1 Fundamentals of Error Correcting Codes

Alice wants to transmit a message $m \in M$ over a noisy channel to Bob. To ensure that Bob is able to recover the message $m$, instead of sending $m$ over the channel, she sends $c = \text{enc}(m)$. On the other side of the channel, Bob receives $c'$, the corrupted version of $c$. He then uses the error correction algorithm to recover $c$ and runs decode to recover $\text{dec}(c) = m$.

$$m \in M \xrightarrow{\text{encode}} c = \text{enc}(m) \xrightarrow{\text{flip} \leq r \text{ bit}} \text{noisy channel} \xrightarrow{\text{error correct}} c' \xrightarrow{\text{decode}} m = \text{dec}(c)$$

It is common to refer to the image of $\text{enc}$ as the code.

Remark 1.1. The noisy channel can be modelled as a stochastic process which changes each input into the channel with probability $p$. However, in theoretical computer science we generally consider the worst case (adversarial) setting.

Definition 1.2. A code $C$ is a $(n, k, d)_q$ code if $C \subseteq \Sigma^n$ where $|\Sigma| = q$, $k = \log_q(|M|)$, and $x, y \in C \Delta(x, y) \geq d$.

Remark 1.3. $n$ is refereed to as the block length, $k$ as the dimension, and $d$ as the distance. $c \in C$ is refereed to as a codeword.

Claim 1.4. It is possible to correct $r \leq \frac{d-1}{2}$ errors using a $(n, k, d)_q$ code $C$.

Error Correction Alg: output $c^* \in C$ which minimizes $\Delta(c^*, c')$ where $c'$ is the corrupted message.

We will now show that $c^* = c$. Suppose $c^* \neq c$. Then by the triangle inequality

$$\Delta(c^*, c) \leq \Delta(c^*, c') + \Delta(c', c)$$

$\Delta(c', c) \leq r$ since the channel flips at most $r$ bits. $\Delta(c^*, c') \leq r$ since $\Delta(c^*, c') = \min_x \Delta(x, c') \leq \Delta(c, c') \leq r$. Therefore

$$\Delta(c^*, c') + \Delta(c', c) \leq r + r = \frac{d-1}{2} + \frac{d-1}{2} = d - 1 < d$$

This implies that there are two distinct codewords $c$ and $c^*$ whose distance is less than $d$. This contradicts the fact that $C$ is a $(n, k, d)_q$ code. Therefore, $c = c^*$.

Observation 1.5. We have shown that the proposed error correction algorithm is correct but this is still a sub-optimal result, as we have not shown that it is efficient. Showing that error correction is efficient may be challenging.

Geometrically, Our error correction algorithm draws a ball of radius $\frac{d-1}{2}$ around $c'$ and outputs the codeword in that ball. In our case, we are guaranteed that there will only be one codeword in that ball. This is what is refereed to as a uniquely decodeble code. However, if we relax that, we may be able to create an algorithm that looks at a polynomial number of codewords in a ball of radius $\frac{d-1}{2}$ around $c'$ and chooses a reasonable one.
2 Existence of Good Codes

Definition 2.1. The rate of an \((n, k, d)_q\) code is \(r = \frac{k}{n}\).

Definition 2.2. The relative distance of an \((n, k, d)_q\) code is \(\delta = \frac{d}{n}\).

Notice that the rate of a code is always \(\leq 1\).

We consider a code "good" if rate and relative distance are constants \((\Omega(1))\).

Theorem 2.3. There exist good codes. More formally, \(\forall n \in \mathbb{N}, \exists (n, k, d)_2\) codes with \(\frac{d}{n}, \frac{k}{n} = \Omega(1)\).

We will prove this by the probabilistic method. We will show that there is constant rate \((r)\) and constant relative distance \((\delta)\) such that for all \(n\), a random code with those dimension \(k = nr\) has a non-zero probability of having distance \(n\delta\).

\(K = 2^k\). Pick \(v_1, v_2, \ldots, v_K\) randomly and independently from \(\{0, 1\}^n\). Notice that

\[\mathbb{E}[\Delta(v_i, v_j)] = \frac{n}{2}\]

Since each bit of \(v_i\) and each bit of \(v_j\) is chosen at random, the expected value of the distance of the bits is \(\frac{1}{2}\). Thus the expected value of the sum of the distance of the bits is \(\frac{n}{2}\).

We are interested in the event that our code is bad, that the distance between 2 vs is low. We will refer to the event \(\Delta(v_i, v_j) < \frac{n}{2}(1 - \epsilon)\) as \(BAD_{ij}\). We can use the Chernoff bound to bound the probability of such an event

\[\mathbb{P}[BAD_{ij}] = \mathbb{P}[\Delta(v_i, v_j) < \frac{n}{2}(1 - \epsilon)] \leq 2^{-\Omega(c'^2 n)}\]

The code is bad if any of the codewords are too close to each other, in other words, if any \(BAD_{ij}\) event occurs. Thus the probability that a code is bad is

\[\mathbb{P}[\bigcup_{i \neq j} BAD_{ij}]\]

By the union bound we have

\[\mathbb{P}[\bigcup_{i \neq j} BAD_{ij}] \leq \left(\begin{array}{c} K \\ 2 \end{array}\right) 2^{-\Omega(c'^2 n)}\]

We merely need to find \(k\) such that \(\left(\begin{array}{c} K \\ 2 \end{array}\right) 2^{-\Omega(c'^2 n)} < 1\). Since \(K^2 > \left(\begin{array}{c} K \\ 2 \end{array}\right)\), it suffices to find \(k\) such that \(K^2 2^{-\Omega(c'^2 n)} < 1\).

\[K^2 2^{-\Omega(c'^2 n)} < 1 \iff 2^k 2^{-\Omega(c'^2 n)} < 1 \iff k = c''(\epsilon) n\]

Where \(c''\) is some function of \(\epsilon\). We have shown the rate is \(c''(\epsilon) = \Omega(1)\).

The distance of the code is \(\frac{1-\epsilon}{2} n\) since no two codewords have distance greater than \(\frac{n}{2}(1 - \epsilon)\). This tells us that the relative rate is \(\frac{1-\epsilon}{2} = \Omega(1)\).
3 Reed-Solomon Codes

Reed-Solomon codes are \((n, k, d)_q\) codes where \(q \geq n\) and \(\Sigma = \mathbb{F}_q\). The Reed-Solomon code with block length \(n\) and dimension \(k\), denoted \(RS_{n,k}\), is a subset of \(\mathbb{F}^n\). We will now provide a construction of \(RS_{n,k}\).

**Construction 3.1.** Fix \(\mathbb{F}^n\). To encode the message \(\hat{a} = (a_1, a_2, \ldots, a_{k-1})\) where \(a_i \in \mathbb{F}\), construct the polynomial \(P_a(x) = \sum_{i=0}^{k-1} a_i x_i\). The codeword \(C_a = (P_a(\alpha_1), P_a(\alpha_2), \ldots, P_a(\alpha_n))\).

In this construction \(S\) does not depend on the message, it is fixed.

**Question 1.** What is the distance of \(RS_{n,k}\)? In other words, what is the closest 2 codewords in \(RS_{n,k}\) can be?

This can be framed as an optimization problem:

\[
d = \min_{\hat{a} \neq \hat{b}} \Delta(C_a, C_b) = \min_{\hat{a} \neq \hat{b}} |\{x \in S| P_a(x) \neq P_b(x)\}|
\]

If \(P_a(x) = P_b(x)\), then \(P_a(x) - P_b(x)\) has a root at \(x\). Since the degree of \(P_a(x) - P_b(x)\) is at most \(k - 1\), and \(P_a(x) - P_b(x)\) has at most \(k - 1\) roots. Thus, there are at most \(k - 1\) values for \(x\) such that \(P_a(x) = P_b(x)\), and always at least \(n - (k - 1) = n - k + 1\) values of \(x\) such that \(P_a(x) \neq P_b(x)\). Therefore

\[
d = \min_{\hat{a} \neq \hat{b}} \Delta(C_a, C_b) = \min_{\hat{a} \neq \hat{b}} |\{x \in S| P_a(x) \neq P_b(x)\}| \geq n - k + 1
\]

**Remark 3.2.** This is actually the best distance that can be achieved (via the well-known Singleton bound).

**Question 2.** Can \(RS_{n,k}\) be used to get good binary codes?

\(RS_{n,k}\) encodes field elements into field elements. But what if we want to transmit messages in binary \((q = \mathbb{F}_2)\)? Simply represent the field elements using binary. \(x \in \mathbb{F}_q\) can be represented using \(\log_2(q)\) bits. Thus the encode function goes from having signature \(\text{enc} : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n\), to having the signature \(\text{enc} : \{0, 1\}^{k \log_2(q)} \rightarrow \{0, 1\}^{n \log_2(q)}\)

Notice that the distance of the code remains unchanged but the relative distance \(\delta = \frac{n - k + 1}{n \log_2(q)} \approx \frac{1}{\log_2(n)}\) decreases. Intuitively, the problem is that a flip of any of the bits in the encoding of a field element results in it being turned into a different field element. Ideally, we want to have to flip many of the bits in an encoding of a field element to turn it into a different element. But we already know how to achieve that: error correcting codes! We will send each binary encoding of a field element using an optimal error correcting code (encoding just \(\log n\) bits). Consequently, one can show that this leads to constant relative distance.

4 Linear Codes

**Definition 4.1.** \(\mathcal{C} \subseteq \mathbb{F}^n\) is a linear code if \(\mathcal{C}\) is a linear subspace.

Linear codes are denoted with square brackets, \([n, k, d]_q\). In this case \(k\) is the dimension of \(\mathcal{C}\) and \(q = |\mathbb{F}|\).

**Definition 4.2.** The Hamming weight of a codeword \(c\) is the number of non-zero symbols in the codeword.
We note that the distance \( d = \min \text{ weight codeword in } \mathcal{C} \).

Notice that this is a natural and equivalent definition of distance since the minimum distance between any two codewords is the same the the min weight over all code words. This is because \( \Delta(c_1, c_2) = \Delta(c_1 - c_2, 0) \). Since \( \mathcal{C} \) is is linear and \( c_1, c_2 \in \mathcal{C}, c_1 - c_2 \in \mathcal{C} \). Call \( c_1 - c_2 \) the codeword \( c_3 \). thus \( \Delta(c_1, c_2) = \Delta(c_1 - c_2, 0) = \Delta(c_3, 0) = \text{weight}(c_3) \).

Remark 4.3. One can show that good linear codes exist by using the probabilistic method.

Claim 4.4. Reed-Solomon codes are linear.

The fact that the sum of two codewords is another codeword is follows easily from \( P_a + P_b = P_{a+b} \).