SOS Lecture 1: Introduction / Boaz Barak

1 Introduction

Admin

Reading Sections 1 and 2 of my survey with Steurer "Sum of Squares proofs and the quest toward optimal algorithms"

Additional reading Introduction of Ryan and Yuan Zhou's paper "Approximability and Proof Complexity"

Lecture notes of Monique Laurent (`https://sites.google.com/site/mastermathsdp/lectures`) and Pablo Parrilo (`http://stellar.mit.edu/S/course/6/sp14/6.256/materials.html`)

Basic task— polynomial equations Given m polynomials $P_1, \ldots, P_m : \mathbb{R}^n \to \mathbb{R}$ of degree d. Find $x \in \mathbb{R}^n$ such that $P_i(x) = 0$ for all i. (Throughout the course accuracy doesn't matter— can think of $|P_i(x)| \leq \epsilon$ for some tiny $\epsilon > 0$.) Unlike linear case, not much difference between equalities and inequalities.

Example - 3SAT Can encode 3SAT formula as degree 3 polynomial equations: the equation $x_i^2 = x_i$ is equivalent to $x_i \in \{0,1\}$. The equation $x_i x_j (1 - x_k) = 0$ is equivalent to $\overline{x_i \wedge x_j \wedge \overline{x_k}} = \overline{x_i} \vee \overline{x_j} \vee x_k$.

Example - clique Given a graph $G = (V, E)$ the following equations encode that x is a 0/1 indicator vector of a k-clique: $x_i^2 = x_i$, $\sum x_i = k$, $x_i x_j = 0$ for all $(i,j) \notin E$.

Other examples (learning, etc..)

Two computational problems:

- (Search): Given satisfiable set of equations $\mathcal{E}$, find a solution x.
- (Refutation): Given unsatisfiable set of equations $\mathcal{E}$, prove that there is no solution.

Exercise Prove that both problems are NP hard even for degree $d = 2$. (Hint: the "decision" problem is easier than both and is NP hard.)

(Note: restricted form of $d = 2$ can be achieved - maximize or minimize $Q(x)$ such that $\sum x_i^2 = 1$.)

The SOS algorithm This is an algorithm that can be thought of as applying to both problems. Given a system of equations $\mathcal{E}$, the algorithm is invoked with some parameter ℓ (known as its *degree*) and takes $n^{O(\ell)}$ time to run.

Given that the problem is NP hard, it does not always return a solution, but it always returns something. Namely, the algorithm will always return either a "degree ℓ pseudo-solution" (later we'll call them "pseudo-distributions") or a proof that no such object exists.

Since every actual solution is in particular a "pseudo-solution" in the latter case we get a proof that the equations are in fact unsatisfiable. In the former case, we don't know whether the equations are satisfiable or not.

Some times, if we're lucky, the pseudo-solution might turn out to be an actual solution. It seems that empirically this is often the case in real-world applications of the SOS algorithm.

History Minkowski, Hilbert, P-satz, Grigoriev, Vorobjov, NZ Shor, Nesterov, Parillo, Lasserre

Why do (normal) people care about SOS Applications to: equilibrium analysis of continuous games, robust and stochastic optimization, statistics and machine learning, software verification, filter design, quantum computation and information, automated theorem proving, dynamics and control (robotics, flight controls, ...)

Analyzing the "falling leaf" mode of the U.S. Navy F/A-18 "Hornet":

A. Chakraborty, P. Seiler, and G. J. Balas. *Susceptibility of F/A-18 flight controllers to the falling-leaf mode: Nonlinear analysis.* Journal of guidance, control, and dynamics, 34(1):7385, 2011

Why do I care about SOS First, the SOS algorithm is a potential candidate to bypass the algorithmic bottleneck of the "Unique Games Conjecture". Namely, to get approximation guarantees to various problems such as "Max Cut", "Sparsest cut" and others (including the unique games and small set expansion problems themselves) that beat the current best known and are better than what would have been optimal if the UGC is true.

We do not know of course whether or not that's the case, and this is a fascinating question, motivating study of the SOS algorithm. To use a rather inaccurate analogy, at the moment, it seems that the SOS algorithm is a bit like quantum computing — for most problems we don't know that it offers advantages over simpler algorithms but there are very few structured examples where it actually helps a lot. However, it is possible that this is just the "tip of the iceberg" and the SOS algorithm actually can improve a large host of problems. In particular, there are many approximation problems for which there is a gap between the performance of the best known algorithm and the bounds given by the strongest known NP-hardness result, but it was shown that the gap would be closed if the UGC is true. If the SOS algorithm turns out to refute to UGC, it would mean that at least for some of these problems it can actually beat the best known algorithmic bounds. Also, regardless of the status of the UGC, recent work on SOS in the context of unsupervised learning problems suggests that it could be quite useful in that domain.

Meta algorithms Another reason to care about the SOS algorithm is that it is a "meta algorithm" and one that is potentially optimal in a variety of settings.

What is a "meta algorithm" and why is this so interesting? A *meta algorithm* is an algorithm that is not tailored for a particular problem, be it SAT, Max-Cut, or Clique, but is completely generic. The SOS algorithm takes as input a formulation of the problem as polynomial equations, and does not try to use any of the structure of the original problem. Nevertheless,

in many cases, we don't know of any tailored algorithm that outperforms (i.e., gets better guarantees in polynomial time) the SOS algorithm.

I find meta algorithms interesting because as a theoretical computer scientist, I want to know which computational problems are easy and which ones are hard, and beyond that to also know *why* they are easy or hard. Explanations such as "problem A is easy because it has an algorithm" or "problem B is hard because it can be reduced to from SAT" are not very satisfying. I am happier with explanations such as "problem A is easy because it is convex" and "problem B is hard because it is not convex" (replacing "convex" with other properties such as having polymorphisms, matroid structure, etc..). To get such an explanation, we would want a meta algorithm that solves all problems that have some property P, together with a proof that problems without P cannot be efficiently solved. The SOS algorithm is a natural candidate for such an algorithm. (We don't want a pathological algorithm that simply enumerates Turing machines or some such thing.)

Another reason to consider meta algorithms is that they can form the basis of hardness conjectures. NP hardness is a very powerful tool, but there are still many problems that we suspect are hard but don't know how to prove NP hardness for. Moreover, there are several domains, such as average-case complexity, where NP hardness doesn't really apply. One avenue to get conditional hardness results is to formulate a conjecture that a particular meta algorithm A (such as SOS) is *optimal* in some domain. That means that, under this conjecture, to prove that a problem X is hard, we only need to show that A can't solve it, something that (while often non-trivial) is usually doable. Of course such hardness results are only worth as much as the assumption they are based on, but it is plausible, or at least possible, that there exist natural meta algorithms that are optimal for large classes of problems. Indeed, the SOS method may well be such an algorithm.

2 Definitions

Distributions The SOS algorithm is designed to solve some equations $\mathcal{E}$. However, even in the best case, it does not return a solution x to $\mathcal{E}$, but rather a *distribution* $\{x\}$ over solutions. This is actually inherent whenever we want to use a convex relaxation (see homework).

Of course a distribution over solutions in particular contains one solution, and so it might seem that this is only better, but there is a catch— the algorithm only returns the *low order moments* of the distribution. These moments are not necessarily sufficient to sample an element from the distribution, or even to find any x in its support.

Of course, we might not always be in the best case, and the algorithm might return an object that is not even a distribution but rather a "pseudo-distribution", which is defined next.

Notation Let $\mathbb{R}_d = \mathbb{R}_d[x]$ be the set of degree d polynomials in variables $x = x_1, \ldots, x_n$. Let $\mathbb{S}_d[x]$ be the subset of $\mathbb{R}_d$ corresponding to sums of squares.

An element of $\mathbb{R}_d$ can be described by n^d numbers. $\mathbb{S}_d$ is a convex cone, which we have a separation oracle for. (See homework exercises.)

Pseudo-distributions We say that $\{x\}$ is a degree-d pseudo-distribution if there is a mapping that takes every polynomial $P \in \mathbb{R}_d[x]$ to a number $\tilde{\mathbb{E}}P$ satisfying:

Normalization $\tilde{\mathbb{E}}1 = 1$.

Linearity $\tilde{\mathbb{E}}[\alpha P + \beta Q] = \alpha\tilde{\mathbb{E}}P + \beta\tilde{\mathbb{E}}Q$

Positivity $\tilde{\mathbb{E}}P^2 \geq 0$ for every $P \in P_{d/2}[x]$.

The number $\tilde{\mathbb{E}}P$ is called the *pseudo expectation* of x.

Exercises Pseudo distributions are indeed more general objects than actual distributions, for degree $d > 2$:

Prove that there exists a degree 4 pseudo-distribution $\{x\}$ such that there is no actual distribution $\{y\}$ such that $\tilde{\mathbb{E}}P(x) = \mathbb{E}P(y)$ for every P of degree at most 4. (Can you do so for degree 3?)

Gaussian quadratic sampling lemma: If $\{x\}$ is a degree $d \geq 2$ pseudo distribution, then there exists a Gaussian distribution $\{y\}$ such that $\tilde{\mathbb{E}}P(x) = \mathbb{E}P(y)$ for every polynomial P of degree at most 2.

Notation Let $\mathbb{R}_d^*$ be the set of linear functions from $\mathbb{R}_d$ to $\mathbb{R}$. Let $\mathbb{E}_d \subseteq \mathbb{R}_d^*$ be the set of degree d pseudo distributions.

Exercise: Let $E \in R_d^*$. Then $E \in \mathbb{E}_d$ if and only if $E1 = 1$ and $E(P) = \sum_{i=1}^r \int Q_i^2(x)P(x)d\mu$ for some measure μ and polynomials $Q_1, \ldots, Q_r \in \mathbb{R}_{d/2}$.

Exercise: $\mathbb{E}_d$ is a convex cone with a separation oracle.

SOS proofs A degree d SOS proof that equations $\mathcal{E} = \{P_1 = \cdots = P_m = 0\}$ are not satisfiable is a sequence of polynomial $Q_1, \ldots, Q_m \in \mathbb{R}_d$ and polynomial $S \in \mathbb{S}_d$ such that

$$\sum P_i Q_i = 1 + S$$

The set of degree d proofs is a convex cone with a separation oracle (exercise).

SOS algorithm - primal formulation We say that a degree d pseudo-distribution $\{x\}$ satisfies a the constraint $\{P = 0\}$ if $\tilde{\mathbb{E}}PQ = 0$ for all $Q \in \mathbb{R}_{d-\deg P}[x]$.

SOS Theorem - primal (Shor,Nesterov, Parrilo, Lasserre). There is a polynomial time algorithm that on input a set of equations $\mathcal{E}$, outputs a degree d pseudo-distribution $\{x\}$ that satisfies $\mathcal{E}$ if and only if such $\{x\}$ exist. (Ignoring here issues of boundedness, numerical accuracy.)

Proof sketch: The set of tuples $(S, Q_1, \ldots, Q_m)$ that form an SOS proof for some equations $\mathcal{E}$ is a convex cone that has an efficient separation oracle, and hence can be optimized over, e.g., using the Ellipsoid algorithm. (In practice, people apparently use interior point methods.) Also, as shown in the homework, finding an SOS proof can be phrased as a *semidefinite program*.

SOS algorithm - dual formulation On input a set of equations $\mathcal{E}$, there is a polynomial-time algorithm that outputs a degree d proof that $\mathcal{E}$ is unsatisfiable if and only if such a proof exists.

Proof sketch: The set of pseudo-expectations is simply a set of positive semidefinite matrices that satisfy some linear equations.

Strong duality exactly one of the conditions has to hold. (See Theorem 2.7 in the survey.)

Positivstellensatz (Stengle 64, Krivine 74) For every unsatisfiable system $\mathcal{E}$ of equalities there exists a finite d s.t. $\mathcal{E}$ has a degree d proof of unsatisfiability.

Exercise Prove P-satz for systems that include the constraint $x_i^2 = x_i$ for all i. In this case, show that d needs to be at most $2n$ (where n is the number of variables).

Corollary: the SOS algorithm does not need more than $n^{O(n)}$ time to solve polynomial equations on n Boolean variables. (Not very impressive bound, but good to know. In all TCS applications I am aware of, it's easy to show that the SOS algorithm will solve the problem in exponential time.)

Views of pseudo-distributions The notion of pseudo-distribution is somewhat counter-intuitive and takes a bit of time to get used to. It can be viewed from the following perspectives:

- Pseudo-distributions is simply a fancy name for a PSD matrix satisfying some linear constraints, which is the dual object to SOS proofs.
- SOS proofs of unbounded degree is a sound and complete proof system in the sense that they can prove any true fact (phrased as polynomial equations) about actual distributions over $\mathbb{R}^n$.

  SOS proofs of degree d is a sound and not complete proof system for actual distributions, but it is a (sound and) complete system for degree d pseudo-distributions, in the sense that any true fact that holds not merely for actual distributions but also for degree d pseudo-distributions has a degree d SOS proof.
- In statistical learning problems (and economics) we often capture our knowledge (or lack thereof) by a distribution. If an unknown quantity X is selected and we are given the observations y about it, we often describe our knowledge of by a the distribution $X|y$. In computational problems, often the observations y completely determine the value X, but pseudo-distribution can still capture our "computational knowledge".
- The proof system view can also be considered as a way to capture our limited computational abilities. In the example above, a computationally unbounded observer can deduce from the observations y all the true facts it implies and hence completely determine X. One way to capture the limits of a computationally bounded observer is that it can only deduce facts using a more limited, sound but not complete, proof system.

Lessons from History It took about 80 years from the time Hilbert showed that polynomials that are not SOS exist non-constructively until Motzkin came up with an explicit example, and even that example has a low degree SOS proof of positivity. One lesson from that is the following:

"Theorem": If a polynomial P is non-negative and "natural" (i.e., constructed by methods known to Hilbert— not including probabilistic method), then this can fact has a low degree SOS proof.

Corollary (Marley, 1980): If you analyze the performance of an SOS based algorithm pretending pseudo-distributions are actual distributions, then unless you used Chernoff+union bound type arguments, then every little thing gonna be alright.

We will use Marley's corollary extensively in analyzing SOS algorithms.