U.C. Berkeley — CS294: Codes and Complexity | Handout N3
Professor Luca Trevisan | Scribe: Stephen Sorkin

9/3/2003

Notes for Lecture 3

3.1 Outline

Today's lecture will cover:

- Efficient Polynomial Multiplication and Evaluation
- Reed-Solomon encoding in $\tilde{O}(n)$ ($\tilde{O}(f(n)) = O(f(n)(\log n)^{O(1)})$)
- Analysis of the Justesen code
- Discussion of Reed-Muller codes
- Concatenation of Reed-Solomon and Reed-Muller codes

3.2 Efficient Polynomial Multiplication and Evaluation

Consider a finite field $\mathcal{F}$ and some element w of order n in $\mathcal{F}$, so that $w, w^2, \ldots, w^n$ are all different, and $w^n = 1$. Now consider polynomials over $\mathcal{F}$ of degree less than or equal to n. Such a polynomial may be represented in two equivalent ways:

- As coefficients:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

- As the value of the function at a set of points:

$$p(1), p(w), \ldots, p(w^{n-1})$$

To convert from coefficients to values is the process of evaluating the polynomial. To convert from values to coefficients is the process of interpolating the polynomial. The following theorem demonstrates the ease of converting between these two representations:

Theorem 1 *Both evaluation and interpolation can be solved with $O(n \log n)$ field operations.*

Corollary 2 *Polynomial multiplication can be done with $O(n \log n)$ field operations.*

We quickly review the two different versions of Reed-Solomon coding discussed in the last lecture. The "RS" version of the Reed-Solomon code has:

$$c_0, \ldots, c_{k-1} \to p(w^0), p(w), \ldots, p(w^{n-1}) \ (p(x) = c_0 + c_1 x + \ldots + c_{k-1} x^{k-1})$$

By the first part of the theorem, a message may be encoded in $O(n \log n)$ field operations. The "RS' " version of the Reed-Solomon code has:

$$c_0, \ldots, c_{k-1} \rightarrow \text{coefficients of } p(x) \cdot g(x)$$

Proof: Of corollary.

Consider:

$$\begin{aligned} A(x) &= a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} \\ B(x) &= b_0 + b_1 x + \ldots + b_{n-1} x^{n-1} \end{aligned}$$

By the first part of the theorem, in $O(n \log n)$ time, we can run an evaluation on both of these polynomials to determine $A(w^0), A(w^1), \ldots, A(w^{n-1}), B(w^0), B(w^1), \ldots, B(W^{n-1})$. With n pointwise multiplications we can find: $A \cdot B(w^0), A \cdot B(w^1), \ldots, A \cdot B(w^{n-1})$. By the second part of the theorem, in $O(n \log n)$ time, we can interpolate these values to find $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ where

$$A \cdot B(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_{n-1} x^{n-1}.$$

Note: sufficiently many of the higher end coefficients of A, B must be zero so that the degree of the product is n or less. $\square$

Proof: Of the first part of the theorem.

Consider some element w of order n, and for simplicity, let n be a power of two. Then consider a polynomial A:

$$\begin{aligned} A(x) &= a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} \\ &= a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2} + \\ &\quad a_1 x + a_3 x^3 + \ldots + a_{n-1} x^{n-1} \\ &= a_0 + a_2 (x^2) + a_4 (x^2)^2 + \ldots + a_{n-2} (x^2)^{\frac{n}{2}-1} + \\ &\quad x \cdot (a_1 + a_3 (x^2) + a_5 (x^2)^2 + \ldots + a_{n-1} (x^2)^{\frac{n}{2}-1}) \\ &= A_{\text{even}}(x^2) + x \cdot A_{\text{odd}}(x^2) \end{aligned}$$

Note that both $A_{\text{even}}, A_{\text{odd}}$ are of degree $\frac{n}{2} - 1$. Since the field is cyclic, it is sufficient to compute $A_{\text{even}}((w^2)^0), A_{\text{even}}((w^2)^1), \ldots, A_{\text{even}}((w^2)^{\frac{n}{2}-1})$, and the corresponding A_{odd} terms, since w^2 is a primitive $\frac{n}{2}$th root of one.

We can now analyze the recursion. Let $T(n)$ be the number of field operations of this algorithm to evaluate a degree n polynomial. The recurrence relation is:

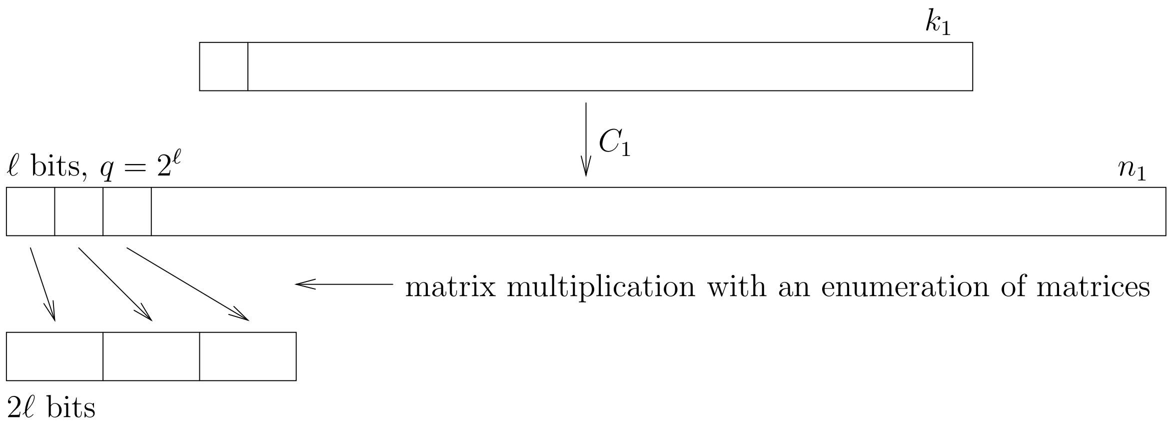
$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

The Master Theorem says that $T(n) = O(n \log n)$ operations in the field. $\square$

3.3 Justesen Code

We have observed that Reed-Solomon codes have a big alphabet, but for many reasons, we would rather have a code over a binary alphabet. Assume that we are given a reasonable $[n_1, k_1, d_1]_q$-code C_1. An example would be a Reed-Solomon code concatenated with another Reed-Solomon code. From before, we saw that it would be an $\left[n_1, \frac{n_1}{4}, \frac{n_1}{4}\right]_q$ code where $q < \log n_1$. We would like to have a $[n, \Omega(n), \Omega(n)]_2$ code.

From a probabilistic argument, there exist $\left[n_2, \frac{n_2}{2}, \frac{n_2}{15}\right]_2$ linear codes. For this concatenation to work, $2^{\frac{n_2}{2}} \sim q$ or $n_2 \sim \log\log n$. An alternate solution is to itemize and use all linear codes with the specified information and block lengths. The following is a graphical depiction of this procedure.



We will prove a more general result and specialize it.

Theorem 3 *Let C_1 be a $[n_1, k_1, d_1]_q$ code, $q = 2^\ell$. Consider a collection of codes $C_2 : \{0,1\}^\ell \to \{0,1\}^{n_2}$ such that:*

- *There are N codes in the collection.*
- *At least $(1-\epsilon)N$ codes are $[n_2, \ell, d_2]_2$.*

By cycling through this collection we will have a $[n_1 n_2, \ell k_1, d_2(d_1 - \epsilon n_1 - \epsilon N)]_2$ code.

A random linear code will be $\left[2\ell, \ell, \frac{\ell}{8}\right]_2$ with probability at least $1 - \frac{1}{2^{\Omega(\ell)}}$. This collection is of size $2^{2\ell^2} = 2^{2(\log q)^2}$. This concatenation will result in a $\left[2n_1\ell, \frac{n_1\ell}{4}, n_1\ell\left(\frac{1}{32} - o(1)\right)\right]_2$ code.

PROOF: Take two different messages. Applying C_1 to each yields two code words of length n_1 that differ in at least d_1 places. Now in each code word, the number of symbols which we apply bad matrices to is less than $\epsilon N \left\lceil \frac{n_1}{N} \right\rceil \le \epsilon N \left(\frac{n_1}{N} + 1\right) = \epsilon n_1 + \epsilon N$. So for each message, there are at least $d_1 - \epsilon n_1 - \epsilon N$ positions where:

- The matrix code is good.

- The first level of encoding uses different symbols.

Each of these positions contribute d_2 to the hamming distance of the final encoding. $\square$

3.4 Reed-Muller Codes

Reed-Muller codes are a generalization of Reed-Solomon Codes in which the polynomial is multi-variate. To be more formal, let $\mathcal{F}_q$ be a finite field of size q and $S \subseteq \mathcal{F}$ (analogous to the x_i in Reed-Solomon), and t, m be other parameters.

A message in the Reed-Muller code is a polynomial in m variables of degree t. The encoding of the message is the value of this polynomial on S^m. As an example consider $\mathcal{F}_7, S = \{2, 3, 4\}, m = 2, t = 2$. Then the polynomial would be:

$$p(x, y) = c_o + c_{10}x + c_{01}y + c_{20}x^2 + c_{02}y^2 + c_{11}xy$$

An example message is $(1, 2, 1, 0, 0, 5)$ which would result in the polynomial $p(x, y) = 1 + 2x + y + 5xy$. The encoding this message would be the evaluation of the polynomial at $(2, 2), (2, 3), (2, 4), (3, 2), \ldots$.

How many coefficients are there in an m-variate, degree t polynomial? There is a coefficient for each monomial, which is of the form:

$$c_{i_1 i_2 \ldots i_m} x_i^{i_1} x_2^{i_2} \ldots x_m^{i_m}, i_1 + i_2 + \ldots + i_m \leq t$$

There is a nice way to count these. We can place stars to count the degree of each variable in unary:

$$\underbrace{***}_{i_i} | \underbrace{****}_{i_2} | \cdots | \underbrace{**}_{i_m} | \quad \underbrace{***}_{t - i_1 - i_2 - \ldots - i_m}$$

In total, there are t stars and m bars. Furthermore, every such sequence is valid and defines exactly one monomial. Hence there are $\binom{t+m}{m}$ or equivalently $\binom{t+m}{t}$ monomials.

The encoding in the Reed-Muller code is of length $|S|^m = n^m$. The message is of length $\binom{t+m}{m} = k$. Therefore this is a $\left[|S|^m, \binom{t+m}{t}, ?\right]_q$ code. In Reed-Solomon, $|S| = n, m = 1, t = k - 1$.

Claim 4 *Reed-Muller is a* $\left[|S|^m, \binom{t+m}{t}, |S|^m \left(1 - \frac{t}{|S|}\right)\right]_q$ *code.*

Theorem 5 *Reed-Muller has minimum distance* $|S|^m \left(1 - \frac{t}{|S|}\right)$.

PROOF: Let p be an m-variate degree t polynomial over $\mathcal{F}, S \subseteq \mathcal{F}, p \neq 0$. We will show that $\mathbf{Pr}_{x \in S^m}[p(x) = 0] \leq \frac{t}{|S|}$ or $\mathbf{Pr}_{x \in S^m}[p(x) \neq 0] \geq 1 - \frac{t}{|S|}$. Since Reed-Muller is a linear code, this guarantees the minimum distance. The proof is by induction on m.

For $m = 1$, this result follows from the fact that p has no more than t roots so that no more than a $\frac{t}{|S|}$ fraction of S can be roots.

For arbitrary m write:

$$p(x_1, \ldots, x_m) = p_0(x_2, \ldots, x_m) + x_1 p_1(x_2, \ldots, x_m) + \ldots + x_1^t p_t(x_2, \ldots, x_m)$$

Since p is nonzero, at least one of these p_i is nonzero. Assume p_i is the last nonzero polynomial. It will have degree $t-i$ or less.

$$p(x_1,\ldots,x_m)=\sum_{j=0}^{i} x_1^j p_j(x_2,\ldots,x_m)$$

Then note:

$$\begin{aligned}\mathbf{Pr}_{x_1,\ldots,x_m}[p(x)=0] &= \mathbf{Pr}[p(x_1,\ldots,x_m)=0\wedge p_i(x_2,\ldots,x_m)=0]+\\ &\quad \mathbf{Pr}[p(x_1,\ldots,x_m)=0\wedge p_i(x_2,\ldots,x_m)\neq 0]\end{aligned}$$

Considering each of these terms:

$$\begin{aligned}\mathbf{Pr}[p(x)=0\wedge p_i(x_2,\ldots,x_m)=0] &\leq \mathbf{Pr}[p_i(x_2,\ldots,x_m)=0]\\ &\leq \frac{t-i}{|S|}\text{ since } p_i \text{ is of degree } t-i \text{ or less}\\ \mathbf{Pr}[p(x)=0\wedge p_i(x_2,\ldots,x_m)\neq 0] &\leq \sum_{x_2,\ldots,x_m|p_i(x_2,\ldots,x_m)\neq 0}\frac{1}{|S|^{m-1}}\mathbf{Pr}_{x_1}[p(x_1,\ldots,x_m)=0]\\ &\leq \sum_{x_2,\ldots,x_m}\frac{1}{|S|^{m-1}}\cdot\frac{i}{|S|}\\ &\leq \frac{i}{|S|}\text{ since the sum is over } \leq |S|^{m-1} \text{ points}\end{aligned}$$

Hence $\mathbf{Pr}_{x_1,\ldots,x_m}[p(x)=0]\leq\frac{t}{|S|}$. □

It follows that Reed-Muller is a $\left[|S|^m,\binom{t+m}{t},|S|^m\left(1-\frac{t}{|S|}\right)\right]_q$ code.

Is this code better than Reed-Solomon? For fixed $t,|S|$, the minimum distance remains the same. The information rate gets worse as m gets larger. However, the alphabet size goes down as m gets larger. Reed-Muller allows a tradeoff between smaller alphabets and information rates.

We can see the other extreme. Let $q=2,|S|=2$ so that $n=2^m,k=\binom{m+1}{m}=m+1$. Also since $d=2^m\left(1-\frac{t}{|S|}\right)=2^{m-1}$ we must have $t=1$ (otherwise $d=0$). This results in a $\left[n,1+\log n,\frac{n}{2}\right]_2$ code. The polynomial for this code looks like:

$$p(x_1,\ldots,x_m)=c_0+c_1x_1+\ldots+c_mx_m$$

In other words it is an inner product. If we discard the first term, c_0, and treat the code as the inner product between the message and all possible strings of length m, we get a code called the Hadamard code, which is $\left[n,\log n,\frac{n}{2}\right]_2$.