Hardness Of Approximation

Lecture 4: Subcode Covering and Proof Gap-3Lin($\frac{1}{2}+\varepsilon, 1-\varepsilon$) is NP-Hard

Instructor: Irit Dinur and Amey Bhangale *Scribe: Yotam Dikstein*

1 Reminder from Last Week

Last week we proved the following theorem:

Theorem 1.1. *For every $\varepsilon > 0$ gap-3Lin($0.99, 1-\varepsilon$) is NP-Hard.*

we now move towards proving the stronger theorem:

Theorem 1.2. *For every $\varepsilon > 0$ gap-3Lin($\frac{1}{2}+\varepsilon, 1-\varepsilon$) is NP-Hard.*

In the end of last lesson, we talked about the *subcode covering property.* Essentially, given a label cover instance, $G = (U, V, E, \Sigma_U, \Sigma_V, \pi_{u,v})$, we want to encode our assignment to the vertices of U, V by codes where for all $u \in U$:

1. We can embed the encodings of all the its neighbours $v \in N(u)$ in the encoding of u itself. Furthermore,
2. That the embedings induce a distribution that is *statistically close* to choosing a bit in the encoding of u uniformly at random. That is,, that the two following distributions are statistically close [1]:
   - Distribution D_1: Choose a bit in the encoding of u uniformly at random.
   - Distribution D_2: Choose a neighbour of u, $v \in N(u)$. Choose a bit in the encoding of v uniformly at random, and output the embedded bit in the side of u.

It is not yet clear where this property comes to play. However, one can imagine that if we have this kind of property, we might be able to encode one side of a label cover instance using the other side.

2 Attempts at Subcode Covering

In this section we describe some unsuccessful attempts for encoding label cover instances with encodings that have the subcode covering property.

2.1 Encoding with the Long Code

A naive attempt is to just use the encoding we saw in the previous lesson. Namely, We encode the assignment on u with the long code, that is.

$$\sigma \in \Sigma_u \mapsto f \in 2^{2^{\Sigma_U}},\ f(\vec{x}) = x_\sigma,$$

$$\tau \in \Sigma_V \mapsto f \in 2^{2^{\Sigma_V}},\ f(\vec{x}) = x_\tau.$$

[1] Our notion of a statistic distance for two distributions that are supported in a set A is $\Delta(D_1, D_2) = \frac{1}{2}\sum_{a\in A} |D_1(a) - D_2(a)|$, where $D_i(a) = Pr_{x\sim D_i}[x = a]$.

In the hard instances we know how to create, the alphabet for the left side is larger than the right side with a factor of at least two, i.e. $|\Sigma_U| \geq 2|\Sigma_V|$. In this case, the size of the encoding of some $v \in V$ is smaller than the encoding of $u \in \Sigma_U$ by a factor of at least $2^{2^{\frac{1}{2}|\Sigma_U|}}$.

In particular, even if u has ℓ neighbours, the support of any distribution D_2 is of size $\leq \ell 2^{2^{|\Sigma_V|}}$, much smaller than the support of D_1 (i.e. all the embeddings of the neighbours only cover a small portion of the bits of the encoding of Σ_U). Thus we have no chance to get the subcode covering property.

2.2 Encoding with the Hadamard Code

One obstacle in our previous attempt, is the fact that the size of the encoding blows up significantly when the alphabet increases - this led to the fact that any a-symmetry in the alphabets in each side, lead to the fact that one side cannot cover the other. To mend this, we will try to encode both sides with a shorter code.

2.2.1 Parrallel Repetition

Let $k \in \mathbb{N}$ and let $\Phi = c_1 \wedge ... \wedge c_m$ be some 3SAT instance. Recall the k-parrallel repetition instances defined in the first homework. Every $u \in U$ corresponds to k 3SAT clauses from Φ (ordered), every $v \in V$ corresponds to k variables (ordered). We choose an edge $(u, v) \in E$ by:

1. Choosing k clauses $c_1, ..., c_k$ from Φ uniformly at random and setting $u = (c_{i_1}, ..., c_{i_k})$.

2. Choosing one variable $x_i \in c_i$ uniformly at random, and setting $v = (x_1, ..., x_k)$.

Our assignments for Σ_U are all the partial assignments for the variables that appear in $(c_{i_1}, ..., c_{i_k})$, that satisfy all the clauses. Our assignments for Σ_V are partial assignments to the variables that appear in v. Our edge constraints for u, v are consistancy - namely that the partial assignment for the variables in v agree with the partial assignment for the variables in u. We can see that to encode a symbol in Σ_U we need $3k$ bits, and to encode a symbol in Σ_V we need k bits.

2.2.2 Hadamard Code for Parallel Repeated Instances

One possible encoding for these instances, is the Hadamard code. That is, for an assignment to $u \in U$ (respectfully $v \in V$),

$$\alpha_u = ((\alpha_{1,1}, \alpha_{1,2}, \alpha_{1,3}), ..., (\alpha_{k,1}, \alpha_{k,2}, \alpha_{k,3})).$$

we encode by the (truth table of) $h : \{0,1\}^{3k} \to \{0,1\}$

$$h_{\alpha_u}(x) = \langle \alpha_u, x \rangle.$$

For any $v \in V$ we can embed α_v in $\{0,1\}^{3k}$ by inserting 0 where we are missing an assignment to a variable. For example if $v = (x_{1,1}, x_{2,3}, x_{3,3}, ..., x_{k,2})$ then we can embed α_v by thinking about it as

$$\alpha_v = ((x_{1,1}, 0, 0), (0, 0, x_{2,3}), (0, 0, x_{3,3}), ..., (0, x_{k,2}, 0)).$$

This obviously can be carried on to the Hadamard encoding.

This embedding is better than the long code, for example we can recover any single bit of α_u by a an appropriate choice of v and x. However, note that the vector α_v is supported by k bits, and a typical u is supported by $3k$ bits. It turns out that the Hadamard encodings of k-bits look typically different than the encodings of $3k$-bits.

$$u = ((x_{1,1}, x_{1,2}, x_{1,3}), (x_{2,1}, x_{2,2}, x_{2,3}), (x_{3,1}, x_{3,2}, x_{3,3}), (x_{4,1}, x_{4,2}, x_{4,3}))$$

$$v = ((x_{1,2}), (x_{2,3}), (x_{3,1}, x_{3,2}, x_{3,3}), (x_{4,1}))$$

Figure 1: Smooth Parallel Repetition

3 A Smooth Parallel Repetition

From the Hadamard encoding attempt we learned that our problem is that we removed too many clauses from each v - thus the Hadamard encoding of each side look different. We need to remove *some* variables, or else $v = u$ and the consistency will mean nothing. Can we create a label cover instance that is hard, while we remove less variables? The answer is YES!

Definition 3.1 (Smooth Parallel Repetition)**.** *Let $0 < \beta < 1$ be some parameter and $k \in \mathbb{N}$. Let G_0 be some 3LIN instance. The beta-smooth parallel repetition of G_0 is the following label cover instance:*

1. *Choose k equations uniformly at random and set $u = (c_{i_1}, ..., c_{i_k})$.*
2. *denote $v = (d_1, ..., d_k)$. For each $1 \leq i \leq k$ with probability β set $d_i = x_i$ for some variable $x_i \in c_i$ chosen uniformly at random. With probability $1 - \beta$ set $d_i = c_i$ (i.e. all variables in the equation).*

See Figure 1.

In this setting, the usual parallel repetition is the smooth parallel repetition where $\beta = 1$.

Let's examine the hardness of this game:

1. In expectation we play the original game βk-times. Since we do an independent check for every i, with very high probability we play more than $\frac{\beta k}{2}$.
2. Suppose our original instance had soundness ≤ 0.99. The Parallel Repetition Theorem promised us that the ℓ-parallel repeated instance has soundness $0.99^{\Theta(\ell)}$. Thus if we play $\frac{\beta k}{2}$-games with high probability, then one can show that our soundness behaves like $0.99^{\frac{\beta k}{2}}$.

Thus if $\beta = \omega(\frac{1}{k})$ then the soundness of this game goes to 0 as k approaches infinity (this can of course be formulated to a proof...).

Now let's consider the size of $u \in U$ and $v \in V$. We still need $3k$ bits to encode u. For v we need $3k - 2\beta k$ bits in expectation (since with probability β we need two bits less). If we take β to be such that $\beta k \leq \sqrt{k}$, we can get the subcode covering property.

Fix $u \in U$. Consider the following distributions:

1. D_1: Choose $x \in \{0,1\}^{3k}$ uniformly at random.
2. D_2: Choose $v \in N(u)$ and $y \in \{0,1\}^{3k-2\beta}$, and embed y in $\{0,1\}^{3k}$ by v's embedding (i.e. put 0's on the variables that v doesn't support, and put the values of y where v is supported)[2]. See Figure 2.

Obviously these distributions aren't the same. Typically a vector in D_2 will have more zero's. However, if we choose *beta* small enough, these distributions are statistically close:

[2] Actually, our real distribution is to choose $v \in V$ and then choose random bits for each variable in v - this could be different than $3k - 2\beta k$ random bits. However, for simplicity we'll ignore this subtle point.

$$u = ((x_{1,1}, x_{1,2}, x_{1,3}), (x_{2,1}, x_{2,2}, x_{2,3}), (x_{3,1}, x_{3,2}, x_{3,3}), (x_{4,1}, x_{4,2}, x_{4,3}))$$

$$v = ((x_{1,2}), (x_{2,3}), (x_{3,1}, x_{3,2}, x_{3,3}), (x_{4,1}))$$

$$y = (1, 0, 1, 1, 0, 1)$$

$$\downarrow$$

$$y^{\uparrow_{u,v}} = (0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0)$$

Figure 2: D2 illustration

Claim 3.2. *The statistical difference between D_1 and D_2 is $O(\beta\sqrt{k})$. That is,*

$$\Delta(D_1, D_2) = \frac{1}{2} \sum_{x \in \{0,1\}^{3k}} |D_1(s) - D_2(s)| = O(\beta\sqrt{k}).$$

Thus our attempt to prove gap-3Lin$(\frac{1}{2} + \varepsilon, 1 - \varepsilon)$ will use the following label cover instances:

1. Begin with an instance for gap-3Lin$(0.99, 1 - \varepsilon)$.

2. Take $k = \frac{1}{\sqrt{\varepsilon}}$ and $\beta = \frac{1}{k^{0.6}}$.

3. Do a β-smooth parallel repetition on this instance. We get an instance where:

   - If we are at the YES case, i.e. the $1-\varepsilon$ case: we have an instance where at least $1-k\varepsilon =\approx 1-\frac{1}{k}$ of the edges are satisfied (we ignore β and note that the probability to full on a satisfied equation is $1-\varepsilon$ for every repetition).
   - If we are at the NO case, then by he above our soundness is $0.99^{\Theta(k^{0.4})}$.

4 The actual reduction

Now we describe the reduction of the instance above to gap-3Lin$(\frac{1}{2} + \varepsilon, 1 - \varepsilon)$.

Initially we want to encode the assignment of every $u \in \Sigma_u$ by its Hadamard code, say f_u. However, we *fold* the assignment, by requiring that for all $v \in V$, all $u_1, u_2 \in N(v)$ and all bits at y in the Hadamard encoding of the label of v: if y is embedded to $x_1 := y^{\uparrow_{u_1 v}}$ by the embedding to u_1, and is embedded to $x_2 := y^{\uparrow_{u_2 v}}$ by the embedding of u_2, then $f_{u_1}(x_1) = f_{u_2}(x_2)$. Here, $y^{\uparrow_{u_1 v}} \in \{0,1\}^{3k}$ is obtained from y by filling 0 in the 'missing' variables locations. See Figure 2.

Our linearity test is as follows (which is easily translated to a 3Lin instance):

1. Choose $u \in U$, that is, $(c_{i_1}, ..., c_{i_k})$ uniformly at random. Denote the constant in the right side of the equation c_{i_j} by a_j (i.e. $c_j : x_{j_1} + x_{j_2} + x_{j_3} = a_j$).

2. Select $x, y \in \{0,1\}^{3k}$ and $b_1, ..., b_k \in \{0,1\}$ all uniformly at random and independent.

3. Check that
$$f(x) + f(y) + f(x + y + \sum_{i=1}^{k} b_j \ell_j) = \sum_{j=1}^{k} a_j b_j.$$
where ℓ_j is the vector in $\{0,1\}^{3k}$ that is the indicator of the indexes of the variables of c_j, that is one on j_1, j_2, j_3 and 0 everywhere else.

This test is very similar to our original linearity test, studied in previous lessons:

$$f(x) + f(y) + f(x + y) = f(0).$$

4.0.1 Why do we add the extra vector $\sum_{i=1}^{k} b_j \ell_j$ to the test?

Note that if f passes the test with probability 1, then f is a linear function *and* $f(\ell_j) = a_j$ for all $1 \leq j \leq k$. consider the assignment that gives the variables x_j the value $f(e_j)$ (f acting on the vector that has 1 on the j coordinate and 0 everywhere else). Then we know that this assignment satisfies all k linear equations. Hence our test actually tests that we encoded a linear test that satisfies all the constraints.

What about soundness? We can prove the following soundness guarantee:

Claim 4.1. *Suppose that the test above passes with probability $\frac{1}{2} + \varepsilon$. Then there exists some $S \subseteq [n]$ s.t. $\chi_S(\ell_j) = (-1)^{a_j}$ and $\hat{f}(S) \geq 2\varepsilon$, where $\hat{f}(S)$ is the Fourier coefficient of f in multiplicative notation.*

In particular, f agrees with χ_S on at least $\frac{1}{2} + \varepsilon$ fraction of the inputs. In other words, the additive version of f agrees with a linear transformation, that satisfies *all* the equations, on at least $\frac{1}{2} + \varepsilon$ fraction of the inputs.

Proof. We abuse notation and denote by $f : \{0,1\}^{3k} \to \{\pm 1\}$ the multiplicative version of f. In this notation our assumption is that

$$\frac{1}{2} + \varepsilon \leq Pr[f(x)f(y)f(x + y + \sum_{j=1}^{k} b_j \ell_j)(-1)^{\sum_{j=1}^{k} a_j b_j} = 1].$$

By the same analysis we did in the second lesson,

$$Pr[f(x)f(y)f(x + y + \sum_{j=1}^{k} b_j \ell_j)(-1)^{\sum_{j=1}^{k} a_j b_j} = 1] = \frac{1}{2} + \frac{1}{2}\mathbb{E}[f(x)f(y)f(x + y + \sum_{j=1}^{k} b_j \ell_j)(-1)^{\sum_{j=1}^{k} a_j b_j}].$$

We decompose $f = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S$ and we get by a similar analysis that

$$\mathbb{E}[f(x)f(y)f(x + y + \sum_{j=1}^{k} b_j \ell_j)(-1)^{\sum_{j=1}^{k} a_j b_j}] =$$

$$\sum_{S \subseteq [n]} \hat{f}(S)^3 \mathbb{E}[\chi_S(\sum_{j=1}^{k} b_j \ell_j)(-1)^{\sum_{j=1}^{k} a_j b_j}]. \tag{4.1}$$

Fix some $S \subseteq [n]$. Since all b_j's are are chosen independently, we can write

$$\mathbb{E}[\chi_S(\sum_{j=1}^{k} b_j \ell_j)(-1)^{\sum_{j=1}^{k} a_j b_j}] = \mathbb{E}[\prod_{j=1}^{k} \chi_S(b_j \ell_j)(-1)^{a_j b_j}] = \prod_{j=1}^{k} \mathbb{E}[\chi_S(b_j \ell_j)(-1)^{a_j b_j}].$$

1. If $\chi_S(\ell_j) = (-1)^{a_j}$ then the expression in the expectation is always 1.
2. Otherwise, it is 1 when $b_j = 0$ and -1 when $b_j = 1$. Thus in expectation it is 0.

We conclude that the only terms that remain in the sum are those where $\chi_S(\ell_j) = (-1)^{a_j}$ for all $1 \leq j \leq k$.

Hence we get that (4.1) is

$$= \sum_{\substack{S\subseteq[n], \\ \forall j, \chi_S(\ell_j)=(-1)^{a_j}}} \hat{f}(S)^3 \leq \max_{\substack{S\subseteq[n], \\ \forall j, \chi_S(\ell_j)=(-1)^{a_j}}} \hat{f}(S) \sum_{\substack{S\subseteq[n], \\ \forall j, \chi_S(\ell_j)=(-1)^{a_j}}} \hat{f}(S).$$

In conclusion, we get that

$$\max_{\substack{S\subseteq[n], \\ \forall j, \chi_S(\ell_j)=(-1)^{a_j}}} \hat{f}(S) \geq 2\varepsilon.$$

$\square$