Lecture 4: Codes based on Concatenation

Error-Correcting Codes (Spring 2016)
Rutgers University
Swastik Kopparty
Scribe: Aditya Potukuchi and Meng-Tsung Tsai

1 Overview

In the last lecture, we studied codes based on univarite polynomials, also known as *Reed-Solomon codes*, and on multivariate polynomials, also known as *Reed-Muller codes*. The Reed-Solomon codes had rate $R = 1 - \delta$, which is optimal (given by the Singleton bound), but required alphabets of large size ($|\Sigma| \geq n$). Multivariate polynomials in a constant number of variables, however, gave codes of positive rate and distance, and required an alphabet size of only $\sim n^{\frac{1}{m}}$, where m is the number of variables. While this is not as bad as the Reed-Solomon codes in this aspect, it still is a growing function of n.

One interesting special cases of Reed-Muller codes is that over an alphabet of size $\log n$, and degree $d = 1$, We get codes of size $|\mathcal{C}| = 2n$, with distance $\delta = \frac{1}{2}$, known as the *Hadamard code*. We will first study a few basic properties of the Hadamard code, and move on to *Concatenation* of codes, which gives us codes of constant rate and distance, over a binary alphabet. We study concatenation codes under both error models, i.e., worst case, and random. In the latter case, we shall study constructions of codes that meet Shannon Capacity.

2 The Hadamard Code

Recall that the extended Hamming code[1] is nothing but the Hamming code with an extra linear constraint, i.e., $\sum_{i\in[n]} x_i = 0$. This is a code of size $\Theta\left(\frac{2^n}{n}\right)$ of distance 4.

Claim 1. *The Hadamard code is the Dual of the extended Hamming Code.*

Proof. The Hadamard code is obtained by evaluating point $x \in \{0,1\}^m$ at all the degree 1 polynomials over $\mathbb{F}_2$ in m variables, which has all $\{0,1\}$ vectors of length $m+1$, starting with 1, as columns. This is exactly the same as the parity check matrix of the extended Hamming code of length 2^m.

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

[1] Problem 3 in Homweork 1.

□

3 Construction of Concatenated Codes

Given a long code over a large alphabet, and a (possibly) short code over a small alphabet, we perform an operation called *concatenation* to get a long code over a small alphabet with "decent" parameters, i.e., with $R > 0$ and $\delta > 0$. This idea was due to Forney in 1970.

The strategy is the following: We use an *outer* code, which is basically the long code over a large alphabet, and encode each alphabet of this code using the *inner* code. We shall call these the *blocks* of the code.

3.1 Choice of Outer and Inner codes

1. The outer code is a Reed-Solomon code $\mathcal{C}$ over $\mathbb{F}_{2^t}$ with rate R. These are codes of length 2^t constructed using polynomials of degree $R \cdot 2^t$.
2. The inner code is any binary code $|\mathcal{C}'| \subseteq \mathbb{F}_2^a$ with dimension t, so $|\mathcal{C}'| = 2^t$.

Let $r := \frac{t}{a}$ to denote the rate of $\mathcal{C}'$. We require that $dist(\mathcal{C}') \geq 1 - H(r)$, and we can find such a linear code by brute force search[2].

3.2 Parameters of the Resulting Code

The size of the code is given by the outer code, i.e., $|\mathcal{C}| = \left(2^t\right)^{R2^t+1}$.

Since the outer code words have minimum distance $1 - R$, and for each block, they differ in, the inner code words have distance $H^{-1}(1-r)$, the distance of the resulting code is $(1-R)H^{-1}(1-r)$. Here H^{-1} is the inverse of the entropy function which takes values between 0 and $\frac{1}{2}$.

The length of a code word in this code is $a2^t$ bits. So the rate is given by $\frac{t \cdot R \cdot 2^t}{a \cdot 2^t} = \frac{tR}{a} = rR$.

3.3 Construction Time

There are three main steps in the construction of these codes, i.e., finding the inner code, via. brute force, construction of the generator matrix for the outer code, and the time taken to 'substituting' the generator matrix of the smaller code in the generator matrix of the longer code.

Finding a small code efficiently takes $\texttt{poly}(2^a) = \texttt{poly}(2^t)$ time, if r is a constant (what about the case when r is not a constant?). The rest of the steps take time $\texttt{poly}(2^t)$. So the whole process takes $\texttt{poly}(2^t)$.

[2]Problem 2 in Homework 1.

3.4 Decoding of Concatention Codes from a Constant Fraction of Errors

We basically import the known algorithm to decode Reed-Solomon codes with a slight modification to correct some constant fraction ϕ of errors. The decoding algorithm comprises of the following two main steps:

Step 1 Decode each block of the Inner code by brute force.

Step 2 Use the Berlekamp-Welch algotithm to decode the resultant string over large alphabet to the nearest Reed-Solomon codeword.

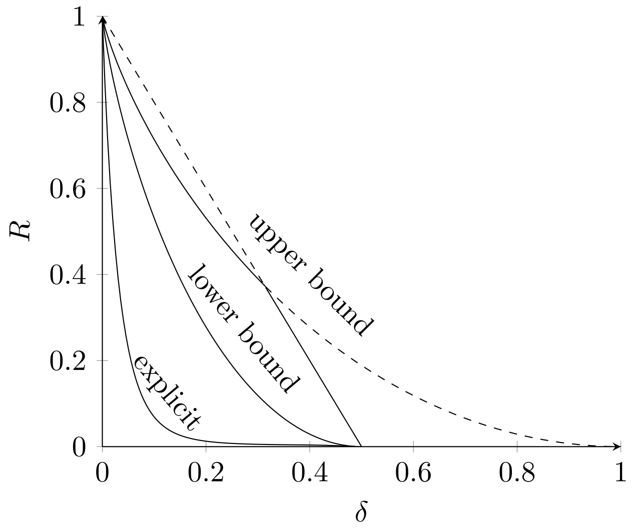
Claim 2. *The above algorithm decodes the concatenation code from $\frac{1}{4}rR$ fraction of errors.*

Proof. If the total fraction of errors is at most $\frac{1}{4}rR$, then the fraction of blocks that have more than $\frac{1}{2}r$ errors is at most $\frac{1}{2}R$ (by the Markov Inequality). After the first step of the algorithm, there is at most $\frac{1}{2}R$ fraction of the blocks that are decoded incorrectly. Since this is within the decoding radius of the Reed-Solomon code, the second step of the algorithm decodes the resulting string correctly. □

One can correct $\frac{1}{2}(1-R)H^{-1}(1-r)$ using a more carefully designed algorithm that keeps track of the errors corrected so far.

4 The story so far..(in the Hamming model)

Although we proved the existence of codes with rate $R = 1 - H(\delta)$, we are far behind in being able to construct them, even in polynomial time. The new line we have on the plot shows that fact.



5 Random Error Model: Codes that meet Shannon capacity

Let $\textsc{Bsc}(p)$ be a binary symmetric channel so that every bit transmitted through the channel has probability p being flipped, i.e. from 0 to 1 or from 1 to 0. Shannon's theorem tells us that for all $R < C = 1 - \mathrm{H}(p)$ (a.k.a. Shannon capacity) there exist an encoder $E : \{0,1\}^{Rn} \to \{0,1\}^n$ and a decoder $D : \{0,1\}^n \to \{0,1\}^{Rn}$ so that the failure probability over $\textsc{Bsc}(p)$ is $e^{-\delta n}$ for some constant $\delta > 0$ (depending on $C - R$). Formally,

$$\Pr_{z\in\{0,1\}^n, \Pr[z_i=1]=p} [D(E(m) \oplus z) \neq m] \leq e^{-\delta n}.$$

The existence of such a pair E, D is guaranteed by a probabilistic argument[3], which says nothing about an efficient construction. Here, we try to construct such a code efficiently.

5.1 A First Attempt

As a first attempt, we try the following code:

1. By brute force, find a code of length $\Theta(\log n)$ that has rate $r = C - \varepsilon$ such that decoding to the nearest codeword has failure probability $\leq \frac{1}{\mathtt{poly}(n)}$. The existence of such a code is guaranteed by Shannon's Theorem.
2. Break the message into $\Theta\left(\frac{n}{\log n}\right)$ blocks of size $\Theta(\log n)$, and encode each block using the code above.

Union bound tells us that the probability that some block is decoded incorrectly is at most $\Theta\left(\frac{n}{\log n} \cdot \frac{1}{\mathtt{poly}(n)}\right)$, which is $\Theta\left(\frac{1}{\mathtt{poly}(n)}\right)$ for a sufficiently good code found in step *1.* above. (Note that the expected number of incorrectly decoded blocks is $\frac{1}{\mathtt{poly}(n)}$, so after a (not very) large number of queries, it is very likely that one block has been incorrectly decoded).

While this seems quite good, our goal is to construct a code that has exponentially small probably of incorrect decoding (as guaranteed by Shannon's theorem). For this purpose, we have the following solutions.

6 The Real Solution

Solution 1 is a code that has exponentially small error probability, and can be constructed, encoded, decoded in $2^{\mathcal{O}(\log^2 n)}$ (quasi-polynomial) time. The idea is using concatenated codes. We use Reed-Solomon codes as outer codes, and codes that meet Shannon capacity as inner codes. The main idea behind this approach is that although some blocks may get incorrectly decoded, to make the decoding truly err, we need that a lot of blocks are incorrectly decoded (dictated by the outer code), which is unlikely.

[3]The proof can be found in the 2nd lecture note.

One of the main ingredients in this recipe, is a linear code of length t, that meets Shannon capacity.

Exercise: For any $\varepsilon > 0$, and $0 \leq p \leq 1$, there exist linear codes of rate at least $1 - H(p) - \varepsilon$ that have exponentially low probability of incorrect decoding.

[Hint: Choose a $m \times n$ generator matrix with each entry uniformly and independently. Proceed exactly as in Shannon's theorem, and bound 'both kinds' of errors, i.e., the one where there are 'too many errors', and the one where there are more than one codewords near the corrupted codeword. Note that in the latter case, it is enough to bound the error probability for one code word (say 0), because in a linear code, all code words 'look the same'. This makes the analysis of this error easier than in Shannon's Theorem]

The next step is to find such a code.

Lemma 1. *It takes $\mathcal{O}\left(2^{t^2}\right)$ time to find a linear code of length t, rate $C - \varepsilon/2$ with exponentially low error probability.*

Proof. One can enumerate all possible linear codes specified by the span of t vectors and pick a feasible linear code. The number of possible linear codes is thus bounded by $\binom{2^t}{t} = 2^{t^2}$, and the time taken to verify the probability of bad decoding for every word is bounded by $\mathcal{O}(2^{2t})$. □

6.1 Choice of Outer and Inner Codes

1. For the outer code we use the Reed-Solomon code over $\mathbb{F}_{2^t}$ of rate $1 - \varepsilon/2$ and length ℓ.
2. For the inner code, we use the code $\mathcal{C}'$ which has length t, rate $C - \varepsilon/2$ probability of incorrect decoding at most $e^{-\delta t}$.

Setting $t = \frac{2}{\delta} \log n$, we get that the probability of incorrect decoding is at most $\frac{1}{n^2}$. We also get $\ell = \frac{\delta n}{2 \log n}$

6.2 Parameters of the Code

The size of the code is, again, given by the outer code $|\mathcal{C}| = (2^t)^{\ell(1-\varepsilon/2)}$.

The rate of this code, similar to the previous case, is $(C - \varepsilon/2)(1 - \varepsilon/2) > C - \varepsilon$.

To compute the failure probability, we note that $\mathcal{C}'$ has failure probability over $\textsc{Bsc}(p)$ bounded by $1/n^2$. To make the concatenated code err over $\textsc{Bsc}(p)$, it requires more than $\varepsilon/4$ fraction of blocks decoded incorrectly, which happens with probability at most

$$\begin{aligned}
\binom{\ell}{\varepsilon\ell/4}\frac{1}{(n^2)^{(\varepsilon/4)\ell}} &\le 2^{\mathrm{H}(\varepsilon/4)\cdot\ell}\frac{1}{2^{(2\log n)(\varepsilon/4)\ell}} \\
&= \frac{1}{2^{\ell(2(\varepsilon/4)\log n - \mathrm{H}(\varepsilon/4))}} \\
&\le \frac{1}{2^{\ell(\varepsilon/4)\log n}} && \text{(note that } \ell = \tfrac{\delta n}{2\log n}\text{)} \\
&= \frac{1}{2^{\frac{\varepsilon}{8}\delta n}} \\
&= \frac{1}{e^{\delta' n}} \text{ for some constant } \delta' > 0 \\
&= \exp(-n)
\end{aligned}$$

6.3 Construction Time

Since we have efficient algorithms to construct the outer code $\mathcal{C}$, similar to the previous case, the total running time is dominated by the construction of inner code $\mathcal{C}'$, which takes $2^{\mathcal{O}(\log^2 n)}$ time due to Lemma 1.

Remark: In terms of running time, the main bottleneck is finding codes of length $\Theta(\log n)$ that meet Shannon capacity. It is natural to attempt to bring down the construction time to $\texttt{poly}(n)$ by replacing $\mathcal{C}'$ by a code of smaller length, say $\sqrt{\log n}$. Here, the problem occurs in the outer code, where we need large alphabet sizes ($\ge l$).

6.4 Decoding

To decode the concatenated code, we find the nearest codeword by brute-force to decode the inner code. This takes time $\mathcal{O}(2^t \cdot \frac{\delta n}{2\log n}) = \texttt{poly}(n)$. Then, we use Berlekamp-Welch algorithm to decode the outer code. In total, it requires $\texttt{poly}(n)$ time.

7 Other Solutions

Solution 2 is a code that meets Shannon capacity C and can be constructed, encoded, decoded in $\texttt{poly}(n, 2^{1/\varepsilon})$ time, a.k.a. Forney code. This code has rate $C - \varepsilon$ and the failure probability over $\textsc{Bsc}(p)$ is at most $e^{-\delta n}$. The idea is the same as the Solution 1, except the following 2 changes.

1. Replace the RS code with a code over $f(\varepsilon)$-size alphabet so that the code has rate $1 - \varepsilon/2$ and can be efficiently decoded from at most $r(\varepsilon) > 0$ fraction of errors.
2. Replace the inner code with a code of length $\mathcal{O}(\log f(\varepsilon))$ with rate $C - \varepsilon/2$ so that the failure probability over $\textsc{Bsc}(p)$ is at most $q(\varepsilon)$, requiring that $q(\varepsilon) < r(\varepsilon)/5$.

Solution 3 is called Polar code that meets Shannon capacity C and can be constructed, encoded, decoded in $\texttt{poly}(n, 1/\varepsilon)$ time.