CSE 533: Error-Correcting Codes **(Autumn 2006)**

Lecture 8: Reed Muller Codes

Lecturer: Venkat Guruswami *Scribe: Prasad Raghavendra*

In the previous lecture, we defined Reed Muller codes and their variants. Today, we will study an efficient algorithm for decoding Reed Muller code when the number of errors are less than half the distance. Then we shall return to our original goal of constructing explicit codes with constant relative distance and rate. Towards this, we will convert Reed Solomon codes in to binary codes.

1 Recap

Let $R(r,m)$ denote the r^{th} order Reed Muller code. Therefore the messages consist of multilinear polynomials in variables $X_1, X_2 \ldots X_m$ of degree at most r. Recall that the length of the code $n = 2^m$, the dimension $k = \sum_{i=0}^{r} \binom{m}{i}$ and the distance is 2^{m-r}. i.e $R(r,m)$ is a $[2^m, \sum_{i=0}^{r} \binom{m}{i}, 2^{m-r}]_2$ linear code.

Some interesting special cases of the Reed Muller code:

- With $r = \frac{m}{2}$, $R(r,m)$ gives a code with rate $\frac{1}{2}$ and distance $d = 2^{\frac{m}{2}} = \sqrt{n}$. Although this is not constant distance, it is a fairly non-trivial code with a good rate.
- With $r = 1$, $R(r,m)$ yeilds a linear code with parameters $[2^m, m+1, 2^{m-1}]$. Further with $r = 1$, a code word consists of the evaluation of a degree 1(linear) function over $\mathbb{F}_2^m$. Hence $R(1,m)$ code consists of the Hadammard codes and their complements.

2 Reed's algorithm

2.1 Notation

We will use $X = (x_1, \ldots, x_m)$ to denote an element of $\mathbb{F}_2^m$. For a subset $S \subseteq \{1, \ldots, m\}$, let X_S denote the restriction of X to indices in S. i.e $|S|$-dimensional vector consisting of x_i for $i \in S$. Denote by $\overline{S}$ the complement of a set S.

Let $R(r,m)$ be a binary Reed Muller code. Let $f : \mathbb{F}_2^m \to \mathbb{F}_2$ be given as a table of values. If f is a codeword then f is a polynomial $P(x_1, \ldots x_m)$ of degree atmost r. For a general function f, define the distance from a polynomial $P(x_1, \ldots, x_m)$ as follows:

$$\Delta(f,p) \quad = \quad |\{a \in \mathbb{F}_2^m | f(a) \neq P(a)\}|$$

2.2 Algorithm

The input consists of a message which is atmost $\frac{d}{2}$ away from a codeword. In particular, we are given a function $f : \mathbb{F}_2^m \to \mathbb{F}_2$ such that there exists a polynomial P of degree r with $\Delta(f, P) < \frac{2^{m-r}}{2}$. The goal of the algorithm is to output polynomial P.
Let us say P is of the form

$$P(X) = \sum_{S \subseteq \{1,\ldots,n\}, |S| \leq r} c_S \prod_{i \in S} x_i$$

For a subset $S \subset \{1, 2, \ldots, m\}$, define the polynomial

$$R_S(X) = \prod_{i \in S} x_i$$

Observation 2.1. *For any proper subset $T \subset S$, we have*

$$\sum_{a \in \mathbb{F}_2^{|S|}} R_T(a) = 0$$

Proof: Let $i \in S - T$, then we can write the above summation as follows :

$$\begin{aligned} \sum_{a \in \mathbb{F}_2^{|S|}} R_T(a) &= \sum_{a \in \mathbb{F}_2^{|S|}, a_i = 0} R_T(a) + \sum_{a \in \mathbb{F}_2^{|S|}, a_i = 1} R_T(a) \\ &= 2 \sum_{a \in \mathbb{F}_2^{|S|}, a_i = 0} R_T(a) = 0 \end{aligned}$$

□

Observation 2.2. *For any set S,*

$$\sum_{a \in \mathbb{F}_2^{|S|}} R_S(a) = 1$$

Proof: Recall that $R_S(a) = \prod_{i \in S} x_S$. Hence for all but one of the values $a \in \mathbb{F}_2^{|S|}$, $R_S(a)$ is 0. Therefore the above identity follows. □

Lemma 2.3. *For all $b \in \mathbb{F}_2^{m-r}$, and a degree r polynomial P the following is true*

$$\sum_{a \in \mathbb{F}_2^m, a_{\overline{S}} = b} P(a) = c_S$$

Proof : Let P_b be the polynomial obtained by substituting $b \in \mathbb{F}_2^{m-r}$ for variables $\{x_i | i \in \overline{S}\}$. Hence P_b is of the following form :

$$P_b(X) = c_S R_S(X) + \sum_{T \subset S} a_T R_T(X)$$

for some a_T.

$$\begin{aligned} \sum_{a \in \mathbb{F}_2^m, a_{\overline{S}} = b} P(a) &= \sum_{y \in \mathbb{F}_2^r} P_b(y) \\ &= c_S \sum_{y \in \mathbb{F}_2^r} R_S(y) + \sum_{T \subset S} a_T \sum_{y \in \mathbb{F}_2^r} R_T(y) \end{aligned}$$

Using observation 2.1,2.2 with the above equation, it follows that

$$\sum_{a \in \mathbb{F}_2^m, a_{\overline{S}} = b} P(a) = c_S$$

□

The lemma 2.3 suggests an algorithmic method to obtain the coefficients c_S of the polynomial P. Simply sum the value of the polynomial for all $a \in \mathbb{F}_2^n$ with $a_{\overline{S}} = b$ for some $b \in \mathbb{F}_2^r$. Further for each of the 2^{m-r} choices for b, the sum ranges over disjoint set of points in $\mathbb{F}_2^m$. That is the sets $\{a \in \mathbb{F}_2^m | a_{\overline{S}} = b\}$ are all disjoint.

By our assumption, f differs from the polynomial P in atmost $2^{m-r-1} - 1$ positions. Hence for atleast $2^{m-r} - (2^{m-r-1} - 1)$ values of b we have

$$\sum_{a \in \mathbb{F}_2^m, a_{\overline{S}} = b} f(a) = \sum_{a \in \mathbb{F}_2^m, a_{\overline{S}} = b} P(a) = c_S$$

Out of the 2^{m-r} sums of the form $\sum_{a \in \mathbb{F}_2^m, a_{\overline{S}} = b} f(a)$, more than $\frac{1}{2}$ fraction of them are equal to c_S. Thus a natural way to compute c_S is to compute majority of all these sums. Towards finding the other lower degree terms in P, reduce the problem as follows :

$$\begin{aligned} f' &= f - \sum_{|S|=r} c_S R_S(x) \\ P' &= P - \sum_{|S|=r} c_S R_S(x) \end{aligned}$$

Since f' is atmost $2^{m-r} < 2^{m-(r-1)}$ away from a degree $r-1$ polynomial P', the above procedure can be used to find the degree $r-1$ terms of P'. Hence iteratively, all the coefficients c_S of P can be computed.

The formal description of the algorithm is given below

Reed's Algorithm

Input : A function $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$ such that there exists a polynomial P of degree r with $\Delta(f, P) < \frac{2^{m-r}}{2}$.
Output : The polynomial P.

$t \leftarrow r, F \leftarrow f, P \leftarrow 0$
While $t \geq 0$ do
For each $S \subset \{1, \ldots m\}$ with $|S| = t$

$$c_S = \text{Majority over all } b \text{ of } \big(\sum_{a \in \mathbb{F}_2^m, a_{\overline{S}} = b} F(a) \big)$$

$$P = P + c_S \prod_{i \in S} x_i$$

For each $x \in \mathbb{F}_2^m$

$$F(x) = F(x) - c_S \prod_{i \in S} x_i$$

$t \leftarrow t - 1$
Output P

3 Extension Fields

For every prime p, the field $\mathbb{F}_p$ consists of $\{0, 1, \ldots, p-1\}$ with addition and multiplication modulo p. For any integer k, the extension field $\mathbb{F}_{p^k}$ is obtained as follows:

Consider the ring $\mathbb{F}_p[x]$ consisting of polynomials over one variable(x) with coefficients over $\mathbb{F}_p$. Let $q(x)$ be an irreducible degree k polynomial in $\mathbb{F}_p[x]$. Since $q(x)$ is irreducible, for any other polynomial $r(x)$ one of the following two cases is true

- $gcd(q(x), r(x)) = 1$: By Euclid's algorithm, we can find $a(x), b(x)$ such that $a(x)r(x) + b(x)q(x) = 1$. i.e there exists $a(x)$ such that $a(x)r(x) \equiv 1 \mod q(x)$
- $gcd(q(x), r(x)) = q(x)$:In this case $a(x) = 0 \mod q(x)$.

That is each polynomial $r(x)$ is either $0 \mod q(x)$ or has an inverse $a(x)$. Hence the set of polynomials modulo $q(x)$ form a field. This field consists of all polynomials over $\mathbb{F}_p$ with degree less than k and is denoted by $\mathbb{F}_p[x]/(q(x))$. Clearly there are p^k elements in this field. Further it can be shown that the field obtained from different degree k irreducible polynomials all behave the same way, i.e are isomorphic to each other.

Notice that the set of polynomials $\mathbb{F}_p[x]$ form a vector space over $\mathbb{F}_p$. Hence the field $F_p[x]/(q(x))$ also form a vector space over the field $\mathbb{F}_p$. Every polynomial of degree less than k can be represented naturally as a length k vector of $\mathbb{F}_p$. This implies a representation/mapping of elements of the extension field $F_p[x]/(q(x))$ as k-dimensional $\mathbb{F}_p$ vectors. So we have a mapping

$$\phi : \frac{F_p[x]}{(q(x))} \to \mathbb{F}_p^k$$

Infact the above mapping ϕ is a linear mapping in the following sense : For any two elements $r(x), s(x)$ of $\frac{F_p[x]}{(q(x))}$ we have

$$\phi(r(x) + s(x)) = \phi(r(x)) + \phi(s(x))$$

In summary, elements of the extension field can be represented as k-dimensional vectors over $\mathbb{F}_p$ in a way that preserves linearity.

3.1 Irreducible Polynomials

Explicit irreducible polynomials are necessary to construct extension fields. It can be shown that there is an abundance of irreducible polynomials, that is a random degree k polynomial is irreducible with high probability. Explicit construction of irreducible polynomials are also known.

Lemma 3.1. *For each $k > 0$ the following polynomial over $\mathbb{F}_2$ is irreducible*

$$P(x) = x^{2 \cdot 3^{k-1}} + x^{3^{k-1}} + 1$$

Proof: Suppose not, let us say there are polynomials $Q(x), R(x)$ over $\mathbb{F}_2$ such that $P(x) = Q(x)R(x)$. Observe that

$$x^{3^k} - 1 = (x^{3^{k-1}} - 1)(x^{2 \cdot 3^{k-1}} + x^{3^{k-1}} + 1)$$

Let $\mathbb{F}_2^*$ be the algebraic closure of $\mathbb{F}_2$. All our arguments will be over $\mathbb{F}_2^*$, of which $\mathbb{F}_2$ is a subfield. Let ζ be the 3^k primitive root of 1. So we have

$$\begin{aligned} \zeta^{3^k} &= 1 \\ \zeta^{3^{k-1}} &\neq 1 \end{aligned}$$

Since $x^{3^k} - 1 = (x^{3^{k-1}} - 1)P(x)$ we get $P(\zeta) = 0$. Hence either $Q(\zeta) = 0$ or $R(\zeta) = 0$, without loss of generality we can assume $Q(\zeta) = 0$. Recall that $\Phi : x \to x^2$ gives an automorphism of the $\mathbb{F}_2^*$ known as the Frechet's mapping. That is for any two elements x, y we have

$$\begin{aligned} \Phi(x) + \Phi(y) &= \Phi(x + y) \\ \Phi(x) * \Phi(y) &= \Phi(xy) \end{aligned}$$

Further the elements of $\mathbb{F}_2 = \{0, 1\}$ are fixed by the mapping $\Phi : x \to x^2$. In particular the coefficients of the polynomial Q are fixed by Φ. Therefore if we apply Φ on the equation $Q(\zeta) = 0$, we get $Q(\zeta^2) = 0$. Applying this repeatedly, we can conclude that $\{\zeta, \zeta^2, \zeta^4, \ldots, \zeta^{2^t}\}$ are all roots of Q. Now let us count the number of distinct elements of the form ζ^{2^t}. Let $n = 3^k$, then the Euler's totient function $\phi(n) = 2 \cdot 3^{k-1}$. By Euler's theorem

$$2^{\phi(n)} = 1 \mod n$$

Clearly this implies that $\zeta^{2^{\phi(n)}} = \zeta$. Since ζ is a primitive n^{th} root of 1, $\zeta^{2^i} = \zeta$ implies $2^i \equiv 1 \mod n$. It can be shown that 2 is a primitive root modulo $n = 3^k$. Therefore for any $i < \phi(n)$, $2^i \neq 1 \mod n$. Hence all the elements $\zeta, \zeta^2, \ldots, \zeta^{2^{\phi(n)-1}}$ are distinct. Recall that all these elements are roots of Q. But since $P(x) = Q(x)R(x)$, the degree of $Q < \phi(n)$. This is a contradiction, since Q cannot have more than roots than its degree. □

4 Reed Solomon Codes

Reed Solomon codes are explicit linear codes that are optimal in that they meet the Singleton bound. However these codes are over an alphabet of size $q \geq \log n$. Now let us try to obtain good codes over a smaller alphabet(say $\mathbb{F}_2$) using Reed Solomon codes.

A natural thing to do is to represent each alphabet by a binary string of length $\log q$. Let us assume q is a power of 2, i.e $q = 2^t$. This can be arranged for by choosing a Reed Solomon code over an extension field of a degree t $\mathbb{F}_2$-irreducible polynomial. Then as observed earlier, there is a natural representation of elements of the extension field as t-dimensional vectors over $\mathbb{F}_2$. Recall that this representation preseves linearity. This is good news, since we not only obtained a binary code but also a binary linear code.

Let us investigate the parameters of the binary linear code obtained. Suppose we started with a $[n, k, d]_{2^t}$ Reed Solomon code, then the length of our new code is nt, and dimension is kt. We know that changing d symbols in the original code can change one codeword to another. It is possible that, each of these d symbols need to be changed at just one bit. Hence the distance still remains d. Therefore the code obtained has parameters $[nt, kt, d]_2$.

However Reed-Solomon codes are popular in practice because errors in real life channels tend to be burst errors. Hence it is more likely that all the d errors are situated together in the same symbol, or very few symbols. In this case, clearly the codeword can be recovered.

We still have not reached our goal of constructing a family of binary codes with constant rate and constant distance.

An Idea :

The conversion from Reed Solomon codes to binary linear codes failed because by changing one bit we could change the original symbol. In other words, two symbols could differ at just one bit in their representation. Hence we should represent the original alphabet such that different symbols differ at several places. This is exactly same as the original problem of error correcting codes. So the idea would be to encode the symbols of the large alphabet using code words of an error correcting code. In the next class, we will use this idea to construct binary codes with constant rate and distance.