(67902) Topics in Theory and Complexity Nov 2, 2006

Lecture 3

Lecturer: Irit Dinur *Scribe: Beny :-)*

1 Reed Solomon codes - Reminder

Given a finite field $\mathbb{F}_q$ and n distinct elements $\alpha_1, \ldots, \alpha_n$ in $\mathbb{F}_q$, a Reed Solomon code with parameters n, q, k encodes words in $\mathbb{F}_q^k$ into words in $\mathbb{F}_q^n$. It is obtained by the encoding function $RS(a) := \langle P_a(\alpha_1), \ldots, P_a(\alpha_n)\rangle$, where the polynomial P_a is defined for every vector $a = (a_0, \ldots, a_{k-1}) \in \mathbb{F}_q^k$ as follows: $P_a(x) = \sum_{j=0}^{k-1} a_j x^j$.

This code has a very good minimum distance of $n - k + 1$. If we choose for example $k = n/2$, we get a distance of about $n/2$ and a rate of $1/2$.

The coding algorithm of RS codes is just the evaluation of the polynomial P_a at n points. Since it is a linear code, we can also think of encoding as multiplying by its generating matrix (a Vandermonde matrix).

If $p \in \mathbb{F}_q^n$ is a codeword, then $p = \langle P_a(\alpha_1), \ldots, P_a(\alpha_n)\rangle$ for some $a \in \mathbb{F}_q^k$. Thus, we can think of p as another description of the polynomial P_a. While a describes P_a by giving its coefficients, p describes it by giving its value at n different points. This description is unambiguous since P_a is a polynomial of degree less than k, and is thus uniquely defined by its value over any k or more points. In the rest of this lecture we will view both input words a, and code words p, as polynomials, and in both cases we actually refer to P_a.

1.1 Decoding in the presence of erasures

We are given a vector $r \in (\mathbb{F}_q \cup \{?\})^n$ of distance at most $e = (n-k)/2$ from the code. We want to find a word p in the code which is closest to r.

Intuitively, r was obtained from a legal codeword by erasing certain letters; these letters are designated by "?". Our goal is to recover the erased letters.

Note that since the distance between two different words in the code is at least $n - k + 1$, it is not possible for two codewords to be at distance at most e from r, so p is uniquely defined. If we look at p and r as defining polynomials of degree less than k over $\mathbb{F}_q$, we note that they must describe the same polynomial since they agree on at least k points. Thus, to obtain p, all we need to do is interpolate using the unerased letters in r.

2 Decoding Reed Solomon Codes in the presence of errors

In this case we are given a vector $r \in \mathbb{F}_q^n$. I.e., the errors present in r are the result of replacing one letter in $\mathbb{F}_q$ with another. As before, we assume r is of distance at most $e = (n-k)/2$ from the code, and we want to find the (unique) word p in the code which is closest to r.

Since we do not know which of the letters in r are correct and which are erroneous, we can not use interpolation as before.

2.1 The Berlekamp-Welch Algorithm

Given r, we look for two non-zero polynomials E and N over $\mathbb{F}_q$ such that:

1. E is of degree $\leq e$
2. N is of degree $\leq e + k - 1$
3. For every $1 \leq i \leq n$ we have that $N(\alpha_i) = r_i E(\alpha_i)$

Once we find E and N (we will see later how to do this), the requested codeword p is just N/E. As we will show, if r is indeed of distance $\leq e$ from the code, E will divide N with no remainder.

The idea behind this algebraic magic is that E is an Error-Locator polynomial in the sense that it evaluates to zero exactly for those α_i for which an error occurred. I.e., for every $1 \leq i \leq n$ we have that $E(\alpha_i) = 0$ iff $r_i \neq p_i$.

Theorem 1 *(The Berlekamp-Welch Algorithm is correct and efficient)*

1. *There exist polynomials E and N satisfying the requirements 1–3 set above.*
2. *E, N and N/E can be efficiently calculated.*
3. *The solution N/E is indeed the closest codeword to r.*

Proof First we will prove the existence of E and N satisfying requirements 1–3 above. Let p be the codeword closest to r. Define E to be the polynomial $E(x) := \prod_{\{1 \leq i \leq n \mid r_i \neq p_i\}}(x - \alpha_i)$, and N to be the polynomial $N(x) := p(x)E(x)$. Note that the degree of E is exactly the distance between r and p. Thus, $deg(E) \leq e$, and $deg(N) = deg(p) + deg(E) \leq k - 1 + e$. Moreover, E divides N, and the quotient N/E is exactly the codeword p.

Observe that by the definition of the RS encoding function, for indices i for which $r_i = p_i$ we have that $r_i = p(\alpha_i)$. On the other hand, for indices i for which $r_i \neq p_i$ the error locating polynomial E satisfies $E(\alpha_i) = 0$. Combining these two observations we obtain that for every $1 \leq i \leq n$ it holds that $N(\alpha_i) = r_i E(\alpha_i)$, as requested.

We now prove that E, N, and N/E can be computed efficiently. From our degree bounds on E and N it follows that we can write $E(x) = \sum_{j=0}^{e} E_j x^j$, and $N(x) = \sum_{j=0}^{e+k-1} N_j x^j$. Our problem is to find the coefficients $E_0, \ldots, E_e$ and $N_0, \ldots, N_{e+k-1}$. Since $\alpha_1, \ldots, \alpha_n$ and r are known, for every $1 \leq i \leq n$ the equality $N(\alpha_i) = r_i E(\alpha_i)$ yields a linear equation in $E_0, \ldots, E_e$ and $N_0, \ldots, N_{e+k-1}$. All we need to do to find E and N is to solve a homogenous system of n linear equations in $(e+1)+(e+k) = 2e+k+1$ variables. Recall that $e = (n-k)/2$. Thus, we have $n+1$ variables and n equations, and we can find in time $O(n^3)$ a non-trivial solution. Note that we do not know the linear dependency between the equations, so the degree of the system may be actually less than n. Once N and E are found, calculating N/E is obviously easy.

It remains to show that even though there are infinitely many polynomials N and E satisfying our set of linear equations, the solution N/E is unique. Since we already demonstrated the existence of N and E for which N/E is exactly the required codeword p, this will complete the proof. Let N, E and N', E' be two pairs of polynomials satisfying our degree bounds and linear equations. We claim that $N/E = N'/E'$, or equivalently, that the product polynomials NE' and $N'E$ are equal. For every $1 \leq i \leq n$ we have that $N(\alpha_i)E'(\alpha_i) = r_i E(\alpha_i)E'(\alpha_i) = r_i E'(\alpha_i)E(\alpha_i) = N'(\alpha_i)E(\alpha_i)$. Observe that NE' and $N'E$ are polynomials of degree at most $e + (e + k - 1) = 2e + k - 1 = n - 1$, that agree on n points, so they must be equal. ■

3 Reed Muller Codes

Recall that a main problem with Reed Solomon codes was the alphabet size q. Since we needed n distinct elements in $\mathbb{F}_q$, we had to have $q \geq n$, i.e., the alphabet had to be at least as large as the code length. One way to get a better alphabet size is to use Reed Muller codes. These codes, which can be viewed as a generalization of Reed Solomon codes (though historically preceding them) use multivariate polynomials instead of univariate polynomials.

To get a feeling of what we can gain, lets look at the following example using bivariate polynomials. Let $a = (a_{0,0}, \ldots, a_{l,l})$ be a vector of length $(l+1)^2$ over $\mathbb{F}_q$, and let S be a subset of $\mathbb{F}_q$ such that $|S| > l + 1$. Define P_a to be the bivariate polynomial $P_a(x, y) := \sum_{i=0}^{l} \sum_{j=0}^{l} a_{i,j} x^i y^j$. Our encoding

function C encodes messages in $\mathbb{F}_q^{(l+1)^2}$ into codewords in $\mathbb{F}_q^{|S|^2}$ by giving the value of P_a for all the vectors in S^2. I.e., $C(a) := \langle P_a(\alpha, \alpha')_{\alpha,\alpha' \in S} \rangle$. Note that this code has codewords of length $n = |S|^2$, while our constraint on the size of the alphabet q is just $q \geq |S|$. Thus, the alphabet size is the square root of the size required by a Reed Solomon code.

3.1 Definition of General Reed Muller codes

We now look at a generalization of the idea presented in the example above, using polynomials of arbitrary maximal degree t, with arbitrarily many variables m. This results in what is known as Reed Muller codes.

Given t and[1] m, let $N_{t,m} = \{\beta \in \mathbb{N}^m \mid \beta_1 + \beta_2 + \ldots + \beta_m \leq t\}$ be the set of vectors in $\mathbb{N}^m$ satisfying that the sum of their components is at most t. Intuitively, every vector β in $N_{t,m}$ represents a possible assignment of values for the powers of the m variables $x_1, \ldots, x_m$ without exceeding the combined degree t. Note that the corresponding set of monomials $\{P \mid P = x_1^{\beta_1} x_2^{\beta_2} \cdots x_m^{\beta_m} \text{ for some } \beta \in N_{t,m}\}$ forms a basis for the polynomials of degree at most t in m variables. As a notational convenience, we write $M_{t,m}$ to designate the size of $N_{t,n}$ and think of the elements of $N_{t,n}$ as indices of the set of numbers $[1..M_{t,n}]$.

Definition 2 *Given a vector a of length $M_{t,m}$ over $\mathbb{F}_q$, let P_a be the multivariate polynomial $P_a(x_1, \ldots, x_m) := \sum_{\beta \in N_{t,n}} a_\beta x_1^{\beta_1} x_2^{\beta_2} \cdots x_m^{\beta_m}$*

Definition 3 (Reed Muller Encoding) *Given m, t, q and $S \subseteq \mathbb{F}_q$ such that $|S|^m > M_{t,m}$ (setting $S = \mathbb{F}_q$ is common), the Reed Muller encoding function takes words in $\mathbb{F}_q^{M_{t,m}}$ to words in $\mathbb{F}_q^{|S|^m}$ by giving the value of P_a for all the vectors in S^m. I.e., $RM(a) := \langle P_a(\alpha_1, \ldots, \alpha_m)_{(\alpha_1,\ldots,\alpha_m) \in S^m} \rangle$*

3.2 Basic Qualities of Reed Muller Codes

A Reed Muller code with parameters t, m, q and S has the following qualities:

- It is a linear code. This is easy to see.
- The required alphabet size q is bounded from below by the only (and obvious) requirement that $q > |S|$. Since our code length n is equal to $|S|^m$, we can use an alphabet size which is the m'th root of the one required for a Reed Solomon code.
- It has a rate of $M_{t,m}/|S|^m = \binom{m+t}{m}/|S|^m$. To see why $M_{t,m} = \binom{m+t}{m}$, Observe that $M_{t,m}$ is exactly the number of ways one can spread t "power units" (think of it as t balls) between m variables and one sinkhole (think of them as buckets). The sinkhole represents the "wasted" units, since we allow the sum of the powers of the variables to be strictly less than t.

  For example, choosing $m \ll t$ and $S = \mathbb{F}_q$, we get a rate of the order of m^t/q^m.
- It has a distance $d \leq n(1 - t/\sqrt[m]{n})$. This follows from the Schwartz-Zippel Lemma that states that the probability that a random vector in S^m will be a root of P_a is at most $t/|S|$. To see why this implies our distance bound, observe that our code is linear, so d is equal to the weight (i.e., number of non-zero coordinates) of the lightest word(s) in the code. Recall that a codeword $RM(a)$ is the vector of the values of the polynomial P_a over all the elements in S^m. Hence, by the Schwartz-Zippel Lemma, $RM(a)$ has at least $|S|^m - |S|^m(t/|S|) = n(1 - t/\sqrt[m]{n})$ non-zero coordinates.

[1] To avoid running into the problem that over $\mathbb{F}_q$ the polynomial $x^q - x$ is actually the zero polynomial, one needs to require that the degree of each individual variable x_i be less than q. We will assume the stronger requirement that $t < q$ which will also simplify our analysis.

3.3 The Schwartz-Zippel Lemma

Lemma 4 (Schwartz-Zippel) *Let P be a non-zero polynomial of degree $\leq t$ in m variables. For every $S \subseteq \mathbb{F}_q$ it must hold that $Pr_{\alpha \in S^m}(P(\alpha) = 0) \leq t/|S|$.*

Proof By induction on the number of variables m. For $m = 1$ it states the familiar result that a non-zero univariate polynomial has at most as many roots as its degree. Assume now that the assumption holds for all polynomials of $m - 1$ variables. Given $P(x_1, \ldots, x_m)$, rewrite it as a polynomial in x_1: $P(x_1, \ldots, x_m) = \sum_{i=0}^{t'} P_i(x_2, \ldots, x_m) x_1^i$. Note that t' may be strictly smaller than t because we omit all the powers of x_1 for which the coefficient polynomial P_i is zero (this does not eliminate all the coefficients because P is not zero). By looking at the coefficient $P_{t'}$ and separately considering the two cases $P_{t'}(\alpha_2, \ldots, \alpha_m) = 0$ and $P_{t'}(\alpha_2, \ldots, \alpha_m) \neq 0$ we get:

$$
\begin{aligned}
& Pr(P(\alpha) = 0) \ = \\
& Pr(P_{t'}(\alpha_2, \ldots, \alpha_m) = 0) Pr(P(\alpha_1) = 0 \mid P_{t'}(\alpha_2, \ldots, \alpha_m) = 0) \\
& + \ Pr(P_{t'}(\alpha_2, \ldots, \alpha_m) \neq 0) Pr(P(\alpha_1) = 0 \mid P_{t'}(\alpha_2, \ldots, \alpha_m) \neq 0) \\
& \leq Pr(P_{t'}(\alpha_2, \ldots, \alpha_m) = 0) + Pr(P(\alpha_1) = 0 \mid P_{t'}(\alpha_2, \ldots, \alpha_m) \neq 0)
\end{aligned}
$$

Since $P_{t'}$ is of degree at most $t - t'$, by the induction hypothesis we have that $Pr(P_{t'}(\alpha_2, \ldots, \alpha_m) = 0) \leq (t - t')/|S|$. Also, in the event that $P_{t'}(\alpha_2, \ldots, \alpha_m) \neq 0$, we have that P, as a polynomial in one variable, is of degree $t' > 0$ and thus has at most t' roots. Hence, $Pr(P(\alpha_1) = 0 \mid P_{t'}(\alpha_2, \ldots, \alpha_m) \neq 0) \leq t'/|S|$. Combining these results we get that $Pr(P(\alpha) = 0) \leq (t - t')/|S| + t'/|S| = t/|S|$ ■

3.4 Hadamard Codes as a Special Case of Reed Muller Codes

It turns out that Hadamard code can be thought of as a special case of Reed Muller codes in the following way:

If we take a Reed Muller code with parameters $t = 1, q = 2$ and $S = \mathbb{F}_q$, we get that $M_{t,m} = m + 1$ and $n = 2^m$. I.e., the resulting code takes binary vectors of length $m + 1$ to binary vectors of length 2^m. Given a message $a = (a_0, a_1, \ldots, a_m)$, the polynomial P_a turns out to be just $a_0 + \sum_{i=1}^{m} a_i x_i$. The reason for this is that by setting $t = 1$ the only monomials allowed are those that contain at most one variable raised to the power 1, with the rest raised to the power 0. If we set $a_0 = 0$, we can think of our code as encoding binary vectors of length m, and the encoding $\mathrm{C}(a_1, \ldots, a_m) := \mathrm{RM}(0, a_1, \ldots, a_m)$ of $a_1, \ldots, a_m$, is the vector of all the values that the linear formula $\sum_{i=1}^{m} a_i x_i$ takes over all binary vectors of length 2^m. We have got an Hadamard[2] code!

[2]For the nitpickers among us, we also have to omit the coordinate in the codeword containing the (zero) value of $P_a(\vec{0})$.