Lecture 8 of the Theoretician's Toolkit

Error-Correcting Codes

January 19, 2010

Lecturer: Johan Håstad
Scribes: Karl Palmskog and Cenny Wenner
Notes updated: 2010-02-10

1 Motivation

If a message is sent in plain text, any single error may prevent us from recovering the original message. Such as sending "cat" and receiving "rat". To circumvent this, redundant bits of information can be introduced to identify and correct errors. A simple example of encoding for this purpose would be to send each symbol thrice. To decode a received message (possibly with errors), one could look at every group of three symbols and taking it to mean whichever symbol appears the most often. For example, if we wish to send "cat", we actually send "cccaaattt". If the received message reads "rccaaattt", the decoding scheme successfully recovers "cat" since the c's still outnumber the r's.

This scheme is able to *correct a single error* at the expense of blowing up the message size by a factor 3, or, in other words, its *information rate* is 1/3. This coding scheme is *efficient*; we can easily implement this encoding and decoding scheme in, e.g., time polynomial in the original message length. In this lecture, we will see how to efficiently correct an arbitrary, but fixed, number of errors with considerably better rate.

2 Introduction

Let Σ be an alphabet and k a natural number. We wish to transfer a message $m \in \Sigma^k$ from a point A to a point B over an error-prone channel. Errors in transferred data manifest as symbols at certain positions being changed into other symbols (which we assume are also in Σ). We want to, on one hand, *detect* when an error has occurred during the transfer of m, and on the other hand, be able to *correct* an error when it is known to have occurred. To achieve this, we map messages to *codewords* in Σ^n, with n a natural number greater than k, before transferring them over the channel. A depiction of our scenario, where we let the mapping $f\colon \Sigma^k \to \Sigma^n$, can be seen in Figure 1.

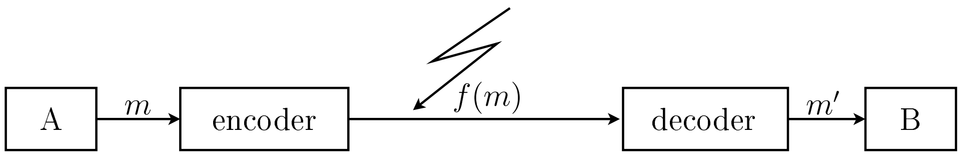

Figure 1: Scenario for Coding

Suppose we only transfer messages in the set $M \subseteq \Sigma^k$ and let the set of codewords be $\mathcal{C} \subseteq \Sigma^n$. The intuition for why it is beneficial to transfer $f(m) \in \mathcal{C}$ instead of simply m is that errors in $f(m)$ more easily result in the codeword not being in $\mathcal{C}$ (which should be easy to check) than errors in m result in a message not being in M.

3 Error-Correcting Codes and Distance

A code $\mathcal{C}$ of length n is a subset of the language Σ^n of length-n strings from an alphabet Σ. A word $\mathbf{w} \in \Sigma^n$ is called a codeword if and only if it is in the code. For two codewords $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, we define their distance as the Hamming distance: the number of coordinates the words' symbols differ at, i.e.

$$d(\mathbf{x}, \mathbf{y}) = |\{1 \leq i \leq n \mid x_i \neq y_i\}|.$$

For a codeword $\mathbf{w} \in \Sigma^n$, the *(minimal) distance to the code*, $d(\mathbf{w}, \mathcal{C})$, is $\min_{\mathbf{x} \in \mathcal{C}} d(\mathbf{w}, \mathbf{x})$. The *(minimal) distance of the code*, is the minimal distance between two codewords,

$$\Delta(\mathcal{C}) = \min_{\mathbf{x}, \mathbf{y} \in \mathcal{C}} d(\mathbf{x}, \mathbf{y}).$$

In Hamming's model, we consider the case when up to some e number of errors may be introduced in the sent message and we wish to be able to retrieve the original message regardless of what these errors are. If we are able to do this, possibly under resource constraints, then we say that the code can *correct e errors.* To avoid degenerate cases, we shall assume $|\mathcal{C}| \geq 2$ and $n > e$.

Lemma 1. *For a code to correct e errors, a minimal distance of at least $2e + 1$ is necessary and sufficient.*

Proof. We note that a word $\mathbf{w}$ is uniquely decodable if and only if it is at distance at most e to a unique codeword. We also require all words at distance at most e to the code to be uniquely decodable (these are the possible received words). If it does not hold, $\mathbf{w}$ must be at a distance at most e to two distinct codewords, making the minimum distance of the code at most $e + e = 2e$. In the other direction, if it does hold, there cannot be two codewords at distance $2e$ since this would imply the existence of a word at distance e to two distinct codewords. □

4 Hamming Balls

A *Hamming Ball* of radius e centered at a point $\mathbf{x}$ is the set of points whose Hamming distance to $\mathbf{x}$ is at most e. In our coding terminology, it is precisely the set of words we may receive if up to e symbols are changed in the sent message, $\mathbf{x}$. One characterization of $\mathcal{C}$, correcting e errors, is as balls covering the space Σ^n as shown projected onto the plane in Figure 2; the centers are the codewords to which every sequence of symbols inside the ball is corrected. Lemma 1 implies that the distance l between two centers is at least $2e + 1$.

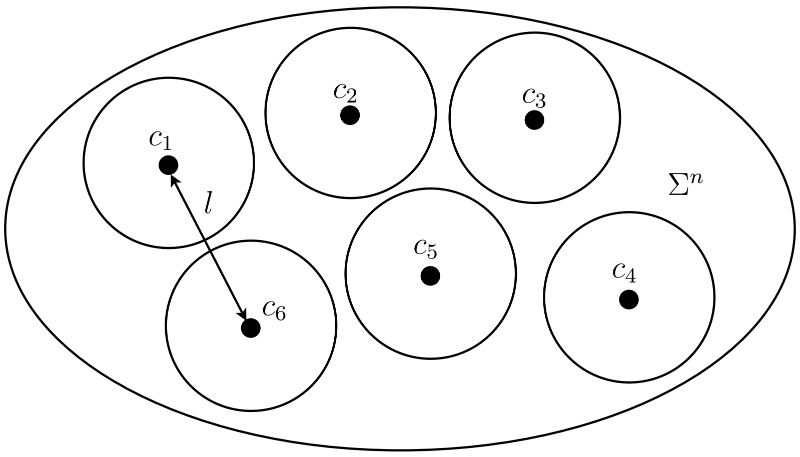

Figure 2: Representation of a Code as Balls

If we let $\Sigma = \{0, 1\}$ and suppose there are 2^k messages, the size of each ball is in fact $1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{e}$, as given by the following lemma.

Lemma 2. *For a code $\mathcal{C}$ over the binary alphabet such that $|\mathcal{C}| = 2^k$,*

$$2^k \times \left(1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{e}\right) \leq 2^n.$$

Proof. Let $\mathbf{x}$ and $\mathbf{y}$ be distinct codewords in $\mathcal{C}$. Denote by $S_e(\mathbf{x})$ and $S_e(\mathbf{y})$ the set of binary words that can be obtained by altering up to e bits in each of these codewords. Adding up the possibilities for altering $0, 1, \ldots, e$ bits we get

$$|S_e(\mathbf{x})| = |S_e(\mathbf{y})| = 1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{e}.$$

Since the codewords are distinct, $S_e(\mathbf{x})$ must be disjoint from $S_e(\mathbf{y})$, whereby our result follows. □

For the code which corrects only one error, Lemma 2 amounts to

$$k \leq n - \lceil \log(n+1) \rceil.$$

5 Linear Codes

Let $(\Sigma, +, \cdot)$ be a finite field and $\mathcal{C}$ a code on Σ for messages of length k. We say that $\mathcal{C}$ is *linear* if $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ implies $\mathbf{x} + \mathbf{y} \in \mathcal{C}$. This means that $\mathcal{C}$ is a linear space of dimension k spanned by k linearly independent vectors. By arranging these vectors into columns we form the *generator matrix* G (of size $k \times n$) for $\mathcal{C}$, which allows us to obtain a codeword $\mathbf{w}$ from the message $\mathbf{b}$ through the equation

$$\mathbf{w} = \mathbf{b}G^T.$$

Consider the set $\mathcal{C}^\perp$ of all vectors orthogonal to $\mathcal{C}$, which by definition is also a linear code. We call $\mathcal{C}^\perp$ the *dual* code to $\mathcal{C}$, and note that if H is a generating matrix (of size $(n-k) \times n$) for $\mathcal{C}^\perp$, it must be the case that

$$H\mathbf{w}^T = \mathbf{0}^T \iff \mathbf{w} \in \mathcal{C},$$

with $\mathbf{0}^T$ a column vector of zeroes. We refer to H as a *parity-check matrix* for $\mathcal{C}$, since it provides a method to check code membership.

Example 1. Let $n = 7$ and $k = 4$. We are given the binary codeword $\mathbf{w} = (1, 1, 0, 1, 1, 0, 0)$ and the parity-check matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Since $H\mathbf{w}^T = [\,0\;1\;0\,]^T \neq \mathbf{0}^T$, $\mathbf{w}$ does not belong to the code defined by H.

6 Hamming Codes

For a codeword $\mathbf{w}$ in a binary code $\mathcal{C}$ we define the *weight* of $\mathbf{w}$ as the number of 1's $\mathbf{w}$ contains. Denoting weight by w, it follows that for two binary codewords $\mathbf{x}$ and $\mathbf{y}$,

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

whence we define the *minimum weight* for the code as $w_{\min} = \min_{\mathbf{w} \in C} w(\mathbf{w})$ for $\mathbf{w} \neq \mathbf{0}$ and conclude that $\Delta(\mathcal{C}) = w_{\min}$.

Consider again codes as balls in the space Σ^n as in Figure 2. We call a code *perfect* if its balls are disjoint and all have equal radius, while still filling up the space. We define a binary *Hamming code* as a linear code with a parity-check matrix in which all columns are distinct and no column consists of only zeroes. Such codes are interesting by virtue of the following results.

Lemma 3. *Binary Hamming codes can correct one error.*

Proof. By Lemma 1, the theorem follows if $w_{\min} \geq 3$. Suppose $w(\mathbf{w}) = 1$ for some $\mathbf{w} \in \mathcal{C}$. Then we will have $H\mathbf{w}^T = \mathbf{h}$, with $\mathbf{h}$ a column of H. By definition of Hamming codes, $\mathbf{h}$ cannot be all-zero, but we still have $\mathbf{h} = \mathbf{0}'$, yielding a contradiction. Suppose $w(\mathbf{w}) = 2$; then $H\mathbf{w}^T = \mathbf{h} + \mathbf{h}'$ with $\mathbf{h}$ and $\mathbf{h}'$ different columns of H. The parity-check equation gives $\mathbf{h} = \mathbf{h}'$, while Hamming codes require distinct columns, yielding the final contradiction. $\square$

Theorem 1. *Binary Hamming codes are perfect.*

Proof. Let r be the number of rows in the check matrix H. Note that we then must have $n = 2^r - 1$ and $k = 2^r - 1 - r$, since each column is a distinct non-zero r-bit binary number. It suffices to show that $\{0,1\}^n$ is completely covered by Hamming balls of radius 1 centered at the codewords. From Lemma 3, we know that any two such balls are disjoint and hence the total number of covered words is the number of balls times the size of a radius-1 ball. Hamming balls of radius 1 have size $n + 1$ by Lemma 2, making the number of covered words $2^k \times (n + 1) = 2^n$. Hence all of $\{0,1\}^n$ must be covered and the code is therefore perfect. $\square$

7 Reed-Solomon Codes

The Reed-Solomon code is a special linear code that encodes messages as polynomials and use that two distinct degree-d polynomials only can agree on at most d points. The benefit of these codes is that for every choice of the maximum number of errors, we can construct an error-correcting code with optimal rate. The downside is that we require the alphabet to be of size proportional to the size of the messages plus the number of errors. In other words, we cannot make the alphabet independent of the length of the message. How to map the codes to a typical binary alphabet will be covered in the next lecture, at a cost of the code's rate.

Definition 1 (Reed-Solomon Code)**.** Assume $|\Sigma| \geq n$ and let $\alpha_1, \dots, \alpha_n$ be arbitrary distinct points in Σ. The *Reed-Solomon Code* (w.r.t. k, Σ and $\{\alpha_i\}_i$) of a message $\mathbf{b} \in \Sigma^k$ is $(p_{\mathbf{b}}(\alpha_1), p_{\mathbf{b}}(\alpha_2), \dots, p_{\mathbf{b}}(\alpha_n))$ where $p_{\mathbf{b}}(\alpha)$ is the polynomial $b_1 + b_2x^1 + b_3x^2 + \cdots + b_kx^{k-1}$.

The Reed-Solomon code for a particular k and n is also denoted $\mathrm{RS}(k, n)$.

Example 2. Let us work in $\Sigma = \mathbb{F}_5$ with $k = 2, n = 5$ and $\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 2, \alpha_4 = 3, \alpha_5 = 4$. In other words, we are going to encode two symbols using five. Let us consider the three messages "00", "13", and "32". The polynomial for these three messages will be, respectively, $p_{00}(\alpha) = 0$, $p_{13}(\alpha) = 1 + 3\alpha$, and $p_{32}(\alpha) = 3 + 2\alpha$. The messages encoded, $(p(\alpha_1), \dots, p(\alpha_5))$, will be $(0, 0, 0, 0, 0)$ for the message "00", $(1, 4, 2, 0, 3)$ for the message "13", and $(3, 0, 2, 4, 1)$ for the message "32". One can show that this encoding can in fact correct a single error.

Theorem 2. *Reed-Solomon codes are linear.*

Proof. $\mathcal{C}$ consists of all degree-k polynomials and the sum of two degree-k polynomials is also a degree-k polynomial. □

Theorem 3. *The minimum distance of a Reed-Solomon code is $n - k + 1$.*

Proof. Since two different codewords of a Reed-Solomon Code must represent two different messages and thereby have been produced by two different polynomials of degree $k - 1$, the two codewords can agree on at most $k - 1$ distinct points. Since the encoded messages are merely the evaluation of the respective polynomials at n distinct points, the two codewords must differ on at least $n - k + 1$ points. This is furthermore strict since every degree-k polynomial produces a codeword. □

Corollary 1. *A Reed-Solomon code can correct up to $(n - k)/2$ errors.*

In the next section, we will describe how to do so efficiently.

7.1 Decoding Reed-Solomon

Let $\mathbf{b}$ be the original message, $\mathbf{x} = C(\mathbf{b}) = (p_{\mathbf{b}}(\alpha_i))_i$ its encoding and $\mathbf{y}$ the received message. We are guaranteed that $y_i = p_{\mathbf{b}}(\alpha_i)$ for all but e of the values of i and wish to recover $\mathbf{b}$.

We consider a degree-e polynomial N that zeroes the points for which $y_i \neq p_{\mathbf{b}}(\alpha_i)$ (and possibly some arbitrary points if fewer than e errors). One choice of N is to let Z be a set of cardinality e containing all the error locations and define $N(x) = \prod_{x_0 \in Z}(x - x_0)$.

We form a degree-$(k + e - 1)$ polynomial $R = Np_{\mathrm{b}}$ and note that for every $i = 1, \ldots, n$, $N(\alpha_i)y_i = R(y_i)$ since $p_{\mathbf{b}}(\alpha_i) = y_i$ whenever y_i was properly transmitted and otherwise N makes both sides 0. This may be interpreted as a system of linear equations with the coefficients of R and N as unknowns. If we could recover these unknowns, we could get $p_{\mathbf{b}}$ through polynomial division: $p_{\mathbf{b}} = R/N$.

By construction, we know that the system has a solution so we are done if we can argue that any solution to the system, giving polynomials N' and R', in fact entails $N'p_{\mathbf{b}} = R'$. Since $N'p_{\mathbf{b}}$ and R' are degree-$(k + e - 1)$ polynomials, it suffices to show that they agree on at least $k + e$ points to claim that the polynomials are identically equal. For the $n - e$ points which were sent correctly, i.e. $y_i = p_{\mathbf{b}}(\alpha_i)$, the equations of the linear system, $N(\alpha_i)y_i = R(\alpha_i)$, gives $N'(\alpha_i)p_{\mathbf{b}}(\alpha_i) = R'(\alpha_i)$ and the polynomials agree. Hence, as long as

$$k + e \leq n - e \iff e \leq (n - k)/2,$$

we can find $p_{\mathbf{b}}$.

7.2 Complexity of Reed-Solomon

The polynomial evaluation of the coding procedure as described can be done in $\mathcal{O}(n^2)$ time in the unit-cost model. The decoding procedure is essentially Gaussian elimination plus a polynomial division. Since we have up to n equations and n unknowns, the Gaussian elimination can be done in $\mathcal{O}(n^3)$. Polynomial long division demands up to $e + 1$ iterations, each of cost $\mathcal{O}(n)$.

8 Applications

The Reed-Solomon codes that will just covered are widely used for error-correction in storage and transmission in our modern society. They allow us to, e.g., read CDs, DVDs, and Blu-Rays with a serious errors in the form of scratches. Other applications include RAID, DSL, and satellite transmission.

9 Further Reading

The book *Algebraic Codes for Data Transmission* [Bla03] covered most of the material of the lecture and more. Section 3 covers the definition of and the properties of linear codes. The Reed-Solomon code is treated in Sections 6 and 7 but reasons differently about how to decode Reed-Solomon.

The introductory sections of Chapter 24 in *Discrete Mathematics* [Big02] gives a motivation for error-correcting codes and concisely covers linear codes, although not Reed-Solomon codes.

An in-depth treatment of the computational complexity for arithmetic on polynomials can be found in *Prime Numbers: A Computational Perspective* [CP05], Section 9.6.

For more applications of Reed-Solomon codes, see Wikipedia [Wik10].

References

[Big02] Norman L. Biggs. *Discrete Mathematics.* Oxford University Press, second edition, 2002.

[Bla03] Richard E. Blahut. *Algebraic Codes for Data Transmission.* Cambridge University Press, 2003.

[CP05] Richard Crandall and Carl Pomerance. *Prime Numbers: A Computational Perspective.* Springer, second edition, 2005.

[Wik10] Wikipedia. Reed-Solomon error correction, 2010 (retrieved February 10, 2010). `http://en.wikipedia.org/wiki/Reed-Solomon_error_correction`.