CS681	Computational Number Theory

Lecture 13: Codes: An Introduction

Instructor: Piyush P Kurur *Scribe: Ramprasad Saptharishi*

Overview

Over the next few lectures we shall be looking at codes, linear codes in particular. In this class we shall look at the motivation and a glimpse at error detection. The details shall be done in the lectures to come.

1 Codes

Suppose you have two parties Alice and Bob who wish to communicate over a channel which could potentially be unreliable. What we mean by unreliable is that some parts of the message could be corrupt or changed. We want to make sure that the recepient can detect such corruption if any, or sometimes even recover the message from the corrupted.

We can assume that the channel sends allows sending some strings over a fixed finite alphabet (a finite field, or bits, or the english alphabet etc). The two questions we need to address here is detection and correction.

1.1 Block Codes

What if Alice needs to send a message, in say english, and the channel has a different alphabet, say binary strings. Then we need some way of converting strings of one alphabet into another. This is achieved by looking at blocks of code.

In the example of english to binary, we could look at ascii codes. Each letter would correspond to a *block* of letters in the channel.

Of course, not all blocks of bits could correspond to meaningful sentences or messages. A block code is in general just a subset of strings. To formally define it:

Definition 1. *Let Σ be the fixed finite alphabet for the channel of communication. A block code $\mathcal{C}$ of length n over this communication is a subset of Σ^n.*

Elements of $\mathcal{C}$ are called code words.

Definition 2. *For any two strings x and y of the same length, the hamming distance is defined as the number of indices that x and y differ in.*

$$d(x, y) = |\{i \ : \ x_i \neq y_i\}|$$

Definition 3. *The distance of a code $\mathcal{C}$ is the minimum distance between its codewords. That is,*

$$d(\mathcal{C}) = \min_{x \neq y} d(x, y)$$

In a sense, the distance of a code is a measure of how much one needs to change to alter one code to another. For example, if the distance of a code was say 5, then it means that there are two strings (messages) x and y that differ at just 5 places. Now suppose x was sent through the channel and those 5 bits were changed due to the noise in the channel, then Bob would receive the message y from Alice while she had sent x. And since y was also a message, Bob could completely misinterpret Alice's message.

We would like codes to have large distance so that it takes a lot of corruption to actually alter one codeword into another. From this, we have a simple observation.

Observation 1. *Let d be the distance of a code $\mathcal{C}$. Then the code is $d-1$-detectable, or if the channel corrupts atmost $d-1$ letters of the message, then the other party can detect that the code word has been corrupted.*

Proof. As we remarked earlier, since the distance is d, it takes atleast d corruptions to change one code word into another. Therefore, on any code word x, something less than d corruptions cannot change it to another codeword. Therefore, if the string Bob received was a codeword, then he knows for sure that Alice had sent that string for sure. □

Therefore, if the channel changed atmost t bits, any code with distance atleast $t + 1$ would allow error detection. But of course, if Bob received "I hobe you", he knows that the message was corrupted but he has no way of determining whether the message was "I love you" or "I hate you". In order to correct t errors, you need a more than just a distance of $t + 1$.

Observation 2. *If $\mathcal{C}$ is a code of distance atleast $2t + 1$, then any message that is corrupt by atmost t bits can be recovered. Or in other words, the code is t-correctable.*

Suppose Alice had sent some codeword x and let us say this was altered through the channel and Bob received it as y. Given that atmost t bits were altered, we want to show that Bob can infact recover the message.

Since Bob knew that atmost t bits are corrupted, he looks at all codewords at a hamming distance of atmost t from y.[1] Now clearly x is a codeword that is present at a hamming distance at most t from y. If x was the only such code word, then Bob knows for sure that the message Alice sent has to be x.

But it must be the case that x is the only such codeword. Suppose not, say there was some other codeword $x' \neq x$ at a distance atmost t from y. Now since x and y differ at most t places and y and x' at atmost t, by the triangle inequality x and x' differ at atmost $2t$ places. But this contradicts the assumption that the distance of the code is atleast $2t + 1$.

Or in other words, if you want to move from one codeword to another through corruption, you need to corrupt $2t + 1$ bits atleast. And therefore if you corrupt just t place, you are definitely closer to where you started from than any other codeword.

But of course, it does not make sense to look at all code words of distances less than t from y to decode. Even besides that, how do we even figure out if a given word is a code word or not. So two important properties that we would want the code to have is efficient detection of codewords and effecient decoding.

2 Linear Codes

Recall that a block code $\mathcal{C}$ is an arbitrary subset of Σ^n. These codes could have no structure underlying them and that inherently makes detecting if a string is a codeword hard. Hence comes the idea for linear codes.

Since our alphabet is finite, we shall assume that the alphabet is infact a finite field $\mathbb{F}_q$. Now our space is $\mathbb{F}_q^n$ which is infact a vector space over $\mathbb{F}_q$ of dimension n. Instead of looking at arbitrary subsets of this space, linear codes restrict themselves to subspaces of $\mathbb{F}_q^n$.

Definition 4. *A $[n, k, d]_q$ linear code $\mathcal{C}$ such that $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$ and has distance d.*
That is, if $x, y \in \mathcal{C}$ then so is $\alpha x + \beta y$ for all $\alpha, \beta \in \mathbb{F}_q$.

To intuitively understand the parameters, we would be encoding messages of length k with codes of length n so that it can error-correct up to $d/2$ errors. Thus we want k to be as close to n as possible and also try to

[1] can be thought of as putting a ball of radius t around y

make d large. It is not possible to arbitrarily increase both but we want some reasonable values.

To see an example of a linear code, consider our field to be $\mathbb{F}_2$. Then if $\mathcal{C} = \{(0,0,0),(1,1,1)\}$ then this is a $[3,1,3]_2$ linear code.

Definition 5. *The weight of any string x is the number of indices of x that have a 1 in it.*

$$wt(x) = |\{i \ : \ x_i = 1\}|$$

Then we have the following easy observation.

Observation 3. *If $\mathcal{C}$ is a linear code, then*

$$d(\mathcal{C}) = \min_{x \neq 0} wt(x)$$

Proof. By definition, $d(\mathcal{C}) = d(x,y)$ but $d(x,y) = wt(x-y)$. Note that since we are looking at a linear code, $x, y \in \mathcal{C}$ also tells us that $x - y \in \mathcal{C}$. Therefore, for every $x \neq y$ we have a corresponding $0 \neq z = x - y$ that is a codeword whose weight is exactly the distance between x and y. □

2.1 Detection

Suppose we have a linear code $\mathcal{C}$ and given a string x we want to check if this is in the code or not. Since we know that our code is a subspace of $\mathbb{F}_q^n$, we can represent $\mathcal{C}$ using a basis. Let us say we are looking at $[n,k,\_]_q$ codes and our basis be $\{b_1, b_2, \cdots, b_k\}$ where each $b_i \in \mathbb{F}_q^n$.

The idea is that we want to construct a matrix H such that $Hx = \bar{0}$ if and only if $x \in \mathcal{C}$. Thus, in terms of tranformations, we want a linear map H such that the kernel of this map is precisely (nothing more, nothing less) $\mathcal{C}$. The question is, how do we find such a matrix?

We first find a transformation that achieves what we want and then try and figure out what the matrix of the transformation should look like. We have a basis $\{b_1, b_2, \cdots, b_k\}$ for our code. Let us extend this first to a basis $\{b_1, b_2, \cdots, b_n\}$ of $\mathbb{F}_q^n$.

Thus, every vector $v \in \mathbb{F}_q^n$ can be written as a unique linear combination of b_is. We do the obvious transformation: map all b_i where $i \leq k$ to 0 and the rest of the basis elements to identity.

$$\begin{aligned} v &= \alpha_1 b_1 + \alpha_2 b_2 + \cdots + \alpha_k b_k + \alpha_{k+1} b_{k+1} + \cdots + \alpha_n b_n \\ Hv &= \alpha_{k+1} b_{k+1} + \cdots + \alpha_n b_n \end{aligned}$$

Now a vector v will be in the kernel of H if and only if all α_i where $i > k$ is zero. Then v has to be a linear combination of just the basis elements of $\mathcal{C}$ and therefore must itself be a codeword.

Now comes the question of how to compute the matrix of transformation of H. Suppose it turns out that that the basis we chose was actually the standard basis $\{e_1, e_2, \cdots, e_n\}$. Then what can we say about the matrix H? Then clearly, it should map all vector $(\alpha_1, \alpha_2, \cdots, \alpha_n)$ to $(0, 0, \cdots, 0, \alpha_{k+1}, \cdots, \alpha_n)$. And this matrix is just

$$\hat{H} = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}_{n \times n}$$

where the I is the identity matrix of order $n - k$. But it's unlikely that we start with such a nice basis. The good news is that we can easily move from one basis to another. We just want a way to send each b_i to e_i so that we can then use the transformation $\hat{H}$ to send it to 0 if $i \leq k$ and keep it non-zero otherwise. Instead of sending b_i to e_i, the other direction is easy to compute.

What if we ask for a matrix B such that $Be_i = b_i$? This is easy because Be_i is just the i-th column of the matrix B. Thus the matrix is just $[b_1 b_2 \cdots b_n]$ where each b_i is now expanded as a column vector; just place each basis element side by side as a column vector and that is the transformation. Now that we have a matrix B that sends e_i to b_i, how do we get a matrix that sends b_i to e_i? Take B^{-1}!

Now look at $\hat{H}B^{-1}$ as a transformation. $\hat{H}B^{-1}b_i = \hat{H}e_i$ which is 0 if $i \leq k$ and e_i otherwise. Thus this matrix $\hat{H}B^{-1}$ is the matrix we were after: a matrix whose kernel is precisely the code $\mathcal{C}$.