18.405J/6.841J: Advanced Complexity Theory Spring 2016

Lecture 18: Unique Games Conjecture and Hardness for MAX-2LIN

Prof. Dana Moshkovitz

Scribe: Anonymous Student
Scribe Date: Spring 2016

1 Optimal inapproximability results

A major application of ideas related to the PCP theorem is in proving optimal hardness/inappoximability results for certain problems. In fact, there is a general framework for proving such results. One starts with a "strong" PCP result (usually phrased in terms of projection games or unique games), and applies an encoding scheme based on the *long code*, which leads to an optimal hardness result. In this lecture, we will apply these ideas to Max-Cut/2LIN. First, consider a harder problem.

Theorem 1. *For any $\varepsilon > 0$, it is NP-hard given a system of equations of the form $\{x + y + z = C(mod\ 2)\}$ to distinguish between the cases:*

- *At least $1 - \varepsilon$ fraction of equations can be satisfied*
- *At most $1/2 + \varepsilon$ fraction of equations can be satisfied*

Note that deciding whether all equations can be satisfied is easy (via Gaussian elimination) and finding an assignment that satisfies half of equations is easy (via random assignment). So the above theorem essentially tells us that we can't even beat randomness by a little bit. Pictorially, if one imagines a graph of time required vs. approximation error for this problem, the time required would be very small and flat up to $1/2$ approximation error, and then there would be an extremely steep spike at $1/2$ all the way to exponential time (assuming 3SAT takes $2^{\Omega(n)}$ time), and then the plot would remain flat again all the way to 1.

The situation is quite different for 2LIN, in which case an algorithm due to Goemans and Williamson tells us that we *can* distinguish between the following cases:

- At least $1 - \varepsilon$ fraction of equations can be satisfied
- At most $1 - \Theta(\sqrt{\varepsilon})$ fraction of equations can be satisfied

Is this result tight? We can't prove it, but we have a conjecture:

Theorem 2. *Assuming the Unique Games Conjecture, $\forall \varepsilon > 0$, it's NP-hard to distinguish, given a 2LIN instance, between the cases*

- *At least $1 - \varepsilon$ fraction of equations can be satisfied*

- *At most* $1 - l\sqrt{\varepsilon}$ *fraction of equations can be satisfied for an appropriate constant* $l > 0$

The remainder of the lecture will define the UGC and sketch the construction for the proof of the above theorem.

2 Projection games

Definition 1. *(Projection games) A* projection game G *is defined by*

1. *A bipartite graph* $G = (X, Y, E)$ *where* X, Y *are the vertex sets and* E *is the set of edges.*
2. *Alphabets associated with* X *and* Y*:* Σ_X, Σ_Y*.*
3. *Projections:* $\forall e \in E$, $\pi_e : \Sigma_X \to \Sigma_Y$*.*

The game is to find vertex assignments $a : X \to \Sigma_X$, $b : Y \to \Sigma_Y$ *which maximize the number of satisfied edges of* G*, where an edge* $e = (x, y)$ *is satisfied if and only if* $\pi_e(a(x)) = b(y)$*. The value of a projection game instance* $\mathrm{val}(G)$ *is equal to the maximum fraction of satisfied edges.*

Note that this problem also goes by the name "label cover". We have the following result related to projection games:

Theorem 3. $\forall \varepsilon$*, given a projection game* G *with alphabet size poly*$(1/\varepsilon)$*, it is NP-hard to distinguish between the cases*

- $\mathrm{val}(G) = 1$
- $\mathrm{val}(G) \leq \varepsilon$

Notice that even if you find an assignment with a short list of labels (say, l labels) for each vertex, and the graph allowing l labels has value $\mathrm{val}_l(G)$, you can get an assignment with one label per vertex and value $\mathrm{val}(G) \geq \frac{1}{l^2}\mathrm{val}_l(G)$ by random guessing. This fact will be used in the hardness result for 2LIN in Section 4.

3 Unique games

Definition 2. *(Unique games) A* unique game *is a projection game where,* $\forall e$*,* π_e *is one-to-one. We can take* $\Sigma_X = \Sigma_Y$ *and drop the bipartiteness requirement.*

Note that unique games is easier than projection games. In fact, it is easy to determine if $\mathrm{val}(G) = 1$ for a unique game. Pick some starting vertex v, and try some assignment at that vertex. Since the projections are bijective, the required values of all of the other vertices that lie in the same connected component as v are uniquely determined. Iterate through the possible values of the starting vertex to decide if the value of that connected component is 1. We now state the Unique Games Conjecture.

Conjecture 1. *(Unique Games Conjecture)* $\forall \delta, \varepsilon > 0$*, given a unique game* G *with alphabet size* $k = k(\varepsilon, \delta)$*, it is NP-hard to distinguish between the cases*

- $\text{val}(G) \geq 1 - \delta$
- $\text{val}(G) \leq \varepsilon$

(sidenote: $k = poly(1/\varepsilon) \cdot exp(1/\delta)$*)*

4 Inapproximability of 2LIN from UGC

We now sketch the construction which one can use to prove the optimal inapproximability result for 2LIN, based on the UGC. This result would prove that the Goemans-Williamson algorithm is tight. The idea is to reduce SAT to the gapped 2LIN problem as stated in Theorem 2. Assuming the UGC, we first reduce SAT to an instance of a unique game G. The next step will be to replace each vertex in G by a cluster of many vertices corresponding to an encoding of each vertex via the *long code*. For the final step, the vertices correspond to variables in the 2LIN instance, and labels of the vertices correspond to assignments of the variables.

Definition 3. *(Long code) The (binary)* long code *encodes* $a \in \Sigma$ *by* $\{f(a)\}$*,* $\forall f : \Sigma \to \{\pm 1\}$.

It is clear that this encoding is extremely inefficient: $\log |\Sigma|$ bits are being encoded in $2^{|\Sigma|}$ bits! However, since $|\Sigma| = \Theta(1)$ for us, this is not a concern. One very important aspect of the long code for our purposes is a certain permutation property. Consider a one-to-one map $\pi : \Sigma \to \Sigma$ and any $a \in \Sigma$. Now consider the long code encoding of $\pi(a)$. For any function $f : \Sigma \to \{\pm 1\}$, a bit in the encoding of $\pi(a)$ is $f(\pi(a))$. But notice that this corresponds to a bit in the long code encoding of a, namely, that bit corresponding to the function $f \circ \pi$. In other words, if we denote the encoding of a by $LC(a)$, we have $(LC(a))_{f \circ \pi} = (LC(\pi(a)))_f$ where we are using functions to index the bits in the encoding. Hence, we see that for any projector π, there is a corresponding permutation of the encoding of the label.

Now, we can add edges between the clusters of vertices in our modified graph which correspond to the above permutation. So now we have seen how to implement the long code encoding on our unique games instance, but we must also enforce the encoding scheme. This is accomplished with a "noise test". The rough idea is as follows. Given $f : \Sigma \to \{\pm 1\}$, define $\tilde{f}(x)$ to equal $f(x)$ with probability $1-\delta$ and $-f(x)$ with probability δ. Now, for the noise test, we are given a function $F : \{f\} \to \{\pm 1\}$, we pick random $f, \tilde{f}$, and we check whether $F(f) = F(\tilde{f})$. There is a lemma which roughly says that if the noise test passes with probability at least $1 - l\sqrt{\delta}$, then F corresponds to a short list of labels $a \in \Sigma$. Hence, by properly adding edges within each cluster, we can enforce that each cluster corresponds to a short list of assignments. The fact we noted in Section 2 relating the value of a projection game allowing a short list of labels to a normal projection game applies here as well, and so we can complete the reduction to 2LIN as desired.

18.405J / 6.841J Advanced Complexity Theory

Spring 2016