STATS 314A: Advanced Statistical Theory
The Sum-of-Squares Algorithmic Paradigm in Statistics
Instructor: Tselil Schramm

Lecture 1
March 30, 2022

Lecture 1: Implementing sum-of-squares algorithms with semidefinite programming

In this lecture we introduce semidefinite programming, a powerful class of efficiently-solvable[1] convex programs. We then describe how sum-of-squares algorithms can be implemented with semidefinite programming. Some bibliographic remarks will be deferred to the end.

These notes have not been reviewed with the same scrutiny applied to formal publications. There may be errors.

1 Recalling the context

Recall that in the previous lecture, we had defined the notion of a *degree-k pseudoexpectation* respecting a set of polynomial axioms/constraints.

Definition 1.1. For a set of polynomial axioms $\mathcal{A} = \{f_i = 0\}_{i\in[M]} \cup \{g_j \geqslant 0\}_{j\in[K]}$, we say that $\widetilde{\mathbf{E}} : \mathbb{R}[x] \to \mathbb{R}$ is a *degree-k pseudoexpectation satisfying A* if it is a linear operator with the following properties:

1. Scaling: $\widetilde{\mathbf{E}}[1] = 1$
2. Non-negativity of squares: $\widetilde{\mathbf{E}}[h^2] \geqslant 0$ for any polynomial $h \in \mathbb{R}[x]$ with $\deg(h) \leqslant k/2$
3. Respecting axioms: $\widetilde{\mathbf{E}}[af_i] = 0$ for all $i \in [M]$ and $a \in \mathbb{R}[x]$ satisfying $\deg(af_i) \leqslant k$, and $\widetilde{\mathbf{E}}[b^2 g_j] \geqslant 0$ for all $j \in [K]$ and $b \in \mathbb{R}[x]$ satisfying $\deg(b^2 g_j) \leqslant k$.

I had promised you that given a polynomial system A, there exists an algorithm which runs in time $\mathrm{size}(A)^{O(k)}$ for finding a corresponding degree-k pseudoexpectation. I also promised you that if $p, q \in \mathbb{R}[x]$ for $x \in \mathbb{R}^N$ satisfy $p \geqslant q$ as a degree-k sum-of-squares inequality, then there is a time $N^{O(k)}$ that finds such a sum-of-squares proof. That algorithm is based on a clever application of a fundamental convex optimization primitive called *semidefinite programming*.

2 Semidefinite Programs

A *semidefinite program* (SDP) is a system of equations in a matrix-valued program variable $Z \in \mathbb{R}^{d\times d}$ that has the following form:

$$Z \succeq 0 \tag{1}$$

$$\langle Z, A_i\rangle = a_i \quad \forall i \in [m], \tag{2}$$

$$\langle Z, B_j\rangle \geqslant b_j \quad \forall j \in [m'], \tag{3}$$

where for all $i \in [m]$, $j \in [m']$, $A_i, B_j \in \mathbb{R}^{d\times d}$ and $a_i, b_j \in \mathbb{R}$. Often, we might be interested in maximizing an *objective function* subject to these constraints; in this case our goal is to determine what is $\max\langle Z, C\rangle$ for some $C \in \mathbb{R}^{d\times d}$ subject to the constraints above.

[1]This comes with some asterices, as we will see.

The constraint (1) requires that Z be *positive semidefinite*, which means that it must be symmetric and have no negative eigenvalues. There are several equivalent characterizations of positive semidefinite matrices:

$$Z \succeq 0 \iff \exists V \in \mathbb{R}^{d\times d} \text{ s.t. } Z = VV^\top \iff \forall v \in \mathbb{R}^d,\ v^\top Z v \geqslant 0.$$

You can check that these are equivalent.

The set of Z which satisfy such a system of equations form a convex set. Hence, a semidefinite program is a convex program, and so if it is feasible we can solve it with one of several off-the-shelf algorithms for finding solutions to convex programs. Alternatively, if it is not feasible, these algorithms will present a certificate of infeasibility.

Remark 2.1. One can transform an SDP with inequality constraints into an SDP with only equality constraints using the following transformation: for each inequality $\langle Z, B_j\rangle \geqslant b_j$, one adds an extra "slack variable" $s_j \geqslant 0$, so that $\langle Z, B_j\rangle - s_j = b_j$. To implement this as a linear constraint, one adds an extra dimension to Z, and the constraint becomes

$$\left\langle \begin{bmatrix} Z & 0 \\ 0 & s_j \end{bmatrix}, \begin{bmatrix} B_j & 0 \\ 0 & -1 \end{bmatrix} \right\rangle = b_j.$$

Since s_j is on the diagonal and the SDP requires that the matrix $\begin{bmatrix} Z & 0 \\ 0 & s_j \end{bmatrix} \succeq 0$, we have that $s_j \geqslant 0$ is enforced. This comes at the cost of increasing the size of the SDP.

2.1 Algorithms for solving SDPs

In practice, *interior point methods* are the method of choice for solving semidefinite programs. Below we state the current best guarantees for interior-point based SDP solvers; even faster methods are available for instances of SDP where the constraint matrices are sparse.

Theorem 2.2 (From [JKL+20]). *There is an interior point method which solves an SDP over matrices of size $d \times d$ and with m equality constraints in time $\tilde{O}(d^{2.5} \cdot m + \sqrt{d} \cdot m^\omega + d^{\omega+1/2}$, where $\omega < 2.38$ is the matrix multiplication exponent. The $\tilde{O}$ hides logarithmic factors in n and in $\frac{1}{\varepsilon}$ for ε the accuracy.*

The *ellipsoid algorithm* is a second method for solving SDPs, or more generally, for determining if a convex set is nonempty. Though it is not as efficient, the ideas behind the algorithm are easy to explain. In order to implement the ellipsoid algorithm on a set K, we require a *separation oracle* for K:

Definition 2.3 (Separation Oracle). A *separation oracle* for a convex set $K \subset \mathbb{R}^M$ is an algorithm that takes as input a point $x \in \mathbb{R}^M$. If $x \in K$, the algorithm returns TRUE, otherwise if $x \notin K$, the algorithm returns a *separating hyperplane*, or a vector $v \in \mathbb{R}^M$ such that $\langle x, v\rangle < 0$ and $\langle y, v\rangle > 0$ for all $y \in K$.

Every convex set must have a separation oracle, by definition of convexity. The special thing about SDPs (and other structured convex set) is that there is a separation oracle which uses $(m + m')d^2 + O(d^3)$ arithmetic operations, since each of the constraints in (2), (3) can be checked in d^2 operations, and the constraint (1) can be checked by computing the eigenvalues of Z using Gaussian elimination in $O(d^3)$ operations. Of course, I am being sloppy here because the inputs to the program are real numbers, and I am ignoring bit complexity issues. Suppose I allow myself $\log(1/\delta)$ bits to represent each number; the running time is now the number of operations times $\log \frac{1}{\delta}$. If the system is infeasible, but there exists some Z' which is $o(\delta)$-close to feasible, I might get a false positive. Conversely, if the only feasible Z are of magnitude $o(\delta)$ (say, $Z = \frac{1}{100}\delta \mathbb{1}$), then we might get a false negative. I'll take the liberty of continuing to ignore this issue, even though this is not 100% kosher. In many of the settings we care about, it is known that the SDP solutions are well-conditioned, and this problem will not arise. See for example [O'D17, RW17] for further discussion.

Algorithm 2.4 (The Ellipsoid Algorithm)**. Input:** a separation oracle oracle for a convex set $K \in \mathbb{R}^M$, a maximum radius R with the guarantee that $K \subset \mathcal{B}_R(0)$, a minimum radius r with the guarantee that if K is nonempty, then there exists $x \in \mathbb{R}^M$ with $\mathcal{B}_r(x) \subset K$.

```
1. Set ellipsoid = $\mathcal{B}_R(0)$.
2. While vol(ellipsoid) $> r^M$:
   (a) Compute the center $x$ of ellipsoid, and call oracle($x$). If the oracle returns true, return NONEMPTY.
   (b) Otherwise, $v$ = oracle($x$) is a separating hyperplane. Update our bounding set ellipsoid to be
       the minimum-volume ellipsoid containing the previous set, excluding the points cut off by the
       hyperplane: $\{y \in \text{ellipsoid} : \langle y, v\rangle > 0\}$
3. Return EMPTY.
```

The idea is that we maintain set, a set which is known to contain K. At each step, we choose a point in set, and we call our oracle on that point. If the oracle says the point is in K, then K is nonempty. Otherwise, we use the separating hyperplane to update our bounding set. By choosing the point to always be at the center of the ellipsoid containing set, we guarantee that whenever we don't hit K, we make progress by removing an $\Omega(\exp(-\frac{1}{M}))$-fraction of the set's volume. In $O(M^2 \cdot \log \frac{R}{r})$ steps, the volume shrinks from $O(R)^M$ to r^M, and if K has volume r^M or more, the centroid of the ellipsoid will hit it at or before this step.

I won't go through the formal analysis, but hopefully the idea is clear. Check out [Goe, LGS88] if you are interested in the details. The upshot is the following theorem:

Theorem 2.5. *The ellipsoid algorithm terminates in $O(M^2 \cdot \log \frac{R}{r})$ steps, each taking $O(1)$ arithmetic operations and one oracle call. If K contains a ball of radius r, the oracle will return* NONEMPTY*, and if K is empty the oracle will return* EMPTY.

Remark 2.6. the special case of semidefinite programming with precision δ, the running time is on the order $O(((m+m')d^2+d^3)\cdot d^6 \cdot \log \frac{R}{r} \log \frac{1}{\delta})$, which is $\mathrm{poly}(d,m)$ if $\frac{R}{r} \leqslant \exp(\mathrm{poly}(d,m))$ and $\frac{1}{\delta} \leqslant \exp(\mathrm{poly}(d,m))$.

3 Sum-of-squares algorithms as semidefinite programs

3.1 Pseudoexpectations

We will show how to solve for a pseudoexpectation operator using a semidefinite program: Let $k \in \mathbb{N}$ be even. Notice that by linearity, the value of a degree-k pseudoexpectation $\widetilde{\mathbf{E}}$ on any polynomial p of degree at most k can be determined using the value of $\widetilde{\mathbf{E}}[x^S]$ for every monomial $x^S = \prod_{i \in S} x_i$ with $|S| \leqslant k$. We will write a semidefinite program that searches for a feasible set of values $\{\widetilde{\mathbf{E}}[x^S]\}_{S \in [n]^{\leqslant k}}$.

Our matrix-valued variable Z will be real symmetric matrix with rows/columns indexed by all subsets of $[n]$ of size at most $k/2$ (including the empty set). For $A, B \in [n]^{\leqslant k/2}$, we will identify $Z_{A,B} = \widetilde{\mathbf{E}}[x^A x^B]$. So, to ensure that $\widetilde{\mathbf{E}}$ is well-defined and also to ensure that the scaling property holds, we enforce the following linear constraints on Z:

$$Z_{\emptyset,\emptyset} = 1, \quad \text{and} \qquad Z_{A,B} = Z_{U,V} \quad \forall A,B,U,V \subset [n]^{\leqslant k/2} \; s.t.\; A \cup B = U \cup V,$$

where we are taking the union as multisets.

Let's assume $\mathcal{A} = \{f_i = 0\}_{i\in[m]}$; that is, let's only work with equality constraints (we leave inequalities as an exercise). In order to ensure that the $\widetilde{\mathbf{E}}$ we define using Z respects the axioms $\mathcal{A}$, we add linear constraints to Z. To this end, consider some $f_i \in \mathcal{A}$, and write it in the monomial basis, $f_i(x) = \sum_{A\in[n]^k} \hat{f}_i(A)\cdot x^A$. Now, we will construct a constraint matrix F_i with rows and columns indexed by $[n]^{\leqslant k/2}$. For each $A \in [n]^k$, choose some arbitrary partition of A into two subsets of size at most $k/2$ each, say $A = A_1 \cup A_2$. Set entry $(F_i)_{A_1,A_2} = \hat{f}_i(A)$. One can then check that

$$\langle Z, F_i\rangle = \sum_{A\in[n]^{\leqslant k}} \hat{f}_i(A)\cdot Z_{A_1,A_2} = \sum_{A\in[n]^{\leqslant k}} \hat{f}_i(A)\cdot \widetilde{\mathbf{E}}[x^{A_1}x^{A_2}] = \sum_{A\in[n]^{\leqslant k}} \hat{f}_i(A)\cdot \widetilde{\mathbf{E}}[x^A] = \widetilde{\mathbf{E}}[f_i],$$

so we can enforce that $\widetilde{\mathbf{E}}$ respects $\widetilde{\mathbf{E}}[f_i] = 0$ by adding the constraint $\langle Z, F_i\rangle = 0$. Notice that there may be many valid choices for F_i, and the symmetries that we have enforced in Z mean that they are all equivalent.

To ensure that we also have $\widetilde{\mathbf{E}}[c_i f_i] = 0$ for all $c_i \in \mathbb{R}[x]$ such that $\deg(c_i f_i) \leqslant k$, we can simply repeat the above for each of the polynomials $x^S \cdot f_i$ where $S \subset [n]$ and $\deg(x^S f_i) \leqslant k$. That is, we add the constraint $x^S f_i = 0$. Since these form a basis for all polynomials $c_i f_i$ of degree at most k, this is sufficient to ensure that $\widetilde{\mathbf{E}}$ respects $\mathcal{A}$.

Claim 3.1. The construction of $\widetilde{\mathbf{E}}$ above yields a valid degree-k pseudoexpectation for the system $\mathcal{A}$.

Proof. By construction, linearity and scaling hold for the operator $\widetilde{\mathbf{E}}$ that we have defined. Above, we have argued that $\widetilde{\mathbf{E}}$ respects $\mathcal{A}$. All that remains is to check the non-negativity of squares; to see that this holds, consider any polynomial p of degree at most $k/2$ written in the monomial basis, $p(x) = \sum_{S\in[n]^{\leqslant k/2}} \hat{p}(S)\cdot x^S$. Let $\hat{p}$ be the vector of p's coefficients, with entries indexed by multisubsets $S \in [n]^{\leqslant k}$. We then have that

$$\widetilde{\mathbf{E}}[p(x)^2] = \hat{p}^\top Z \hat{p} \geqslant 0,$$

by the positive semidefiniteness of Z. This completes the proof. □

Running time. Our matrix variable Z has dimension $n^{O(k)} \times n^{O(k)}$, and $m \cdot n^{O(k)}$ constraints from $\mathcal{A}$, as well as $n^{O(k)}$ linear constraints of the form $Z_{A,B} = Z_{U,V}$ which enforce the consistency. So using the guarantees of the ellipsoid algorithm, our algorithm runs in time $\mathrm{poly}(m,n)^k$. When k is constant, this is polynomial time.

3.2 Finding sum-of-squares proofs

In the first lecture, we asserted that one can use $(\mathrm{size}(\mathcal{A}))^{O(k)}$-sized SDPs to find a sum-of-squares proof of degree-k that $p \geqslant q$ for $p, q \in \mathbb{R}[x]$ whenever this is true. We'll save this for a homework exercise.

We comment as well that the Semidefinite Program above is a *feasibility* program. One can check (by applying convex duality) that the SDP's convex dual is one that searches for a proof of the form

$$-1 = \sum_t s_t^2 + \sum_{i\in[m]} c_i f_i,$$

with $s_t, c_i \in \mathbb{R}[x]$ satisfying $\deg(s_t) \leqslant k/2$ and $\deg(c_i f_i) \leqslant k$ for all t, i. This is a sum-of-squares *refutation* of the polynomial system $\mathcal{A}$; since the above inequality cannot hold for real x if $f_i(x) = 0$ for all i, this shows that no x can satisfy all of the constraints in $\mathcal{A}$.

4 Conclusion

Bibliographic remarks. Interior point methods were first applied to Linear Programs by Karmarkar [Kar84]; the citation given above, [JKL+20], holds the current record for the (theoretical) fastest interior point method for SDPs. The ellipsoid algorithm was discovered by Khachiyan [Kha80]. The algorithm has a storied history, and we won't attempt to survey it here. The ellipsoid algorithm now belongs to a family of methods called *cutting plane methods*, in which one starts with a set which is known to contain the target convex set and iteratively cuts down its size. We remark that since then, there have been several specializex cutting plane methods for semidefinite programs obtaining better running times, the current record is held by [JLSW20].

Sum-of-squares programming originated in several independent works by Lasserre [Las01], Nesterov [Nes00], Parrilo [Par00], and Shor [Sho87] near the end of the 20th century. The proofs-to-algorithms paradigm was popularized in the algorithms community starting with the work of Barak, Brandao, Harrow, Kelner, Steurer and Zhou [BBH+12] (see also [OZ13, BKS14, BKS15]).

Thanks to Jay Mardia and Louigi Addario-Berry for helpful suggestions in improving the presentation of these notes.

Contact. Comments are welcome at tselil@stanford.edu.

References

[BBH+12] Boaz Barak, Fernando GSL Brandao, Aram W Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 307–326, 2012. 5

[BKS14] Boaz Barak, Jonathan A Kelner, and David Steurer. Rounding sum-of-squares relaxations. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 31–40, 2014. 5

[BKS15] Boaz Barak, Jonathan A Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 143–151, 2015. 5

[Goe] Michel Goemans. Lecture notes on the ellipsoid algorithm. MIT 18.433 course notes. 3

[JKL+20] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *2020 IEEE 61st annual symposium on foundations of computer science (FOCS)*, pages 910–918. IEEE, 2020. 2, 5

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 944–953, 2020. 5

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984. 5

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. 5

[Las01] Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001. 5

[LGS88] Loszlo Lovasz, Martin Grotschel, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988. 3

[Nes00] Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, pages 405–440. Springer, 2000. 5

[O'D17] Ryan O'Donnell. Sos is not obviously automatizable, even approximately. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 3

[OZ13] Ryan O'Donnell and Yuan Zhou. Approximability and proof complexity. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1537–1556. SIAM, 2013. 5

[Par00] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000. 5

[RW17] Prasad Raghavendra and Benjamin Weitz. On the bit complexity of sum-of-squares proofs. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 3

[Sho87] Naum Zuselevich Shor. An approach to obtaining global extremums in polynomial mathematical programming problems. *Cybernetics*, 23(5):695–700, 1987. 5