6.440 Essential Coding Theory Feb 27, 2008

Lecture 7

Lecturer: Madhu Sudan *Scribe: Huy Nguyen*

1 Administrative

- Problem set 2 is out.
- Problem set 1 is still to be graded.

2 Lecture Overview

Binary codes

- Concatenated codes[Forney]
- Justesen codes
- BCH codes

3 Review

1. Rozencraft ensemble is an ensemble of functions that maps $x \in \mathbb{F} \rightarrow \langle x, ax \rangle$. This construction gives good code for most values of α but we do not know which α.
2. Reed-Solomon code is the code that maps each message $M(\cdot)$, a polynomial with coefficients in $\mathbb{F}_q (q \geq n)$, to an n-tuple $\langle M(\alpha_1), \cdots, M(\alpha_n) \rangle$. The catch here is that it is not a binary code.
3. Multivariate polynomial codes: There are many ways to apply multivariable polynomials to build codes but there is no real punchline here.
4. Hadamard codes

   $$HH^T = nI \text{ where } H \in \{-1, +1\}^{n \times n}$$

   The codewords are the rows of the matrix $\begin{bmatrix} H \\ -H \end{bmatrix}$.

Today we will use the Rozencraft ensemble and the Reed-Solomon code to build nice families of codes. Recall the goal from last time was to construct good code over small field size. Ideally we want to get binary codes from Reed-Solomon codes, which requires $q \geq n$.

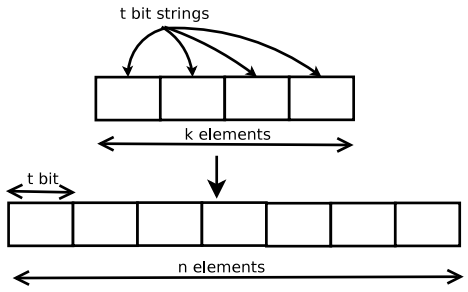

Figure 1: The encoding process of the naive code

4 Naive method to convert non-binary code to binary

In this section we will describe a naive idea to get binary codes. Let's start with Reed-Solomon (RS) code over $\mathbb{F}_{2^t}$, $t = \log n$. We can represent each field element using a t-bit string. Suppose the original RS code is a $[n, k, n-k]_n$ code (this code can be obtained by evaluating the polynomial at all points of the field $\mathbb{F}_{2^t}$).

The encoding maps a message in $\{0,1\}^{kt}$ to a codeword in $\{0,1\}^{nt}$. We can find a representation of the code such that the distance of the code does not change (it might even go up). For example we can use a mapping that preserve correspondence $\mathbb{F}_{2^t} \to \mathbb{F}_2^t$. This yields a $[nt, kt, n-k]_2$ code. Let's compute the relative rate and distance:

$$\begin{aligned} R &= \frac{k}{n} = \frac{kt}{nt} \\ \text{old } \delta &= \frac{n-k}{n} \\ \text{new } \delta &= \frac{n-k}{n \log n} \to 0 \end{aligned}$$

Obviously the new relative distance is much worse than the old relative distance. Recall that the distance is equal to the minimum number of ones in each non-zero codewords. The problem here is that we cannot avoid the sparse representations if the representation is too compact (we only use t bits to represent 2^t values). Fortunately, the error-correcting code comes to the rescue here. We can use error-correcting codes to represent $\mathbb{F}_{2^t}$ much more nicely as described in the next section.

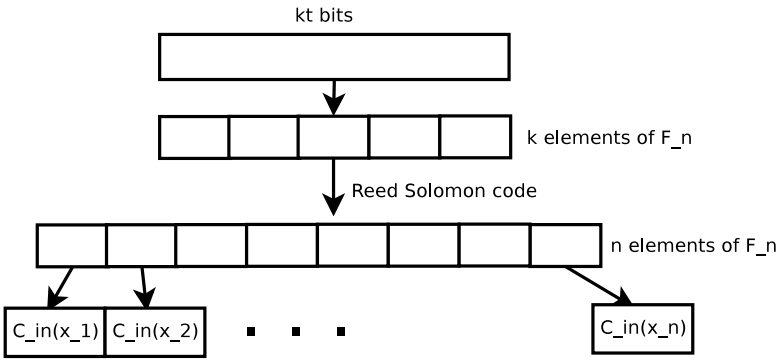

Figure 2: The encoding process of $RS \cdot C_{in}$

5 Concatenated codes [Forney '66]

Let's pick some nice code C_{in} (the inner code), $C_{in} : \{0,1\}^t \to \{0,1\}^{2t}$. Suppose C_{in} is some $[2t, t, \delta_{in} \cdot 2t]$ code where δ_{in} is a good constant. We know that C_{in} exists. For example we can pick $\delta_{in} = H^{-1}(\frac{1}{2})$ (Gilbert-Varshamov code). Use C_{in} to represent elements of $\mathbb{F}_{2^t}$.

The entire process $RS \cdot C_{in}$ gives a composed code $[2nt, kt, ?]_2$. But what is the distance? We can apply a simple counting argument to compute distance. The original kt-bit message is non-zero. When we convert it to k elements of $\mathbb{F}_n$, this sequence is also non-zero. After applying the Reed-Solomon code, we get n elements of $\mathbb{F}_n$, and at least $n-k$ of which are non-zero. Finally, after applying C_{in} to each of n elements, at least $n-k$ of them are non-zero codewords. Each non-zero codeword in C_{in} has at least $\delta_{in} \cdot 2t$ non-zero bits. Therefore, $RS \cdot C_{in}$ is a $[2nt, kt, \delta_{in}(n-k)(2t)]_2$ code.

In general, if C_{outer} is a $[n_1, k_1, d_1]$ code and C_{in} is a $[n_2, k_2, d_2]$ code, the composed code is a $[n_1 n_2, k_1 k_2, d_1 d_2]$ code. We only need to make sure that the number of codewords of C_{in} is equal to the field size of C_{outer}. For example, C_{in} is $[n_2, k_2, d_2]_q$ and C_{outer} is $[n_1, k_1, d_1]_{q^{k_2}}$. Similarly, if the rate and distance of C_{outer} are R_1, δ_1 and of C_{in} are R_2, δ_2, concatenation gives the rate $R = R_1 R_2$ and distance $\delta = \delta_1 \delta_2$.

The good thing about this code is that C_{outer} can be over a large alphabet size, which is usually easier to construct. The bad news is we still don't know how to get $R_2, \delta_2 > 0$. Forney's idea is that because n_2 is small compared to n_1 (in the specific case of $RS \cdot C_{in}$, n_2 is only logarithmic compared to n_1). Therefore, we can use the Gilbert-Varshamov search, which has running time exponential in n_2 but still polynomial in n_1, to find a good inner code. For example, $k = \frac{n}{2}, R_1 = \frac{1}{2}, d = \frac{n}{2}, \delta_1 = \frac{1}{2}$. The outer alphabet has size $2^{k_2} \geq n \to k_2 \geq \log n$. Using Varshamov search, we get the inner code of length $2 \log n$, and distance $d_2 = H^{-1}(\frac{1}{2})(2 \log n)$ over field size $q = 2$. The

concatenation gives a $[2n\log n, \frac{n}{2}\log n, H^{-1}(\frac{1}{2})\log n\frac{n}{2}]_2$ code. This code has rate $R = \frac{1}{4}$ and distance $\delta = H^{-1}(\frac{1}{2})\cdot\frac{1}{2}$.

There has been a debate in the community regarding whether this coding scheme is explicit or not. Some people think it is explicit because it has a polynomial time construction. On the flip side, some people think it is not explicit and is still "search". Formally, their definition of explicit is that we should be able to compute the (i, j) entry of the generator matrix in time $poly(\log n)$.

6 Justesen's code

In Forney's code, we use the same C_{in} to encode n elements of $\mathbb{F}_{2^t}$ in a RS codeword. However, there is no reason why we have to use the same C_{in}. In fact, we only need to make sure that most of the encoding of n elements are good. Recall that actually most of the codes in the Rozencraft ensemble are good, so we can just use the ensemble.

The encoding process is as follows. Pick $\mathbb{F}_{2^t}, n = 2^t$. Assume $\mathbb{F}_n = \{\alpha_1, \cdots, \alpha_n\}$. Let $k = \frac{n}{2}$. Let the message $m(x) = (m_0, \cdots, m_{k-1}) \in \mathbb{F}_n^k \equiv \{0,1\}^{k\log n}$. Let $M(x) = \sum m_i x^i$. We encode $m(x)$ by $\langle M(\alpha_i), \alpha_i M(\alpha_i)\rangle_{i=1}^n$. We won't go into the details of the proof but the distance of this code is roughly the same as that of Forney's code.

Note that in the example we gave, the rate $\delta = H^{-1}(\frac{1}{2})\frac{1}{2}$ is not as good as the bound $H^{-1}(\frac{3}{4})$ for $R = \frac{1}{2}$ but this is basically the best we know how to do construct explicitly.

7 BCH code

Let's revisit the problem of finding codes with small distance. If we want a code of distance 3, we can use Hamming code. What about some higher distance? Says we want a code of distance 5.

Applying the Hamming bound, we get

$$\begin{aligned} 2^k \cdot (1 + \binom{n}{1} + \binom{n}{2}) &\leq 2^n \\ 2^k \cdot \theta(n^2) &\leq 2^n \\ n - k &\geq (2 - o(1))\log n \end{aligned}$$

On the other hand, let's take RS code and choose $n - k = 5$, we get the code $[n, n-4, 5]_n$. Applying the naive idea, we get the code $[n\log n, (n-4)\log n, 5]_2$. Let $n' = n\log n$, $k' = (n-4)\log n$. We get a code $[n', k', 5]_2$ where $n' - k' = 4\log n \leq 4\log n'$.

This code differs from the Hamming bound by $2\log n$. Can we do better? As it turns out, BCH code can actually achieve the Hamming bound. Let's see how.

Let C_1 be the code over $\mathbb{F}_n = \mathbb{F}_{2^t}, n = 2^t$ with the following parity check matrix:

$$H = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \alpha_1^3 \\ 1 & \alpha_2 & \alpha_2^2 & \alpha_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \alpha_n^3 \end{bmatrix}$$

C_1 is a $[n, n-4, 5]_n$ code. The reason why the code has distance 5 is as follows. If the code has distance 4, there exists 4 rows of H that are linearly dependent. However, this cannot happen because the submatrix consisting of 4 rows of H is a Vandemonde matrix whose determinant is non-zero when the elements are distinct.

Now consider $C_{BCH} = C \cap \{0,1\}^n$. Clearly, the length and the distance of the code do not change so $C_{BCH} = [n, ?, 5]_2$. The main question here is how many codewords there are in C_{BCH}. We know that the all zero codeword is in C_{BCH} but is there any other codeword?

Let's represent all entries in $\mathbb{F}_n$ by vectors such that:

$$\begin{aligned} \mathbb{F}_{2^t} &\longleftrightarrow \mathbb{F}_2^t \\ \alpha &\longleftrightarrow V_\alpha \text{ (respect addition)} \\ 1 &\longleftrightarrow (100\cdots0) \end{aligned}$$

Apply the representation above to H and we get a new matrix:

$$H' = \begin{bmatrix} 1 & V_{\alpha_1} & V_{\alpha_1^2} & V_{\alpha_1^3} \\ 1 & V_{\alpha_2} & V_{\alpha_2^2} & V_{\alpha_1^3} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & V_{\alpha_n} & V_{\alpha_n^2} & V_{\alpha_1^3} \end{bmatrix}$$

If a $\{0,1\}$ vector $X = (x_1 \cdots x_n)$ satisfies $XH' = 0$ then

$$\begin{aligned} \sum x_i V_{\alpha_i} &= 0 \\ V_{\sum x_i \alpha_i} &= 0 \\ \sum x_i \alpha_i &= 0 \\ XH &= 0 \end{aligned}$$

Consider a matrix $\tilde{H}$ equal to H' with the third column removed:

$$\tilde{H} = \begin{bmatrix} 1 & V_{\alpha_1} & V_{\alpha_1^3} \\ 1 & V_{\alpha_2} & V_{\alpha_1^3} \\ \vdots & \vdots & \vdots \\ 1 & V_{\alpha_n} & V_{\alpha_1^3} \end{bmatrix}$$

Claim 1 *For any $X \in \{0,1\}^n$ such that $X\tilde{H} = 0$, $XH = 0$.*

Proof The only question is whether $\sum x_i\alpha_i^2 = 0$. Over $\mathbb{F}_{p^t}$, $(x+y)^p = x^p + y^p \ \forall x, y$. Therefore,

$$\begin{aligned}\sum x_i\alpha_i^2 &= \sum x_i^2\alpha_i^2 \\ &= (\sum x_i\alpha_i)^2 \\ &= \sum x_i\alpha_i \\ &= 0\end{aligned}$$

The second and third columns of $\tilde{H}$ impose on the code $\log n$ linear constraints each so the dimension of the code is $n - 2\log n - 1$. Thus, C_{BCH} is a $[n, n - 2\log n - 1, 5]$ code. ■

In general, the BCH code of distance d has the following parity check matrix:

$$\begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{d-2} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{d-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{d-2} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & \alpha_1 & \alpha_1^3 & \cdots & \alpha_1^{d-2} \\ 1 & \alpha_2 & \alpha_2^3 & \cdots & \alpha_2^{d-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^3 & \cdots & \alpha_n^{d-2} \end{bmatrix}$$

The number of columns is $1 + \lceil \frac{d-2}{2} \rceil \log n$ so the binary code we get satisfies $n - k \leq \lceil \frac{d-2}{2} \rceil \log n$.

By the Hamming bound for code of length n and distance d,

$$2^k \binom{n}{\frac{d}{2}} \leq 2^n \rightarrow n - k \geq \frac{d}{2} \log \frac{n}{d}$$

In the case $d = n^{o(1)}$, $n - k \geq \frac{d}{2} \log n$. Thus, BCH is essentially optimal as long as d is small.

The problem is more difficult for bigger alphabet. Consider code over the ternary alphabet $\{0, 1, 2\}$. The Hamming bound is $n - k \geq \frac{d}{2} \log_3 n$. BCH technique gives $n - k = \frac{2}{3} d \log_3 n$ and we do not know how to get a better code in this case.