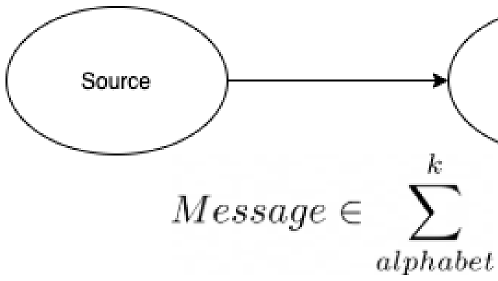
COMS 4995-2: Advanced Algorithms (Spring'20) April 30th, 2020

Lecture 26: Error-Correcting Codes

Instructor: *Alex Andoni* Scribes: *Daniel Beaglehole, David Rincon-Cruz*

1 Introduction



Error-correcting codes arise in the context of sending messages from a source to a destination. In particular, suppose we would like to communicate a message $m \in \Sigma^k$ from a source to a destination, where Σ is what is called the alphabet (could be bits or even up to the word level). The idea is that we send a longer message than the original to protect the original message from possible corruption.

Consider then two functions - the encoder $E : \Sigma^k \to \Sigma^n$ and a decoder $D : \Sigma^n \to \Sigma^k$, presumably with $n > k$. We then would like that our encoder/decoder has the property that for any message $(f_0, ..., f_{k-1}) = f$:

$$D(E(f) + \text{"}corruption\text{"}) = f$$

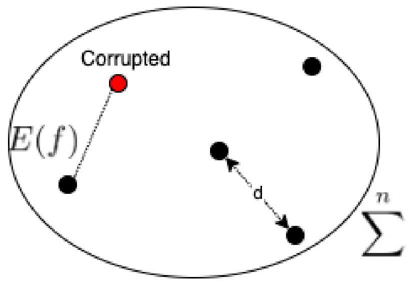
Some standard corruption models are the following. 1.) erasure: replace e characters with question marks (e.g. $0 \to ?$) 2.) error (substitution): e characters got changed, and you don't know which (e.g. $0 \to 1$) 3.) deletion/insertion - a message in Σ^n got transformed into Σ^{n+1} (e.g. $000 \to 0100$ or 00).

Our study is motivated, for example, by two applications: communication and memory storage. E.g. How can effectively communicate messages in the face of possible noise? Or, how can we store data such that it will remain in tact despite corruption?

Imagine storing information on k files on k hard drives - if one fails that's a major problem - you lose the file. One naïve way to make fault tolerant is by repetition: repeat each file on $\alpha > 1$ hard drives. In this case, the number of hard-drives would then be $n = \alpha * k$. With α copies of each file, if $(\alpha - 1)$ hard-drives fail, we will still be okay, as there is a remaining hard-drive! Can we do better? Yes: we will attempt to encode our message (files) such that for constant α, $n = \alpha * k$ hard-drives are resilient to far more errors.

2 Codes

Definition: a code is a set $\mathcal{C} \subset \Sigma^n$ where $\mathcal{C} = \{E(f) : f \in \Sigma^k\}$.



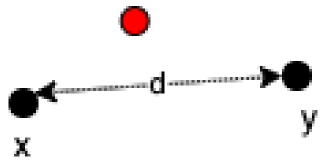
A natural decoding would then be: to map $m \in \Sigma^n$ into closest code in $\mathcal{C}$ and then decode f. Using this scheme, when can we decode our corrupted message? (How many coordinates can be affected?)

1.) **Erasure**. Suppose there is an adversary that deletes the minimum number of letters such that the code can no longer be decoded. The number of erasure errors our system can tolerate by this adversary is $\leq d-1$, where d is defined below.

Definition. $d := \min\{dist(x,y) : x, y \in C, x \neq y\}$.

If there are at least d erasures, then delete the position where x and y differ, and then you can't tell whether the encoded message is x or y.

2.) **Errors (substitution)**. Decoding: map the corrupted message $m \in \Sigma^n$ into closest codeword in $\mathcal{C}$. The decoding is correct iff $e \leq (d-1)/2$, where e is defined to be the number of errors in the encoded message.



Definition: a code $\mathcal{C}$ has parameters (k, n, d), where k - length of the original message, n - length of codeword, d - length of minimum distance between two distinct codewords.

E.g. the repetition code has the parameters $(k, n = \alpha * k, \alpha - 1)$.

. There exists a code $\mathcal{C}$ with $n = \alpha * k$, $d = \beta * k$, for $\alpha = O(1)$, $\beta = \Omega(1)$.

In other words, we can design a code that is tolerant to a constant fraction of failures - much better than a constant number (e.g. $\alpha - 1$ as in repetition).

Proof: take $\mathcal{C}$ = set of 2^k random vectors in $0, 1^n$ (for $\Sigma = \{0, 1\}$). Let the encoder E be any map of $\{0, 1\}^k \to e$ that is bijective. Our decoder D will then decode to the closest codeword.

How can we compute E, D (particularly D)? Naïvely, we can find closest codeword in time 2^k by bruteforce checking all the possible messages and their codewords. However, we'd like D to run in time $poly(k)$. This leads to the following definition for Reed-Solomon Codes.

Definition: Reed-Solomon (RS) code: $\mathcal{C} := \{f(\alpha_1), \dots, f(\alpha_n) : f(x) = \sum_{i=0}^{k-1} f_i x^i$, where $f_0, \dots, f_{k-1} \in \mathbb{F}_p\}$ for some finite field $\mathbb{F}_p$, and where $\alpha_1, \dots, \alpha_n$ are distinct in $\mathbb{F}_p$.

For the RS codes, an input message is encoded into a polynomial with coefficients equal to the message. I.e. the message $(f_0, f_1, \dots, f_{k-1})$ is encoded by E to $(f(\alpha_1), \dots, f(\alpha_n))$.

What are the parameters of the RS code, as we have defined them? Suppose here that $n = \Theta(k)$. Then, we have the following lemma.

E.g. $n = 2k \implies d = k + 1$, which implies that the message can tolerate k erasures or $k/2$ substitutions (constant fraction of the transmitted message).

Proof: Take 2 distinct RS codewords. I.e. $A(x) = \sum f_i x^i$, $B(x) = \sum g_i x^i$, with $\{f_i\} \neq \{g_i\}$. Then the distance between $E(f)$ and $E(g)$ can be found by considering the number of α_i's such that $A(\alpha_i) = B(\alpha_i)$.

Both $E(f)$ and $E(g)$ are polynomials of degree $\leq k - 1$. Therefore, at most $(k - 1)$ of the $alpha_i$'s can be equal (by the fundamental theorem of algebra). In particular, $C(x) = A(x) - B(x)$ can have at most $k - 1$ roots, and a root of $C(x)$ occurs iff $A(x) = B(x)$. This completes the proof of the lemma, which also completes the proof of Shannon's Theorem. $\square$

3 Algorithms for Encoding/Decoding

For encoding, we can directly compute the encoded message using our choice of coding scheme. However, for decoding algorithms we consider the different possible corruption schemes.

Erasure errors: We want a decoding algorithm to decode correctly provided that the number of errors $e \leq n - k = d - 1$. In the case of RS codes, this poses the following computational problem.

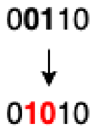
Computational Problem: Suppose we get a vector $(y_1, y_2, \dots, y_n)$ where for $i \in S$, $|S| <= e$, $y_i =?$ (as in the case of erasure). Otherwise, $y_i = f(alpha_i)$. How can we recover $f(x) = \sum_{i=0}^{k-1} f_i x^i$? Suppose the original message is $(f_0, \dots, f_{k-1})$, which is the same as the description of the entire polynomial f.

Algorithm: Polynomial Interpolation. Given evaluation of $f(x)$ in $n - e \geq k$ values. Let's set these evaluations up as a linear system of equations: $\sum_{i=0}^{k-1} f_i \alpha_j^i = y_j, j \notin S$.

Here we have K unknowns (f_i) - our original message, and $\geq k$ equations $(n - |s| \geq k)$. This linear system of equations can be represented by $Af = y$ where A is the Vandermonde matrix:

$$A = \begin{pmatrix} \alpha_1^0 & \alpha_1^1 & \cdots & \alpha_1^{k-1} \\ \alpha_2^0 & \alpha_2^1 & \cdots & \alpha_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_n^0 & \alpha_n^1 & \cdots & \alpha_n^{k-1} \end{pmatrix}$$

What about the case of substitution errors? The challenge here is we don't know S (position of errors). Trying all possible sets S too costly, as $\binom{n}{|S|}$ is approximately $\binom{n}{n/2}$, which is approx. $2^{\Theta(n)}$.



Berlekamp-Welch Algorithm (Decoding): Let the Error polynomial be $E(x) := \Pi(x - \alpha_i)$, where the product is over $y_i \neq f(\alpha_i)$. We don't know $E(x)$ otherwise we could recover the set of errors and we could do erasure decoding, but we'll try to learn it.

Property (*): $E(\alpha_i) * (f(x) - y_i) = 0, \forall i \in [n]$

Let $N(x) := E(x) * f(x)$. Then we claim:

Claim: There exist polynomials $N(\alpha_i) = y_i E(\alpha_i)$, where $N(x)$ has degree $e + k - 1$, and $E(x)$ has degree e.

The proof of this claim follows from property (*).

Algorithm: Learn/compute polynomial $N(x)$ of degree $e + k - 1$, $E(x)$ of degree e Such that $N(\alpha_i) = y_i E(\alpha_i)$

Proof: Setting up a system of linear equations Number of unknowns is $(e + k - 1) + 1 + e + 1$. The number of equations is n. We can set up this system as long as $(e + k - 1) + 1 + e + 1 \leq n$, i.e. $e \leq (n - k - 1)/2 = d/2 - 1$.

Therefore, RS is the best possible code for large Σ.