PACM IEeAS Seminar

The complexity of detecting planted solutions November 16, 2015

Lecturer: Santosh Vempala Scribe: Kiran Vodrahalli

Contents

1 Overview

We get some lower bounds on planted clique and planted k-SAT (which are similar reductions of the NP-hard problems) on a specific class of algorithms, statistical algorithms: for instance, PCA, MCMC, gradient descent, large or small degree, linear and conic programming.

2 Introduction

Santosh Vempala did his Ph.D. at CMU and joined the faculty at MIT. Now he is at Georgia Tech. This talk is based on two articles. Here are the two main problems: planted clique, planted k-SAT.

2.1 Random Planted Clique

A clique in a random graph. Finding whether a graph has a clique of size k is an NP-hard problem; we don't really understand it. How about an easier problem - what if the input is random? We want to solve the clique problem on average. We pick a graph by placing edges with probability $1/2$, and then on average slove. The largest clique is of size $\sim 2\log n$. A greedy algorithm finds one of size $\log n$ - you want to succeed with probabiltiy 1 as $n \to \infty$ (just pick maximal clique). An open problem is: Can you find a clique of size $1.01\log n$ in polynomial time (Frieze).

Let's make the problem easier: Planted clique (Jerrum 92). Start witha random graph, add a clique of size $k >> 2\log n$ on subset of k vertices. You want to find the planted clique. Or even detection: Can you tell which of two inputs (one w/planted clique, one with not) in polynomial time?

One thing we can observe is that we can solve this problem $n^{\mathcal{O}(\log n)}$ with the trivial enumeration, for $k > 2\log n$. This makes it interesting from complexity point of view. What is the true complexity

of the problem? If you plant a clique of size sufficiently large, $k \geq c\sqrt{n \log n}$, then something simple works with high probability: Look at degrees, and pick the highest degree vertices. You can improve this to $k = \Omega(\sqrt{n})$: Here build an adjacency matrix and compute its top eigenvector, and do some refinement. Our main question is **Is $k = \Omega(\sqrt{n})$ a computational barrier?** (the gap between $2 \log n$ and $\sqrt{n}$ - is there some algorithm that achieves a bound here which is tight?)

2.2 Planted k-SAT

This problem is planted k-SAT. We have a Boolean formula where each clause is exactly k literals and there are m clauses. There are n literals (variables). We would like to determine whether a given Boolean formula has a satisfying assignment. To simplify, let us consider the case where the clauses are drawn at random. We ask, can you find a satisfying assignment if the number of clauses m increases? At some point, there is a transition from satisfiable to unsatisfiable. The threshold was understood for a long time, this transition happens at around a linear number of formulas. The open problem is to find a satisfying assignment or a refutation in polytime. Again, we have no idea how to do this as in Problem 1, so let's go to the planted version of this problem. Fix some assignment on the variables $\sigma : X \to \{-1, 1\}$, and then there is a distribution on clauses Q. The probability of a clause depends on the varialbes in the clause, which have some values according to the planted assignment. This determines the probability of a clause.

$$\mathbf{P}\{C\} = \frac{Q(\sigma(C))}{\sum_C Q(\sigma(C'))}$$

In this situation, we would like to detect whether or not there is a planted assignment, or whether we are drawing completely at random. To rephrase, you get C, and $\mathbf{P}\{\sigma(C)\}$, and find the hidden assignment, or even jsut distinguish. Information theoretically, with high probability the planted solution can be recovered from $m = \mathcal{O}(n \log n)$ clauses (it becomes unique). The best known polytime algorithm takes $m = \Omega\left(n^{k/2} \log n\right)$ clauses. So this gives you a gap between n and $n^{k/2}$. So where is the truth?

There are many reasonable algorithmic ideas: spectral methods, local search, convex optimization via relaxation, MCMC, EM/Simulated annealing.

Here is one successful technique: Clique via principal component:

$$\max_{u:\|u\|=1} \|Au\|^2 = \max_u \sum_i (A_i \cdot u)^2$$

A theorem by Furedi gives that $\|R\|_2 \leq (2 + o(1))\sqrt{n}$, where $A = B + R$, where B has 1 and 0, and R takes on values $1, -1$. This algorithm works for $k \geq c\sqrt{n}$. There are lower bounds for SDPs that show you can't do better than $k \geq c\sqrt{n}$. To get arbitrarily small c, you have to do enumerative search on $n^{1/c}$ subgraphs, and one of them will work. There is an even simpler algorithm for fixed $\sqrt{n}$. Find the minimum degree vertex, remove, repeat: At some point you will stop and then grow the algorithm to a maximum clique.

The principal component is the second moment, you're maximizing the second moment of the matrix. Why not try higher moments? How about the r^{th} moment, which you could maximize as

$$\max_{u:\|u\|=1} \|Au\|^r = \max_u \sum_i (A_i \cdot u)^r$$

Frieze and Kannan showed you could do this for $r = 3$ you'd get $n^{1/3}$, we generalize to $n^{O(1/r)}$. However, maximizing higher moments is hard! However, you can find local optima in other cases, for instance ICA. But not in planted clique.

2.3 Bipartite Random Planted Clique

We want to turn this problem into a problem where the input itself is a distribution, which will turn out to be quite useful. We have a distribution on $1, -1$ vectors, fix a subset S with $|S| = k$. A random vector x is generated for $i \notin S$, $x_i \sim Ber(1/2)$. Otherwise, with probability k/n for $i \in S$, $x_i = 1$, otherwise, with probability $1 - k/n$, $x_i \sim Ber(1/2)$. The problem is to find the planted subset S. You want the bipartite since we can give a proof on this one. For this version, the eigenvector techniques work. It looks like the same problem, but now we have a chance to prove some lower bounds. Each row x_i corresponds to a point on the other side of the bipartite graph. There are two types of points, "red" and "black".

Input drawn from a distribution is a common framework in machine learning (labeled, unlabeled, etc.). Formally, D is a family of distributions over domain X. F is a set of solutions to the problem (here, which subset is planted). Then $Z : D \to 2^r$ gives you a mapping to valid solutions for each input distribution. The problem is find a valid solution given access to random samples from the input distribution. Many examples: For instance, compute average, principal component, convex programming, etc.

3 Planted k-SAT

This is very naturally a problem over input distributions. You have a distribution Q over k-tuples of variables based on a fixed unknown σ. Planted uniform k-SAT: you are not allowed to pick everyone on the false side, each clause must have at least one true literal. Another variant is noisy k-XOR SAT. You could decide in your clause distribution: Number of clauses with number of settings of true variables even could have probability p uniformly, and q for odd. For $k = 2$, this is like the Stochastic Block Model. Formally, it is the Sensor-Block Model, because of the noise. For this problem there are some nice upper bounds for special cases. Flaxman $'03$ gave an algorithm that looked at $\mathcal{O}(n \log n)$ literals. Cooper-Frieze $'10$ gave $\tilde{\mathcal{O}}(n^{3/2})$ for all planted 3-SAT distributions. More recently there is an $\tilde{\mathcal{O}}(n^{r/2})$ statistical algorithm via reduction to bipartite stochastic block model. It means it is not $r - 1$ independent, but it is r independent. So it is effectively a rank function; $1 \leq r \leq k$. You can reduce planted k-SAT to a noisy r-XOR-SAT, form a matrix of size $n^{r/2} \times n^{r/2}$, find th etop singular vector and recover the planted assignment (you get enough equations to solve for). This is ok for even r, what do you do in the odd case? SDP works, but you can also reduce to a bipartite planted stochastic block model with an unequal number of vertices on each side (for instance, $\lceil r \rceil$ and $\lfloor r \rfloor$). You can do power iteration for the singular vectors. You want to recover the partition between A, B (partitioned). This means $x = \frac{Ay}{\|Ay\|_2}$, $y = \frac{A^T x}{\|A^T x\|}$. Recently it has been proved that you need $\Omega(n_1^{1/3} n_2^{2/3})$, where n_2 is the bigger side between A and B (the partitions of the vertices, $|A| = n_1, |B| = n_2$). But the following works is more efficient: use only a random sample of entries from A, and stop after $\mathcal{O}(\log n)$ iterations, this will give you a pair correlated. Read the paper Florescu-Perkins 2015.

4 How to Prove Lower Bounds?

We would like to convince you that the current bounds are the best for a reason. The smallest integer r is such that there is a nonempty $S \subseteq \{1, 2, \cdots, k\}$ such that there is a nonzero discrete Fourier coefficient $\hat{Q}(S) = 0$ (or the independent stops at r (not independent for $r - 1$)). Equivalently the distribution on clauses is $(r - 1)$-wise independent, but not r-wise independent. Is $m = \Omega(n^{r/2})$ a computational barrier?

Consider this framework due to Kearns '93. We don't really know how to do lower bounds on algorithms, so we need to restrict algorithms somehow. This is a mild restriction: First of all, only for algorithms with input as a distribution. You typically get samples from distribution and do something with it. Here this is not allowed. You supply a function to the algorithm and get either expectation on a random sample up to some tolerance τ, or the values themselves. In equations, for any $f : X \to [0, 1]$, STAT(τ) outputs $\mathbf{E}_D[f(x)] \pm \tau$. τ is chosen by algorithm. The complexity of algorithm is evaluated as the number of calls to this oracle. A valid objection to this original model is that τ is expected. Here we also define $VSTAT(t)$ outputs $\mathbf{E}_D[f(x)]$ to within standard deviation of t independent random samples. Then 1-STAT outputs $f : X \to \{0, 1\}$ on one random x (you just get the expectation of a given input). The other cases give you adversarial settings. An even stronger oracle allows MSTAT(L) with $f : X \to \{0, 1, \cdots, L-1\}$ (no longer binary).

Note that all of our previous algorithms can be run in this framework! PCA, MCMC, gradient descent, large or small degree, linear and conic programming and so on can be done statistically. The only algorithm which we don't have a statistical counterpart is Gaussian elimination (2 over a field). The goal is to show that every statistical algorithm must be slow, so that it cannot have polynomial complexity. This of course does not rule out all algorithms, just all known approaches.

How do we prove this? Well there are lots of very different instances. The idea is you have lots of very different instances: One distribution for each possible planted clique subset; these distributions are different. How do we capture this? Well, we talk about the correlation of distributions. The base distribution is U, which is typically uniform. Then $f, g : X \to R$, so $\langle f, g\rangle_U = \mathbf{E}_U[f(x)g(x)]$. Correlation is defined as $\rho(D_1, D_2) = \langle (D_1/U - 1, D_2/U - 1\rangle$.

The definition of a statistical dimension: We want to have a bunch of distributions which are NOT solutions. We have to at least eliminate these such that $\rho(D_i, D_j) \leq \beta$ if $i = j$, γ if $i \neq j$. Then $SD(Z, \gamma, \beta) = d$, where $\forall f \in F$, $\exists D_f = \{D_1, D_2, \cdots, D_d\} \subseteq D \setminus Z^{-1}(f)$. You specify γ and β, you get a d. A theorem gives that any statistical algorithm needs $d \cdot \frac{\tau^2 - \gamma}{\beta - \gamma}$ queries to STAT(τ). Why is this useful? Suppose we are looking for a parity function of n variables, U uniform over Boolean variables, D_f uniform over vectors with same parity. For each f there is only one distribution. $\rho(D_f, D_g) = 1$ if $f = g$, 0 otherwise. Then SD(parity, 0, 1) $= 2^n - 1$. Finding a parity needs exponential queries for any statistical algorithm! Note that noise parity is not solvable by Gaussian Elimination (it is not a natural choice; there may be some super sophisticated approach which is not statistical).

What about planted clique? There is one distribution per k-subset. We have that $\rho(D_2, D_T) \leq \frac{2^\lambda k^2}{n^2}$, where $S, T : \lambda = |S \cap T|$. Most pairs of distributions are far, but not tall.

We extend the notion of statistical dimension: instead of every pair must have low correlation, we saw every large subset must have low average correlation. Here we get you need d queries to STAT($\sqrt{\bar{\gamma}}$), or VSTAT($1/(3\bar{\gamma})$). For planted bi-clique, then for $l \leq k$,

$$SD(Z, \frac{2^{l+1}k^2}{n^2}) \geq (1/2)n^{2l\delta}$$

. With $l = c \log n$, for any $\delta > 0, k < n^{0.5-\delta}$, any statistical algorithm for the planted bi-clique problem needs $n^{\Omega(\log n)}$ queries to VSTAT.

We give another theorem: Any statistical algorithm that detects a planting must use $\tilde{\Omega}(n^{r/2})$ calls to MSTAT($n^{r/2}$). We now go to another notion of statistical dimension. We have to look at the spectral norm of the set of distributions. Think of h as any query. Look at the discrimination norm: The query that maximizes the very best query which distinguishes the expectation of your function from the case where it would be uniform: $\kappa_2(D, U) = \max_{h, \|h\|=1} \mathbf{E}_{D' \sim D}[|\mathbf{E}_{D'}[h] - \mathbf{E}_U[h]|]$.

Definition 4.1. The statistical dimension (SDN) of the planted problem Q with discrimination norm κ is the largest integer d so that $\forall D' \subseteq D$, $|D'| \geq \frac{|D|}{d}$, $\kappa_2(D', U) \leq \kappa$.

Then, we get the following result:

Theorem 4.2. *For planted k-SAT,* $SDN(Z, \frac{c\log(q)^{r/2}}{n^{r/2}}) \geq q$.

5 Shattering vs. Paring

Definition 5.1. Shattering.
As m increases, solutions change from one large cluster to exponentially many separated clusters.

Notably, this notion empirically coincides with algorithmic tractability.

Definition 5.2. Paring.
As m increases, the fraction of solutions that can be discarded goes up from inverse exponential to inverse polynomial.

Notably, there is a direct link between this definition and algorithm tractability.

6 Consequences

Refuting solvability for k-SAT is also hard for statistical algorithms and can be solved with complexity $n^{r/2}$. Feige's hypothesis (the weaker and more general version) holds for statistical algorithms. See this post for more details. Goldreich's PRG is secure against statistical attacks. Since convex programs can be solved by statistical algorithms, we have concrete lower bounds for stochastic convex programs: Namely, the dimension must be $n^{\Omega(r)}$ to even approximate solutions (i.e. exponential in input size to convex program).