and define the normalized distance $\delta = d/n$. Starting with the Gilbert-Varshamov bound discussed in Problem 1.12, show that $\delta^*(r)$ as defined on page 7 fulfills $\delta^*(r) \geq h_2^{-1}(1-r)$.

1.15 (Pairwise Independence for Generator Ensemble). Let $\mathbb{F} = \mathbb{F}_2$ and consider the following slight generalization of Elias' generator ensemble $\mathcal{G}(n, k)$, call

it $\tilde{\mathcal{G}}(n,k)$. To sample from $\tilde{\mathcal{G}}(n,k)$ choose a random element C from $\mathcal{G}(n,k)$ and a random translation vector c from $\mathbb{F}_2^n$. The random sample $\tilde{C}$ is $\tilde{C} \triangleq C + c$, i.e., a translated version of the code C. Prove that in $\tilde{\mathcal{G}}(n,k)$ codewords have a uniform distribution and that pairs of codewords are independent. More precisely, let $\{u^{[i]}\}_i$, where i ranges from 0 to $2^k - 1$, denote the set of binary k-tuples, ordered in some fixed but arbitrary way. For $G \in \mathcal{G}(n,k)$ and $c \in \mathbb{F}_2^n$, let $x^{[i]}(G,c) \triangleq c + u^{[i]}G$ be the i-th codeword. Prove that for $j \neq i$, $\mathrm{P}\{X^{[j]} \mid X^{[i]} = x\}$ is uniform, i.e., for any $v \in \mathbb{F}_2^n$, $\mathrm{P}\{X^{[j]} = v \mid X^{[i]} = x\} = \frac{1}{2^n}$.

1.16 (Mean and Second Moment for $\mathcal{G}(n,k)$). Consider Elias' generator ensemble $\mathcal{G}(n,k)$, $0 < k \leq n$, as described in Definition **??**. Recall that its design rate is equal to $r = k/n$. For $C \in \mathcal{G}(n,k)$, let $A(C,w)$ denote the codewords in C of Hamming weight w. Show that

$$\begin{aligned}
\mathbb{E}_C[A(C,w=0)] &= 1 + \frac{2^{nr}-1}{2^n} \triangleq \bar{A}(w=0),\\
\mathbb{E}_C[A(C,w)] &= \binom{n}{w}\frac{2^{nr}-1}{2^n} \triangleq \bar{A}(w), w \geq 1,\\
\mathbb{E}_C[A^2(C,w)] &= \bar{A}(w) + \frac{(2^{nr}-1)(2^{nr}-2)}{2^{2n}}\binom{n}{w}^2, w \geq 1,\\
\mathbb{E}_C[(A(C,w)-\bar{A}(w))^2] &= \bar{A}(w) - \frac{2^{nr}-1}{2^{2n}}\binom{n}{w}^2, w \geq 1.
\end{aligned}$$

What is the expected number of distinct codewords in an element chosen uniformly at random from $\mathcal{G}(n,k)$?

1.17 (Mean and Second Moment for $\mathcal{H}(n,k)$). Consider Gallager's parity-check ensemble $\mathcal{H}(n,k)$, $0 < k \leq n$, as described in Definition 1.26. Recall that its design rate is equal to $r = k/n$. For $C \in \mathcal{H}(n,k)$, let $A(C,w)$ denote the codewords in C of Hamming weight w. Show that

$$\begin{aligned}
A(C,w=0)] &= 1,\\
\mathbb{E}_C[A(C,w)] &= \binom{n}{w}2^{-(n-k)} \triangleq \bar{A}(w), \quad w \geq 1,\\
\mathbb{E}_C[A^2(C,w)] &= \bar{A}(w) + \binom{n}{w}\left(\binom{n}{w} - 1\right)2^{-2n(1-r)}, \quad w \geq 1.\\
\mathbb{E}_C[(A(C,w)-\bar{A}(w))^2] &= \bar{A}(w) - \binom{n}{w}2^{-2n(1-r)}, \quad w \geq 1.
\end{aligned}$$

What is the expected number of distinct codewords in an element chosen uniformly at random from $\mathcal{H}(n,k)$?

1.18 (Wolf Trellis – Bahl, Cocke, Jelinek, and Raviv [1], Wolf [64]). There is an important graphical representation of codes, called *trellis*. A trellis $T = (V, E)$ of rank n is a finite directed graph with vertex set V and edge set E. Each vertex is assigned a "depth" in the range $\{0, \cdots, n\}$, and each edge connects a vertex at depth i to one at depth $i+1$, for some $i = 0, 1, \cdots, n-1$. The Wolf trellis for a binary linear code C with $(n-k) \times n$ parity-check matrix $H = (h_1^T, \cdots, h_n^T)$, where h_j^T is the jth column of H, is defined as follows. At depth i, $i = 0, \cdots, n$, the vertex set consists of 2^{n-k} vertices which we identify with the set of binary $(n-k)$-tuples. A vertex v at depth i is connected to a vertex u at depth $(i+1)$ with label $\lambda \in \{0, 1\}$ if there exists a codeword $c = (c_1, \cdots, c_n) \in C$ with $c_{i+1} = \lambda$ such that

$$v = \sum_{j=1}^{i} c_j h_j \quad \text{and} \quad u = \sum_{j=1}^{i+1} c_j h_j,$$

where by convention $\sum_{j=1}^{0} c_j h_j = (0, \cdots, 0)$.

Draw the Wolf trellis for the $[7, 4, 3]$ Hamming code with parity-check matrix given in (1.25). There is a one-to-one correspondence between the labeled paths in the Wolf trellis and the codewords of C. Also, many codewords "share" edges and so the total number of edges is, in general, much smaller than the number of codewords times the length of the code. Hence, the Wolf trellis is a compact representation of a code. Many important algorithms can be performed on the trellis of a code (e.g., decoding, determination of minimum distance etc.) We revisit this topic in Section 6.1 when discussing convolutional codes.

1.19 (MacWilliams Identities – MacWilliams [40]). For linear codes there is a simple relationship between the weight distribution of a code and the weight distribution of its dual code. Let C be a binary linear $[n, k]$ code. Let A_i, $i = 0, \cdots, n$, be the number of codewords in C of weight i. Note that $A_0 = 1$ and that $|C| = \sum_{i=0}^{n} A_i$. We call the collection $\{A_i\}_{0 \le i \le n}$ the weight distribution of the code C. The MacWilliams identity states that

$$A_j^{\perp} = \frac{1}{|C|} \sum_{i=0}^{n} A_i P_j(i), \ 0 \le j \le n,$$

where $P_j(i) = \sum_{l=0}^{n} (-1)^l \binom{i}{l} \binom{n-i}{j-l}$.

1. Prove that $\sum_{i=0}^{n} A_i \binom{n-i}{n-w} = \sum_{|E|=w} |C(E)|$ for every $0 \le w \le n$. Here, E denotes a subset of $\{1, \cdots, n\}$ and $|E| = w$ denotes a subset of $\{1, \cdots, n\}$ of size w. Finally, for a given E, $C(E)$ denotes the subcode of C with zeros outside E. More precisely, $C(E)$ is the collection of those elements of C which have non-zero elements only within the coordinates indexed by E. Note that the zero codeword is always in $C(E)$ so that $C(E)$ is not empty.

2. Let H_E denote the submatrix formed by the columns of H that are indexed by E. Prove that $C(E)$ is a binary linear code with dimension $\dim(C(E)) = w - \operatorname{rank}(H_E)$.

3. Prove that $w - \operatorname{rank}(H_E) = k - \operatorname{rank}(G_{\bar{E}})$ where $\bar{E}$ is the complement of E within $\{1, \cdots, n\}$. It can be noticed that this property is also the central element used in Theorem 3.77 (see Lemma 3.75).

4. Let $\{A_i^\perp\}_{0 \le i \le n}$ denote the weight distribution of the dual code $C^\perp$. Prove that

$$\sum_{i=0}^{n-u} A_i^\perp \binom{n-i}{u} = \sum_{|E|=n-u} |C^\perp(E)| \tag{1.62}$$

$$= \sum_{|E|=n-u} 2^{n-u-\operatorname{rank}(G_E)} \tag{1.63}$$

$$= 2^{n-k-u} \sum_{|E|=n-u} 2^{u-\operatorname{rank}(H_{\bar{E}})} \tag{1.64}$$

$$= 2^{n-k-u} \sum_{i=0}^{u} A_i \binom{n-i}{n-u}. \tag{1.65}$$

5. Use $\sum_i (-1)^{j-i} \binom{n-j}{n-i} \binom{n-i}{n-w} = \delta_{j,w}$ to prove $A_j^\perp = \frac{1}{|C|} \sum_{i=0}^n A_i P_j(i)$, $0 \le j \le n$, where $P_j(i) = \sum_{l=0}^n (-1)^l \binom{i}{l} \binom{n-i}{j-l}$. Hint: You might have to consider the generating function $\sum_{i=0}^n P_j(i) x^i = (1+x)^{n-i}(1-x)^i$ and to expand it using the form $(1-x)^n (1 + \frac{2x}{1-x})^{n-i}$.

6. Use your result to calculate the weight distribution of the dual of the $[7, 4, 3]$ Hamming code.

1.20 (Upper Bound On Error Probability Via Weight Distribution). Consider a linear binary code C of length n and dimension k. Let A_w denote the number of words of weight w and let

$$A(D) \triangleq \sum_{w=0}^{n} A_w D^w.$$

$A(D)$ is called the *weight enumerator* of the code. Consider the parity-check ensemble $\mathcal{H}(n, k)$.

Let x be a *fixed* binary word of length n. Argue that for a randomly chosen element $C \in \mathcal{H}(n, k)$

$$\mathrm{P}\{x \in C\} = \begin{cases} 1, & x = 0, \\ 2^{-(n-k)}, & \text{if no such } x \text{ exists.} \end{cases}$$

Hint: What is the probability that x fulfills *one* parity-check equation, i.e., that $\sum_j H_{i,j}x_j = 0$ for some fixed i?

Since H is random A_w is a random variable as well and we denote it by $A_w(H)$. Use the previous result and argue that *every* code in this ensemble contains the all-zero word, in particular

$$\bar{A}_0 \triangleq \mathbb{E}[A_0(H)] = 1,$$

and that the *expected* number of codewords of weight w, $w > 0$, is given by

$$\bar{A}_w \triangleq \mathbb{E}[A_w(H)] = \binom{n}{w} 2^{-(n-k)}.$$

Define the *average* weight enumerator as

$$\bar{A}(D) \triangleq \sum_{w=0}^{n} \bar{A}_w D^w.$$

Using the above result on $\bar{A}_w$ and your mastery of formal power sums, write $\bar{A}(D)$ in a compact form.

For a fixed code C, we will now derive a simple upper bound on its block error probability under ML decoding when transmitting over an additive white Gaussian noise channel with noise variance σ^2. More precisely, assume that the input takes values in $\{\pm 1\}$ and that to each component of the transmitted codeword an iid. Gaussian random variable with variance σ^2 is added. Let Y denote the received word and let $\hat{x}^{\mathrm{ML}}(y)$ denote the ML decoding rule. Finally, let $\mathrm{P_B}$ denote the resulting probability of block error. Justify each step in the following sequence

$$\begin{aligned}
\mathrm{P_B} &= \mathrm{P}\{\hat{x}^{\mathrm{ML}}(Y) \neq 0 \,|\, X = 0\} = \mathrm{P}\{\hat{x}^{\mathrm{ML}}(Y) \in \mathcal{C} \setminus \{0\} \,|\, X = 0\} \\
&= \mathrm{P}\{\max_{x \in \mathcal{C} \setminus \{0\}} p(Y \,|\, x) \geq p(Y \,|\, 0) \,|\, X = 0\} \\
&\leq \sum_{x \in \mathcal{C} \setminus \{0\}} \mathrm{P}\{p(Y \,|\, x) \geq p(Y \,|\, 0) \,|\, X = 0\} \\
&= \sum_{x \in \mathcal{C} \setminus \{0\}} \mathrm{P}\{|x - Y|^2 \leq |Y|^2 \,|\, X = 0\} = \sum_{x \in \mathcal{C} \setminus \{0\}} Q\left(\frac{\sqrt{w(x)}}{\sigma}\right) \\
&= \sum_{w=1}^{n} A_w Q\left(\frac{\sqrt{w}}{\sigma}\right) \leq \frac{1}{2} \sum_{w=1}^{n} A_w \mathrm{e}^{-\frac{w}{2\sigma^2}} \\
&= \frac{1}{2}\left[A(\mathrm{e}^{-\frac{1}{2\sigma^2}}) - A_0\right].
\end{aligned}$$

Collect all the previous results to conclude that the *average* block error probability for our ensemble can be bounded by

$$\mathbb{E}[\mathrm{P_B}(H)] \leq \frac{\left(1 + \mathrm{e}^{-\frac{1}{2\sigma^2}}\right)^n - 1}{2^{n-k+1}}.$$

1.21 (Sufficient Statistic).) Consider transmission of X chosen with probability $p_X(x)$ from some code C and let Y denote the received observation. Further, let $Z = f(Y)$, where $f(\cdot)$ is a given function. Prove that Z constitutes a sufficient statistic for X given Y if and only if $p_{Y|X}(y|x)$ can be brought into the form $a(x,z)b(y)$ for some suitable functions $a(\cdot,\cdot)$ and $b(\cdot)$.

1.22 (Bound on Binomials). Let $0 \le k \le m$ and $k, m \in \mathbb{N}$. Justify the following steps.

$$\frac{1}{\binom{m}{k}} = \frac{k!}{m(m-1)\cdots(m-k+1)} = \frac{k!}{m^k} \mathrm{e}^{-\sum_{i=1}^{k-1} \ln(1-i/m)} \le \frac{k!}{m^k} \mathrm{e}^{k^2/m}.$$

1.23 (Bound on Sum of Binomials). Prove the upper bound stated in (1.59).

Hint: Consider the binomial identity $\sum_{k=0}^{n} \binom{n}{k} x^k = (1+x)^n$ with $x = m/(n-m)$.

1.24 (Chain Rule). Give a proof of the chain rule $H(X,Y) = H(X) + H(Y|X)$.

Hint: Write $p_{X,Y}(x,y)$ as $p_X(x)p_{Y|X}(y|x)$.

1.25 (Non-Negativity of Mutual Information). Prove that $I(X;Y) \ge 0$.

Hint: Write $I(X;Y)$ as $-\sum_{x,y} p_{X,Y}(x,y) \log \frac{p_X(x)p_Y(y)}{p_{X,Y}(x,y)}$ and apply the Jensen's inequality (1.61).

1.26 (Fano Inequality). Prove the Fano inequality (1.49).

Hint: Define the random variable E, $E \in \{0,1\}$, which takes on the value 1 if $\hat{x}(Y) \neq X$. Expand $H(E,X|Y)$ both as $H(X|Y)+H(E|X,Y)$ as well as $H(E|Y)+H(X|E,Y)$, equate the two terms, and bound $H(E|X,Y)$, $H(E|Y)$, as well as $H(X|E,Y)$.

1.27 (The Capacity of the BSC Rederived). Start with (1.45) and show that the capacity of the BSC(ϵ) is equal to $\mathrm{C}_{\mathrm{BSC}}(\epsilon) = 1 - h_2(\epsilon)$ bits per channel use. ◇

References

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, *Optimal decoding of linear codes for minimizing symbol error rate*, IEEE Trans. Inform. Theory, 20 (1974), pp. 284–287. [33, 38, 42, 64, 66]

[2] A. Barg, *Complexity issues in coding theory*, in Handbook of Coding Theory, V. S. Pless and W. C. Huffman, eds., Elsevier Science, Amsterdam, 1998, pp. 649–754. [32, 42]

[3] A. Barg and G. D. Forney, Jr., *Random codes: Minimum distances and error exponents*, IEEE Trans. Inform. Theory, 48 (2002), pp. 2568–2573. [32, 42]

[4] A. Barg and A. McGregor, *Distance distribution of binary codes and the error probability of decoding*, IEEE Trans. Inform. Theory, 51 (2005), pp. 4237–4246. [33, 42]

[5] L. A. Bassalygo, *New upper bounds for error correcting codes*, Problemy Peredachi Informatsii, 1 (1965), pp. 41–44. [32, 43]

[6] E. R. Berlekamp, *Algebraic Coding Theory*, Aegean Park Press, revised ed., 1984. [33, 43]

[7] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, *On the inherent intractability of certain coding problems*, IEEE Trans. Inform. Theory, 24 (1978), pp. 384–386. [32, 43]

[8] R. E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge University Press, 2003. [33, 43]

[9] R. C. Bose and D. K. Ray-Chaudhuri, *On a class of error correcting binary group codes*, Info. and Control, 3 (1960), pp. 68–79. [32, 43]

[10] S. A. Cook, *The complexity of theorem proving procedures*, in 3rd STOC, 1971, pp. 151–158. [32, 43]

[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, New York, 1991. [33, 43]

[12] P. Delsarte, *An algebraic approach to the association schemes of coding theory*. Philips Research Reports Supplements no. 10, 1973. [32, 43]

[13] R. Descartes, Geometrie *1636, in A Source Book in Mathematics*, Harvard Univ. Press, Cambridge, 1969. [33, 43]

[14] I. Dumer, D. Micciancio, and M. Sudan, *Hardness of approximating the minimum distance of a linear code*, IEEE Trans. Inform. Theory, 49 (2003), pp. 22–37. [32, 43]

[15] P. Elias, *Coding for noisy channels*, in IRE International Convention Record, Mar. 1955, pp. 37–46. [32, 43]

[16] R. M. Fano, *A heuristic discussion of probabilistic decoding*, IEEE Trans. Inform. Theory, 9 (1963), pp. 64–74. [33, 43]

[17] Forney, *Review of random tree codes*. Appendix A, Final Report, Contract NAS2-3637, NASA CR73176, NASA Ames Res. Ctr., Dec. 1967. [33, 43]

[18] G. D. Forney, Jr., *Concatenated Codes*, PhD thesis, MIT, 1966. [33, 43]

[19] ———, *Concatenated Codes*, MIT Press, 1966. [33, 43]

[20] ———, *Convolutional codes I: Algebraic structure*, IEEE Trans. Inform. Theory, 16 (1970), pp. 720–738. [32, 43, 374, 385]

[21] ———, *Correction to 'Convolutional codes I: Algebraic structure'*, IEEE Trans. Inform. Theory, 17 (1971), p. 360. [32, 43, 374, 385]

[22] ———, *Structural analysis of convolutional codes via dual codes*, IEEE Trans. Inform. Theory, 19 (1973), pp. 512–518. [32, 43]

[23] ———, *The Viterbi algorithm*, IEEE Proceedings, 61 (1973), pp. 268–278. [33, 44]

[24] ———, *Convolutional codes II: Maximum-likelihood decoding*, Inform. Contr., 25 (1974), pp. 222–266. [33, 44, 374, 385]

[25] R. G. Gallager, *Low-density parity-check codes*, IRE Transactions on Information Theory, 8 (1962), pp. 21–28. [32, 44, 63, 66, 158, 168, 276, 295, 428, 432, 443, 444]

[26] ———, *A simple derivation of the coding theorem and some applications*, IEEE Trans. Inform. Theory, 11 (1965), pp. 3–18. [33, 44]

[27] ———, *Information Theory and Reliable Communication*, Wiley, 1968. [32, 33, 44]

[28] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979. [32, 44]

[29] E. N. Gilbert, *A comparison of signaling alphabets*, Bell System Tech. J., 31 (1952), pp. 504–522. [31, 44]

[30] V. Guruswami and M. Sudan, *Improved decoding of Reed-Solomon and algebraic-geometry codes*, in Proceedings 39th Symp. Foundations Computer Sci., 1998. [32, 44]

[31] R. W. Hamming, *Error detecting and error correcting codes*, Bell System Tech. J., 26 (1950), pp. 147–160. [31, 44]

[32] C. Heegard and S. B. Wicker, *Turbo Coding*, Kluwer Academic Publ., 1999. [33, 44, 64, 67]

[33] J. A. Heller, *Short constraint length convolutional codes*. Jet Prop. Lab., Space Prog. Summary 37-54, 1968. [33, 44]

[34] ———, *Improved performance of short constraint length convolutional codes*. Jet Prop. Lab., Space Prog. Summary 37-56, 1969. [33, 44]

[35] P. Henrici, *Applied and Computational Complex Analysis, Vol. 1*, John Wiley & Sons, New York, 1974. [33, 44]

[36] A. Hocquenghem, *Codes correcteurs d'erreurs*, Chiffres, 2 (1959), pp. 147–156. [32, 44]

[37] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*, IEEE Press, 1999. [33, 44, 374, 386]

[38] R. Kötter and A. Vardy, *Algebraic soft-decision decoding of Reed-Solomon codes*, IEEE Trans. Inform. Theory, 49 (2003), pp. 2809–2825. [32, 44]

[39] S. Lin and D. J. Costello, Jr., *Error Control Coding*, Prentice-Hall, 2nd ed., 2004. [33, 44]

[40] F. J. MacWilliams, *A theorem on the distribution of weights in a systematic code*, Bell System Tech. J., 42 (1963), pp. 79–94. [33, 39, 44]

[41] F. J. MacWilliams and N. J. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977. [33, 45]

[42] R. J. McEliece, *The Theory of Information and Coding : A Mathematical Framework for Communication*, Addison-Wesley, 1977. [33, 45]

[43] ———, *The 2004 Shannon lecture*, in Proc. of the IEEE Int. Symposium on Inform. Theory, Chicago, USA, June 27 – July 2 2004, IEEE. [33, 45]

[44] R. J. McEliece, E. R. Rodemich, H. Rumsey, Jr., and L. R. Welch, *New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities*, IEEE Trans. Inform. Theory, 23 (1977), pp. 157–166. [32, 45]

[45] J. K. Omura, *On the Viterbi decoding algorithm*, IEEE Trans. Inform. Theory, 15 (1969), pp. 177–179. [33, 45]

[46] P. M. Piret, *Convolutional Codes: An Algebraic Approach*, MIT Press, 1988. [33, 45, 374, 387]

[47] V. S. Pless, *Introduction to the Theory of Error-Correcting Codes*, John Wiley and Sons, 1989. [33, 45]

[48] V. S. Pless and W. C. Huffman, eds., *Handbook of Coding Theory*, Elsevier Science, Amsterdam, 1998. [31, 33, 45]

[49] I. S. Reed and G. Solomon, *Polynomial codes over certain finite fields*, J. SIAM, 8 (1960), pp. 300–304. [32, 45]

[50] I. Sason and S. Shamai, *Performance Analysis of Linear Codes under Maximum-Likelihood Decoding: A Tutorial*, vol. 3 of Foundations and Trends in Communications and Information Theory, NOW, Delft, the Netherlands, July 2006. Available online at http://www.ee.technion.ac.il/people/sason/monograph.html. [32, 45]

[51] C. Schlegel and L. C. Perez, *Trellis and Turbo Coding*, Wiley-IEEE Press, 2004. [33, 45]

[52] S. Shamai and I. Sason, *Variations on the Gallager bounds, connections, and applications*, IEEE Trans. Inform. Theory, 48 (2002), pp. 3029–3051. [32, 45]

[53] C. E. Shannon, *A mathematical theory of communication*, Bell System Tech. J., 27 (1948), pp. 379–423, 623–656. [31, 33, 45]

[54] C. E. Shannon, R. G. Gallager, and E. R. Berlekamp, *Lower bounds to error probability for coding on discrete memoryless channels - II*, Inform. Contr., 10 (1967), pp. 522–552. [33, 45]

[55] D. A. Spielman, *The complexity of error-correcting codes*, Lecture Notes in Computer Science, 1279 (1997), pp. 67–84. [32, 45]

[56] M. Sudan, *Algorithmic issues in coding theory*, in 17th Conference on Foundations of Software Technology and Theoretical Computer Science, Kharagpur, India, 1997. [32, 45]

[57] ———, *Decoding Reed-Solomon codes beyond the error-correction diameter*, in Proc. 35th Annual Allerton Conference on Communication, Control and Computing, Monticello, IL, 1997. [32, 46]

[58] J. H. van Lint, *Introduction to Coding Theory*, Springer Verlag, second ed., 1992. [31, 33, 46]

[59] A. Vardy, *The intractability of computing the minimum distance of a code*, IEEE Trans. Inform. Theory, 43 (1997), pp. 1757–1766. [32, 46]

[60] R. R. Varshamov, *Estimate of the number of signals in error correcting codes*, Doklady Akad. Nauk SSSR, 117 (1957), pp. 739–741. [31, 46]

[61] A. J. Viterbi, *Error bounds of convolutional codes and an asymptotically optimum decoding algorithm*, IEEE Trans. Inform. Theory, 13 (1967), pp. 260–269. [33, 46, 63, 68]

[62] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979. [33, 46]

[63] B. Vucetic and J. Yuan, *Turbo codes: principles and applications*, Kluwer Academic Publ., 2000. [33, 46]

[64] J. K. Wolf, *Efficient maximum likelihood decoding of linear block codes using a trellis*, IEEE Trans. Inform. Theory, 24 (1978), pp. 76–80. [33, 38, 46]

[65] J. M. Wozencraft, *Sequential decoding for reliable communication*, Research Lab. of Electronics Tech. Rept. 325, MIT, Cambridge, MA, 1957. [33, 46]

[66] H. L. Yudkin, *Channel state testing in information decoding*, PhD thesis, MIT, Cambridge, MA, 1964. [33, 46]

Chapter 2

FACTOR GRAPHS

This chapter is largely about the following question: how can we efficiently compute marginals of multivariate functions. A surprisingly large number of computational problems can be phrased in this way. The decoding problem, which is the focus of this book, is an important particular case.

§2.1. Distributive Law

Let $\mathbb{F}$ be a field (think of $\mathbb{F} = \mathbb{R}$) and let $a, b, c \in \mathbb{F}$. The *distributive law* states that

$$ab + ac = a(b + c). \tag{2.1}$$

This simple law, properly applied, can significantly reduce computational complexity: consider e.g. the evaluation of $\sum_{i,j} a_i b_j$ as $(\sum_i a_i)(\sum_j b_j)$. Factor graphs provide an appropriate framework to systematically take advantage of the distributive law.

Let's start with an example. Consider a function f with factorization

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_1, x_2, x_3) f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5). \tag{2.2}$$

We are interested in the *marginal* of f with respect to x_1. With some abuse of notation, we denote this marginal by $f(x_1)$,

$$f(x_1) \triangleq \sum_{x_2, x_3, x_4, x_5, x_6} f(x_1, x_2, x_3, x_4, x_5, x_6) = \sum_{\sim x_1} f(x_1, x_2, x_3, x_4, x_5, x_6).$$

In the previous line we introduced the notation $\sum_{\sim \dots}$ to denote a summation over all variables contained in the expression *except* the ones listed. This convention will save us from a flood of notation. Assume that all variables take values in a finite alphabet, call it $\mathcal{X}$. Determining $f(x_1)$ for all values of x_1 by brute force requires $\Theta\left(|\mathcal{X}|^6\right)$ operations, where we assume a somewhat naive computational model in which all operations (addition, multiplication, function evaluations, ...) have constant cost. But we can do better: taking advantage of the factorization, we can rewrite $f(x_1)$ as

$$f(x_1) = \Big[\sum_{x_2, x_3} f_1(x_1, x_2, x_3) \Big] \Big[\sum_{x_4, x_6} f_2(x_1, x_4, x_6) f_3(x_4) \sum_{x_5} f_4(x_4, x_5) \Big].$$

Fix x_1. The evaluation of the first factor can be accomplished with $\Theta\left(|\mathcal{X}|^2\right)$ operations. The second factor depends only on x_4, x_5 and x_6. It can be evaluated efficiently in the following manner. For each value of x_4, determine $\sum_{x_5} f_4(x_4, x_5)$ and store

the result. This has computational complexity $\Theta\left(|\mathcal{X}|^2\right)$ and requires $\Theta\left(|\mathcal{X}|\right)$ storage. Now multiply by $f_2(x_1, x_4, x_6) f_3(x_4)$ and sum over x_4 and x_6. Therefore, the evaluation of the second factor requires $\Theta\left(|\mathcal{X}|^2\right)$ operations as well. Since there are $|\mathcal{X}|$ values for x_1, the overall task has complexity $\Theta\left(|\mathcal{X}|^3\right)$. This compares very favorably to $\Theta\left(|\mathcal{X}|^6\right)$, the complexity of the brute force approach.

§2.2. Graphical Representation of Factorizations

Consider a function and its factorization. (As a running example we will use the function and factorization given in (2.2).) Associate with this factorization a *factor graph* as follows. For each variable draw a *variable node* (circle) and for each factor draw a *factor node* (square). Connect a variable node to a factor node by an *edge* if and only if the corresponding variable appears in this factor. The resulting graph for our running example is shown on the left of Figure 2.3. The factor graph is *bipartite*. This means that the set of vertices is partitioned into two groups (the set of nodes corresponding to variables and the set of nodes corresponding to factors) and that an edge always connects a variable node to a factor node. For our particular example the factor graph is a (bipartite) *tree*. This means that there are no *cycles* in the graph, i.e., there is one and only one path between each pair of nodes. As we will show in

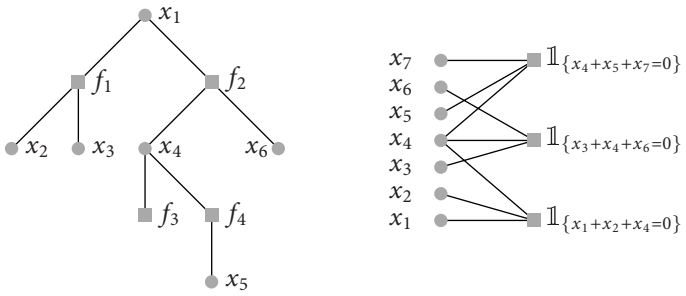

Figure 2.3: Left: Factor graph of f given in (2.2). Right: Factor graph for the code membership function defined in Example 2.4.

the next section, in this important special case there is a simple *message-passing* algorithm for computing marginals efficiently. This remains true in the slightly more general scenario where the factor graph forms a *forest*, i.e., the factor graph is disconnected and it is composed of a collection of trees. In order to keep things simple we will assume a single tree and ignore this straightforward generalization.

Example 2.4 (Special Case: Tanner Graph). Consider the binary linear code

$C(H)$ defined[1] by the parity-check matrix

$$H = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}.$$

Let $\mathbb{F}_2$ denote the binary field with elements $\{0, 1\}$. The function $f(x_1, \cdots, x_7)$, taking inputs in $\mathbb{F}_2^7$,

$$f(x_1, \cdots, x_7) \triangleq \begin{cases} 1, & \text{if } Hx^T = 0^T, \\ 0, & \text{otherwise,} \end{cases}$$

where $x = (x_1, \cdots, x_7)$, is the indicator function of $\{x \in C(H)\}$ (code membership function). We can factor f as

$$f(x_1, \ldots, x_7) = \mathbb{1}_{\{x_1+x_2+x_4=0\}} \mathbb{1}_{\{x_3+x_4+x_6=0\}} \mathbb{1}_{\{x_4+x_5+x_7=0\}}.$$

The symbol $\mathbb{1}_{\{\cdot\}}$ denotes an *indicator function*. It is 1 if the condition inside the braces is fulfilled and 0 otherwise. The corresponding factor graph is shown on the right in Figure 2.3. The factor graph for the function f, deciding code membership in the code $C(H)$, is called the *Tanner* graph of H. We will have much more to say about it in Section 3.3. ◇

It is hopefully clear at this point that *any* (binary) linear block code has a Tanner graph representation. But more is true: e.g., if you are familiar with convolutional codes take a peak at Figure 6.5. It represents a convolutional code in terms of a factor graph. Throughout the rest of this book we will encounter factor graphs for a wide range of codes and a wide range of applications.

§2.3. Recursive Determination of Marginals

Consider the factorization of a generic function g and assume that the associated factor graph is a tree (by definition it is always bipartite). Assume we are interested in the marginalization of g with respect to the variable z, i.e., we are interested in $g(z) \triangleq \sum_{\sim z} g(z, \ldots)$. Since by assumption the factor graph of g is a bipartite tree, g has a generic factorization of the form

$$g(z, \ldots) = \prod_{k=1}^{K} [g_k(z, \ldots)]$$

[1] To be precise, we mean here the code C whose parity-check matrix is H not the dual code.

for some integer K with the following crucial property: z appears in each of the factors g_k, but all other variables appear in *only one* factor. For our running example this is the factorization

$$f(x_1, \dots) = [f_1(x_1, x_2, x_3)]\,[f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5)]\,,$$

so that $K = 2$. The generic factorization and the particular instance for our running example f are shown in Figure 2.5. Taking into account that the individual factors

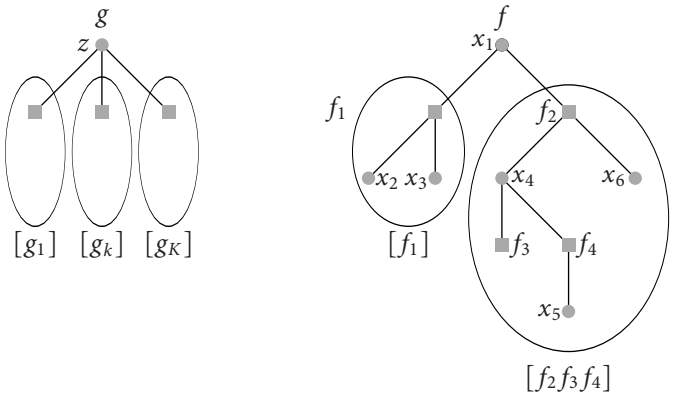

Figure 2.5: Generic factorization and the particular instance.

$g_k(z, \dots)$ only share the variable z, an application of the distributive law leads to

$$\sum_{\sim z} g(z, \dots) = \underbrace{\sum_{\sim z} \prod_{k=1}^{K} [g_k(z, \dots)]}_{\text{marginal of product}} = \underbrace{\prod_{k=1}^{K} \Big[\sum_{\sim z} g_k(z, \dots) \Big]}_{\text{product of marginals}}\,. \tag{2.6}$$

In words, the marginal $\sum_{\sim z} g(z, \dots)$ is the product of the individual marginals $\sum_{\sim z} g_k(z, \dots)$. In terms of our running example we have

$$f(x_1) = \Big[\sum_{\sim x_1} f_1(x_1, x_2, x_3) \Big] \Big[\sum_{\sim x_1} f_2(x_1, x_4, x_6) f_3(x_4) f_4(x_4, x_5) \Big].$$

This single application of the distributive law leads, in general, to a non-negligible reduction in complexity. But we can go further and apply the same idea recursively to each of the terms $g_k(z, \dots)$.

In general, each g_k is itself a product of factors. In Figure 2.5 these are the factors of g that are grouped together in one of the ellipsoids. Since the factor graph is a

bipartite tree, g_k must in turn have a generic factorization of the form

$$g_k(z,\dots) = \underbrace{h(z,z_1,\dots,z_J)}_{\text{kernel}} \prod_{j=1}^{J} \underbrace{\big[h_j(z_j,\dots)\big]}_{\text{factors}},$$

where z appears only in the "kernel" $h(z,z_1,\dots,z_J)$ and each of the z_j appears *at most twice*, possibly in the kernel and in at most one of the factors $h_j(z_j,\dots)$. All other variables are again unique to a single factor. For our running example we have for $[f_2 f_3 f_4]$

$$f_2(x_1,x_4,x_6)f_3(x_4)f_4(x_4,x_5) = \underbrace{f_2(x_1,x_4,x_6)}_{\text{kernel}} \underbrace{[f_3(x_4)f_4(x_4,x_5)]}_{x_4} \underbrace{[1]}_{x_6}.$$

The generic factorization and the particular instance for our running example f are shown in Figure 2.7. Another application of the distributive law gives

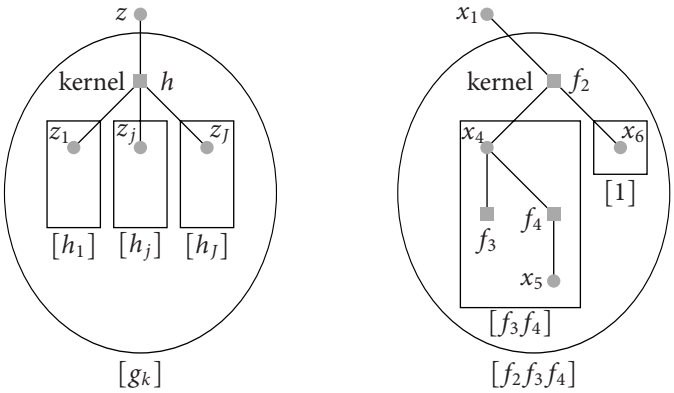

Figure 2.7: Generic factorization of g_k and the particular instance.

$$\begin{aligned}\sum_{\sim z} g_k(z,\dots) &= \sum_{\sim z} h(z,z_1,\dots,z_J) \prod_{j=1}^{J} \big[h_j(z_j,\dots)\big] \\ &= \sum_{\sim z} h(z,z_1,\dots,z_J) \underbrace{\prod_{j=1}^{J} \Big[\sum_{\sim z_j} h_j(z_j,\dots)\Big]}_{\text{product of marginals}}. \end{aligned} \tag{2.8}$$

In words, the desired marginal $\sum_{\sim z} g_k(z,\dots)$ can be computed by multiplying the kernel $h(z,z_1,\dots,z_J)$ with the individual marginals $\sum_{\sim z_j} h_j(z_j,\dots)$ and summing out all remaining variables other than z.

We are back to where we started. Each factor $h_j(z_j, \dots)$ has the same generic form as the original function $g(z, \dots)$, so that we can continue to break down the marginalization task into smaller pieces. This recursive process continues until we have reached the leaves of the tree. The calculation of the marginal then follows the resursive splitting in reverse. In general nodes in the graph compute marginals, which are functions over $\mathcal{X}$ and pass these on to the next level. In the next section we will elaborate on this method of computation, known as message-passing: the marginal functions are messages. The message combining rules at function nodes is explicit in (2.8). A variable node simply performs pointwise multiplication. Let us consider the initialization of the process. At the leaf nodes the task is simple. A function leaf node has the generic form $g_k(z)$, so that $\sum_{\sim z} g_k(z) = g_k(z)$: this means that the initial message sent by a function leaf node is the function itself. To find out the correct initialization at a variable leaf node consider the simple example of computing $f(x_1) = \sum_{\sim x_1} f(x_1, x_2)$. Here, x_2 is the variable leaf node. By the message-passing rule (2.8) the marginal $f(x_1)$ is equal to $\sum_{\sim x_1} f(x_1, x_2) \cdot \mu(x_2)$, where $\mu(x_2)$ is the initial message that we send from the leaf variable node x_2 towards the kernel $f(x_1, x_2)$. We see that in order to get the correct result this initial message should be the constant function 1.

§2.4. Efficient Marginalization Via Message Passing

In the previous section we have seen that, in the case where the factor graph is a tree, the marginalization problem can be broken down into smaller and smaller tasks according to the structure of the tree.

This gives rise to the following efficient *message-passing* algorithm. The algorithm proceeds by sending messages along the edges of the tree. Messages are *functions* on $\mathcal{X}$, or, equivalently, vectors of length $|\mathcal{X}|$. The messages signify marginals of parts of the function and these parts are combined to form the marginal of the whole function. Message-passing originates at the leaf nodes. Messages are passed up the tree and as soon as a node has received messages from all its children, the incoming messages are processed and the result is passed up to the parent node.

Example 2.9 (Message-Passing Algorithm for the Running Example). Consider this procedure in detail for the case of our running example as shown in Figure 2.10. The top left-most figure shows the factor graph. Message-passing starts at the leaf nodes as shown in the middle figure on the top. The variable leaf nodes x_2, x_3, x_5 and x_6 send the constant function 1 as discussed at the end of the previous section. The factor leaf node f_3 sends the function f_3 up to its parent node. In the next time step the factor node f_1 has received messages from both its children and can therefore proceed. According to (2.8), the message it sends up to its parent node x_1 is the product of the incoming messages times the "kernel" f_1, after summing

out all variable nodes except x_1, i.e., the message is $\sum_{\sim x_1} f_1(x_1, x_2, x_3)$. In the same manner factor node f_4 forwards to its parent node x_4 the message $\sum_{\sim x_4} f_4(x_4, x_5)$. This is shown in the right-most figure in the top row. Now variable node x_4 has received messages from all its children. It forwards to its parent node f_2 the product of its incoming messages, in agreement with (2.6), which says that the marginal of a product is the product of the marginals. This message, which is a function of x_4, is $f_3(x_4) \sum_{\sim x_4} f(x_4, x_5) = \sum_{\sim x_4} f_3(x_4) f_4(x_4, x_5)$. Next, function node f_2 can forward its message, and, finally, the marginalization is achieved by multiplying all incoming messages at the root node x_1. ◇

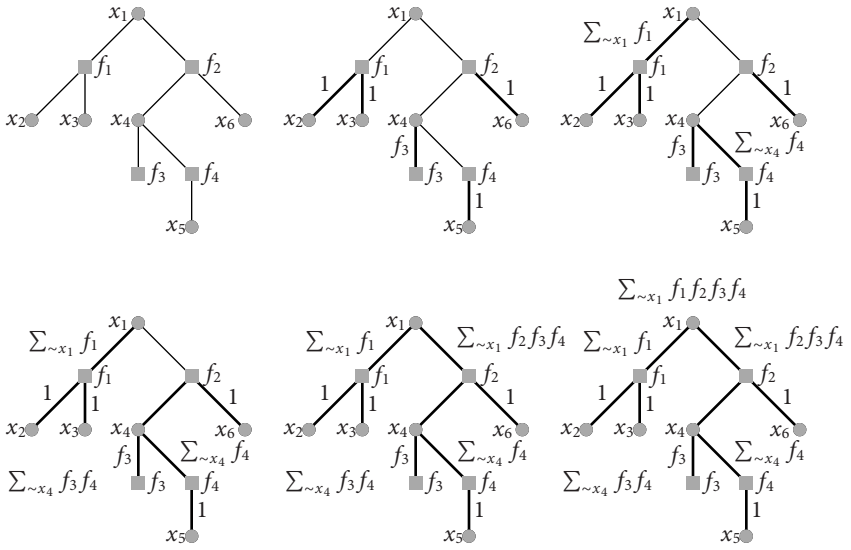

Figure 2.10: Marginalization of function f via message passing. Message passing starts at the leaf nodes. A node which has received messages from all its children processes the messages and forwards the result to its parent node. Bold edges indicate edges along which messages have already been sent.

Before stating the message-passing rules formally, there is one important generalization which we must make. Whereas so far we have considered the marginalization of a function f with respect to a *single* variable x_1 we are actually interested in marginalizing for *all* variables. We have seen that a single marginalization can be performed efficiently if the factor graph of f is a *tree*, and that the complexity of the computation essentially depends on the largest degree of the factor graph and the size of the underlying alphabet. Consider now the problem of computing *all* marginals. We can draw for each variable a tree rooted in this variable and execute

the single marginal message-passing algorithm on each rooted tree. It is easy to see, however, that the algorithm does not depend on which node is the root of the tree and that in fact all the computations can be performed simultaneously on a single tree. Simply start at all leaf nodes and for every edge compute the outgoing message along this edge as soon as you have received the incoming messages along all *other* edges that connect to the given node. Continue in this fashion until a message has been sent in both directions along every edge. This computes *all* marginals so it is more complex than computing a single marginal but only by a factor roughly equal to the average degree of the nodes. We now summarize the set of message-passing rules.

Messages, which we denote by μ, are functions on $\mathcal{X}$. Message passing starts at leaf nodes. Consider a node and one of its adjacent edges, call it $\mathtt{e}$. As soon as the *incoming* messages to the node along all *other* adjacent edges have been received these messages are processed and the result is *sent out* along $\mathtt{e}$. This process continues until messages along all edges in the tree have been processed. In the final step the marginals are computed by combining *all* messages which enter a particular variable node. The initial conditions and processing rules are summarized in Figure 2.11. Since the messages represent probabilities or *beliefs* the algorithm is also known as the *belief propagation* (BP) algorithm. From now on we will mostly refer to it under this name.

§2.5. Decoding via Message Passing

§2.5.1. Bitwise MAP Decoding

Assume we transmit over a binary-input ($X_i \in \{\pm 1\}$) memoryless ($p_{Y|X}(y|x) = \prod_{i=1}^n p_{Y_i|X_i}(y_i|x_i)$) channel using a linear code $C(H)$ defined by its parity-check matrix H and assume that codewords are chosen uniformly at random. The rule for the *bitwise* MAP decoder reads:

$$
\begin{aligned}
\hat{x}_i^{\mathrm{MAP}}(y) &= \operatorname{argmax}_{x_i \in \{\pm 1\}} p_{X_i|Y}(x_i|y) \\
\text{(law of total probability)} \qquad &= \operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} p_{X|Y}(x|y) \\
\text{(Bayes)} \qquad &= \operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} p_{Y|X}(y|x) p_X(x) \\
(2.12) \qquad &= \operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \left(\prod_j p_{Y_j|X_j}(y_j|x_j) \right) \mathbb{1}_{\{x \in C\}},
\end{aligned}
$$

where in the last step we have used the fact that the channel is memoryless and that codewords have uniform prior. In the above formulation we consider y as a constant (since it is given to the decoding algorithm as an input). Therefore, we write $\sum_{\sim x_i}$ to indicate a summation over all components of x except x_i.

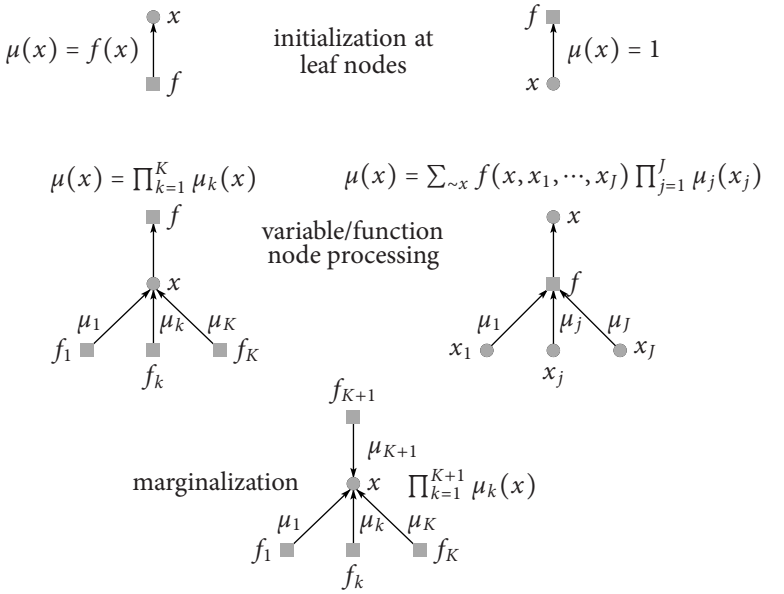

Figure 2.11: Message-passing rules. The top row shows the initialization of the messages at the leaf nodes. The middle row corresponds to the processing rules at the variable and function nodes, respectively. The bottom row explains the final marginalization step.

Assume that the code membership function $\mathbb{1}_{\{x \in \mathcal{C}\}}$ has a factorized form. From (2.12) it is then clear that the bitwise decoding problem is equivalent to calculating the marginal of a factorized function and choosing the value which maximizes this marginal.

Example 2.13 (Bitwise MAP Decoding). Consider the parity-check matrix given in Example 2.4. In this case $\operatorname{argmax}_{x_i \in \{\pm 1\}} p_{X_i \mid Y}(x_i \mid y)$ can be factorized as

$$\operatorname{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \left(\prod_{j=1}^{7} p_{Y_i \mid X_j}(y_j \mid x_j) \right) \mathbb{1}_{\{x_1 + x_2 + x_4 = 0\}} \mathbb{1}_{\{x_3 + x_4 + x_6 = 0\}} \mathbb{1}_{\{x_4 + x_5 + x_7 = 0\}}.$$

The corresponding factor graph is shown in Figure. 2.14. This graph includes the Tanner graph of H, but in addition also contains the factor nodes which represent the effect of the channel. For this particular case the resulting graph is a tree. We can therefore apply the message-passing algorithm to this example in order to perform bit MAP decoding. ◇

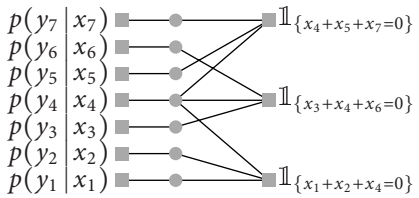

Figure 2.14: The factor graph for the MAP decoding of our running example.

§2.5.2. Simplification of Message-Passing Rules for Bitwise MAP Decoding

In the binary case a message $\mu(x)$ can be thought of as a real-valued vector of length two, $(\mu(1), \mu(-1))$ (here we think of the bit values as $\{\pm 1\}$). The initial such message emitted by variable leaf node i is $(p_{Y_i \mid X_i}(y_i \mid 1), p_{Y_i \mid X_i}(y_i \mid -1))$ (see Figure 2.14). Recall that at a variable node of degree $(K+1)$ the message-passing rule calls for a pointwise multiplication,

$$\mu(1) = \prod_{k=1}^{K} \mu_k(1), \qquad \mu(-1) = \prod_{k=1}^{K} \mu_k(-1).$$

Introduce the *ratio* $r_k \triangleq \mu_k(1)/\mu_k(-1)$. The initial such ratios are the likelihood ratios associated with the channel observations. We have

$$r = \frac{\mu(1)}{\mu(-1)} = \frac{\prod_{k=1}^{K} \mu_k(1)}{\prod_{k=1}^{K} \mu_k(-1)} = \prod_{k=1}^{K} r_k,$$

i.e., the ratio of the outgoing message at a variable node is the product of the incoming ratios. If we define the log-likelihood ratios $l_k = \ln(r_k)$, then the processing rule reads $l = \sum_{k=1}^{K} l_k$.

Consider now the ratio of an outgoing message at a check node which has degree $(J+1)$. For a check node the associated "kernel" is

$$f(x, x_1, \ldots, x_J) = \mathbb{1}_{\{\prod_{j=1}^{J} x_j = x\}}.$$

Since in the current context we assume that the x_i take values in $\{\pm 1\}$ (and not $\mathbb{F}_2$) we wrote $\prod_{j=1}^{J} x_j = x$ (instead of $\sum_{j=1}^{J} x_j = x$). We therefore have

$$\begin{aligned} r = \frac{\mu(1)}{\mu(-1)} &= \frac{\sum_{\sim x} f(1, x_1, \ldots, x_J) \prod_{j=1}^{J} \mu_j(x_j)}{\sum_{\sim x} f(-1, x_1, \ldots, x_J) \prod_{j=1}^{J} \mu_j(x_j)} \\ &= \frac{\sum_{x_1,\ldots,x_J : \prod_{j=1}^{J} x_j = 1} \prod_{j=1}^{J} \mu_j(x_j)}{\sum_{x_1,\ldots x_J : \prod_{j=1}^{J} x_j = -1} \prod_{j=1}^{J} \mu_j(x_j)} = \frac{\sum_{x_1,\ldots,x_J : \prod_{j=1}^{J} x_j = 1} \prod_{j=1}^{J} \frac{\mu_j(x_j)}{\mu_j(-1)}}{\sum_{x_1,\ldots,x_J : \prod_{j=1}^{J} x_j = -1} \prod_{j=1}^{J} \frac{\mu_j(x_j)}{\mu_j(-1)}} \end{aligned}$$

$$= \frac{\sum_{x_1,\ldots,x_J:\prod_{j=1}^J x_j=1} \prod_{j=1}^J r_j^{(1+x_j)/2}}{\sum_{x_1,\ldots,x_J:\prod_{j=1}^J x_j=-1} \prod_{j=1}^J r_j^{(1+x_j)/2}} = \frac{\prod_{j=1}^J (r_j+1) + \prod_{j=1}^J (r_j-1)}{\prod_{j=1}^J (r_j+1) - \prod_{j=1}^J (r_j-1)}. \tag{2.15}$$

The last step warrants some remarks. If we expand out $\prod_{j=1}^J (r_j+1)$, then we get the sum of all products of the individual terms r_j, $j = 1, \cdots, J$. E.g., $\prod_{j=1}^3 (r_j+1) = 1 + r_1 + r_2 + r_3 + r_1 r_2 + r_1 r_3 + r_2 r_3 + r_1 r_2 r_3$. Similarly, $\prod_{j=1}^J (r_j - 1)$ is the sum of all products of the individual terms r_j, where all products consisting of d terms such that $J - d$ is odd have a negative sign. E.g., we have $\prod_{j=1}^3 (r_j - 1) = -1 + r_1 + r_2 + r_3 - r_1 r_2 - r_1 r_3 - r_2 r_3 + r_1 r_2 r_3$. From this follows that

$$\prod_{j=1}^J (r_j+1) + \prod_{j=1}^J (r_j-1) = 2 \sum_{x_1,\ldots,x_J:\prod_{j=1}^J x_j=1} \prod_{j=1}^J r_j^{(1+x_j)/2}.$$

Applying the analogous reasoning to the denominator, the equality follows. If we divide both numerator and denominator by $\prod_{j=1}^J (r_j+1)$, we see that (2.15) is equivalent to the statement $r = \frac{1+\prod_j \frac{r_j-1}{r_j+1}}{1-\prod_j \frac{r_j-1}{r_j+1}}$, which in turn implies $\frac{r-1}{r+1} = \prod_j \frac{r_j-1}{r_j+1}$. From $r = \mathsf{e}^l$ we see that $\frac{r-1}{r+1} = \tanh(l/2)$. Combining theses two statements we have $\tanh(l/2) = \frac{r-1}{r+1} = \prod_{j=1}^J \frac{r_j-1}{r_j+1} = \prod_{j=1}^J \tanh(l_j/2)$, so that

$$l = 2\tanh^{-1}\left(\prod_{j=1}^J \tanh(l_j/2)\right). \tag{2.16}$$

We summarize: in the case of transmission over a binary channel the messages can be compressed to a single real quantity. In particular, if we choose this quantity to be the log-likelihood ratio (log of the ratio of the two likelihoods) then the processing rules take on a particularly simple form: at variables nodes messages add, and at check nodes the processing rule is stated in (2.16).

§2.5.3. Forney-Style Factor Graphs

Factor graphs (FG) represent one particular language to formulate the relationship between a function and its local components. One popular alternative is the representation in terms of *Forney-style factor graphs* (FSFG). These graphs are sometimes also called *normal graphs*.

Consider the FG shown in the left hand side of Figure 2.17. Note that each variable node has degree one or two. We can therefore convert the *bipartite* graph into a *regular* (in the sense of not bipartite) graph by representing variable nodes as

Figure 2.17: Left: A standard FG in which each variable node has degree at most two. Right: The equivalent FSFG. The variables in the FSFG are associated with the edges in the FG.

(half)edges. The result is shown on the right hand side of Figure 2.17. This is the FSFG representation. In general, a variable node might have degree larger than two. In this case it is easy to *replicate* such a variable node a sufficient number of times by an *equality factor* as shown in Figure 2.18. The left side shows a variable node of degree $K + 1$. The right side shows the representation as an FSFG. The K additional variables $x_1, \cdots, x_K$ are enforced to be equal to the original variable x by an "equality factor", i.e., $x = x_1 = \cdots = x_K$. Figure 2.19 compares the standard FG for the MAP

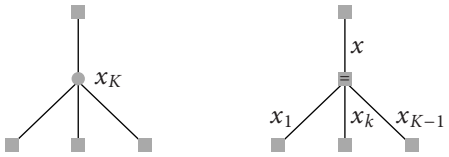

Figure 2.18: Representation of a variable node of degree K as a FG (left) and the equivalent representation as an FSFG (right).

decoding problem of our running example with the corresponding FSFG.

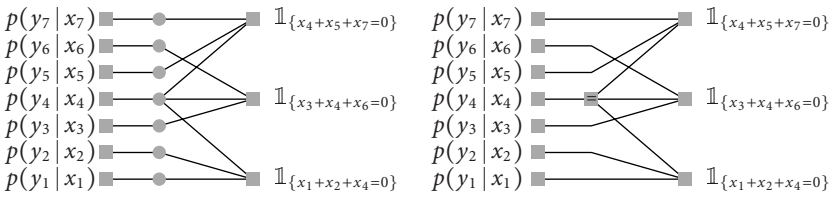

Figure 2.19: The standard FG and the corresponding FSFG for the MAP decoding problem of our running example.

The relationship between the standard FG and the FSFG is straightforward and little effort is required to move from one representation to the other. The message-passing rules carry over verbatim. In fact, in the setting of FSFGs we only need the factor node processing rule: for a generic node of degree $(J + 1)$, the outgoing mes-

sage along edge x is

$$\mu(x) = \sum_{\sim x} f(x, x_1, \cdots, x_J) \prod_{j=1}^{J} \mu_j(x_j). \tag{2.20}$$

Recall that variables in the FSFG are represented by edges. Some thought shows that in order to compute the final marginalization with respect to a certain variable, we need to multiply the two messages that flow along the corresponding edge.

FSFGs have fewer nodes and are therefore typically simpler. Note that "internal" edges represent internal or "state" variables, whereas "half-edges" (like the bottom edge on the right of Figure 2.17) represent external variables which can be connected to other systems on graphs.

§2.5.4. Generalization to Commutative Semirings

We started with a discussion of the distributive law assuming that the underlying algebraic structure was a field $\mathbb{F}$ and used it to derive efficient marginalization algorithms. A closer look at our derivation shows that actually all that was needed was the fact that we were working in a *commutative semiring*. In a commutative semiring $\mathbb{K}$ the two operations, which we call "+" and "$\cdot$" satisfy the following axioms: (i) the operations "+" and "$\cdot$" are commutative ($x + y = y + x$; $x \cdot y = y \cdot x$) and associative ($x + (y + z) = (x + y) + z$; $x \cdot (y \cdot z) = (x \cdot y) \cdot z$) with identity elements denoted by "0" and "1", respectively; (ii) the *distributive law* $(x + y) \cdot z = (x \cdot z) + (y \cdot z)$ holds for any triple $x, y, z \in \mathbb{K}$. In comparison to a field $\mathbb{F}$ we do not require the existence of an inverse with respect to either operation.

Table 2.21 lists several commutative semirings which are useful in the context of coding. The task of verifying that each of these examples indeed fulfills the stated axioms is relegated to Problem 2.1. The most important example in the context of iterative coding is the so-called *sum-product* semiring, in which the two operations are standard addition and multiplication (this is the example which we have used so far). As we will see shortly, this is the relevant semiring if we want to minimize the *bit* error probability.

The second most important example is the *max-sum* semiring. Again we operate over the reals but now addition is replaced by maximization and multiplication is replaced by addition. All previous statements and algorithms stay valid if we perform this simple substitution of operations. We will soon see that the max-sum semiring is the proper setting for performing *block* decoding. To be concrete, consider the distributive law for the max-sum semiring. If in (2.1) we replace addition with maximization and multiplication with addition then we get

$$\max\{x + y, x + z\} = x + \max\{y, z\},$$

$\mathbb{K}$	"$(+,0)$"	"$(\cdot,1)$"	description
$\mathbb{F}$	$(+,0)$	$(\cdot,1)$	
$\mathbb{F}[x,y,\cdots]$	$(+,0)$	$(\cdot,1)$	
$\mathbb{R}_{\geq 0}$	$(+,0)$	$(\cdot,1)$	sum-product
$\mathbb{R}_{>0}\cup\{\infty\}$	$(\min,\infty)$	$(\cdot,1)$	min-product
$\mathbb{R}_{\geq 0}$	$(\max,0)$	$(\cdot,1)$	max-product
$\mathbb{R}\cup\{\infty\}$	$(\min,\infty)$	$(+,0)$	min-sum
$\mathbb{R}\cup\{-\infty\}$	$(\max,-\infty)$	$(+,0)$	max-sum
$\{0,1\}$	$(\texttt{OR},0)$	$(\texttt{AND},1)$	Boolean

Table 2.21: A list of commutative semirings which are relevant for iterative coding. The entry $\mathbb{F}[x,y,\cdots]$ denotes the set of polynomials in the variables $x,y,\ldots$, with coefficients in the field $\mathbb{F}$, and the usual polynomial arithmetic.

and, more generally,

$$\max_{i,j}\{x_i+y_j\}=\max_i\{x_i\}+\max_j\{y_j\}.$$

What is the *marginalization* of a function $f(x_1,\cdots,x_n)$ of n real variables in the context of the max-sum semiring? By replacing the operations we see that it is *maximization*, i.e.,

$$f(x_1)\triangleq\max_{x_2,\cdots,x_n} f(x_1,\cdots,x_n)=\max_{\sim x_1} f(x_1,\cdots,x_n). \tag{2.22}$$

As before, if the factor graph of the function $f(x_1,\cdots,x_n)$ is a tree, this maximization can be accomplished efficiently by a message-passing algorithm operating over the max-sum semiring. The message-passing rules are formally identical. More precisely: the original variable node processing rule $\mu(z)=\prod_{k=1}^K\mu_k(z)$ is transformed into the rule $\mu(z)=\sum_{k=1}^K\mu_k(z)$, and the function node processing rule, which previously was $\mu(z)=\sum_{\sim z} f(z,z_1,\cdots,z_J)\prod_{j=1}^J\mu_j(z_j)$, now reads

$$\mu(z)=\max_{\sim z}\Big\{f(z,z_1,\cdots,z_J)+\sum_{j=1}^J\mu_j(z_j)\Big\}.$$

The final marginalization step, which used to consist of computing the product $\prod_{k=1}^{K+1}\mu_k(z)$, now requires to evaluate the sum $\sum_{k=1}^{K+1}\mu_k(z)$.

§2.5.5. Blockwise MAP Decoding

Assume we are transmitting over a binary memoryless channel using a linear code $C(H)$ defined by its parity-check matrix H and assume that codewords are chosen uniformly at random from $C(H)$. The processing rule for the optimum block

decoder is

$$\text{(Bayes)} \qquad \hat{x}^{\mathrm{MAP}}(y) = \operatorname{argmax}_x p_{X|Y}(x\,|\,y) = \operatorname{argmax}_x p_{Y|X}(y\,|\,x)p_X(x)$$

$$\text{(memoryless)} \qquad = \operatorname{argmax}_x \left(\prod_j p_{Y_j|X_j}(y_j\,|\,x_j) \right) \mathbb{1}_{\{x\in C\}}.$$

To emphasize the similarity to the optimum bit decoder, consider the i-th bit of $\hat{x}^{\mathrm{MAP}}(y)$, write it as $(\hat{x}^{\mathrm{MAP}}(y))_i$. We have

$$\begin{aligned}\big(\hat{x}^{\mathrm{MAP}}(y)\big)_i &= \operatorname{argmax}_{x_i\in\{\pm1\}} \max_{\sim x_i} \left(\prod_j p_{Y_j|X_j}(y_j\,|\,x_j) \right) \mathbb{1}_{\{x\in C\}} \\ &= \operatorname{argmax}_{x_i\in\{\pm1\}} \max_{\sim x_i} \sum_j \log p_{Y_j|X_j}(y_j\,|\,x_j) + \log\big(\mathbb{1}_{\{x\in C\}}\big).\end{aligned}$$

If we compare this with (2.12) we see that the two criteria only differ by a substitution of the two basic operations – addition goes into maximization and multiplication goes into addition (the initial messages are of course also different – we use likelihoods for bitwise decoding and log-likelihoods for blockwise decoding). Therefore, blockwise decoding can be accomplished if we employ the max-sum algebra instead of the sum-product algebra.

It is common to write the blockwise decoder in the equivalent form

$$\big(\hat{x}^{\mathrm{MAP}}(y)\big)_i = \operatorname{argmin}_{x_i\in\{\pm1\}} \min_{\sim x_i} \sum_{j=1}^{n} -\log p_{Y_j|X_j}(y_j\,|\,x_j) - \log\big(\mathbb{1}_{\{x\in C\}}\big).$$

If the channel output is discrete, so that we deal with probability mass functions, then this form is more convenient since the involved metric $-\log p_{Y_j|X_j}(y_j\,|\,x_j)$ is positive. Formally this means that we use the min-sum algebra instead of the max-sum algebra. In the sequel we adhere to this custom and use the min-sum algebra for optimum block decoding.

§2.6. Limitations of Cycle-Free Codes

The previous sections have shown a way of performing MAP decoding efficiently, assuming that the corresponding Tanner graph is a tree. Unfortunately, the class of codes which admit a tree-like (binary) Tanner graph is not powerful enough to perform well.

Lemma 2.23 (Bad News about Cycle-Free Codes). Let C be a *binary* linear code of rate r which admits a binary Tanner graph that is a forest. Then C contains at least $\frac{2r-1}{2}n$ codewords of weight 2.

Proof. Without loss of generality we can assume that the Tanner graph is connected. Otherwise the code $C = C[n,k]$ is of the form $C = C_1 \times C_2$, where $C_1 = C_1[n_1,k_1]$, $C_2 = C_2[n_2,k_2]$, $n = n_1 + n_2$, $n_1, n_2 \geq 1$, and $k = k_1 + k_2$, i.e., each codeword is the concatenation of a codeword from C_1 with a codeword from C_2. Applying the bound to each component (to keep things simple we assume there are only two such components),

$$\begin{aligned}\frac{2r_1-1}{2}n_1 + \frac{2r_2-1}{2}n_2 &= \frac{2\frac{k_1}{n_1}-1}{2}n_1 + \frac{2\frac{k_2}{n_2}-1}{2}n_2 \\ &= \frac{2k_1-n_1}{2} + \frac{2k_2-n_2}{2}n_2 = \frac{2k-n}{2} = \frac{2r-1}{2}n.\end{aligned}$$

Let us therefore assume that the Tanner graph of the code consists of a single tree. The graph has n variable nodes and $(1-r)n$ check nodes since by the tree property all check nodes (i.e., the respective equations) are linearly independent. The total number of nodes in the tree is therefore $(2-r)n$. Again by the tree property, there are $(2-r)n-1 < (2-r)n$ edges in this graph. Since each such edge connects to exactly one variable node, the average variable node degree is upper bounded by $2-r$. It follows that there must be at least nr variable nodes which are leaf nodes, since each internal variable node has degree at least 2. Since there are in total $(1-r)n$ check nodes and since every leaf variable node is connected to exactly one check node, it follows that at least $rn - (1-r)n = (2r-1)n$ leaf variable nodes are connected to check nodes which are adjacent to multiple leaf variable nodes. Each such variable node can be paired-up with one of the other such leaf nodes to give rise to a weight-two codeword. □

We see that cycle-free codes (of rate above one-half) necessarily contain many low-weight codewords and, hence, have a large probability of error. This is bad news indeed. As discussed in more detail in Problems 4.50 and 4.51, also codes of rate below one-half necessarily contain low-weight codewords, and the problem persists even if we allow a small number of cycles.

§2.7. Message-Passing on Codes with Cycles

We started with an efficient algorithm to compute the marginals of functions whose factor graph is a tree. Next we saw that the decoding task can be phrased as such a marginalization problem, both for minimizing bit or block error probability. But we now know that codes with a cycle-free (binary) Tanner graph are not powerful enough for transmission at low error rates. Tanner graphs of good codes necessarily have many cycles. So how shall we proceed?

First, one can resort to more powerful graphical models. We discuss in Chapter 6 (terminated) convolutional codes. Although terminated convolutional codes

are linear block codes (with a particular structure) and have therefore a standard binary Tanner graph representation, we will see that convolutional codes possess a cycle-free representation (and therefore the BP algorithm can be used to perform MAP decoding) if we allow *state* nodes. By increasing the size of the allowed state space one can approach capacity. However, these state nodes come at the price of increased decoding complexity and as discussed in the introduction, the complexity-gap trade-off is not very favorable. Another possibility is to consider non-binary codes. Unfortunately, complexity is again increased considerably by allowing non-binary alphabets. Finally, one can define the message-passing algorithm even in the case where cycles are present. Except for some degenerate cases, message passing in the presence of cycles is strictly suboptimal, see Problem 3.11. But as we will see in Chapters 3 and 4, excellent performance can be achieved. For codes with cycles message-passing no longer performs maximum-likelihood decoding. We will therefore spend a considerable effort on learning tools that allow us to determine the performance of such a combination.

Notes

Tanner proposed in [36] to represent codes as bipartite graphs and to visualize iterative decoding as a message-passing algorithm on such a graph. The framework of factor graphs discussed in this chapter is the result of a collaborative effort by Wiberg [38], Wiberg, Loeliger, and Kötter [39, 38], as well as Kschischang, Frey, and Loeliger [23]. It is not the only graphical model suitable for iterative decoding. Indeed, we have discussed the notion of Forney-style factor graphs in Section 2.5.3. These were introduced by Forney [14], who called them *normal* graphs. As shown in [14], normal graphs allow for an elegant local *dualization* of the graph. Extensions of this idea were discussed by Mao and Kschischang [30]. A further equivalent graphical language was put forth around the same time by Aji and McEliece [1] (see also the article by Shafer and Shenoy [35].) The message-passing algorithm which we derived via the factor-graph approach is known under many different names (iterative decoding, belief-propagation, message-passing, probabilistic decoding, ...). It was gradually realized that, what might appear as different algorithms (invented in many different communities), are in fact special cases of the same basic principle. Let us trace here just a few of those instances. Probably the first such instance is the transfer-matrix method of statistical mechanics. It is explored in detail in [9] in conjunction with the so-called *Bethe Ansatz* [10, 11] and it goes back at least to the 1930s. In the setting of communications, it was Gallager [16] who introduced LDPC codes and the related message-passing algorithm in 1960. Viterbi introduced his so-called Viterbi algorithm for the decoding of convolutional codes [37]. (The connection between the Viterbi algorithm and message-passing decoding is discussed in detail in

Section 6.1 and Problem 6.2.)

In the mid-sixties, Baum and Welch developed an algorithm to estimate the parameters of hidden Markov models. This algorithm is known as the *Baum-Welch* algorithm. For a list of publications we refer the reader to the papers by Baum and Petrie [6], Baum and Sell [8], and Baum, Petrie, Soules, and Weiss [7]. One can apply the Baum-Welch algorithm to perform bit MAP decoding of a convolutional code. This was first done by Bahl, Cocke, Jelinek, and Raviv [2] and the algorithm is now known as the *BCJR* algorithm. In 1977, Dempster, Laird, and Rubin investigated the *expectation-maximization* (EM) algorithm [12] which in turn includes the Baum-Welch algorithm as a special case, see [32].

In 1983 Kim and Pearl introduced the *belief propagation* algorithm [21, 34] to solve statistical inference problems. That the turbo decoding algorithm is in fact an instance of belief propagation was realized by MacKay and Neal in [29] and also by Frey and Kschischang [15].

An in-depth discussion of all these connections can be found in the article of McEliece, MacKay, and Cheng [31], the article of Kschischang and Frey [22] as well as the book of Heegard and Wicker [20].

Our exposition of the factor graph approach follows closely the one in [23]. The generalization of the approach to semi-rings is due to Aji and McEliece [1]. As we have seen, this generalization makes it possible to view a large class of algorithms simply as special instances of the same principle.

A set of applications for the factor graph framework is discussed in the paper by Worthen and Stark [40]. If you are looking for tutorial articles concerning factor graphs we recommend the paper by Loeliger [25].

It was shown by Etzion, Trachtenberg, and Vardy [13] that binary codes which possess a cycle-free Tanner graph (without state nodes) necessarily have small minimum distance. We discussed only the simple case of codes with rate r at least one half. In this case we saw that the minimum distance is at most 2. If $r < \frac{1}{2}$, then the above authors showed that the minimum distance is at most $2/r$.

Battail, Decouvelaere, and Godlewski were early pioneers in the area of combining "soft information" stemming from various partial descriptions of a code into one final estimate [4]. However, they did not discuss the notion of feedback, i.e., iterative decoding. Battail, Decouvelaere, and Godlewski termed their coding method *replica* coding, see also [5, 3]. Hagenauer, Offer, and Papke [18] introduced the "log-likelihood algebra", which contains the message-passing rule at variable and check nodes.

The factor-graph approach has also inspired an implementation of message-passing decoding by analog computation. This has been pioneered by two research groups, Hagenauer, Winklhofer, Offer, Méasson, Mörz, Gabara, and Yan [19, 17, 33], as well as Loeliger, Lustenberger, Helfenstein, and Tarköy [26, 27, 28].

Problems

2.1 (Factor Graphs for Semirings). Consider the examples listed in Table 2.21. Show in each case that it forms indeed a commutative semirings.

2.2 (Message-Passing Algorithm for the BEC). Starting from the message-passing rules summarized in Figure 2.11, derive the decoding algorithm for the binary erasure channel (BEC) (see Section 3.1 for a discussion of this channel model). What is the message alphabet and what are the computation rules? Simplify the rules as far as possible.

2.3 (Min-Sum Algorithm for BEC). Apply the min-sum algebra to the decoding of LDPC ensembles over the BEC. What are the initial messages and what are the processing rules? Show that the messages which are a priori two-tuples can be compressed into a single number. Finally, show that the resulting message-passing rules are identical to the ones using the sum-product semiring. In words, over the BEC (locally optimal) iterative bit and blockwise decoding are identical.

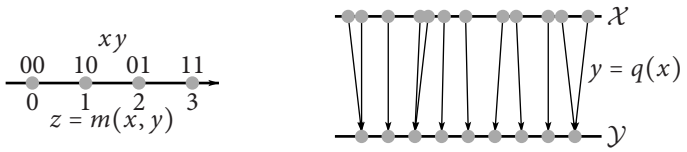

Figure 2.24: Left: Mapping $z = m(x, y)$. Right: Quantizer $y = q(x)$.

2.4 (Message Passing for Mappers – Loeliger [24]). Assume that the two binary symbols x and y are mapped by a function m into one 4-AM symbol, call it z, as shown on the left of Figure 2.24. In more detail, $m : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. Such a mapper is e.g. useful as part of a multilevel transmission scheme. Draw the corresponding FSFG. Starting from the general message-passing rule stated in (2.20) and assuming that the incoming messages are $\mu_{x,m}(x)$, $\mu_{y,m}(y)$ and $\mu_{z,m}(z)$, respectively, what are the outgoing messages $\mu_{m,x}(x)$, $\mu_{m,y}(y)$ and $\mu_{m,z}(z)$?

2.5 (Message Passing for Quantizers – Loeliger [24]). Consider a quantizer as shown on the right in Figure 2.24. More precisely, let $\mathcal{X}$ be a finite input alphabet and $\mathcal{Y}$ be a finite output alphabet at let q be the quantization function, $q : \mathcal{X} \to \mathcal{Y}$. Draw the corresponding FSFG. Starting from the general message-passing rule stated in (2.20) and assuming that the incoming messages are $\mu_{x,q}(x)$ and $\mu_{y,q}(y)$, respectively, what are the outgoing messages $\mu_{q,x}(x)$ and $\mu_{q,y}(y)$?

References

[1] S. M. Aji and R. J. McEliece, *The generalized distributive law*, IEEE Trans. Inform. Theory, 46 (2000), pp. 325–343. [63, 64, 66]

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, *Optimal decoding of linear codes for minimizing symbol error rate*, IEEE Trans. Inform. Theory, 20 (1974), pp. 284–287. [33, 38, 42, 64, 66]

[3] G. Battail, *Building long codes by combination of simple ones, thanks to weighted-output decoding*, in Proc. URSI ISSSE, Sept. 1989, pp. 634–637. [64, 66]

[4] G. Battail, M. Decouvelaere, and P. Godlewski, *Replication decoding*, IEEE Trans. Inform. Theory, 25 (1979), pp. 332–345. [64, 66, 288, 294]

[5] G. Battail and M. S. El-Sherbini, *Coding for radio channels*, Ann. Télécommun., 37 (1982), pp. 75–96. [64, 66]

[6] L. E. Baum and T. Petrie, *Statistical inference for probabilistic functions of finite state Markov chains*, Ann. Math. Stat., 37 (1966), pp. 1554–1536. [64, 66]

[7] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Ann. Math. Stat., 41 (1970), pp. 164–171. [64, 66]

[8] L. E. Baum and G. R. Sell, *Growth transformations for functions on manifolds*, Pac. J. Math., 27 (1968), pp. 211–227. [64, 66]

[9] R. J. Baxter, *Exactly Solved Models in Statistical Mechanics*, Academic Press, London, 1982. [63, 66]

[10] H. A. Bethe, *On the theory of metals, I. Eigenvalues and eigenfunctions of a linear chain of atoms*, Zeits. f. Physik, 71 (1931), pp. 205–226. [63, 66]

[11] ———, *Statistical theory of superlattices*, Proc. Roy. Soc., A150 (1935), pp. 552–75. [63, 66]

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society B, 39 (1977), pp. 1–38. [64, 66]

[13] T. Etzion, A. Trachtenberg, and A. Vardy, *Which codes have cycle-free Tanner graphs?*, IEEE Trans. Inform. Theory, 45 (1999), pp. 2173 – 2181. [64, 66]

[14] G. D. Forney, Jr., *Codes on graphs: Normal realizations*, IEEE Trans. Inform. Theory, 47 (2001), pp. 520–548. [63, 66]

[15] B. J. Frey and F. R. Kschischang, *Probability propagation and iterative decoding*, in Allerton Conf. on Communication, Control and Computing, Sept. 1996. [64, 66]

[16] R. G. Gallager, *Low-density parity-check codes*, IRE Transactions on Information Theory, 8 (1962), pp. 21–28. [32, 44, 63, 66, 158, 168, 276, 295, 428, 432, 443, 444]

[17] J. Hagenauer, E. Offer, C. Méasson, and M. Mörz, *Decoding and equalization with analog non-linear networks*, European Trans. on Telecommunications (ETT), (1999), pp. 107–128. [64, 67]

[18] J. Hagenauer, E. Offer, and L. Papke, *Iterative decoding of binary block and convolutional codes*, IEEE Trans. Inform. Theory, 42 (1996), pp. 429–445. [64, 67]

[19] J. Hagenauer and M. Winklhofer, *The analog decoder*, in Proc. of the IEEE Int. Symposium on Inform. Theory, Cambridge, MA, USA, August 16–21 1998, p. 145. [64, 67]

[20] C. Heegard and S. B. Wicker, *Turbo Coding*, Kluwer Academic Publ., 1999. [33, 44, 64, 67]

[21] J. H. Kim and J. Pearl, *A computational model for causal and diagnostic reasoning in inference systems*, in IJCAI, 1983, pp. 190–193. [64, 67]

[22] F. R. Kschischang and B. J. Frey, *Iterative decoding of compound codes by probability propagation in graphical models*, IEEE Journal on Selected Areas in Communications, (1998), pp. 219–230. [64, 67]

[23] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, *Factor graphs and the sum-product algorithm*, IEEE Trans. Inform. Theory, 47 (2001), pp. 498–519. [63, 64, 67]

[24] H.-A. Loeliger, *Some remarks on factor graphs*, in Proc. of the 3rd Int. Symp. on Turbo Codes and Related Topics, Brest, France, Sept. 2003, pp. 111–115. [65, 67]

[25] ———, *An introduction to factor graphs*, Signal Processing Magazine, 21 (2004), pp. 28–41. [64, 67]

[26] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarköy, *Probability propagation and decoding in analog VLSI*, in Proc. of the IEEE Int. Symposium on Inform. Theory, Cambridge, MA, USA, August 16–21 1998, p. 146. [64, 67]

[27] ———, *Decoding in analog VLSI*, IEEE Commun. Mag., (1999), pp. 99–101. [64, 67]

[28] ———, *Probability propagation and decoding in analog VLSI*, IEEE Trans. Inform. Theory, 47 (2001), pp. 837–843. [64, 67]

[29] D. J. C. MacKay and R. M. Neal, *Good codes based on very sparse matrices*, in Cryptography and Coding. 5th IMA Conference, C. Boyd, ed., no. 1025 in Lecture Notes in Computer Science, Springer, Berlin, 1995, pp. 100–111. [64, 67, 428, 433]

[30] Y. Mao and F. R. Kschischang, *On factor graphs and the Fourier transform*, IEEE Trans. Inform. Theory, 51 (2005), pp. 1635–1649. [63, 67]

[31] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, *Turbo decoding as an instance of Pearl's 'belief propagation' algorithm*, IEEE J. Select. Areas Commun., 16 (1998), pp. 140–152. [64, 67]

[32] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, John Wiley & Sons, New York, 1997. [64, 68]

[33] M. Mörz, T. Gabara, R. Yan, and J. Hagenauer, *An analog 0.25 μm BiCMOS tailbiting MAP decoder*, in Proc. of the IEEE Int. Solid-State Circuits Conference, San Francisco, CA, USA, Feb. 2000, pp. 356–357. [64, 68]

[34] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, San Mateo, 1988. [64, 68]

[35] G. R. Shafer and P. P. Shenoy, *Probability propagation*, Ann. Math. Art. Intel., 2 (1990), pp. 327–352. [63, 68]

[36] R. M. Tanner, *A recursive approach to low complexity codes*, IEEE Trans. Inform. Theory, 27 (1981), pp. 533–547. [63, 68, 160, 170, 274, 298]

[37] A. J. Viterbi, *Error bounds of convolutional codes and an asymptotically optimum decoding algorithm*, IEEE Trans. Inform. Theory, 13 (1967), pp. 260–269. [33, 46, 63, 68]

[38] N. Wiberg, *Codes and Decoding on General Graphs*, PhD thesis, Linköping University, S-581 83, Linköping, Sweden, 1996. [63, 68, 278, 299, 375, 388]

[39] N. Wiberg, H.-A. Loeliger, and R. Kötter, *Codes and iterative decoding on general graphs*, European Transactions on Telecommuncations, 6 (1995), pp. 513–526. [63, 68]

[40] A. P. Worthen and W. E. Stark, *Unified design of iterative receivers using factor graphs*, IEEE Trans. Inform. Theory, 47 (2001), pp. 843–850. [64, 68]