Hardness Of Approximation

Lecture 2: Hardness of gap$(0.99, 1-\varepsilon)$-3LIN, LTCs, and Fourier Analysis

Instructor: Irit Dinur and Amey Bhangale *Scribe: Orr Paradise*

The PCP theorem implies $\mathcal{NP}$-hardness of approximating formula satisfiability. In this lecture we will take our first steps towards showing $\mathcal{NP}$-hardness of approximating solutions to linear equations.

Definition 0.1. An instance of gap(s, c)-3LIN is a system of linear equations over $\mathbb{F}_2$, with each equations in the system consisting of three variables.[1] The problem is distinguishing between instances for which there is an assignment that satisfies at least a c-fraction of equations, and instances in which no assignment satisfies more than an s-fraction of equations.

Theorem 0.2. *For all $\varepsilon > 0$, gap$(\frac{1}{2}+\varepsilon, 1-\varepsilon)$-3LIN is $\mathcal{NP}$-hard.*

Theorem 0.2 is an example of the strong results obtained using the PCP theorem: For any instance of 3LIN, finding an assignment that satisfies at least half of the equations is trivial (try the all-true and all-false assignments), and if the instance is solvable then a solution can be efficiently found by Gaussian elimination. However, if the instance is even *slightly* not-solvable (i.e. almost all of its equations can be simultaneously satisfied), then finding a solution even *slightly* better than trivial is hard.

Today we will develop some of the tools used in obtaining this result, and obtain a weaker version of it:

Theorem 0.3 (Theorem 0.2, weaker)**.** *For all $\varepsilon > 0$, gap$(0.99, 1-\varepsilon)$-3LIN is $\mathcal{NP}$-hard.*

1 Discrete Fourier analysis

Remark 1.1. For convenience, we switch to multiplicative Boolean notation: bit $b \in \{0,1\}$ is replaced with $(-1)^b \in \{\pm 1\}$, and a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is replaced with $h\colon \{0,1\}^n \to \{\pm 1\}$ given by $h(x) = (-1)^{f(x)}$.

We define an inner product on $\{\pm 1\}^{2^n}$ by $\langle f, g\rangle := \mathbb{E}_x[f(x)g(x)]$. The *Fourier characters* are $\chi_a\colon \{0,1\}^n \to \{\pm 1\}$ given by $\chi_a(x) := (-1)^{\sum_{i=1}^n a_i x_i}$ for each $a \in \{0,1\}^n$. Observe that these functions are *multiplicative*, i.e. $\chi_a(x+y) = \chi_a(x)\cdot\chi_a(y)$, which is analogous to *linearity* in the different notation of Remark 1.1.

Claim 1.2. *The Fourier characters form an orthonormal basis for $\{\pm 1\}^{2^n}$.*

Proof. For any $a, b \in \{0,1\}^n$,

$$\langle \chi_a, \chi_b\rangle = \mathbb{E}_x\left[\prod_{i=1}^n (-1)^{(a_i+b_i)x_i}\right] = \prod_{i=1}^n \left[\mathbb{E}_{x_i}\left[(-1)^{(a_i+b_i)x_i}\right]\right]$$

where the rightmost inequality uses independence of the x_i's. Now, for any $i \in [n]$, if $a_i \neq b_i$ then $\mathbb{E}_{x_i}[(-1)^{(a_i+b_i)x_i}] = \mathbb{E}_{x_i}[(-1)^{x_i}] = 0$ and so if $a \neq b$ we have $\langle \chi_a, \chi_b\rangle = 0$. On the other hand, if $a_i = b_i$ then $\mathbb{E}_{x_i}[(-1)^{(a_i+b_i)x_i}] = \mathbb{E}_{x_i}[(-1)^0] = 1$, therefore if $a = b$ then $\langle \chi_a, \chi_b\rangle = 1$. We showed that $\{\chi_a\}_{a\in\{0,1\}^n}$ is a set of 2^n orthonormal vectors and thus is an orthonormal basis for $\{\pm 1\}^{2^n}$. $\square$

[1] We assume instances do not have contradictions. That is, that each equation in the system has an assignment that satisfies it.

As with any orthonormal basis, each function $f\colon \{0,1\}^n \to \{\pm 1\}$ can be uniquely written as a linear combination of the Fourier characters; $f = \sum_{a\in\{0,1\}^n} \widehat{f}(a)\chi_a$, where $\widehat{f} := \langle f, \chi_a\rangle$. By orthonormality,

$$\langle f, f\rangle = \left\langle \sum_a \widehat{f}(a)\chi_a, \sum_b \widehat{f}(b)\chi_b \right\rangle = \sum_{a,b} \widehat{f}(a)\widehat{f}(b)\,\langle \chi_a, \chi_b\rangle = \sum_a \widehat{f}(a)^2$$

This identity, known as Parseval's equality, implies that $\sum_a \widehat{f}(a)^2 = \mathbb{E}_x[f(x)^2] = 1$, since $f(x) \in \{\pm 1\}$.

These basic facts will suffice for now, but we've barely scratched the surface discrete Fourier analysis—the reader is enthusiastically referred to [ODo14] for more.

2 Locally testable codes

Theorem 0.3 is proved by a reduction from (a strong version of) the PCP theorem which replaces each constraint with a gadget based on a *locally testable code (LTC)*. An *error correcting code* $C \subseteq \{0,1\}^n$ is a set of *codewords* such that any two distinct codewords are far apart. Such a code is *locally testable* if it admits a *tester* T that distinguishes between inputs in the code and those far from it based only on a few queries. A formal and deeper discussion can be found in [Gol17, Chapter 13]; we move on to two concrete examples.

2.1 The Hadamard code

Viewing elements of $\{0,1\}^{2^n}$ as Boolean function on n bits, the *Hadamard code*, denoted $\mathbf{Had} \subseteq \{0,1\}^{2^n}$, consists of all *linear* functions Boolean functions; that is, $f \in C$ if and only if for all $x, y \in \{0,1\}^n$ it holds that $f(x) + f(y) = f(x+y)$, with addition over $\mathbb{F}_2$. The *local test* tests that this property holds for a random choice of x and y as follows.

Algorithm 2.1 (Hadamard codeword test)**.** Given access to a function $f\colon \{0,1\}^n \to \{0,1\}$:

1. Sample $x, y \in \{0,1\}^n$ uniformly at random.
2. Query $f(x)$, $f(y)$ and $f(x+y)$.
3. Accept if and only $f(x) + f(y) = f(x+y)$.

The tester issues three queries to f, and it is clearly *complete*: it accepts a linear f with probability 1. The soundness of the Hadamard tester is captured by the following claim.

Claim 2.2. *For any $\varepsilon \in [0, 1/2]$, if $f\colon \{0,1\}^n \to \{0,1\}$ is accepted by the Hadamard tester (Algorithm 2.1) with probability at least $\frac{1}{2}+\varepsilon$, then there exists a Hadamard codeword $h \in$ **Had** such that $\mathbb{P}_x[f(x) = h(x)] \geq \frac{1}{2}+\varepsilon$.*

Proof. Fourier analysis is more convenient in multiplicative notation, so we will equivalently show that for each $g\colon \{0,1\}^n \to \{\pm 1\}$, if $\mathbb{P}_x[g(x)g(y) = g(x+y)] \geq \frac{1}{2}+\varepsilon$ (i.e. the test accepts g w.p. at least $\frac{1}{2}+\varepsilon$) then there is $a \in \{0,1\}^n$ such that $\mathbb{P}_x[g(x) = \chi_a(x)] \geq \frac{1}{2}+\varepsilon$. This suffices, as the Fourier characters $\{\chi_a\}_a$ form the Hadamard code **Had**.

The first step is tying the *agreement* of g with any χ_a to g's respective Fourier coefficient:

$$\widehat{g}(a) = \mathbb{E}_x[g(x)\chi_a(x)] = \mathbb{P}_x[g(x) = \chi_a(x)] + (-1)\mathbb{P}_x[g(x) \neq \chi_a(x)] = 2\cdot \mathbb{P}_x[g(x) = \chi_a(x)] - 1 \qquad (1)$$

At the other end, we tie the acceptance probability of g with its Fourier coefficients, observing that the equation $g(x)g(y)g(x+y) = 1$ holds if and only g is accepted when the tester samples $x, y \in \{0,1\}^n$, and is equal to -1 otherwise. Thus

$$\mathbb{E}_{x,y}[g(x)g(y)g(x+y)] = 2 \cdot \mathbb{P}[g \text{ is accepted}] - 1 \geq 2\varepsilon \tag{2}$$

Combining Equations (1) and (2), what's left is to find an $a \in \{0,1\}^n$ with $\widehat{g}(a) \geq \mathbb{E}_{x,y}[g(x)g(y)g(x+y)]$. To do this, we utilize our newly-gained knowledge in Fourier analysis:

$$\begin{aligned}
\mathbb{E}_{x,y}[g(x) \cdot g(y) \cdot g(x+y)] &= \mathbb{E}_{x,y}\left[\left(\sum_a \widehat{g}(a)\chi_a(x)\right) \cdot \left(\sum_b \widehat{g}(b)\chi_b(y)\right) \cdot \left(\sum_c \widehat{g}(c)\chi_c(x+y)\right)\right] \qquad (3)\\
&= \sum_{a,b,c} \widehat{g}(a) \cdot \widehat{g}(b) \cdot \widehat{g}(c) \cdot \mathbb{E}_{x,y}[\chi_a(x)\chi_b(y)\chi_c(x+y)] \\
&= \sum_{a,b,c} \widehat{g}(a) \cdot \widehat{g}(b) \cdot \widehat{g}(c) \cdot \mathbb{E}_x[\chi_a(x)\chi_c(x)] \cdot \mathbb{E}_y[\chi_b(y)\chi_c(y)] \\
&= \sum_a \widehat{g}(a)^3
\end{aligned}$$

Where the last two equations use the independence of x and y, multiplicativity of χ_c, and orthonormality of the Fourier characters. Lastly, we recall that $\sum_a \widehat{g}(a)^2 = 1$ since g is Boolean, so letting $a_{\max}$ be a maximizer of $\max_a \widehat{g}(a)$, we have

$$\mathbb{E}_{x,y}[g(x) \cdot g(y) \cdot g(x+y)] = \sum_{a,b,c} \widehat{g}(a)^3 \leq \widehat{g}(a_{\max}) \cdot \sum_a \widehat{g}(a)^2 = \widehat{g}(a_{\max})$$

□

2.2 LTC soundness: 99% vs. 1%

In general, the soundness of LTC tests has different interpretations depending on the correlation of the input with the code.

- Suppose the test passes with probability 99%. A *stability* result shows that if an input passes with high probability then it is close to some codeword. An example for such a result is Claim 2.2 when ε is close to $1/2$.
- At the other end, if an input passes the codeword test with probability slightly better (say, 1% more) then a random input, then it is nontrivially correlated with a codeword. Claim 2.2 resides in this regime as well, when taking ε to be close to 0.
  - As a follow-up, we might seek to obtain a *list-decoding bound* on the number of codewords that can be nontrivially correlated with the input. For the Hadamard code, this corresponds to a bound on the number of $a \in \{0,1\}^n$ for which $\widehat{g}(a) \geq \varepsilon$. Recalling that $\sum_a \widehat{g}(a)^2 = 1$ for Boolean g, we have that at most $1/\varepsilon^2$ codewords can be ε-correlated with g.[2]

2.3 The long code

We say that $f\colon \{0,1\}^n \to \{0,1\}$ is a *dictator* function if there exists $i \in [n]$ such that for all $x \in \{0,1\}^n$ it holds that $f(x) = x_i$. The *long code* consists of all dictator functions on n bits, and is denoted by **Dict**. It's local test is described below:

[2]In fact, this bound is tight: A Boolean function $f\colon \{0,1\}^k \to \{\pm 1\}$ is *bent* if $|\widehat{f}(a)| = 2^{-k/2}$ for all a. For any $\varepsilon = 2^{-k}$ consider a bent function $f\colon \{0,1\}^k \to \{\pm 1\}$ with domain extended to $\{0,1\}^n$ by "ignoring" the last $n-k$ coordinates. For more on bent functions and their applications, see [ODo14, Section 6.3].

Algorithm 2.3. Fix a parameter $\delta \in [0,1]$. Given access to a function $f\colon \{0,1\}^n \to \{0,1\}$:

1. Sample $x, y \in \{0,1\}^n$ uniformly at random.
2. Sample $\mu \in \{0,1\}^n$ according to the following process. For each $i \in [n]$, with probability δ sample μ_i uniformly at random from $\{0,1\}$, and with probability $1-\delta$ set $\mu_i = 0$.
3. Accept if and only if $f(x) + f(y) = f(x+y+\mu)$.

Algorithm 2.3 is the same as the Hadamard code test (Algorithm 2.1), except for the addition of a *noise vector* μ which is used to distinguish between a dictator function and any other linear function.

Claim 2.4 (Completeness of the long code test)**.** *If* $f \in$ **Dict** *then the long code test accept with probability* $1-\delta/2$*.*

Proof. Suppose $f(z) = z_i$ for all z. The test passes if and only if $\mu_i = 0$, which occurs with probability $1-\delta/2$. □

The soundness claim is proved using Fourier analysis so we switch back to multiplicative notation, replacing addition with multiplication and bit b with $(-1)^b$. In particular, the test checks that $g(x)g(y) = g(x+y+\mu)$, where μ_i is uniformly sampled from $\{\pm 1\}$ with probability δ, and is set to be 1 with probability $(1-\delta)$.

Claim 2.5 (Soundness of the long code test)**.** *For* $a \in \{0,1\}^n$*, the* Hamming weight *of* a*, denoted* $|a|$*, is the number of* $i \in [n]$ *for which* $a_i = 1$*. For all* $\delta \in [0,1]$*, if* $g\colon \{0,1\}^n \to \{\pm 1\}$ *is accepted by the long code test with probability at least* $\frac{1}{2} + \varepsilon$ *then* $\max_a \widehat{g}(a) \cdot (1-\delta)^{|a|} \geq 2\varepsilon$*.*

Claim 2.5 means that if g passes the test with good probability then not only is it correlated with a multiplicative function χ_a, but it must be that a is sparse, i.e. that χ_a depends on few variables.

Proof. The proof follows the proof of Claim 2.2, except we need to account for the noise vector in Equation (3). For each $a \in \{0,1\}^n$,

$$\mathbb{E}_\mu[\chi_a(\mu)] = \mathbb{E}_\mu\left[\prod_{i=1}^{n} (-1)^{a_i \mu_i}\right] = \prod_{i=1}^{n} \mathbb{E}_{\mu_i}[(-1)^{a_i \mu_i}] = (1-\delta)^{|a|}$$

where the rightmost equality is because if $a_i = 0$ then $\mathbb{E}_{\mu_i}[(-1)^{a_i\mu_i}] = 1$, and otherwise $\mathbb{E}_{\mu_i}[(-1)^{a_i\mu_i}] = \mathbb{E}_{\mu_i}[(-1)^{\mu_i}] = 1-\delta$.

Now, just as we calculated Equation (3), we have

$$\begin{aligned}\mathbb{E}_{x,y}[g(x)\cdot g(y)\cdot g(x+y+\mu)] &= \sum_{a,b,c} \widehat{g}(a)\cdot\widehat{g}(b)\cdot\widehat{g}(c)\cdot \mathbb{E}_x[\chi_a(x)\chi_c(x)]\cdot\mathbb{E}_y[\chi_b(y)\chi_c(y)]\cdot\mathbb{E}_\mu[\chi_c(\mu)] \\ &= \sum_a \widehat{g}(a)^3 \cdot (1-\delta)^{|a|} \leq \max_a \widehat{g}(a)\cdot(1-\delta)^{|a|}\end{aligned}$$

□

3 Hardness of gap$(0.99, 1-\delta)$-3LIN

With these tools in hand, we may now begin working towards a proof of Theorem 0.3. The proof is by reduction from (a strong version of) the PCP theorem:

Theorem 3.1. *For all* $\varepsilon > 0$*,* gap$(\varepsilon, 1)$-`LabelCover` *is* $\mathcal{NP}$*-hard. Furthermore,* $\mathcal{NP}$*-hardness holds even when instances* $G = (U, V, E, \Pi)$ *are guaranteed to have the following properties:*

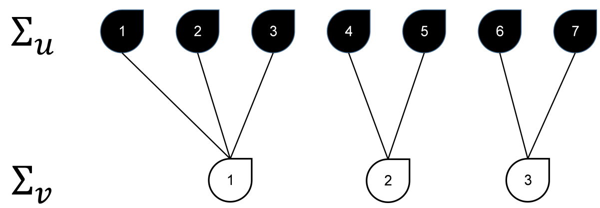

Figure 1: An example of a projective constraint $\Pi_{u,v} \colon [7] \to [3]$. Notice how the top layer *projects* onto the bottom one.

- Biregularity: *any two vertices on the same side (U or V) have the same degree.*
- Projectivity: *for any $\{u, v\} \in E$, each $\sigma_u \in \Sigma_u$ has exactly one $\sigma_v \in \Sigma_v$ such that $(\sigma_u, \sigma_v) \in \Pi_{u,v}$. In other words, we can think of $\Pi_{u,v}$ as a function $\Pi_{u,v} \colon \Sigma_u \to \Sigma_v$. (See Figure 1)*

The main idea behind the reduction is to replace each vertex w with $2^{|\Sigma_w|}$ variables (see Figure 2), and add linear equations asserting that an assigment to these variables corresponds to a long code of a label $\sigma_w \in \Sigma_w$ that satisfies the constraints in which w participates. Rather than explicitly describing the system of linear equations, we shall describe a *probabilistic verifier* of these assertions—the equation system is obtained by enumerating over the random coins of the verifier just as in the equivalence of the "proof system" and the "gap($\cdot$)-`LabelCover`" views of the PCP theorem.

Given a biregular and projective instance $G = (U, V, E, \Pi)$ of `LabelCover`, create $2^{|\Sigma_w|}$ variables for each $w \in U \cup V$. Given access to an assignment, the verifier runs as follows:

1. Sample a uniformly random edge $\{u, v\} \in E$. Denote the assignment to the $2^{|\Sigma_u|}$ variables created from u by $f \colon \{0,1\}^{\Sigma_u} \to \{0,1\}$. Similarly, let $g \colon \{0,1\}^{\Sigma_v} \to \{0,1\}$ denote the assignment to the $2^{|\Sigma_v|}$ variables obtained from v.
2. Do one of the following tests with probability $1/3$ each:
   (a) Run the long code test (Algorithm 2.3) on f.
   (b) Run the long code test on g.
   (c) Check that the labels (allegedly) encoded by f and g are *consistent* with the constraint $\Pi_{u,v}$: uniformly sample $x \in \{0,1\}^{\Sigma_v}$ and $y \in \{0,1\}^{\Sigma_u}$ and check that
   $$f(y) + f(\widetilde{x} + y) = g(x) \tag{4}$$
   where, for each $\sigma \in \Sigma_u$, the σth coordinate of $\widetilde{x}$ is $\widetilde{x}_\sigma \coloneqq x_{\Pi_{u,v}(\sigma)}$.

Before turning to the analysis of the reduction, let's have another look at the consistency test. Suppose f and g are indeed long codes of labels $\sigma_u \in \Sigma_u$ and $\sigma_v \in \Sigma_v$, respectively. Then, $f(\widetilde{x}) = \widetilde{x}_{\sigma_u} = x_{\Pi_{u,w}(\sigma_u)}$ and $g(x) = x_{\sigma_v}$. Now, if σ_u and σ_v satisfy $\Pi_{u,v}$ (i.e., $\Pi_{u,v}(\sigma_u) = \sigma_v$)) then Equation (4) holds for *any* choice of x, y.

Why do we add and subtract $f(y)$ to the consistency check? While x is uniformly distributed in $\{0,1\}^{\Sigma_v}$, $\widetilde{x}$ is not at all uniformly distributed in $\{0,1\}^{\Sigma_u}$, so if we were to check only that $f(\widetilde{x}) = g(x)$, an adversary could corrupt an encoding of a non-satisfying label on the support of $\widetilde{x}$ so that the consistency check passes. In other words, step 2a tests that f is close to some long code codeword f', but such closeness only guarantees that f and f' agree on a uniformly random input, while $\widetilde{x}$ is not uniformly random. This pitfall is avoided using *self-correction:* with $y \in \{0,1\}^{\Sigma_u}$ distributed uniformly at random, the queries to y and $y + \widetilde{x}$ are uniformly random (though dependent), and if f was a valid codeword then the self-corrected value $f(y) + f(y + \widetilde{x})$ will equal the original one $f(\widetilde{x})$.

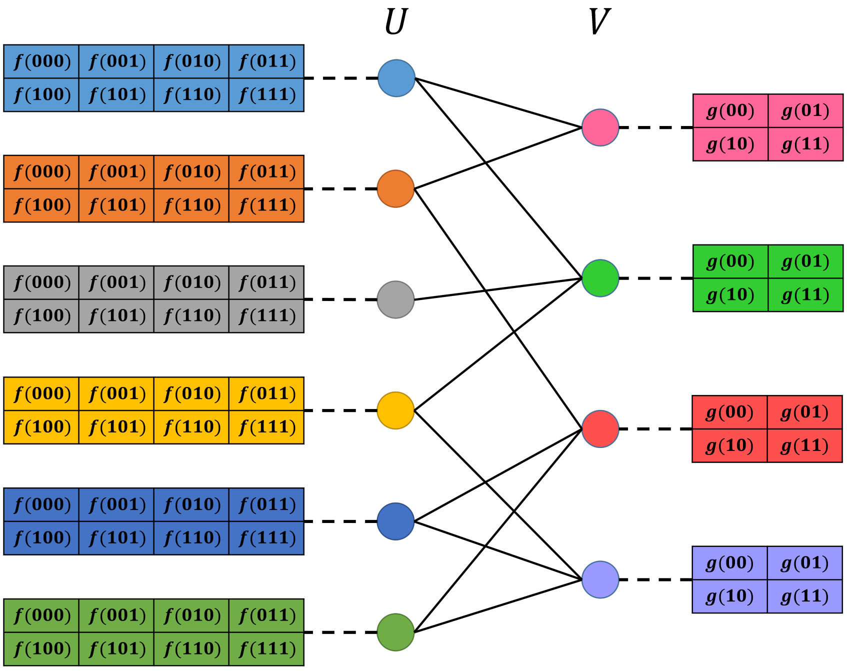

Figure 2: The reduction replaces each vertex (circle) with variables corresponding to the long code of a label (square). In this example, $|\Sigma_u| = 3$ for each $u \in U$ and $|\Sigma_v| = 2$ for each $v \in V$.

3.1 Up next

The reduction *almost* works (consider the all-zeros assignment). In the next lecture, we will fix the reduction and analyze its soundness, proving Theorem 0.3.

References

[ODo14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. ISBN: 978-1-10-703832-5. URL: http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions.

[Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. ISBN: 978-1-107-19405-2.