(67902) Topics in Theory and Complexity Nov 8, 2006

Lecture 2

Lecturer: Irit Dinur *Scribe: Noa Eidelstein*

1 Reminders from last week

Definition 1 *A $(n,k,d)_q$-code is a map $C:[q]^k \to [q]^n$ such that $min_{a\neq b} dist(C(a),C(b)) \geq d$.*

We saw some examples of codes:

- Hamming code, which is a $(2^l-1, 2^l-1-l, 3)_2$-code.
- Greedy code. The problem with this code is that it is not algorithmically explicit.
- Random linear code.

Are these codes "good"? We are interested in two parameters: the **relative distance** $\delta = \frac{d}{n}$, and the **rate** $r = \frac{k}{n}$. We are looking for infinite families of codes in which both δ and r are nonzero. We also want explicit encoding and decoding algorithms.

2 Decoding a Generic Linear Code

Problem Given matrix $G \in M_{k\times n}$ (the generating matrix of the code), and $r \in \{0,1\}^n$. Suppose also that r is close to the code, i.e. there exists a_0 such that $dist(a_0G, r) \leq \frac{d}{2}$. Find $a \in \{0,1\}^k$ such that $r = aG + e$, where $wt(e) < \frac{d}{2}$.

For General matrix G this problem is NP-hard. In fact,

Theorem 2 *For every $\delta > 0$, the following decision problem is NP-hard: Given vector r and matrix G in which there are at most 3 1's in each column (and the rest elements are 0's), decide between:*

1. *There exists $x \in \{0,1\}^k$ such that $\frac{wt(xG-r)}{n} \leq \delta$ (r is close to some codeword).*
2. *For every $x \in \{0,1\}^k$, $\frac{wt(xG-r)}{n} \geq \frac{1}{2} - \delta$ (r is far from all codewords).*

Corollary 3 *If $P \neq NP$ then there is no generic algorithm that given G and r outputs the nearest codeword to r.*

3 Decoding a Random Linear Code

Suppose the matrix G in the previous section was not arbitrary but rather randomly selected. Does this make the problem easier?

Problem Given a random matrix $G \in M_{k\times n}$ (the generating matrix of the code), and $r \in \{0,1\}^n$. Suppose also that r is close to the code, i.e. there exists a_0 such that $dist(a_0G, r) \leq \frac{d}{2}$. Find $a \in \{0,1\}^k$ such that $r = aG + e$, where $wt(e) < \frac{d}{2}$. In other words, we want algorithm such that

$$Pr_G[\text{The algorithm works for every } r \text{ close enough to } G] > \frac{2}{3}.$$

This problem is not known to be NP-hard, (in fact there aren't any average-case problems that are known to be NP-hard).

Remarks

1. This problem is considered "hard" (i.e. people think it is not in P).

2. There are parameters k, n for which this problem is easy. For example, take $k = \log n$. Then there are $2^{\log n} = n$ elements in $\{0,1\}^n$. We go over these n elements, and for every element a, calculate $wt(aG - r)$.

3. **Theorem 4 (BKW)** *: This problem is solvable for $k = \log n \cdot \log\log n$.*

3.1 This problem as Learning problem

Consider the following problem: There exists numbers $\alpha_1, ...\alpha_k$. These numbers are unknown, but there is a machine that, given $a \in \{0,1\}^k$, returns $(\sum_{i=1,..,k} a_i\alpha_i) mod 2$. This machine makes a mistake with probability ϵ. We want to find the numbers $\alpha_1, ...\alpha_k$. The algorithm is: choose $a \in \{0,1\}^k$, ask the machine for answer (which is $(\sum_{i=1,..,k} a_i\alpha_i) mod 2$ with probability $1-\epsilon$), and continue until you can recover the numbers α_i.

Notice that when $\epsilon = 0$, all we need is to do this sample until we get k independent a's. For $\epsilon > 0$, this problem is considered hard.

3.1.1 What is the connection to encoding?

Suppose we had n samples. We produce matrix $G \in M_{k\times n}$ and vector $r \in \{0,1\}^n$ as follows. In the i'th column of G, put the vector $a \in \{0,1\}^k$ that we had in the i-th step, and take r_i to be the answer of the machine in this step.

The condition that r is close to the function $a \mapsto (\sum_{i=1,..,k} a_i\alpha_i) mod 2$ is equivalent to the condition that $\alpha G - r \leq \epsilon n$.

Originally, the [BKW]-theorem said that this learning problem can be solved with $n = 2^{\frac{k}{lgk}}$ number of samples.

4 Hadamard Code

Definition 5 *Given $[n,k,d]_q$-code (the [] brackets mean this code is linear!), generating matrix G and parity-check matrix H (remember that the code words are $\{aG : a \in [q]^k\} = \{y \in [q]^n : yH = 0\}$), define the* **dual code** *to be code with generating matrix $G^\dagger = H^t$ and parity-check matrix $H^\dagger = G^t$.*

Definition 6 Hadamard code *is the dual code of Hamming code. It is easy to see that the generating matrix of Hadamard code is*

$$G := Had = \begin{pmatrix} 0 & 0 & 0 & ... & 1 \\ \vdots & \vdots & \vdots & ... & \vdots \\ 0 & 0 & 0 & ... & 1 \\ 0 & 1 & 1 & ... & 1 \\ 1 & 0 & 1 & ... & 1 \end{pmatrix} \in M_{k\times 2^k-1}$$

Since G has $2^k - 1$ columns, we can take their indices to be the nonzero elements of $\{0,1\}^k$: to every index $1 \leq i \leq 2^k - 1$, there corresponds $\beta = \beta(i) \in \{0,1\}^k$, which is the binary representation of i. Notice that $(aG)_i =< a, \beta(i) > mod 2$, i.e the i-th element of the vector aG is equal to the inner product (mod 2) of a and the corresponding index. This means that to every $0 \neq \beta \in \{0,1\}^k$, there exists some bit in aG which equals $(\sum_{i=1,..,k} a_i\beta_i) mod 2$. In other words, every nonzero linear function with input a has a bit in Hadamard code that returns the value of this function in a.

Corollary 7 *Every linear code is a subset of the bits of Hadamard code (since Hadamard codes gives all possible nonzero linear functions).*

4.1 The parameters of Hadamard code

Rate: $r = \frac{k}{n} = \frac{k}{2^k-1}$, thus the rate is logarithmic.
Relative distance: We want to calculate $min_{0 \neq a \in \{0,1\}^k} wt(aG)$.

Let j be the minimal index such that $a_j \neq 0$. Notice that for every β, $< a, \beta >=< a, \beta + e_j > +1$ (where e_j is the j-th unit vector). In particular, $< a, \beta > \neq < a, \beta + e_j >$. Thus, for half of the β's $< a, \beta >= 0$, and for the other half $< a, \beta >= 1$. Since $\beta = 0$ does not appear in the matrix, the number of indices i in which $(aG)_i =< a, \beta(i) >= 1$ is exactly $\frac{2^k}{k}$. Thus $min_{0 \neq a \in \{0,1\}^k} wt(aG) = \frac{2^k}{k}$, and the relative distance is $\delta = \frac{\frac{2^k}{k}}{2^k-1} \approx \frac{1}{2}$.

4.2 Decoding Hadamard code

Problem: Given $A \in \{0,1\}^{2^k-1}$. We know that there exists a such that $wt(a \cdot Had - A) < (\frac{1}{4} - \epsilon)2^k$. We want polynomial decoding algorithm.

Lemma 8 *If B is a code word, then to every index β, $B[e_j] = B[\beta] + B[\beta + e_j]$.*

Proof B is a code word, thus there exists b such that $B = bG$. Now, $B[e_j] =< b, e_j >= b_j$, and $B[\beta] + B[\beta + e_j] =< b, \beta > + < b, \beta + e_j > =< b, \beta > + < b, \beta > + < b, e_j > \equiv_2 < b, e_j >= b_j$, as required. ■

Algorithm for decoding the i-th bit of a

1. Take random indices $\beta_1, ..., \beta_m$.
2. Calculate $A[\beta_j] + A[\beta_j + e_i]$. By the lemma, if the bits of a were not changed, we get exactly a_i.
3. Retrun the value that was achieved in the majority of the cases.

4.3 List-decoding of Hadamard code

Given word A, in which at most $\frac{1}{4} - \epsilon$ of the bits were changed. We look at the ball of radius $d - \epsilon d$ around it. This ball might contain several code words.

List-decoding: Find all code words that are contained in a ball of radius $d - \epsilon d$ around a given word.

Algorithm (Goldreich, Levin): Given $A \in \{0,1\}^{2^k-1}$, it is possible to find all words that are close to A in $\frac{1}{2} + \epsilon$ of the bits, in time that is polynomial in k (there are at most $\frac{1}{\epsilon^2}$ such words).

5 Reed-Solomon Code

Definition 9 *A* **field** *is a set F with addition and multiplication operations $+, \cdot$ respectively and special elements 0,1 such that :*

- *addition forms a commutative group on the elements of F.*
- *multiplication forms a commutative group on the elements of $F \setminus \{0\}$.*
- *there is a distribution-law of multiplication over addition*

Theorem 10 *For any prime p and $k \in \mathbb{N}$ there exists a (unique) field F_{p^k} of order p^k.*

Definition 11 *$F[x]$ denotes the ring of polynomials over F. The elements of $F[x]$ are of the form $p(x) = \sum_{i=0,..,d} a_i x^i$ where $a_i \in F$ and $d \in \mathbb{N}$ is called the* **degree** *of p.*

Definition 12 Reed Solomon Code: *Let $\mathbb{F}_q$ be a field of order q, and let $\alpha_1, ..., \alpha_n \in \mathbb{F}_q$ be distinct numbers($n \leq q$). The Reed-Solomon code $RS_{q,k,n}$ is a map $RS : \mathbb{F}_q^k \to \mathbb{F}_q^n$ defined by:*

$$a = (a_0, ..., a_{k-1}) \mapsto < P_a(\alpha_i) >_{i=1,..,n}, \text{ where } P_a(x) = \sum_{i=1}^{k-1} a_i x^i.$$

Lemma 13 *RS is a linear code.*

Proof For every a, b,

$$RS(a+b) =< P_{a+b}(\alpha_i) >_{i=1,..,n} =< \sum_j (a+b)_j \alpha_i^j >_{i=1,..,n} =$$

$$=< \sum_j a_j \alpha_i^j >_{i=1,..,n} + < \sum_j b_j \alpha_i^j >_{i=1,..,n} =$$

$$=< P_a(\alpha_i) >_{i=1,..,n} + < P_b(\alpha_i) >_{i=1,..,n} = RS(a) + RS(b).$$

Similarly, it is easy to see that $RS(\lambda a) = \lambda RS(a)$ for every $a \in \mathbb{F}_q^k$ and $\lambda \in \mathbb{F}_q$. ■

Taking the standard basis for polynomials of degree $\leq k-1$ over $\mathbb{F}_q$, $\{1, x, x^2, ..., x^{k-1}\}$, we get the following generating matrix of $RS_{q,k,n}$-code:

$$\begin{pmatrix} 1 & 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & ... & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & ... & \alpha_n^2 \\ \vdots & \vdots & \vdots & ... & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \alpha_3^{k-1} & ... & \alpha_n^{k-1} \end{pmatrix} \in M_{k \times n}(\mathbb{F}_q).$$

Notice that this is Vandermonde matrix.

Lemma 14 *The minimal distance of $RS_{q,k,n}$-code is $n-k+1$*

Proof Given $a, b \in \mathbb{F}_q^k, a \neq b$, we ask in how many indices i, $P_a(\alpha_i) \neq P_b(\alpha_i)$. This is equivalent to the question: in how many i's, $(P_a - P_b)(\alpha_i) \neq 0$. $P_a - P_b$ is a polynomial of degree $k-1$, and it is nonzero since $a \neq b$. Thus, $P_a - P_b$ has at most $k-1$ roots. This means that in at least $n-(k-1)$ α_i's we have $P_a(\alpha_i) \neq P_b(\alpha_i)$. Thus, the minimal distance of the code is at least $n-k+1$. On the other hand, it is obvious that one can construct a polynomial with exactly $k-1$ roots. Thus the minimal distance is equal to $n-k+1$. ■

Corollary 15 *The $RS_{q,k,n}$-code is $(n, k, n-k+1)_q$-code.*

Remark:

1. **The projection bound:** For every (n, k, d)-code holds $k + d \leq n + 1$ (proof: in exercise 1).
   Notice that in the RS-code, the projection bound holds as equality.
2. If we take $k = \frac{n}{2}$ we get $d \approx \frac{n}{2}$, and thus both δ and r are nonzero.
3. The problem with the RS-code is that we need $q > n$. Thus, the letters themselves are big, and this can increase the likelihood of errors (if the letters are represented in bits, say). Yet, the RS-code is useful in some settings, where the errors tend to come together.