Lecture 20

Lecturer: Madhu Sudan *Scribe: Jan Vondrák*

In this lecture, we present a few hardness results for decoding problems. In summary, it turns out that decoding under very general assumptions is usually NP-hard. This means that for example, we can't expect an all-purpose algorithm that would decode all linear codes. However, it does not tell us much about the impracticality of coding because in practice we design codes with special properties.

1 Nearest Codeword Problem

Consider the nearest codeword problem for a general linear code (NCP).

Input:

- linear code defined by generator matrix G
- received codeword $r \in \mathbb{F}_2^n$
- integer t

Output: Decide whether

$$\exists c \in C(G); \Delta(c, r) \leq t.$$

Theorem 1 *(Bodekamp, McEliece, van Tilborg, 1978)*
The Nearest Codeword Problem is NP-complete.

Proof NCP is obviously in NP. To prove NP-hardness, we show a reduction to the MAX-CUT problem, which is the following:

For a given graph H and integer k, we should decide whether there is a subset of vertices S such that

$$|\{\{i,j\} \in E(H) : i \in S, j \notin S\}| \geq k.$$

The MAX-CUT problem is known to be NP-complete.

Let H be the given graph. We want to construct a code such that distances between codewords will correspond to cuts in H. Very naturally, we define the generator matrix G as the incidence matrix of H: G is a $V \times E$ matrix such that for $i \in V, e \in E$:

- $G_{ie} = 1$ if $i \in e$.
- $G_{ie} = 0$ otherwise.

Note that a codeword $x = z \cdot G \in \mathbb{F}_2^E$ corresponds to a cut where $S = \{i \in V : z_i = 1\}$, because

$$x_e = \sum_{i \in V} G_{ie} z_i = \sum_{i \in e} z_i = |e \cap S| \pmod 2$$

which is 1 if and only if e is in the cut of S. Conversely, each cut corresponds to a codeword whose weight is equal to the size of the cut. Therefore there is a cut of size $\geq k$ in H iff there is a codeword of weight at least k in $C(G)$, or in other words, there is a codeword of distance at most $m - k$ from $(1, 1, \ldots 1qy)$ where $m = |E|$ is the size of the codeword. Therefore, we define the transformation as follows

$$(H, k) \longrightarrow (G, (1, 1, \ldots 1), m - k)$$

which reduces NCP to MAX-CUT. ■

Moreover, it's NP-hard to approximate MAX-CUT within some constant factor $1 + \epsilon$. This implies that it's NP-hard to approximate the nearest codeword distance within some constant factor. However, while there exists a γ-approximation algorithm for MAX-CUT (Goemans and Williamson, $\gamma \doteq 1.139$), NCP cannot be approximated within *any* constant factor. We prove this by *amplifying* the inapproximability threshold from γ to γ^2, and hence to any constant factor.

Lemma 2 *Consider the Nearest Codeword Problem restricted to $r = (1, 1, \dots 1)$. Suppose we can decide in polynomial time whether*

$$\exists c \in C(G); \Delta(r, c) \leq t$$

or

$$\forall c \in C(G); \Delta(r, c) > \gamma^2 t$$

guaranteed than one of these is true, for some $\gamma > 1$. Then we can solve the same problem with γ instead of γ^2.

Proof Let C be a code of length m with generator matrix G. We define a larger code C^2 of length m^2. The codewords in C^2 are $m \times m$ matrices where for each $b, c^{(1)}, c^{(2)}, \dots, c^{(m)} \in C$, there is a codeword $X(b, c^{(1)}, c^{(2)}, \dots, c^{(m)}) \in C^2$ defined as follows:

$$X(i, j) = b(j) + c^{(j)}(i).$$

It is obvious that C^2 is a linear code as well. If C has k generating vectors, C^2 has $(m + 1)k$ generating vectors, which may not be independent, but that's okay. The blow-up is polynomial.

Now consider the codeword $a \in C$ of maximum weight (nearest to r). Let's choose

- $b = a$
- $c^{(j)} = a$ if $a_j = 0$
- $c^{(j)} = 0$ if $a_j = 1$

If the weight of a is $m - k$, then it's easy to see that the weight of $X(b, c^{(1)}, c^{(2)}, \dots, c^{(m)})$ is $m(m - k) + (m - k)k = m^2 - k^2$. Also, if we consider the optimal way to construct a codeword in C^2 of large weight, it turns out that $m^2 - k^2$ is the highest value we can achieve. Therefore C has a codeword of weight $\geq m - k$ if and only if C^2 has a codeword of weight $m^2 - k^2$.

This implies that if we can approximate the nearest codeword to $r = (1, 1, \dots 1)$ in C^2 within a factor of γ^2, we can do the same in C within a factor of γ, in polynomial time. ∎

By iterating this procedure a constant number of times, we arrive at a contradiction for any constant-factor approximation algorithm. We can actually iterate this even a non-constant number of times. For example, after $O(\log^{1-\epsilon} n)$ iterations, we get the following result.

Theorem 3 *If NCP can be γ-approximated in polynomial time for $\gamma = 2^{\log^{1-\epsilon} n}, \epsilon > 0$, then*

$$NP \subseteq DTIME\left(2^{poly(\log n)}\right).$$

In fact, the same γ-approximation is known to imply $P = NP$, but the proof is much more involved.

Now we turn to another result which does not speak about decoding a linear code given on input, but a linear code which can be hardwired in the algorithm. Even this is not likely to be possible in general! In other words, there are "bad" linear codes for which the Nearest Codeword Problem is NP-hard.

Theorem 4 *(Bruck,Naor) There exists a linear code C such that the Nearest Codeword Problem restricted to C is NP-hard.*

Proof Again, we want to reduce our problem to MAX-CUT. Here, however, we cannot use the code itself to simulate the graph. The code must be fixed. Therefore, we construct a code C which will (in some sense) represent the complete graph K_n. The code will have length $m = n(n-1)$ (two bits for each edge in K_n) and the codewords will be characteristic vectors of cuts in K_n, where the bit for each edge is repeated twice.

Let H be the input graph for MAX-CUT. We transform it to a vector r of length $n(n-1)$ which encodes the edges in H:

- If $(i,j) \in E(H)$ then $r(i,j) = (1,1)$.
- If $(i,j) \notin E(H)$ then $r(i,j) = (0,1)$.

Then the distance between r and a codeword c corresponding to a cut K is

$$\sum_{(i,j)\in E} 2(1-c(i,j)) + \sum_{(i,j)\notin E} 1 = \binom{n}{2} + |E| - 2|E \cap K|.$$

This achieves its minimum when $|E \cap K|$ is maximum, therefore by finding the nearest neighbor we would find the maximum cut. ■

Our last hardness result is regarding the question of determining the minimum distance of a given code.

Theorem 5 *Given a linear code on the input, it is NP-hard to determine its minimum distance.*

We reduce this problem to another decoding problem, which is Bounded Distance Decoding (BDD). This problem is specified as follows:

Input:

- linear code C
- received vector r
- integer t
- *promise that* $\Delta(C) > t$

Output: Decide whether

$$\exists c \in C; \Delta(r,c) \leq t.$$

Suppose that we can find the minimum distance in polynomial time. Then for a given instance of BDD, (C,r,t), we define a new code

$$D = [C \cup \{r\}]$$

by adding the codeword r and taking a linear closure. Assuming the distance of C is more than t, the distance of the new code will drop below t if and only if

$$\exists c \in C; \Delta(r,c) \leq t.$$

Thus we would be able to solve BDD in polynomial time.

It remains to prove the NP-hardness of BDD. We know that NCP is NP-hard to approximate, for example within a factor of 100. The idea is to use a special code C which would allow us to 100-approximate NCP using BDD. The code C should have the following properties:

- $\Delta(C) > 100t$

- $\exists r \in C; \forall c \in C; \Delta(r, c) \leq 70t$

Then we could do the following: For a given code A and vector v, we would run BDD on input

$$(A|C, v|r, 100t).$$

The $|$ symbol denotes simply concatenation of vectors or codes. Note that $\Delta(A|C) \geq \Delta(C) > 100t$. Then we consider two cases:

- If $\Delta(A, v) \leq t$ then
$$\Delta(A|C, v|r) \leq \Delta(A, v) + 70t \leq 71t.$$
- If $\Delta(A, v) > 100t$ then
$$\Delta(A|C, v|r) \geq \Delta(A, v) > 100t.$$

and we can distinguish these two cases using BDD. However, then we would be able to decide whether the nearest codeword to v in A is closer than t or farther than $100t$.

The only problem with this proof is that such a code C is not known. However, a similar proof can be given using a code C with slightly weaker properties, which can be proved to exist.