6.440 Essential Coding Theory Feb 25, 2013

Lecture 6

Lecturer: M. Sudan *Scribe: Timothy Chu*

1 Algebraic Codes Continued: Overview

The scope of these notes will cover

1. Reed Solomon, Reed Muller, and Hadamard Codes
2. Duals of (Linear) Codes
3. Properties of finite fields, which will lead us to the Wozencraft Ensemble of Codes (which we did not finish during this lecture). For now, think of our foray into the Wozencraft ensemble of codes as an exercise understanding finite fields and what they can do for you. However, we do not actually get to the explicit description of the Wozencraft ensemble.

1.1 Basic Notation Review

Recall from lecture 1 that we described codes using the notation $(n, k, d)_q; R, \delta$ where:

1. The alphabet Σ has $|\Sigma| = q$, and codewords reside in Σ^n.
2. $\triangle(C) \geq d$ (where $\triangle$ is the Hamming Distance metric).
3. $|C| \geq q^k$.

R and δ are defined as $\lim_{n\to\infty} \inf \frac{k}{n}$ and $\lim_{n\to\infty} \inf \frac{d}{n}$. Families of codes are described by the point (δ, R).

Previously, we introduced Reed Solomon and Reed Muller codes, which we will review shortly in these notes.

For algebraic codes presented in these notes, we work over a finite field. These notes assume some basic familiarity about finite fields; namely, that most familiar operations with polynomials over the field $\mathbb{R}$ or $\mathbb{Q}$ (long division, interpolation, factorization) apply to the finite field F_q of q elements.

2 General Ideas for Algebraic/polynomial Codes

- Message = coefficients of polynomials
- Encoding = Evaulation of polynomials at different values in , so

- Evaluation $\Rightarrow$ Encoding
- Interpolation $\Rightarrow$ Decoding from an error free code.

2.1 Review of Reed Solomon Codes

Recall the *singleton bound*, proven in lecture 3, which states that $k \leq n - d + 1$, or $R \leq 1 - \delta$.

Reed Solomon codes are algebraic (linear) codes which meet the singleton bound (as long as $q \geq n$) and in practice, they are fast to decode (there exist fast decoding techniques due to Berlekamp, which will not be elaborated here).

A Reed Solomon code is created by choosing a set $(\alpha_1, \alpha_2, \ldots \alpha_n)$ of distinct elements from $\mathbb{F}_q$. Message $m = (m_0, m_1, \ldots m_{k-1}) \in \mathbb{F}_q^k$ is associated with a polynomial $m(x) := m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \ldots + m_0$. m is encoded via $\text{Enc}(m) \equiv m(\alpha_1, \alpha_2, \ldots \alpha_n) \in \mathbb{F}_q^n$.

The code has dimension k and the polynomial $m(x)$ has degree $k-1$. The Reed Solomon code is linear (see note below), so if two code words $x, y \in \mathbb{F}^q$, then $x - y \in C$. Polynomials of degree $k-1$ in a finite field F_q have no more than $k-1$ roots, so $x - y$ has no more than $k-1$ bits with value 0. So $\triangle(x,y) \geq n - k + 1$ for all x and y, so $d \geq n - k + 1$. This inequality is the reverse of the Singleton bound, which holds for all codes, and thus Reed Muller meets the Singleton Bound exactly.

Note: it is not difficult to see from the encoding procedure that the Reed Solomon code is linear: specifically, the generator matrix $\in \mathbb{F}_q^{k \times n}$ of the code is

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1^1 & \alpha_2^1 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

3 Hadamard Code

The Hadamard code is a linear code which has Hamming Distance roughly meets the Plotkin bound from lecture 4. If $n = 2^j$, then $d = 2^{j-1}$ (good) and $R = \frac{j+1}{2^j}$ (bad). The Hadamard Codes are described at the ends of the notes of the previous lecture.

The Hadamard Code is a special case of a first-order Reed Muller code over the field $\mathbb{F}_2$.

4 Review of Reed Muller Codes

A Reed Muller code can be constructed based on polynomials of low degree on a finite field $\mathbb{F}_q$. Let m and r be integers where m is thought of to be much larger than r. Then the messages in this code are all m-multivariate polynomials P of degree at most r where each variable has individual degree at most $q-1$. The Reed Muller encoding of P is the evaluation of P on all $x \in F_q^m$ (the values of P over all choice of values of the m variables). There are $\binom{m+r}{m}$ elements in the message space.

For formal construction of the Reed Muller codes, see the scribe notes for lecture 5.

5 Performance of Reed Solomon and Reed Muller Codes

5.1 Reed Solomon Code

-Evaluate univariate polynomials of degree $< k$ over $\mathbb{F}_q$ and looking at the coefficients. Code is an $[n, k, n-k+1]_q$ code with $n \leq q$.

Performance: $r + \delta \to 1 + \epsilon$.

5.2 Reed Muller Codes

-Evaluate m-variate polynomials of degree $\leq r$ (where m is thought of as much greater than r) over $\mathbb{F}_q$, over the entire domain $\left(\mathbb{F}_q^m\right)$.

-Individual degree of each variable is $\leq q-1$ (We arrange for this since $x^q = x$ for all $x \in \mathbb{F}_q$).

Parameters of Reed Muller Codes

- $n = q^m$.
- $k = \binom{m+r}{m}$ if $r < q$, and $\left(\frac{r}{m}\right)^m, \binom{m}{r} \leq k \leq \binom{m+r}{r}$ for general r. If you think of r as constant while m goes to infinity, the lower bound of $\binom{m}{r}$ is pretty good.
- If $r = a(q-1) + b$ where $b < q$, then $\delta = q^{-a} \cdot \left(1 - \frac{b}{q}\right)$.

An explanation of for the values of r and δ can be found in the scribe notes of lecture 5. Let's now look at some examples, which we generate by plugging in choice values into the parameters of the Reed Muller codes.

Example 1 *Linear Polynomials ($r = 1$).*

- $n = q^m$
- $k = m + 1$
- $\delta = 1 - \frac{1}{q}$
- Number of codewords $= q^{m+1} = q \cdot n$

In the special case when $q = 2$, we have $\delta = \frac{1}{2}$. This is exactly the same construction as the Hadamard Code. Since the Reed Muller encodes messages as linear polynomials, we can write each such polynomial f as $\sum_{i=1}^{m} a_i x_i + a_0$, where a_i are constants and the x_i's are the m variables in the Reed Muller code.

We encode f by evaluating it at all points $\{\alpha_1, \alpha_2, \ldots \alpha_m\}, \alpha_i \in \mathbb{F}_2$. Then the encoding is

$\sum_{i=1}^{m} a_i \alpha_i + a_0$ evaluated at all $\alpha_1, \alpha_2, \ldots \alpha_m \in \mathbb{F}_2^m$

Let's look at another example.

Example 2 *Bivariate polynomials*

Parameters: We establish that $m = 2, r = \frac{q}{2}$. Then

- $n = q^2$
- $k = \binom{m+r}{r} = \binom{\frac{q}{2}+2}{2} \sim \frac{q^2}{8} = \frac{n}{8}$
- $R = \frac{k}{n} = \frac{1}{8}$
- If $q = \sqrt{n}$, then $\delta = \frac{1}{2}$.

This example demonstrates the tradeoff between rate and distance versus alphabet size. In this example, our alphabet size is $\sqrt{n}$, which is much smaller than n, the minimum alphabet size required for Reed Solomon code (which is the Reed Muller code for single variate polynomials): Reed Solomon obtains the bounds $\frac{1}{2}$ and $\frac{1}{2}$ for R and δ respectively, instead of $\frac{1}{8}$ and $\frac{1}{2}$. But q would be at least n in Reed Solomon as opposed to $\sqrt{n}$ in this example.

Because the alphabet of the codes are finite fields, and it is a reasonable problem to ask how one might construct large finite fields, the size of the alphabet is a relevant consideration in coding theory. (Maybe).

Now let's look at an example that is of significant interest in computational complexity theory.

Example 3 *Let* $m = \frac{\log n}{\log \log n}, q = \log^2 n, r = \frac{1}{2} \log^2 n$

(Example analysis incomplete)

Then

- $\delta = \frac{1}{2}$
- $q = \log^3 n$.

The rate goes to 0, but we're interested in seeing how quickly it goes to 0. Note that $k \geq n^{1/3}$. But how should we precisely bound k? We would say $k \geq \left(\frac{r}{m}\right)^m = (\log n)^{\frac{\log n}{\log \log n}}$.

$n = q^m$, and so $k \geq \Omega\left(n^{1/2}\right)$

Moral of the story:

- $\delta = 1/2$ (which is really good), $|\Sigma| = \text{polylog}(n)$.

- k vs n relationship is polynomial.

(Place picture here with R and δ on the axes: Gilbert Varshamov bound, for $q = 2$).

6 Duals of (Linear) Error Correcting Codes

If $C = [n, k, d]_q$ code over $\mathbb{F}_q$, the **dual** of the code is the set

$$C^{\perp} = \{x \in \mathbb{F}_q^n | \forall y \in C, \langle x, y\rangle = 0\}$$

where $\langle x, y\rangle$ is the standard dot product.

Because $C^{\perp}$ is the space of elements perpendicular to C, the dimension of $C^{\perp}$ is $n - k$. Thus $C^{\perp} = [n, n - k, d]_q$, where we have not said anything about d yet.

Lemma 4 *If G is the generator matrix for C and H is the parity check matrix, then G^T is the parity check matrix for $C^{\perp}$ and H^T is the generator matrix.*

Proof Let x be an element of our message space. We note that the space spanned by xH^T has the right number of dimensions to be a candidate for the space $C^{\perp}$, and $H^T G^T = (GH)^T = 0$. It remains to show that $\langle xH^T$ and $x'G$ are perpendicular for all x, x' in the message space.
However,

$$\langle xH^T x'G\rangle = xH^T G^T x' = x(GH)^T x' = 0$$

completing the proof of our lemma. ■

Now we would like to figure out the distance d in the dual code of C. Let's examine a few examples and their duals to try and figure this out.

Example 5 *Reed Solomon Code*

The Reed Solomon code is a *Maximal Distance Separable (MDS) code*, meaning that the singleton bound is tight. Recall that the Singleton bound states that $A_q(n, d) \leq q^{n-d+1}$ where $A_q(n, d)$ is the number of codewords in an $\mathbb{F}_q$ *block code* of length n with minimum distance d.

Lemma 6 *The dual of Linear MDS code is MDS.*

Proof Suppose C is a code with generator matrix G with columns $c_1, c_2, \dots c_n$. This is a k by n matrix. Note that in an MDS code, every linear combination of the rows has Hamming weight at least $n - k + 1$.

By lemma 4, $C^{\perp}$ has G^T as its parity check matrix. We wish to show now that if $C = [n, k, n - k + 1]$ code, then $C^{\perp} = [n, n - k + 1, k + 1]$ code. That is, every subset of k columns of G, the generator matrix, is linearly independent. The proof is as follows:

Proof Suppose that some k columns are linearly dependent. Consider the k by k submatrix M formed by these columns. Since the columns are linearly dependent, the rank of M is less than k so the rows of M have some linearly dependence. Therefore there exists some linear combination of the rows of M that sums to 0, so we can use this same linear combination on the rows of G whose sum has at least k 0's, and thus has Hamming weight $\leq n-k$. But we've established that any linear combination of the rows of G in an MDS code must have Hamming weight at least $n-k+1$. Contradiction.

■

We've shown that the dual of an $[n, k, n-k+1]_q$ code is an $[n, n-k+1, k+1]_q$ code, which is MDS as desired.

■

However, it turns out the dual of Reed Solomon codes are still Reed Solomon codes, so we do not get any new codes from taking the dual of a Reed Solomon code.

Now let's turn our attention to the dual of Reed Muller codes.

Lemma 7 *The dual of a Reed Muller Code is Reed-Muller.*

If we have a Reed Muller code in the primal with $\mathbb{F}_q, m, r$, the dual of the code has the same alphabet $\mathbb{F}_q$, the number of variables will remain m. In the initial Reed Muller code, we took the monomials of maximum degree r. If Reed Muller code C is generated by all monomials of degree $< r$, then we conjecture that $C^{\perp}$ is generated by the set of monomials with degree $< m(q-1) - r$. We leave it as an exercise to verify that the sum of the dimensions of C and our candidate $C^{\perp}$ is indeed q^m.

Before we prove this conjecture, let's look at a picture.

.... (Incomplete section) Suppose $r = \frac{3}{2}q$. Let's look at all the integer degree monomials we could have.

(Picture, bivariate polynomials, plotting the permissable exponents of x and y. We look at the area of the square in the graph. The area of the dual correpsonds to the complement of the primal in the picture).

....

Now let's go back to our proof that the dual of a Reed Muller code C is Reed Muller, and in fact this dual is generated by the set of monomials with degree $< m(q-1) - r$. To do this, we must first prove a property of finite fields. Namely, summing a non-zero monomial in m variables with degree less than $m(q-1)$ over all possible values $\mathbb{F}_q^m$ gives 0. We leave it as an exercise to show that it suffices to prove the next claim.

6.1 Some Properties of Finite Fields

Claim 8 *If $P, Q \in \mathbb{F}_q[x]$ and deg(P) + deg $(Q) < q-1$. Then $\sum_{\alpha \in \mathbb{F}_q} P(\alpha) \cdot Q(\alpha) = 0$ (in $\mathbb{F}_q$).*

Proof Note: all equalities in the proof are for $\mathbb{F}_q$.

We can expand $P(x)Q(x)$ as the sum of monomials with degrees less than $q-1$. Therefore if we can show that $\sum_{\alpha\in\mathbb{F}_q} x^n = 0$ for all $n < q-1$, then adding some combination of these proves our claim.

Lemma 9 $\sum_{\alpha\in\mathbb{F}_q} x^n = 0$ *when* $n < q-1$.

Proof Let $\alpha_1, \alpha_2, \ldots \alpha_q$ be all the elements in F_q. Because $\mathbb{F}$ is a field (and thus has inverses), we can show without too much difficulty that for any $\beta \in \mathbb{F}_q$ that $\beta\alpha_1, \beta\alpha_2, \ldots \beta\alpha_n$ is a permutation of $\alpha_1, \ldots \alpha_n$ for any $\beta \in \mathbb{F}_q$. Namely, this is true when β is a primitive root of $\mathbb{F}_q$, meaning that $\beta^x = 0 \Leftrightarrow x \equiv 0 \pmod{q-1}$. (The existence of primitive roots is a well-known property of finite fields, and is asserted here without proof).

Since $\beta\alpha_1, \beta\alpha_2, \ldots \beta\alpha_n$ is a permutation of $\alpha_1, \ldots \alpha_n$ for any $\beta \in \mathbb{F}_q$, then

$$\sum_{\alpha\in\mathbb{F}_q} (\beta\alpha)^n = \sum_{\alpha\in\mathbb{F}_q} \alpha^n$$

$$\Rightarrow \sum_{\alpha\in\mathbb{F}_q} (\beta^n - 1)\alpha^n = 0$$

Since $n < q-1$, then $\beta^n \neq 1$ and thus $\beta^n - 1 \neq 0$. Therefore $\sum_{\alpha\in\mathbb{F}_q} \alpha^n = 0$ as desired. ■

The lemma is proven, and therefore our initial claim is proven as well. ■

Additionally, $\sum_{i=0}^{q-2} \alpha^i = 0$ if $\alpha \neq 1$. This comes from noting that $\sum_{i=0}^{q-2} \alpha^i = \frac{\alpha^{q-1}-1}{\alpha-1}$ for $\alpha \neq 1$, and it is a well known fact of finite fields that any $\alpha \neq 0 \in F_q$ satisfies $\alpha^{q-1} - 1 = 0$.

6.2 Return to Proof that Dual of Reed Muller is Reed Muller

Recall that the generator matrix G for a Reed Muller code with parameterse m and r has columns corresponding to elements of F_q^m representing each possible input into our message (written as coefficients of a polynomial), rows corresponding to a monomial of degree $< r$, and entries corresponding to the valuation of the elements of F_q^m corresponding to the column evaluated on the m-variate monomial of degree $< r$ corresponding to the rows.

We assert that a valid generator matrix $G^\perp$ for $C^\perp$ consists of the same construction but with each row corresponding to monomials of degree $< m(q-1) - r$. The dimensions are correct, so from lemma 4, we only need to check that $G\left(G^\perp\right)^T = 0$. However, each entry of $G\left(G^\perp\right)^T$ is a sum of some monomial of degree $< m(q-1)$, and so the proofs in the previous subsection allow us to say each entry is indeed 0. Therefore the generator matrix for the Reed Muller code consisting of polynomials of degree $< m(q-1) - r$ is indeed a parity check matrix of appropriate dimension for our original Reed Muller code consisting of polynomials of degree $< r$. This shows that the dual of a Reed Muller code is indeed a Reed Muller code.

6.3 How do we describe finite fields?

Finite fields of size q are generally described by roots of $\alpha^q - \alpha \forall \alpha \in \mathbb{F}_q$. That such roots form a finite field of size q will not be proven here, but the result can be found in any texts on fields. However, for the purpose of computing, we would like to be able to explicitly construct finite fields of a particular size.

7 Constructive Codes

7.1 Case: Prime Fields: $q = p$

When q is prime, we can represent elements of $\mathbb{F}_q$ by integers from 0 to $p-1$, adding and multiplying them using the rules of $\pmod{p}$. However, if we want to specify the approximate size of our field, then we are out of luck at this stage.

Suppose we want to find a prime between 2^{l-1} and 2^l. We can find such a prime deterministically in time 2^l and we can find it in randomized time with $poly(l)$. (Proofs omitted).

7.2 Non-prime Fields

It's well known that the number of elements of a field must be the power of a prime.

Suppose $q = p^t$, given p and t. How can I construct F_q?

There exist methods that can "find" F_q in deterministic time $poly(t, p)$ and randomized time $poly(t, \log p)$.

The most 'hands-on' representation of F_{p^t} is constructed by selecting an irreducible polynomial $g(x) \in F_p[x]$ where deg $g = t$ and g is monic. It is a known fact of finite fields that F_q can be represented as $F_p[x]/g(x)$. Finding this polynomial g is used to obtain the bounds listed above. (Exact method omitted).

There are a few other methods we can try.

One approach we might try is to find a class of irreducible polynomials and use them to generate our finite field. One such example are the polynomials $x^{3^{2s}} + x^{3^s} + 1$, which are irreducible for all s. (No proof given). This is not great, since the degrees of these polynomials are powers of 3, which this method can only construct fields of the form $F_p^{3^{2s}}$. But we can get a reasonable approximation with a multiplicative factor of 3. These polynomials are nice to work with since they are very sparse, and this gives us a very concrete way to find a field of some approximate degree.

Let's try to be a little more abstract, and develop some cleaner ways of thinking about finite fields. You should always keep the following two methods in mind.

Method 1 An incomplete representation.

F_{p^t} is isomorphic to F_p^t with a linear isomorphism that preserves multiplication by any element of F_p (which has a natural inclusion into F_{p^t} as the multiples of p^{t-1} in F_{p^t}. Such an isomorphism is called a F_p *linear isomorphism.* The isomorphism respects addition and multiplication by F_p. To spell it out formally,

Let V be F_p linear isomorphism from F_{p^t} to F_p^t. Then

$$V(\alpha + \beta) = V(\alpha) + V(\beta)$$

and for any $\sigma \in F_p$, $\sigma V(\alpha) = V(\sigma\alpha)$

Method 2 Complete representation

The idea is to represent $\alpha \in F_{p^t}$ as a set of linear maps $T(\alpha)$ operating on $\mathbb{F}_p^t$, where T satisfies the properties that

- $T(\alpha)T(\beta) = T(\alpha\beta)$
- $T(\alpha) + T(\beta) = T(\alpha + \beta)$.

(Bijective?) Representations $T(\alpha)$ exist (though we will not prove this) and can be represented as t by t matrices $M(\alpha)$ with entries in the field F_q. The idea of representing F_{p^t} as a linear map is an idea inherited from the mathematical discipline of Representation Theory.

8 Teaser for next time

(I forgot to copy down what went here).