CS369H: Hierarchies of Integer Programming Relaxations Spring 2016-2017

Lecture 8: Statistical vs Computational Complexity Tradeoffs via SoS

Pravesh Kothari *Scribes: Carrie and Vaggos*

Overview We show how Sum of Squares can be used to understand computational thresholds for average-case-complexity in problems concerned with recovering hidden structures/signals surrounded by random noise. Examples include Planted Clique, Sparse PCA, and Refuting Random Constraint Satisfaction Problems (CSPs).

1 The Planted Clique Problem

We are given a random graph G on n-vertices. We are promised that there are two possible generative models (distributions) for this graph, and we want to figure out **from which of the two distributions** the graph G was sampled. The options are:

- $G(n, \frac{1}{2})$ – The Erdos-Renyi graph on n-vertices with $p = \frac{1}{2}$ (each edge is present independently with probability $\frac{1}{2}$).
- $G(n, \frac{1}{2}, \omega) = G(n, \frac{1}{2}) + K_\omega$, where K_ω is a clique on ω-vertices (which vertices exactly are not known but you know that the size of the clique is ω). The way to think about this process is that you first generate a random graph from $G(n, \frac{1}{2})$, then you choose a uniformly random set of vertices of size ω and then you complete the clique on those chosen ω vertices.

So the question is, for what ω can we distinguish (correctly and with high probability[1]) between the two cases above, and how? We would like to emphasize that the information theoretical question is *when is it possible* to distinguish between the two distributions given the graph (in other words, when does our graph contain enough information) and to answer that question we are **computationally unbounded** and that the computational/algorithmic question is *when can we efficiently*[2] find the hidden structure, if there is one. You can think of the first question as the *decision* version of a problem and the second question as its *search* version. Here the hidden signal/structure in the input is the clique and the *strength* of the signal is the size of the clique, that's why we try to get some result depending on the natural parameter ω. As we will see later, for Sparse PCA and Refuting Random CSPs we will have different natural strength parameters.

In graphs drawn from the standard Erdos-Renyi graph distribution $G(n, \frac{1}{2})$, we expect cliques of size at most $(2 + o(1)) \log n$[3]. Using an (inefficient) algorithm to compute the max clique of the given graph we can distinguish between the two cases if $\omega > 2 \log n$. Therefore the **statistical threshold** for this problem is $\omega \approx 2 \log n$, because if $\omega < 2 \log n$, it is impossible to distinguish between a planted clique model and $G(n, \frac{1}{2})$.

[1] With high probability (w.h.p.) means $1 - 1/poly(n)$ as it is usually the case.

[2] i.e. in polynomial time.

[3] Note that the constants here matter a lot. In fact, finding a clique of size $1 \cdot \log n$ can be done efficiently using a greedy algorithm, whereas we don't know how to find a slightly larger clique of size $1.1 \cdot \log n$ in polynomial time.

But the computational threshold (the clique size threshold at which there are efficient algorithms that can distinguish whether there is a planted clique of that size or not) is $\omega \in \Theta(\sqrt{n})$. There is a huge gap between the statistical and computational threshold for this problem. So what's the distinguisher/algorithm to solve the problem when $\omega \in \Theta(\sqrt{n})$?

The state of the art algorithm for planted clique is the following and is really simple. Define $A(G)$ as:

$$A(G)(i,j) = \begin{cases} 1 & \text{if } (i,j) \sim G \\ -1 & \text{otherwise} \end{cases}$$

We set the diagonal of A to be 1, so that the following computation is cleaner, although even if it was 0, everything we say here can still be made to work (by just replacing $(\omega-1)^2$ instead which wouldn't affect the asymptotic behaviour). With high probability, if $G \sim G(n, \frac{1}{2})$, $||A||_\infty \leq \Theta(\sqrt{n})$. On the other hand, if a graph is drawn from the distribution with the planted clique of size ω, we can show that this clique would induce a large operator norm for A and to be precise $||A||_\infty \geq \omega$[4]. To see this, take the $\{0,1\}$ vector x on the graph's vertices which is the indicator vector for the clique. Then, $x^T A x = \omega^2 \implies \frac{x^T A x}{x^T x} = \frac{\omega^2}{\omega} = \omega \implies ||A||_\infty \geq \omega$. In particular, adding a clique of size $\omega \in \Theta(\sqrt{n})$ would be enough to make the algorithm succeed.

The algorithm that we used belongs to the family of *spectral* algorithms and we say that $\omega \in \Theta(\sqrt{n})$ is the computational threshold for this problem, because we know of no other way to obtain better results (i.e. smaller values of ω) in polynomial time. Let's now move on to our second example of this broader "Statistical vs Computational thresholds" question.

2 Sparse PCA

This is the Sparse Principal Components Analysis [MW15, BR13] and again we will be asked to distinguish between two different distributions. Here however, we will observe multiple samples of the distributions (and not just one as it was the case in Planted Clique) and the natural strength parameter will be the number n of samples that we observe. The two different cases we want to distinguish are the following:

- Standard normal distribution $\mathcal{N}(0, \mathcal{I}), \mathcal{I} \in \mathbb{R}^{p \times p}$ (p-dimensional Gaussian).
- Normal distribution with a "sparse spike" $\mathcal{N}(0, \Sigma), \Sigma = \mathcal{I} + \lambda v v^T$, where v is k-sparse, i.e.

$$v_i \in \{-1, 0, 1\}, \forall i \in \{1, 2, \ldots, p\}$$

$$\sum_{i=1}^{p} v_i^2 = k$$

Let's say we observe n samples, all from one of the above 2 distributions. Since the key difference between the two cases is located in the covariance matrix ($\mathcal{I}$ vs Σ), the natural approach here is to compute the empirical covariance matrix. Below, we give **two** algorithms (one inefficient and one efficient) for distinguishing between these two distributions. They usually go by the name of "sparse eigenvalue programs":

[4] As you can see, this is actually true for any graph with a clique of size ω.

- Common step for both algorithms: Build sample covariance matrix: $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$

- 1st algorithm (inefficient): Solve the following polynomial program[5] PP1:

$$\max \langle x, \hat{\Sigma} x \rangle$$

$$\sum_{i=1}^{p} x_i^2 = k \quad (k\text{-sparsity constraint})$$

$$x_i^3 = x_i, \forall i \in \{1, 2, \ldots, p\}$$

- 2nd algorithm (efficient): Relax the above to get the following SDP:

$$\max \operatorname{Tr}(\hat{\Sigma} \cdot X)$$

$$\operatorname{Tr}(X) = k \quad (\text{relaxed } k\text{-sparsity})$$

$$X \succeq 0$$

$$-1 \leq X_{ij} \leq 1, \forall i, j \in \{1, 2, \ldots, p\}$$

Note that the polynomial program PP1 above, even though it may not be efficiently computable, it is still a statistical quantity that we can compute given enough time. This is the analog of the maximum clique statistic we used in our first example. So a natural question is to understand how many samples does it take for PP1 to actually distinguish between the two cases (you can think of λ as a constant e.g. $\lambda \approx 0.1$). The answer is that the number of samples $n > \frac{k \log p}{\lambda^2}$.

What about the efficient distinguisher using the SDP? The obvious answer could be $n = p \log p$ because with that many samples (p might be really large compared to the sparsity k), I can always estimate the whole covariance matrix, but can we get away with much less samples? It turns out that we can use the SDP with $n \geq \frac{k^2 \log p}{\lambda^2}$.

Two things to notice in this Sparse PCA example is that the information theoretic bound for the number of samples required to distinguish between the two cases is $n > \frac{k \log p}{\lambda^2}$ whereas the computational threshold is only slightly larger with $n \geq \frac{k^2 \log p}{\lambda^2}$ samples and it is achieved using a semidefinite program, rather than a spectral technique. This means that in this example the best algorithm is an SDP.

Exercise 8.1: Provide an argument for why the information theoretic threshold should grow as $n \geq \Omega(\frac{k \log p}{\lambda^2})$ (where you can think of λ as a small constant and ignore it) and why does it have to grow with the $\log p$ of the dimension?

Question: For what values of k is the above $n \geq \frac{k^2 \log p}{\lambda^2}$ threshold meaningful and why can't we just compute the largest eigenvector of the covariance matrix?
Answer: When k is large ($k \geq \sqrt{n}$), it is true that we can just compute the top eigenvector and the corresponding eigenvalue of the sampled covariance matrix, because we are only interested in the distinguishing problem and the eigenvalue will be large. However, for general k, computing the eigenvector itself will not be useful as it may not be sparse as we want it. This is why we need to use the PP1 and relax it to the

[5] Note this is not an SDP and we don't claim that we can solve it efficiently.

SDP when we explicitly state the sparsity constraint. Here, as it was expected, the algorithm becomes bogus when $k \geq \sqrt{n}$, in which case we don't have to use it and instead we should compute the largest eigenvalue and eigenvector. You can think of our algorithm here as useful for regimes where $k \approx n^{0.1}$ for example.

Question: What is the behaviour of the spectral algorithm we saw before for Planted Clique and of the SDP for Sparse PCA when we allow semirandom perturbations to the input?
Answer: You can think as semirandom perturbations as perturbations of the input that can only make the structure/solution we are seeking even more pronounced. For example, in the Planted Clique problem, a semirandom perturbation would be to delete some of the edges **outside** the planted clique. Even though this seems helpful for an algorithm that tries to find the hidden structure, as it makes the clique more pronounced in the graph, usually spectral algorithms are fooled by such perturbations. Intuitively, this is because there is some chance that by deleting edges, another direction in the spectrum of the matrix pops up as the dominant instead of the eigenvalue corresponding to the clique. Here, however, since we are only interested in the distinguishing problem we could still succeed as the $||A||_\infty$ would be large compared to the random case. On the other hand, SDPs usually have some intrinsic robustness properties against such semirandom perturbations and we expect them not to be fooled.

3 Planted CSP

3.1 Random CSPs vs Planted CSPs

The last example for today will be the Planted Constraint Satisfaction Problem. Again we will be asked to distinguish between two cases. The first distribution is the analog of $G(n, \frac{1}{2})$ from our Planted Clique example and the $\mathcal{N}(0, \mathcal{I})$ from the Sparse PCA. We begin by defining a random instance of the classic 3SAT problem, which we will denote by $\mathcal{H}(n, 3, \Delta)$ (this is our first of the two distributions).

Here is how you generate a sample from this distribution: first, you choose $m = \Delta n$ random triples on $[n]$ (let's say we have a set of n variables) and these triples will correspond to constraint clauses as follows: for each triple $t := \{i, j, k\}$, choose b_t^i, b_t^j, b_t^k uniformly in $\{0, 1\}$. So then, the constraint corresponding to the triple $t = \{i, j, k\}$ is $b_t^i x_i \vee b_t^j x_j \vee b_t^k x_k$, where $b_t^i x_i = x_i$ if $b_t^i = 0$ and $b_t^i x_i = \overline{x_i}$ if $b_t^i = 1$. In other words, the bit b_t^i tells us whether to negate the variable or not.

In the previous examples, the planted version of the problem was pretty clear, however in the Planted CSPs case, if we are not careful with our planted instance, we may end up trivialising the problem and making it uninteresting. That's why the planted version will seem a bit complicated at first.

Naturally, we would like to create a distribution of instances that look random as before, where the number of constraints satisfied by *any* assignment of the n variables is really small but with a "protected" solution, i.e. an assignment that satisfies the formula. This is called "planting an assignment" in the random CSP. How can we plant a solution x^*? So the second distribution will be denoted as $\mathcal{H}(n, 3, \Delta) + x^*$ and is as follows:

First, we choose $m = \Delta n$ random triples $t = \{i, j, k\}$ on $[n]$ as before. Then, as a second step, we choose the *planted assignment* x^* uniformly at random from $\{0, 1\}^n$. The question now is **how can we make this particular x^* assignment satisfy all the constraints**? This will be the third step: Choose b_t^i, b_t^j, b_t^k

uniformly in $\{0,1\}$ **but subject to the constraint** that:[6]

$$b_t^i \oplus b_t^j \oplus b_t^k = x_i^* \oplus x_j^* \oplus x_k^* \oplus 1$$

Now it is easy to see that x^* is a satisfying assignment for all the constraints and is left as an exercise.

Exercise 8.2: Show that the way x^* is defined, it will be a satisfying assignment.

The strength of the signal here is the number $\Delta = \frac{m}{n}$ which determines how many clauses will the CSP formula have. The question we are interested in is how small can Δ be and still be able to distinguish between the two distributions above ($\mathcal{H}(n,3,\Delta)$ and $\mathcal{H}(n,3,\Delta) + x^*$).

The natural distinguisher we would want to use here is the statistical quantity MAX-SAT := the maximum possible number of clauses satisfied by any boolean assignment. This is not an efficient distinguisher and it is not hard to show using Chernoff that when $m \geq 10n$ and the CSP is drawn from $\mathcal{H}(n,3,\Delta)$ the value of MAX-SAT$\leq \frac{7}{8} + \epsilon$. However, for $\mathcal{H}(n,3,\Delta) + x^*$ the value will be 1 and we will be able to distinguish. If we want computational efficiency then we need $m \geq n^{3/2}$ and as with the clique problem we have a huge gap. The best algorithm used to achieve it is a spectral algorithm acting on a natural matrix associated with the constraints of the CSP.

Summing up, the statistical threshold for this problem is: $m \geq 10n$ and the computational threshold is: $m \geq n^{3/2}$. The 2 basic questions we want to answer is "are these gaps inherent?" and "how do we predict the true computational threshold for these and other problems?"

The goal of this class is to use Sum-of-Squares (SOS) to answer both of these questions [BQ12, FPV15] and an *informal* hope/statement/claim one might have regarding SOS is that it is the best polynomial time algorithm we can use to solve these problems, in the sense that the threshold we get for SOS is likely to be the threshold for all polynomial time algorithms.

The phenomenon we observe is that the best algorithms for all the above problems is either spectral algorithms or SDPs and they are all captured by the SOS framework. In addition to its generality, the SOS framework is amenable to analysis and we can figure out tight lower bounds for it.

3.2 Algorithm for planted CSPs

We will define a relevant problem that we call "Refutation Question" and we will try to design an algorithm to do the following:
****Refutation Question****:
Input: A CSP (3SAT) instance $\mathcal{I}$.
Goal: Output either Unsatisfiable or "Don't know".
The guarantee we want for the algorithm is the following:

- If output is "unsat", then the instance $\mathcal{I}$ is unsatisfiable.
- $\Pr(\text{algorithm outputs "unsat"}) \geq 1 - o(1)$ for $\mathcal{I}$ drawn from $\mathcal{H}(n,3,\Delta)$.

In other words, the algorithm needs to be correct whenever it claims that the instance is unsatisfiable, i.e. if the algorithm says "unsat" then the instance $\mathcal{I}$ better be unsatisfiable. So far, if we only ask for the first

[6] $\oplus$ means XOR as usual

bullet above, then the algorithm can always answer "Don't Know" and it would always be correct. That is the point of the second bullet above, that demands that the algorithm should output "unsat" w.h.p. whenever the instance is drawn according to $\mathcal{H}(n, 3, \Delta)$.[7]

So for what m is efficient refutation possible? Intuitively, refutation should be easier as the number of clauses grows and the answer for efficient refutation will be $m = n^{3/2}$ as we mentioned above.

3.3 Motivation and Brief History for refuting CSPs

Feige, 2002 [Fei02]: Feige's Hypothesis: He conjectured that for $m < o(n^{3/2})$, there does not exist a polynomial time refutation algorithm. The original motivation was to make an average-case hardness assumption from which you can derive hardness for other problems. So, believe it or not the original motivation was to prove worst-case hardness. In recent years, people have used this hypothesis to prove new average-case lower bounds.

Daniely, Linial, Shalev-Shwartz, 2014 [DLSS14]: In machine learning community, these authors had a breakthrough result regarding proving hardness results starting from a hypothesis similar to Feige's. Usually, a lot of machine learning problems come with a natural distribution attached to them and it is difficult to prove average-case hardness results for them and there were important open problems for example, the hardness of learning DNF (disjunctive normal form) formulas. Using their hypothesis, they derived hardness for DNFs.

Allen, O'Donnell, Witmer, 2015 [AOW15]: Here comes the interesting part of the story:[8] these authors falsified the previous hypothesis by giving an algorithm that could refute a random CSP of some specific form.

3.4 Refuting 4-XOR CSPs

Refutation Problem for 4-XOR: This will be just a slight modification of the 3-SAT version we saw before, but now the predicates are 4-XORs. This just makes the presentation a bit easier, and one can get results for other CSPs using this framework.

So how do we generate such a random 4-XOR formula? First , we choose $m = \Delta n$ random 4-tuples on $[n]$. For each 4-tuple $t = \{i, j, k, l\}$, choose b_t uniformly in $\{0, 1\}$. Then, each constraint is $x_i \oplus x_j \oplus x_k \oplus x_l = b_t$. If m, the number of constraints, is more than $10n$, i.e. $\Delta > 10$, with high probability, the fraction of satisfiable constraints is $\leq \frac{1}{2} + 0.01$.

We present a simple and efficient spectral algorithm for refutation when $m \geq \Omega(n^2)$:

- Build the pairwise adjacency matrix:[9]

$$A_{(i,j),(k,l)} = \begin{cases} 2b_{i,j,k,l} - 1 & \text{if } (i, j, k, l) \text{ was sampled} \\ 0 & \text{otherwise} \end{cases}$$

[7] Note that w.h.p. an instance generated by $\mathcal{H}(n, 3, \Delta)$ is going to be unsatisfiable, so there is trivial inefficient brute-force algorithm that can solve the "Refutation Question".

[8] Everything we discussed so far is about polynomial time algorithms, but there are other papers as well that are interested in subexponential algorithms for refuting CSPs. Papers in this line of work are due to Raghavendra, Rao, Schramm [RRS16].

[9] Tuples in the matrix indices are not ordered and the matrix is symmetric.

- return "unsat" if $||A||_\infty \leq O(1)$.

We want the algorithm to have the same guarantees as for the ****Refutation Question**** from above. So why does the algorithm work? Note the following:

- Note that the $||A||_\infty$ is an upper bound for the following maximization problem

$$\max_{x \in \{\pm 1\}^n} \sum_{i,j,k,l} A_{i,j,k,l} \cdot x_i x_j x_k x_l$$

  and that if the formula is satisfiable then the value of this maximization problem is in the order of $\Omega(m)$. To see this, just take a y vector with n^2 coordinates $\{\pm 1\}$ such that $y_{ij} = x_i x_j$ (the satisfying assignment) and rewrite the objective, noting that we get a quadratic form. Normalizing by the norm of the vector y which is $||y|| = \sqrt{n^2} = n$ we conclude that in the case of a satisfiable formula we have $||A||_\infty \geq \frac{m}{n^2}$ which is large compared to a random instance where we get $||A||_\infty$ is at most some constant.

- We need to certify that the algorithm will not say "unsat" on a satisfiable instance. But if the algorithm outputs "unsat", this happened because the $||A||_\infty$ was small (at most some constant). However, since we had $m \geq \Omega(n^2)$ clauses, the instance cannot have been satisfiable because then the satisfying assignment would be a witness of a large $||A||_\infty$, hence a contradiction.

- The last thing we need to argue is how large can the spectral norm of a random matrix be. Note that for a random $N \times N$ matrix with $\{\pm 1, 0\}$ entries, where every entry is non-zero with probability p, we have $||A||_\infty \leq \sqrt{Np} \cdot \text{polylog}(N)$ w.h.p.

- Here $N = n^2$ and we have an $n^2 \times n^2$ matrix, and p is about[10] $\frac{m}{n^4}$.

3.5 Extensions to arbitrary CSPs

Now we will see how to extend the above algorithm to refuting other CSPs by essentially a reduction to refuting XORs. The main idea is really simple:

- Let $\mathcal{P}$ be an arbitrary 4-ary $\{0,1\}$ predicate (where each clause has four $\{0,1\}$ literals) and let $X_S(x) = \Pi_{i \in S}(2x_i - 1)$ be the $\{\pm 1\}$ version of the XOR of the bits in set S given an assignment x. The predicate $\mathcal{P}$ can be uniquely rewritten as a linear combination of XORs (using its Fourier transform): $\mathcal{P}(x) = \sum_{S \subseteq [4]} \hat{P}(S) \cdot X_S(x)$, where $\hat{P}(S)$ is the corresponding Fourier coefficient.[11]

- Previously, when we were trying to refute the XOR formula, our general strategy was to upper bound the quantity $\max_{x \in \{\pm 1\}^n} \sum_{i,j,k,l} A_{i,j,k,l} \cdot x_i x_j x_k x_l \leq ||A||_\infty \cdot n^2$. We sort of want to copy this argument.

[10] Note that each 4-tuple is present in the formula with probability $\frac{m}{\binom{n}{4}}$.

[11] The Fourier coefficients are just some numbers — no reason to be scared of them. Specifically, $\hat{P}(S) = \mathbb{E}[P(X)\Pi_{j \in S} X_j]$, where X is uniform on $\{\pm 1\}^n$.

What we had before was an XOR formula whereas now we have a linear combination of XOR formulas for the general predicate $\mathcal{P}$. What we will try to do, is to look at the following quantity instead: $\max_{x\in\{\pm 1\}^n} \sum_{S\subseteq[4]} A_S \cdot \hat{P}(S) \cdot X_S(x)$, where A_S is a natural matrix that occurs in this context.[12]

- In the above sum there are 16 possible sets and for each possible set we get an XOR instance (4-XOR specifically) and we can use as a black box the algorithm we described previously for refuting 4-XOR instances. If we can refute all of them, meaning that we get a small spectral norm for all 16 cases (of course a small constant remains small when multiplied by 16), then we can refute the whole predicate $\mathcal{P}$.

3.6 Beyond 4-CSPs

What we did for the arbitrary 4-ary predicates can easily be generalized for larger arities. Before, we basically start from the original predicate $\mathcal{P}$, we express it as a sum of at most 16 terms each of which is an XOR instance and we can apply the algorithm from previously to each of them. In particular, for every possibly 4-CSP if the number of constraints is large ($m = \Omega(n^2)$) then the refutation problem is solvable using just this simple spectral algorithm from above. If we had an even arity CSP then we can handle it really easily by using the above reduction to XORs and for $2d$-arity CSPs we would need m to be at least $\Omega(n^d)$.

For odd arities, the problem is a little bit more complicated. The reason is that we don't know if refuting the XOR version of a CSP is the best way to refute it. Maybe there is a better way to do it. To see an example on why odd predicates introduce some challenges where the above XOR trick is not optimal is the following:

- Take $\mathcal{P} : \{0,1\}^4 \to \{0,1\}$ where $P(x) = x_1 \oplus x_2 \oplus x_3$. The trick we used above will guide us into a requirement of the form $m = \Omega(n^2)$ by the 4-dependence of the predicate even though the formula actually depends only on 3 variables and the spectral threshold should be $\Omega(n^3/2)$ instead. We deduce that sometimes the above algorithm seems wasteful.

As a final comment, there is way to define a very simple parameter which is a function of the predicate and that would correspond to the "true" degree of the predicate, rather than the "syntactic" degree that may be misleading as we saw before. This however goes beyond the scope of this lecture.

Regarding references settling the complexity for refuting CSPs, for upper bounds you can take a look at O'Donnell, Allen, Witmer [AOW15] and Raghavendra, Rao, Schramm [RRS16] whereas for matching SOS lower bounds at Kothari, Mori, O'Donnell, Witmer [KMOW17].

4 SOS Lower Bounds for Random CSPs

We will now see a classical lower bound due to Grigoriev from 2001 (see Boaz Barak's notes [Bar14]) for the k-XOR problem, for any $k \geq 3$. We will focus in the 3-XOR case.

Theorem: Let $\mathcal{I} = \mathcal{I}(x)$ be a random 3XOR instance with $m = n^{3/2-\epsilon}$ constraints. Then:

[12] It is not important to exactly define it right now, but it is a kind of parity matrix that is naturally associated with the 4-tuples that were sampled.

- w.h.p. $val(\mathcal{I}) \leq 0.51$, yet however
- $\exists$ a solution $\tilde{\mathbb{E}}$ to a degree $n^{\Omega(\epsilon)}$ SOS SDP that proves that the SDP value of $\mathcal{I}$ is 1. (i.e. 100% satisfiable)

Let's parse the statement of the theorem. The first claim is really simple, just follows from a simple Chernoff bound and it tells us that the fraction of constraints satisfied by any assignment cannot be much larger that 0.5. The second claim is more interesting and tells us something about SOS: despite $val(\mathcal{I}) \leq 0.51$, there is some SOS solution of some $d = n^{\Omega(\epsilon)}$ rounds, meaning an algorithm running in $\tilde{O}(2^{n^{\epsilon}})$ time such that it can satisfy all of the constraints. In other words, SOS believes the instance to be completely satisfiable even though the instance is very far from being satisfiable.

For the pseudoexpectation $\tilde{\mathbb{E}}$: It is just a linear map from $\mathbb{P}_d$ (the space of all degree $d = n^{\Omega(\epsilon)}$ polynomials) and in our case we want that it satisfies the following properties (only the first 2 are general, while the third is required to prove the theorem):[13]

1. $\tilde{\mathbb{E}}(q \cdot x_i^2) = \tilde{\mathbb{E}}(q \cdot 1), \forall i, \forall$ degree $d-2$ polynomials q.
2. $\tilde{\mathbb{E}}(q^2) \geq 0, \forall$ degree $d/2$ polynomials q.
3. Here we also want, $\tilde{\mathbb{E}}(\mathcal{I}(x)) = 1$. We will actually be able to prove something stronger in our case: $\forall i, j, k$ triples in $\mathcal{I}(x)$:
$$\tilde{\mathbb{E}}(q \cdot x_i x_j x_k) = \tilde{\mathbb{E}}(q \cdot b_{ijk})$$

So we will now try to find such a pseudoexpectation satisfying all of the above properties and we would have proved the lower bound (and the theorem) since this pseudoexpectation would give value 1 to the instance $\mathcal{I}$ despite the fact that it is far from being satisfiable.

How can we define the pseudoexpectation? To define it, since it is a linear operator, it is enough to give its values on any basis of degree at most d polynomials, because by linearity the pseudoexpectation of every possible polynomial will then be fixed.

In the way we define it, it better be true that:

- $\tilde{\mathbb{E}}(x_i x_j x_k) = b_{ijk}, \forall i, j, k$ triples that are present in the formula $\mathcal{I}$.
- If we had $x_1 x_2 x_3 = b_{123}$ and $x_2 x_4 x_5 = b_{245}$, by XOR-ing them together we would get $x_1 x_4 x_3 x_5 = b_{123} b_{245}$. So we can derive a new constraint and we would like the SDP to respect the new constraint if it respected the initial two constraints. In some sense, we want to give some values in a small number of monomials like $x_1 x_2 x_3$ etc. in such away so that we respect the "local" facts, like the above derivation. Note, however, that since the instance is unsatisfiable, if we were to look at a large number of monomials and respect all of the possible derivations, then we would end up in a contradiction of the form $x_1 x_2 x_3 = +1$ and $x_1 x_2 x_3 = -1$. Somehow the hope is, that looking at small sets of constraints wouldn't yield any contradictions and hence we may be able to fool the SDP by not looking at large unions of constraints; this will indeed give us the SOS lower bound we want.

[13] The first constraint makes use that we are on the $\{\pm 1\}$ hypercube since x's are such that $x_i^2 = 1$ and in the third constraint we make use of $x_i x_j x_k = b_{ijk}, \forall i, j, k \in \mathcal{I}$.

- Let's implement this "local" facts constraint more concretely as follows: For every equation $X_T = b_T$ (for some set T of equations) that can be derived by at most $s \leq d$ equations in $\mathcal{I}$, we set $\tilde{\mathbb{E}}(X_T) = b_T$. For every monomial Q that cannot be derived from at most s equations, set $\tilde{\mathbb{E}}(X_Q) = 0$ (the SDP thinks that there is no implied constraint for that monomial Q).

We need to explain what is the parameter s and why the above process will yield a well defined pseudo-expectation. The parameter s will just be the degree of the SOS hierarchy that we are shooting for, that's why we had $s \leq d$. The remaining important task is to show that the above process will not yield any contradiction (even though $\mathcal{I}$ is unsatisfiable) and hence that the SDP is actually fooled. Notice that so far we haven't exploited the fact that we are given a random 3-XOR instance. Now we will make use of that and in particular of the so-called *expansion property* of random 3-CSPs.

Expansion Property: For every collection of $s \leq d(\approx n^{\epsilon})$ equations in $\mathcal{I}$ there are $\geq 1.51s$ distinct variables.

Exercise 8.3: Prove the expansion property above for random 3-CSPs.

Claim: The pseudoexpectation $\tilde{\mathbb{E}}$ is well-defined.

I am worried about the following bad event: For the same monomial, the SDP derived two different values, i.e. $\exists$ two different sets of equations T_1, T_2 with $|T_1| \leq s, |T_2| \leq s$ such that:

$$\oplus_{i \in T_1} E_i = \oplus_{i \in T_2} E_i \implies \oplus_{T_1 \cup T_2} E_i = 0 \implies \emptyset$$

where E_i is the set of variables appearing in the i^{th} equation. To understand why we get a contradiction, note that the LHS will always be 0 since it is the XOR of two quantities with the same values, whereas the RHS will be a product of some b's and since the b's were chosen at random, with probability 1/2, the RHS will be 1, hence getting a contradiction. Using the expansion property, we have w.h.p. that this bad event ($\oplus_{i \in T_1} E_i = \oplus_{i \in T_2} E_i$) will not happen.

Now we need to show the second property that $\tilde{\mathbb{E}}(q^2) \geq 0, \forall$ degree $d/2$ polynomials q. As a comment for most SOS proofs, almost always all properties are trivial to show, except for this non-negativity constraint. The main idea is the following:

- Any such polynomial q can be written (this is just the usual monomial expansion) as $q = \sum_{|S| \leq d/2} q_S X_S$ and when we square, we get $q^2 = \sum_{|S|,|T| \leq d/2} q_S q_T X_S X_T$.
- For some of the monomials, the SDP has a value that was derived using the derivations on small sets of equations, but some others were set to 0, because we couldn't derive any value by the above process. What we do is to define an equivalence class for these monomials using the pseudoexpectation.
- Define $S \sim T$ if $\tilde{\mathbb{E}}(X_S X_T)$ is non-zero. This forms an equivalence class.
- Now we are almost done: Set $q_i = \sum_{S \in \mathcal{C}_i} \hat{q}_S X_S$ and this means that summing over all the equivalence classes: $q = \sum_{\mathcal{C}_i} q_i$.

- Finally, let's do the crucial non-negativity computation:

$$\tilde{\mathbb{E}}(q^2) = \tilde{\mathbb{E}}\left(\sum_{\mathcal{C}_i} q_i\right)^2 = \sum_i \tilde{\mathbb{E}}(q_i^2) + \sum_{i \neq j} \tilde{\mathbb{E}}(q_i q_j) = \sum_i \tilde{\mathbb{E}}(q_i^2)$$

Exercise 8.4: Prove that the last equality above is true.

- Finally, we have $\sum_i \tilde{\mathbb{E}}(q_i^2) \geq 0$ because $\tilde{\mathbb{E}}(q_i^2) = \sum_{S \in \mathcal{C}_i, T \in \mathcal{C}_i} \tilde{\mathbb{E}}(X_S X_T \hat{q}_S \hat{q}_T) =$

$$= \left(\sum_{S \in \mathcal{C}_i} \tilde{\mathbb{E}}(X_S) \cdot \hat{q}_S\right)^2 \geq 0.$$

Question: How can we think of this $n^{3/2-\epsilon}$ threshold?
Answer: We need to find where this proof breaks down when $m \geq \Omega(n^{3/2})$, since then there is an algorithm which means that SOS is not fooled, when we have more constraints m. There is only one place that breaks down and this is the expansion property. Then, the Chernoff + Union bound break down at this threshold.

Open Question: Is there a derandomized version of this lower bound? Is there a deterministic instance that we can use to prove hardness? This has to do with deterministic constructions of expanders (note that, in some sense, all we needed from the random instance was to satisfy the expansion property) and a relative reference for deterministic constructions of expanders can be found here.

References

[AOW15] Sarah R Allen, Ryan ODonnell, and David Witmer. How to refute a random csp. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 689–708. IEEE, 2015.

[Bar14] Boaz Barak. Sum of squares upper bounds, lower bounds, and open questions. 2014.

[BQ12] Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. *Computational Complexity*, 21(1):83–127, 2012.

[BR13] Quentin Berthet and Philippe Rigollet. Computational lower bounds for sparse pca. *arXiv preprint arXiv:1304.0828*, 2013.

[DLSS14] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 441–448. ACM, 2014.

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543. ACM, 2002.

[FPV15] Vitaly Feldman, Will Perkins, and Santosh Vempala. On the complexity of random satisfiability problems with planted solutions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 77–86. ACM, 2015.

[KMOW17] Pravesh K Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any csp. *arXiv preprint arXiv:1701.04521*, 2017.

[MW15] Tengyu Ma and Avi Wigderson. Sum-of-squares lower bounds for sparse pca. In *Advances in Neural Information Processing Systems*, pages 1612–1620, 2015.

[RRS16] Prasad Raghavendra, Satish Rao, and Tselil Schramm. Strongly refuting random csps below the spectral threshold. *arXiv preprint arXiv:1605.00058*, 2016.