CS 229r Essential Coding Theory, Lecture 7 Feb 14, 2017

Lecture 7

Instructor: Madhu Sudan *Scribes: Vasileios Nakos*

1 Last Lecture

Last week we talked about Reed-Solomon codes [RS60]. The Reed-Solomon codes are $[n, k, n-k+1]_n$ codes that are optimal, in the sense that they match the pigeonhole principle.

On weakness of Reed-Solomon codes is their alphabet size, which has to be at least n. A fair amount of work in Coding Theory tries to understand how to fix it this disadvantage. We do not expect to achieve the same rate with smaller alphabet, because of the q-ary Plotkin bound which states that $R \leq 1 - \frac{q}{q-1}\delta$. Best known codes achieve $R \geq 1 - \delta - \frac{1}{\sqrt{q}}$.

2 Today

In today's lecture, we will see concatenated codes, Reed-Muller codes and BCH codes. All these cods try to reduce the alphabet size while still achieving good rate and small relative distance.

3 Concatenated Codes

Today we will talk about concatenated codes [FF66]. Take a $(N, K, D)_Q$ code, which we will call Outer code (C_{out}) and a $(n, k, d)_2$ code which we will call inner code (C_{in}). We enforce that $Q = 2^k$. Now this means that C_{out} takes messages of length K in an alphabet of size 2^k and encodes them to messages of length N, while C_{in} takes messages of length k and encodes them to messages of length n in an alphabet of size 2. The concatenated code then maps each message $(m_1, \ldots, m_K)$ to $(C_{in}(m'_1), C_{in}(m'_2), \ldots, C_{in}(m'_K)$, where $(m'_1, m'_2, \ldots, m'_N) = C_{out}(m_1, \ldots, m_k)$. This means that one can use as an outer code a Reed-Solomon code (which inevitably has a large alphabet) and as an inner code a Hadamard code, which as we will see later has alphabet size 2, and reduce the alphabet size down to 2. Such a code is called Justesen code.

It can be proved that the concatenated code is $(N \cdot n, K \cdot k, D \cdot d)_2$. To understand the limits of the technique of code concatenation, a result that is useful is the Zyablov bound, which is a lower bound on the rate R and relative distance of concatenated codes. This bound is the best known bound for the binary code.

There is a very simple construction that matches the Zyablov bound in poly(n) time, although we will not prove it. We employ the following construction: Let $m_0, \ldots, m_{K-1} \in \mathbb{F}_k$ where $F_K = \{a_1, \ldots, a_K\}$ with $K = 2^k$. Then the message is the polynomial $M(x) = \sum_i m_i x^i$ and the encoding of the message equals $< M(a_1), a_1, M(a_2), a_2, \ldots, M(a_K), a_K >$ when we view each one of these symbols as a k bit sequence. This achieves the Zyablov bound (exercise).

4 Reed-Muller codes

In this section we will discuss Reed-Muller codes [Mul54]. These codes were discovered by Muller and provided a decoding algorithm by Reed. As we mentioned in the previous section, our goal is to reduce the alphabet size. For univariate codes we need $n \leq q$. For bivariate polynomials we need $n \leq q^2$ and we expect that for multivariate polynomials we need $n \leq q^m$, where m is the degree of the polynomial. So we can start building larger codes with $m \to \infty$, but what is the price we need to pay ?

The construction of the generalised Reed-Muller codes ($q > 2$) is the following:View messages as m-variate polynomials of degree at most r over $\mathbb{F}_q$. The encoding is the evaluations over the whole space, that is q^m points.

Let now $N(m,q,r) = |\{(r_1,\ldots,r_m) : |0 \leq r_q \leq q-1, \sum_{i=1}^m r_i\}|$, which is the set of all monomials that can appear in a Reed-Muller code and is a basis of the vector space of all codewords. To understand how $N(m,q,r)$ grows we look at some special cases.

- A simple case is hen $r < q$ we have that $N(m,q,r) = \binom{r+m}{m}$. In particular for $m = O(1)$, and $r, q \to +\infty$ we have that $\binom{r+m}{m} \sim \frac{r^m}{m!}$.
- A popular choice in CS theory is the following settinf of parameters. Pick $\delta = \frac{1}{2}$, $m = \frac{\log k}{\log\log k}, q = \log^2 k, r = \frac{q}{2}$. The length of the code is $n = q^m = k^2$, which is still polynomial in k. This is interesting because alphabet is poly$(\log n)$.
- Another interesting special case is the Hadamard code. For $q = 2, r = 1, m \to +\infty$ we have that $N(q,m,r) = m+1$ and we take a $[2^m, m+1, 2^{m-1}]_2$ code, which is called the Hadamard codes. If we also take all these vectors and write them in binary and then transform 0s to -1s, we get 2^{m+1} vectors in $\{-1,+1\}^{2^m}$ such that their pairwise inner products are non-negative.
- $q = 2, n = 2^m$. Then $N(m,q,r) = \sum_{i=0}^r \binom{m}{i}$. Then $\delta = 2^{-r}$. The worst polynomial is $x_1 \cdot x_2 \cdot \ldots \cdot x_r$ for which every x_i should be set to 1 in order to differ from the 0 polynomial, which happens with probability 2^{-r}.

We will now see the notion of a dual code and how Hadamard and Hamming codes are the dual of one another. Define $C^\perp = \{x \in \mathbb{F}_q^n \ \langle x,y\rangle = 0, \forall y \in C\}$. For linear codes C which are generated by a parity check matrix H, then the dual of C is generated by H^T.

One can see that the dual of RM(q,m,r) is RM$(q,m,m(q-1)-r)$. The monomials that generate all codewords of RM(q,m,r) are of the form $\Pi_{i=1}^m x_i^{i_j}$ with$\sum_{i=1}^j i_j leq r$, while the monomials that generate RM$(q,m,m(q-1)-r)$

The following relations hold:

- (Reed-Solomon)$^\perp$ = Reed-Solomon
- (Reed-Muller)$^\perp$ = Reed-Muller
- (Hadamard)$^\perp$ = Hamming

From the above, one concludes that Hamming code is a special case of Reed-Muller code, since its dual is the Hadamard code and hence a Reed-Muller code and we know that the dual of any Reed-Muller code is a Reed-Muller code.

The relative distance of Reed-Muller codes can be computed using the Schwartz-Zippel lemma: any two m-variate polynomials of degree at most r over $\mathbb{F}_q$ agree on at most $\frac{r}{q}$ fraction of $\mathbb{F}_q^m$.

Exercise: Using induction, prove the Schwart-Zippel lemma. The lemma is tight for the polynomials 0 and $(x-a_1)(x-a_2)\ldots(x-a_r)$ where $a_1,\ldots,a_r$ are non-zero elements of $\mathbb{F}_q$ all different with each other.

5 BCH codes

In this section we describe BCH codes and prove their performance. BCH codes were found in late 50s-early 60s by two dfferent groups of researchers [BRC60, Hoc59].

We start with codes over $\mathbb{F}_{2^l}$ and we imagine d being a constant, let us say 20. For $d = 3$ we already have the Hamming codes, which we proved are optimal. Then we build a Reed-Solomon code over $\mathbb{F}_{2^l}$ of distance 20 (note that $n = 2^l$ because the alphabet size equals n). This means that $k = n - 20 + 1 = n - 19$.

Thus, we get a $[n, n-19, 20]_n \subseteq \mathbb{F}_{2^l}^n$. Let this code be C. The BCH code now is just $C \cap \mathbb{F}_2^n$, that is the codewords of C that have only 0 and 1 and no other letter of the alphabet appear.

We will prove that there are "plenty" of such codewords. For that, we look at the parity check matrix H of C, the i-th column of which is $[a_1^{i-1}, a_2^{i-1}, \ldots, a_n^{i-1}]^T$. The question now is how many $\beta \in \mathbb{F}_2^n$ are there such that $\beta H = 0$. The first approach is to use the fact that $\mathbb{F}_{2^l} \cong \mathbb{F}_2^l$ and make β live in $\mathbb{F}_2^{ln}$, while H to be a $n \times dl$ matrix with element in $\mathbb{F}_2$. So BCH membership is expressible by dl $\mathbb{F}_2$ constraints on $\beta_1, \ldots, \beta_n$. This gives a $[n, n - d\log n, d]_2$ code, since $l = \log n$.

But we can imporve over this approach by considering the inner product of β with the 2nd column: $\langle \beta, H_2 \rangle = \sum \beta_j a_j = 0$ and square this expression. This gives $\sum \beta_j a_j^2 = 0$, which means that the second constraint is enforced by the first! This implies that we can throw away every second constraint. This implies that we can get a $[n, n - \frac{d}{2}\log n + O(1), d]_2$.

Why are BCH codes important? Because the Hamming bound that states that the number of codewords is at most $\frac{2^n}{\binom{n}{\frac{d}{2}}} \sim \frac{2^n}{n^{\frac{d}{2}}}$, which after taking logarithms becomes $n - \frac{d}{2}\log n + O(1)$. This means that BCH codes achieve the optimal bound up to constants.

References

[BRC60] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[FF66] G David Forney and G David Forney. *Concatenated codes*, volume 11. Citeseer, 1966.

[Hoc59] Alexis Hocquenghem. Codes correcteurs derreurs. *Chiffres*, 2(2):147–56, 1959.

[Mul54] David E Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE Professional Group on Electronic Computers*, (3):6–12, 1954.

[RS60] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.