6.440 Essential Coding Theory Feb 25, 2008

Lecture 06

Lecturer: Madhu Sudan *Scribe: Christina Wright*

1 Admin

- PS2 will go out today, Due next Friday (03/07)
- mailing list? [handed around new signed sheet]
- scribe? sign up.

2 Today: Algebraic Codes

- Wozencraft Emsemble
- Reed-Solomon
- Multivariate Polynomials
- concatenation; Forney; Justesen [not covered]

3 Review

We've worked with (n, k, d) codes, $R \equiv \frac{k}{n}$, $\delta \equiv \frac{d}{n}$.
'Gilbert-Varshamov' proved existence of codes.
Now we want to construct them.
Though we still won't achieve $R = 1 - H(\delta)$ today.

3.1 Algebra Review

- Finite fields exist and can be computed efficiently.
- Polynomials over (finite) fields have few zeros.

Both of these facts will be useful for constructing error-correcting codes.

4 Wozencraft Ensemble

The Wozencraft ensemble is not itself a code, but rather a collection of codes $\{C_\alpha\}$ where all codes C_α have $R = \frac{1}{2}$. At least one code satisfies $\delta \geq H^{-1}(\frac{1}{2})$ [or $R = 1 - H(\delta)$]. This ensemble of codes relies only on the first algebraic fact: that finite fields exist.

Code	Ensemble Size
Gilbert	2^{2^k}
Varshamov	2^{n^2}
Wozencraft	2^n

4.1 Construction

- k bits $\rightarrow n = 2k$ bits (other variations are possible)
- Choose a finite field, $\mathfrak{F}$, of size 2^k, $\mathfrak{F} = \mathfrak{F}_{2^k} \leftrightarrow \mathfrak{F}_2^k$ (mapping is addition preserving)
- code maps one elements in $\mathfrak{F}$ to two elements in $\mathfrak{F}$: $C_\alpha : x \rightarrow < x, \alpha x >$ $, x \in \mathfrak{F}, \alpha \neq 0$

4.2 Behavior

Lemma 1 *Choose α at random. Let $\tau = H^{-1}(\frac{1}{2}) - \epsilon$. Then*

$$Pr_{\alpha \in \mathfrak{F}}[\delta(C_\alpha) \leq \tau] \rightarrow exp(-k) \tag{1}$$

Where $\mathfrak{F}$ is the multiplicative group (no zeros).

Claim 1 *$< x, y > \neq < 0, 0 >$ then there exists at most one α such that $< x, y > \in C_\alpha$. aka C_α's are disjoint.* ***Proof:*** *$\alpha = x^{-1}y$ if x is invertible.*

Elementary Fact 1 *For linear codes C: $\Delta(C) = \min_{x \neq y}\{\Delta(x, y)\}$. If we fix $y = 0$ then we get $\Delta(C) = \min_{x \neq 0, x \in C}\{wt(x)\}$*

We say α is **bad** if $\exists < x, y > \in C_\alpha$ s.t. $0 < wt(< x, y >) < \tau n$. (not enough distance) Each vector can make at most one code bad, since it can only belong to one code. Thus,
\# α bad $\leq$ (\# vectors $< x, y >$ s.t. $0 < wt(< x, y >) < \tau n) \leq 2^{H(\tau)n}$

4.2.1 How many α's?

$2^{n/2}$. So,

$$Pr[\alpha\ bad] \leq \frac{2^{H(\tau)n}}{2^{n/2}} = 2^{-\epsilon' n} \tag{2}$$

4.2.2 What does the generator matrix of C_α look like?

$k \times 2k$: $[I|M_\alpha]$. Thus, $< v_\beta > [I|M_\alpha] =< v_{\alpha\beta} >$

5 Reed-Solomon

The Wozencraft ensemble relied on the fact that finite fields exist, now we will use the second fact (that polynomials over finite fields have few roots) to create codes.

idea: [diagram]

For Reed-Solomon codes we choose 3 parameters: Σ, n, k

- $\Sigma = \mathfrak{F}_q$ (large)
- n distinct points in $\mathfrak{F}_q$: $\alpha_1, \alpha_2, ..., \alpha_n \in \mathfrak{F}_q$ $[n \leq q]$
- $1 \leq k \leq n$

We can then think of our message, $m_0...m_{k-1} \in \Sigma^k = \mathfrak{F}^k$, as a polynomial: $M(x) = \sum_{i=0}^{k-1} m_i x^i$. Then the encoding of the message is $M \rightarrow < M(\alpha_1), M(\alpha_2), ..., M(\alpha_n) >$. This is a linear code which maps $\Sigma^k \rightarrow \Sigma^n$.

$$\begin{aligned} distance(RS) &= \min_{M \neq 0} \#\alpha \ s.t. \ M(\alpha) \neq 0 \\ &= n - \max_{M \neq 0} \#\alpha \ s.t. \ M(\alpha) \neq 0 \\ &= n - (k-1) \end{aligned}$$

The last line utilizes the limitation on the number of roots a polynomial can have. This code is great if you want to use a large alphabet.

5.1 Linear Codes and Duels

If we have a code C, with a generator matrix G, and a parity check matrix H, then the **dual** of that code is $C' = C^\perp$, with generator matrix $G' = H^T$, and parity check matrix $H' = G^T$.

[diagram]

A code that achieves the Singleton Bond (concretely, not in the limit) is called **Maximum Distance Separable (MDS)**.

Lemma 1 *MDS linear code* $\implies$ *duel is also MDS code*

6 Multivariate Polynomial Codes

Now we want to construct codes that use smaller alphabets.

[diagram]

Schwartz-Zippel lemma 1 $[r < q]$ *Let f be a degree r non-zero multivariate polynomial over $\mathfrak{F}_q$ then*

$$Pr_{\alpha_1 \dots \alpha_m \in_R \mathfrak{F}_q^m}[f(\alpha) = 0] \leq \frac{r}{q} \tag{3}$$

Proof induction on m. omitted. ■

This bound is tight. Also note that the right side of the equation does not involve m.

6.1 construction

We specify the code by $(\mathfrak{F}_q, r, m)$

- $\mathfrak{F}_q^k \to \mathfrak{F}_q^n$
- $k = \#$ coefficients $= \binom{r+m}{r} \geq (\frac{r}{m})^m$ or $(\frac{m}{r})^r$
- $n = q^m$
- $\delta = 1 - r/q$

example: $m = 2$ then $\mathfrak{F}_q^k \to \mathfrak{F}_q^{q^2}$ and $r = q/2, k = \binom{r}{2} \approx \frac{q^2}{8}$. $R = 1/8, \delta = 1/2$. The alphabet size is order square root of the length of the code.

So we have a loss in rate from the Reed-Solomon code, but a smaller alphabet size by a factor of a square root. In general R is roughly $(\frac{1}{m})^m$

PCP: $k \to poly(k)$ and $\delta(C) = 1/2$ (or some constant). We can get Σ to be exponentially small, $q = (log(k))^2$

6.2 Reed-Muller (or Hagamard) Codes

We want a binary alphabet ($q = 2$). $r = 0$ doesn't give us enough to work with, so let's try $r = 1$. The k, the number of coefficients, is $(m + 1)$ choose 1, which is just $(m + 1)$. So we have the coefficients $a_0, a_1, ..., a_m$ which gives the polynomial $A(x_1, ..., x_m) = a_0 + \sum_{i=1}^{m} a_i x_i$. We map $(m + 1)$ bits to 2^m bits, and achieve $\delta(C) = 1 - r/q = 1/2$, which is tight by the Plotkin Bound.

We can construct a code from a Hagamard matrix, H. A Hagamard matrix is an $n \times n$ matrix with entries ± 1. When we have a Hagamard matrix that satisfies $HH^T = n * I$ then we can create a nice error correcting code. We consider each row to be a codeword, giving n words of length n. ($log(n)$ bits $\rightarrow n$ bits) Note that the distance between any two rows of this matrix is 1/2.

To construct the Hagamard code we use the matrix $\left[\frac{H}{-H}\right]$. We know the distance between the first row of H and the first row of $-H$ must be n since they differ at every location. The the distance between another row i of H and the first row of $-H$ is n minus the distance between row 1 and row i of H, which is 1/2. So the distance comes to $n - 1/2$.