CS 229r Essential Coding Theory Feb 24, 2020

Lecture 8

Instructor: Madhu Sudan *Scribe: Franklyn Wang*

1 Feburary 24th, 2020

Last class, we discussed the Reed-Muller codes. Recall that a Reed-Muller code is a multivariate polynomial over $\mathbb{F}_q$ with m variables and degree at most r. The message space is all polynomials of degree at most r with individual degrees at most $q-1$. Then, the encoding is simply evaluation on each element of $\mathbb{F}_q^m$. We use the Schwartz-Zippel lemma: if $r < q$, then two multivariate polynomials can agree on at most r/q of the possible inputs.

An extension of the Schwartz-Zippel lemma shows further that for $r = a(q-1)+b$ where $0 \le b < q$, then two polynomials disagree on at least $q^{-a}(1-b/q)$ fraction of the possible inputs.

These codes always have length $n = q^m$. The distance lemma gives $d = n[q^{-a}(1-b/q)]$. For each of these codes, we can calculate their dimension by counting the number of monomials that satisfy the conditions, which gives us

$$k(q,m,r) = |\{(e_1, e_2, \ldots e_m) \in \mathbb{Z}^m | 0 \le e_i < q, \sum e_i \le r\}|$$

In the case of $m = 1$, we recover the Reed-Solomon codes, where we get $n = q, k = r+1$, and $d = n-k+1$. In the case where $m = 2$, we get $n = q^2$, $k = n(1-\delta)^2/2$, and $d = \delta n$, where δ is a parameter we can choose.

Reed-Muller codes offer an incredible level of versatility, derived from having three parameters that we can manipulate. A few examples:

- In the case where we hold $q, r \le q-1$ constant while letting m grow large, we have

$$k(q,m,r) = \binom{r+m-1}{m-1} \sim \left(\frac{m}{r}\right)^r$$

  In other words Reed-Muller codes yield codes of constant alphabet size and relative distance with dimension growing as an arbitrary polynomial in the logarithm of the block length.

- We can get polynomial rate codes when $r = q/2$ and $m = q^{\varepsilon}$, which gives us a code that has parameters $[n, \frac{1}{2}n^{1-\varepsilon}, n/2]_{(\log n)^{1/\varepsilon}}$. Here, the alphabet size grows very slowly, and the distance is fixed at $n/2$. However, the dimension of the code increases polynomially in n. These particular codes are extremely useful in the field of Probabilistically Checkable Proofs.

1.1 Binary Alphabet

Consider the case in which $q = 2, r = 1, m \to \infty$. Then we get $n = 2^m$, and we obtain that the code has dimension $m+1$, and that there are 2^{m+1} codes. This is equal to $2n$ codes. If you convert the codewords to vectors in $\{-1,1\}^n$, each of which have distance $n/2$, this is equivalent to saying that the vectors have nonpositive dot product with each other. However, recall that we showed that the most vectors one can have in $\{-1,1\}^n$ whose dot products are nonpositive is $2n$, so this has obtained the sharpest possible bound. In fact, the only examples which attain this bound are rotations of the coordinate axes, which implies that this is a rotation of the coordinate axes. These codes are known as Hadamard codes.

We will prove on our problem set that the dual of the Reed-Solomon codes is also a Reed-Solomon code. In fact, it will turn out to be the case that the dual of Reed-Muller codes are also Reed-Muller codes, and that looking at the dual of Reed-Muller codes is a very productive endeavor.

2 The BCH codes

Assume that we have a binary alphabet, and that $d = 10$. With the greedy construction, we get a number of codewords equal to approximately

$$\frac{2^n}{n^{10}}$$

On the other hand, we know that the number of codewords cannot exceed

$$\frac{2^n}{n^5}$$

by the Hamming bound.

These bounds aren't equal, so which one is correct? The answer turns out to be: It's complicated, but closer to $2^n/n^5$ than $2^n/n^{10}$.

In retrospect, the construction is simple, but a tricky mathematical argument is needed to establish it. We start wtih a code over a big alphabet – $\{0, 1, \ldots, q-1\}^n = \mathbb{F}_q^n$, where $q = 2^t$, and then take the intersection of this code with $\mathbb{F}_2^n$. First, take $\hat{C} = C \cap \mathbb{F}_2^n$. *A priori*, it is not even obvious that $\hat{C}$ should have size greater than one. It's important to distinguish what finding $\mathbb{F}_2$ in $\mathbb{F}_q$ means here – it refers to the set of elements

$$\{x | x \in \mathbb{F}_q, x^2 - x = 0\}.$$

Exercise 1. *Show that we can find a code $\hat{C}$ with nontrivial size, without doing any of the following. In order to solve this, you will need to slightly relax the use of $\mathbb{F}_2^n$. Rather, show that there exists n values $v_1, v_2, \ldots v_n$ so that*

$$C \cap (\{v_1, v_1 + 1\} \times \{v_2, v_2 + 1\}\} \ldots \times \{v_n, v_n + 1\})$$

has a large size for any set of vectors C.

Take C to be a Reed-Solomon code with $k = n - d$ over $\mathbb{F}_2^t$, where $2^t = q = n$, so that the code has distance d. Now take $\hat{C} = C \cap \mathbb{F}_2^n$.

Note that $\Delta(\hat{C})$ is at least d, because we've taken a subset of a code with distance d. The question is, how large is this code? Instead of looking at the code, we will instead look at the parity check matrix. The parity check matrix of the Reed-Solomon code is

$$H = \begin{bmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{d-1} \\ 1 & \alpha_2 & \ldots & \alpha_2^{d-1} \\ \vdots & \vdots & & \vdots \\ 1 & \alpha_n & \ldots & \alpha_n^{d-1} \end{bmatrix}$$

Note that this matrix has n rows and d columns. Now, recall that the elements $\alpha_i \in \mathbb{F}_q$ form a vector space over $\mathbb{F}_2$, which lets us elongate each row into its coordinates under a basis of $\mathbb{F}_q$ over $\mathbb{F}_2$. This lets us create a matrix $\hat{H}$ that has n rows and dt columns. Importantly, we now have that $\hat{C}$ can be defined as the dual code of $\hat{H}$, whose entries now all lie in $\mathbb{F}_2$. This matrix has n rows and dt columns, which means that the dual has n rows and at least $n - dt$ columns[1]. Thus, it has size at least

$$2^{n-dt} = \frac{2^n}{2^{dt}} = \frac{2^n}{n^d}.$$

However, this is not the size we wanted! Are we stuck now? As it turns out, the answer is no, we are in fact very close. We need one more fact from algebra.

Theorem 2 (Frobenius Endomorphism)**.** *For $a, b \in \mathbb{F}_q$ in any finite field of characteristic p, we have that $(a + b)^p = a^p + b^p$.*

[1]The inequality is needed because the expanded parity check matrix is not of full rank.

Proof. Note that by the binomial theorem,

$$(a+b)^p = \sum_{i=0}^{p} \binom{p}{i} a^i b^{p-i}$$

Note that each term except for the first and last is zero since the field has characteristic p, which yields the result. $\square$

In terms of linear algebra, the matrix $\hat{H}$ does not have full rank. For example, on the very first row, expanding this element to t full columns gives us $t-1$ columns which are all zeroes. Clearly, these columns can be removed without changing the parity checks in any meaningful way. More importantly, note that if $\mathbf{y} \in \mathbb{F}_2^n$ is orthogonal to the second and third column, these constraints simplify to

$$y_1\alpha_1 + y_2\alpha_2 + \ldots + y_n\alpha_n = 0$$
$$y_1\alpha_1^2 + y_2\alpha_2^2 + \ldots + y_n\alpha_n^2 = 0$$

Since $y_i \in \mathbb{F}_2$, note that $y_i = y_i^2$, so the second equation rearranges to

$$y_1^2\alpha_1^2 + y_2^2\alpha_2^2 + \ldots + y_n^2\alpha_n^2 = 0.$$

By the Frobenius endomorphism, we have that this becomes $(y_1\alpha_1+y_2\alpha_2+\ldots+y_n\alpha_n)^2 = 0$, which is implied by $y_1\alpha_1 + y_2\alpha_2 + \ldots + y_n\alpha_n = 0$. Therefore, these constraints are redundant (in fact exactly equivalent), and we can remove every column of the form

$$\begin{bmatrix}\alpha_1^{2k} & \alpha_2^{2k} & \ldots & \alpha_n^{2k}\end{bmatrix}$$

from the matrix by the same argument. This results in a matrix whose size is $n \times dt/2$. This means that the dual code has at least $n - dt/2$ columns, which implies a size of at least

$$2^{n-dt/2} = \frac{2^n}{2^{dt/2}} = \frac{2^n}{n^{d/2}}$$

which is precisely the bound given by Hamming. Thus, at least in this case, $2^n/n^5$ was closer than $2^n/n^{10}$.

3 Limitations and Extensions of the BCH code

But (and this is a big but) we have not just solved all of error correcting codes. We can see this in a variety of ways. Say that we were in $\mathbb{F}_3^n$. The greedy code has $3^n/n^d$ codewords. The hamming argument gets a bound of $\frac{3^n}{n^{d/2}}$. If we do the BCH proof again, we get that the lower bound is

$$3^n/n^{2d/3},$$

because we can only remove the multiples of three. Note that this lies between the greedy bound and the Hamming bound. Even if we are still working in $\mathbb{F}_2^n$ but $d = n^\varepsilon$, we still have strange results. Note that above we made the approximation that $\binom{n}{d} \approx n^d$ because d was a constant. In the case where $d = n^\varepsilon$, we have that

$$\binom{n}{d} \approx \left(\frac{n}{d}\right)^d \approx n^{(1-\varepsilon)d}$$

and note now that the greedy algorithm obtains $2^n/(n^{(1-\varepsilon)d})$ and the Hamming bound is $2^n/(n^{(1-\varepsilon)d/2})$. The BCH code gives us $2^n/n^{d/2}$ codewords, which again lies in between the greedy algorithm and the Hamming bound. We know that the greedy bound cannot be optimal. However, we still don't know whether the Hamming bound is attainable in this regime.

The dual of a BCH code is still a BCH code. However, it is also necessarily a code with very small dimension. Well, if the code is low-dimensional, can we at least hope for a high distance? The dual BCH code has parameters given by $[n, d/2 \log n, n/2 - d\sqrt{n}]$. The distance of these codes are very close to $n/2$, which is remarkable. At the end of the first lecture, we went over a few optimal codes of various distances. The optimal code of distance one has size $n - 1$ (checksums), the optimal code of distance two has size $n - \log_2(n + 1)$ (the Hamming code), the optimal code of distance three has size $n - \log_2(n + 1) - 1$. Many of the codes that this BCH algorithm produces resemble those optimal codes.

A lesson that these codes have shown is that in fact, the use of abstract algebra to create codes has truly done better than random. The BCH code, the Hamming code, and the Hadamard code all outperform random codes. Almost *any* algebraic geometry code outperforms the random code in some domain.

4 Exercises

Exercise 3. *Why is it the case that, in the Reed-Muller codes, we only consider multivariate polynomials where each individual degree is at most $q - 1$?*

Exercise 4. *In the proof of the BCH code, we showed that the matrix $\hat{H}$ does not have full rank, and as such we can remove several redundant columns. However, our removal of columns was quite crude, removing only blocks of columns at a time. Is it possible that we can remove even more columns and thus improve the size of the code even more?*