Essential Coding Theory

Venkatesan Guruswami Atri Rudra[1] Madhu Sudan

January 9, 2026

[1]Department of Computer Science and Engineering, University at Buffalo, SUNY. Work supported by NSF CAREER grant CCF-0844796.

Foreword

This book is based on lecture notes from coding theory courses taught by Venkatesan Guruswami at University at Washington and CMU; by Atri Rudra at University at Buffalo, SUNY and by Madhu Sudan at Harvard and MIT.

This version is dated **January 9, 2026**. For the latest version, please go to

http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/book/

The material in this book is supported in part by the National Science Foundation under CAREER grant CCF-0844796. Any opinions, findings and conclusions or recomendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

Contents

List of Figures

Part I

The Basics

Chapter 1

The Fundamental Question

1.1 Overview

Communication is a fundamental need of our modern lives. In fact, communication is something that humans have been doing for a long time. For simplicity, let us restrict ourselves to English. It is quite remarkable that different people speaking English can be understood pretty well: even if e.g. the speaker has an accent. This is because English has some built-in redundancy, which allows for "errors" to be tolerated. We will pick an example from one of the author's experiences conversing with his two-year-old son, Akash. When Akash started to speak his own version of English, which we will dub "Akash English," we got examples such as the one illustrated below:

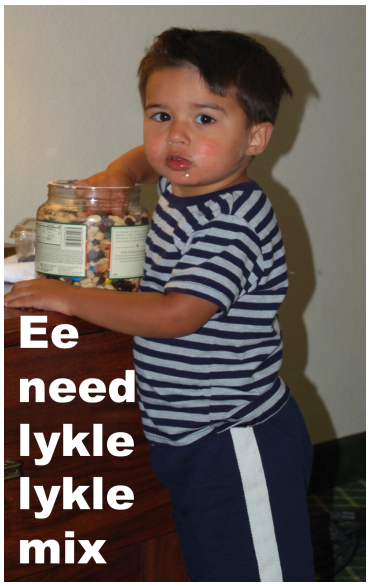

Figure 1.1: Decoding for Akash English, one gets "I need little little (trail)mix."

With some practice Akash's parents were able to "decode" what Akash really meant. In fact, Akash could communicate even if he did not say an entire word properly and gobbled up part(s) of word(s).

The above example shows that having redundancy in a language allows for communication even in the presence of (small amounts of) differences and errors. Of course, in our modern digital world, all kinds of entities communicate (and most of the entities do not communicate in English, or any natural language for that matter). Errors are also present in the digital world, so these digital communications also use redundancy.

Error-correcting codes (henceforth, just codes) are clever ways of representing data so that one can recover the original information even if parts of it are corrupted. The basic idea is to judiciously introduce redundancy so that the original information can be recovered even when parts of the (redundant) data have been corrupted.

For example, when packets are transmitted over the Internet, some of the packets get corrupted or dropped. Packet drops are resolved by the TCP layer by a combination of sequence numbers and ACKs. To deal with data corruption, the TCP/IP protocol uses a form of error correction called CRC Checksum [33]. From a theoretical point of view, the checksum is a terrible code since it does not have good error correction properties (for that matter so is English). However, on the Internet, the current dominant mode of operation is to detect errors and if errors have occurred, then ask for retransmission. This is the reason why the use of checksum has been hugely successful in the Internet. However, there are other communication applications where re-transmission is not an option. Codes are used when transmitting data over the telephone line or via cell phones. They are also used in deep space communication and in satellite broadcast (for example, TV signals are transmitted via satellite). Indeed, asking the Mars Rover to re-send an image just because it got corrupted during transmission is not an option–this is the reason that for such applications, the codes used have always been very sophisticated.

Codes also have applications in areas not directly related to communication. In particular, in the applications above, we want to communicate over space. Codes can also be used to communicate over time. For example, codes are used heavily in data storage. CDs and DVDs work fine even in presence of scratches precisely because they use codes. Codes are used in Redundant Array of Inexpensive Disks (RAID) [9] and error correcting memory [8]. Sometimes, in the Blue Screen of Death displayed by Microsoft Windows family of operating systems, you might see a line saying something along the lines of "parity check failed"–this happens when the code used in the error-correcting memory cannot recover from error(s). Also, certain consumers of memory, e.g. banks, do not want to suffer from even one bit flipping (this e.g. could mean someone's bank balance either got halved or doubled–neither of which are welcome[1]). Codes are also deployed in other applications such as paper bar codes; for example, the bar code used by UPS called MaxiCode [7]. Unlike the Internet example, in all of these applications, there is no scope for "re-transmission."

In this book, we will mainly think of codes in the communication scenario. In this

[1]This is a bit tongue-in-cheek: in real life banks have more mechanisms to prevent one-bit flip from wreaking havoc.

framework, there is a sender who wants to send (say) k message symbols over a noisy channel. The sender first *encodes* the k message symbols into n symbols (called a *codeword*) and then sends it over the *channel*. The receiver gets a *received word* consisting of n symbols. The receiver then tries to *decode* and recover the original k message symbols. Thus, encoding is the process of adding redundancy and decoding is the process of removing errors.

Unless mentioned otherwise, in this book we will make the following assumption:

Note

The sender and the receiver only communicate via the channel.[a] In other words, other than some setup information about the code, the sender and the receiver do not have any other information exchange (other than of course what was transmitted over the channel). In particular, no message is more likely to be transmitted over another.

[a] The scenario where the sender and receiver have a "side-channel" is an interesting topic that has been studied but is outside the scope of this book.

The fundamental question that will occupy our attention for almost the entire book is the tradeoff between the amount of redundancy used and the number of errors that can be corrected by a code. In particular, we would like to understand:

Comment out 4 lines below for tagging check

Question 1.1.1 (Main Question)**.** *How much redundancy do we need to correct a given amount of errors? (We would like to correct as many errors as possible with as little redundancy as possible.)*

Note that maximizing error correction and minimizing redundancy are contradictory goals: a code with higher redundancy should be able to tolerate a greater number of errors. By the end of this chapter, we will see a formalization of this question.

Once we determine the optimal tradeoff, we will be interested in achieving this optimal tradeoff with codes that come equipped with *efficient* encoding and decoding. (A DVD player that tells its consumer that it will recover from a scratch on a DVD by tomorrow is not exactly going to be a best-seller.) In this book, we will primarily define efficient algorithms to be ones that run in polynomial time.[2]

1.2 Some Definitions and Codes

To formalize Question 1.1.1, we begin with the definition of a code.

Definition 1.2.1 (Code)**.** *A code of* block length n *over an* alphabet Σ *is a subset of* Σ^n*. Typically, we will use* q *to denote the alphabet size* $|\Sigma|$*.*[3]

[2] Readers unfamiliar with runtime analysis are referred to Appendix **??**. Coming back to the claim on efficiency– we are not claiming that this is the correct notion of efficiency in practice. However, we believe that it is a good definition as the "first cut"– quadratic or cubic time algorithms are definitely more desirable than exponential time algorithms: see Section **??** for more on this.

[3] Note that q need not be a constant and can depend on n: we'll see codes in this book where this is true.

Remark 1.2.2. *We note that the ambient space Σ^n can be viewed as a set of sequences, vectors or functions. In other words, we can think of a vector $(v_1, \ldots, v_n) \in \Sigma^n$ as just the sequence $v_1, \ldots, v_n$ (in order) or a vector tuple $(v_1, \ldots, v_n)$ or as the function $f : [n] \to \Sigma$ such that $f(i) = v_i$. Sequences assume least structure on Σ and hence are most generic. Vectors work well when Σ has some structure (and in particular is what is known as a* field*, which we will see next chapter). Functional representation will be convenient when the set of coordinates has structure (e.g., [n] may come from a finite field of size n). For now, however, the exact representation does not matter and the reader can work with representation as sequences.*

We will also frequently use the following alternate way of looking at a code. Given a code $C \subseteq \Sigma^n$, with $|C| = M$, we will think of C as a mapping of the following form:

$$C : [M] \to \Sigma^n. \tag{1.1}$$

In the above equation (1.1), we have used the notation $[M]$ for any integer $M \geq 1$ to denote the set $\{1, 2, \ldots, M\}$.

We will also need the notion of *dimension* of a code.

Definition 1.2.3 (Dimension of a code)**.** *Given a code $C \subseteq \Sigma^n$, its* dimension *is given by*

$$k \stackrel{\text{def}}{=} \log_q |C|.$$

Let us begin by looking at two specific codes. Both codes are defined over $\Sigma = \{0, 1\}$ (also known as *binary codes*). In both cases $|C| = 2^4$ and we will think of each of the 16 messages as a 4 bit vector.

We first look at the so-called *parity code*, which we will denote by $C_{\oplus}$. Given a message $(x_1, x_2, x_3, x_4) \in \{0, 1\}^4$, its corresponding codeword is given by

$$C_{\oplus}(x_1,\ x_2,\ x_3,\ x_4) = (x_1,\ x_2,\ x_3,\ x_4,\ x_1 \oplus x_2 \oplus x_3 \oplus x_4), \tag{1.2}$$

where the $\oplus$ denotes the XOR (also known as the EXOR or Exclusive-OR) operator. In other words, the parity code appends the parity of the message bits (or takes the remainder of the sum of the message bits when divided by 2) at the end of the message. For example, the message $(1, 0, 0, 1)$ will have a 0 appended at the end while $(1, 0, 0, 0)$ will have a 1 appended at the end. Note that such a code uses the minimum amount of non-zero redundancy.

The second code we will look at is the so-called *repetition code*. This is a very natural code (and perhaps the first code one might think of). The idea is to repeat every message bit a fixed number of times. For example, we repeat each of the 4 message bits 3 times and we use $C_{3,rep}$ to denote this code. Given a message $(x_1, x_2, x_3, x_4) \in \{0, 1\}^4$, its corresponding codeword is given by

$$C_{3,rep}(x_1,\ x_2,\ x_3,\ x_4) = (x_1,\ x_1,\ x_1,\ x_2,\ x_2,\ x_2,\ x_3,\ x_3,\ x_3,\ x_4,\ x_4,\ x_4). \tag{1.3}$$

Let us now try to look at the tradeoff between the amount of redundancy and the number of errors each of these codes can correct. Even before we begin to answer the question, we need to define how we are going to measure the amount of redundancy. One natural way to define redundancy for a code with dimension k and block length n is by their difference $n - k$. By this definition, the parity code uses the least amount of redundancy. However, one "pitfall" of such a definition is that it does not distinguish between a code with $k = 100$ and $n = 102$ and another code with dimension and block length 2 and 4, respectively. The first code uses 0.02 bits of redundancy per message bit while the second code uses 1 bit of redundancy per message bit. Thus, in the relative sense, the latter code is using more redundancy. This motivates the following notion of measuring redundancy.

Definition 1.2.4 (Rate of a code)**.** *The* rate *of a code with dimension k and block length n is given by*

$$R \stackrel{\text{def}}{=} \frac{k}{n}.$$

Note that the higher the rate, the lesser the amount of redundancy in the code. Thus, when constructing or analyzing codes, we will be interested in lower bounding the rate of a code. (Occasionally we will also be sloppy and say that a code "has rate R" when we really mean it "has rate at least R.") Also note that as $k \leq n$,[4]

$$R \leq 1.$$

In other words, the rate of a code is the average amount of real information in each of the n symbols transmitted over the channel. So, in some sense, rate captures the complement of redundancy. However, for historical reasons, we will deal with the rate R (instead of the more natural $1 - R$) as our notion of redundancy. Given the above definition, $C_{\oplus}$ and $C_{3,rep}$ have rates of $\frac{4}{5}$ and $\frac{1}{3}$. As expected, the parity code has a higher rate than the repetition code.

We have formalized the notion of redundancy as the rate of a code as well as other parameters of a code. However, to formalize Question 1.1.1, we still need to formally define what it means to correct errors. We do so next.

1.3 Error Correction

Before we formally define error correction, we will first formally define the notion of *encoding*.

Definition 1.3.1 (Encoding function)**.** *Let $C \subseteq \Sigma^n$. An equivalent description of the code C is an injective mapping $E : [|C|] \to \Sigma^n$ called the encoding function.*

Next we move to error correction. Informally, we can correct a received word if we can recover the transmitted codeword (or equivalently the corresponding message). This "reverse" process is called *decoding*.

[4]Further, in this book, we will always consider the case $k > 0$ and $n < \infty$ and hence, we can also assume that $R > 0$.

Definition 1.3.2 (Decoding function)**.** *Let $C \subseteq \Sigma^n$ be a code. A mapping $D : \Sigma^n \to [|C|]$ is called a decoding function for C.*

The definition of a decoding function by itself does not give anything interesting. What we really need from a decoding function is for the function to recover the transmitted message. To understand this notion, we first need to understand the nature of errors that we aim to tackle. In particular, if a transmitter transmits $\mathbf{u} \in \Sigma^n$ and the receiver receives $\mathbf{v} \in \Sigma^n$, how do we quantify the amount of "error" that has happened during this transmission? While multiple notions are possible, the most central one, and the one we will focus on for most of this book, is based on "Hamming distance," a notion of distance that captures how close are two given sequences $\mathbf{u}$ and $\mathbf{v}$.

Definition 1.3.3 (Hamming distance)**.** *Given two vectors $\mathbf{u}, \mathbf{v} \in \Sigma^n$ the* Hamming distance *between $\mathbf{u}$ and $\mathbf{v}$, denoted by $\Delta(\mathbf{u}, \mathbf{v})$, is the number of positions in which $\mathbf{u}$ and $\mathbf{v}$ differ. We also define the* relative Hamming distance*, denoted $\delta(\mathbf{u}, \mathbf{v})$, to be the quantity $\delta(\mathbf{u}, \mathbf{v}) = \frac{1}{n}\Delta(\mathbf{u}, \mathbf{v})$.*

Note that the relative Hamming distance normalizes the distance so that $\delta(\mathbf{u}, \mathbf{v})$ always lies in the interval $[0, 1]$ (for every n, Σ and strings $\mathbf{u}, \mathbf{v} \in \Sigma^n$). This normalization will be useful when we study the asymptotic behavior of encoding and decoding functions, i.e., as $n \to \infty$. For now, though we will focus mostly on the (non-relative) Hamming distance.

The Hamming distance is a distance in a very formal mathematical sense: see Exercise 1.5. Note that the definition of Hamming distance depends only on the *number* of differences and not the nature of the difference. For example, consider the vectors $\mathbf{u} = 00000$ and $\mathbf{v} = 10001$. One can see that their Hamming distance is $\Delta(\mathbf{u}, \mathbf{v}) = 2$. Now consider the vector $\mathbf{w} = 01010$. Note that even though $\mathbf{v} \neq \mathbf{w}$, we again have a Hamming distance $\Delta(\mathbf{u}, \mathbf{w}) = 2$.

To return to the quantification of errors, from now on we will say that if $\mathbf{u}$ is transmitted and $\mathbf{v}$ is received then $\Delta(\mathbf{u}, \mathbf{v})$ errors occurred during transmission. This allows us to quantify the performance of an encoding/decoding function, or equivalently the underlying code as we do next.

Definition 1.3.4 (t-Error Channel)**.** *An n-symbol t-Error Channel over the alphabet Σ is a function $\mathrm{Ch} : \Sigma^n \to \Sigma^n$ that satisfies $\Delta(\mathbf{v}, \mathrm{Ch}(\mathbf{v})) \leq t$ for every $\mathbf{v} \in \Sigma^n$.*

Definition 1.3.5 (Error Correcting Code)**.** *Let $C \subseteq \Sigma^n$ be a code and let $t \geq 1$ be an integer. C is said to be a t-*error-correcting code *if there exists a decoding function D such that for every message $\mathbf{m} \in [|C|]$ and every t-error channel Ch we have $D(\mathrm{Ch}(C(\mathbf{m}))) = \mathbf{m}$.*

Thus, a t-error-correcting code is one where there is a decoding function that corrects any pattern of t errors. For example, consider the case when the codeword $(0, 0, 0, 0)$ is transmitted. Then a 1-error-correcting code (over the alphabet $\{0, 1\}$) should be able to decode from any of the following received words:

$$(0, 0, 0, 0),\ (1, 0, 0, 0),\ (0, 1, 0, 0),\ (0, 0, 1, 0),\ (0, 0, 0, 1).$$

Figure 1.2 illustrates how the definitions we have examined so far interact.

We will also very briefly look at a weaker form of error recovery called *error detection*.

Figure 1.2: Coding process

Definition 1.3.6 (Error detection code)**.** *Let $C \subseteq \Sigma^n$ be a code and let $t \geq 1$ be an integer. C is said to be a* t-error-detecting code *if there exists a detecting procedure D such that for every message $\mathbf{m}$ and every received vector $\mathbf{v} \in \Sigma^n$ satisfying $\Delta(C(\mathbf{m}), \mathbf{v}) \leq t$, it holds that D outputs a 1 if $\mathbf{v} = C(\mathbf{m})$ and 0 otherwise. In other words*

$$D(\mathbf{v}) = \begin{cases} 1 & \text{if } \mathbf{v} = C(\mathbf{m}) \\ 0 & \text{otherwise} \end{cases}.$$

Thus, a t-error-detecting code is one where if the transmission has at least one error and at most t errors, then the decoding function detects the error (by outputting 0). Note that a t-error correcting code is also a t-error detecting code (but not necessarily the other way round): see Exercise 1.1. Although error detection might seem like a weak error recovery model, it is useful in settings where the receiver can ask the sender to re-send the message. For example, error detection is used quite heavily in the Internet.

Finally, we also consider a more benign model of errors referred to as "erasures," where a symbol is merely (and explicitly) omitted from the transmission (as opposed to being replaced by some other symbol). More specifically, if a symbols is erased, then it is *replaced* by a special symbol "?" that is not a member of the alphabet Σ. For example, if $(0, 0, 0, 0)$ was transmitted and the second symbols was erased by the channel, then the vector $(0, ?, 0, 0)$ will be received.

Definition 1.3.7 (t-Erasure Channel)**.** *An n-symbol t-Erasure Channel over the alphabet Σ is a function $\mathrm{Ch} : \Sigma^n \to (\Sigma \cup \{?\})^n$ that satisfies $\Delta(\mathbf{v}, \mathrm{Ch}(\mathbf{v})) \leq t$ for every $\mathbf{v} \in \Sigma^n$ (where both arguments to $\Delta(\cdot, \cdot)$ are viewed as elements of $(\Sigma \cup \{?\})^n$) and for every $i \in [n]$ such that $\mathbf{v}_i \neq \mathrm{Ch}(\mathbf{v})_i$ we have $\mathrm{Ch}(\mathbf{v})_i = ?$.*

A coordinate i such that $\mathrm{Ch}(\mathbf{v})_i = ?$ is called an erasure. We may now define erasure correcting codes analogously to error-correcting codes.

Definition 1.3.8 (Erasure Correcting Code)**.** *Let $C \subseteq \Sigma^n$ be a code and let $t \geq 1$ be an integer. C is said to be a* t-erasure-correcting code *if there exists a decoding function D such that for every message $\mathbf{m} \in [|C|]$ and for every t-erasure channel* Ch *we have $D\left(\mathrm{Ch}(C(\mathbf{m}))\right) = \mathbf{m}$.*

With the above definitions in place, we are now ready to look at the error correcting capabilities of the codes we looked at in the previous section.

1.3.1 Error-Correcting Capabilities of Parity and Repetition Codes

In Section 1.2, we looked at examples of parity code and repetition code with the following properties:

$$C_{\oplus} : q = 2, k = 4, n = 5, R = 4/5.$$
$$C_{3,rep} : q = 2, k = 4, n = 12, R = 1/3.$$

We will start with the repetition code. To study its error-correcting capabilities, we will consider the following natural decoding function. Given a received word $\mathbf{y} \in \{0,1\}^{12}$ (where recall the transmitted codeword is of the form $(x_1, x_1, x_1, x_2, x_2, x_2, x_3, x_3, x_3, x_4, x_4, x_4)$ for some $(x_1, x_2, x_3, x_4) \in \{0,1\}^4$), divide it up into four consecutive blocks (y_1, y_2, y_3, y_4) where every block consists of three bits. Then, for every block y_i $(1 \leq i \leq 4)$, output the majority bit as the message bit. We claim this decoding function can correct any error pattern with at most 1 error (see Exercise 1.2.) For example, if a block of 010 is received, since there are two 0's we know the original message bit was 0. In other words, we have argued the following error correcting capability of $C_{3,rep}$:

Proposition 1.3.9. *$C_{3,rep}$ is a 1-error correcting code.*

However, it is not too hard to see that $C_{3,rep}$ cannot correct two errors. For example, if both of the errors happen in the same block and a block in the received word is 010, then the original block in the codeword could have been either 111 or 000. Therefore in this case, no decoder can successfully recover the transmitted message.[5]

Thus, we have pin-pointed the error-correcting capabilities of the $C_{3,rep}$ code: it can correct one error, but not two or more. However, note that the argument assumed that the error positions can be located arbitrarily. In other words, we are assuming that the channel noise behaves arbitrarily (subject to a bound on the total number of errors). However, we can model the noise differently. We now briefly digress to look at this issue in slightly more detail.

Digression: Channel Noise. As was mentioned above, until now we have been assuming the following noise model, which was first studied by Hamming:

> Any error pattern can occur during transmission as long as the total number of errors is bounded. Note that this means that the location as well as the nature[6] of the errors is arbitrary.

We will frequently refer to Hamming's model as the *Adversarial Noise Model*. It is important to note that the atomic unit of error is a symbol from the alphabet. For example, if the error pattern[7] is $(1, 0, 1, 0, 0, 0)$ and we consider the alphabet to be $\{0, 1\}$, then the pattern has two

[5] Recall we are assuming that the decoder has no side information about the transmitted message.

[6] For binary codes, there is only one kind of error: a bit flip. However, for codes over a larger alphabet, say $\{0, 1, 2\}$, 0 being converted to a 1 and 0 being converted into a 2 are both errors, but are different kinds of errors.

[7] If $\mathbf{v}$ is transmitted and $\mathrm{Ch}(\mathbf{v})$ is received then the 'difference' between $\mathrm{Ch}(\mathbf{v})$ and $\mathbf{v}$ is the error pattern. For binary alphabet the difference is the XOR operator.

errors (since the first and the third locations in the vector have a non-zero value, i.e. value of 1). However, if our alphabet is $\{0,1\}^3$ (i.e. we think of the vector above as $((1,0,1),(0,0,0))$, with $(0,0,0)$ corresponding to the zero element in $\{0,1\}^3$), then the pattern has only one error. Thus, by increasing the alphabet size we can also change the adversarial noise model. As the book progresses, we will see how error correction over a larger alphabet is easier than error correction over a smaller alphabet.

However, the above is not the only way to model noise. For example, we could also have following error model:

> No more than 1 error can happen in any contiguous three-bit block.

First note that, for the error model above, no more than four errors can occur when a codeword in $C_{3,rep}$ is transmitted. (Recall that in $C_{3,rep}$, each of the four bits is repeated three times.) Second, note that the decoding function that takes the majority vote of each block can successfully recover the transmitted codeword for *any* error pattern, while in the worst-case noise model it could only correct at most one error. This channel model is admittedly contrived, but it illustrates the point that the error-correcting capabilities of a code (and a decoding function) are crucially dependent on the noise model.

A popular alternate noise model is to model the channel as a stochastic process. As a concrete example, let us briefly mention the *binary symmetric channel with crossover probability* $0 \leq p \leq 1$, denoted by BSC_p, which was first studied by Shannon. In this model, when a (binary) codeword is transferred through the channel, every bit flips independently with probability p.

Note that the two noise models proposed by Hamming and Shannon are in some sense two extremes: Hamming's model assumes *no* knowledge about the channel (except that a bound on the total number of errors is known[8] while Shannon's noise model assumes *complete* knowledge about how noise is produced. In this book, we will consider only these two extreme noise models. In real life, the situation often is somewhere in between.

For real life applications, modeling the noise model correctly is an extremely important task, as we can tailor our codes to the noise model at hand. However, in this book we will not study this aspect of designing codes at all, and will instead mostly consider the worst-case noise model. Informally, if one can communicate over the worst-case noise model, then one could use the same code to communicate over nearly every other noise model with the same amount of noise.

We now return to $C_{\oplus}$ and examine its error-correcting capabilities in the worst-case noise model. We claim that $C_{\oplus}$ cannot correct even one error. Suppose $\mathbf{y} = 10000$ is the received word. Then we know that an error has occurred, but we do not know which bit was flipped. This is because the two codewords $\mathbf{u} = 00000$ and $\mathbf{v} = 10001$ differ from the received word $\mathbf{y}$ in exactly one bit. As we are assuming that the receiver has no side information about the transmitted codeword, no decoder can know what the transmitted codeword was.

[8] A bound on the total number of errors is necessary; otherwise, error correction would be impossible: see Exercise 1.3.

Thus, from an error-correction point of view, $C_\oplus$ is a terrible code (as it cannot correct even 1 error). However, we will now see that $C_\oplus$ can *detect* one error. Consider Algorithm 1.3.1.

Algorithm 1.3.1 Error Detector for Parity Code

INPUT: Received word $\mathbf{y} = (y_1, y_2, y_3, y_4, y_5)$
OUTPUT: 1 if $\mathbf{y} \in C_\oplus$ and 0 otherwise

1: $b \leftarrow y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5$
2: RETURN $1 \oplus b$ ▷ If there is no error, then $b = 0$ and hence we need to "flip" the bit for the answer

Note that when no error has occurred during transmission, $y_i = x_i$ for $1 \leq i \leq 4$ and $y_5 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, in which case $b = 0$ and we output $1 \oplus 0 = 1$ as required. If there is a single error then either $y_i = x_i \oplus 1$ (for exactly one $1 \leq i \leq 4$) or $y_5 = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus 1$. It can be check that in this case, $b = 1$. In fact, one can extend this argument to obtain the following result (see Exercise 1.4).

Proposition 1.3.10. *The parity code $C_\oplus$ can* detect *an odd number of errors.*

Let us now revisit the example that showed that one cannot correct one error using $C_\oplus$. Recall, we considered two codewords in $C_\oplus$, $\mathbf{u} = 00000$ and $\mathbf{v} = 10001$ (which are codewords corresponding to messages 0000 and 1000, respectively). Now consider the scenarios in which $\mathbf{u}$ and $\mathbf{v}$ are each transmitted and a single error occurs resulting in the received word $\mathbf{r} = 10000$. Thus, given the received word $\mathbf{r}$ and the fact that at most one error can occur, the decoder has no way of knowing whether the original transmitted codeword was $\mathbf{u}$ or $\mathbf{v}$. Looking back at the example, it is clear that the decoder is "confused" because the two codewords $\mathbf{u}$ and $\mathbf{v}$ do not differ in many positions. This notion is formalized in the next section.

1.4 Distance of a Code

We now turn to a new parameter associated with a code that we call the minimum distance of a code. As we will see later, minimum distance is connected to the other parameters, including the error-correction and error-detection capacity of the code. However, due to the cleanliness of the definition, it will often be the first of the parameters we will explore when studying a new error-correcting code.

Definition 1.4.1 (Minimum distance)**.** *Let $C \subseteq \Sigma^n$. The* minimum distance *(or just* distance*) of C, denoted $\Delta(C)$, is defined to be*

$$\Delta(C) = \min_{\mathbf{c}_1 \neq \mathbf{c}_2 \in C} \Delta(\mathbf{c}_1, \mathbf{c}_2).$$

We also define the relative minimum distance *of C to be $\delta(C)$, is defined to be*

$$\delta(C) = \min_{\mathbf{c}_1 \neq \mathbf{c}_2 \in C} \delta(\mathbf{c}_1, \mathbf{c}_2).$$

In other words, $\Delta(C)$ is the minimum distance between two distinct codewords in C. We note that the repetition code $C_{3,rep}$ has distance 3 (recall (1.3)). Indeed, any two distinct messages will differ in at least one of the message bits. After encoding, the difference in one message bit will translate into a difference of three bits in the corresponding codewords. For example

$$C_{3,rep}(0,0,0,0) = (0,0,0,0,0,0,0,0,0,0,0,0) \text{ and } C_{3,rep}(1,0,0,0) = (1,1,1,0,0,0,0,0,0,0,0,0).$$

We now claim that the distance of $C_{\oplus}$ is 2. This is a consequence of the following observations. If two messages $\mathbf{m}_1$ and $\mathbf{m}_2$ differ in at least two places then $\Delta(C_{\oplus}(\mathbf{m}_1), C_{\oplus}(\mathbf{m}_2)) \geq 2$ (even if we just ignored the parity bits). If two messages differ in exactly one place then the parity bits in the corresponding codewords are different, which implies a Hamming distance of 2 between the codewords. For example,

$$C_{\oplus}(1,0,0,0) = (1,0,0,0,1) \text{ and } C_{\oplus}(1,0,0,1) = (1,0,0,1,0).$$

Thus, $C_{\oplus}$ has a smaller distance than $C_{3,rep}$ and can correct less number of errors than $C_{3,rep}$. This suggests that a larger distance implies greater error-correcting capabilities. The next result formalizes this intuition. As we will see, minimum distance exactly captures both the ability to recover from errors as also the notion of erasures (Definition 1.3.8).

Proposition 1.4.2. *Given a code C, the following are equivalent:*

1. *C has minimum distance $d \geq 2$,*
2. *If d is odd, C can correct $(d-1)/2$ errors.*
3. *C can detect $d-1$ errors.*
4. *C can correct $d-1$ erasures.*

Remark 1.4.3. *Property (2) above for even d is slightly different. In this case, one can correct up to $\frac{d}{2}-1$ errors but cannot correct $\frac{d}{2}$ errors. (See Exercise 1.6.)*

Before we prove Proposition 1.4.2, let us apply it to the codes $C_{\oplus}$ and $C_{3,rep}$ which have distances of 2 and 3, respectively. Proposition 1.4.2 implies the following facts that we have already proved:

- $C_{3,rep}$ can correct 1 error (Proposition 1.3.9).
- $C_{\oplus}$ can detect 1 error but cannot correct 1 error (Proposition 1.3.10).

The proof of Proposition 1.4.2 will need the following decoding function. *Maximum likelihood decoding* (MLD) is a well-studied decoding method for error correcting codes. The MLD function outputs the codeword $\mathbf{c} \in C$, which is as close as possible to the received word in Hamming distance (with ties broken arbitrarily).[9] More formally, the MLD function denoted by $D_{MLD} : \Sigma^n \to C$ is defined as follows. For every $\mathbf{y} \in \Sigma^n$,

$$D_{MLD}(\mathbf{y}) = \arg\min_{\mathbf{c} \in C} \Delta(\mathbf{c}, \mathbf{y}).$$

Algorithm 1.4.1 is a naive implementation of the MLD.

Algorithm 1.4.1 Naive Maximum Likelihood Decoder

INPUT: Received word $\mathbf{y} \in \Sigma^n$

OUTPUT: $D_{MLD}(\mathbf{y})$

Pick an arbitrary $\mathbf{c} \in C$ and assign $\mathbf{z} \leftarrow \mathbf{c}$
FOR every $\mathbf{c}' \in C$ such that $\mathbf{c} \neq \mathbf{c}'$ DO
    IF $\Delta(\mathbf{c}', \mathbf{y}) < \Delta(\mathbf{z}, \mathbf{y})$ THEN
        $\mathbf{z} \leftarrow \mathbf{c}'$
RETURN $\mathbf{z}$

Proof of Proposition 1.4.2 We will complete the proof in two steps. First, we will show that if property 1 is satisfied then so are properties 2, 3 and 4 (we prove this via three implications (1) implies (2), (1) implies (3) and (1) implies (4)). Then we show that if property 1 is not satisfied then none of the properties 2, 3 or 4 hold (again via the corresponding three implications).

Item 1. **implies** 2. Assume C has distance d. We first prove 2 (for this case assume that $d = 2t + 1$). We now need to show that there exists a decoding function such that for all error patterns with at most t errors it always outputs the transmitted message. We claim that the MLD function has this property. Assume this is not so and let $\mathbf{c}_1$ be the transmitted codeword and let $\mathbf{y}$ be the received word. Note that

$$\Delta(\mathbf{y}, \mathbf{c}_1) \leq t. \tag{1.4}$$

As we have assumed that MLD does not work, $D_{MLD}(\mathbf{y}) = \mathbf{c}_2 \neq \mathbf{c}_1$. Note that by the definition of MLD,

$$\Delta(\mathbf{y}, \mathbf{c}_2) \leq \Delta(\mathbf{y}, \mathbf{c}_1). \tag{1.5}$$

[9] Technically, as per Definition 1.3.2, a decoder should output a message while MLD outputs a codeword. However, since we only consider code of distance at least one in this book, there is a bijection between codewords and message so this syntatic difference does not matter.

Consider the following set of inequalities:

$$\begin{aligned}\Delta(\mathbf{c}_1, \mathbf{c}_2) &\leq \Delta(\mathbf{c}_2, \mathbf{y}) + \Delta(\mathbf{c}_1, \mathbf{y}) && (1.6)\\ &\leq 2\Delta(\mathbf{c}_1, \mathbf{y}) && (1.7)\\ &\leq 2t && (1.8)\\ &= d - 1, && (1.9)\end{aligned}$$

where (1.6) follows from the triangle inequality (see Exercise 1.5), (1.7) follows from (1.5) and (1.8) follows from (1.4). (1.9) implies that the distance of C is at most $d-1$, which is a contradiction.

Item 1. **implies** 3. We now show that property 3 holds. That is, we need to describe an algorithm that can successfully detect whether errors have occurred during transmission (as long as the total number of errors is bounded by $d-1$). Consider the following error detection algorithm: check if the received word $\mathbf{y} = \mathbf{c}$ for some $\mathbf{c} \in C$ (this can be done via an exhaustive check). If no errors occurred during transmission, $\mathbf{y} = \mathbf{c}_1$, where $\mathbf{c}_1$ was the transmitted codeword and the algorithm above will accept (as it should). On the other hand if $1 \leq \Delta(\mathbf{y}, \mathbf{c}_1) \leq d-1$, then by the fact that the distance of C is d, $\mathbf{y} \notin C$ and hence the algorithm rejects, as required.

Item 1. **implies** 4. Finally, we prove that property 4 holds. Let $\mathbf{y} \in (\Sigma \cup \{?\})^n$ be the received word. First we claim that there is a unique $\mathbf{c} = (c_1, \ldots, c_n) \in C$ that agrees with $\mathbf{y}$ (i.e. $y_i = c_i$ for every i such that $y_i \neq ?$). Indeed, for the sake of contradiction, assume that this is not true, i.e. there exists two distinct codewords $\mathbf{c}_1, \mathbf{c}_2 \in C$ such that both $\mathbf{c}_1$ and $\mathbf{c}_2$ agree with $\mathbf{y}$ in the unerased positions. Note that this implies that $\mathbf{c}_1$ and $\mathbf{c}_2$ agree in the positions i such that $y_i \neq ?$. Thus, $\Delta(\mathbf{c}_1, \mathbf{c}_2) \leq |\{i | y_i = ?\}| \leq d-1$, which contradicts the assumption that C has distance d.

Given the uniqueness of the codeword $\mathbf{c} \in C$ that agrees with $\mathbf{y}$ in the unerased position, an algorithm to find $\mathbf{c}$ is as follows: go through all the codewords in C and output the desired codeword.

Item $\neg$1. **implies** $\neg$2. For the other direction of the proof, assume that property 1 does not hold, that is, C has distance $d-1$. We now show that property 2 cannot hold: i.e., for every decoding function there exists a transmitted codeword $\mathbf{c}_1$ and a received word $\mathbf{y}$ (where $\Delta(\mathbf{y}, \mathbf{c}_1) \leq (d-1)/2$) such that the decoding function cannot output $\mathbf{c}_1$. Let $\mathbf{c}_1 \neq \mathbf{c}_2 \in C$ be codewords such that $\Delta(\mathbf{c}_1, \mathbf{c}_2) = d-1$ (such a pair exists as C has distance $d-1$). Now consider a vector $\mathbf{y}$ such that $\Delta(\mathbf{y}, \mathbf{c}_1) = \Delta(\mathbf{y}, \mathbf{c}_2) = (d-1)/2$. Such a $\mathbf{y}$ exists as d is odd and by the choice of $\mathbf{c}_1$ and $\mathbf{c}_2$. Figure 1.3 gives an illustration of such a $\mathbf{y}$ (matching color implies that the vectors agree on those positions).

Now, since $\mathbf{y}$ could have been generated if *either* of $\mathbf{c}_1$ or $\mathbf{c}_2$ were the transmitted codeword, no decoding function can work in this case.[10]

[10]Note that this argument is just a generalization of the argument that $C_{\oplus}$ cannot correct 1 error.

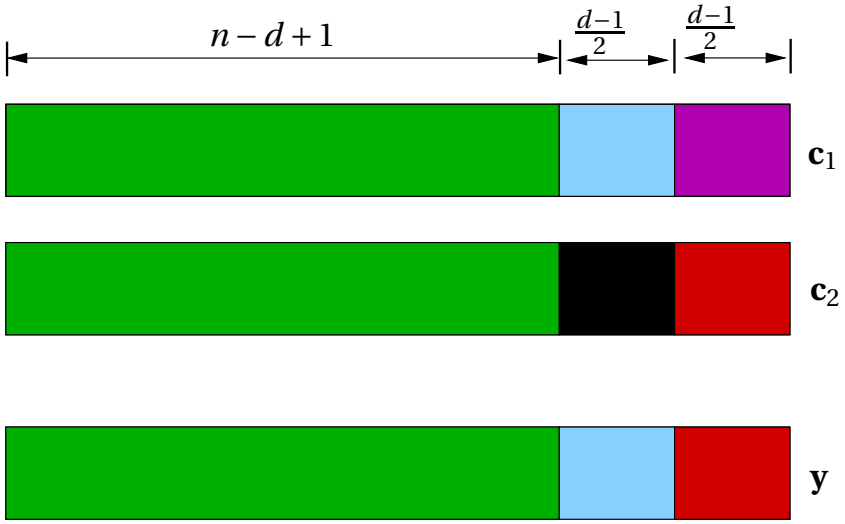

Figure 1.3: Bad example for unique decoding.

Item ¬1. **implies** ¬3. For the remainder of the proof, assume that the transmitted word is $\mathbf{c}_1$ and there exists another codeword $\mathbf{c}_2$ such that $\Delta(\mathbf{c}_2, \mathbf{c}_1) = d - 1$. To see why property 3 is not true, let $\mathbf{y} = \mathbf{c}_2$. In this case, either the error detecting algorithm detects no error, or it declares an error when $\mathbf{c}_2$ is the transmitted codeword and no error takes place during transmission.

Item ¬1. **implies** ¬4. We finally argue that property 4 does not hold. Let $\mathbf{y}$ be the received word in which the positions that are erased are exactly those where $\mathbf{c}_1$ and $\mathbf{c}_2$ differ. Thus, given $\mathbf{y}$ both $\mathbf{c}_1$ and $\mathbf{c}_2$ could have been the transmitted codeword, and no algorithm for correcting (at most $d - 1$) erasures can work in this case. ■

Proposition 1.4.2 implies that Question 1.1.1 can be reframed as

Question 1.4.4 (Main question: reframed)**.** *What is the largest rate R that a code with distance d can have?*

We have seen that the repetition code $C_{3,rep}$ has distance 3 and rate 1/3. A natural follow-up question (which is a special case of Question 1.4.4) is to ask

Question 1.4.5 (Special case of Question 1.4.4)**.** *Can we have a code with distance* 3 *and rate $R > \frac{1}{3}$?*

1.5 Hamming Code

With the above question in mind, let us consider the so-called *Hamming code*, which we will denote by C_H. Given a message $(x_1, x_2, x_3, x_4) \in \{0, 1\}^4$, its corresponding codeword is given

by

$$C_H(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4, x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_3 \oplus x_4, x_1 \oplus x_2 \oplus x_4).$$

It can be verified that this code has the following parameters:

$$C_H : q = 2, k = 4, n = 7, R = 4/7.$$

We will show shortly that C_H has a distance of 3. We would like to point out that we could have picked the three parities differently. The reason we mention the three particular parities above is due to historical reasons. We leave it as an exercise to define an alternate set of parities such that the resulting code still has a distance of 3: see Exercise 1.9.

Before we move on to determining the distance of C_H, we will need another definition.

Definition 1.5.1 (Hamming Weight)**.** *Let $q \geq 2$. Given any vector $\mathbf{v} \in \{0, 1, 2, \ldots, q-1\}^n$, its Hamming weight, denoted by $wt(\mathbf{v})$ is the number of non-zero symbols in $\mathbf{v}$.*

For example, if $\mathbf{v} = 01203400$, then $wt(\mathbf{v}) = 4$.

We now look at the distance of C_H.

Proposition 1.5.2. *C_H has a distance of 3.*

Proof. We will prove the claimed distance by using two properties of C_H:

$$\min_{\mathbf{c} \in C_H, \mathbf{c} \neq \mathbf{0}} wt(\mathbf{c}) = 3, \tag{1.10}$$

and

$$\min_{\mathbf{c} \in C_H, \mathbf{c} \neq \mathbf{0}} wt(\mathbf{c}) = \min_{\mathbf{c}_1 \neq \mathbf{c}_2 \in C_H} \Delta(\mathbf{c}_1, \mathbf{c}_2) \tag{1.11}$$

The proof of (1.10) follows from a case analysis on the Hamming weight of the message bits. Let us use $\mathbf{x} = (x_1, x_2, x_3, x_4)$ to denote the message vector.

- Case 0: If $wt(\mathbf{x}) = 0$, then $C_H(\mathbf{x}) = \mathbf{0}$, which means we do not have to consider this codeword.
- Case 1: If $wt(\mathbf{x}) = 1$ then at least two parity check bits in $(x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_3 \oplus x_4, x_1 \oplus x_2 \oplus x_4)$ are 1 (see Exercise 1.10). So in this case, $wt(C_H(\mathbf{x})) \geq 3$.
- Case 2: If $wt(\mathbf{x}) = 2$ then at least one parity check bit in $(x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_3 \oplus x_4, x_1 \oplus x_2 \oplus x_4)$ is 1 (see Exercise 1.11). So in this case, $wt(C_H(\mathbf{x})) \geq 3$.
- Case 3: If $wt(\mathbf{x}) \geq 3$ then those message bits themselves imply that $wt(C_H(\mathbf{x})) \geq 3$.

Thus, we can conclude that $\min_{\mathbf{c} \in C_H, \mathbf{c} \neq \mathbf{0}} wt(\mathbf{c}) \geq 3$. Further, note that $wt(C_H(1, 0, 0, 0)) = 3$, which implies that $\min_{\mathbf{c} \in C_H, \mathbf{c} \neq \mathbf{0}} wt(\mathbf{c}) \leq 3$. This along with the lower bound that we just obtained proves (1.10).

We now turn to the proof of (1.11). For the rest of the proof, let $\mathbf{x} = (x_1, x_2, x_3, x_4)$ and $\mathbf{y} = (y_1, y_2, y_3, y_4)$ denote the two distinct messages. Using associativity and commutativity of the $\oplus$ operator, we obtain that

$$C_H(\mathbf{x}) + C_H(\mathbf{y}) = C_H(\mathbf{x} + \mathbf{y}),$$

where the "+" operator is just the bit-wise $\oplus$ of the operand vectors[11]. Further, it can be verified that for two vectors $\mathbf{u}, \mathbf{v} \in \{0,1\}^n$, we have:

$$\Delta(\mathbf{u}, \mathbf{v}) = wt(\mathbf{u} + \mathbf{v})$$

(see Exercise 1.12). Thus, we have

$$\begin{aligned} \min_{\mathbf{x} \neq \mathbf{y} \in \{0,1\}^4} \Delta(C_H(\mathbf{x}), C_H(\mathbf{y})) &= \min_{\mathbf{x} \neq \mathbf{y} \in \{0,1\}^4} wt(C_H(\mathbf{x} + \mathbf{y})) \\ &= \min_{\mathbf{x} \neq \mathbf{0} \in \{0,1\}^4} wt(C_H(\mathbf{x})), \end{aligned}$$

where the second equality follows from the observation that $\{\mathbf{x} + \mathbf{y} | \mathbf{x} \neq \mathbf{y} \in \{0,1\}^n\} = \{\mathbf{x} \in \{0,1\}^n | \mathbf{x} \neq \mathbf{0}\}$. Recall that $wt(C_H(\mathbf{x})) = 0$ if and only if $\mathbf{x} = \mathbf{0}$ and this completes the proof of (1.11). Combining (1.10) and (1.11), we conclude that C_H has a distance of 3. □

The second part of the proof could also be shown in the following manner. It can be verified that the Hamming code is the set $\{\mathbf{x} \cdot G_H | \mathbf{x} \in \{0,1\}^4\}$, where G_H is the following matrix (where we think $\mathbf{x}$ as a row vector).[12]

$$G_H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

For example, the first column in G_H gives the first codeword bit of x_1 and the fifth column of G_H gives the codeword bit $x_2 \oplus x_3 \oplus x_4$.

In fact, any binary code (of dimension k and block length n) that is generated[13] by a $k \times n$ matrix is called a *binary linear code*. (Both $C_\oplus$ and $C_{3,rep}$ are binary linear codes: see Exercise 1.13.) This implies the following simple fact.

Lemma 1.5.3. *For any binary linear code C and any two messages $\mathbf{x}$ and $\mathbf{y}$, $C(\mathbf{x}) + C(\mathbf{y}) = C(\mathbf{x} + \mathbf{y})$.*

[11] E.g. $(0,1,1,0) + (1,1,1,0) = (1,0,0,0)$.

[12] Indeed $(x_1, x_2, x_3, x_4) \cdot G_H = (x_1, x_2, x_3, x_4, x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_3 \oplus x_4, x_1 \oplus x_2 \oplus x_4)$, as desired.

[13] That is, $C = \{\mathbf{x} \cdot G | \mathbf{x} \in \{0,1\}^k\}$, where addition is the $\oplus$ operation and multiplication is the AND operation.

Proof. For any binary linear code, we have a generator matrix G. The following sequence of equalities (which follow from the distributivity and associativity properties of the Boolean EXOR and AND operators) proves the lemma:

$$\begin{aligned} C(\mathbf{x}) + C(\mathbf{y}) &= \mathbf{x} \cdot G + \mathbf{y} \cdot G \\ &= (\mathbf{x} + \mathbf{y}) \cdot G \\ &= C(\mathbf{x} + \mathbf{y}). \end{aligned}$$

□

We stress that in the lemma above, $\mathbf{x}$ and $\mathbf{y}$ need *not* be distinct. Note that due to the fact that $b \oplus b = 0$ for every $b \in \{0, 1\}$, $\mathbf{x} + \mathbf{x} = \mathbf{0}$, which along with the lemma above implies that $C(\mathbf{0}) = \mathbf{0}$.[14] We can infer the following result from the above lemma and the arguments used to prove (1.11) in the proof of Proposition 1.5.2.

Proposition 1.5.4. *For any binary linear code, its minimum distance is equal to the minimum Hamming weight of any non-zero codeword.*

Thus, we have seen that C_H has distance $d = 3$ and rate $R = \frac{4}{7}$ while $C_{3,rep}$ has distance $d = 3$ and rate $R = \frac{1}{3}$. Thus, the Hamming code is provably better than the repetition code (in terms of the tradeoff between rate and distance) and thus, answers Question 1.4.5 in the affirmative. The next natural question is

Question 1.5.5 (Codes better than C_H)**.** *Can we have a distance* 3 *code with a rate higher than that of* C_H*?*

We will address this question in the next section.

1.6 Hamming Bound

Now we switch gears to present our first tradeoff between redundancy (in the form of the dimension of a code) and its error-correction capability (in the form of its distance). In particular, we will first prove a special case of the so-called Hamming bound for a distance of 3.

We begin with another definition.

Definition 1.6.1 (Hamming Ball)**.** *For any vector* $\mathbf{x} \in [q]^n$*,*

$$B(\mathbf{x}, e) = \{\mathbf{y} \in [q]^n | \Delta(\mathbf{x}, \mathbf{y}) \leq e\}.$$

In other words, a Hamming ball of *radius* e, centered at $\mathbf{x}$, contains all vectors within Hamming distance at most e of $\mathbf{x}$.

Next, we prove an upper bound on the dimension of *every* code with distance 3.

[14] This of course should not be surprising as for any matrix G, we have $\mathbf{0} \cdot G = \mathbf{0}$.

Theorem 1.6.2 (Hamming bound for $d = 3$)**.** *Every binary code with block length n, dimension k, distance $d = 3$ satisfies*

$$k \leq n - \log_2(n+1).$$

Proof. Given any two codewords, $\mathbf{c}_1 \neq \mathbf{c}_2 \in C$, the following is true (as C has distance[15] 3):

$$B(\mathbf{c}_1, 1) \cap B(\mathbf{c}_2, 1) = \emptyset. \tag{1.12}$$

See Figure 1.4 for an illustration.

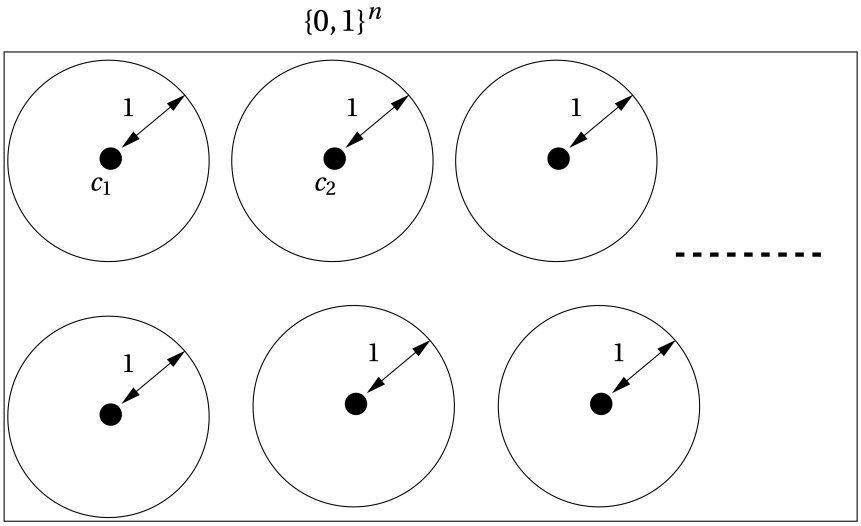

Figure 1.4: Hamming balls of radius 1 are disjoint. The figure is technically not correct: the balls above are actually balls in the Euclidean space, which is easier to visualize than the Hamming space.

Note that for all $\mathbf{x} \in \{0,1\}^n$ (see Exercise 1.16),

$$|B(\mathbf{x}, 1)| = n + 1. \tag{1.13}$$

Now consider the union of all Hamming balls centered around some codeword; their union is a subset of $\{0,1\}^n$. In other words,

$$\left| \bigcup_{\mathbf{c} \in C} B(\mathbf{c}, 1) \right| \leq 2^n. \tag{1.14}$$

[15] Assume that $\mathbf{y} \in B(\mathbf{c}_1, 1) \cap B(\mathbf{c}_2, 1)$, that is $\Delta(\mathbf{y}, \mathbf{c}_1) \leq 1$ and $\Delta(\mathbf{y}, \mathbf{c}_2) \leq 1$. Thus, by the triangle inequality $\Delta(\mathbf{c}_1, \mathbf{c}_2) \leq 2 < 3$, which is a contradiction.

As (1.12) holds for every pair of distinct codewords,

$$\begin{aligned}\left|\bigcup_{\mathbf{c}\in C} B(\mathbf{c},1)\right| &= \sum_{\mathbf{c}\in C} |B(\mathbf{c},1)| \\ &= \sum_{\mathbf{c}\in C} (n+1) \qquad (1.15)\\ &= 2^k \cdot (n+1), \qquad (1.16)\end{aligned}$$

where (1.15) follows from (1.13) and (1.16) follows from the fact that C has dimension k. Combining (1.16) and (1.14), we get

$$2^k(n+1) \leq 2^n,$$

or equivalently

$$2^k \leq \frac{2^n}{n+1}.$$

Taking $\log_2$ of both sides we get the desired bound:

$$k \leq n - \log_2(n+1).$$

□

Thus, Theorem 1.6.2 shows that for $n = 7$, C_H has the largest possible dimension for any binary code of block length 7 and distance 3 (as for $n = 7$, $n - \log_2(n+1) = 4$). In particular, it also answers Question 1.5.5 for $n = 7$ in the negative. Next, will present the general form of Hamming bound.

1.7 Generalized Hamming Bound

We start with a new notation.

Definition 1.7.1. *A code $C \subseteq \Sigma^n$ with dimension k and distance d will be called an $(n,k,d)_\Sigma$ code. We will also refer to it as an $(n,k,d)_{|\Sigma|}$ code.*

We now proceed to generalize Theorem 1.6.2 to any distance d (from $d = 3$).

Theorem 1.7.2 (Hamming Bound for any d)**.** *For every $(n,k,d)_q$ code*

$$k \leq n - \log_q \left(\sum_{i=0}^{\left\lfloor \frac{d-1}{2} \right\rfloor} \binom{n}{i} (q-1)^i \right).$$

Proof. The proof is a straightforward generalization of the proof of Theorem 1.6.2. For notational convenience, let $e = \left\lfloor \frac{d-1}{2} \right\rfloor$. Given any two codewords, $\mathbf{c}_1 \neq \mathbf{c}_2 \in C$, the following is true (as C has distance[16] d):

$$B(\mathbf{c}_1, e) \cap B(\mathbf{c}_2, e) = \emptyset. \tag{1.17}$$

We claim that for all $\mathbf{x} \in [q]^n$,

$$|B(\mathbf{x}, e)| = \sum_{i=0}^{e} \binom{n}{i} (q-1)^i. \tag{1.18}$$

Indeed any vector in $B(\mathbf{x}, e)$ must differ from $\mathbf{x}$ in exactly $0 \leq i \leq e$ positions. In the summation, $\binom{n}{i}$ is the number of ways of choosing the differing i positions and in each such position, a vector can differ from $\mathbf{x}$ in $q-1$ ways.

Now consider the union of all Hamming balls centered around a codeword. Obviously, their union is a subset of $[q]^n$. In other words,

$$\left| \bigcup_{\mathbf{c} \in C} B(\mathbf{c}, e) \right| \leq q^n. \tag{1.19}$$

As (1.17) holds for every pair of distinct codewords,

$$\begin{aligned} \left| \bigcup_{\mathbf{c} \in C} B(\mathbf{c}, e) \right| &= \sum_{\mathbf{c} \in C} |B(\mathbf{c}, e)| \\ &= q^k \sum_{i=0}^{e} \binom{n}{i} (q-1)^i, \end{aligned} \tag{1.20}$$

where (1.20) follows from (1.18) and the fact that C has dimension k. Combining (1.20) and (1.19) and taking $\log_q$ of both sides we will get the desired bound:

$$k \leq n - \log_q \left(\sum_{i=0}^{e} \binom{n}{i} (q-1)^i \right).$$

□

Note that the Hamming bound gives a partial answer to Question 1.4.4. In particular, any code of distance d can have rate R at most

$$1 - \frac{\log_q \left(\sum_{i=0}^{e} \binom{n}{i} (q-1)^i \right)}{n}.$$

Further, the Hamming bound also leads to the following definition:

[16] Assume that $\mathbf{y} \in B(\mathbf{c}_1, e) \cap B(\mathbf{c}_2, e)$, that is $\Delta(\mathbf{y}, \mathbf{c}_1) \leq e$ and $\Delta(\mathbf{y}, \mathbf{c}_2) \leq e$. Thus, by the triangle inequality, $\Delta(\mathbf{c}_1, \mathbf{c}_2) \leq 2e \leq d-1$, which is a contradiction.

Definition 1.7.3. *Codes that meet Hamming bound are called* perfect codes.

In other words, a perfect code leads to the following perfect "packing": if one constructs Hamming balls of radius $\left\lfloor \frac{d-1}{2} \right\rfloor$ around all the codewords, then we would cover the entire ambient space, i.e. every possible vector will lie in one of these Hamming balls.

One example of perfect code is the $(7, 4, 3)_2$ Hamming code that we have seen in this chapter (so is the family of general Hamming codes that we will see in the next chapter). A natural question to ask is if

Question 1.7.4 (Perfect Codes)**.** *Other than the Hamming codes, are there any other perfect (binary) codes?*

We will see the answer in Section 2.4.

1.8 Family of codes

Until now, we have mostly studied specific codes with *fixed* block lengths and dimensions. However, when we perform an asymptotic study of codes, it makes more sense to talk about a family of codes and study their asymptotic rate and distance. We define these notions next.

Definition 1.8.1 (Code families, Rate and Distance)**.** *Let $\{n_i\}_{i\geq 1}$ be an increasing sequence of block lengths and suppose there exists sequences $\{k_i\}_{i\geq 1}$, $\{d_i\}_{i\geq 1}$ and $\{q_i\}_{i\geq 1}$ such that for all $i \geq 1$ there exists an $(n_i, k_i, d_i)_{q_i}$ code C_i. Then the sequence $\mathcal{C} = \{C_i\}_{i\geq 1}$ is a family of codes. The rate of $\mathcal{C}$ is defined as*

$$R(\mathcal{C}) = \lim_{i\to\infty} \left\{ \frac{k_i}{n_i} \right\},$$

when the limit exists. The relative distance of $\mathcal{C}$ is defined as

$$\delta(\mathcal{C}) = \lim_{i\to\infty} \left\{ \frac{d_i}{n_i} \right\},$$

when the limit exists. If for all $i \geq 1$, $q_i = q$ then $\mathcal{C}$ is referred to as a family of q-ary codes.[17] [18]

For instance, we will in Section 2.4 see that Hamming code of Section 1.5 can be extended to an entire family of codes. Specifically, $\mathcal{C}_H = \{C_i\}_{i\in\mathbb{Z}^+}$, with C_i being an $(n_i.k_i, d_i)$-code with $n_i = 2^i - 1, k_i = 2^i - i - 1, d_i = 3$ and thus,

$$R(\mathcal{C}_H) = \lim_{i\to\infty} 1 - \frac{i}{2^i - 1} = 1,$$

[17] In all codes we will study these limits will exist, but of course it is possible to construct families of codes where the limits do not exist.

[18] While a central goal is to understand q-ary families of codes, families over growing alphabets turn out to be useful both to illustrate ideas and to get interesting q-ary families.

and

$$\delta(\mathcal{C}_H) = \lim_{i \to \infty} \frac{3}{2^i - 1} = 0.$$

A significant focus of this text from now on will be on families of codes. This is necessary as we will be studying the asymptotic behavior of algorithms on codes, which does not make sense for a fixed code. For example, when we say that a decoding algorithm for a code $\mathcal{C}$ takes $O(n^2)$ time, we would be implicitly assuming that $\mathcal{C}$ is a family of codes and that the algorithm has an $O(n^2)$ running time when the block length is large enough. From now on, unless mentioned otherwise, whenever we talk about a code, we will be implicitly assuming that we are talking about a family of codes.

Given that we can only formally talk about asymptotic run time of algorithms, we now also state our formal notion of efficient algorithms:

Note

We'll call an algorithm related to a code of block length n to be *efficient* if it runs in time polynomial in n.

For all the specific codes that we will study in this book, the corresponding family of codes will be a "family" in a more natural sense. By this we mean that all the specific codes in a family of codes will be the "same" code except with different parameters. A bit more formally, we will consider families $\{C_i\}_{i \geq 1}$, where given only the 'index' i, one can compute a sufficient description of C_i efficiently.[19]

Finally, the definition of a family of codes allows us to present the final version of the big motivating question for the book. The last formal version of the main question we considered was Question 1.4.4, where we were interested in the tradeoff of rate R and distance d. The comparison was somewhat unfair because R was a ratio while d was an integer. A more appropriate comparison should be between rate R and the relative distance δ. Further, we would be interested in tackling the main motivating question for families of codes, which results in the following final version:

Question 1.8.2 (Main Question- formal)**.** *Given q, what is the optimal tradeoff between $R(\mathcal{C})$ and $\delta(\mathcal{C})$ that can be achieved by some family $\mathcal{C}$ of q-ary codes?*

A natural special case of Question 1.8.2 is whether the rate and relative distance of a family of codes can be simultaneously positive. We formulate this special case as a separate question below.

Question 1.8.3 (Asymptotically Good Codes)**.** *Does there exist a constant q and a q-ary family of codes $\mathcal{C}$ such that $R(\mathcal{C}) > 0$ and $\delta(\mathcal{C}) > 0$ hold* simultaneously*?*

Codes that have the above property are called *asymptotically good*. For the curious reader, we will present many asymptotically good codes in the rest of this book, though a priori the existence of these is not immediate.

[19] We stress that this is not *always* going to be the case. In particular, we will consider "random" codes where this efficient constructibility will not be true.

1.9 Exercises

Exercise 1.1. *Show that every t-error correcting code is also t-error detecting but not necessarily the other way around.*

Exercise 1.2. *Prove Proposition 1.3.9.*

Exercise 1.3. *Show that for every integer n, there is no code with block length n that can handle arbitrary number of errors.*

Exercise 1.4. *Prove Proposition 1.3.10.*

Exercise 1.5. *A distance function on Σ^n (i.e. $d : \Sigma^n \times \Sigma^n \to \mathbb{R}$) is called a* metric *if the following conditions are satisfied for every $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \Sigma^n$:*

1. $d(\mathbf{x}, \mathbf{y}) \geq 0$.
2. $d(\mathbf{x}, \mathbf{y}) = 0$ *if and only if* $\mathbf{x} = \mathbf{y}$.
3. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$.
4. $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$. *(This property is called the* triangle inequality*.)*

Prove that the Hamming distance is a metric.

Exercise 1.6. *Let C be a code with distance d for even d. Then argue that C can correct up to $d/2 - 1$ many errors but cannot correct $d/2$ errors. Using this or otherwise, argue that if a code C is t-error correctable then it either has a distance of $2t + 1$ or $2t + 2$.*

Exercise 1.7. *In this exercise, we will see that one can convert arbitrary codes into code with slightly different parameters:*

1. *Prove that if there exists an $(n, k, d)_\Sigma$ code then there also exists an $(n - 1, k, d - 1)_\Sigma$ code. Specifically, show how to convert an $(n, k, d)_\Sigma$ code C into an $(n - 1, k, d - 1)_\Sigma$ code.*
2. *For odd d, prove that if an $(n, k, d)_2$ code exists, then there also exists an $(n+1, k, d+1)_2$ code. Specifically, show how to convert an $(n, k, d)_2$ code C into an $(n + 1, k, d + 1)_2$ code.*

Note: *Your conversion should not assume anything else about the code other than the parameters of the code C. Also your conversion should work for every $n, k, d \geq 1$ and every Σ.*

Exercise 1.8. *In this problem we will consider a noise model that has both errors and erasures. In particular, let C be an $(n, k, d)_\Sigma$ code. As usual a codeword $\mathbf{c} \in C$ is transmitted over the channel and the received word is a vector $\mathbf{y} \in (\Sigma \cup \{?\})^n$, where as before a ? denotes an erasure. We will use s to denote the number of erasures in $\mathbf{y}$ and e to denote the number of (non-erasure) errors that occurred during transmission. To decode such a vector means to output a codeword $\mathbf{c} \in C$ such that the number of positions where $\mathbf{c}$ disagree with $\mathbf{y}$ in the $n - s$ non-erased positions is at most e. For the rest of the problem assume that*

$$2e + s < d. \tag{1.21}$$

1. *Argue that the output of the decoder for any C under (1.21) is unique.*
2. *Let C be a binary code (but not necessarily linear). Assume that there exists a decoder D that can correct from $< d/2$ many errors in $T(n)$ time. Then under (1.21) one can perform decoding in time $O(T(n))$.*

Exercise 1.9. *Define codes other than C_H with $k = 4, n = 7$ and $d = 3$.*

Hint: *Refer to the proof of Proposition 1.5.2 to figure out the properties needed from the three parities.*

Exercise 1.10. *Argue that if $wt(\mathbf{x}) = 1$ then at least two parity check bits in $(x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_2 \oplus x_4, x_1 \oplus x_3 \oplus x_4)$ are 1.*

Exercise 1.11. *Argue that if $wt(\mathbf{x}) = 2$ then at least one parity check bit in $(x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_2 \oplus x_4, x_1 \oplus x_3 \oplus x_4)$ is 1.*

Exercise 1.12. *Prove that for any $\mathbf{u}, \mathbf{v} \in \{0, 1\}^n$, $\Delta(\mathbf{u}, \mathbf{v}) = wt(\mathbf{u} + \mathbf{v})$.*

Exercise 1.13. *Argue that $C_\oplus$ and $C_{3,rep}$ are binary linear codes.*

Exercise 1.14. *Let G be a generator matrix of an $(n, k, d)_2$ binary linear code. Then G has at least kd ones in it.*

Exercise 1.15. *Argue that in any binary linear code, either all codewords begin with a 0 of exactly half of the codewords begin with a 0.*

Exercise 1.16. *Prove (1.13).*

Exercise 1.17. *Show that there is no binary code with block length 4 that achieves the Hamming bound.*

Exercise 1.18. $^{(*)}$ *There are n people in a room, each of whom is given a black/white hat chosen uniformly at random (and independent of the choices of all other people). Each person can see the hat color of all other people, but not their own. Each person is asked if they wish to guess their own hat color. They can either guess, or abstain. Each person makes their choice without knowledge of what the other people are doing. They either win collectively, or lose collectively. They win if at least one person does not abstain* and *all the people who don't abstain guess their hat color correctly. They lose if all people abstain, or if some person guesses their color incorrectly. Your goal below is to come up with a strategy that will allow the n people to win with pretty high probability. We begin with a simple warm-up:*

1. *Argue that the n people can win with probability at least $\frac{1}{2}$.*

Next we will see how one can really bump up the probability of success with some careful modeling, and some knowledge of Hamming codes. (Below are assuming knowledge of the general Hamming code (see Section 2.4). If you do not want to skip ahead, you can assume that $n = 7$ in the last part of this problem.)

2. *Lets say that a directed graph G is a subgraph of the n-dimensional hypercube if its vertex set is $\{0,1\}^n$ and if $u \to v$ is an edge in G, then u and v differ in at most one coordinate. Let $K(G)$ be the number of vertices of G with in-degree at least one, and out-degree zero. Show that the probability of winning the hat problem equals the maximum, over directed subgraphs G of the n-dimensional hypercube, of $K(G)/2^n$.*

3. *Using the fact that the out-degree of any vertex is at most n, show that $K(G)/2^n$ is at most $\frac{n}{n+1}$ for any directed subgraph G of the n-dimensional hypercube.*

4. *Show that if $n = 2^r - 1$, then there exists a directed subgraph G of the n-dimensional hypercube with $K(G)/2^n = \frac{n}{n+1}$.*
   Hint: *This is where the Hamming code comes in.*

1.10 Bibliographic Notes

Coding theory owes its origin to two remarkable papers: one by Shannon [37] and the other by Hamming [21] both of which were published within a couple of years of each other. Shannon's paper defined the BSC_p channel (among others) and defined codes in terms of its encoding function. Shannon's paper also explicitly defined the decoding function. Hamming's work defined the notion of codes as in Definition 1.2.1 as well as the notion of Hamming distance. Both the Hamming bound and the Hamming code are (not surprisingly) due to Hamming. The specific definition of Hamming code that we used in this book was the one proposed by Hamming and is also mentioned in Shannon's paper (which pre-dates Hamming's) with attribution to Hamming. The notion of erasures was defined by Elias [14]. Most exercises of this chapter are based on [21]. The hat problem in Exercise 1.18 is from Ebert, Merkle and Vollmer [13].

Chapter 2

A Look at Some Nicely Behaved Codes: Linear Codes

One motivation for the topic of this chapter is the following question: How we can represent a code? Or more specifically, how many bits does it take to describe a code $C : [q]^k \longrightarrow [q]^n$? In general, a code $C : [q]^k \longrightarrow [q]^n$ can be stored using nq^k symbols from $[q]$ (n symbols for each of the q^k codewords) or $nq^k \log q$ bits. For constant rate codes, this is exponential space, which is prohibitive even for modest values of k like $k = 100$. A natural question is whether we can do better. To have any hope of doing so, a succinct representation the code must have some extra structure. It turns out that one broad class of codes that do possess extra structure than general codes, is what are called *linear codes*. We have already seen binary linear codes in Section 1.5, that is: $C \subseteq \{0,1\}^n$ is a linear code if for all $\mathbf{c}_1, \mathbf{c}_2 \in C$, $\mathbf{c}_1 + \mathbf{c}_2 \in C$, where the "+" denotes bit-wise XOR. In this chapter, we will see more general linear codes. We will see that they not only offer enough structure to get succinct representations, but they also possess several other nice properties.

To define general linear codes, we first need to introduce general finite fields and vector spaces over such fields and we do so first before returning to codes.

2.1 Groups and Finite Fields

To define linear subspaces, we will need to work with (finite) fields. At a high level, we need finite fields since when we talk about codes, we deal with finite symbols/numbers and we want to endow these symbols with the same math that makes arithmetic over real numbers work. Finite fields accomplish this precise task. We begin with a quick overview of fields. We start with the more elementary notion of a group.

Definition 2.1.1. *A group $\mathbb{G}$ is given by a pair $(S, \circ)$, where S is the set of elements and $\circ$ is a function $S \times S \to S$ with the following properties:*

- Closure: *For every $a, b \in S$, we have $a \circ b \in S$.*

- ASSOCIATIVITY: $\circ$ *is associative: that is, for every* $a, b, c \in S$, $a \circ (b \circ c) = (a \circ b) \circ c$.
- IDENTITY: *There exists* distinct *a special elements* $e \in S$ *such that for every* $a \in S$ *we have* $a \circ e = e \circ a = a$.
- INVERSE: *For every* $a \in S$, *there exists its unique* inverse a^{-1} *such that* $a \circ a^{-1} = a^{-1} \circ a = e$.

If $\mathbb{G} = (S, \circ)$ *satisfies all the properties except the existence of inverses then* $\mathbb{G}$ *is called a monoid. We say* $\mathbb{G}$ *is* commutative *if for every* $a, b \in S$, $a \circ b = b \circ a$.

We often use the same letter to denote the group (or other algebraic structures) and the set of elements.

We now turn to the definition of a field. Informally speaking, a field is a set of elements on which one can do addition, subtraction, multiplication and division and still stay in the set.

Definition 2.1.2. *A field* $\mathbb{F}$ *is given by a triple* $(S, +, \cdot)$, *where* S *is the set of elements and* $+, \cdot$ *are functions* $S \times S \to S$ *with the following properties:*

- *Addition:* $(S, +)$ *form a commutative group with identity element denoted* $0 \in S$.
- *Multiplication:* $(S \setminus \{0\}, \cdot)$ *form a commutative group with identity element* $1 \in S \setminus \{0\}$.[1]
- *Distributivity:* $\cdot$ *distributes over* $+$: *that is, for every* $a, b, c \in S$, $a \cdot (b + c) = a \cdot b + a \cdot c$.

Again we typically use the same letter to denote the field and its set of elements. We also use $-a$ to denote the additive inverse of $a \in \mathbb{F}$ and a^{-1} to denote the multiplicative inverse of $a \in \mathbb{F} \setminus \{0\}$.

We note that in the above definition we have not explicitly argued that $a \cdot 0 = 0 = 0 \cdot a$ for any $a \in S$. (Technically this means $(S, \cdot)$ is a *commutative monoid*.) This is because this property is implied by Definition 2.1.2– see Exercise 2.1.

With the usual semantics for $+$ and $\cdot$, $\mathbb{R}$ (set of real number) is a field, but $\mathbb{Z}$ (set of integers) is not a field as division of two integers results in a rational number that need not be an integer (the set of rational numbers itself is a field though: see Exercise 2.2). In this course, we will exclusively deal with *finite fields*. As the name suggests these are fields with a finite set of elements. (We will overload notation and denote the size of a field $|\mathbb{F}| = |S|$.) The following is a well known result.

Theorem 2.1.3 (Size of Finite Fields)**.** *Every finite field has size* p^s *for some prime* p *and integer* $s \geq 1$. *Conversely for every prime* p *and integer* $s \geq 1$ *there exists a field* $\mathbb{F}$ *of size* p^s.

[1] Note that we do not include 0 since it does not have a multiplicative inverse.

One example of a finite field that we have seen is the field with $S = \{0, 1\}$, which we will denote by $\mathbb{F}_2$ (we have seen this field in the context of binary linear codes). For $\mathbb{F}_2$, addition is the XOR operation, while multiplication is the AND operation. The additive inverse of an element in $\mathbb{F}_2$ is the number itself while the multiplicative inverse of 1 is 1 itself.

Let p be a prime number. Then the integers modulo p form a field, denoted by $\mathbb{F}_p$ (and also by $\mathbb{Z}_p$), where the addition and multiplication are carried out modulo p. For example, consider $\mathbb{F}_7$, where the elements are $\{0, 1, 2, 3, 4, 5, 6\}$. We have $(4 + 3) \mod 7 = 0$ and $4 \cdot 4 \mod 7 = 2$. Further, the additive inverse of 4 is 3 as $(3+4) \mod 7 = 0$ and the multiplicative inverse of 4 is 2 as $4 \cdot 2 \mod 7 = 1$.

More formally, we prove the following result.

Lemma 2.1.4. *Let p be a prime. Then $\mathbb{F}_p = (\{0, 1, \ldots, p-1\}, +_p, \cdot_p)$ is a field, where $+_p$ and $\cdot_p$ are addition and multiplication modulo p.*

Proof. The properties of associativity, commutativity, distributivity and identities hold for integers and hence, they hold for $\mathbb{F}_p$. The closure property follows since both the "addition" and "multiplication" are done modulo p, which implies that for any $a, b \in \{0, \ldots, p-1\}$, $a +_p b, a \cdot_p b \in \{0, \ldots, p-1\}$. Thus, to complete the proof, we need to prove the existence of unique additive and multiplicative inverses.

Fix an arbitrary $a \in \{0, \ldots, p-1\}$. Then we claim that its additive inverse is $p - a \mod p$. It can be verified that $a + p - a = 0 \mod p$. Next we argue that this is the unique additive inverse. To see this note that the sequence $a, a+1, a+2, \ldots, a+p-1$ are p consecutive numbers and thus, exactly one of them is a multiple of p, which happens for $b = p - a \mod p$, as desired.

Now fix an $a \in \{1, \ldots, p-1\}$. Next we argue for the existence of a unique multiplicative universe a^{-1}. Consider the set of numbers $T = \{a \cdot_p b | b \in \{1, \ldots, p-1\}\}$. We claim that all these numbers are unique. To see this, note that if this is not the case, then there exist $b_1 \neq b_2 \in \{0, 1, \ldots, p-1\}$ such that $a \cdot b_1 = a \cdot b_2 \mod p$, which in turn implies that $a \cdot (b_1 - b_2) = 0 \mod p$. Since a and $b_1 - b_2$ are non-zero numbers, this implies that p divides $a \cdot (b_1 - b_2)$. Further, since a and $|b_1 - b_2|$ are both at most $p-1$, this implies that multiplying a and $(b_1 - b_2) \mod p$ results in p, which is a contradiction since p is prime. Thus, we have argued that $|T| = p - 1$ and since each number in T is in $[p-1]$, we have that $T = [p-1]$. Thus, we can conclude that there exists a unique element b such that $a \cdot b = 1 \mod p$ and thus, b is the required a^{-1}. □

One might think that there could be different finite fields with the same number of elements. However, this is not the case:

Theorem 2.1.5. *For every prime power q there is a* unique *finite field with q elements (up to isomorphism[2]).*

Thus, we are justified in just using $\mathbb{F}_q$ to denote a finite field on q elements.

[2]An isomorphism $\phi : S \to S'$ is a bijective map (such that $\mathbb{F} = (S, +, \cdot)$ and $\mathbb{F}' = (S', \oplus, \circ)$ are fields) where for every $a_1, a_2 \in S$, we have $\phi(a_1 + a_2) = \phi(a_1) \oplus \phi(a_2)$ and $\phi(a_1 \cdot a_2) = \phi(a_1) \circ \phi(a_2)$. In other words, an ismorphisms is a map between representations that 'preserves' the effect of operators on elements.

2.2 Vector Spaces and Linear Subspaces

Definition 2.2.1 (Vector Space)**.** *A vector space V over a field $\mathbb{F}$ is given by a triple $(T, +, \cdot)$ such that $(T, +)$ form a commutative group and $\cdot$, referred to as the scalar product, is a function $\mathbb{F} \times T \to T$ such that for every $a, b \in \mathbb{F}$ and $\mathbf{u}, \mathbf{v} \in T$ we have $(a+b)\cdot\mathbf{u} = a\cdot\mathbf{u}+b\cdot\mathbf{u}$ and $a \cdot (\mathbf{u} + \mathbf{v}) = a \cdot \mathbf{u} + a \cdot \mathbf{v}$.*

The most common vector space we will focus on is $\mathbb{F}^n$ with $+$ representing coordinatewise addition in $\mathbb{F}$ and $a \cdot \mathbf{u}$ representing the coordinatewise scaling of $\mathbf{u}$ by a.

We are finally ready to define the notion of linear subspaces of $\mathbb{F}^n$.

Definition 2.2.2 (Linear Subspace)**.** *A* non-empty *subset $S \subseteq \mathbb{F}^n$ is a linear subspace if the following properties hold:*

1. *For every $\mathbf{x}, \mathbf{y} \in S$, $\mathbf{x} + \mathbf{y} \in S$, where the addition is vector addition over $\mathbb{F}$ (that is, do addition componentwise over $\mathbb{F}$).*

2. *For every $a \in \mathbb{F}$ and $\mathbf{x} \in S$, $a \cdot \mathbf{x} \in S$, where the multiplication is done componentwise over $\mathbb{F}$.*

Here is a (trivial) example of a linear subspace of $\mathbb{F}_5^3$:

$$S_1 = \{(0,0,0), (1,1,1), (2,2,2), (3,3,3), (4,4,4)\}. \tag{2.1}$$

Note that for example $(1,1,1) + (3,3,3) = (4,4,4) \in S_1$ and $2 \cdot (4,4,4) = (3,3,3) \in S_1$ as required by the definition. Here is another somewhat less trivial example of a linear subspace over $\mathbb{F}_3^3$:

$$S_2 = \{(0,0,0), (1,0,1), (2,0,2), (0,1,1), (0,2,2), (1,1,2), (1,2,0), (2,1,0), (2,2,1)\}. \tag{2.2}$$

Note that $(1,0,1) + (0,2,2) = (1,2,0) \in S_2$ and $2 \cdot (2,0,2) = (1,0,1) \in S_2$ as required.

Remark 2.2.3. *Note that the second property implies that $\mathbf{0}$ is contained in every linear subspace. Further for any subspace over $\mathbb{F}_2$, the second property is redundant: see Exercise 2.5.*

Before we state some properties of linear subspaces, we state some relevant definitions.

Definition 2.2.4 (Span)**.** *Given a set $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_\ell\}$. The* span *of B is the set of vectors*

$$\left\{ \sum_{i=1}^{\ell} a_i \cdot \mathbf{v}_i \middle| a_i \in \mathbb{F}_q \text{ for every } i \in [\ell] \right\}.$$

Definition 2.2.5 (Linear (in)dependence of vectors)**.** *We say that* $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k$ *are* linearly independent *if for every* $1 \leq i \leq k$ *and for every* $(k-1)$*-tuple* $(a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k) \in \mathbb{F}_q^{k-1}$*,*

$$\mathbf{v}_i \neq a_1\mathbf{v}_1 + \ldots + a_{i-1}\mathbf{v}_{i-1} + a_{i+1}\mathbf{v}_{i+1} + \ldots + a_k\mathbf{v}_k.$$

In other words, $\mathbf{v}_i$ *is not in the span of the set* $\{\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_n\}$ *for every* $1 \leq i \leq k$*. We say that* $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_k$ *are* linearly dependent *if they are not linearly independent.*

For example the vectors $(1, 0, 1), (1, 1, 1) \in S_2$ are linearly independent since

- $a_1 \cdot (1, 0, 1) = (a_1, 0, a_1) \neq (1, 1, 1)$ for any $a_1 \in \{0, 1\}$.
- $a_2 \cdot (1, 1, 1) = (a_2, a_2, a_2) \neq (1, 0, 1)$ for any $a_2 \in \{0, 1\}$.

Definition 2.2.6 (Rank of a matrix)**.** *The* rank *of matrix in* $\mathbb{F}_q^{k\times k}$ *is the maximum number of linearly independent rows (or columns). A matrix in* $\mathbb{F}_q^{k\times n}$ *with rank* $\min(k, n)$ *is said to have* full *rank.*

One can define the *row (column) rank* of a matrix as the maximum number of linearly independent rows (columns). However, it is a well-known theorem that the row rank of a matrix is the same as its column rank. For example, the matrix below over $\mathbb{F}_3$ has full rank (see Exercise 2.6):

$$G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}. \tag{2.3}$$

Any linear subspace satisfies the following properties (the full proof can be found in any standard linear algebra textbook).

Theorem 2.2.7. *If* $S \subseteq {\mathbb{F}_q}^n$ *is a linear subspace then*

1. $|S| = q^k$ *for some* $k \geq 0$*. The parameter* k *is called the* dimension *of* S*.*
2. *There exists at least one set of linearly independent vectors* $\mathbf{v}_1, ..., \mathbf{v}_k \in S$ *called* basis elements *such that every* $\mathbf{x} \in S$ *can be expressed as* $\mathbf{x} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + ... + a_k\mathbf{v}_k$ *where* $a_i \in \mathbb{F}_q$ *for* $1 \leq i \leq k$*. In other words, there exists a full rank* $k \times n$ *matrix* G *(also known as a* generator matrix*) with entries from* $\mathbb{F}_q$ *such that every* $\mathbf{x} \in S$*,* $\mathbf{x} = (a_1, a_2, \ldots, a_k) \cdot G$ *where*

$$G = \begin{pmatrix} \longleftarrow & \mathbf{v}_1 & \longrightarrow \\ \longleftarrow & \mathbf{v}_2 & \longrightarrow \\ & \vdots & \\ \longleftarrow & \mathbf{v}_k & \longrightarrow \end{pmatrix}.$$

3. *There exists a full rank* $(n-k) \times n$ *matrix* H *(called a* parity check matrix*) such that for every* $\mathbf{x} \in S$*,* $H\mathbf{x}^T = \mathbf{0}$*.*

4. G and H are orthogonal*, that is, $G \cdot H^T = \mathbf{0}$.*

Proof Sketch.

Property 1. We begin with the proof of the first property. For the sake of contradiction, let us assume that $q^k < |S| < q^{k+1}$, for some $k \geq 0$. Iteratively, we will construct a set of linearly independent vectors $B \subseteq S$ such that $|B| \geq k+1$. Note that by the definition of a linear subspace the span of B should be contained in S. However, this is a contradiction as the size of the span of B is at least[3] $q^{k+1} > |S|$.

To complete the proof, we show how to construct the set B in a greedy fashion. In the first step pick $\mathbf{v}_1$ to be any non-zero vector in S and set $B \leftarrow \{\mathbf{v}_1\}$ (we can find such a vector as $|S| > q^k \geq 1$). Now say after the step t (for some $t \leq k$), $|B| = t$. Now the size of the span of the current B is $q^t \leq q^k < |S|$. Thus there exists a vector $\mathbf{v}_{t+1} \in S \setminus B$ that is linearly independent of vectors in B. Set $B \leftarrow B \cup \{\mathbf{v}_{t+1}\}$. Thus, we can continue building B until $|B| = k+1$, as desired.

Property 2. We first note that we can pick $B = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ to be any set of k linearly independent vectors– this just follows from the argument above for **Property** 1.1. This is because the span of B is contained in S. However, since $|S| = q^k$ and the span of B has q^k vectors, the two have to be the same.

Property 3. Property 3 above follows from another fact that every linear subspace S has a null space $N \subseteq \mathbb{F}_q^n$ such that for every $\mathbf{x} \in S$ and $\mathbf{y} \in N$, $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Further, it is known that N itself is a linear subspace of dimension $n-k$. (The claim that N is also a linear subspace follows from the following two facts: for every $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{F}_q^n$, (i) $\langle \mathbf{x}, \mathbf{y}+\mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle$ and (ii) for any $a \in \mathbb{F}_q$, $\langle \mathbf{x}, a\mathbf{y} \rangle = a \cdot \langle \mathbf{x}, \mathbf{y} \rangle$.) In other words, there exists a generator matrix H for it. This matrix H is called the parity check matrix of S.

Property 4. See Exercise 2.9. □

As examples, the linear subspace S_1 in (2.1) has as one of its generator matrices

$$G_1 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

and as one of its parity check matrices

$$H_1 = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 2 & 1 \end{pmatrix}.$$

Further, the linear subspace S_2 in (2.2) has G_2 as one of its generator matrices and has the following as one of its parity check matrices

$$H_2 = \begin{pmatrix} 1 & 1 & 2 \end{pmatrix}.$$

Finally, we state another property of linear subspaces that is useful.

[3] See Exercise 2.8.

Lemma 2.2.8. *Given matrix G of dimension $k \times n$ that is a generator matrix of subspace S_1 and matrix H of dimension $(n-k) \times n$ that is a parity check matrix of subspace S_2 such that $GH^T = \mathbf{0}$, then $S_1 = S_2$.*

Proof. We first prove that $S_1 \subseteq S_2$. Given any $\mathbf{c} \in S_1$, there exists $\mathbf{x} \in \mathbb{F}_q^k$ such that $\mathbf{c} = \mathbf{x}G$. Then,

$$H \cdot \mathbf{c}^T = H \cdot (\mathbf{x}G)^T = HG^T\mathbf{x}^T = \left(GH^T\right)^T \mathbf{x}^T = \mathbf{0},$$

which implies that $\mathbf{c} \in S_2$, as desired.

To complete the proof note that as H has full rank, its null space (or S_2) has dimension $n-(n-k) = k$ (this follows from a well known fact from linear algebra called the *rank-nullity theorem*). Now as G has full rank, the dimension of S_1 is also k. Thus, as $S_1 \subseteq S_2$, it has to be the case that $S_1 = S_2$.[4] □

2.3 Linear Codes and Basic Properties

We now return to the topic of codes and introduce the central concept for this chapter as well as much of this text.

Definition 2.3.1 (Linear Codes)**.** *Let q be a prime power (i.e. $q = p^s$ for some prime p and integer $s \geq 1$). $C \subseteq \mathbb{F}_q^n$ is a* linear code *if it is a* linear subspace *of $\mathbb{F}_q^n$. If C has dimension k and distance d then it will be referred to as an $[n, k, d]_q$ or just an $[n, k]_q$ code.*

Theorem 2.2.7 now gives two alternate characterizations of an $[n, k]_q$ linear code C: first, C is generated by a $k \times n$ generator matrix G. Second, C is defined by a $(n-k) \times n$ parity check matrix H. Since these are important concepts for us, we define these formally below before giving examples and consequences.

Definition 2.3.2 (Generator and Parity Check Matrices)**.** *If C is an $[n, k]_q$ linear code then there exists a matrix $G \in \mathbb{F}_q^{k \times n}$ of rank k satisfying*

$$C = \{\mathbf{x} \cdot G | \mathbf{x} \in \mathbb{F}_q^k\}.$$

G is referred to as a generator matrix of C. In other words, the code C is the set of all possible linear combinations of rows of G.

If C is an $[n, k]_q$ linear code then there exists a matrix $H \in \mathbb{F}_q^{(n-k) \times n}$ of rank $n-k$ satisfying

$$C = \{\mathbf{y} \in \mathbb{F}_q^n | H \cdot \mathbf{y}^T = \mathbf{0}\}.$$

H is referred to as a parity check matrix of C.

[4] If not, $S_1 \subset S_2$ which implies that that $|S_2| \geq |S_1| + 1$. The latter is not possible if both S_1 and S_2 have the same dimension.

Note that we require G and H to have full row rank (i.e., the rows of G are linearly independent and the same holds for H). Sometimes we will consider matrices $M \in \mathbb{F}_q^{m \times n}$ that are not of full row rank. These can still be used to generate a code $C = \{\{\mathbf{x} \cdot G | \mathbf{x} \in \mathbb{F}_q^m\}$ though the code C will not be an $[n, m]_q$ code. We will still refer to C as the code generated by M in such a case, though the phrase "generator matrix" will be reserved for full rank matrices.

Note that neither the generator matrix nor the parity check matrix are unique for a given code. However, all generator matrices (and parity check matrices) have the same dimensions, i.e. all are $k \times n$ (and $(n-k) \times n$ respectively) matrices. We give examples of these matrices for the case of the $[7, 4, 3]_2$ Hamming code below.

- The $[7, 4, 3]_2$ Hamming code has the following generator matrix:

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

- The following matrix is a parity check matrix of the $[7, 4, 3]_2$ Hamming code:

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

  Indeed, it can be easily verified that $G \cdot H^T = \mathbf{0}$. Then Lemma 2.2.8 proves that H is a parity check matrix of the $[7, 4, 3]_2$ Hamming code.

We now look at some consequences of the above characterizations of an $[n, k]_q$ linear code C. We started this chapter with a quest for succinct representation of a code. Note that both the generator matrix and the parity check matrix can be represented using $O(n^2)$ symbols from $\mathbb{F}_q$. Note that this is much smaller than the exponential representation of a general code. More precisely we have the following result on succinct representations of a linear code (see also Exercise 2.11):

Proposition 2.3.3. *Any $[n, k]_q$ linear code can be represented with $\min(nk, n(n-k))$ symbols from $\mathbb{F}_q$.*

There is an encoding algorithm for C that runs in $O(n^2)$ (in particular $O(kn)$) time– given a message $\mathbf{m} \in \mathbb{F}_q^k$, the corresponding codeword $C(\mathbf{m}) = \mathbf{m} \cdot G$, where G is the generator matrix of C. (See Exercise 2.12.)

Proposition 2.3.4. *For any $[n, k]_q$ linear code, given its generator matrix, encoding can be done with $O(nk)$ operations over $\mathbb{F}_q$.*

There is an error-detecting algorithm for C that runs in $O(n^2)$. This is a big improvement over the naive brute force exponential time algorithm (that goes through all possible codewords $\mathbf{c} \in C$ and checks if $\mathbf{y} = \mathbf{c}$). (See Exercise 2.13.)

Proposition 2.3.5. *For any $[n, k]_q$ linear code, given its parity check matrix, error detection can be performed in $O(n(n-k))$ operations over $\mathbb{F}_q$.*

Next, we look at some alternate characterizations of the distance of a linear code.

2.3.1 On the Distance of a Linear Code

Linear codes admit a nice characterization of minimum distance in terms of the Hamming weight of non-zero codewords, which we have seen for the special case of binary linear codes (Proposition 1.5.4). Recall that we use $\mathrm{wt}(x)$ to denote the Hamming weight of a vector $x \in \Sigma^n$, i.e., the number of non-zero coordinates in x.

Proposition 2.3.6. *For every $[n, k, d]_q$ code C, we have*

$$d = \min_{\substack{\mathbf{c} \in C, \\ \mathbf{c} \neq \mathbf{0}}} \mathrm{wt}(\mathbf{c}).$$

Proof. To show that d is the same as the minimum weight we show that d is no more than the minimum weight and d is no less than the minimum weight.

First, we show that d is no more than the minimum weight. We can see this by considering $\Delta(\mathbf{0}, \mathbf{c}')$ where $\mathbf{c}'$ is the non-zero codeword in C with minimum weight; its distance from $\mathbf{0}$ is equal to its weight. Thus, we have $d \leq wt(\mathbf{c}')$, as desired.

Now, to show that d is no less than the minimum weight, consider $\mathbf{c_1} \neq \mathbf{c_2} \in C$ such that $\Delta(\mathbf{c}_1, \mathbf{c}_2) = d$. Note that $\mathbf{c}_1 - \mathbf{c_2} \in C$ (this is because $-\mathbf{c_2} = -1 \cdot \mathbf{c}_2 \in C$, where -1 is the additive inverse of 1 in $\mathbb{F}_q$ and $\mathbf{c}_1 - \mathbf{c}_2 = \mathbf{c}_1 + (-\mathbf{c}_2)$, which is in C by the definition of linear codes). Now note that $wt(\mathbf{c_1} - \mathbf{c_2}) = \Delta(\mathbf{c_1}, \mathbf{c_2}) = d$, since the non-zero symbols in $\mathbf{c}_1 - \mathbf{c}_2$ occur exactly in the positions where the two codewords differ. Further, since $\mathbf{c}_1 \neq \mathbf{c}_2$, $\mathbf{c}_1 - \mathbf{c}_2 \neq \mathbf{0}$, which implies that the minimum Hamming weight of any non-zero codeword in C is at most d. □

Next, we look at another property implied by the parity check matrix of a linear code.

Proposition 2.3.7. *For every $[n, k, d]_q$ code C with parity check matrix H, d equals the size of the smallest subset of columns of H that are linearly dependent.*

Proof. By Proposition 2.3.6, we need to show that the minimum weight of a non-zero codeword in C is the minimum number of linearly dependent columns. Let t be the minimum number of linearly dependent columns in H. To prove the claim we will show that $t \leq d$ and $t \geq d$.

For the first direction, Let $\mathbf{c} \neq \mathbf{0} \in C$ be a codeword with $wt(\mathbf{c}) = d$. Now note that, by the definition of the parity check matrix, $H \cdot \mathbf{c}^T = \mathbf{0}$. Working through the matrix multiplication, this gives us that $\sum_{i=1}^{n} c_i H^i = \mathbf{0}$, where

$$H = \begin{pmatrix} \uparrow & \uparrow & & \uparrow & & \uparrow \\ H^1 & H^2 & \cdots & H^i & \cdots & H^n \\ \downarrow & \downarrow & & \downarrow & & \downarrow \end{pmatrix}$$

and $\mathbf{c} = (c_1, \ldots, c_n)$. Note that we can skip multiplication for those columns for which the corresponding bit c_i is zero, so for $H \cdot \mathbf{c}^T$ to be zero, those H^i with $c_i \neq 0$ are linearly dependent. This means that $d \geq t$, as the columns corresponding to non-zero entries in $\mathbf{c}$ are one instance of linearly dependent columns.

For the other direction, consider the minimum set of columns from H, $H^{i_1}, H^{i_2}, \ldots, H^{i_t}$ that are linearly dependent. This implies that there exists non-zero elements $c'_{i_1}, \ldots, c'_{i_t} \in \mathbb{F}_q$ such that $c'_{i_i} H^{i_1} + \ldots + c'_{i_t} H^{i_t} = \mathbf{0}$. (Note that all the c'_{i_j} are non-zero as no set of less than t columns are linearly dependent.) Now extend $c'_{i_1}, \ldots, c'_{i_t}$ to the vector $\mathbf{c}'$ such that $c'_j = 0$ for $j \notin \{i_1, \ldots, i_t\}$. Note that we have $H \cdot (\mathbf{c}')^T = \mathbf{0}$ and thus, we have $\mathbf{c}' \in C$. This in turn implies that $d \leq wt(\mathbf{c}') = t$ (where recall t is the minimum number of linearly independent columns in H). □

2.4 Hamming Codes

We now change gears and look at the general family of linear codes, which were discovered by Hamming. So far, we have seen the $[7, 4, 3]_2$ Hamming code (in Section 1.5). In fact, for any $r \geq 2$ there is a $[2^r - 1, 2^r - r - 1, 3]_2$ Hamming code. Thus in Section 1.5, we have seen this code for $r = 3$.

Definition 2.4.1 (Binary Hamming Codes)**.** *For any positive integer r, define the matrix $\mathbf{H}_r \in \mathbb{F}_2^{r \times (2^r - 1)}$ to be the $r \times (2^r - 1)$ matrix whose ith column $\mathbf{H}_r^i$ is the binary representation of i, for $1 \leq i \leq 2^r - 1$. (Note that such a representation is a vector in $\{0, 1\}^r$.)*

The $[2^r - 1, 2^r - r - 1]_2$ Hamming code, denoted by $C_{H,r}$, is the code with parity check matrix $\mathbf{H}_r$.

In other words, the general $[2^r - 1, 2^r - r - 1]_2$ Hamming code is the code

$$\{\mathbf{c} \in \{0, 1\}^{2^r - 1} | \mathbf{H}_r \cdot \mathbf{c}^T = \mathbf{0}\}.$$

For example, for the case we have seen ($r = 3$),

$$\mathbf{H}_3 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

and the resulting code was a $[7, 4, 3]_2$ code.

Next we argue that the above Hamming code has distance 3 (in Proposition 1.5.2, we argued this for $r = 3$).

Proposition 2.4.2. *The Hamming code* $[2^r - 1, 2^r - r - 1, 3]_2$ *has distance 3.*

Proof. No two columns in $\mathbf{H}_r$ are linearly dependent. If they were, we would have $\mathbf{H}_r^i + \mathbf{H}_r^j = \mathbf{0}$, but this is impossible since they differ in at least one bit (being binary representations of integers, $i \neq j$). Thus, by Proposition 2.3.7, the distance is at least 3. It is at most 3, since (e.g.) $\mathbf{H}_r^1 + \mathbf{H}_r^2 + \mathbf{H}_r^3 = \mathbf{0}$. □

Now note that under the Hamming bound for $d = 3$ (Theorem 1.6.2), $k \leq n - \log_2(n+1)$, so for $n = 2^r - 1$, $k \leq 2^r - r - 1$. Hence, the Hamming code is a perfect code. (See Definition 1.7.3.)

In Question 1.7.4, we asked which codes are perfect codes. Interestingly, the only perfect binary codes are the following:

- The Hamming codes which we just studied.
- The trivial $[n, 1, n]_2$ codes for odd n (which have 0^n and 1^n as the only codewords): see Exercise 2.24.
- Two codes due to Golay [17].

2.5 Efficient Decoding of Hamming codes

We have shown that the Hamming code has a distance of 3 and thus, by Proposition 1.4.2, can correct one error. However, this is a *combinatorial* result and does not give us an efficient algorithm. One obvious candidate for decoding is the MLD function (Algorithm 1.4.1). Unfortunately, the only implementation of MLD that we know is the one in Algorithm 1.4.1, which will take time $2^{\Theta(n)}$, where n is the block length of the Hamming code.

However, we can do much better. Consider the following simple algorithm: given the received word $\mathbf{y}$, first check if it is indeed a valid codeword. If it is, we are done. Otherwise, flip each of the n bits and check if the resulting vector is a valid codeword. If so, we have successfully decoded from one error. If none of the checks are successful, then we declare a decoding failure. Algorithm 2.5.1 formally presents this algorithm (where $C_{H,r}$ is the $[2^r - 1, 2^r - r - 1, 3]_2$ Hamming code).[5]

It can be verified that Algorithm 2.5.1 can correct up to 1 error. If each of the checks $\mathbf{y}' \in C_{H,r}$ can be done in $T(n)$ time, then the time complexity of the proposed algorithm will be $O(nT(n))$. Note that since $C_{H,r}$ is a linear code (and dimension $k = n - O(\log n)$) by Proposition 2.3.5, we have $T(n) = O(n \log n)$. Thus, the proposed algorithm has running time $O(n^2 \log n)$.

Note that Algorithm 2.5.1 can be generalized to work for any linear code C with distance $2t + 1$ (and hence, can correct up to t errors): go through all possible error vectors $\mathbf{z} \in [q]^n$

[5] Formally speaking, a decoding algorithm should return the transmitted message $\mathbf{x}$ but Algorithm 2.5.1 actually returns $C_{H,r}(\mathbf{x})$. However, since $C_{H,r}$ is a linear code, it is not too hard to see that one can obtain $\mathbf{x}$ from $C_{H,r}(\mathbf{x})$ in $O(n^3)$ time: see Exercise 2.25. Further, for $C_{H,r}$ one can do this in $O(n)$ time: see Exercise 2.26.

Algorithm 2.5.1 Naive Decoder for Hamming Code

INPUT: Received word $\mathbf{y}$
OUTPUT: $\mathbf{c}$ if $\Delta(\mathbf{y}, \mathbf{c}) \leq 1$ else `Fail`

1: IF $\mathbf{y} \in C_{H,r}$ THEN
2: RETURN $\mathbf{y}$
3: FOR $i = 1 \dots n$ DO
4: $\mathbf{y}' \leftarrow \mathbf{y} + \mathbf{e}_i$ ▷ $\mathbf{e}_i$ is the ith standard basis vector
5: IF $\mathbf{y}' \in C_{H,r}$ THEN
6: RETURN $\mathbf{y}'$
7: RETURN `Fail`

(with $wt(\mathbf{z}) \leq t$) and check if $\mathbf{y} - \mathbf{z}$ is in the code or not. Algorithm 2.5.2 presents the formal algorithm (where C is an $[n, k, 2t + 1]_q$ code).

Algorithm 2.5.2 Decoder for Any Linear Code

INPUT: Received word $\mathbf{y}$
OUTPUT: $\mathbf{c} \in C$ if $\Delta(\mathbf{y}, \mathbf{c}) \leq t$ else `Fail`

1: FOR $i = 0 \dots t$ DO
2: FOR $S \subseteq [n]$ such that $|S| = i$ DO
3: FOR $\mathbf{z} \in \mathbb{F}_q^n$ such that $wt(\mathbf{z}_S) = wt(\mathbf{z}) = i$ DO
4: IF $\mathbf{y} - \mathbf{z} \in C$ THEN
5: RETURN $\mathbf{y} - \mathbf{z}$
6: RETURN `Fail`

The number of error patterns $\mathbf{z}$ considered by Algorithm 2.5.2 is[6] $\sum_{i=0}^{t} \binom{n}{i}(q-1)^i \leq O((nq)^t)$. Furthermore by Proposition 2.3.5, Step 4 can be performed with $O(n^2)$ operations over $\mathbb{F}_q$. Thus, Algorithm 2.5.2 runs with $O(n^{t+2}q^t)$ operations over $\mathbb{F}_q$, which for q being a small polynomial in n, is $n^{O(t)}$ operations. In other words, the algorithm will have polynomial running time for codes with a constant distance (though the running time would not be practical even for moderate values of t).

However, it turns out that for Hamming codes there exists a decoding algorithm with an $O(n^2)$ running time. To see this, first note that if the received word $\mathbf{y}$ has no errors, then $\mathbf{H}_r \cdot \mathbf{y}^T = \mathbf{0}$. If not, then $\mathbf{y} = \mathbf{c} + \mathbf{e}_i$, where $\mathbf{c} \in C$ and $\mathbf{e}_i$ is the unit vector with the only nonzero element at the i-th position. Thus, if $\mathbf{H}_r^i$ stands for the i-th column of $\mathbf{H}_r$,

$$\mathbf{H}_r \cdot \mathbf{y}^T = \mathbf{H}_r \cdot \mathbf{c}^T + \mathbf{H}_r \cdot (\mathbf{e}_i)^T = \mathbf{H}_r \cdot (\mathbf{e}_i)^T = \mathbf{H}_r^i,$$

where the second equality follows as $\mathbf{H}_r \cdot \mathbf{c}^T = 0$, which in turn follows from the fact that $\mathbf{c} \in C$. In other words, $\mathbf{H}_r \cdot \mathbf{y}^T$ gives the *location* of the error. This leads to Algorithm 2.5.3.

Note that $\mathbf{H}_r$ is an $r \times n$ matrix where $n = 2^r - 1$ and thus, $r = \Theta(\log n)$. This implies Step 1 in Algorithm 2.5.3, which is a matrix vector multiplication can be done in time

[6] Recall (1.18).

Algorithm 2.5.3 Efficient Decoder for Hamming Code

INPUT: Received word $\mathbf{y}$

OUTPUT: $\mathbf{c}$ if $\Delta(\mathbf{y}, \mathbf{c}) \leq 1$ else `Fail`

1: $\mathbf{b} \leftarrow H_r \cdot \mathbf{y}^T$.
2: Let $i \in [n]$ be the number whose binary representation is $\mathbf{b}$
3: IF $\mathbf{y} - \mathbf{e}_i \in C_H$ THEN
4: RETURN $\mathbf{y} - \mathbf{e}_i$
5: RETURN `Fail`

$O(n \log n)$. By a similar argument and by Proposition 2.3.5 Step 3 can be performed in $O(n \log n)$ time, and therefore Algorithm 2.5.3 overall runs in $O(n \log n)$ time. Thus,

Theorem 2.5.1. *The $[n = 2^r - 1, 2^r - r - 1, 3]_2$ Hamming code is 1-error correctable. Furthermore, decoding can be performed in time $O(n \log n)$.*

2.6 Dual of a Linear Code

Until now, we have thought of parity check matrix as defining a code via its null space. However, we are not beholden to think of the parity check matrix in this way. A natural alternative is to use the parity check matrix as a generator matrix. The following definition addresses this question.

Definition 2.6.1 (Dual of a code)**.** *Let H be a parity check matrix of a code C, then the code generated by H is called the dual of C. The dual of a code C is denoted by $C^\perp$.*

It is obvious from the definition that if C is an $[n, k]_q$ code, then $C^\perp$ is an $[n, n-k]_q$ code. Applying duality to the Hamming codes and a close relative, we get two families of codes described below.

Definition 2.6.2 (Simplex and Hadamard Codes)**.** *For positive integer r the* Simplex Code *$C_{Sim,r}$ is the code generated by H_r. (Equivalently $C_{Sim,r} = C_{H,r}^\perp$.) For positive integer r the* Hadamard Code *$C_{Had,r}$ is the $[2^r, r]_2$ code generated by the $r \times 2^r$ matrix H'_r obtained by adding the all zero column to (say in front of columns in) H_r.*

We claim that $C_{Sim,r}$ and $C_{Had,r}$ are $[2^r - 1, r, 2^{r-1}]_2$ and $[2^r, r, 2^{r-1}]_2$ codes respectively. The claimed block length and dimension follow from the definition of the codes, while the distance follows from the following result.

Proposition 2.6.3. *$C_{Sim,r}$ and $C_{Had,r}$ both have distances of 2^{r-1}.*

Proof. We first show the result for $C_{Had,r}$. In fact, we will show something stronger: every non-zero codeword in $C_{Had,r}$ has weight exactly equal to 2^{r-1} (the claimed distance

follows from Proposition 2.3.6). Consider a message $\mathbf{x} \neq 0$. Let its ith entry be $x_i = 1$. $\mathbf{x}$ is encoded as

$$\mathbf{c} = (x_1, x_2, \ldots, x_r)(H_r^0, H_r^1, \ldots, H_r^{2^r-1}),$$

where H_r^j is the binary representation of $0 \leq j \leq 2^r - 1$ (that is, the set set of vector H_r^j is exactly the set of all the vectors in $\{0,1\}^r$). Further note that the jth bit of the codeword $\mathbf{c}$ is $\langle \mathbf{x}, H_r^j \rangle$. Group all the columns of the generator matrix into pairs $(\mathbf{u}, \mathbf{v})$ such that $\mathbf{v} = \mathbf{u} + \mathbf{e}_i$ (i.e. $\mathbf{v}$ and $\mathbf{u}$ are the same except in the ith position). For example for $r = 3$ and $i = 2$, the paired up columns are marked with the same color below:

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

Notice that this partitions all the columns into 2^{r-1} disjoint pairs. Then,

$$\langle \mathbf{x}, \mathbf{v} \rangle = \langle \mathbf{x}, \mathbf{u} + \mathbf{e}_i \rangle = \langle \mathbf{x}, \mathbf{u} \rangle + \langle \mathbf{x}, \mathbf{e}_i \rangle = \langle \mathbf{x}, \mathbf{u} \rangle + x_i = \langle \mathbf{x}, \mathbf{u} \rangle + 1.$$

Thus we have that $\langle \mathbf{x}, \mathbf{v} \rangle$ is the negation of $\langle \mathbf{x}, \mathbf{u} \rangle$, i.e. exactly one of $\langle \mathbf{x}, \mathbf{v} \rangle$ and $\langle \mathbf{x}, \mathbf{u} \rangle$ is 1. As the choice of the pair $(\mathbf{u}, \mathbf{v})$ was arbitrary, we have proved that for any non-zero codeword $\mathbf{c}$ such that $\mathbf{c} \in C_{Had,r}$, $wt(\mathbf{c}) = 2^{r-1}$.

For the simplex code, we observe that all codewords of $C_{Had,r}$ are obtained by padding a 0 to the beginning of the codewords in $C_{Sim,r}$, which implies that all non-zero codewords in $C_{Sim,r}$ also have a weight of 2^{r-1}, which completes the proof. □

We remark that the family of Hamming code has a rate of 1 and a (relative) distance of 0 while the families of Simplex/Hadamard codes have a rate of 0 and a relative distance of 1/2. Thus neither gives a positive answer to Question 1.8.3 and so the quest for an asymptotically good code remains ongoing for now (and we will get to these in future chapters).

2.7 Exercises

Exercise 2.1. *Let $(S, +, \cdot)$ be a field (as per Definition 2.1.2). Then argue that $a \cdot 0 = 0 \cdot a = 0$ for every $a \in S$.*

Exercise 2.2. *Prove that the set of rationals (i.e. the set of reals of the form $\frac{a}{b}$, where both a and $b \neq 0$ are integers), denoted by $\mathbb{Q}$, is a field.*

Exercise 2.3. *Let q be a prime power. Let $x \in \mathbb{F}_q$ such that $x \notin \{0, 1\}$. Then prove that for any $n \leq q - 1$:*

$$\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1}.$$

Exercise 2.4. *The main aim of this exercise is to prove the following identity that is true for any $\alpha \in \mathbb{F}_q$:*

$$\alpha^q = \alpha \tag{2.4}$$

To make progress towards the above we will prove a sequence of properties of groups. *A group G is a pair $(S, \circ)$ where the operator $\circ : G \times G \to G$ such that $\circ$ is commutative*[7] *and the elements of S are closed under $\circ$. Further, there is a special element $\iota \in S$ that is the identity element and every element $a \in S$ has an inverse element $b \in S$ such that $a \circ b = \iota$. Note that a finite field $\mathbb{F}_q$ consists of an* additive group *with the $+$ operator (and 0 as additive identity) and a* multiplicative *group on the non-zero elements of $\mathbb{F}_q$ (which is also denoted by $\mathbb{F}_q^*$) with the $\cdot$ operator (and 1 as the multiplicative identity).*[8]

For the rest of the problem let $G = (S, \cdot)$ be a multiplicative group with $|G| = m$. Prove the following statements.

1. *For any $\beta \in G$, let $o(\beta)$ be the smallest integer o such that $\beta^o = 1$. Prove that such an $o \leq m$ always exists. Further, argue that $T = \{1, \beta, \ldots, \beta^{o-1}\}$ also forms a group. $(T, \cdot)$ is called a* sub-group *of G and $o(\beta)$ is called the* order *of β.*

2. *For any $g \in G$, define the* coset *(w.r.t. T) as*

   $$gT = \{g \cdot \beta | \beta \in T\}.$$

   Prove that if $h^{-1} \cdot g \in T$ then $gT = hT$ and $gT \cap hT = \emptyset$ otherwise. Further argue that these cosets partition the group G into disjoint sets.

3. *Argue that for any $g \in G$, we have $|gT| = |T|$.*

4. *Using the above results or otherwise, argue that for any $\beta \in G$, we have*

   $$\beta^m = 1.$$

5. *Prove (2.4).*

Exercise 2.5. *Prove that for $q = 2$, the second condition in Definition 2.2.2 is implied by the first condition.*

Exercise 2.6. *Prove that G_2 from (2.3) has full rank.*

Exercise 2.7. *In this problem we will look at the problem of solving a system of linear equations over $\mathbb{F}_q$. That is, one needs to solve for unknowns $x_1, \ldots, x_n$ given the following m linear equations (where $a_{i,j}, b_i \in \mathbb{F}_q$ for $1 \leq i \leq m$ and $1 \leq j \leq n$):*

$$a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1.$$

[7] Technically, G is an *abelian* group.

[8] Recall Definition 2.1.2.

$$a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2.$$

$$\vdots$$

$$a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n = b_m.$$

1. *(Warm-up) Convince yourself that the above problem can be stated as* $A \cdot \mathbf{x}^T = \mathbf{b}^T$*, where* A *is an* $m \times n$ *matrix over* $\mathbb{F}_q$*,* $\mathbf{x} \in \mathbb{F}_q^n$ *and* $\mathbf{b} \in \mathbb{F}_q^m$*.*

2. *(Upper Triangular Matrix) Assume* $n = m$ *and that* A *is upper triangular, i.e. all diagonal elements* ($a_{i,i}$) *are non-zero and all lower triangular elements* ($a_{i,j}$*,* $i > j$) *are* 0*. Then present an* $O(n^2)$ *time*[9] *algorithm to compute the unknown vector* $\mathbf{x}$*.*

3. *(Gaussian Elimination) Assume that* A *has full rank (or equivalently a rank of* n*.)*

   (a) *Prove that the following algorithm due to Gauss converts* A *into an upper triangular matrix. By permuting the columns if necessary make sure that* $a_{1,1} \neq 0$*. (Why can one assume w.l.o.g. that this can be done?) Multiply all rows* $1 < i \leq n$ *with* $\frac{a_{1,1}}{a_{i,1}}$ *and then subtract* $a_{1,j}$ *from the* (i,j)*th entry* $1 \leq j \leq n$*. Recurse with the same algorithm on the* $(n-1) \times (n-1)$ *matrix* A' *obtained by removing the first row and column from* A*. (Stop when* $n = 1$*.)*

   (b) *What happens if* A *does not have full rank? Show how one can modify the algorithm above to either upper triangulate a matrix or report that it does not have full rank. (Convince yourself that your modification works.)*

   (c) *Call a system of equations* $A \cdot \mathbf{x}^T = \mathbf{b}^T$ *consistent if there exists a solution to* $\mathbf{x} \in \mathbb{F}_q^n$*. Show that there exists an* $O(n^3)$ *algorithm that finds the solution if the system of equations is consistent and* A *has full rank (and report "fail" otherwise).*

4. *(*$m < n$ *case) Assume that* A *has full rank, i.e. has a rank of* m*. In this scenario either the system of equations is inconsistent or there are* q^{n-m} *solutions to* $\mathbf{x}$*. Modify the algorithm from above to design an* $O(m^2 n)$ *time algorithm to output the solutions (or report that the system is inconsistent).*

   - *Note that in case the system is consistent there will be* q^{n-m} *solutions, which might be much bigger than* $O(m^2 n)$*. Show that this is not a problem as one can represent the solutions as system of linear equations. (I.e. one can have* $n - m$ *"free" variables and* m *"bound" variables.)*

5. *(*$m > n$ *case) Assume that* A *has full rank, i.e. a rank of* n*. In this scenario either the system of equations is inconsistent or there is a unique solution to* $\mathbf{x}$*. Modify the algorithm from above to design an* $O(m^2 n)$ *time algorithm to output the solution (or report that the system is inconsistent).*

[9] For this problem, any basic operation over $\mathbb{F}_q$ takes unit time.

6. *(Non-full rank case) Give an $O(m^2 n)$ algorithm for the general case, i.e. the $m \times n$ matrix A need not have full rank. (The algorithm should either report that the system of equations is inconsistent or output the solution(s) to $\mathbf{x}$.)*

Exercise 2.8. *Prove that the span of k linearly independent vectors over $\mathbb{F}_q$ has size exactly q^k.*

Exercise 2.9. *Let G and H be a generator and parity check matrix of the same linear code of dimension k and block length n. Then $G \cdot H^T = \mathbf{0}$.*

Exercise 2.10. *Let C be an $[n,k]_q$ linear code with a generator matrix with no all zeros columns. Then for every position $i \in [n]$ and $\alpha \in \mathbb{F}_q$, the number of codewords $\mathbf{c} \in C$ such that $c_i = \alpha$ is exactly q^{k-1}.*

Exercise 2.11. *Prove Proposition 2.3.3.*

Exercise 2.12. *Prove Proposition 2.3.4.*

Exercise 2.13. *Prove Proposition 2.3.5.*

Exercise 2.14. *A set of vector $S \subseteq \mathbb{F}_q^n$ is called t-wise independent if for every set of positions I with $|I| = t$, the set S projected to I has each of the vectors in $\mathbb{F}_q^t$ appear the same number of times. (In other words, for every choice of $I \subseteq [n]$ with $|I| = t$, if one picks a vector $(X_1, \ldots, X_n)$ uniformly at random from S then the variables $\{X_i | i \in I\}$ are distributed uniformly and independently random over $\mathbb{F}_q$).*

Prove that any linear code C whose dual $C^\perp$ has distance $d^\perp$ is $(d^\perp - 1)$-wise independent.

Exercise 2.15. *A set of vectors $S \subseteq \mathbb{F}_2^k$ is called ε-biased sample space if the following property holds. Pick a vector $X = (x_1, \ldots, x_k)$ uniformly at random from S. Then X has* bias *at most ε, that is, for every $I \subseteq [k]$,*

$$\left| \Pr\left(\sum_{i \in I} x_i = 0 \right) - \Pr\left(\sum_{i \in I} x_i = 1 \right) \right| \leq \varepsilon.$$

We will look at some connections of such sets to codes.

1. *Let C be an $[n,k]_2$ code such that all non-zero codewords have Hamming weight in the range $\left[\left(\frac{1-\varepsilon}{2}\right) n, \left(\frac{1+\varepsilon}{2}\right) n\right]$. Then there exists an ε-biased space of size n in $\mathbb{F}_2^k$.*

2. *Let C be an $[n,k]_2$ code such that all non-zero codewords have Hamming weight in the range $\left[\left(\frac{1}{2} - \gamma\right) n, \left(\frac{1}{2} + \gamma\right) n\right]$ for some constant $0 < \gamma < 1/2$. Then there exists an ε-biased space in $\mathbb{F}_2^k$ of size $n^{O(\gamma^{-1} \cdot \log(1/\varepsilon))}$.*

Exercise 2.16. *Let C be an $[n, k, d]_q$ code. Let $\mathbf{y} = (y_1, \ldots, y_n) \in (\mathbb{F}_q \cup \{?\})^n$ be a received word*[10] *such that $y_i =?$ for at most $d-1$ values of i. Present an $O(n^3)$ time algorithm that outputs a codeword $\mathbf{c} = (c_1, \ldots, c_n) \in C$ that agrees with y in all un-erased positions (i.e., $c_i = y_i$ if $y_i \neq ?$) or states that no such $\mathbf{c}$ exists. (Recall that if such a $\mathbf{c}$ exists then it is unique.)*

Exercise 2.17. *In the chapter, we did not talk about how to obtain the parity check matrix of a linear code from its generator matrix. In this problem, we will look at this "conversion" procedure.*

(a) Prove that any generator matrix $\mathbf{G}$ of an $[n, k]_q$ code C (recall that $\mathbf{G}$ is a $k \times n$ matrix) can be converted into another equivalent generator matrix of the form $\mathbf{G}' = [\mathbf{I}_k | \mathbf{A}]$, where $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{A}$ is some $k \times (n-k)$ matrix. By "equivalent," we mean that the code generated by $\mathbf{G}'$ has a linear bijective map to C.

Note that the code generated by $\mathbf{G}'$ has the message symbols as its first k symbols in the corresponding codeword. Such codes are called systematic codes. In other words, every linear code can be converted into a systematic code. Systematic codes are popular in practice as they allow for immediate access to the message symbols.

(b) Given an $k \times n$ generator matrix of the form $[\mathbf{I}_k | \mathbf{A}]$, give a corresponding $(n-k) \times n$ parity check matrix. Briefly justify why your construction of the parity check matrix is correct.

Hint: *Try to think of a parity check matrix that can be decomposed into two submatrices: one will be closely related to $\mathbf{A}$ and the other will be an identity matrix, though the latter might not be a $k \times k$ matrix).*

(c) Use part (b) to present a generator matrix for the $[2^r - 1, 2^r - r - 1, 3]_2$ Hamming code.

Exercise 2.18. *So far in this book we have seen that one can modify one code to get another code with interesting properties (for example, the construction of the Hadamard code from the Simplex code from Section 2.6 and Exercise 1.7). In this problem you will need to come up with more ways of constructing new codes from existing ones.*

Prove the following statements (recall that the notation $(n, k, d)_q$ code is used for general codes with q^k codewords where k need not be an integer, whereas the notation $[n, k, d]_q$ code stands for a linear code *of dimension k):*

1. *If there exists an $(n, k, d)_{2^m}$ code, then there also exists an $(nm, km, d' \geq d)_2$ code.*
2. *If there exists an $[n, k, d]_{2^m}$ code, then there also exists an $[nm, km, d' \geq d]_2$ code.*
3. *If there exists an $[n, k, d]_q$ code, then there also exists an $[n-d, k-1, d' \geq \lceil d/q \rceil]_q$ code.*

[10] A ? denotes an erasure.

4. *If there exists an $[n, k, \delta n]_q$ code, then for every $m \geq 1$, there also exists an $(n^m, k/m, (1-(1-\delta)^m) \cdot n^m)_{q^m}$ code.*

5. *If there exists an $[n, k, \delta n]_2$ code, then for every* odd *$m \geq 1$, there also exists an $\left[n^m, k, \frac{1}{2} \cdot (1-(1-2\delta)^m) \cdot n^m\right]_2$ code.*

Note: *In all the parts, the only things that you can assume about the original code are only the parameters given by its definition– nothing else!*

Exercise 2.19. *Let C_1 be an $[n, k_1, d_1]_q$ code and C_2 be an $[n, k_2, d_2]_q$ code. Then define a new code as follows:*

$$C_1 \ominus C_2 = \{(\mathbf{c}_1, \mathbf{c}_1 + \mathbf{c}_2) | \mathbf{c}_1 \in C_1, \mathbf{c}_2 \in C_2\}.$$

Next we will prove interesting properties of this operations on codes:

1. *If G_i is the generator matrix for C_i for $i \in [2]$, what is a generator matrix for $C_1 \ominus C_2$?*

2. *Argue that $C_1 \ominus C_2$ is an $[2n, k_1 + k_2, d \stackrel{\text{def}}{=} \min(2d_1, d_2)]_q$ code.*

3. *Assume there exists algorithms $\mathcal{A}_i$ for code C_i for $i \in [2]$ such that: (i) $\mathcal{A}_1$ can decode from e errors and s erasures such that $2e + s < d_1$ and (ii) $\mathcal{A}_2$ can decode from $\lfloor (d_2 - 1)/2 \rfloor$ errors. Then argue that one can correct $\lfloor (d-1)/2 \rfloor$ errors for $C_1 \ominus C_2$.*
   Hint: *Given a received word $(\mathbf{y}_1, \mathbf{y}_2) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$, first apply $\mathcal{A}_2$ on $\mathbf{y}_2 - \mathbf{y}_1$. Then create an intermediate received word for $\mathcal{A}_1$.*

4. *We will now consider a recursive construction of a binary linear code that uses the $\ominus$ operator. For integers $0 \leq r \leq m$, we define the code $C(r, m)$ as follows:*

   - *$C(r, r) = \mathbb{F}_2^r$ and $C(0, r)$ is the code with only two codewords: the all ones and all zeroes vector in $\mathbb{F}_2^r$.*
   - *For $1 < r < m$, $C(r, m) = C(r, m-1) \ominus C(r-1, m-1)$.*

   Determine the parameters of the code $C(r, m)$.

Exercise 2.20. *Let C_1 be an $[n_1, k_1, d_1]_2$ binary linear code, and C_2 an $[n_2, k_2, d_2]$ binary linear code. Let $C \subseteq \mathbb{F}_2^{n_1 \times n_2}$ be the subset of $n_2 \times n_1$ matrices whose rows belong to C_1 and whose columns belong to C_2. C is called the tensor of C_1 and C_2 and is denoted by $C_1 \otimes C_2$.*

Prove that C is an $[n_1 n_2, k_1 k_2, d_1 d_2]_2$ binary linear code.

Further, if $\mathbf{G}_1$ and $\mathbf{G}_2$ are generator matrices of C_1 and C_2, construct a genertor matrix of $C_1 \otimes C_2$ from $\mathbf{G}_1$ and $\mathbf{G}_2$. In particular, argue that given $\mathbf{G}_1$ and $\mathbf{G}_2$, a generator matrix of $C_1 \otimes C_2$ can be computed in polynomimal time.

Hint: *For the latter problem, it might be useful to think of the codewords and messages as vectors instead of matrices.*

Exercise 2.21. *In Section 2.4 we considered the* binary *Hamming code. In this problem we will consider the more general q-ary Hamming code. In particular, let q be a prime power and $r \geq 1$ be an integer. Define the following $r \times n$ matrix $H_{q,r}$, where each column is an non-zero vector from $\mathbb{F}_q^r$ such that the first non-zero entry is 1. For example,*

$$H_{3,2} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix}$$

In this problem we will derive the parameters of the code. Define the generalized Hamming code $C_{H,r,q}$ to be the linear code whose parity check matrix is $H_{q,r}$. Argue that

1. *The block length of $C_{H,r,q}$ is $n = \frac{q^r-1}{q-1}$.*
2. *$C_{H,q,r}$ has dimension $n - r$.*
3. *$C_{H,q,r}$ has distance 3.*

Exercise 2.22. *In Section 2.6, we considered the binary Hadamard code. In this problem we will consider the more general q-ary Hadamard code. In particular, let q be a prime power and $r \geq 1$ be an integer. Define the following $r \times q^r$ matrix $\overline{H}_{q,r}$, where each columns in a vector in $\mathbb{F}_q^r$. In this problem we will derive the parameters of the code. Define the generalized Hadamard code $C_{Had,r,q}$ to be the linear code whose parity check matrix is $\overline{H}_{q,r}$. Argue that*

1. *The block length of $C_{Had,r,q}$ is $n = q^r$.*
2. *$C_{Had,q,r}$ has dimension r.*
3. *$C_{Had,q,r}$ has distance $\left(1 - \frac{1}{q}\right) \cdot n$.*

Exercise 2.23. *Design the best 6-ary code (family) with distance 3 that you can.*

Hint: *Start with a 7-ary Hamming code.*

Exercise 2.24. *Prove that the $[n, 1, n]_2$ code for odd n (i.e. the code with the all zeros and all ones vector as it only two codewords) attains the Hamming bound (Theorem 1.7.2).*

Exercise 2.25. *Let C be an $[n, k]_q$ code with generator matrix G. Then given a codeword $\mathbf{c} \in C$ one can compute the corresponding message in time $O(kn^2)$.*

Exercise 2.26. *Given a $\mathbf{c} \in C_{H,r}$, one can compute the corresponding message in time $O(n)$.*

Exercise 2.27. *Let C be an $(n, k)_q$ code. Prove that if C can be decoded from e errors in time $T(n)$, then it can be decoded from $n + c$ errors in time $O((nq)^c \cdot T(n))$.*

Exercise 2.28. *Show that the bound of kd of the number of ones in the generator matrix of any binary linear code (see Exercise 1.14) cannot be improved for every code.*

Exercise 2.29. *Let C be a linear code. Then prove that $\left(C^{\perp}\right)^{\perp} = C$.*

Exercise 2.30. *Note that for any linear code C, the codewords $\mathbf{0}$ is in both C and $C^{\perp}$. Show that there exists a linear code C such that it shares a non-zero codeword with $C^{\perp}$.*

Exercise 2.31. *We go into a bit of diversion and look at how finite fields are different from infinite fields (e.g. $\mathbb{R}$). Most of the properties of linear subspaces that we have used for linear codes (e.g. notion of dimension, the existence of generator and parity check matrices, notion of duals) also hold for linear subspaces over $\mathbb{R}$.*[11] *One trivial property that holds for linear subspaces over finite fields that does not hold over $\mathbb{R}$ is that linear subspaces over $\mathbb{F}_q$ with dimension k has size q^k (though this is a trivial consequence that $\mathbb{F}_q$ is a finite field while $\mathbb{R}$ is an infinite field). Next, we consider a more subtle distinction.*

Let $S \subseteq \mathbb{R}^n$ be a linear subspace over $\mathbb{R}$ and let $S^{\perp}$ is the dual of S. Then show that

$$S \cap S^{\perp} = \{\mathbf{0}\}.$$

By contrast, linear subspaces over finite fields can have non-trivial intersection with their duals (see e.g. Exercise 2.30).

Exercise 2.32. *A linear code C is called* self-orthogonal *if $C \subseteq C^{\perp}$. Show that*

1. *The binary repetition code with even number of repetitions is self-orthogonal.*

2. *The Hadamard code $C_{Had,r}$ is self-orthogonal.*

Exercise 2.33. *A linear code C is called self dual if $C = C^{\perp}$. Show that for*

1. *Any self dual code has dimension $n/2$.*

2. *Prove that the following code is self-dual*

$$\{(\mathbf{x}, \mathbf{x}) | \mathbf{x} \in \mathbb{F}_2^k\}.$$

Exercise 2.34. *Given a code C a* puncturing *of C is another code C' where the same set of positions are dropped in all codewords of C. More precisely, if $C \subseteq \Sigma^n$ and the set of punctured positions is $P \subseteq [n]$, then the punctured code is $\{(c_i)_{i \notin P} | (c_1, \ldots, c_n) \in C\}$.*

Prove that a linear code with no repetitions (i.e. there are no two positions $i \neq j$ such that for every codeword $\mathbf{c} \in C$, $c_i = c_i$) is a puncturing of the Hadamard code. Hence, Hadamard code is the "longest" linear code that does not repeat.

[11] A linear subspace $S \subseteq \mathbb{R}^n$ is the same as in Definition 2.2.2 where all occurrences of the finite field $\mathbb{F}_q$ is replaced by $\mathbb{R}$.

Exercise 2.35. *In this problem we will consider the* long code. *For the definition, we will use the functional way of looking at the ambient space as mentioned in Remark 1.2.2. A long code of dimension k is a binary code such that the codeword corresponding to $\mathbf{x} = \mathbb{F}_2^k$, is the function $f : \{0,1\}^{2^k} \to \{0,1\}$ defined as follows. For any $\mathbf{m} \in \{0,1\}^{\mathbb{F}_2^k}$, we have $f((m_\alpha)_{\alpha \in \mathbb{F}_2^k}) = m_{\mathbf{x}}$. Derive the parameters of the long code.*

Finally, argue that the long code is the code with the longest block length such that the codewords do not have a repeated coordinate (i.e. there does not exists $i \neq j$ such that for every codeword $\mathbf{c}$, $c_i = c_j$). (Contrast this with the property of Hadamard code above.)

Exercise 2.36. *Given a linear code $C \subseteq \mathbb{F}_2^n$, define its generating function to be a $2n$-variate polynomial over variables $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ given by $G_C(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{w} \in C} P_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ where $P_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \left(\prod_{\{i \in [n] | w_i = 0\}} x_i\right) \cdot \left(\prod_{\{i \in [n] | w_i = 1\}} y_i\right)$. For $w \in \{0, \ldots, n\}$, let A_C^w denote the number of codewords of weight w and let $A_C(z) = \sum_{w=0}^n A_C^w z^w$ be the "weight enumerator" polynomial of C.*

1. *For every $\mathbf{w} \in \mathbb{F}_2^n$, prove that $P_{\mathbf{w}}(\mathbf{x} + \mathbf{y}, \mathbf{x} - \mathbf{y}) = \sum_{\mathbf{v} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{v}, \mathbf{w} \rangle} P_{\mathbf{v}}(\mathbf{x}, \mathbf{y})$.*
2. *Prove that $G_{C^\perp}(\mathbf{x}, \mathbf{y}) = \frac{1}{|C|} G_C(\mathbf{x} + \mathbf{y}, \mathbf{x} - \mathbf{y})$.*
3. *Prove that $A_C(z) = G_C(1, \ldots, 1, z, \ldots, z)$.*
4. *Prove that $A_{C^\perp}(z) = \frac{(1+z)^n}{|C|} A_C\left(\frac{1-z}{1+z}\right)$.*
5. *Conclude that $A_{C^\perp}^w = \frac{1}{|C|} \sum_{u=0}^n A_C^u \left(\sum_{i=0}^u (-1)^i \binom{u}{i} \binom{n-u}{w-i}\right)$. In other words, the distributions of the weights $(A_C^0, \ldots, A_C^n)$ of the primal code completely determine the distributions of weights $(A_{C^\perp}^0, \ldots, A_{C^\perp}^n)$ of the dual code!*

2.8 Bibliographic Notes

The background material on algebra is essentially folklore. Readers interested in a more extensive treatment are referred to classical texts such as by Artin [2]. For a perspective focussing more on finite fields, see the text by Lidl and Niederreiter [28]. Linear codes arose already in the paper of Hamming [21] and were systematically studied by Slepian [39]. The answer to Question 1.7.4 was given by van Lint [41] and Tietavainen [40]. Hadamard codes (Definition 2.6.2) are named after the work of mathematician Jacques Hadamard and in particular the notion of Hadamard matrices which are self-orthogonal matrices with +1/-1 entries.

Exercises 2.14 and 2.15 come from the theory of pseudorandomness, which we will cover more extensively in Chapter **??**. The long codes in Exercise 2.35 were introduced by Bellare, Goldreich and Sudan [4]. Exercise 2.36 is based on the MacWilliams Identity proved by MacWilliams [29].

Chapter 3

Probability as Fancy Counting and the q-ary Entropy Function

In the chapters to come we will explore questions of the form: "Given n, k, d and q does an $(n, k, d)_q$ code exist?" To answer such questions, we will apply the "probabilistic method" — the method that demonstrates the existence of an object with a given property by showing that a randomly chosen object has the property with positive probability. To elaborate on this sentence, we need to introduce the basic language and tools of probability theory which we do in Section 3.1.

We then introduce the probabilistic method in Section 3.2. We even apply the method to answer a very simple question:

Question 3.0.1. *Does there exist a* $[2, 2, 1]_2$ *code?*

We note that the answer to the above question is trivially yes: just pick the generator matrix to be the 2×2 identity matrix. But our proof will have the advantage of generalizing to broader settings, though we save the generalizations for later chapters.

Finally in Section 3.3 we introduce the "entropy function" which turns out to be central in the understanding of limits of codes (both existence and non-existence).

3.1 A Crash Course on Probability

In this section we review basic concepts in probability theory, specialized to the needs of this book. Specifically,, we introduce distributions, events and random variables, and give some tools to analyze them.

In this book, we will only consider probability distributions defined over finite spaces. In particular, given a finite domain $\mathbb{D}$, a probability *distribution* is defined as a function

$$p : \mathbb{D} \to [0, 1] \text{ such that } \sum_{x \in \mathbb{D}} p(x) = 1,$$

where $[0, 1]$ is shorthand for the interval of all real numbers between 0 and 1.

G	$\mathcal{U}(G)$	V_{00}	V_{01}	V_{10}	V_{11}
$M_{0,0,0,0}$	$\frac{1}{16}$	0	0	0	0
$M_{0,0,0,1}$	$\frac{1}{16}$	0	1	0	1
$M_{0,0,1,0}$	$\frac{1}{16}$	0	1	0	1
$M_{0,0,1,1}$	$\frac{1}{16}$	0	2	0	2
$M_{0,1,0,0}$	$\frac{1}{16}$	0	0	1	1
$M_{0,1,0,1}$	$\frac{1}{16}$	0	1	1	0
$M_{0,1,1,0}$	$\frac{1}{16}$	0	1	1	2
$M_{0,1,1,1}$	$\frac{1}{16}$	0	2	1	1

G	$\mathcal{U}(G)$	V_{00}	V_{01}	V_{10}	V_{11}
$M_{1,0,0,0}$	$\frac{1}{16}$	0	0	1	1
$M_{1,0,0,1}$	$\frac{1}{16}$	0	1	1	2
$M_{1,0,1,0}$	$\frac{1}{16}$	0	1	1	0
$M_{1,0,1,1}$	$\frac{1}{16}$	0	2	1	1
$M_{1,1,0,0}$	$\frac{1}{16}$	0	0	2	2
$M_{1,1,0,1}$	$\frac{1}{16}$	0	1	2	1
$M_{1,1,1,0}$	$\frac{1}{16}$	0	1	2	1
$M_{1,1,1,1}$	$\frac{1}{16}$	0	2	2	0

Table 3.1: Uniform distribution over $\mathbb{F}_2^{2\times 2}$ along with values of four random variables. (Eq. (3.1) defines the notation used in the G/first column of the tables.)

An *event* $\mathbb{E}$ is a predicate over the domain $\mathbb{D}$, i.e. it maps every element of $\mathbb{D}$ to "true" or "false". Equivalently an event is a subset of the domain $\mathbb{D}$, i.e., those elements that are mapped to true. We switch between "logical" or "set-theoretic" notation to denote combinations of events. So the disjunction of events $\mathbb{E}_1$ and $\mathbb{E}_2$ may be denoted $\mathbb{E}_1 \vee \mathbb{E}_2$ or $\mathbb{E}_1 \cup \mathbb{E}_2$. Similarly, the conjunction of $\mathbb{E}_1$ and $\mathbb{E}_2$ may be denoted $\mathbb{E}_1 \wedge \mathbb{E}_2$ or $\mathbb{E}_1 \cap \mathbb{E}_2$; and the negation of $\mathbb{E}_1$ may be denote $\neg\mathbb{E}_1$ or $\overline{\mathbb{E}_1}$.

In this book, we will primarily deal with the following special distribution:

Definition 3.1.1 (Uniform Distribution)**.** *The* uniform distribution *over* $\mathbb{D}$*, denoted by* $\mathcal{U}_{\mathbb{D}}$*, is given by*

$$\Pr_{\mathcal{U}_{\mathbb{D}}}(x) = \frac{1}{|\mathbb{D}|} \text{ for every } x \in \mathbb{D}.$$

Typically, we will drop the subscript when the domain $\mathbb{D}$ *is clear from the context.*

For example, consider the domain $\mathbb{D} = \mathbb{F}_2^{2\times 2}$, i.e. the set of all 2×2 matrices over $\mathbb{F}_2$. (Note that each such matrix is a generator matrix of some $[2,2]_2$ code.) The first two columns of Table 3.1 list the elements of this $\mathbb{D}$ along with the corresponding probabilities for the uniform distribution, with $M_{b_{00},b_{10},b_{10},b_{11}}$ denoting the following matrix

$$M_{b_{00},b_{10},b_{10},b_{11}} = \begin{pmatrix} b_{00} & b_{10} \\ b_{10} & b_{11} \end{pmatrix}. \tag{3.1}$$

Typically, we will be interested in a real-valued function defined on $\mathbb{D}$ and how it behaves under a probability distribution defined over $\mathbb{D}$. This is captured by the notion of a random variable[1]:

Definition 3.1.2 (Random Variable)**.** *Let* $\mathbb{D}$ *be a finite domain and* $I \subset \mathbb{R}$ *be a finite*[2] *subset. Let* p *be a probability distribution defined over* $\mathbb{D}$*. A* random variable *is a function:*

$$V : \mathbb{D} \to I.$$

[1]We note that the literature on probability theory allows for more general random variables, but for our purposes we restrict only to real-valued ones.

[2]In general, I need not be finite. However, for this book this definition suffices.

The expectation *of V is defined as*

$$\mathbb{E}[V] = \sum_{x \in \mathbb{D}} p(x) \cdot V(x).$$

For example, given $(i, j) \in \{0, 1\}^2$, let V_{ij} denote the random variable $V_{ij}(G) = wt((i, j) \cdot G)$, for any $G \in \mathbb{F}_2^{2\times 2}$. The last four columns of Table 3.1 list the values of these four random variables.

Of particular interest in this book will be binary random variables, i.e., with $I = \{0, 1\}$. In particular, given an *event* E over $\mathbb{D}$, we will define its *indicator variable* to be a function $\mathbb{1}_E : \mathbb{D} \to \{0, 1\}$ such that for any $x \in \mathbb{D}$:

$$\mathbb{1}_E(x) = \begin{cases} 1 & \text{if } x \in E \\ 0 & \text{otherwise.} \end{cases}$$

For example,

$$\mathbb{1}_{V_{01}=0}\left(\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\right) = 1 \text{ and } \mathbb{1}_{V_{01}=0}\left(\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}\right) = 0.$$

In most cases we will shorten this notation to $\mathbb{1}_{E(x)}$ or simply $\mathbb{1}_E$. Finally, sometimes we will abuse notation and use E instead of $\mathbb{1}_E$.

As a further use of indicator variables, consider the expectations of the four indicator variables:

$$\begin{aligned} \mathbb{E}\left[\mathbb{1}_{V_{00}=0}\right] &= 16 \cdot \frac{1}{16} = 1. \\ \mathbb{E}\left[\mathbb{1}_{V_{01}=0}\right] &= 4 \cdot \frac{1}{16} = \frac{1}{4}. \qquad (3.2) \\ \mathbb{E}\left[\mathbb{1}_{V_{10}=0}\right] &= 4 \cdot \frac{1}{16} = \frac{1}{4}. \qquad (3.3) \\ \mathbb{E}\left[\mathbb{1}_{V_{11}=0}\right] &= 4 \cdot \frac{1}{16} = \frac{1}{4}. \qquad (3.4) \end{aligned}$$

3.1.1 Some Useful Results

Before we proceed, we record a simple property of indicator variables that will be useful. (See Exercise 3.1.)

Lemma 3.1.3. *Let E be any event. Then*

$$\mathbb{E}\left[\mathbb{1}_E\right] = \Pr\left[E \text{ is true}\right].$$

Next, we state a simple yet useful property of expectation of a sum of random variables:

Proposition 3.1.4 (Linearity of Expectation)**.** *Given random variables $V_1, \ldots, V_m$ defined over the same domain $\mathbb{D}$ and with the same probability distribution p, we have*

$$\mathbb{E}\left[\sum_{i=1}^{m} V_i\right] = \sum_{i=1}^{m} \mathbb{E}\left[V_i\right].$$

Proof. For notational convenience, define $V = V_1 + \cdots + V_m$. Thus, we have

$$\mathbb{E}[V] = \sum_{x \in \mathbb{D}} V(x) \cdot p(x) \tag{3.5}$$

$$= \sum_{x \in \mathbb{D}} \left(\sum_{i=1}^{m} V_i(x)\right) \cdot p(x) \tag{3.6}$$

$$= \sum_{i=1}^{m} \sum_{x \in \mathbb{D}} V_i(x) \cdot p(x) \tag{3.7}$$

$$= \sum_{i=1}^{m} \mathbb{E}[V_i]. \tag{3.8}$$

In the equalities above, (3.5) and (3.8) follow from the definition of expectation of a random variable. (3.6) follows from the definition of V and (3.7) follows by switching the order of the two summations. □

As an example, we have

$$\mathbb{E}\left[\mathbb{1}_{V_{01}=0} + \mathbb{1}_{V_{10}=0} + \mathbb{1}_{V_{11}=0}\right] = \frac{3}{4} \tag{3.9}$$

Frequently, we will need to deal with the probability of the union of events. We will use the following result to upper bound such probabilities:

Proposition 3.1.5 (Union Bound)**.** *Given m binary random variables $A_1, \ldots, A_m$, we have*

$$\Pr\left[\left(\bigvee_{i=1}^{m} A_i\right) = 1\right] \leq \sum_{i=1}^{m} \Pr\left[A_i = 1\right].$$

Proof. For every $i \in [m]$, define

$$S_i = \{x \in \mathbb{D} | A_i(x) = 1\}.$$

Then we have

$$\Pr\left[\left(\bigvee_{i=1}^{m} A_i\right) = 1\right] = \sum_{x \in \cup_{i=1}^{m} S_i} p(x) \tag{3.10}$$

$$\leq \sum_{i=1}^{m} \sum_{x \in S_i} p(x) \tag{3.11}$$

$$= \sum_{i=1}^{m} \Pr[A_i = 1]. \tag{3.12}$$

In the above, (3.10) and (3.12) follow from the definition of S_i. (3.11) follows from the fact that some of the $x \in \cup_i S_i$ get counted more than once. □

We remark that the union bound is tight when the events are *disjoint*. (In other words, using the notation in the proof above, when $S_i \cap S_j = \emptyset$ for every $i \neq j$.)

As an example, let $A_1 = \nVdash_{V_{01}=0}$, $A_2 = \nVdash_{V_{10}=0}$ and $A_3 = \nVdash_{V_{11}=0}$. Note that in this case the event $A_1 \vee A_2 \vee A_3$ is the same as the event that there exists a non-zero $\mathbf{m} \in \{0,1\}^2$ such that $wt(\mathbf{m} \cdot G) = 0$. Thus, the union bound implies (that under the uniform distribution over $\mathbb{F}_2^{2\times 2}$)

$$\Pr\left[\text{There exists an } \mathbf{m} \in \{0,1\}^2 \setminus \{(0,0)\}, \text{ such that } wt(\mathbf{m}G) = 0\right] \leq \frac{3}{4}. \tag{3.13}$$

Finally, we present three bounds on the probability of a random variable deviating significantly from its expectation. The first bound holds for any random variable:

Lemma 3.1.6 (Markov Bound)**.** *Let V be a non-negative random variable. Then for any $t > 0$,*

$$\Pr[V \geq t] \leq \frac{\mathbb{E}[V]}{t}.$$

In particular, for any $a \geq 1$,

$$\Pr[V \geq a \cdot \mathbb{E}[V]] \leq \frac{1}{a}.$$

Proof. The second bound follows from the first bound by substituting $t = a \cdot \mathbb{E}[V]$. Thus, to complete the proof, we argue the first bound. Consider the following sequence of relations:

$$\begin{aligned}
\mathbb{E}[V] &= \sum_{i \in [0,t)} i \cdot \Pr[V = i] + \sum_{i \in [t,\infty)} i \cdot \Pr[V = i] && (3.14)\\
&\geq \sum_{i \geq t} i \cdot \Pr[V = i] && (3.15)\\
&\geq t \cdot \sum_{i \geq t} \Pr[V = i] && (3.16)\\
&= t \cdot \Pr[V \geq t]. && (3.17)
\end{aligned}$$

In the above relations, (3.14) follows from the definition of expectation of a random variable and the fact that V is non-negative. (3.15) follows as we have dropped some non-negative terms. (3.16) follows by noting that in the summands $i \geq t$. (3.17) follows from the definition of $\Pr[V \geq t]$.

The proof is complete by noting that (3.17) implies the claimed bound. □

The second bound is stated in terms of the *variance* of a random variable, which we define first:

Definition 3.1.7 (Variance)**.** *Let V be a random variable. Its* variance *is defined as*

$$\mathrm{Var}\left[V\right] = \mathbb{E}\left[\left(V^2 - \mathbb{E}\left[V\right]\right)^2\right].$$

The standard deviation *of V is defined as $\sigma\left[V\right] = \sqrt{\mathrm{Var}\left[V\right]}$.*

We have the following bound:

Lemma 3.1.8 (Chebyschev Bound)**.** *Let V be a random variable such that $\mathrm{Var}\left[V\right] \neq 0$. Then for any $t > 0$, we have*

$$\Pr\left[|V - \mathbb{E}\left[V\right]| \geq t\right] \leq \frac{\mathrm{Var}\left[V\right]}{t^2}.$$

Proof. The claim follows from the the following sequence of relations:

$$\begin{aligned}\Pr\left[|V - \mathbb{E}\left[V\right]| \geq t\right] &= \Pr\left[\left(V - \mathbb{E}\left[V\right]\right)^2 \geq t^2\right] \\ &\leq \frac{\mathbb{E}\left[\left(V - \mathbb{E}\left[V\right]\right)^2\right]}{t^2} \\ &= \frac{\mathrm{Var}\left[V\right]}{t^2}.\end{aligned}$$

In the above the inequality follows from Markov's inequality (Lemma 3.1.6) and the last equality follows from definition of variance. □

The third bound works only for sums of *independent* random variables. We begin by defining independent random variables:

Definition 3.1.9 (Independence)**.** *Two random variables A and B are called* independent *if for every a and b in the ranges of A and B respectively, we have*

$$\Pr[A = a \wedge B = b] = \Pr[A = a] \cdot \Pr[B = b].$$

For example, for the uniform distribution in Table 3.1, let A denote the bit $G_{0,0}$ and B denote the bit $G_{0,1}$. It can be verified that these two random variables are independent. In fact, it can be verified all the random variables corresponding to the four bits in G are independent random variables. (We'll come to a related comment shortly.)

Another related concept that we will use is that of probability of an event happening conditioned on another event happening:

Definition 3.1.10 (Conditional Probability)**.** *Given two events A and B defined over the same domain and probability distribution, we define the probability of A* conditioned on *B as*

$$\Pr[A|B] = \frac{\Pr[A \text{ and } B]}{\Pr[B]}.$$

For example, note that

$$\Pr[\not\models_{V_{01}=1} | G_{0,0} = 0] = \frac{4/16}{1/2} = \frac{1}{2}.$$

The above definition implies that two events A and B are independent if and only if $\Pr[A] = \Pr[A|B]$. We will also use the following result later on in the book (see Exercise 3.2):

Lemma 3.1.11. *For any two events A and B defined on the same domain and the probability distribution:*

$$\Pr[A] = \Pr[A|B] \cdot \Pr[B] + \Pr[A|\neg B] \cdot \Pr[\neg B].$$

Next, we state a deviation bound that asserts that the sum of independent random variables takes values close to its expectation with high probability. We only state it for sums of binary random variables, which is the form that will be needed in the book. We refer to this bound as the "Chernoff bound" though we note that this is part of a larger body of work and the bibliographic notes give more details.

Theorem 3.1.12 (Chernoff Bound)**.** *Let $X_1, \ldots, X_m$ be independent binary random variables and define $X = \sum X_i$. Then the* multiplicative *Chernoff bound states that for $0 < \varepsilon \leq 1$,*

$$\Pr\left[|X - \mathbb{E}(X)| > \varepsilon \mathbb{E}(X)\right] < 2e^{-\varepsilon^2 \mathbb{E}(X)/3},$$

and the additive *Chernoff bound states that*

$$\Pr\left[|X - \mathbb{E}(X)| > \varepsilon m\right] < 2e^{-\varepsilon^2 m/2}.$$

We omit the proof, which can be found in any standard textbook on randomized algorithms.

Finally, we present an alternate view of uniform distribution over product spaces and then use that view to prove a result that we will use later in the book. Given probability distributions p_1 and p_2 over domains $\mathbb{D}_1$ and $\mathbb{D}_2$ respectively, we define the product distribution $p_1 \times p_2$ over $\mathbb{D}_1 \times \mathbb{D}_2$ as follows: every element $(x, y) \in \mathbb{D}_1 \times \mathbb{D}_2$ under $p_1 \times p_2$ is picked by choosing x from $\mathbb{D}_1$ according to p_1 and y is picked *independently* from $\mathbb{D}_2$ under p_2. This leads to the following observation (see Exercise 3.4).

Lemma 3.1.13. *For any $m \geq 1$, the distribution $\mathcal{U}_{\mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_m}$ is identical[3] to the distribution $\mathcal{U}_{\mathbb{D}_1} \times \mathcal{U}_{\mathbb{D}_2} \times \cdots \times \mathcal{U}_{\mathbb{D}_m}$.*

For example, the uniform distribution in Table 3.1 can be described equivalently as follows: pick each of the four bits in G independently and uniformly at random from $\{0, 1\}$.

We conclude this section by proving the following result:

[3]We say two distributions p_1 and p_2 on $\mathbb{D}$ are identical if for every $x \in \mathbb{D}$, $p_1(x) = p_2(x)$.

Lemma 3.1.14. *Given a non-zero vector $\mathbf{m} \in \mathbb{F}_q^k$ and a uniformly random $k \times n$ matrix G over $\mathbb{F}_q$, the vector $\mathbf{m} \cdot G$ is uniformly distributed over $\mathbb{F}_q^n$.*

Proof. Let the (j,i)th entry in G $(1 \leq j \leq k, 1 \leq i \leq n)$ be denoted by g_{ji}. Note that as G is a random $k \times n$ matrix over $\mathbb{F}_q$, by Lemma 3.1.13, each of the g_{ji} is an independent uniformly random element from $\mathbb{F}_q$. Now, note that we would be done if we can show that for every $1 \leq i \leq n$, the ith entry in $\mathbf{m} \cdot G$ (call it b_i) is an independent uniformly random element from $\mathbb{F}_q$. To finish the proof, we prove this latter fact. If we denote $\mathbf{m} = (m_1, \ldots, m_k)$, then $b_i = \sum_{j=1}^{k} m_j g_{ji}$. Note that the disjoint entries of G participate in the sums for b_i and b_j for $i \neq j$. Given our choice of G, this implies that the random variables b_i and b_j are independent. Hence, to complete the proof we need to prove that b_i is a uniformly independent element of $\mathbb{F}_q$. The rest of the proof is a generalization of the argument we used in the proof of Proposition 2.6.3.

Note that to show that b_i is uniformly distributed over $\mathbb{F}_q$, it is sufficient to prove that b_i takes every value in $\mathbb{F}_q$ equally often over all the choices of values that can be assigned to $g_{1i}, g_{2i}, \ldots, g_{ki}$. Now, as $\mathbf{m}$ is non-zero, at least one of the its element is non-zero. Without loss of generality assume that $m_1 \neq 0$. Thus, we can write $b_i = m_1 g_{1i} + \sum_{j=2}^{k} m_j g_{ji}$. Now, for every fixed assignment of values to $g_{2i}, g_{3i}, \ldots, g_{ki}$ (note that there are q^{k-1} such assignments), b_i takes a different value for each of the q distinct possible assignments to g_{1i} (this is where we use the assumption that $m_1 \neq 0$). Thus, over all the possible assignments of $g_{1i}, \ldots, g_{ki}$, b_i takes each of the values in $\mathbb{F}_q$ exactly q^{k-1} times, which proves our claim. □

3.2 The Probabilistic Method

The *probabilistic method* is a very powerful method in combinatorics which can be used to show the existence of objects that satisfy certain properties. In this course, we will use the probabilistic method to prove existence of a code $\mathcal{C}$ with certain property $\mathcal{P}$. Towards that end, we define a distribution $\mathcal{D}$ over all possible codes and prove that when $\mathcal{C}$ is chosen according to $\mathcal{D}$:

$$\Pr\left[\mathcal{C} \text{ has property } \mathcal{P}\right] > 0 \text{ or equivalently } \Pr\left[\mathcal{C} \text{ doesn't have property } \mathcal{P}\right] < 1.$$

Note that the above inequality proves the existence of $\mathcal{C}$ with property $\mathcal{P}$.

As an example consider Question 3.0.1. To answer this in the affirmative, we note that the set of all $[2,2]_2$ linear codes is covered by the set of all 2×2 matrices over $\mathbb{F}_2$. Then, we let $\mathcal{D}$ be the uniform distribution over $\mathbb{F}_2^{2\times 2}$. Then by Proposition 2.3.6 and (3.13), we get that

$$\Pr_{\mathcal{U}_{\mathbb{F}_2^{2\times 2}}}\left[\text{There is no } [2,2,1]_2 \text{ code}\right] \leq \frac{3}{4} < 1,$$

which by the probabilistic method answers the Question 3.0.1 in the affirmative.

For the more general case, when we apply the probabilistic method, the typical approach will be to define (sub-)properties $P_1, \ldots, P_m$ such that $\mathcal{P} = P_1 \wedge P_2 \wedge P_3 \ldots \wedge P_m$ and show

that for every $1 \leq i \leq m$:

$$\Pr\left[\mathcal{C} \text{ doesn't have property } P_i\right] = \Pr\left[\overline{P_i}\right] < \frac{1}{m}.$$

Finally, by the union bound, the above will prove that[4] $\Pr\left[\mathcal{C} \text{ doesn't have property } \mathcal{P}\right] < 1$, as desired.

As an example, an alternate way to answer Question 3.0.1 in the affirmative is the following. Define $P_1 = \nVdash_{V_{01} \geq 1}$, $P_2 = \nVdash_{V_{10} \geq 1}$ and $P_3 = \nVdash_{V_{11} \geq 1}$. (Note that we want a $[2,2]_2$ code that satisfies $P_1 \wedge P_2 \wedge P_3$.) Then, by (3.2), (3.3) and (3.4), we have for $i \in [3]$,

$$\Pr\left[\mathcal{C} \text{ doesn't have property } P_i\right] = \Pr\left[\overline{P_i}\right] = \frac{1}{4} < \frac{1}{3},$$

as desired.

Finally, we mention a special case of the general probabilistic method that we outlined above. In particular, let $\mathcal{P}$ denote the property that the randomly chosen $\mathcal{C}$ satisfies $f(\mathcal{C}) \leq b$. Then we claim (see Exercise 3.5) that $\mathbb{E}[f(C)] \leq b$ implies that $\Pr[\mathcal{C} \text{ has property } \mathcal{P}] > 0$. Note that this implies that $\mathbb{E}[f(C)] \leq b$ implies that there exists a code $\mathcal{C}$ such that $f(C) \leq b$.

3.3 The q-ary Entropy Function

Finally, in this chapter we introduce a fundamental function — the "entropy" function — that plays a central role in the analysis of the limits of codes. For example, in Section 4.1 of Chapter 4 we will show how this function captures an upper bound on the rate of codes as a function of the relative distance. Later in Section 4.2 of Chapter 4 we will see that this function captures a lower bound on the rate of codes obtained by the probabilistic method.

We begin with the definition of the entropy function.

Definition 3.3.1 (q-ary Entropy Function)**.** *Let q be an integer and x be a real number such that $q \geq 2$ and $0 \leq x \leq 1$. Then the* q-ary entropy function *is defined as follows:*

$$H_q(x) = x \log_q(q-1) - x \log_q(x) - (1-x) \log_q(1-x).$$

Figure 3.1 presents a pictorial representation of the H_q function for the first few values of q. For the special case of $q = 2$, we will drop the subscript from the entropy function and denote $H_2(x)$ by just $H(x)$, that is, $H(x) = -x \log x - (1-x) \log(1-x)$, where $\log x$ is defined as $\log_2(x)$ (we are going to follow this convention for the rest of the book).

Under the lens of Shannon's entropy function, $H(x)$ denotes the entropy of the distribution over $\{0,1\}$ that selects 1 with probability x and 0 with probability $1-x$. However, there is no similar analogue for the more general $H_q(x)$. The reason why this quantity will turn out to be so central in this book is that it is very closely related to the "volume" of a Hamming ball. We make this connection precise in the next subsection.

[4] Note that $\overline{P} = \overline{P_1} \vee \overline{P_2} \vee \cdots \vee \overline{P_m}$.

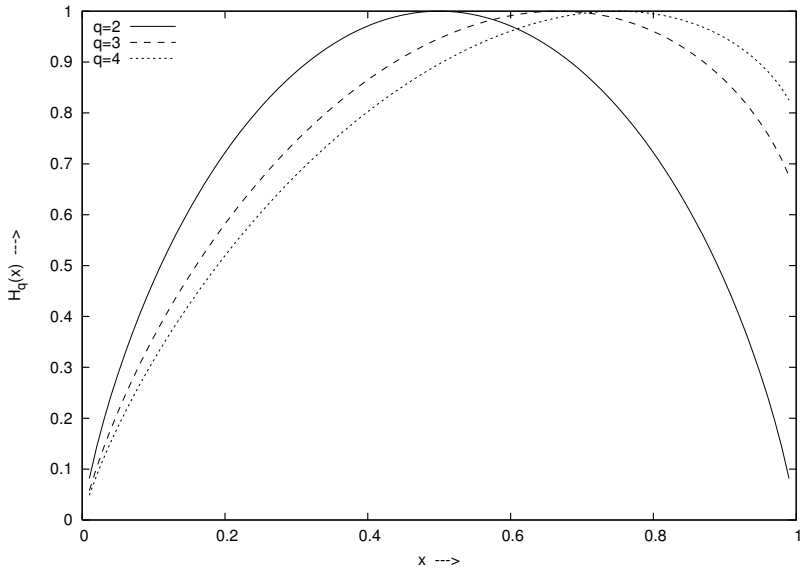

Figure 3.1: A plot of $H_q(x)$ for $q = 2, 3$ and 4. The maximum value of 1 is achieved at $x = 1 - 1/q$.

3.3.1 Volume of Hamming Balls

It turns out that in many of our combinatorial results, we will need good upper and lower bounds on the volume of a Hamming ball. Next we formalize the notion of the volume of a Hamming ball:

Definition 3.3.2 (Volume of a Hamming Ball)**.** *Let $q \geq 2$ and $n \geq r \geq 1$ be integers. Then the volume of a Hamming ball of radius r is given by*

$$Vol_q(r, n) = |B_q(\mathbf{0}, r)| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

The choice of $\mathbf{0}$ as the center for the Hamming ball above was arbitrary: since the volume of a Hamming ball is independent of its center (as is evident from the last equality above), we could have picked any point as the center.

We will prove the following result:

Proposition 3.3.3. *Let $q \geq 2$ be an integer and $0 \leq p \leq 1 - \frac{1}{q}$ be a real number. Then:*

(i) $Vol_q(pn, n) \leq q^{H_q(p)n}$; and

(ii) for large enough n, $Vol_q(pn, n) \geq q^{H_q(p)n - o(n)}$.

Proof. We start with the proof of *(i)*. Consider the following sequence of relations:

$$
\begin{aligned}
1 &= (p+(1-p))^n \\
&= \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} && (3.18)\\
&= \sum_{i=0}^{pn} \binom{n}{i} p^i (1-p)^{n-i} + \sum_{i=pn+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \\
&\geq \sum_{i=0}^{pn} \binom{n}{i} p^i (1-p)^{n-i} && (3.19)\\
&= \sum_{i=0}^{pn} \binom{n}{i} (q-1)^i \left(\frac{p}{q-1}\right)^i (1-p)^{n-i} \\
&= \sum_{i=0}^{pn} \binom{n}{i} (q-1)^i (1-p)^n \left(\frac{p}{(q-1)(1-p)}\right)^i \\
&\geq \sum_{i=0}^{pn} \binom{n}{i} (q-1)^i (1-p)^n \left(\frac{p}{(q-1)(1-p)}\right)^{pn} && (3.20)\\
&= \sum_{i=0}^{pn} \binom{n}{i} (q-1)^i \left(\frac{p}{q-1}\right)^{pn} (1-p)^{(1-p)n} && (3.21)\\
&\geq Vol_q(pn,n) q^{-H_q(p)n}. && (3.22)
\end{aligned}
$$

In the above, (3.18) follows from the binomial expansion. (3.19) follows by dropping the second sum and (3.20) follows from the facts that $\frac{p}{(q-1)(1-p)} \leq 1$ (as[5] $p \leq 1-1/q$). Rest of the steps except (3.22) follow from rearranging the terms. (3.22) follows as $q^{-H_q(p)n} = \left(\frac{p}{q-1}\right)^{pn}(1-p)^{(1-p)n}$.

(3.22) implies that

$$1 \geq Vol_q(pn,n) q^{-H_q(p)n},$$

which proves *(i)*.

We now turn to the proof of part *(ii)*. For this part, we will need Stirling's approximation for $n!$ (Lemma A.1.2).

[5]Indeed, note that $\frac{p}{(q-1)(1-p)} \leq 1$ is true if $\frac{p}{1-p} \leq \frac{q-1}{1}$, which in turn is true if $p \leq \frac{q-1}{q}$, where the last step follows from Lemma A.2.1.

By the Stirling's approximation, we have the following inequality:

$$\begin{aligned}\binom{n}{pn} &= \frac{n!}{(pn)!((1-p)n)!} \\ &> \frac{(n/e)^n}{(pn/e)^{pn}((1-p)n/e)^{(1-p)n}} \cdot \frac{1}{\sqrt{2\pi p(1-p)n}} \cdot e^{\lambda_1(n)-\lambda_2(pn)-\lambda_2((1-p)n)} \\ &= \frac{1}{p^{pn}(1-p)^{(1-p)n}} \cdot \ell(n), \end{aligned} \tag{3.23}$$

where $\ell(n) = \frac{e^{\lambda_1(n)-\lambda_2(pn)-\lambda_2((1-p)n)}}{\sqrt{2\pi p(1-p)n}}$.

Now consider the following sequence of relations that complete the proof:

$$Vol_q(pn, n) \geq \binom{n}{pn}(q-1)^{pn} \tag{3.24}$$

$$> \frac{(q-1)^{pn}}{p^{pn}(1-p)^{(1-p)n}} \cdot \ell(n) \tag{3.25}$$

$$\geq q^{H_q(p)n-o(n)}. \tag{3.26}$$

In the above (3.24) follows by only looking at the last term in the sum that defined $Vol_q(pn, n)$. (3.25) follows from (3.23) while (3.26) follows from the definition of $H_q(\cdot)$ and the fact that for large enough n, $\ell(n)$ is $q^{-o(n)}$. □

Next, we consider how the q-ary entropy function behaves for various ranges of its parameters.

3.3.2 Other Properties of the q-ary Entropy function

This section uses asymptotic analysis in few places. Reader who wish to brush up their knowledge of asymptotic analysis are referred to Appendix **??**.

We begin by recording the behavior of the q-ary entropy function for large q.

Proposition 3.3.4. *For small enough ε, $1 - H_q(\rho) \geq 1 - \rho - \varepsilon$ for every $0 < \rho \leq 1 - 1/q$ if and only if q is $2^{\Omega(1/\varepsilon)}$.*

Proof. We first note that by definition of $H_q(\rho)$ and $H(\rho)$,

$$\begin{aligned} H_q(\rho) &= \rho \log_q(q-1) - \rho \log_q \rho - (1-\rho)\log_q(1-\rho) \\ &= \rho \log_q(q-1) + H(\rho)/\log_2 q. \end{aligned}$$

Now if $q \geq 2^{1/\varepsilon}$, we get that

$$H_q(\rho) \leq \rho + \varepsilon$$

as $\log_q(q-1) \leq 1$ and $H(\rho) \leq 1$. Thus, we have argued that for $q \geq 2^{1/\varepsilon}$, we have $1 - H_q(\rho) \geq 1 - \rho - \varepsilon$, as desired.

Next, we consider the case when $q = 2^{o(1/\varepsilon)}$. We begin by claiming that for small enough ε,

$$\text{if } q \geq 1/\varepsilon^2 \text{ then } \log_q(q-1) \geq 1-\varepsilon.$$

Indeed, $\log_q(q-1) = 1 + (1/\ln q)\ln(1-1/q) = 1 - O\left(\frac{1}{q \ln q}\right)$,[6] which is at least $1-\varepsilon$ for $q \geq 1/\varepsilon^2$ (and small enough ε).

Finally, if $q = 2^{o(\frac{1}{\varepsilon})}$, then for fixed ρ,

$$H(\rho)/\log q = \varepsilon \cdot \omega(1).$$

Then for $q = 2^{o(\frac{1}{\varepsilon})}$ (but $q \geq 1/\varepsilon^2$) we have

$$\rho \log_q(q-1) + H(\rho)/\log q \geq \rho - \varepsilon + \varepsilon \cdot \omega(1) > \rho + \varepsilon,$$

which implies that

$$1 - H_q(\rho) < 1 - \rho - \varepsilon,$$

as desired. For $q \leq 1/\varepsilon^2$, Lemma 3.3.5 shows that $1 - H_q(\rho) \leq 1 - H_{1/\varepsilon^2}(\rho) < 1 - \rho - \varepsilon$, as desired. □

We will also be interested in how $H_q(x)$ behaves for fixed x and increasing q:

Lemma 3.3.5. *Let $q \geq 2$ be an integer and let $0 \leq \rho \leq 1 - 1/q$, then for any real $m \geq 1$ such that*

$$q^{m-1} \geq \left(1 + \frac{1}{q-1}\right)^{q-1}, \tag{3.27}$$

we have

$$H_q(\rho) \geq H_{q^m}(\rho).$$

Proof. Note that $H_q(0) = H_{q^m}(0) = 0$. Thus, for the rest of the proof we will assume that $\rho \in (0, 1-1/q]$.

As observed in the proof of Proposition 3.3.4, we have

$$H_q(\rho) = \rho \cdot \frac{\log(q-1)}{\log q} + H(\rho) \cdot \frac{1}{\log q}.$$

Using this, we obtain

$$H_q(\rho) - H_{q^m}(\rho) = \rho\left(\frac{\log(q-1)}{\log q} - \frac{\log(q^m-1)}{m \log q}\right) + H(\rho)\left(\frac{1}{\log q} - \frac{1}{m \log q}\right).$$

The above in turn implies that

$$\frac{1}{\rho} \cdot m \log q \cdot (H_q(\rho) - H_{q^m}(\rho)) = \log(q-1)^m - \log(q^m-1) + \frac{H(\rho)}{\rho}(m-1)$$

[6] The last equality follows from the fact that by Lemma A.2.2, for $0 < x < 1$, $\ln(1-x) = -O(x)$.

$$
\begin{aligned}
&\geq \log(q-1)^m - \log(q^m-1) + \frac{H(1-1/q)}{1-1/q}(m-1) && (3.28)\\
&= \log(q-1)^m - \log(q^m-1) + (m-1)\left(\log\frac{q}{q-1} + \frac{\log q}{q-1}\right)\\
&= \log\left(\frac{(q-1)^m}{q^m-1}\cdot\left(\frac{q}{q-1}\right)^{m-1}\cdot q^{\frac{m-1}{q-1}}\right)\\
&= \log\left(\frac{(q-1)\cdot q^{m-1}\cdot q^{\frac{m-1}{q-1}}}{q^m-1}\right)\\
&\geq 0 && (3.29)
\end{aligned}
$$

In the above (3.28) follows from the fact that $H(\rho)/\rho$ is decreasing[7] in ρ and that $\rho \leq 1-1/q$. (3.29) follows from the claim that

$$(q-1)\cdot q^{\frac{m-1}{q-1}} \geq q.$$

Indeed the above follows from (3.27).

Finally, note that (3.29) completes the proof. □

Since $(1+1/x)^x \leq e$ (by Lemma A.2.5), we also have that (3.27) is also satisfied for $m \geq 1 + \frac{1}{\ln q}$. Further, we note that (3.27) is satisfied for every $m \geq 2$ (for any $q \geq 3$), which leads to the following (also see Exercise 3.6):

Corollary 3.3.6. *Let $q \geq 3$ be an integer and let $0 \leq \rho \leq 1-1/q$, then for any $m \geq 2$, we have*

$$H_q(\rho) \geq H_{q^m}(\rho).$$

Next, we look at the entropy function when its input is very close to 1.

Proposition 3.3.7. *For small enough $\varepsilon > 0$,*

$$H_q\left(1 - \frac{1}{q} - \varepsilon\right) \leq 1 - c_q\varepsilon^2,$$

where c_q is a constant that only depends on q.

Proof. The intuition behind the proof is the following. Since the derivative of $H_q(x)$ is zero at $x = 1-1/q$, in the Taylor expansion of $H_q(1-1/q-\varepsilon)$ the ε term will vanish. We

[7] Indeed, $H(\rho)/\rho = \log(1/\rho) - (1/\rho - 1)\log(1-\rho)$. Note that the first term is decreasing in ρ. We claim that the second term is also decreasing in ρ – this e.g. follows from the observation that $-(1/\rho-1)\ln(1-\rho) = (1-\rho)(1+\rho/2!+\rho^2/3!+\cdots) = 1-\rho/2-\rho^2(1/2-1/3!)-\cdots$ is also decreasing in ρ.

will now make this intuition more concrete. We will think of q as fixed and $1/\varepsilon$ as growing. In particular, we will assume that $\varepsilon < 1/q$. Consider the following equalities:

$$\begin{aligned}
H_q(1-1/q-\varepsilon) &= -\left(1-\frac{1}{q}-\varepsilon\right)\log_q\left(\frac{1-1/q-\varepsilon}{q-1}\right)-\left(\frac{1}{q}+\varepsilon\right)\log_q\left(\frac{1}{q}+\varepsilon\right)\\
&= -\log_q\left(\frac{1}{q}\left(1-\frac{\varepsilon q}{q-1}\right)\right)+\left(\frac{1}{q}+\varepsilon\right)\log_q\left(\frac{1-(\varepsilon q)/(q-1)}{1+\varepsilon q}\right)\\
&= 1-\frac{1}{\ln q}\left[\ln\left(1-\frac{\varepsilon q}{q-1}\right)-\left(\frac{1}{q}+\varepsilon\right)\ln\left(\frac{1-(\varepsilon q)/(q-1)}{1+\varepsilon q}\right)\right]\\
&= 1+o(\varepsilon^2)-\frac{1}{\ln q}\left[-\frac{\varepsilon q}{q-1}-\frac{\varepsilon^2q^2}{2(q-1)^2}-\left(\frac{1}{q}+\varepsilon\right)\left(-\frac{\varepsilon q}{q-1}\right.\right.\\
&\qquad\left.\left.-\frac{\varepsilon^2q^2}{2(q-1)^2}-\varepsilon q+\frac{\varepsilon^2q^2}{2}\right)\right] \qquad (3.30)\\
&= 1+o(\varepsilon^2)-\frac{1}{\ln q}\left[-\frac{\varepsilon q}{q-1}-\frac{\varepsilon^2q^2}{2(q-1)^2}\right.\\
&\qquad\left.-\left(\frac{1}{q}+\varepsilon\right)\left(-\frac{\varepsilon q^2}{q-1}+\frac{\varepsilon^2q^3(q-2)}{2(q-1)^2}\right)\right]\\
&= 1+o(\varepsilon^2)-\frac{1}{\ln q}\left[-\frac{\varepsilon^2q^2}{2(q-1)^2}+\frac{\varepsilon^2q^2}{q-1}-\frac{\varepsilon^2q^2(q-2)}{2(q-1)^2}\right] \qquad (3.31)\\
&= 1-\frac{\varepsilon^2q^2}{2\ln q(q-1)}+o(\varepsilon^2)\\
&\le 1-\frac{\varepsilon^2q^2}{4\ln q(q-1)}
\end{aligned}$$

(3.32)

(3.30) follows from the fact that for $|x|<1$, $\ln(1+x)=x-x^2/2+x^3/3-\ldots$ (Lemma A.2.2) and by collecting the ε^3 and smaller terms in $o(\varepsilon^2)$. (3.31) follows by rearranging the terms and by absorbing the ε^3 terms in $o(\varepsilon^2)$. The last step is true assuming ε is small enough. □

Next, we look at the entropy function when its input is very close to 0.

Proposition 3.3.8. *For small enough $\varepsilon > 0$,*

$$H_q(\varepsilon) = \Theta\left(\frac{1}{\log q}\cdot\varepsilon\log\left(\frac{1}{\varepsilon}\right)\right).$$

Proof. By definition

$$H_q(\varepsilon) = \varepsilon\log_q(q-1)+\varepsilon\log_q(1/\varepsilon)+(1-\varepsilon)\log_q(1/(1-\varepsilon)).$$

Since all the terms in the RHS are positive we have

$$H_q(\varepsilon) \ge \varepsilon\log(1/\varepsilon)/\log q. \qquad (3.33)$$

Further, by Lemma A.2.2, $(1-\varepsilon)\log_q(1/(1-\varepsilon)) \leq 2\varepsilon/\ln q$ for small enough ε. Thus, this implies that

$$H_q(\varepsilon) \leq \frac{2+\ln(q-1)}{\ln q} \cdot \varepsilon + \frac{1}{\ln q} \cdot \varepsilon \ln\left(\frac{1}{\varepsilon}\right). \tag{3.34}$$

(3.33) and (3.34) proves the claimed bound.

□

We will also work with the inverse of the q-ary entropy function. Note that $H_q(\cdot)$ on the domain $[0, 1-1/q]$ is a bijective map into $[0,1]$. Thus, we define $H_q^{-1}(y) = x$ such that $H_q(x) = y$ and $0 \leq x \leq 1-1/q$. Finally, we will need the following lower bound:

Lemma 3.3.9. *For every $0 < y \leq 1-1/q$ and for every small enough $\varepsilon > 0$,*

$$H_q^{-1}(y - \varepsilon^2/c'_q) \geq H_q^{-1}(y) - \varepsilon,$$

where $c'_q \geq 1$ is a constant that depends only on q.

Proof. It is easy to check that $H_q^{-1}(y)$ is a strictly increasing convex function when $y \in [0,1]$. This implies that the derivative of $H_q^{-1}(y)$ increases with y. In particular, $(H_q^{-1})'(1) \geq (H_q^{-1})'(y)$ for every $0 \leq y \leq 1$. In other words, for every $0 < y \leq 1$, and (small enough) $\delta > 0$,

$$\frac{H_q^{-1}(y) - H_q^{-1}(y-\delta)}{\delta} \leq \frac{H_q^{-1}(1) - H_q^{-1}(1-\delta)}{\delta}.$$

Proposition 3.3.7 along with the facts that $H_q^{-1}(1) = 1-1/q$ and H_q^{-1} is increasing completes the proof if one picks $c'_q = \max(1, 1/c_q)$ and $\delta = \varepsilon^2/c'_q$. □

3.4 Exercises

Exercise 3.1. *Prove Lemma 3.1.3.*

Exercise 3.2. *Prove Lemma 3.1.11.*

Exercise 3.3. *In this exercise, we will see a common use of the Chernoff bound (Theorem 3.1.12). Say we are trying to determine an (unknown) value $x \in \mathbb{F}$ to which we have access to via a randomized algorithm $\mathcal{A}$ that on input (random) input $\mathbf{r} \in \{0,1\}^m$ outputs an estimate $\mathcal{A}(\mathbf{r})$ of x such that*

$$\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{r}) = x] \geq \frac{1}{2} + \gamma,$$

for some $0 < \gamma < \frac{1}{2}$. Then show that for any $t \geq 1$ with $O\left(\frac{t}{\gamma^2}\right)$ calls to $\mathcal{A}$ one can determine x with probability at least $1 - e^{-t}$.

Hint: *Call $\mathcal{A}$ with independent random bits and take majority of the answer and then use the Chernoff bound.*

Exercise 3.4. *Prove Lemma 3.1.13.*

Exercise 3.5. *Let $\mathcal{P}$ denote the property that the randomly chosen $\mathcal{C}$ satisfies $f(\mathcal{C}) \leq b$. Then $\mathbb{E}[f(C)] \leq b$ implies that $\Pr[\mathcal{C}$ has property $\mathcal{P}] > 0$.*

Exercise 3.6. *Prove that for any $Q \geq q \geq 2$ and $\rho \leq 1 - 1/q$, we have $H_Q(\rho) \leq H_q(\rho)$.*

Exercise 3.7. *Prove that for $p < \frac{1}{2}$, we have $H_2(p) \leq O(p \log p)$.*

3.5 Bibliographic Notes

The Chernoff bounds of this chapter come from a family of bounds on the concentration of sums of random variables around their expectation. They originate with the work of Chernoff [10] though Chernoff himself attributes the bound to personal communication with Rubin [3, Page 340]. These bounds and variations are ubiquitous in information theory and computer science — see for instance [11, 32, 31]. Proofs of various concentration bounds can e.g. be found in [12].

The use of the probabilistic method in combinatorics seems to have originated in the early 40s and became especially well known after works of Erdös, notably [15]. Shannon's adoption of the method in [37] is one of the first applications in a broader setting. For more on the probabilistic method, see the book by Alon and Spencer [1].

The entropy function also dates back to Shannon [37]. Shannon's definition is more general and applies to discrete random variables. Our specialization to a two parameter function (namely a function of q and p) is a special case derived from applying the original definition to some special random variables.

Part II

The Combinatorics

Chapter 4

What Can and Cannot Be Done-I

In this chapter, we will try to tackle Question 1.8.2. We will approach this trade-off in the following way:

> If we fix the relative distance of the code to be δ, what is the best rate R that we can achieve?

While we will not be able to pin down the exact optimal relationship between R and δ, we will start establishing some limits. Note that an upper bound on R is a *negative* result in that it establishes that codes with certain parameters do not exist. Similarly, a lower bound on R is a *positive* result.

In this chapter, we will consider only one positive result, i.e. a lower bound on R called the Gilbert-Varshamov bound in Section 4.2. In Section 4.1, we recall a negative result that we have already seen– Hamming bound and state its asymptotic version to obtain an upper bound on R. We will consider two other upper bounds: the Singleton bound (Section 4.3), which gives a tight upper bound for large enough alphabets (but not binary codes) and the Plotkin bound (Section 4.4), which gives a stronger upper bound than Singleton bound for binary codes.

4.1 Asymptotic Version of the Hamming Bound

We have already seen an upper bound in Section 1.7 due to Hamming. However, we had stated this as an upper bound on the dimension k in terms of n, q and d. In this section we convert this into a relation on R versus δ.

Consider any $(n, k, d)_q$ code with rate $R = k/n$ and relative distance $\delta = d/n$. Recall that Theorem 1.7.2 implies the following:

$$R = \frac{k}{n} \leq 1 - \frac{\log_q Vol_q\left(\left\lfloor \frac{d-1}{2} \right\rfloor, n\right)}{n}$$

Recall further that Proposition 3.3.3 states the following lower bound on the volume of a Hamming ball:

$$Vol_q\left(\left\lfloor\frac{d-1}{2}\right\rfloor, n\right) \geq q^{H_q\left(\frac{\delta}{2}\right)n-o(n)}.$$

Taking logarithms to base q of both sides above, and dividing by n yields that the second term in the right hand side of the inequality above is lower bounded by $H_q(\delta/2)-o(1)$, where the $o(1)$ term tends to 0 as $n \to \infty$. Thus Theorem 1.7.2 implies that for a q-ary of code C of rate R, relative distance δ and block length n, we have:

$$R \leq 1 - H_q\left(\frac{\delta}{2}\right) + o(1), \tag{4.1}$$

where the $o(1)$ term tends to 0 as $n \to \infty$. Thus for an infinite family of q-ary codes $\mathcal{C}$, by taking limits as $n \to \infty$, we get the following asymptotic Hamming bound (see Exercise 4.1).

Proposition 4.1.1 (Asymptotic Hamming Bound)**.** *Let $\mathcal{C}$ be an infinite family of q-ary codes with rate $R = R(\mathcal{C})$ and relative distance $\delta = \delta(\mathcal{C})$. Then we have:*

$$R \leq 1 - H_q\left(\frac{\delta}{2}\right).$$

Figure 4.1 gives a pictorial description of the asymptotic Hamming bound for binary codes.

4.2 Gilbert-Varshamov Bound

Next, we will switch gears by proving our first non-trivial lower bound on R in terms of δ. (In fact, this is the only positive result on the R vs δ tradeoff question that we will see in this book.) In particular, we will prove the following result:

Theorem 4.2.1 (Gilbert-Varshamov Bound)**.** *Let $q \geq 2$. For every $0 \leq \delta < 1 - \frac{1}{q}$ there exists a family of q-ary codes $\mathcal{C}$ with rate $R(\mathcal{C}) \geq 1 - H_q(\delta)$ and relative distance $\delta(\mathcal{C}) \geq \delta$. If q is a prime power then there exists such a q-ary family of linear codes. Furthermore, for every $0 \leq \varepsilon \leq 1 - H_q(\delta)$ and integer n, if a matrix G is picked uniformly from $\mathbb{F}_q^{k\times n}$ for $k = n(1 - H_q(\delta) - \varepsilon)$, then G generates a code of rate $1 - H_q(\delta) - \varepsilon$ and relative distance at least δ with probability strictly greater than $1 - q^{-\varepsilon n}$.*

The bound of the theorem is referred to as the GV bound. For a pictorial description of the GV bound for binary codes, see Figure 4.1. We will present the proofs for general codes and linear codes in Sections 4.2.1 and 4.2.2 respectively.

In what follows we first prove the existence of a non-linear code of rate $1 - H_q(\delta)$ and relative distance at least δ. Later we show how to get a linear code, and with high probability (when $\varepsilon > 0$). (Note that the existence of a linear code is implied by the final part using $\varepsilon = 0$.)

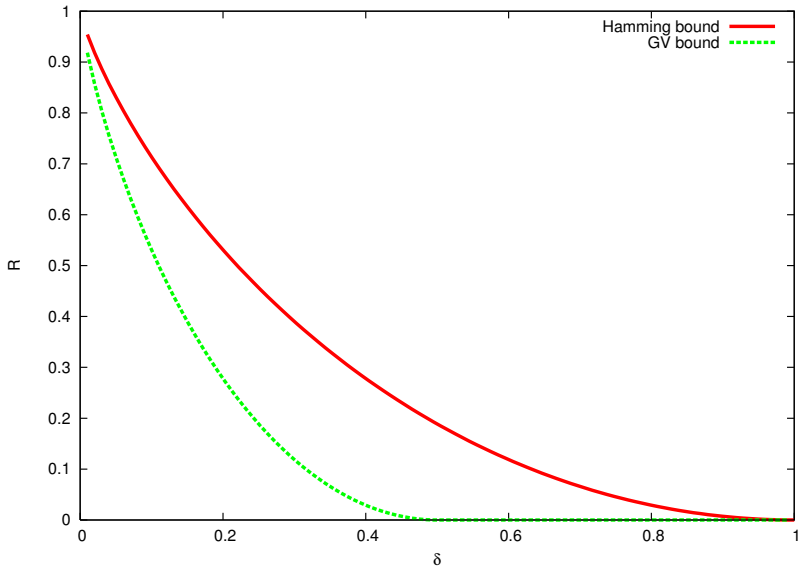

Figure 4.1: The Hamming and GV bounds for binary codes. Note that any point below the GV bound is achievable by some code while no point above the Hamming bound is achievable by any code. In this part of the book we would like to push the GV bound as much up as possible while at the same time try and push down the Hamming bound as much as possible.

4.2.1 Greedy Construction

We will prove Theorem 4.2.1 for general codes by a greedy construction described next: Fix an integer n and let $d = \delta n$. Start with the empty code $C \subseteq [q]^n$ and then keep on adding strings to C that are at Hamming distance at least d from all the existing words in C. Algorithm 4.2.1 presents a formal description of the algorithm and Figure 4.2 illustrates the first few executions of this algorithm.

Algorithm 4.2.1 Gilbert's Greedy Code Construction

INPUT: n, q, d

OUTPUT: A code $C \subseteq [q]^n$ of distance $d \geq 1$

1: $C \leftarrow \emptyset$

2: WHILE there exists a $\mathbf{v} \in [q]^n$ such that $\Delta(\mathbf{v}, \mathbf{c}) \geq d$ for every $\mathbf{c} \in C$ DO

3: Add $\mathbf{v}$ to C

4: RETURN C

We claim that Algorithm 4.2.1 terminates and the C that it outputs has distance d. The latter is true by step 2, which makes sure that in Step 3 we never add a vector $\mathbf{c}$ that will make the distance of C fall below d. For the former claim, note that, if we cannot add $\mathbf{v}$ at some point, we cannot add it later. Indeed, since we only add vectors to C, if a vector $\mathbf{v} \in [q]^n$ is ruled out in a certain iteration of Step 2 because $\Delta(\mathbf{c}, \mathbf{v}) < d$, then in all future

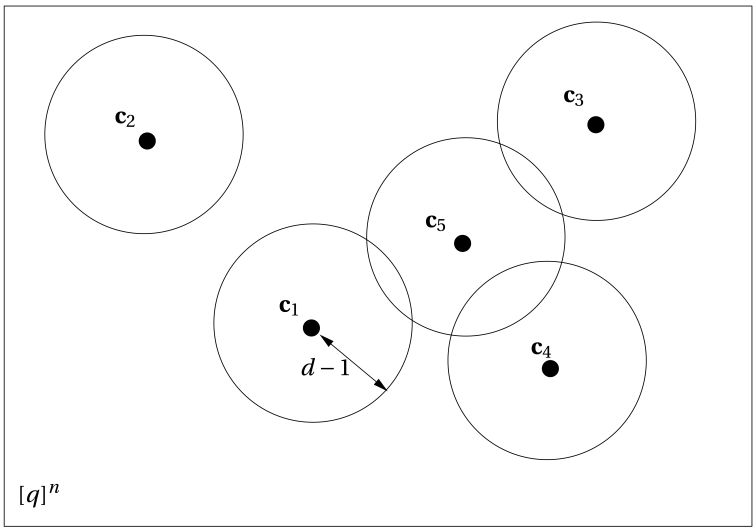

Figure 4.2: An illustration of Gilbert's greedy algorithm for the first five iterations.

iterations, we have $\Delta(\mathbf{v}, \mathbf{c}) < d$ and thus, this $\mathbf{v}$ will never be added in Step 3 in any future iteration.

The running time of Algorithm 4.2.1 is $q^{O(n)}$. To see this, note that Step 2 in the worst-case could be repeated for every vector in $[q]^n$, that is at most q^n times. In a naive implementation, for each iteration, we cycle through all vectors in $[q]^n$ and for each vector $\mathbf{v} \in [q]^n$, iterate through all (at most q^n) vectors $\mathbf{c} \in C$ to check whether $\Delta(\mathbf{c}, \mathbf{v}) < d$. If no such $\mathbf{c}$ exists, then we add $\mathbf{v}$ to C. Otherwise, we move to the next $\mathbf{v}$. However, note that we can do slightly better– since we know that once a $\mathbf{v}$ is "rejected" in an iteration, it'll keep on being rejected in the future iterations, we can fix up an ordering of vectors in $[q]^n$ and for each vector $\mathbf{v}$ in this order, check whether it can be added to C or not. If so, we add $\mathbf{v}$ to C, else we move to the next vector in the order. This algorithm has time complexity $O(nq^{2n})$, which is still $q^{O(n)}$.

Further, we claim that after termination of Algorithm 4.2.1

$$\bigcup_{\mathbf{c} \in C} B(\mathbf{c}, d-1) = [q]^n.$$

This is because if the above is not true, then there exists a vector $\mathbf{v} \in [q]^n \setminus C$, such that $\Delta(\mathbf{v}, \mathbf{c}) \geq d$ and hence $\mathbf{v}$ can be added to C. However, this contradicts the fact that Algorithm 4.2.1 has terminated. Therefore,

$$\left| \bigcup_{\mathbf{c} \in C} B(\mathbf{c}, d-1) \right| = q^n. \tag{4.2}$$

It is not too hard to see that

$$\sum_{\mathbf{c}\in C} |B(\mathbf{c}, d-1)| \geq \left| \bigcup_{\mathbf{c}\in C} B(\mathbf{c}, d-1) \right|,$$

which by (4.2) implies that

$$\sum_{\mathbf{c}\in C} |B(\mathbf{c}, d-1)| \geq q^n$$

or since the volume of a Hamming ball is translation invariant,

$$\sum_{\mathbf{c}\in C} Vol_q(d-1, n) \geq q^n.$$

Since $\sum_{\mathbf{c}\in C} Vol_q(d-1,n) = Vol_q(d-1,n) \cdot |C|$, we have

$$\begin{aligned} |C| &\geq \frac{q^n}{Vol_q(d-1,n)} \\ &\geq \frac{q^n}{q^{nH_q(\delta)}} \qquad (4.3) \\ &= q^{n(1-H_q(\delta))}, \end{aligned}$$

as desired. In the above, (4.3) follows from the fact that

$$\begin{aligned} Vol_q(d-1,n) &\leq Vol_q(\delta n, n) \\ &\leq q^{nH_q(\delta)}, \qquad (4.4) \end{aligned}$$

where the second inequality follows from the upper bound on the volume of a Hamming ball in Proposition 3.3.3.

We thus conclude that for every q, n and δ there exists a code of rate at least $n(1-H_q(\delta))$. We state this formally as a lemma below.

Lemma 4.2.2. *For every pair of positive integers n, q and real $\delta \in [0,1]$ there exists an $(n, k, \delta n)_q$ code satisfying $q^k \geq \frac{q^n}{Vol_q(d-1,n)}$.*

In particular, for every positive integer q and real $\delta \in [0, 1-1/q]$ there exists an infinite family of q-ary codes C of rate R and distance δ satisfying $R \geq 1 - H_q(\delta)$.

It is worth noting that the code from Algorithm 4.2.1 is not guaranteed to have any special structure. In particular, even storing the code can take exponential space. We have seen in Proposition 2.3.3 that linear codes have a much more succinct representation. Thus, a natural question is:

Question 4.2.3. *Do linear codes achieve the $R \geq 1 - H_q(\delta)$ tradeoff that the greedy construction achieves?*

Next, we will answer the question in the affirmative.

4.2.2 Linear Code Construction

Now we will show that a random linear code, with high probability, lies on the GV bound. The construction is a use of the probabilistic method (Section 3.2).

Proof of Theorem 4.2.1. By Proposition 2.3.6, we are done if we can show that there exists a $k \times n$ matrix $\mathbf{G}$ of full rank (for $k = (1 - H_q(\delta) - \varepsilon)n$) such that

$$\text{For every } \mathbf{m} \in \mathbb{F}_q^k \setminus \{\mathbf{0}\}, wt(\mathbf{mG}) \geq d.$$

We will prove the existence of such a $\mathbf{G}$ by the probabilistic method. Pick a random linear code by picking a random $k \times n$ matrix $\mathbf{G}$ where each of kn entries is chosen uniformly and independently at random from $\mathbb{F}_q$. Fix $\mathbf{m} \in \mathbb{F}_q^k \setminus \{\mathbf{0}\}$. Recall that by Lemma 3.1.14, for a random $\mathbf{G}$, $\mathbf{mG}$ is a uniformly random vector from $\mathbb{F}_q^n$. Thus, for every non-zero vector $\mathbf{m}$, we have

$$\begin{aligned} \Pr_{\mathbf{G}}[wt(\mathbf{mG}) < d] &= \frac{Vol_q(d-1,n)}{q^n} \\ &\leq \frac{q^{nH_q(\delta)}}{q^n} \qquad (4.5) \\ &\leq q^{-k} \cdot q^{-\varepsilon n}, \qquad (4.6) \end{aligned}$$

where (4.5) follows from the fact that the condition $wt(\mathbf{mG}) < d$ is equivalent to the condition that $\mathbf{mG} \in B(\mathbf{0}, d-1)$ and the fact that $\mathbf{mG}$ is uniformly random in $\mathbb{F}_q^n$, (4.5) follows from (4.4) and (4.6) uses $k \leq n(1 - H_q(\delta) - \varepsilon)$. There are $q^k - 1$ non-zero vectors $\mathbf{m}$ and taking the union over all such vectors and applying the union bound (Lemma 3.1.5) we have

$$\begin{aligned} \Pr_{\mathbf{G}}[\text{There exists a non-zero } \mathbf{m} \text{ s.t. } \mathrm{wt}(\mathbf{mG}) < d] &\leq (q^k - 1) \cdot q^{-k} \cdot q^{-\varepsilon n} \\ &< q^{-\varepsilon \cdot n}. \end{aligned}$$

Fix a matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ such that for every non-zero $\mathbf{M}$ we have $\mathrm{wt}(\mathbf{mG}) \geq d$. The argument above has shown that a random matrix has this property with probability strictly greater than $1 - q^{-\varepsilon n}$. By Proposition 2.3.6 this implies that the code generated by $\mathbf{G}$ has distance at least d. To conclude the theorem we only need to argue that the code has dimension k, i.e., that $\mathbf{G}$ has full rank. But this also follows immediately from the property that for every $\varepsilon \geq 0$, we have that the probability that the code generated by a uniformly random matrix has distance less than or equal to d is strictly less than 1. Thus using the probabilistic method we conclude there exists a matrix $\mathbf{G}$ such that the code it generates in an $[n, k, d]_q$ code. Furthermore if $\varepsilon > 0$ then the probability that the code does not have distance d is exponentially small, specifically at most $q^{\varepsilon n}$.

To conclude we need to verify that the code generated by $\mathbf{G}$ has dimension k, i.e., that $\mathbf{G}$ has full rank. But note that an equivalent definition of $\mathbf{G}$ not having full rank is that there exists a non-zero vector $\mathbf{M}$ such that $\mathbf{mG} = \mathbf{0}$. But the existence of such a vector $\mathbf{m}$ would imply $\mathrm{wt}(\mathbf{mG}) = 0 < d$ contradicting the property that for every non-zero $\mathbf{M}$ we have $\mathrm{wt}(\mathbf{mG}) \geq d$. We thus conclude that $\mathbf{G}$ generates a code of rate $k/n = 1 - H_q(\delta) - \varepsilon$ and relative distance δ. The theorem follows.

□

Discussion. We now digress a bit to stress some aspects of the GV bound and its proof. First, note that that proof by the probabilistic method shows something stronger than just the existence of a code, but rather gives a high probability result. Furthermore, as pointed out explicitly for the non-linear setting in Lemma 4.2.2, the result gives a lower bound not only in the asymptotic case but also one for every choice of n and k. The proof of the GV bound in the non-linear case gives a similar non-asymptotic bound in the linear setting also.

Note that we can also pick a random linear code by picking a random $(n-k) \times n$ parity check matrix. This also leads to a alternate proof of the GV bound: see Exercise 4.2.

Finally, we note that Theorem 4.2.1 requires $\delta < 1 - \frac{1}{q}$. An inspection of Gilbert and Varshamov's proofs shows that the only reason the proof required that $\delta \leq 1 - \frac{1}{q}$ is because it is needed for the volume bound (recall the bound in Proposition 3.3.3)– $Vol_q(\delta n, n) \leq q^{H_q(\delta)n}$– to hold. It is natural to wonder if the above is just an artifact of the proof or if better codes exist. This leads to the following question:

Question 4.2.4. *Does there exists a code with $R > 0$ and $\delta > 1 - \frac{1}{q}$?*

We will return to this question in Section 4.4.

4.3 Singleton Bound

We will now change gears again and prove an upper bound on R (for fixed δ). We start by proving the Singleton bound.

Theorem 4.3.1 (Singleton Bound)**.** *For every $(n, k, d)_q$ code,*

$$k \leq n - d + 1.$$

Consequently, if C is an infinite family of codes of rate R and relative distance δ then $R \leq 1 - \delta$.

Note that the asymptotic bound hold for any family of codes, even those where the alphabet may grow (arbitrarily) with the length of the code.

Proof. We start by proving the non-asymptotic bound first. The asymptotic version follows easily and is shown at the end.

Let $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M$ be the codewords of an $(n, k, d)_q$ code C. Note that we need to show $M \leq q^{n-d+1}$. To this end, we define $\mathbf{c}'_i$ to be the prefix of the codeword $\mathbf{c}_i$ of length $n-d+1$ for every $i \in [M]$. See Figure 4.3 for a pictorial description.

We now claim that for every $i \neq j$, $\mathbf{c}'_i \neq \mathbf{c}'_j$. For the sake of contradiction, assume that there exits an $i \neq j$ such that $\mathbf{c}'_i = \mathbf{c}'_j$. Notice this implies that $\mathbf{c}_i$ and $\mathbf{c}_j$ agree in all the first $n - d + 1$ positions, which in turn implies that $\Delta(\mathbf{c}_i, \mathbf{c}_j) \leq d - 1$. This contradicts the fact that C has distance d. Thus, M is the number of prefixes of codewords in C of length $n - d + 1$, which implies that $M \leq q^{n-d+1}$ as desired.

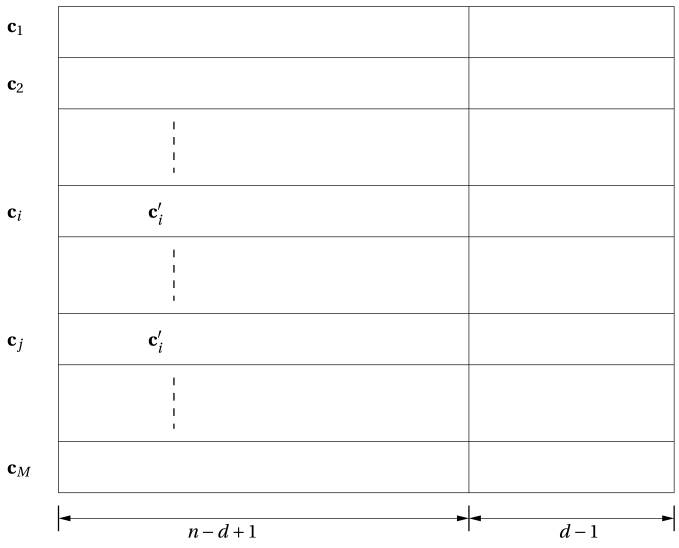

Figure 4.3: Construction of a new code in the proof of the Singleton bound.

To get the asymptotic bound, assume some infinite family of codes $\mathcal{C}$ has rate $R = R(\mathcal{C}) = 1 - \delta + \varepsilon$ for some $\varepsilon > 0$. Then there must exist an $n > 2/\varepsilon$ and a code $C_n \in \mathcal{C}$ that is an $(n, k, d)_q$ code with $k \geq n(1 - \delta + \varepsilon)$ and $d \geq \delta n$. By our choice of n we thus have $k \geq n - d + 2$ contradicting the non-asymptotic bound proved above.

□

Figure 4.4 presents a pictorial description of the asymptotic version of the Singleton bound. It is worth noting that the bound is *independent* of the alphabet size. As is evident from Figure 4.4, the Singleton bound is worse than the Hamming bound for binary codes. However, this bound is better for larger alphabet sizes. In fact, we will look at a family of codes called Reed-Solomon codes in Chapter 5 that meets the Singleton bound. However, the alphabet size of the Reed-Solomon codes increases with the block length n. Thus, a natural follow-up question is the following:

Question 4.3.2. *Given a fixed $q \geq 2$, does there exist a q-ary code that meets the Singleton bound?*

We'll see an answer to this question in the next section.

4.4 Plotkin Bound

In this section, we will study the Plotkin bound, which will answer Questions 4.2.4 and 4.3.2. We start by stating the bound.

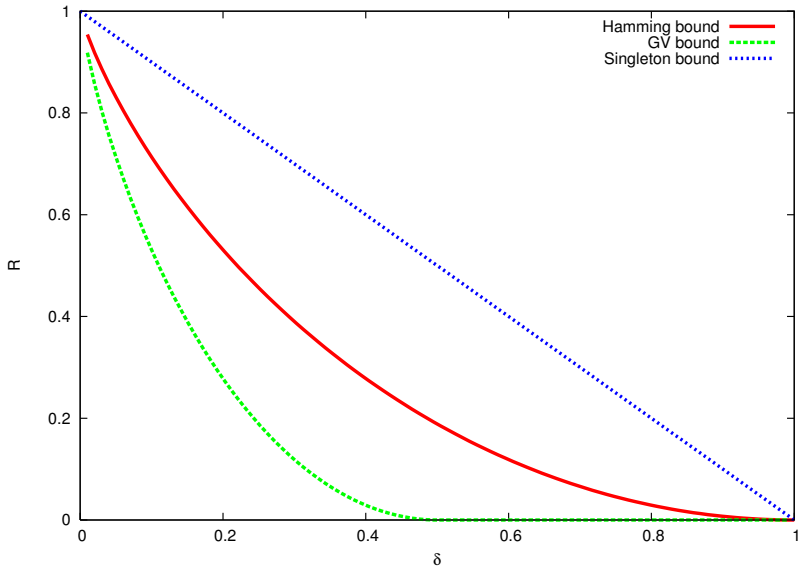

Figure 4.4: The Hamming, GV and Singleton bound for binary codes.

Theorem 4.4.1 (Plotkin bound)**.** *The following hold for any code $C \subseteq [q]^n$ with distance at least d:*

1. *If $d = \left(1 - \frac{1}{q}\right) n$, $|C| \leq 2qn$.*
2. *If $d > \left(1 - \frac{1}{q}\right) n$, $|C| \leq \frac{qd}{qd-(q-1)n}$.*

Note that the Plotkin bound (Theorem 4.4.1) implies that a code with relative distance $\delta \geq 1 - \frac{1}{q}$, must necessarily have $R = 0$, which answers Question 4.2.4 in the negative.

Before proving Theorem 4.4.1, we make a few remarks. We first note that the upper bound in the first part of Theorem 4.4.1 can be improved to $2(q-1)n$ for $q \geq 2$. (See Exercise 4.13.) Second, it can be shown that this bound is tight for $q = 2$. (See Exercise 4.14.) Third, the statement of Theorem 4.4.1 gives a trade-off only for relative distance greater than $1 - 1/q$. However, as the following corollary shows, the result can be extended to work for $0 \leq \delta \leq 1 - 1/q$. (See Figure 4.5 for an illustration for binary codes.)

Corollary 4.4.2. *Let C be an infinite family of q-ary codes with relative distance $0 \leq \delta \leq 1 - \frac{1}{q}$ and rate R. Then*

$$R \leq 1 - \left(\frac{q}{q-1}\right) \delta.$$

Proof. Assume for contradiction that $\mathcal{C}$ is an infinite family of q-ary codes with rate $R = 1 - \left(\frac{q}{q-1}\right) \delta + \varepsilon$ for some $\varepsilon > 0$. Let $C \in \mathcal{C}$ be a code of block length $n \geq \frac{3}{\varepsilon} \cdot \log\left(\frac{1}{\varepsilon}\right)$

with distance $d \leq \delta n$ and message length $k \geq Rn$. We argue now that an appropriate "shortening" of C yields a code contradicting Theorem 4.4.1.

Partition the codewords of C so that codewords within a partition agree on the first $n - n'$ symbols, where $n' = \left\lfloor \frac{qd}{q-1} \right\rfloor - 1$. (We will see later why this choice of n' makes sense.) In particular, for every $\mathbf{x} \in [q]^{n-n'}$, define the 'prefix code'

$$C_{\mathbf{x}} = \{(c_{n-n'+1}, \ldots c_n) \mid (c_1 \ldots c_N) \in C, (c_1 \ldots c_{n-n'}) = \mathbf{x}\}.$$

In other words $C_{\mathbf{x}}$ consists of the n'-length suffixes of all codewords of C that start with the string $\mathbf{x}$.)

By definition $C_{\mathbf{x}}$ is a q-ary code of block length $n' = \left\lfloor \frac{qd}{q-1} \right\rfloor - 1$. We claim that it also has distance at least d for every $\mathbf{x}$: To see this, suppose for some $\mathbf{x}$, $\mathbf{c}_1 \neq \mathbf{c}_2 \in C_{\mathbf{x}}$, $\Delta(\mathbf{c}_1, \mathbf{c}_2) < d$. But this yields two codewords of C, namely $(\mathbf{x}, \mathbf{c}_1)$ and $(\mathbf{x}, \mathbf{c}_2)$, a Hamming distance is less than d from each other, contradicting the assumption that $\Delta(C) \geq d$.

Since $n' < \left(\frac{q}{q-1}\right) d$ (by definition of n') and thus, $d > \left(1 - \frac{1}{q}\right) n'$. Applying Theorem 4.4.1 to $C_{\mathbf{x}}$ we get that

$$|C_{\mathbf{x}}| \leq \frac{qd}{qd - (q-1)n'} \leq qd \leq qn, \tag{4.7}$$

where the second inequality follows from the fact that $qd - (q-1)n'$ is a positive integer and the third is immediate from $d \leq n$.

We now use the bound on $|C_{\mathbf{x}}|$ for all $\mathbf{x}$ to get a bound on $|C|$. Note that by the definition of $C_{\mathbf{x}}$:

$$|C| = \sum_{\mathbf{x} \in [q]^{n-n'}} |C_{\mathbf{x}}|,$$

which by (4.7) implies that

$$|C| \leq \sum_{\mathbf{x} \in [q]^{n-n'}} qn = q^{n-n'+1+\frac{\log n}{\log q}} \leq q^{n - \frac{q}{q-1}d + 1 + \log n} \leq q^{n\left(1 - \delta \cdot \frac{q}{q-1} + \varepsilon\right)},$$

where the first inequality uses the definition of n' and the final inequality uses the fact that $n \geq \frac{3}{\varepsilon} \cdot \log\left(\frac{1}{\varepsilon}\right)$. We conclude that $R \leq 1 - \left(\frac{q}{q-1}\right)\delta + \varepsilon$. Since this holds for every $\varepsilon > 0$ the corollary follows. □

Note that Corollary 4.4.2 implies that for any q-ary code of rate R and relative distance δ (where q is a *constant* independent of the block length of the code), $R < 1 - \delta$. In other words, this answers Question 4.3.2 in the negative.

Let us pause for a bit at this point and recollect the bounds on R versus δ that we have proved till now, which are all depicted in Figure 4.5 (for $q = 2$). The GV bound is the best known lower bound at the time of writing of this book. Better upper bounds are known and we will see one such trade-off (called the Elias-Bassalygo bound) in Section **??**.

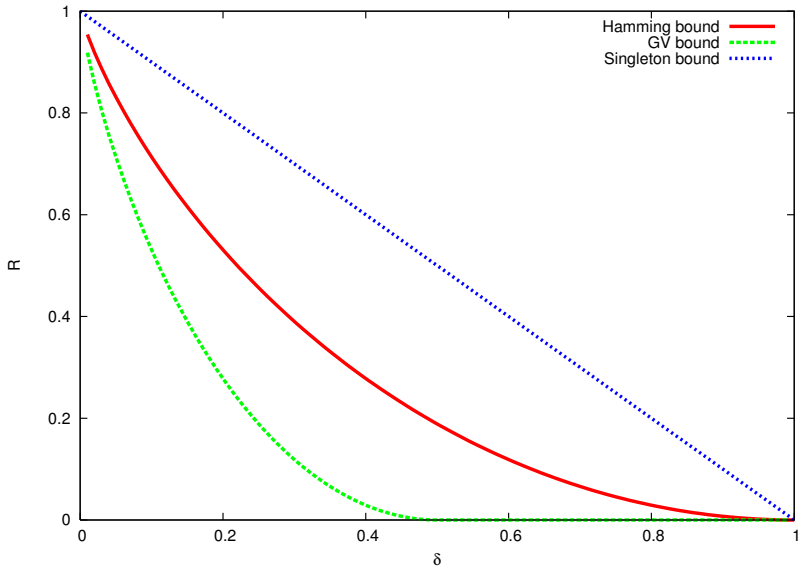

Figure 4.5: The current bounds on the rate R vs. relative distance δ for binary codes. The GV bound is a lower bound on R while the other three bounds are upper bounds on R.

Now, we turn to the proof of Theorem 4.4.1, for which we will need two more lemmas. The first lemma deals with vectors over real spaces. We quickly recap the necessary definitions. Consider a vector $\mathbf{v}$ in $\mathbb{R}^n$, that is, a tuple of n real numbers. This vector has (Euclidean) norm $\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$, and is a unit vector if and only if its norm is 1. The inner product of two vectors, $\mathbf{u}$ and $\mathbf{v}$, is $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_i u_i \cdot v_i$. The following lemma gives a bound on the number of vectors that can exist such that every pair is at an obtuse angle with each other.

Lemma 4.4.3 (Geometric Lemma)**.** *Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m \in \mathbb{R}^N$ be non-zero vectors.*

1. *If $\langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq 0$ for all $i \neq j$, then $m \leq 2N$.*
2. *Let $\mathbf{v}_i$ be unit vectors for $1 \leq i \leq m$. Further, if $\langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq -\varepsilon < 0$ for all $i \neq j$, then $m \leq 1 + \frac{1}{\varepsilon}$.*[1]

(Both items 1 and 2 are tight: see Exercises 4.15 and 4.16.) The proof of the Plotkin bound will need the existence of a map from codewords to real vectors with certain properties, which the next lemma guarantees.

Lemma 4.4.4 (Mapping Lemma)**.** *For every q and n, there exists a function $f : [q]^n \longrightarrow \mathbb{R}^{nq}$ such that for every $\mathbf{c}_1, \mathbf{c}_2 \in [q]^n$ we have*

$$\langle f(\mathbf{c}_1), f(\mathbf{c}_2) \rangle = 1 - \left(\frac{q}{q-1} \right) \left(\frac{\Delta(\mathbf{c}_1, \mathbf{c}_2)}{n} \right).$$

[1]Note that since $\mathbf{v}_i$ and $\mathbf{v}_j$ are both unit vectors, $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ is the cosine of the angle between them.

Consequently we get:

1. *For every* $\mathbf{c} \in [q]^n$, $\|f(\mathbf{c})\| = 1$.
2. *If* $\Delta(\mathbf{c}_1, \mathbf{c}_2) \geq d$ *then we have* $\langle f(\mathbf{c}_1), f(\mathbf{c}_2)\rangle \leq 1 - \left(\frac{q}{q-1}\right)\left(\frac{d}{n}\right)$.

We defer the proofs of the Geometric Lemma and the Mapping Lemma to the end of the section and turn instead to proving Theorem 4.4.1 using the lemmas.

Proof of Theorem 4.4.1. Let $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m\}$ be a q-ary code of block length n and distance d. Let $f : [q]^n \to \mathbb{R}^{nq}$ be the function from Lemma 4.4.4. Then for all i we have that $f(\mathbf{c}_i)$ is a unit length vector in R^{nq}. Furthermore for all $i \neq j$, we have

$$\langle f(\mathbf{c}_i), f(\mathbf{c}_j)\rangle \leq 1 - \left(\frac{q}{q-1}\right)\frac{d}{n}.$$

Thus $f(\mathbf{c}_1), \ldots, f(\mathbf{c}_m)$ give us unit vectors in R^{nq} to which we can apply Lemma 4.4.3 and this will yield the upper bounds claimed on $m = |C|$ in the theorem statement.

For part 1 of the theorem, if $d = \left(1 - \frac{1}{q}\right) n = \frac{(q-1)n}{q}$, then for all $i \neq j$, we have

$$\langle f(\mathbf{c}_i), f(\mathbf{c}_j)\rangle \leq 0.$$

So by the first part of Lemma 4.4.3, $m \leq 2nq$, as desired.

For part 2, if $d > \left(\frac{q-1}{q}\right) n$ then for all $i \neq j$ we have

$$\langle f(\mathbf{c}_i), f(\mathbf{c}_j)\rangle \leq 1 - \left(\frac{q}{q-1}\right)\frac{d}{n} = -\left(\frac{qd - (q-1)n}{(q-1)n}\right).$$

Let $\varepsilon \stackrel{\text{def}}{=} \left(\frac{qd-(q-1)n}{(q-1)n}\right) > 0$. We can apply the second part of Lemma 4.4.3 to $f(\mathbf{c}_1), \ldots, f(\mathbf{c}_m)$ and ε to get $m \leq 1 + \frac{(q-1)n}{qd-(q-1)n} = \frac{qd}{qd-(q-1)n}$, as desired. □

4.4.1 Proof of Geometric and Mapping Lemmas

We now prove Lemmas 4.4.3 and 4.4.4. We start with Lemma 4.4.3, namely the Geometric Lemma.

Proof of Lemma 4.4.3. We prove both parts using linear algebra over the reals.

We start by proving the first part of the lemma. This part is also linear algebraic but involves a few more steps.

We first focus on a subset of the m vectors that has a positive inner product with some fixed vector $\mathbf{u}$. Specifically we pick $\mathbf{u}$ to be a generic vector in $\mathbb{R}^N$ so that $\langle \mathbf{u}, \mathbf{v}_i\rangle \neq 0$ for every i. Such vector exists since the set of vectors satisfying $\langle \mathbf{u}, \mathbf{v}_i\rangle = 0$ is a dimension $N-1$ linear subpace of $\mathbb{R}^N$ (since $\mathbf{v}_i \neq \mathbf{0}$). And the union of N such linear subspaces (one for each $i \in [N]$) cannot cover all of $\mathbb{R}^N$.

Assume w.l.o.g. that at least half of the $\mathbf{v}_i$'s have a positive inner product with $\mathbf{u}$ (if not we can work with $-\mathbf{u}$ instead) and assume further that these are the first $\ell \geq m/2$ vectors by renumbering the vectors. We now show that $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ are linearly independent. This suffices to prove the first part, since linear independence implies $\ell \leq N$ and thus $m \leq 2\ell \leq 2N$.

Assume for contradiction that there is a linear dependency among the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$, i.e., there exist $\alpha_1, \ldots, \alpha_\ell$ with at least one $\alpha_i \neq 0$ such that $\sum_{i \in [\ell]} \alpha_i \mathbf{z}_i = 0$. Note we can assume that at least one α_i is positive since if all are non-negative we can negate all α_i's to get a positive α_i. Further, by renumbering the indices we can assume that there exists $k \geq 1$ such that $\alpha_1, \ldots, \alpha_k > 0$ and $\alpha_{k+1}, \ldots, \alpha_\ell \leq 0$.

Let $\mathbf{w} = \sum_{i=1}^{k} \alpha_i \mathbf{v}_i$ By the definition of α_i's we have that $\mathbf{w} = -\sum_{j=k+1}^{\ell} \alpha_j \mathbf{v}_j$. We first argue that $\mathbf{w} \neq 0$ by using the vector $\mathbf{u}$. Note that we have

$$\langle \mathbf{u}, \mathbf{w} \rangle = \langle \mathbf{u}, \sum_{i=1}^{k} \alpha_i \mathbf{v}_i \rangle = \sum_{i=1}^{k} \alpha_i \langle \mathbf{u}, \mathbf{v}_i \rangle \geq \alpha_1 \langle \mathbf{u}, \mathbf{v}_1 \rangle > 0.$$

We thus conclude $\mathbf{w}$ has a non-zero inner product with some vector and hence can not be the zero vector.

But now we have the following contradiction:

$$0 < \langle \mathbf{w}, \mathbf{w} \rangle = \left\langle \sum_{i=1}^{k} \alpha_i \mathbf{v}_i, -\sum_{j=k+1}^{\ell} \alpha_j \mathbf{v}_j \right\rangle = -\sum_{i=1, j=k+1}^{k,\ell} \alpha_i \alpha_j \langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq 0,$$

where the first inequality uses $\mathbf{w} \neq 0$, the first equality uses the two definitions of $\mathbf{w}$ namely $\mathbf{w} = \sum_{i=1}^{k} \alpha_i \mathbf{v}_i = -\sum_{j=k+1}^{\ell} \alpha_j \mathbf{v}_j$, and the final inequality holds for every term in the summation. Specifically for every $0 \leq i \leq k$ and $k+1 \leq j \leq \ell$ we have $\alpha_i \geq 0$, $\alpha_j \leq 0$ and $\langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq 0$ and so $-\alpha_i \alpha_j \langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq 0$. We conclude that $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ must be linearly independent and this proves the first part of the lemma.

We now move on to the proof of the second part. Define $\mathbf{z} = \mathbf{v}_1 + \ldots + \mathbf{v}_m$. Now consider the following sequence of relationships:

$$\|\mathbf{z}\|^2 = \sum_{i=1}^{m} \|\mathbf{v}_i\|^2 + 2 \sum_{i<j} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq m + 2 \cdot \binom{m}{2} \cdot (-\varepsilon) = m(1 - \varepsilon m + \varepsilon).$$

The inequality follows from the facts that each $\mathbf{v}_i$ is a unit vector and the assumption that for every $i \neq j$, $\langle \mathbf{v}_i . \mathbf{v}_j \rangle \leq -\varepsilon$. As $\|\mathbf{z}\|^2 \geq 0$,

$$m(1 - \varepsilon m + \varepsilon) \geq 0.$$

Since $m \geq 1$, we have that

$$1 - \varepsilon m + \varepsilon \geq 0$$

or

$$\varepsilon m \leq 1 + \varepsilon.$$

Thus, we have $m \leq 1 + \frac{1}{\varepsilon}$, as desired.

Alternate proof of first part. We now present an alternate proof of the first result, which we do by induction on n. Note that in the base case of $N = 0$, we have $m = 0$, which satisfies the claimed inequality $m \leq 2N$.

In the general case, we have $m \geq 1$ non-zero vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^N$ such that for every $i \neq j$,

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq 0. \tag{4.8}$$

Since rotating all the vectors by the same amount does not change the sign of the inner product (nor does scaling any of the vectors), w.l.o.g. we can assume that $\mathbf{v}_m = \langle 1, 0, \ldots, 0 \rangle$. For $1 \leq i \leq m-1$, denote the vectors as $\mathbf{v}_i = \langle \alpha_i, \mathbf{y}_i \rangle$, for some $\alpha_i \in \mathbb{R}$ and $\mathbf{y}_i \in \mathbb{R}^{N-1}$. Now, for any $i \neq 1$, $\langle \mathbf{v}_1, \mathbf{v}_i \rangle = 1 \cdot \alpha_i + \sum_{i=2}^{m} 0 = \alpha_i$. However, we know from (4.8) that $\langle \mathbf{v}_1, \mathbf{v}_i \rangle \leq 0$, which in turn implies that

$$\alpha_i \leq 0. \tag{4.9}$$

Next, we claim that at most one of $\mathbf{y}_1, \ldots, \mathbf{y}_{m-1}$ can be the all zeroes vector, $\mathbf{0}$. If not, assume w.l.o.g., that $\mathbf{y}_1 = \mathbf{y}_2 = \mathbf{0}$. This in turn implies that

$$\begin{aligned} \langle \mathbf{v}_1, \mathbf{v}_2 \rangle &= \alpha_1 \cdot \alpha_2 + \langle \mathbf{y}_1, \mathbf{y}_2 \rangle \\ &= \alpha_1 \cdot \alpha_2 + 0 \\ &= \alpha_1 \cdot \alpha_2 \\ &> 0, \end{aligned}$$

where the last inequality follows from the subsequent argument. As $\mathbf{v}_1 = \langle \alpha_1, \mathbf{0} \rangle$ and $\mathbf{v}_2 = \langle \alpha_2, \mathbf{0} \rangle$ are non-zero, we have that $\alpha_1, \alpha_2 \neq 0$. (4.9) then implies that $\alpha_1, \alpha_2 < 0$. However, $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle > 0$ contradicts (4.8).

Thus, w.l.o.g., assume that $\mathbf{y}_1, \ldots, \mathbf{y}_{m-2}$ are all non-zero vectors. Further, note that for every $i \neq j \in [m-2]$, $\langle \mathbf{y}_i, \mathbf{y}_j \rangle = \langle \mathbf{v}_i, \mathbf{v}_j \rangle - \alpha_i \cdot \alpha_j \leq \langle \mathbf{v}_i, \mathbf{v}_j \rangle \leq 0$. Thus, we have reduced problem on m vectors with dimension N to an equivalent problem on $m-2$ vectors with dimension $N-1$. By induction we have $m - 2 \leq 2(N-1)$ and thus implying $m \leq 2N$.

□

Finally, we prove the Mapping Lemma, i.e., Lemma 4.4.4.

Proof of Lemma 4.4.4. We begin by defining a map $\phi : [q] \to \mathbb{R}^q$ which essentially satisfies the requirements of the lemma statement for the case $n = 1$ (up to some normalization constant). Then, we essentially apply ϕ separately to each coordinates of a word to get the map $f : [q]^n \to \mathbb{R}^{nq}$ that satisfies the claimed properties. We now fill in the details.

Let $\mathbf{e}_i$ denote the unit vector along the ith direction in $\mathbb{R}^q$, i.e.,

$$\mathbf{e}_i = \left\langle 0, 0, \ldots, \underbrace{1}_{i^{\text{th}}\text{position}}, \ldots, 0 \right\rangle.$$

Let $\overline{\mathbf{e}} = \frac{1}{q} \sum_{i \in [q]} \mathbf{e}_i = \langle 1/q, 1/q, \ldots, 1/q \rangle$. Note that we have $\langle \mathbf{e}_i, \mathbf{e}_j \rangle = 1$ if $i = j$ and 0 otherwise. Note also $\langle \overline{\mathbf{e}}, \mathbf{e}_i \rangle = \langle \overline{\mathbf{e}}, \overline{\mathbf{e}} \rangle = 1/q$ for every i.

Now we define $\phi : [q] \to \mathbb{R}^q$ to be $\phi(i) = \mathbf{e}_i - \overline{\mathbf{e}}$. For every pair $i, j \in [q]$ we have

$$\langle \phi(i), \phi(j) \rangle = \langle \mathbf{e}_i - \overline{\mathbf{e}}, \mathbf{e}_j - \overline{\mathbf{e}} \rangle = \langle \mathbf{e}_i, \mathbf{e}_j \rangle - \langle \mathbf{e}_i, \overline{\mathbf{e}} \rangle - \langle \overline{\mathbf{e}}, \mathbf{e}_j \rangle + \langle \overline{\mathbf{e}}, \overline{\mathbf{e}} \rangle = \langle \mathbf{e}_i, \mathbf{e}_j \rangle - 1/q.$$

Thus, for every $i \in [q]$, we get:

$$\|\phi(i)\|^2 = \langle \mathbf{e}_i, \mathbf{e}_i \rangle - 1/q = \frac{(q-1)}{q}. \tag{4.10}$$

Also for every $i \neq j \in [q]$, we have:

$$\langle \phi(i), \phi(j) \rangle = -\frac{1}{q}. \tag{4.11}$$

We are now ready to define our final map $f : [q]^n \to \mathbb{R}^{nq}$. For every $\mathbf{c} = (c_1, \ldots, c_n) \in [q]^n$, define

$$f(\mathbf{c}) = \sqrt{\frac{q}{n(q-1)}} \cdot (\phi(c_1), \phi(c_2), \ldots, \phi(c_n)) .$$

(The multiplicative factor $\sqrt{\frac{q}{n(q-1)}}$ will be used to ensure below that $f(\mathbf{c})$ for every $\mathbf{c} \in [q]^n$ is a unit vector.)

To complete the proof, we will show that f satisfies the claimed properties. We begin with condition 1. Note that

$$\|f(\mathbf{c})\|^2 = \frac{q}{(q-1)n} \cdot \sum_{i=1}^{n} \|\phi(i)\|^2 = 1,$$

where the first equality follows from the definition of f and the second equality follows from (4.10).

We now turn to the second condition. For notational convenience, define $\mathbf{c}_1 = (x_1, \ldots, x_n)$ and $\mathbf{c}_2 = (y_1, \ldots, y_n)$. Consider the following sequence of relations:

$$\begin{aligned}
\langle f(\mathbf{c}_1), f(\mathbf{c}_2) \rangle &= \sum_{\ell=1}^{n} \langle f(x_\ell), f(y_\ell) \rangle \\
&= \left[\sum_{\ell : x_\ell \neq y_\ell} \langle \phi(x_\ell), \phi(y_\ell) \rangle + \sum_{\ell : x_\ell = y_\ell} \langle \phi(x_\ell), \phi(y_\ell) \rangle \right] \cdot \left(\frac{q}{n(q-1)} \right) \\
&= \left[\sum_{\ell : x_\ell \neq y_\ell} \left(\frac{-1}{q} \right) + \sum_{\ell : x_\ell = y_\ell} \left(\frac{q-1}{q} \right) \right] \cdot \left(\frac{q}{n(q-1)} \right) \qquad (4.12) \\
&= \left[\Delta(\mathbf{c}_1, \mathbf{c}_2) \left(\frac{-1}{q} \right) + (n - \Delta(\mathbf{c}_1, \mathbf{c}_2)) \left(\frac{q-1}{q} \right) \right] \cdot \left(\frac{q}{n(q-1)} \right) \qquad (4.13) \\
&= 1 - \Delta(\mathbf{c}_1, \mathbf{c}_2) \left(\frac{q}{n(q-1)} \right) \left[\frac{1}{q} + \frac{q-1}{q} \right] \\
&= 1 - \left(\frac{q}{q-1} \right) \left(\frac{\Delta(\mathbf{c}_1, \mathbf{c}_2)}{n} \right),
\end{aligned}$$

as desired. In the above, (4.12) is obtained using (4.11) and (4.10) while (4.13) follows from the definition of the Hamming distance. □

4.5 Exercises

Exercise 4.1. *Given an infinite family of q-ary codes C of relative distance , and $\varepsilon > 0$ prove that there exists an n_0 such that for all $n \geq n_0$, if $C_n \in C$ is an $[n,k]_q$ code, then $k/n < 1 - H_q(\delta/2) + \varepsilon$. Use this to conclude Proposition 4.1.1.*

Exercise 4.2. *Pick a $(n-k)\times n$ matrix H over $\mathbb{F}_q$ at random. Show that with high probability the code whose parity check matrix is H achieves the GV bound.*

Exercise 4.3. *Recall the definition of an ε-biased space from Exercise 2.15. Show that there exists an ε-biased space of size $O(k/\varepsilon^2)$.*

Hint: *Recall part 1 of Exercise 2.15.*

Exercise 4.4. *Argue that a random linear code as well as its dual both lie on the corresponding GV bound.*

Exercise 4.5. *In Section 4.2.2, we saw that random linear code meets the GV bound. It is natural to ask the question for general random codes. (By a random $(n,k)_q$ code, we mean the following: for each of the q^k messages, pick a random vector from $[q]^n$. Further, the choices for each codeword is independent.) We will do so in this problem.*

1. *Prove that a random q-ary code with rate $R > 0$ with high probability has relative distance $\delta \geq H_q^{-1}(1 - 2R - \varepsilon)$. Note that this is worse than the bound for random linear codes in Theorem 4.2.1.*

2. *Prove that with high probability the relative distance of a random q-ary code of rate R is at most $H_q^{-1}(1-2R) + \varepsilon$. In other words, general random codes are worse than random linear codes in terms of their distance.*
   Hint: *Use Chebyshev's inequality (Lemma 3.1.8).*

Exercise 4.6. *We saw that Algorithm 4.2.1 can compute an $(n,k)_q$ code on the GV bound in time $q^{O(n)}$. Now the construction for linear codes is a randomized construction and it is natural to ask how quickly can we compute an $[n,k]_q$ code that meets the GV bound. In this problem, we will see that this can also be done in $q^{O(n)}$ deterministic time, though the deterministic algorithm is not that straight-forward anymore.*

1. *Argue that Theorem 4.2.1 gives a $q^{O(kn)}$ time algorithm that constructs an $[n,k]_q$ code on the GV bound. (Thus, the goal of this problem is to "shave" off a factor of k from the exponent.)*

2. *A $k \times n$ Toeplitz Matrix $A = \{A_{i,j}\}_{i=1,\ j=1}^{k,\ n}$ satisfies the property that $A_{i,j} = A_{i-1,j-1}$. In other words, any diagonal has the same value. For example, the following is a 4×6*

Toeplitz matrix:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 1 & 2 & 3 & 4 & 5 \\ 8 & 7 & 1 & 2 & 3 & 4 \\ 9 & 8 & 7 & 1 & 2 & 3 \end{pmatrix}$$

A random $k \times n$ Toeplitz matrix $T \in \mathbb{F}_q^{k \times n}$ is chosen by picking the entries in the first row and column uniformly (and independently) at random.

Prove the following claim: For any non-zero $\mathbf{m} \in \mathbb{F}_q^k$, the vector $\mathbf{m} \cdot T$ is uniformly distributed over $\mathbb{F}_q^n$, that is for every $\mathbf{y} \in \mathbb{F}_q^n$, $\Pr[\mathbf{m} \cdot T = \mathbf{y}] = q^{-n}$.

Hint: *Write down the expression for the value at each of the n positions in the vector $\mathbf{m} \cdot T$ in terms of the values in the first row and column of T. Think of the values in the first row and column as variables. Then divide these variables into two sets (this "division" will depend on $\mathbf{m}$) say S and $\overline{S}$. Then argue the following: for every fixed $\mathbf{y} \in \mathbb{F}_q^n$ and for every fixed assignment to variables in S, there is a unique assignment to variables in $\overline{S}$ such that $\mathbf{m}T = \mathbf{y}$.*

3. *Briefly argue why the claim in part 2 implies that a random code defined by picking its generator matrix as a random Toeplitz matrix with high probability lies on the GV bound.*

4. *Conclude that an $[n, k]_q$ code on the GV bound can be constructed in time $q^{O(k+n)}$.*

Exercise 4.7. *Show that one can construct the parity check matrix of an $[n, k]_q$ code that lies on the GV bound in time $q^{O(n)}$.*

Exercise 4.8. *So far in Exercises 4.6 and 4.7, we have seen two constructions of $[n, k]_q$ code on the GV bound that can be constructed in $q^{O(n)}$ time. For constant rate codes, at the time of writing of this book, this is fastest known construction of any code that meets the GV bound. For $k = o(n)$, there is a better construction known, which we explore in this exercise.*

We begin with some notation. For the rest of the exercise we will target a distance of $d = \delta n$. Given a message $\mathbf{m} \in \mathbb{F}_q^k$ and an $[n, k]_q$ code C, define the indicator variable:

$$W_{\mathbf{m}}(C) = \begin{cases} 1 & \text{if } wt(C(\mathbf{m})) < d \\ 0 & \text{otherwise.} \end{cases}$$

Further, define

$$D(C) = \sum_{\mathbf{m} \in \mathbb{F}_q^k \setminus \{\mathbf{0}\}} W_{\mathbf{m}}(C).$$

We will also use $D(G)$ and $W_{\mathbf{m}}(G)$ to denote the variables above for the code C generated by G.

Given an $k \times n$ matrix M, we will use M^i to denote the ith column of M and $M^{\leq i}$ to denote the column submatrix of M that contains the first i columns. Finally below we

will use $\mathcal{G}$ to denote a uniformly random $k \times n$ generator matrix and G to denote a specific instantiation of the generator matrix. We will arrive at the final construction in a sequence of steps. In what follows define $k < (1 - H_q(\delta))n$ for large enough n.

1. *Argue that C has a distance d if and only if $D(C) < 1$.*

2. *Argue that $\mathbb{E}[D(\mathcal{G})] < 1$.*

3. *Argue that for any $1 \leq i < n$ and fixed $k \times n$ matrix G,*

$$\min_{\mathbf{v} \in \mathbb{F}_q^k} \mathbb{E}\left[D(\mathcal{G}) | \mathcal{G}^{\leq i} = G^{\leq i}, \mathcal{G}^{i+1} = \mathbf{v}\right] \leq \mathbb{E}\left[D(\mathcal{G}) | \mathcal{G}^{\leq i} = G^{\leq i}\right].$$

4. *We are now ready to define the algorithm to compute the final generator matrix G: see Algorithm 4.5.1. Prove that Algorithm 4.5.1 outputs a matrix G such that the linear code generated by G is an $[n, k, \delta n]_q$ code. Conclude that this code lies on the GV bound.*

5. *Finally, we will analyze the run time of Algorithm 4.5.1. Argue that Step 2 can be implemented in $\text{poly}\left(n, q^k\right)$ time. Conclude Algorithm 4.5.1 can be implemented in time $\text{poly}\left(n, q^k\right)$.*

   Hint: *It might be useful to maintain a data structure that keeps track of one number for every non-zero $\mathbf{m} \in \mathbb{F}_q^k$ throughout the run of Algorithm 4.5.1.*

Algorithm 4.5.1 $q^{O(k)}$ time algorithm to compute a code on the GV bound

INPUT: Integer parameters $1 \leq k \neq n$ such that $k < (1 - H_q(\delta)n)$
OUTPUT: An $k \times n$ generator matrix G for a code with distance δn
1: Initialize G to be the all 0s matrix ▷ This initialization is arbitrary
2: FOR every $1 \leq i \leq n$ DO
3: $\quad G^i \leftarrow \arg\min_{\mathbf{v} \in \mathbb{F}_q^k} \mathbb{E}\left[D(\mathcal{G}) | \mathcal{G}^{\leq i} = G^{\leq i}, \mathcal{G}^{i+1} = \mathbf{v}\right]$
4: RETURN G

Exercise 4.9. *In this problem we will derive the GV bound using a graph-theoretic proof, which is actually equivalent to the greedy proof we saw in Section 4.2.1. Let $1 \leq d \leq n$ and $q \geq 1$ be integers. Now consider the graph $G_{n,d,q} = (V, E)$, where the vertex set is the set of all vectors in $[q]^n$. Given two vertices $\mathbf{u} \neq \mathbf{v} \in [q]^n$, we have the edge $(u, v) \in E$ if and only if $\Delta(\mathbf{u}, \mathbf{v}) < d$. An* independent set *of a graph $G = (V, E)$ is a subset $I \subseteq V$ such that for every $u \neq v \in I$, we have that (u, v) is* not *an edge. We now consider the following sub-problems:*

1. *Argue that any independent set C of $G_{n,d,q}$ is a q-ary code of distance d.*

2. *The* degree *of a vertex in a graph G is the number of edges incident on that vertex. Let Δ be the maximum degree of any vertex in $G = (V, E)$. Then argue that G has an independent set of size at least $\frac{|V|}{\Delta+1}$.*

3. *Using parts 1 and 2 argue the GV bound.*

Exercise 4.10. *In this problem we will improve slightly on the GV bound using a more sophisticated graph-theoretic proof. Let $G_{n,d,q}$ and N and Δ be as in the previous exercise (Exercise 4.9). So far we used the fact that $G_{n,d,q}$ has many vertices and small degree to prove it has a large independent set, and thus to prove there is a large code of minimum distance d. In this exercise we will see how a better result can be obtained by counting the number of "triangles" in the graph. A triangle in a graph $G = (V, E)$ is a set $\{u, v, w\} \subset V$ of three vertices such that all three vertices are adjancent, i.e., $(u, v), (v, w), (w, u) \in E$. For simplicity we will focus on the case where $q = 2$ and $d = n/5$, and consider the limit as $n \to \infty$.*

1. *Prove that a graph on N vertices of maximum degree Δ has at most $O(N\Delta^2)$ triangles.*

2. *Prove that the number of triangle in graph $G_{n,d,2}$ is at most*

$$2^n \cdot \sum_{0 \le e \le 3d/2} \binom{n}{e} \cdot 3^e.$$

Hint: *Fix u and let e count the number of coordinates where at least one of v or w disagree with u. Prove that e is at most $3d/2$.*

3. *Simplify the expression in the case where $d = n/5$ to show that the number of triangles in $G_{n,n/5,2}$ is $O(N \cdot \Delta^{2-\eta})$ for some $\eta > 0$.*

4. *A famous result in the "probabilistic method" shows (and you don't have to prove this), that if a graph on N vertices of maximum degree Δ has at most $O(N \cdot \Delta^{2-\eta})$ triangles, then it has an independent set of size $\Omega(\frac{N}{\Delta} \log \Delta)$. Use this result to conclude that there is a binary code of block length n and distance $n/5$ of size $\Omega(n2^n/\binom{n}{n/5})$. (Note that this improves over the GV-bound by an $\Omega(n)$ factor.)*

Exercise 4.11. *Use part 1 from Exercise 1.7 to prove the Singleton bound.*

Exercise 4.12. *Let C be an $(n, k, d)_q$ code. Then prove that fixing any $n - d + 1$ positions uniquely determines the corresponding codeword.*

Exercise 4.13. *Our goal in this problem is to improve the bound in part 1 in Theorem 4.4.1. Towards that end,*

1. *Prove that the following holds for every $k \ge 1$. There exists $k + 1$ vectors $\mathbf{v}_i^k \in \mathbb{R}^k$ for $i \in [k + 1]$ such that (1) $\left\|\mathbf{v}_i^k\right\|_2^2 = 1$ for every $i \in [k + 1]$ and (2) $\left\langle \mathbf{v}_i^k, \mathbf{v}_j^k \right\rangle = -\frac{1}{k}$ for every $i \ne j \in [k + 1]$.*

2. *Using the above part, or otherwise, prove the following result. Let C be a q code of block length n and distance $\left(1-\frac{1}{q}\right)n$. Then $|C| \leq 2(q-1)n$. (Note that this is a factor $q/(q-1)$ better than part 1 in Theorem 4.4.1.)*

Exercise 4.14. *Prove that the bound in Exercise 4.13 is tight for $q = 2$– i.e. there exists binary codes C with block length n and distance $n/2$ such that $|C| = 2n$.*

Exercise 4.15. *Prove that part 1 of Lemma 4.4.3 is tight.*

Exercise 4.16. *Prove that part 2 of Lemma 4.4.3 is tight.*

Exercise 4.17. *In this exercise we will prove the Plotkin bound (at least part 2 of Theorem 4.4.1) via a purely combinatorial proof.*

Given an $(n, k, d)_q$ code C with $d > \left(1-\frac{1}{q}\right)n$ define

$$S = \sum_{\mathbf{c}_1 \neq \mathbf{c}_2 \in C} \Delta(\mathbf{c}_1, \mathbf{c}_2).$$

For the rest of the problem think of C has an $|C| \times n$ matrix where each row corresponds to a codeword in C. Now consider the following:

1. *Looking at the contribution of each column in the matrix above, argue that*

$$S \leq \left(1-\frac{1}{q}\right) \cdot n|C|^2.$$

2. *Look at the contribution of the rows in the matrix above, argue that*

$$S \geq |C|\left(|C|-1\right) \cdot d.$$

3. *Conclude part 2 of Theorem 4.4.1.*

Exercise 4.18. *In this exercise, we will prove the so called* Griesmer Bound. *For any $[n, k, d]_q$, prove that*

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{q^i} \right\rceil.$$

Hint: *Recall Exercise 2.18.*

Exercise 4.19. *Use Exercise 4.18 to prove part 2 of Theorem 4.4.1 for linear codes.*

Exercise 4.20. *Use Exercise 4.18 to prove Theorem 4.3.1 for linear codes.*

4.6 Bibliographic Notes

Theorem 4.2.1 was proved for general codes by Gilbert ([16]) and for linear codes by Varshamov ([42]). Hence, the bound is called the Gilbert-Varshamov bound. The Singleton bound (Theorem 4.3.1) is due to Singleton [38], though versions of this result with the same simple proof seem to have appeared earlier in the work of Joshi [25] who only states the bound for the case $q = 2$. For larger (but still constant) values of q, better lower bounds than the GV bound (i.e., results on the existence of codes) are known. In particular, for every prime power $q \geq 49$, there exist linear codes, called *algebraic geometric* (or AG) codes that outperform the corresponding GV bound[2]. AG codes are out of the scope of this book. An introduction to this class of codes can be found, for instance, in a survey by Hholdt, van Lint, and Pellikaan [23]. Exercise 4.10 is from the work of Jiang and Vardy [24].

[2]AG codes are only defined for q being a square or a prime and achieve a rate $R \geq 1 - \delta - \frac{1}{\sqrt{q}-1}$. The lower bound of 49 comes from the fact that it is the smallest square of a prime for which this bound improves on the q-ary GV bound.

Chapter 5

The Greatest Code of Them All: Reed-Solomon Codes

Reed-Solomon codes have been studied a lot in coding theory, and are ubiquitous in practice. These codes are basic and based only very elementary algebra. Yet they are optimal in the sense that they exactly meet the Singleton bound (Theorem 4.3.1). For every choice of n and k satisfying $k \leq n$ there is a Reed-Solomon code of dimension k, block length n and distance $n - k + 1$. As if this were not enough, Reed-Solomon codes turn out to be more versatile: they are fully explicit and they have many applications outside of coding theory. (We will see some applications later in the book.)

These codes are defined in terms of univariate polynomials (i.e. polynomials in one unknown/variable) with coefficients from a finite field $\mathbb{F}_q$. It turns out that polynomials over $\mathbb{F}_p$, for prime p, also help us describe finite fields $\mathbb{F}_{p^s}$, for $s > 1$. We start with a quick review of polynomials over finite fields (for a more careful review, please see Appendix **??**). This will allow us to define Reed-Solomon codes over every field $\mathbb{F}_q$, which we do in the second part of this chapter. Finally in the third part of this chapter we discuss "Maximum Distance Separable" (MDS) codes, which are codes that meet the Singleton bound. We discuss their properties (which in turn are also properties of the Reed-Solomon codes, since they are MDS codes).

5.1 Polynomials and Finite Fields

We start by reviewing the notion of a (univariate) polynomial over a field and define basic notions such as degree, evaluation and root of a polynomial. We conclude with the "degree mantra" that relates the degree to the number of roots.

We begin with the formal definition of a (univariate) polynomial.

Definition 5.1.1. *A polynomial over a variable X and a finite field $\mathbb{F}_q$ is given by a finite sequence $(f_0, f_1, \ldots, f_d)$ with $f_i \in \mathbb{F}_q$ and is denoted by $F(X) = \sum_{i=0}^{d} f_i X^i$. The* degree *of $F(X)$, denoted $\deg(F)$, is the largest index i such that $f_i \neq 0$.*

For example, $2X^3 + X^2 + 5X + 6$ is a polynomial over $\mathbb{F}_7$ of degree 3. We ignore leading zeroes in the definition of a polynomial. For example $0X^4 + 2X^3 + X^2 + 5X + 6$ is the same polynomial as $2X^3 + X^2 + 5X + 6$.

Next, we define some useful notions related to polynomials. We begin with the notion of degree of a polynomial.

We let $\mathbb{F}_q[X]$ denote the set of polynomials over $\mathbb{F}_q$, that is, with coefficients from $\mathbb{F}_q$. Let $F(X), G(X) \in \mathbb{F}_q[X]$ be polynomials. Then $\mathbb{F}_q[X]$ has the following natural operations defined on it:

Addition:

$$F(X) + G(X) = \sum_{i=0}^{\max(\deg(F),\deg(G))} (f_i + g_i)X^i,$$

where the addition on the coefficients is done over $\mathbb{F}_q$. For example, over $\mathbb{F}_2$,

$$X + (1 + X) = X \cdot (1 + 1) + 1 \cdot (0 + 1) = 1$$

(recall that over $\mathbb{F}_2$, $1 + 1 = 0$).[1]

Multiplication:

$$F(X) \cdot G(X) = \sum_{i=0}^{\deg(F)+\deg(G)} \left(\sum_{j=0}^{\min(i,\deg(F))} f_j \cdot g_{i-j} \right) X^i,$$

where all the operations on the coefficients are over $\mathbb{F}_q$. For example, over $\mathbb{F}_2$, $X(1 + X) = X + X^2$; $(1 + X)^2 = 1 + 2X + X^2 = 1 + X^2$, where the latter equality follows since $2 \equiv 0 \bmod 2$.

Next, we define evaluations of a polynomial.

Definition 5.1.2. *Given a polynomial $F(X) \in \mathbb{F}_q[X]$ and $\alpha \in \mathbb{F}_q$, the* evaluation *of $F(X)$ at α, denoted $F(\alpha)$ is $\sum_{i=0}^{\deg F} f_i \alpha^i$. Note that $F(\alpha) \in \mathbb{F}_q$.*[2]

Finally, polynomials don't have multiplicative inverses, but one can divide polynomials by each other and get quotients and residues. The following proposition defines this notion and states some basic properties.

[1] This will be a good time to remember that operations over a finite field are much different from operations over integers/reals. For example, over reals/integers $X + (X + 1) = 2X + 1$.

[2] While this definition requires the coefficients of F and α to come from the same field, it also extends naturally to the case where one of these is from a field $\mathbb{F}_Q$ extending $\mathbb{F}_q$. Since $\mathbb{F}_q \subseteq \mathbb{F}_Q$, if $\alpha \in \mathbb{F}_q$ and $F(X) \in \mathbb{F}_Q[X]$ then the evaluation is well-defined since $\alpha \in \mathbb{F}_Q$. If $F(X) \in \mathbb{F}_q[X]$ then we use the fact that $\mathbb{F}_q[X] \subseteq \mathbb{F}_Q[X]$ to get a definition of $F(\alpha)$. In both cases $F(\alpha) \in \mathbb{F}_Q$.

Proposition 5.1.3 (Polynomial Division)**.** *Given polynomial $f(X), g(X) \in \mathbb{F}_q[X]$ there exist unique polynomials $q(X)$, the quotient, and $r(X)$, the remainder, with $\deg(r) < \deg(g)$ such that $f(X) = q(X)g(X) + r(X)$. If $g(X) = X - \alpha$ for $\alpha \in \mathbb{F}_q$, then $r(X)$ is the degree 0 polynomial $f(\alpha)$, i.e., the evaluation of f at α.*

Definition 5.1.4. *$\alpha \in \mathbb{F}_q$ is a* root *of a polynomial $F(X)$ if $F(\alpha) = 0$.*

For instance, 1 is a root of $1 + X^2$ over $\mathbb{F}_2$.

We now state a basic property of polynomials, the "Degree Mantra", that will be crucial to our use of polynomials to build error-correcting codes. We also introduce the notion of irreducible polynomials whose existence is closely related to the existence of finite fields of prime power size. Finally, motivated by the need to make fields and field operations fully constructive, we briefly remark on the construction of irreducible polynomials.

Proposition 5.1.5 ("Degree Mantra")**.** *A nonzero polynomial $f(X)$ of degree t over a field $\mathbb{F}_q$ has at most t distinct roots in $\mathbb{F}_q$.*

Proof. We will prove the theorem by induction on t. If $t = 0$, we are done. Now, consider $f(X)$ of degree $t > 0$. If f has no roots then we are done, else let $\alpha \in \mathbb{F}_q$ be a root of f. Let $g(X) = X - \alpha$. By the fundamental rule of division of polynomials (Proposition 5.1.3) we have that $f(X) = (X - \alpha)q(X) + f(\alpha) = (X - \alpha)q(X)$. It follows that the degree of $q(X)$ satisfies $\deg(f) = 1 + \deg(q)$, and thus $\deg(q) = t - 1$. Note further that if $\beta \neq \alpha$ is a root of f then we have that $q(\alpha) = f(\beta) \cdot (\beta - \alpha)^{-1}$ and so β is also a root of q. By induction we have that q has at most $t - 1$ roots, and this f has at most t distinct roots (the at most $t - 1$ roots of q plus the root at α).

□

The codes we will construct in this chapter do not need any more algebra, except to describe the finite fields that they work over. To understand finite fields beyond those of prime size, we now describe some more basic properties of polynomials.

5.1.1 Irreducibility and Field Extensions

We will start with a special class of polynomials, called irreducible polynomials, which are analogous to how prime numbers are special for natural numbers.

Definition 5.1.6. *A polynomial $F(X)$ is irreducible if for every $G_1(X), G_2(X)$ such that $F(X) = G_1(X)G_2(X)$, we have* $\min(\deg(G_1), \deg(G_2)) = 0$

For example, $1 + X^2$ is not irreducible over $\mathbb{F}_2$, as

$$(1 + X)(1 + X) = 1 + X^2.$$

However, $1 + X + X^2$ is irreducible, since its non-trivial factors have to be from the linear terms X or $X + 1$. However, it can be checked that neither is a factor of $1 + X + X^2$. (In

fact, one can show that $1 + X + X^2$ is the only irreducible polynomial of degree 2 over $\mathbb{F}_2$– see Exercise 5.4.) A word of caution: if a polynomial $E(X) \in \mathbb{F}_q[X]$ has no root in $\mathbb{F}_q$, it does *not* mean that $E(X)$ is irreducible. For example consider the polynomial $(1+X+X^2)^2$ over $\mathbb{F}_2$– it does not have any root in $\mathbb{F}_2$ but it obviously is not irreducible.

The main reason we consider irreducibility of polynomials in this book is that irreducible polynomials lead us to non-prime fields. Just as the set of integers modulo a prime is a field, so is the set of polynomials modulo an irreducible polynomial, and these fields can have non-prime size. We start by first asserting that they form a field; and then turn to properties such as size later.

Theorem 5.1.7. *Let $E(X)$ be an irreducible polynomial of degree $s \geq 2$ over $\mathbb{F}_p$, p prime. Then the set of polynomials in $\mathbb{F}_p[X]$ modulo $E(X)$, denoted by $\mathbb{F}_p[X]/E(X)$, is a field.*

The proof of the theorem above is similar to the proof of Lemma 2.1.4, so we only sketch the proof here. In particular, we will explicitly state the basic tenets of $\mathbb{F}_p[X]/E(X)$.

- Elements are polynomials in $\mathbb{F}_p[X]$ of degree at most $s-1$. Note that there are p^s such polynomials.
- Addition: $(F(X)+G(X)) \mod E(X) = F(X) \mod E(X) + G(X) \mod E(X) = F(X)+G(X)$. (Since $F(X)$ and $G(X)$ are of degree at most $s-1$, addition modulo $E(X)$ is just plain polynomial addition.)
- Multiplication: $(F(X) \cdot G(X)) \mod E(X)$ is the unique polynomial $R(X)$ with degree at most $s-1$ such that for some $A(X)$, $R(X) + A(X)E(X) = F(X) \cdot G(X)$
- The additive identity is the zero polynomial, and the additive inverse of any element $F(X)$ is $-F(X)$.
- The multiplicative identity is the constant polynomial 1. It can be shown that for every element $F(X)$, there exists a unique multiplicative inverse $(F(X))^{-1}$.

For example, for $p = 2$ and $E(X) = 1 + X + X^2$, $\mathbb{F}_2[X]/(1 + X + X^2)$ has as its elements

$$\{0, 1, X, 1 + X\}.$$

The additive inverse of any element in $\mathbb{F}_2[X]/(1 + X + X^2)$ is the element itself while the multiplicative inverses of

$$1, X \text{ and } 1 + X$$

in $\mathbb{F}_2[X]/(1 + X + X^2)$ are

$$1, 1 + X \text{ and } X$$

respectively.

Next we turn to the size of the field $\mathbb{F}_q[x]/E(X)$ for an irreducible polynomial E.

Lemma 5.1.8. *Let $E(x) \in \mathbb{F}_q[x]$ be an irreducible polynomial of degree s. Then $\mathbb{F}_q[x]/E(x)$ is a field of size q^s.*

Proof. This follows from the fact that the elements of $\mathbb{F}_q[x]/E(x)$ are in one to one correspondence with set of remainders of all polynomials in $\mathbb{F}_q[X]$ when divided by $E(X)$ which in turn is simply the set of all polynomials of degree less than s. The number of such polynomials equals q^s (there are q possibilities for the coefficient of X^i for every $0 \leq i < s$). □

Thus a natural question to ask is if an irreducible polynomials exist for every degree. Indeed, they do. The following theorem asserts this and the reader may find a proof in Appendix **??**.

Theorem 5.1.9. *For all $s \geq 2$ and $\mathbb{F}_p$, there exists an irreducible polynomial of degree s over $\mathbb{F}_p$. In fact, the number of such monic irreducible polynomials is $\Theta\left(\frac{p^s}{s}\right)$.*

The result is true even for general finite fields $\mathbb{F}_q$ and not just prime fields but we stated the version over prime fields for simplicity.

Now recall that Theorem 2.1.5 states that for every prime power p^s, there is a unique field $\mathbb{F}_{p^s}$. This along with Theorems 5.1.7, Lemma 5.1.8 and 5.1.9 imply that:

Corollary 5.1.10. *The field $\mathbb{F}_{p^s}$ is $\mathbb{F}_p[X]/E(X)$, where $E(X)$ is an irreducible polynomial of degree s.*

The facts about irreducible polynomials listed above give sufficient information not only to determine when finite fields exist, but also how to represent them so as to be able to add, multiply or invert elements, given an irreducible polynomial of degree s over $\mathbb{F}_p$. To make our ability to work with fields completely algorithmic we need one more ingredient — one that allows us to find an irreducible polynomial of degree s in $\mathbb{F}_p$ fast. We now turn to this question.

5.1.2 Finding Irreducible Polynomials

Given any monic [3] polynomial $E(X)$ of degree s, it can be verified whether it is an irreducible polynomial by checking if the following two conditions hold (where $\gcd(F(X), G(X))$ is the greatest common denominator (or factor) of polynomials $F(X)$ and $G(X)$):

- $\gcd(E(X), X^{q^s} - X) = E(X)$, and
- For every $t \notin \{1, s\}$ that divides s, we have $\gcd(E(X), X^{q^t} - X) = 1$

[3] I.e. the coefficient of the highest degree term is 1. It can be checked that if $E(X) = e_s X^s + e_{s-1} X^{s-1} + \cdots + 1$ is irreducible, then $e_s^{-1} \cdot E(X)$ is also an irreducible polynomial.

This is true as every irreducible polynomial in $\mathbb{F}_q[X]$ of degree exactly s divides the polynomial $X^{q^s} - X$ (see Proposition **??**). Since Euclid's algorithm for computing the $\gcd(F(X), G(X))$ can be implemented in time polynomial in the minimum of $\deg(F)$ and $\deg(G)$ and $\log q$ (see Section **??**), this implies that checking whether a given polynomial of degree s over $\mathbb{F}_q[X]$ is irreducible can be done in time $\text{poly}(s, \log q)$. It turns out we can improve upon the complexity of checking whether a given polynomial is irreducible slightly (see Exercise 5.5).

We now turn to the question of finding an irreducible polynomial, given q and s. A brute force algorithm can simply enumerate all monic polynomials of degree s over $\mathbb{F}_q$ and test each one for irreducibility. This takes $\text{poly}(q^s)$ time. To get a more efficient algorithm we use randomness and Theorem 5.1.9, while will give us a Las Vegas algorithm[4] to generate an irreducible polynomial of degree s over $\mathbb{F}_q$. We give the code below, but note that the idea of the algorithm is to keep on generating random polynomials until it comes across an irreducible polynomial (Theorem 5.1.9 implies that the algorithm will check $O(p^s)$ polynomials in expectation). Algorithm 5.1.1 presents the formal algorithm.

Algorithm 5.1.1 Generating Irreducible Polynomial

INPUT: Prime power q and an integer $s > 1$
OUTPUT: A monic irreducible polynomial of degree s over $\mathbb{F}_q$

```
b ← 0
WHILE b = 0 DO
    F(X) ← X^s + Σ_{i=0}^{s-1} f_i X^i, where each f_i is chosen uniformly at random from F_q.
    IF gcd(F(X), X^{q^s} − X) = F(X) THEN
        b ← 1.
    FOR all t ∉ {1, s} that divides s DO
        IF gcd(F(X), X^{q^t} − X) ≠ 1) THEN
            b ← 0.
RETURN F(X)
```

The above discussion implies the following:

Corollary 5.1.11. *There is a Las Vegas algorithm to generate an irreducible polynomial of degree s over any $\mathbb{F}_q$ in expected time* $\text{poly}(s, \log q)$.

The above implies that we can 'construct' a finite field $\mathbb{F}_q$ in in randomized $\text{poly}(\log q)$ time. (See Exercise 5.6 for more including details on what it means to 'construct' a finite field.)

This concludes our discussion of polynomials, polynomial arithmetic and properties of polynomials. We now turn to using them to building codes.

[4] A Las Vegas algorithm is a randomized algorithm which always succeeds and we consider its time complexity to be its expected worst-case run time.

5.2 Reed-Solomon Codes

Recall that the Singleton bound (Theorem 4.3.1) states that for every $(n,k,d)_q$ code, $k \leq n-d+1$. Next, we will study Reed-Solomon codes, which meet the Singleton bound (i.e. satisfy $k = n-d+1$) but have the unfortunate property that $q \geq n$. Note that this implies that the Singleton bound is tight, at least for $q \geq n$.

We begin with the definition of Reed-Solomon codes.

Definition 5.2.1 (Reed-Solomon code)**.** *Let $\mathbb{F}_q$ be a finite field, and choose n and k satisfying $k \leq n \leq q$. Fix a sequence $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ...\alpha_n)$ of n distinct elements (also called* evaluation points*) from $\mathbb{F}_q$. We define an encoding function for Reed-Solomon code $\mathrm{RS}_q[\boldsymbol{\alpha}, k] : \mathbb{F}_q^k \to \mathbb{F}_q^n$ as follows. Map a message $\mathbf{m} = (m_0, m_1, ..., m_{k-1})$ with $m_i \in \mathbb{F}_q$ to the degree $k-1$ polynomial.*

$$\mathbf{m} \mapsto f_{\mathbf{m}}(X),$$

where

$$f_{\mathbf{m}}(X) = \sum_{i=0}^{k-1} m_i X^i. \tag{5.1}$$

Note that $f_{\mathbf{m}}(X) \in \mathbb{F}_q[X]$ is a polynomial of degree at most $k-1$. The encoding of $\mathbf{m}$ is the evaluation of $f_{\mathbf{m}}(X)$ at all the α_i's :

$$\mathrm{RS}_q[\boldsymbol{\alpha}, k](\mathbf{m}) = \left(f_{\mathbf{m}}(\alpha_1), f_{\mathbf{m}}(\alpha_2), ..., f_{\mathbf{m}}(\alpha_n)\right).$$

When $q, \boldsymbol{\alpha}$ and k are known from context, we suppress them in the notation and simply refer to the map as RS*. We call the image of this map, i.e., the set $\{\mathrm{RS}[\mathbf{m}] | \mathbf{m} \in \mathbb{F}_q^k\}$, the Reed-Solomon code or* RS *code. A common special case is $n = q-1$ with the set of evaluation points being $\mathbb{F}^* \stackrel{\text{def}}{=} \mathbb{F} \setminus \{0\}$.*

For example, the first row below are all the codewords in the $[3,2]_3$ Reed-Solomon codes where the evaluation points are $\mathbb{F}_3$ (and the codewords are ordered by the corresponding messages from $\mathbb{F}_3^2$ in lexicographic order where for clarity the second row shows the polynomial $f_{\mathbf{m}}(X)$ for the corresponding $\mathbf{m} \in \mathbb{F}_3^2$ in gray):

(0,0,0),	(1,1,1),	(2,2,2),	(0,1,2),	(1,2,0),	(2,0,1),	(0,2,1),	(1,0,2),	(2,1,0)
0,	1,	2,	X,	X+1,	X+2,	2X,	2X+1,	2X+2

Notice that by definition, the entries in $\{\alpha_1, ..., \alpha_n\}$ are distinct and thus, must have $n \leq q$.

In what follows we will describe the basic properties of Reed-Solomon codes. In principle we should refer to the codes as $\mathrm{RS}_q[\boldsymbol{\alpha}, k]$ since all the parameters are needed to specify the code. However for notation simplicity we will assume k, n, q, and $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ are fixed and satisfy $k \leq n \leq q$ allowing us to refer to the resulting code as simply RS. (Thus all results below hold for every such choice of k, n, q and $\alpha_1, \ldots, \alpha_n$.)

Claim 5.2.2. RS *codes are linear codes.*

Proof. The proof follows from the fact that if $a \in \mathbb{F}_q$ and $f(X), g(X) \in \mathbb{F}_q[X]$ are polynomials of degree $\leq k-1$, then $af(X)$ and $f(X)+g(X)$ are also polynomials of degree $\leq k-1$. In particular, let messages $\mathbf{m}_1$ and $\mathbf{m}_2$ be mapped to $f_{\mathbf{m}_1}(X)$ and $f_{\mathbf{m}_2}(X)$ where $f_{\mathbf{m}_1}(X), f_{\mathbf{m}_2}(X) \in \mathbb{F}_q[X]$ are polynomials of degree at most $k-1$ and because of the mapping defined in (5.1), it can be verified that:

$$f_{\mathbf{m}_1}(X) + f_{\mathbf{m}_2}(X) = f_{\mathbf{m}_1+\mathbf{m}_2}(X),$$

and

$$af_{\mathbf{m}_1}(X) = f_{a\mathbf{m}_1}(X).$$

In other words,

$$\mathrm{RS}(\mathbf{m}_1) + \mathrm{RS}(\mathbf{m}_2) = \mathrm{RS}(\mathbf{m}_1 + \mathbf{m}_2)$$

$$a\mathrm{RS}(\mathbf{m}_1) = \mathrm{RS}(a\mathbf{m}_1).$$

Therefore RS is a $[n,k]_q$ linear code. □

The second and more interesting claim is the following:

Claim 5.2.3. *The minimum distance of* RS *is* $n-k+1$.

The claim on the distance follows from Proposition 5.1.5 which asserted that every non-zero polynomial of degree $k-1$ over $\mathbb{F}_q[X]$ has at most $k-1$ roots. The proof below uses this to prove a lower bound on the distance. The upper bound follows from the Singleton Bound (Theorem 4.3.1). Details below.

Proof of Claim 5.2.3. Fix arbitrary $\mathbf{m}_1 \neq \mathbf{m}_2 \in \mathbb{F}_q^k$. Note that $f_{\mathbf{m}_1}(X), f_{\mathbf{m}_2}(X) \in \mathbb{F}_q[X]$ are distinct polynomials of degree at most $k-1$ since $\mathbf{m}_1 \neq \mathbf{m}_2 \in \mathbb{F}_q^k$. Then $f_{\mathbf{m}_1}(X) - f_{\mathbf{m}_2}(X) \neq 0$ also has degree at most $k-1$. Note that $wt(\mathrm{RS}(\mathbf{m}_2) - \mathrm{RS}(\mathbf{m}_1)) = \Delta(\mathrm{RS}(\mathbf{m}_1), \mathrm{RS}(\mathbf{m}_2))$. The weight of $\mathrm{RS}(\mathbf{m}_2) - \mathrm{RS}(\mathbf{m}_1)$ is n minus the number of zeroes in $\mathrm{RS}(\mathbf{m}_2) - \mathrm{RS}(\mathbf{m}_1)$, which is equal to n minus the number of roots that $f_{\mathbf{m}_1}(X) - f_{\mathbf{m}_2}(X)$ has among $\{\alpha_1, ..., \alpha_n\}$. That is,

$$\Delta(\mathrm{RS}(\mathbf{m}_1), \mathrm{RS}(\mathbf{m}_2)) = n - |\{\alpha_i \mid f_{\mathbf{m}_1}(\alpha_i) = f_{\mathbf{m}_2}(\alpha_i)\}|.$$

By Proposition 5.1.5, $f_{\mathbf{m}_1}(X) - f_{\mathbf{m}_2}(X)$ has at most $k-1$ roots. Thus, the weight of $\mathrm{RS}(\mathbf{m}_2) - \mathrm{RS}(\mathbf{m}_1)$ is at least $n-(k-1) = n-k+1$. Therefore $d \geq n-k+1$, and since the Singleton bound (Theorem 4.3.1) implies that $d \leq n-k+1$, we have $d = n-k+1$.[5] The argument above also shows that distinct polynomials $f_{\mathbf{m}_1}(X), f_{\mathbf{m}_2}(X) \in \mathbb{F}_q[X]$ are mapped to distinct codewords. (This is because the Hamming distance between any two codewords is at least $n-k+1 \geq 1$, where the last inequality follows as $k \leq n$.) Therefore, the code contains q^k codewords and has dimension k. The claim on linearity of the code follows from Claim 5.2.2. □

[5] See Exercise 5.2 for an alternate direct argument.

We thus have an exact understanding of the dimension and distance of the Reed-Solomon codes, which we summarize in the theorem below. The theorem also notes that the parameters match those of the Singleton Bound. Recall that the Plotkin bound (Corollary 4.4.2) implies that to achieve the Singleton bound, the alphabet size cannot be a constant. Thus, some growth of q with n is unavoidable to match the Singleton bound, and the Reed-Solomon codes match it with $q \geq n$.

Theorem 5.2.4. RS *is a* $[n, k, n-k+1]_q$ *code. That is,* RS *codes match the Singleton bound.*

Finally, we describe a generator matrix for RS codes. Such a matrix is guaranteed to exist by Claim 5.2.2, but now we give an explicit one. By Definition 5.2.1, any basis $f_{\mathbf{m}_1}, ..., f_{\mathbf{m}_k}$ of polynomial of degree at most $k-1$ gives rise to a basis $\mathrm{RS}(\mathbf{m}_1), ..., \mathrm{RS}(\mathbf{m}_k)$ of the code. A particularly nice polynomial basis is the set of monomials $1, X, ..., X^i, ..., X^{k-1}$. The corresponding generator matrix, whose ith row (numbering rows from 0 to $k-1$) is

$$(\alpha_1^i, \alpha_2^i, ..., \alpha_j^i, ..., \alpha_n^i)$$

and this generator matrix is called the *Vandermonde* matrix of size $k \times n$:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_j & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_j^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_1^i & \alpha_2^i & \cdots & \alpha_j^i & \cdots & \alpha_n^i \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_j^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix}$$

The class of codes that match the Singleton bound have their own name, which we define and study next.

5.3 Maximum Distance Separable Codes and Properties

Definition 5.3.1 (MDS codes)**.** *An* $(n, k, d)_q$ *code is called* Maximum Distance Separable (MDS) *if* $d = n - k + 1$.

Thus, Reed-Solomon codes are MDS codes.

Next, we prove an interesting property of an MDS code $C \subseteq \Sigma^n$ with integral dimension k. We begin with the following notation.

Definition 5.3.2. *For every subset of indices* $S \subseteq [n]$ *of size exactly* k *and a code* $C \subseteq \Sigma^n$, C_S *is the set of all codewords in* C *projected onto the indices in* S.

MDS codes have the following nice property that we shall prove for the special case of Reed-Solomon codes first and subsequently for the general case as well.

Proposition 5.3.3. *Let $C \subseteq \Sigma^n$ of integral dimension k be an MDS code, then for all $S \subseteq [n]$ such that $|S| = k$, we have $|C_S| = \Sigma^k$.*

Before proving Proposition 5.3.3 in its full generality, we present its proof for the special case of Reed-Solomon codes.
Consider any $S \subseteq [n]$ of size k and fix an arbitrary $\mathbf{v} = (v_1, \ldots, v_k) \in \mathbb{F}_q^k$, we need to show that there exists a codeword $\mathbf{c} \in$ RS (assume that the RS code evaluates polynomials of degree at most $k-1$ over $\alpha_1, \ldots, \alpha_n \subseteq \mathbb{F}_q$) such that $\mathbf{c}_S = \mathbf{v}$. Consider a generic degree $k-1$ polynomial $F(X) = \sum_{i=0}^{k-1} f_i X^i$. Thus, we need to show that there exists $F(X)$ such that $F(\alpha_i) = v_i$ for all $i \in S$, where $|S| = k$.

For notational simplicity, assume that $S = [k]$. We think of f_i's as unknowns in the equations that arise out of the relations $F(\alpha_i) = v_i$. Thus, we need to show that there is a solution to the following system of linear equations:

$$\begin{pmatrix} p_0 & p_1 & \cdots & p_{k-1} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ \alpha_1 & \alpha_i & \alpha_k \\ \alpha_1^2 & \alpha_i^2 & \alpha_k^2 \\ \vdots & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_i^{k-1} & \alpha_k^{k-1} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_k \end{pmatrix}$$

The above constraint matrix is a Vandermonde matrix and is known to have full rank (see Exercise 5.3). Hence, by Exercise 2.7, there always exists a unique solution for $(p_0, \ldots, p_{k-1})$. This completes the proof for Reed-Solomon codes.

Next, we prove the property for the general case which is presented below

Proof of Proposition 5.3.3. Consider a $|C| \times n$ matrix where each row represents a codeword in C. Hence, there are $|C| = |\Sigma|^k$ rows in the matrix. The number of columns is equal to the block length n of the code. Since C is Maximum Distance Separable, its distance $d = n - k + 1$.

Let $S \subseteq [n]$ be of size exactly k. It can be verified that for every $\mathbf{c}^i \neq \mathbf{c}^j \in C$, the corresponding projections $\mathbf{c}^i_S$ and $\mathbf{c}^j_S \in C_S$ are not the same. As otherwise $\triangle(\mathbf{c}^i, \mathbf{c}^j) \leq d-1$, which is not possible as the minimum distance of the code C is d. Therefore, every codeword in C gets mapped to a distinct codeword in C_S. As a result, $|C_S| = |C| = |\Sigma|^k$. As $C_S \subseteq \Sigma^k$, this implies that $C_S = \Sigma^k$, as desired. □

Proposition 5.3.3 implies an important property in pseudorandomness: see Exercise 5.14 for more.

5.4 Exercises

Exercise 5.1. *Prove that every function $f : \mathbb{F}_q \to \mathbb{F}_q$ is equivalent to a polynomial $P(X) \in \mathbb{F}_q[X]$ of degree at most $q-1$: that is, for every $\alpha \in \mathbb{F}_q$*

$$f(\alpha) = P(\alpha).$$

Furthermore, prove the choice of this polynomial P is unique.

Exercise 5.2. *For every $[n,k]_q$ Reed-Solomon code, i.e., for every $\mathrm{RS}_q[\boldsymbol{\alpha}, k]$ for every choice of $k \leq n \leq q$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, exhibit two codewords that are at Hamming distance exactly $n-k+1$.*

Exercise 5.3. *Let $\alpha_1, \ldots, \alpha_k$ be distinct elements in a field $\mathbb{F}$. Consider the $k \times k$ Vandermonde matrix $V(\alpha_1, \ldots, \alpha_k)$ whose (i,j)'th entry is α_i^{j-1} for $i, j \in \{1, 2, \ldots, k\}$. Prove that $V(\alpha_1, \ldots, \alpha_k)$ has full rank. Use this property to prove that a Reed-Solomon code of dimension k can efficiently correct $n-k$ erasures.*

Exercise 5.4. *Prove that $X^2 + X + 1$ is the unique irreducible polynomial of degree two over $\mathbb{F}_2$.*

Exercise 5.5. *Let $s \geq 1$ be an integer and let r be the number of prime divisors of s and let $\tau(s)$ be the number of divisors of s. In this problem we will consider the number of* gcd *operations we need to decide whether a given polynomial of degree s is irreducible or not.*

1. *Prove that $\tau(s)-1$ calls to* gcd *are enough to decide if a degree s polynomial is irredicible or not.*
   Hint: *This is what is used in Algorithm 5.1.1.*

2. *Let $p_1, \ldots, p_r$ be the prime divisors of s. Then prove that a degree s polynomial $E(X)$ is irreducible iff*
   - $\gcd(E(X), X^{q^s} - X) = E(X)$, *and*
   - *For every $i \in [r]$, we have* $\gcd\left(E(X), X^{q^{\frac{s}{p_i}}} - X\right) = 1$

3. *Using the above part or otherwise argue that $r+1$ calls to* gcd *are enough to decide if a degree s polynomial is irredicible or not. Further, argue that this is exponentially fewer calls than the result in the first part.*
   Hint: *Prove and then use the fact that $\tau(s) \geq 2^r$.*

Exercise 5.6. *In this problem we will consider what it means to 'construct' a finite field. For simplicity, assume that $q = p^s$ for some $s \geq 1$. A* representation *of a finite field $\mathbb{F}_q$ is a triple (S, θ, f) where $S \subset \{0,1\}^*$ with $|S| = p^s$ is set of representations of elements on $\mathbb{F}_q$, θ is some 'auxiliary' representation and a bijection $f : \mathbb{F}_{p^s} \to S$. For every $\alpha \in \mathbb{F}_{p^s}$, $f(\alpha)$ is the* representation *Also implicit in this definition is given $\alpha, \beta \in \mathbb{F}_p^s$ how one computes*

$f(\alpha)+f(\beta), -f(\alpha), f(\alpha)\cdot f(\beta)$. *Further, one needs to identify the additive and multiplicative identities in* S. *Finaly, given a non-zero element* $\alpha \in \mathbb{F}_{p^s}$, *compute* $f(\alpha)^{-1}$. *The auxiliary representation* θ *can be used to implement these operations.*

We call a representation efficient *of all of the operations can be supported in* $\mathrm{poly}(\log q)$ *time. In this problem we will explore the problem of constructing an efficient representation of a finite field in* $\mathrm{poly}(\log q)$ *(randomized) time.*

1. *Let* $E(X)$ *be an irreducible polynomial of degree* s. *Given* $E(X)$, *prove that the repreentation* $\mathbb{F}_p[X]/E(X)$ *(i.e.* $\theta = E(X)$ *and for every* $\mathbf{u} \in \mathbb{F}_p^s$,[6] $f(\mathbf{u}) = f_{\mathbf{u}}(X)$ *as per* (5.1) *and the additive and multiplicative identities are the* 0 *and* 1 *polynomials) is an efficient representation.*

   Hint: *The following fact might be useful: for every* $\alpha \in \mathbb{F}_q^*$, $\alpha^{q-2} = \alpha^{-1}$.

2. *Using the above part or otherwise prove that for every prime* p *and integer* $s \geq 1$, *an efficient representation of* $\mathbb{F}_{p^s}$ *can be computed in (randomized)* $\mathrm{poly}(s \log p)$ *time.*

Exercise 5.7. *In Exercise 2.17, we saw that any linear code can be converted in to a systematic code. In other words, there is a map to convert Reed-Solomon codes into a systematic one. In this exercise the goal is to come up with an explicit encoding function that results in a systematic Reed-Solomon code.*

In particular, given the set of evaluation points $\alpha_1, \ldots, \alpha_n$, *design an explicit map* f *from* $\mathbb{F}_q^k$ *to a polynomial of degree at most* $k-1$ *such that the following holds. For every message* $\mathbf{m} \in \mathbb{F}_q^k$, *if the corresponding polynomial is* $f_{\mathbf{m}}(X)$, *then the vector* $(f_{\mathbf{m}}(\alpha_i))_{i\in[n]}$ *has the message* $\mathbf{m}$ *appear in the corresponding codeword (say in its first* k *positions). Further, prove that this map results in an* $[n, k, n-k+1]_q$ *code.*

Exercise 5.8. *Let* $\boldsymbol{\alpha} \subseteq \mathbb{F}_q^q$ *be a vector enumerating all the elements of the field* $\mathbb{F}_q$. *Prove that*

$$(\mathrm{RS}_q[\boldsymbol{\alpha}, k])^{\perp} = (\mathrm{RS}_q[\boldsymbol{\alpha}, q-k])\,.$$

that is, the dual of these Reed-Solomon code are Reed-Solomon codes themselves. Conclude that the class of Reed-Solomon codes contain self-dual code (see Exercise 2.33 for a definition).

Exercise 5.9. *We have defined Reed-Solomon codes as evaluation codes. They are sometimes also defined in an alternate way, as coefficients of polynomials with pre-specified roots, and this exercise will demonstrate the equivalence of the two ways.*

Let $\mathbb{F}_q$ *be a field, and* $\mathbb{F}_q^*$ *be the multiplicative group of its nonzero elements. Let* $n = q-1$ *and let* α *be a generator of* $\mathbb{F}_q^*$ *so that the vector* $\boldsymbol{\alpha} = (1, \alpha, \ldots, \alpha^{n-1})$ *has all distinct elements and* $\alpha^n = 1$. *Consider the Reed-Solomon code over a field* $\mathbb{F}_q$ *with evaluation points being* $\boldsymbol{\alpha}$*:*

$$\mathrm{RS}_q[\boldsymbol{\alpha}, k] = \{(p(1), p(\alpha), \ldots, p(\alpha^{n-1})) \;\mid\; p(X) \in \mathbb{F}[X] \text{ has degree} \leq k-1\}.$$

[6] Note that there is a bijection between $\mathbb{F}_{p^s}$ to $\mathbb{F}_p^s$ and hence we can define f on $\mathbb{F}_p^s$ instead of $\mathbb{F}_{p^s}$.

Prove that

$$\begin{aligned}\mathrm{RS}_q[\boldsymbol{\alpha}, k] \;&=\; \{(c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}^n \mid C(\alpha^\ell) = 0 \text{ for } 1 \le \ell \le n-k\ , \\ &\qquad \text{where } C(X) = c_0 + c_1 X + \cdots + c_{n-1}X^{n-1}\}\ . \end{aligned} \tag{5.2}$$

Hint: *Exercise 2.3 might be useful.*

Exercise 5.10 (Generalized Reed-Solomon Codes)**.**

For a field $\mathbb{F}$ with $|\mathbb{F}| \ge n$, an n-tuple $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ of n distinct *elements of $\mathbb{F}$, and a vector $\mathbf{v} = (v_1, v_2, \ldots, v_n) \in (\mathbb{F}^*)^n$ of n (not necessarily distinct)* nonzero *elements from $\mathbb{F}$, the* Generalized Reed-Solomon code $\mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, k, \mathbf{v}]$ *is defined as follows:*

$$\mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, k, \mathbf{v}] = \{(v_1 \cdot p(\alpha_1), v_2 \cdot p(\alpha_2), \ldots, v_n \cdot p(\alpha_n)) \;\mid\; p(X) \in \mathbb{F}[X] \text{ has degree } < k\}\ . \tag{5.3}$$

(In particular, note that $\mathrm{RS}_q[\boldsymbol{\alpha}, k] = \mathrm{GRS}_{\mathbb{F}_q}[\boldsymbol{\alpha}, k, (1, \ldots, 1)]$.)

1. *Prove that $\mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, k, \mathbf{v}]$ is an $[n, k, n-k+1]_{\mathbb{F}}$ linear code.*
2. *Prove that the dual code of $\mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, k, \mathbf{v}]$ is*

   $$\mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, k, \mathbf{v}]^\perp = \mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, n-k, \mathbf{u}]$$

   for $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in (\mathbb{F}^*)^n$ where for $i = 1, 2, \ldots, n$,

   $$u_i = \frac{1}{v_i \prod_{j \ne i}(\alpha_i - \alpha_j)}\ .$$

   Hint: *First show that it suffices to prove that for every polynomial p of degree $< k$ and every polynomial q of degree $< n-k$, it is the case that $\sum_{i=1}^n u_i v_i p(\alpha_i) q(\alpha_i) = 0$. Next, express an arbitrary polynomial h of degree $< n$ in terms of the Lagrange polynomials L_i that satisfy $L_i(\alpha_j) = 1$ if $i = j$ and 0 otherwise. Apply to the polynomial $h = p \cdot q$ and use the fact that the coefficient of x^{n-1} in h is zero.*
3. *Prove that the dual of $\mathrm{RS}[\boldsymbol{\alpha}, k]$, when α enumerates all elements of $\mathbb{F}_q^*$, is the variant of a Reed-Solomon code that maps a message polynomial $m(X)$ with degree $< n-k$ to evaluations of $X \cdot m(X)$ on $\boldsymbol{\alpha}$.*
4. *Derive Exercise 5.8 as a corollary of Part 2.*

Exercise 5.11. *In this problem we will look at a very important class of codes called BCH codes*[7]*.*

Fix an integer m and let $q = 2^m$ and $n = q - 1$. Let non-zero elements of the field $\mathbb{F}_{2^m}$ be $\{\eta_1, \ldots, \eta_n\}$ and let $\boldsymbol{\alpha} = (\eta_1, \ldots, \eta_n)$. Given non-negative integer $k \le n$, the binary BCH code, denoted $C_{\mathrm{BCH}} = C_{\mathrm{BCH}}(m, k)$, is defined as $\mathrm{RS}_{2^m}[\boldsymbol{\alpha}, k] \cap \mathbb{F}_2^n$. In other words C_{BCH} consists of those codewords in the Reed-Solomon code $\mathrm{RS}_{2^m}[\boldsymbol{\alpha}, k]$ all of whose coordinates lie in the subfield $\mathbb{F}_2 \subseteq \mathbb{F}_{2^m}$.

[7] The acronym BCH stands for Bose-Chaudhuri-Hocquenghem, the discoverers of this family of codes.

1. *Let $d = n - k + 1$. Prove that C_{BCH} is a binary linear code of distance at least d and dimension at least $n - (d-1)\log_2(n+1)$.*

   Hint: *Use the characterization (5.2) of the Reed-Solomon code from Exercise 5.9.*

2. *Prove a better lower bound of $n - \left\lceil \frac{d-1}{2} \right\rceil \log_2(n+1)$ on the dimension of C_{BCH}.*

   Hint: *There are redundant checks among the parity checks (5.2) defining C_{BCH}, using the fact that the coefficients are in $\mathbb{F}_2$.*

3. *For $d = 3$, C_{BCH} is the same as another code we have seen. What is that code?*

4. *Define the subcode of C_{BCH} with a global parity check, i.e., the condition $c_1 + c_2 + \cdots + c_n = 0$ (over $\mathbb{F}_2$). Let d be an even integer. Show how to use the BCH code with a global parity check to construct a binary linear code of distance at least d and dimension at least $n - (d/2 - 1)\log_2(n+1) - 1$.*

5. *Conclude that for all n of the form $2^m - 1$ and integers d, $2 \le d < n/\log_2(n_1)$, one can construct an $[n, k', d']_2$ binary linear code with $d' \ge d$ and $k' \ge n - \left\lfloor \frac{d-1}{2} \right\rfloor \log_2(n+1) - 1$.*

6. *Prove that the $\left\lfloor \frac{d-1}{2} \right\rfloor$ factor cannot be any smaller.*
   Hint: *What does the Hamming bound say?*

Exercise 5.12. *In this exercise, we will consider BCH-like codes in the theme of Exercise 5.11, but applied to the GRS codes of Exercise 5.10. Consider the Generalized Reed-Solomon code $C_{\mathrm{GRS}} = \mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, k, \mathbf{v}]$ defined in (5.3) of dimension k and block length n over a field $\mathbb{F} = \mathbb{F}_{2^m}$. Now, define its binary intersection code $C^* := C_{\mathrm{GRS}} \cap \mathbb{F}_2^n$, which will be the object of study in this exercise.*

1. *Prove that C^* is a code of distance at least $d := n - k + 1$.*

2. *Prove that C^* is a binary linear code of rate at least $1 - \frac{(n-k)m}{n}$.*
   Hint: *How many parity checks are needed to define this code?*

3. *Let $\mathbf{c} \in \mathbb{F}_2^n$ be a nonzero binary vector. Prove that for every choice of the evaluation points sequence $\boldsymbol{\alpha}$ there are at most $(2^m - 1)^k$ choices of the vector $\mathbf{v}$ for which $\mathbf{c} \in C_{\mathrm{GRS}}$.*

4. *Using the above, prove that if the integer D satisfies $\mathrm{Vol}_2(n, D-1) < (2^m - 1)^{n-k}$ (where $\mathrm{Vol}_2(n, D-1) = \sum_{i=0}^{D-1} \binom{n}{i}$), then there exists a vector $\mathbf{v} \in (\mathbb{F}^*)^n$ such that the minimum distance of the binary code C^* is at least D.*

5. *Using parts 2 and 4 above, prove that the family of codes $\mathrm{GRS}_{\mathbb{F}}[\boldsymbol{\alpha}, k, \mathbf{v}] \cap \mathbb{F}_2^n$ contains binary linear codes that meet the Gilbert-Varshamov bound.*

Exercise 5.13. *Recall the definition of Hadamard codes from Section 2.6: the $[2^r, r, 2^{r-1}]_2$ Hadamard code is generated by the $r \times 2^r$ matrix whose ith (for $0 \leq i \leq 2^r - 1$) column is the binary representation of i. This exercise gives a polynomial view of Hadamard codes.*

Specifically, prove that the Hadamard codeword for the message $(m_1, m_2, \ldots, m_r) \in \{0,1\}^r$ is the evaluation of the (multivariate) polynomial $m_1X_1 + m_2X_2 + \cdots + m_rX_r$ (where $X_1, \ldots, X_r$ are the r variables) over all the possible assignments to the variables $(X_1, \ldots, X_r)$ from $\{0,1\}^r$.

Using the definition of Hadamard codes above (re)prove the fact that the code has distance 2^{r-1}.

Exercise 5.14. *Recall the definition of t-wise independence from Exercise 2.14, namely, a set $S \subseteq \mathbb{F}_q^n$ is said to be a t-wise independent source (for some $1 \leq t \leq n$) if for every $I \subseteq [n]$ with $|I| = t$, a uniformly random sample $(X_1, \ldots, X_n)$ from S satisfies the property that the variables $\{X_i | i \in I\}$ are uniform and independent over $\mathbb{F}_q$. (Note that such a sample can be obtained using $\log_2 |S|$ random bits.) We will explore properties of these objects in this exercise.*

1. *Let C be a linear code that does not have any coordinate that is 0 for every codeword. Prove that C is a 1-wise independent source.*

2. *Prove that every $[n,k]_q$ MDS code is a k-wise independent source but is not a $k+1$-wise independent source.*

3. *Using Part 2 or otherwise, prove that there exists a k-wise independent source $S \subseteq \mathbb{F}_q^m$ of size at most q^k for $q \geq m$. Now show how to pick q so that S can be viewed as a k-wise independent source in $\mathbb{F}_2^{m \log_2 q}$ of size at most $(2m)^k$. Finally set m and q as functions of n and k to show that $k \cdot (\log_2 n - \log_2 \log_2 n + O(1))$-random bits are enough to sample from a k-wise independent source over $\mathbb{F}_2^n$.*

4. *For $0 < p \leq 1/2$, we say the n binary random variables $X_1, \ldots, X_n$ are p-biased and t-wise independent if any of the t random variables are independent and $\Pr[X_i = 1] = p$ for every $i \in [n]$. For the rest of the problem, let p be a power of $1/2$. Then show that any $t \cdot \log_2(1/p)$-wise independent random variables can be converted into t-wise independent p-biased random variables. Conclude that one can construct such sources with $t \log_2(1/p)(1 + \log_2(n \log_2(1/p)))$ uniformly random bits. Then improve this bound to $t(1 + \max(\log_2(1/p), \log_2 n))$ uniformly random bits.*

Exercise 5.15. *In this exercise, we improve over the randomness used in Part 3 of Exercise 5.14 to sample from a k-wise independent source over $\mathbb{F}_2^n$, by nearly a factor of 2. Specifically, use Exercises 2.14 and 5.11 part 5 to prove the following: for every integers n, k with $1 \leq k \leq n$, at most $\lfloor \frac{k}{2} \rfloor \log_2(2n))$ random bits are enough to compute n-bits that are k-wise independent.*

Exercise 5.16. *In many applications, errors occur in "bursts"—i.e., all the error locations are contained in a contiguous region (think of a scratch on a DVD or disk). In this problem we will use how one can use Reed-Solomon codes to correct bursty errors.*

An error vector $\mathbf{e} \in \{0,1\}^n$ *is called a* t-single burst *error pattern if all the non-zero bits in* $\mathbf{e}$ *occur in the range* $[i, i+t-1]$ *for some* $1 \leq i \leq n = t+1$*. Further, a vector* $\mathbf{e} \in \{0,1\}^n$ *is called a* (s,t)*-burst error pattern if it is the union of at most* s t*-single burst error pattern (i.e. all non-zero bits in* $\mathbf{e}$ *are contained in one of at most* s *contiguous ranges in* $[n]$*).*

We call a binary code $C \subseteq \{0,1\}^n$ *to be* (s,t)-burst error correcting *if one can uniquely decode from any* (s,t)*-burst error pattern. More precisely, given an* (s,t)*-burst error pattern* $\mathbf{e}$ *and any codeword* $\mathbf{c} \in C$*, the only codeword* $\mathbf{c}' \in C$ *such that* $(\mathbf{c}+\mathbf{e}) - \mathbf{c}'$ *is an* (s,t)*-burst error pattern satisfies* $\mathbf{c}' = \mathbf{c}$*.*

1. *Prove that if* C *is* (st)*-error correcting (in the sense of Definition 1.3.5), then it is also* (s,t)*-burst error correcting. Conclude that for every* $\varepsilon > 0$*, there exists code with rate* $\Omega(\varepsilon^2)$ *and block length* n *that is* (s,t)*-burst error correcting for every* s, t *such that* $s \cdot t \leq \left(\frac{1}{4} - \varepsilon\right) \cdot n$*.*

2. *Prove that for every rate* $R > 0$ *and for large enough* n*, there exist* (s,t)*-burst error correcting as long as* $s \cdot t \leq \left(\frac{1-R-\varepsilon}{2}\right) \cdot n$ *and* $t \geq \Omega\left(\frac{\log n}{\varepsilon}\right)$*. In particular, one can correct from* $\frac{1}{2} - \varepsilon$ *fraction of burst-errors (as long as each burst is "long enough") with rate* $\Omega(\varepsilon)$ *(compare this with item 1).*

   Hint: *Use Reed-Solomon codes.*

Exercise 5.17. *In this problem, we will consider the number-theoretic counterpart of Reed-Solomon codes. Let* $1 \leq k < n$ *be integers and let* $p_1 < p_2 < \cdots < p_n$ *be* n *distinct primes. Denote* $K = \prod_{i=1}^{k} p_i$ *and* $N = \prod_{i=1}^{n} p_i$*. The notation* $\mathbb{Z}_M$ *stands for integers modulo* M*, i.e., the set* $\{0, 1, \ldots, M-1\}$*. Consider the* Chinese Remainder code *defined by the encoding map* $E : \mathbb{Z}_K \to \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \cdots \times \mathbb{Z}_{p_n}$ *defined by:*

$$E(m) = (m \mod p_1,\ m \mod p_2,\ \cdots,\ m \mod p_n)\ .$$

(Note that this is not a code in the usual sense we have been studying since the symbols at different positions belong to different alphabets. Still notions such as distance of this code make sense and are studied in the question below.)

Suppose that $m_1 \neq m_2$*. For* $1 \leq i \leq n$*, define the indicator variable* $b_i = 1$ *if* $E(m_1)_i \neq E(m_2)_i$ *and* $b_i = 0$ *otherwise. Prove that* $\prod_{i=1}^{n} p_i^{b_i} > N/K$*.*

Use the above to deduce that when $m_1 \neq m_2$*, the encodings* $E(m_1)$ *and* $E(m_2)$ *differ in at least* $n - k + 1$ *locations.*

Exercise 5.18. *In this problem, we will consider derivatives over a finite field* $\mathbb{F}_q$*. Unlike the case of derivatives over reals, derivatives over finite fields do not have any physical interpretation but as we shall see shortly, the notion of derivatives over finite fields is still a useful concept. In particular, given a polynomial* $f(X) = \sum_{i=0}^{t} f_i X^i$ *over* $\mathbb{F}_q$*, we define its derivative as*

$$f'(X) = \sum_{i=0}^{t-1} (i+1) \cdot f_{i+1} \cdot X^i.$$

Further, we will denote by $f^{(i)}(X)$*, the result of applying the derivative on* f i *times. In this problem, we record some useful facts about derivatives.*

1. *Define* $R(X,Z) = f(X+Z) = \sum_{i=0}^{t} r_i(X) \cdot Z^i$. *Then for every* $j \geq 1$,

$$f^{(j)}(X) = j! \cdot r_j(X).$$

2. *Using part 1 or otherwise, show that for every* $j \geq \mathrm{char}(\mathbb{F}_q)$,[8] $f^{(j)}(X) \equiv 0$.

3. *Let* $j < \mathrm{char}(\mathbb{F}_q)$. *Further, assume that for every* $0 \leq i < j$, $f^{(i)}(\alpha) = 0$ *for some* $\alpha \in \mathbb{F}_q$. *Then prove that* $(X-\alpha)^j$ *divides* $f(X)$.

4. *Finally, prove the following generalization of the degree mantra (Proposition 5.1.5). Let* $f(X)$ *be a non-zero polynomial of degree* t *and* $m \leq \mathrm{char}(\mathbb{F}_q)$. *Then there exists at most* $\left\lfloor \frac{t}{m} \right\rfloor$ *distinct elements* $\alpha \in \mathbb{F}_q$ *such that* $f^{(j)}(\alpha) = 0$ *for every* $0 \leq j < m$.

Exercise 5.19. *In this exercise, we will consider a code that is related to Reed-Solomon codes and uses derivatives from Exercise 5.18. These codes are called* derivative codes.

Let $m \geq 1$ *be an integer parameter and consider parameters* $k < \mathrm{char}(\mathbb{F}_q)$ *and* n *such that* $m < k < nm$. *Then the derivative code with parameters* (n,k,m) *is defined as follow. Consider any message* $\mathbf{m} \in \mathbb{F}_q^k$ *and let* $f_{\mathbf{m}}(X)$ *be the message polynomial as defined for the Reed-Solomon code. Let* $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ *be distinct elements. Then the codeword for* $\mathbf{m}$ *is given by*

$$\begin{pmatrix} f_{\mathbf{m}}(\alpha_1) & f_{\mathbf{m}}(\alpha_2) & \cdots & f_{\mathbf{m}}(\alpha_n) \\ f_{\mathbf{m}}^{(1)}(\alpha_1) & f_{\mathbf{m}}^{(1)}(\alpha_2) & \cdots & f_{\mathbf{m}}^{(1)}(\alpha_n) \\ \vdots & \vdots & \vdots & \vdots \\ f_{\mathbf{m}}^{(m-1)}(\alpha_1) & f_{\mathbf{m}}^{(m-1)}(\alpha_2) & \cdots & f_{\mathbf{m}}^{(m-1)}(\alpha_n) \end{pmatrix}.$$

1. *Prove that the above code is linear over* $\mathbb{F}_q$, *meaning that if* $c_1, c_2 \in (\mathbb{F}_q^m)^n$ *are codewords, then so is* $\alpha c_1 + \beta c_2$ *for all* $\alpha, \beta \in \mathbb{F}_q$. *Here we define* $\alpha \mathbf{v}$ *for* $\alpha \in \mathbb{F}_q$ *and* $\mathbf{v} \in \mathbb{F}_q^m$ *as multiplication of coordinates of* v *by* α, *and as usual* αc_1 *is componentwise multiplication of symbols of* c_1 *by* α.

2. *Prove that the above code has rate* $k/(nm)$ *and distance at least* $n - \left\lfloor \frac{k-1}{m} \right\rfloor$.

Exercise 5.20. *In this exercise, we will consider another code related to Reed-Solomon codes that are called* Folded Reed-Solomon *codes. We will see a lot more of these codes in Chapter* **??**.

Let $m \geq 1$ *be an integer parameter and let* $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ *are distinct elements such that for some element* $\gamma \in \mathbb{F}_q^*$, *the sets*

$$\{\alpha_i, \alpha_i\gamma, \alpha_i\gamma^2, \ldots, \alpha_i\gamma^{m-1}\}, \tag{5.4}$$

[8] $\mathrm{char}(\mathbb{F}_q)$ denotes the *characteristic* of $\mathbb{F}_q$. That is, if $q = p^s$ for some prime p, then $\mathrm{char}(\mathbb{F}_q) = p$. Any natural number i in $\mathbb{F}_q$ is equivalent to $i \mod \mathrm{char}(\mathbb{F}_q)$.

are pair-wise disjoint for different $i \in [n]$. Then the folded Reed-Solomon code with parameters $(m, k, n, \gamma, \alpha_1, \ldots, \alpha_n)$ is defined as follows. Consider any message $\mathbf{m} \in \mathbb{F}_q^k$ and let $f_{\mathbf{m}}(X)$ be the message polynomial as defined for the Reed-Solomon code. Then the codeword for $\mathbf{m}$ is given by:

$$\begin{pmatrix} f_{\mathbf{m}}(\alpha_1) & f_{\mathbf{m}}(\alpha_2) & \cdots & f_{\mathbf{m}}(\alpha_n) \\ f_{\mathbf{m}}(\alpha_1 \cdot \gamma) & f_{\mathbf{m}}(\alpha_2 \cdot \gamma) & \cdots & f_{\mathbf{m}}(\alpha_n \cdot \gamma) \\ \vdots & \vdots & \vdots & \vdots \\ f_{\mathbf{m}}(\alpha_1 \cdot \gamma^{m-1}) & f_{\mathbf{m}}(\alpha_2 \cdot \gamma^{m-1}) & \cdots & f_{\mathbf{m}}(\alpha_n \cdot \gamma^{m-1}) \end{pmatrix}.$$

Prove that the above code has rate $k/(nm)$ and distance at least $n - \left\lfloor \frac{k-1}{m} \right\rfloor$.

Exercise 5.21. *In this problem we will see that Reed-Solomon codes, derivative codes (Exercise 5.19) and folded Reed-Solomon codes (Exercise 5.20) are all essentially special cases of a large family of codes that are based on polynomials. We begin with the definition of these codes.*

Let $m \geq 1$ be an integer parameter and define $m < k \leq n$. Further, let $E_1(X), \ldots, E_n(X)$ be n polynomials over $\mathbb{F}_q$, each of degree m. Further, these polynomials pair-wise do not have any non-trivial factors (i.e. $\gcd(E_i(X), E_j(X))$ has degree 0 for every $i \neq j \in [n]$.) Consider any message $\mathbf{m} \in \mathbb{F}_q^k$ and let $f_{\mathbf{m}}(X)$ be the message polynomial as defined for the Reed-Solomon code. Then the codeword for $\mathbf{m}$ is given by:

$$(f_{\mathbf{m}}(X) \mod E_1(X), f_{\mathbf{m}}(X) \mod E_2(X), \ldots, f_{\mathbf{m}}(X) \mod E_n(X)).$$

In the above we think of $f_{\mathbf{m}}(X) \mod E_i(X)$ as an element of $\mathbb{F}_{q^m}$. In particular, given given a polynomial of degree at most $m-1$, we will consider any bijection between the q^m such polynomials and $\mathbb{F}_{q^m}$. We will first see that this code is MDS and then we will see why it contains Reed-Solomon and related codes as special cases.

1. *Prove that the above code has rate $k/(nm)$ and distance at least $n - \left\lfloor \frac{k-1}{m} \right\rfloor$.*

2. *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ be distinct elements. Define $E_i(X) = X - \alpha_i$. Prove that for this special case the above code (with $m = 1$) is the Reed-Solomon code.*

3. *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ be distinct elements. Define $E_i(X) = (X - \alpha_i)^m$. Prove that for this special case the above code is the derivative code (with an appropriate mapping from polynomials of degree at most $m-1$ and $\mathbb{F}_q^m$, where the mapping could be different for each $i \in [n]$ and can depend on $E_i(X)$).*

4. *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ be distinct elements and $\gamma \in \mathbb{F}_q^*$ such that (5.4) is satisfied. Define $E_i(X) = \prod_{j=0}^{m-1}(X - \alpha_i \cdot \gamma^j)$. Prove that for this special case the above code is the folded Reed-Solomon code (with an appropriate mapping from polynomials of degree at most $m-1$ and $\mathbb{F}_q^m$, where the mapping could be different for each $i \in [n]$ and can depend on $E_i(X)$).*

Exercise 5.22. *In this exercise we will develop a sufficient condition to determine the irreducibility of certain polynomials called the* Eisenstein's criterion.

Let $F(X,Y)$ be a polynomial of $\mathbb{F}_q$. Think of this polynomial as over X with coefficients as polynomials in Y over $\mathbb{F}_q$. Technically, we think of the coefficients as coming from the ring of polynomials in Y over $\mathbb{F}_q$. We will denote the ring of polynomials in Y over $\mathbb{F}_q$ as $\mathbb{F}_q(Y)$ and we will denote the polynomials in X with coefficients from $\mathbb{F}_q(Y)$ as $\mathbb{F}_q(Y)[X]$.

In particular, let

$$F(X,Y) = X^t + f_{t-1}(Y) \cdot X^{t-1} + \cdots + f_0(Y),$$

where each $f_i(Y) \in \mathbb{F}_q(Y)$. Let $P(Y)$ be a prime *for $\mathbb{F}_q(Y)$ (i.e. $P(Y)$ has degree at least one and if $P(Y)$ divides $A(Y) \cdot B(Y)$ then $P(Y)$ divides at least one of $A(Y)$ or $B(Y)$). If the following conditions hold:*

(i) $P(Y)$ divides $f_i(Y)$ for every $0 \leq i < t$; but

(ii) $P^2(Y)$ does not divide $f_0(Y)$

then $F(X,Y)$ does not have any non-trivial factors over $\mathbb{F}_q(Y)[X]$ (i.e. all factors have either degree t or 0 in X).

In the rest of the problem, we will prove this result in a sequence of steps:

1. *For the sake of contradiction assume that $F(X,Y) = G(X,Y) \cdot H(X,Y)$ where*

$$G(X,Y) = \sum_{i=0}^{t_1} g_i(Y) \cdot X^I \text{ and } H(X,Y) = \sum_{i=0}^{t_2} h_i(Y) \cdot X^i,$$

   where $0 < t_1, t_2 < t$. Then prove that $P(Y)$ does not divide both of $g_0(Y)$ and $h_0(Y)$.

   For the rest of the problem WLOG assume that $P(Y)$ divides $g_0(Y)$ (and hence does not divide $h_0(Y)$).

2. *Prove that there exists an i^* such that $P(Y)$ divide $g_i(Y)$ for every $0 \leq i < i^*$ but $P(Y)$ does not divide $g_{i^*}(Y)$ (define $g_t(Y) = 1$).*

3. *Prove that $P(Y)$ does not divide $f_i(Y)$. Conclude that $F(X,Y)$ does not have any non-trivial factors, as desired.*

Exercise 5.23. *We have mentioned objects called algebraic-geometric (AG) codes, that generalize Reed-Solomon codes and have some amazing properties: see for example, Section 4.6. The objective of this exercise is to construct one such AG code, and establish its rate vs distance trade-off.*

Let p be a prime and $q = p^2$. Consider the equation

$$Y^p + Y = X^{p+1} \tag{5.5}$$

over $\mathbb{F}_q$.

1. *Prove that there are exactly p^3 solutions in $\mathbb{F}_q \times \mathbb{F}_q$ to (5.5). That is, if $S \subseteq \mathbb{F}_q^2$ is defined as*
$$S = \left\{(\alpha, \beta) \in \mathbb{F}_q^2 \mid \beta^p + \beta = \alpha^{p+1}\right\}$$
then $|S| = p^3$.

2. *Prove that the polynomial $F(X, Y) = Y^p + Y - X^{p+1}$ is irreducible over $\mathbb{F}_q$.*
Hint: *Exercise 5.22 could be useful.*

3. *Let $n = p^3$. Consider the evaluation map $\mathrm{ev} : \mathbb{F}_q[X, Y] \to \mathbb{F}_q^n$ defined by*
$$\mathrm{ev}(f) = (f(\alpha, \beta) : (\alpha, \beta) \in S) \ .$$
Prove that if $f \neq 0$ and is not divisible by $Y^p + Y - X^{p+1}$, then $\mathrm{ev}(f)$ has Hamming weight at least $n - \deg(f)(p + 1)$, where $\deg(f)$ denotes the total *degree of f.*
Hint: *You are allowed to make use of* Bézout's theorem, *which states that if $f, g \in \mathbb{F}_q[X, Y]$ are nonzero polynomials* with no common factors, *then they have at most $\deg(f)\deg(g)$ common zeroes.*

4. *For an integer parameter $\ell \geq 1$, consider the set $\mathcal{F}_\ell$ of bivariate polynomials*
$$\mathcal{F}_\ell = \{f \in \mathbb{F}_q[X, Y] \mid \deg(f) \leq \ell, \deg_X(f) \leq p\}$$
where $\deg_X(f)$ denotes the degree of f in X.
Prove that $\mathcal{F}_\ell$ is an $\mathbb{F}_q$-linear space of dimension $(\ell + 1)(p + 1) - \frac{p(p+1)}{2}$.

5. *Consider the code $C \subseteq \mathbb{F}_q^n$ for $n = p^3$ defined by*
$$C = \{\mathrm{ev}(f) \mid f \in \mathcal{F}_\ell\} \ .$$
Prove that C is a linear code with minimum distance at least $n - \ell(p + 1)$.

6. *Deduce a construction of an $[n, k]_q$ code with distance $d \geq n - k + 1 - p(p - 1)/2$.*
(Note that Reed-Solomon codes have $d = n - k + 1$, whereas these codes are off by $p(p-1)/2$ from the Singleton bound. However they are much longer than Reed-Solomon codes, with a block length of $n = q^{3/2}$, and the deficiency from the Singleton bound is only $o(n)$.)

Exercise 5.24. *Since Reed-Solomon codes are linear codes, by Proposition 2.3.5, one can do error detection for Reed-Solomon codes in quadratic time. In this problem, we will see that one can design even more efficient error detection algorithm for Reed-Solomon codes. In particular, we will consider* data streaming algorithms *(see Section* **??** *for more motivation on this class of algorithms). A data stream algorithm makes a sequential pass on the input taking only poly-logarithmic time on each location in the input and uses only poly-logarithmic space. In this problem we show that there exists a* randomized *data stream algorithm to solve the error detection problem for Reed-Solomon codes. We do so by first defining a problem unrelated to Reed-Solomon codes that can be solved by a data stream algorithm. (The solution*

will actually use Reed-Solomon codes, but this use is accidental and unrelated to the goal of the second part.) In the second part of the problem we will solve the error-detection problem for Reed-Solomon codes in the data-streaming setting using the solution to the first part as a black-box.

1. *For a sequence* $\boldsymbol{\sigma} = ((i_1, \alpha_i), \ldots, (i_n, \alpha_n)) \in ([m] \times \mathbb{F}_q)^n$ *define* $\mathbf{y} = \mathbf{y}(\boldsymbol{\sigma}) \in \mathbb{F}_q^m$ *to be the vector given by* $y_\ell = \sum_{\{j \in [n] | i_j = \ell\}} \alpha_j$ *for* $\ell \in [m]$*. Give a randomized data stream algorithm that given as input a sequence* $\boldsymbol{\sigma} = ((i_1, \alpha_1), \ldots, (i_n, \alpha_n)) \in ([m] \times \mathbb{F}_q)^n$ *that outputs* 0 *if and only if* $\mathbf{y} = \mathbf{y}(\boldsymbol{\sigma}) = \mathbf{0}$*, with probability at least* $2/3$*. Your algorithm should take at most* $\mathrm{polylog}(q(m+n))$ *time per position of input* $\boldsymbol{\sigma}$ *and use at most* $O(\log q(m+n))$ *space. For simplicity, you can assume that given an integer* $t \geq 1$ *and prime power* q*, the algorithm has oracle access to an irreducible polynomial of degree* t *over* $\mathbb{F}_q$*.*

   Hint: *Instead of computing and storing the vector* $\mathbf{y}$*, you should compute* $E(\mathbf{y})_j$*, i.e., the* j*th coordinate of an appropriate error-correcting encoding function* $E : \mathbb{F}_q^\ell \to \mathbb{F}_q^L$*, where* $j \in [L]$ *is chosen uniformly at random. To ensure this coordinate of the encoding function can be computed quickly, you may use a Reed-Solomon code.*

2. *Given* $[q, k]_q$ *Reed-Solomon code* C *(i.e. with the evaluation points being* $\mathbb{F}_q$*), present a data stream algorithm for error detection of* C *with* $O(\log q)$ *space and* $\mathrm{polylog} q$ *time per position of the received word. The algorithm should work correctly with probability at least* $2/3$*. You should assume that the data stream algorithm has access to the values of* k *and* q *(and knows that* C *has* $\mathbb{F}_q$ *as its evaluation points).*

   Hint: *Part 1 and Exercise 5.8 should be helpful.*

5.5 Bibliographic Notes

Reed-Solomon codes were invented by Reed and Solomon [34] in the form described in Definition 5.2.1, i.e., as evaluations of polynomials. Later, Gorenstein and Zierler [18] showed that for specific choices of $\boldsymbol{\alpha}$, the resulting Reed-Solomon code is actually a "BCH code". (This is the connection explored in Exercise 5.9.) BCH codes were themselves discovered slightly earlier in the independent works of Bose and Ray-Chaudhuri [5] and Hocquenghem [22]. We note that the original definitions of BCH codes used the coefficients of polynomials to represent codewords (analogous to the alternate definition of Reed-Solomon codes in Exercise 5.9). The equivalent definition of these codes used in Exercise 5.11 as subcodes of Reed-Solomon codes, again uses the above mentioned connection from [18].

The Chinese Remainder Codes in Exercise 5.17 are due to Mandelbaum [30]. The Derivative Codes in Exercise 5.19 are due to Rosenbloom and Tsfasman [36]. They form an important subclass of Multiplicity Codes invented by Kopparty, Saraf and Yekhanin [26]. The Folded Reed-Solomon codes in Exercise 5.20 were introduced by Krachovsky [27] and highlighted by the work of Guruswami and Rudra [20]. Exercise 5.21 is based on the work of Guruswami and Kopparty [19].

Bibliography

[1] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992.

[2] M. Artin. *Algebra*. Prentice-Hall Of India Pvt. Limited, 1996.

[3] John Bather. A conversation with herman chernoff. *Statistical Science*, 11(4):335–350, 1996.

[4] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[5] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3:68–79, 1960.

[6] P.S. Bullen. *Handbook of Means and Their Inequalities*. Mathematics and Its Applications. Springer Netherlands, 2010.

[7] Donald G. Chandler, Eric P. Batterman, and Govind Shah. Hexagonal, information encoding article, process and system. *US Patent Number 4,874,936*, October 1989.

[8] C. L. Chen and M. Y. Hsiao. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. *IBM Journal of Research and Development*, 28(2):124–134, 1984.

[9] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[10] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23(4):493–507, December 1952.

[11] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., second edition edition, 2005.

[12] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[13] Todd Ebert, Wolfgang Merkle, and Heribert Vollmer. On the autoreducibility of random sequences. *SIAM J. Comput.*, 32(6):1542–1569, 2003.

[14] Peter Elias. Coding for two noisy channels. In Cherry, editor, *Informtion Theory*, pages 61–74. Butterworth, 1956.

[15] Paul Erdös. Some remarks on the theory of graphs. *Bulletin of the American Mathematical Society*, 53:292–294, 1947.

[16] E. N. Gilbert. A comparison of signalling alphabets. *Bell System Technical Journal*, 31:504–522, 1952.

[17] M. J. E. Golay. Notes on digital coding. *Proceedings of the IRE*, 37:657, 1949.

[18] Daniel Gorenstein and Neal Zierler. A class of error-correcting codes in p^m symbols. *Journal of the Society for Industrial and Applied Mathematics*, 9(2):207–214, 1961.

[19] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Comb.*, 36(2):161–185, 2016.

[20] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Trans. Inf. Theory*, 54(1):135–150, 2008.

[21] Richard W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29:147–160, April 1950.

[22] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres (Paris)*, 2:147–156, 1959.

[23] Tom Hholdt, J. H. van Lint, and Ruud Pellikaan. Algebraic geometry codes. In W. C. Huffamn V. S. Pless and R. A.Brualdi, editors, *Handbook of Coding Theory*. North Holland, 1998.

[24] Tao Jiang and Alexander Vardy. Asymptotic improvement of the gilbert-varshamov bound on the size of binary codes. *IEEE Trans. Inf. Theory*, 50(8):1655–1664, 2004.

[25] D. D. Joshi. A note on upper bounds for minimum distance codes. *Inf. Control.*, 1(3):289–295, 1958.

[26] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28:1–28:20, 2014.

[27] Victor Yu. Krachkovsky. Reed-solomon codes for correcting phased error bursts. *IEEE Trans. Inf. Theory*, 49(11):2975–2984, 2003.

[28] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their applications*. Cambridge University Press, Cambridge, MA, 1986.

[29] Florence Jessie MacWilliams. A theorem on the distribution of weights in a systematic code. *Bell Systems Technical Journal*, 42:79–94, 1963.

[30] David Mandelbaum. Error correction in residue arithmetic. *IEEE Transactions on Computers*, C-21(6):538–545, 1972.

[31] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[32] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[33] Larry L. Peterson and Bruce S. Davis. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, San Francisco, 1996.

[34] Irving S. Reed and Gustav Solomon. Polynomial codes over certain finite fields. *SIAM Journal on Applied Mathematics*, 8(2):300–304, 1960.

[35] Herbert Robbins. A remark on Stirling's formula. *Amer. Math. Monthly*, 62:26–29, 1955.

[36] M. Yu. Rosenbloom and M. A. Tsfasman. Codes for the m-metric. *Problemy Peredachi Informatsii*, 33(1):55–63, 1997.

[37] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[38] R. Singleton. Maximum distance q -nary codes. *Information Theory, IEEE Transactions on*, 10(2):116 – 118, apr 1964.

[39] David Slepian. A class of binary signaling alphabets. *The Bell System Technical Journal*, 35(1):203–234, 1956.

[40] Aimo Tietavainen. On the nonexistence theorems for perfect error-correcting codes. *SIAM Journal of Applied Mathematics*, 24(1):88–96, 1973.

[41] Jacobus H. van Lint. Nonexistence theorems for perfect error-correcting codes. In *Proceedings of the Symposium on Computers in Algebra and Number Theory*, pages 89–95, 1970.

[42] R. R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akadamii Nauk*, 117:739–741, 1957.

Appendix A

Some Useful Facts

A.1 Some Useful Inequalities

Recall that the binomial coefficient for integers $a \leq b$, defined as

$$\binom{b}{a} = \frac{b!}{a!(b-a)!}.$$

We begin with a simple lower bound on the binomial coefficient:

Lemma A.1.1. *For all integers $1 \leq a \leq b$, we have*

$$\binom{b}{a} \geq \left(\frac{b}{a}\right)^a.$$

Proof. The following sequence of relations completes the proof:

$$\binom{b}{a} = \prod_{i=0}^{a-1} \frac{b-i}{a-i} \geq \prod_{i=0}^{a-1} \frac{b}{a} = \left(\frac{b}{a}\right)^a.$$

In the above, the first equality follows from definition and the inequality is true since $b \geq a$ and $i \geq 0$. □

We state the next set of inequalities without proof (see [35] for a proof):

Lemma A.1.2 (Stirling's Approximation)**.** *For every integer $n \geq 1$, we have*

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\lambda_1(n)} < n! < \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\lambda_2(n)},$$

where

$$\lambda_1(n) = \frac{1}{12n+1} \text{ and } \lambda_2(n) = \frac{1}{12n}.$$

We prove another inequality involving Binomial coefficient.

Lemma A.1.3. *For all integers $1 \leq a \leq b$, we have*

$$\binom{b}{a} \leq \left(\frac{eb}{a}\right)^a.$$

Proof. First note that

$$\binom{b}{a} = \frac{b(b-1)\cdots(b-a+1)}{a!} \leq \frac{b^a}{a!}.$$

The final bound follows from the fact that

$$a! > \left(\frac{a}{e}\right)^a,$$

which in turns follows from the following relationships:

$$\frac{a^a}{a!} < \sum_{i=0}^{\infty} \frac{a^i}{i!} = e^a.$$

□

We next state Bernoulli's inequality:

Lemma A.1.4 (Bernoulli's Inequality)**.** *For every real numbers $k \geq 1$ and $x \geq -1$, we have*

$$(1+x)^k \geq 1 + kx.$$

Proof Sketch. We only present the proof for integer k. For the full proof see e.g. [6].

For the base case of $k = 1$, the inequality holds trivially. Assume that the inequality holds for some integer $k \geq 1$ and to complete the proof, we will prove it for $k+1$. Now consider the following inequalities:

$$\begin{aligned}(1+x)^{k+1} &= (1+x)\cdot(1+x)^k \\ &\geq (1+x)\cdot(1+kx) \\ &= 1 + (k+1)x + kx^2 \\ &\geq 1 + (k+1)x,\end{aligned}$$

as desired. In the above, the first inequality follows from the inductive hypothesis and the second inequality follows from the fact that $k \geq 1$. □

Lemma A.1.5. *For $|X| \leq 1$,*

$$\sqrt{1+x} \leq 1 + \frac{x}{2} - \frac{x^2}{16}.$$

Proof. Squaring the RHS we get

$$\left(1+\frac{x}{2}-\frac{x^2}{16}\right)^2 = 1+\frac{x^2}{4}+\frac{x^4}{256}+x-\frac{x^2}{16}-\frac{x^3}{32} = 1+x+\frac{3x^2}{16}-\frac{x^3}{32}+\frac{x^4}{256} \geq 1+x,$$

as desired. □

We will also use the Cauchy-Schwarz inequality:

Lemma A.1.6. *For any vector $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have*

$$|\langle \mathbf{x}, \mathbf{z} \rangle| \leq \|\mathbf{x}\|_2 \cdot \|\mathbf{z}\|_2.$$

A.2 Some Useful Identities and Bounds

We start off with an equivalence between two inequalities.

Lemma A.2.1. *Let $a, b, c, d > 0$. Then $\frac{a}{b} \leq \frac{c}{d}$ if and only if $\frac{a}{a+b} \leq \frac{c}{c+d}$.*

Proof. Note that $\frac{a}{b} \leq \frac{c}{d}$ if and only if

$$\frac{b}{a} \geq \frac{d}{c}.$$

The above is true if and only if

$$\frac{b}{a}+1 \geq \frac{d}{c}+1,$$

which is same as $\frac{a}{a+b} \leq \frac{c}{c+d}$. □

Next, we state some infinite sums that are identical to certain logarithms (the proofs are standard and are omitted).

Lemma A.2.2. *For $|x| < 1$,*

$$\ln(1+x) = x - \frac{x^2}{2!} + \frac{x^3}{3!} - \cdots.$$

We can use the above to prove some bounds on $\ln(1+x)$ (we omit the proof):

Lemma A.2.3. *For $0 \leq x < 1$, we have*

$$x - x^2/2 \leq \ln(1+x) \leq x,$$

and for $0 \leq x \leq 1/2$, we have

$$-x - x^2 \leq \ln(1-x) \leq -x.$$

We can use the above bounds to further prove boounds on the (binary) entropy function:

Lemma A.2.4. *For $x \leq 1/4$, we have*

$$1 - 5x^2 \leq H(1/2 - x) \leq 1 - x^2.$$

Proof. By definition $H(1/2 - x) = 1 - 1/2 \log(1 - 4x^2) + x \log(1 - 2x)/(1 + 2x)$, and using the approximations for $\ln(1 + x)$ from Lemma A.2.3, we have, for $x < 1/4$,

$$\begin{aligned} H(1/2 - x) &\leq 1 + \frac{1}{2 \ln 2} \cdot (4x^2 + 16x^4) + \frac{1}{\ln 2} \cdot (-2x^2) - \frac{1}{\ln 2} \cdot (2x^2 - 2x^3) \\ &= 1 - \frac{2}{\ln 2} \cdot x^2 + \frac{2}{\ln 2} \cdot x^3 + \frac{8}{\ln 2} \cdot x^4 \\ &\leq 1 - \frac{x^2}{\ln 2} \qquad \text{(A.1)} \\ &\leq 1 - x^2. \end{aligned}$$

In the above, (A.1) follows by using our assumption that $x \leq 1/4$.

Using the other sides of the approximations we also have:

$$\begin{aligned} H(1/2 - x) &\geq 1 + \frac{1}{2 \ln 2} \cdot (4x^2) + \frac{1}{\ln 2} \cdot (-2x^2 - 4x^3) - \frac{1}{\ln 2} \cdot (2x^2) \\ &\geq 1 - \frac{3x^2}{\ln 2} \\ &\geq 1 - 5x^2, \end{aligned}$$

where the second inequality uses our assumption that $x \leq 1/4$. □

The following fact follows from the well-known fact that $\lim_{x \to \infty} (1 + 1/x)^x = e$:

Lemma A.2.5. *For every real $x > 0$,*

$$\left(1 + \frac{1}{x}\right)^x \leq e.$$