CS 270 Algorithms **Spring 2021**

Lecture 20: The Sum-of-Square Semidefinite Programming Hierarchy

Lecturer: Prasad Raghavendra Scribe: Haydn Gwyn, Naveen Durvasula, Selina Kim

20.1 Introduction

Today we continue the previous lecture's discussion of semidefinite programming.

Recall the general setup: we seek a positive semidefinite matrix $M \in \mathbb{R}^{n \times n}$, whose entries satisfy certain linear constraints. Equivalently, we can suppose that the entries of M are in fact the pairwise inner products of a set $\{v_1, \ldots, v_n\}$ of vectors, so our conditions on the entries of M becomes conditions on the inner products of $v_1, \ldots, v_n$. All of this has a very geometric feel and seems to necessitate some level of geometric intuition around the problems that we approach. The goal today is to reveal the generality of semidefinite programming by showing how it can be used to solve polynomial systems. These take the form

$$\begin{cases} P_1(x_1, \ldots, x_n) \geq 0 \\ P_2(x_1, \ldots, x_n) \geq 0 \\ \vdots \\ P_n(x_1, \ldots, x_n) \geq 0 \end{cases}$$

where P_i is a polynomial function of $x_1, \ldots, x_n$. Note that we can just as well include conditions of the form $P(x) \leq 0$ and $P(x) = 0$, as

$$P(x_1, \ldots, x_n) \leq 0 \iff -P(x_1, \ldots, x_n) \geq 0$$

$$P(x_1, \ldots, x_n) = 0 \iff P(x_1, \ldots, x_n) \geq 0 \wedge P(x_1, \ldots, x_n) \leq 0$$

The language of polynomial systems turns out to be a very expressive one; there are many problems that can be reduced to or well-approximated by solving a system of polynomial inequalities. The example we treat today is the minimum vertex cover problem.

In the minimum vertex cover problem, we are given a graph $G = (V, E)$ as input, and our goal is to determine the smallest subset $S \subseteq V$ such that every edge $(i, j) \in E$ is covered by S. That is, for every edge (i, j) of E, we have $i \in S$ or $j \in S$ (or both).

How can we express this condition in terms of polynomial constraints? We aim to produce the indicator vector for S. That is, we seek a vector x satisfying

$$x_i = \begin{cases} 1 & i \in S \\ 0 & i \notin S \end{cases}$$

Observe that

$$x_i \in \{0, 1\} \iff x_i^2 - x_i = 0$$

This is how we enforce the 0-1 condition. Additionally, since we want the set S to cover G, we note that

$$i \in S \vee j \in S \iff (1 - x_i)(1 - x_j) = 0$$

We will have a condition of the form on the right for each $(i, j) \in E$. We have now enforced that x is an indicator vector of some set S and that S covers G. We want to minimize the size of our vertex cover, so we seek

$$\operatorname{argmin} \sum_{i \in V} x_i$$

We render this as a polynomial inequality by writing

$$\sum_{i \in V} x_i \leq C$$

for a constant C, so we have turned the minimum vertex cover problem into the polynomial system

$$\begin{cases} x_i^2 - x_i & \forall i \in V \\ (1 - x_i)(1 - x_j) = 0 & \forall (i, j) \in E \\ \sum_{i \in V} x_i \leq C \end{cases}$$

As a further example of the power of polynomial systems, notice that we can express MaxCut as a polynomial system:

$$\begin{cases} x_i^2 = 1 & \forall i \in V \\ \sum_{(i,j) \in E} (x_i - x_j)^2 \geq C \end{cases}$$

Note how we have similarly employed the technique of converting an argmax into a $\geq C$.

20.2 Convexifying the Feasible Set

A frequent barrier in solving polynomial systems is that the set over which we're working is not convex (or is even discrete). As such, we aim to convexify our feasible set. To that end, let $P = \{P_i(x) \geq 0\}$ denote the collection of polynomial constraints, and let $\mathcal{S}$ denote the set of solutions to $\mathcal{P}$. Instead of directly considering $\mathcal{S}$, we could instead consider the set $\Delta_{\mathcal{S}}$ of probability distributions over $\mathcal{S}$.

In the MaxCut problem, the solution set would be the following:

$$\mathcal{S} = \{\text{set of } \pm 1 \text{ assignments with value} \geq C\}$$

The probability distributions over $\mathcal{S}$ may then be written as vectors of dimension $|\mathcal{S}|$. Unfortunately, the set of probability distributions over $\mathcal{S}$ can be very large. Naively, in the above example, if we assume that every element of $\{\pm 1\}^n$ is a solution, then the set of probability distributions lies in the space $\mathbb{R}^{2^n}$. Although we obtain one linear constraint in that any probability distribution must sum to 1, the dimension is still exponentially large and thus does not offer an efficient approach. With that said, this convexification idea gives a mental picture that is close to what we are looking for.

Due to the above reasoning, processing the entirety of any probability distribution μ over the set $\mathcal{S}$ of solutions is unfeasible. However, the moments of this distribution might be easier to handle. Recall the polynomial system for the vertex cover problem:

$$\begin{cases} x_i^2 - x_i & \forall i \in V \\ (1 - x_i)(1 - x_j) = 0 & \forall (i, j) \in E \\ \sum_{i \in V} x_i \leq C \end{cases}$$

We first aim to find the degree 1 and degree 2 moments of any distribution over solutions to the above inequalities. Letting μ be such a distribution, and letting $x \sim \mu$, we define

$$X_i := \mathbb{E}[x_i] \qquad X_{ij} := \mathbb{E}[x_i x_j]$$

for $i, j \in V$. Observe that we have defined $n + n + \binom{n}{2}$ (notably, polynomially many) variables. To find these moments, let's review our constraints. We have

$$x_i^2 - x_i = 0 \implies \mathbb{E}\left[x_i^2 - x_i\right] = 0 \implies \mathbb{E}\left[x_i^2\right] - \mathbb{E}[x_i] = 0 \implies X_{ii} - X_i = 0$$

We see that indeed every polynomial constraint that we have turns into a linear constraint on the moments. In particular,

$$(1 - x_i)(1 - x_j) = 0 \implies \mathbb{E}_{x\sim\mu}[(1 - x_i)(1 - x_j)] = 0 \implies 1 - X_i - X_j + X_{ij} = 0$$

Finally,

$$\sum_{i\in V} x_i \leq C \implies \mathbb{E}\left[\sum_{i\in V} x_i - C\right] \leq 0 \implies \sum_{i\in V} X_i - C \leq 0$$

Thus, we have obtained a linear program.

20.3 Moment Matrices

There are further constraints on the moments that must be true in general. For example,

$$(x_1 - x_5 + x_7)^2 \geq 0 \implies \mathbb{E}\left[(x_1 - x_5 + x_7)^2\right] \geq 0 \implies X_{11} + X_{55} + X_{77} - 2X_{15} - 2X_{57} + 2X_{71} \geq 0$$

Upon expanding out the above square, we get a further linear constraint on the moments. More generally, we can consider any polynomial function of $x_1, \ldots, x_n$ and square it to obtain a function that must be non-negative:

$$p(x) := \left(p_0 + \sum p_i x_i\right)^2 \geq 0 \implies 0 \leq \mathbb{E}[p(x)] = p_0^2 + 2\sum_i p_0 p_i X_i + \sum_i p_i^2 X_{ii} + \sum_{i\neq j} p_i p_j X_{ij} \geq 0$$

The function $p(x)$ has a more concise expression that we can offer after introducing moment matrices.

The *second moment matrix* takes the following form:

$$M_2 := \begin{array}{c} \\ 1 \\ x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & x_1 & x_2 & \cdots & x_j & \cdots & x_n \end{array} \\ \begin{pmatrix} 1 & X_1 & X_2 & \cdots & X_j & \cdots & X_n \\ X_1 & X_{11} & X_{12} & \cdots & X_{1j} & \cdots & X_{1n} \\ X_2 & X_{21} & X_{22} & \cdots & X_{2j} & \cdots & X_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ X_i & X_{i1} & X_{i2} & \cdots & X_{ij} & \cdots & X_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ X_n & X_{n1} & X_{n2} & \cdots & X_{nj} & \cdots & X_{nn} \end{pmatrix} \end{array}$$

Notice that if we set $\boldsymbol{p} = (p_0, p_1, \ldots, p_n)$, then we have

$$\left(p_0 + \sum p_i x_i\right)^2 \geq 0 \iff \boldsymbol{p}^T M_2 \boldsymbol{p} \geq 0$$

This condition holds for any choice of $\boldsymbol{p}$, which is equivalent to the statement that M_2 is positive semidefinite. But notice now that we seek a PSD matrix M_2 whose entries satisfy certain constraints - the exact setup of semidefinite programming!

We can rewrite the constraints of our vertex cover problem as an SDP problem

$$\begin{cases} X_{ii} - X_i = 0 & \forall i \\ 1 - X_i - X_j + X_{ij} = 0 & \forall (i,j) \in E \\ \sum_i x_i \leq C \\ M_2 \succeq 0 \end{cases}$$

Observe that in this formulation, we did not need to appeal to any geometric intuition as we did in the MaxCut problem. We refer to this as a degree-2 SDP, as we only recover the second moments of the true distribution of solutions. We note that this is a relaxation of our original problem (the relaxation came when we passed from constraints on the variables x_i to constraints on the moments X_{ij} by taking expecations) – recovering more detailed information about the true distribution is still intractable.

Why stop at degree 2? Suppose we want to find such an SDP for higher moments. Fix $d \in \mathbb{N}$. The degree $2d$ sum-of-squares SDP relaxation has variables given by the moments up to degree $2d$. Such moments appear as

$$X_i := \mathbb{E}[x_i] \qquad X_{ij} := \mathbb{E}[x_i x_j] \qquad X_{ijk} := \mathbb{E}[x_i x_j x_k] \qquad \cdots \qquad X_{i_1 i_2 \cdots i_{2d}} = \mathbb{E}[x_{i_1} x_{i_2} \cdots x_{i_{2d}}]$$

which yields

$$\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{2d} \sim n^{2d}$$

variables in all. As before, polynomial constraints translate into linear constraints on the moments, as we can write

$$p(x) = \sum_\sigma p_\sigma x_\sigma \geq 0 \implies \sum p_\sigma X_\sigma \geq 0$$

where $\sigma = (i_1, \ldots, i_k)$ is a tuple of indices with $k \leq 2d$. This gives linear constraints on the moments. The analog to M_2 above is the *$2d$th moment matrix* M_{2d}, which takes the form

$$\begin{array}{c} \\ 1 \\ x_1 \\ \vdots \\ x_n \\ x_1^2 \\ \vdots \\ x_n^2 \\ \vdots \\ x_\sigma \\ \vdots \\ x_n^d \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} 1 & x_1 & \cdots & x_n & x_1^2 & \cdots & x_n^2 & \cdots & x_{\sigma'} & \cdots & x_n^d \end{array} \\
\begin{pmatrix}
1 & X_1 & \cdots & X_n & X_{11} & \cdots & X_{nn} & \cdots & X_{\sigma'} & \cdots & X_{nn\cdots n} \\
X_1 & X_{11} & \cdots & X_{1n} & X_{111} & \cdots & X_{1nn} & \cdots & X_{\{1\}\cup\sigma'} & \cdots & X_{1nn\cdots n} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
X_n & X_{n1} & \cdots & X_{nn} & X_{n11} & \cdots & X_{nnn} & \cdots & X_{\{n\}\cup\sigma'} & \cdots & X_{nnn\cdots n} \\
X_{11} & X_{111} & \cdots & X_{11n} & X_{1111} & \cdots & X_{11nn} & \cdots & X_{\{11\}\cup\sigma'} & \cdots & X_{11nn\cdots n} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
X_{nn} & X_{nn1} & \cdots & X_{nnn} & X_{nn11} & \cdots & X_{nnnn} & \cdots & X_{\{nn\}\cup\sigma'} & \cdots & X_{nnnn\cdots n} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
X_\sigma & X_{\sigma\cup\{1\}} & \cdots & X_{\sigma\cup\{n\}} & X_{\sigma\cup\{11\}} & \cdots & X_{\sigma\cup\{nn\}} & \cdots & X_{\sigma\cup\sigma'} & \cdots & X_{\sigma\cup\{nn\cdots n\}} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
X_{nn\cdots n} & X_{nn\cdots n1} & \cdots & X_{nn\cdots nn} & X_{nn\cdots n11} & \cdots & X_{nn\cdots nnn} & \cdots & X_{\{nn\cdots n\}\cup\sigma'} & \cdots & X_{nn\cdots nnn\cdots n}
\end{pmatrix}
\end{array}$$

Each row and each column of this matrix corresponds to a monomial of degree at most d (this includes monomials such as $x_1^3 x_2^2 x_3$ that include multiple variables). Noting that the square of any polynomial function of the variables x_i must be nonnegative again gives us that our moment matrix - in this case M_{2d} - is positive semidefinite, which yields a degree-$2d$ SDP.

20.3.1 Pseudoexpectation

As nice as the mathematics are, the notation - the necessity of working with matrices, X_σ, and squared polynomial functions - bogs us down a bit. We now make a notational switch that makes this problem appear cleaner. In the degree 2 case, we keep track of the $n^2 + n$ numbers X_i and X_{ij}. These numbers allow us to compute the expectations of any degree 2 polynomial:

$$\mathbb{E}\left[\sum p_{ij} x_i x_j + \sum q_j x_j + q_0\right] = \sum p_{ij} X_{ij} + \sum q_j X_j + q_0$$

Thus, instead of thinking of our SDP solution as a set of around n^2 different variables, we can think of our SDP solution as a *pseudoexpectation* operator

$$\tilde{E} : \{\text{degree} \leq 2 \text{ polynomial}\} \to \mathbb{R}$$

that maps the polynomial to its moment. That is, it can be thought of as a linear functional that satisfies the constraints

$$\begin{cases} \tilde{E}\left[p_i(x)\right] \geq 0 & \forall p_i(x) \geq 0 \\ \tilde{E}\left[q^2(x)\right] \geq 0 & \forall q \text{ with degree} \leq 1 \end{cases}$$

where the top constraint captures the idea of converting our SDP constraints into constraints on the moments, and the bottom constraint captures the idea that any squared polynomial function should have nonnegative expectation. This notation makes the generalization to higher degrees much more clear: given a polynomial system $\mathcal{P} = \{p_i(x) \geq 0 \mid \forall i\}$, the degree d Sum-Of-Squares (SoS) SDP relaxation finds a linear functional $\tilde{E} : \{\text{degree} \leq 2d \text{ polynomial}\} \to \mathbb{R}$ satisfying the constraints

$$\begin{cases} \tilde{E}\left[p_i(x)\right] \geq 0 & \forall p_i(x) \geq 0 \\ \tilde{E}\left[q^2(x)\right] \geq 0 & \forall q \text{ with degree} \leq d \end{cases}$$

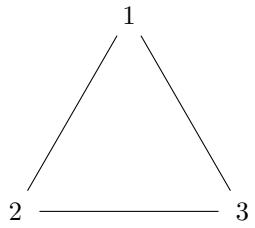
20.4 Hierarchy of Sum-Of-Squares Relaxations

Given any polynomial system, we have a sequence of SDP relaxations that we obtain by taking the order-d problem. These problems become increasingly computationally complex – each problem has n^{2d} variables, whence it takes $n^{O(d)}$ time to solve. If we go up to the degree n problem, we must get the exact solution. Thus, we can think of this collection of problems as belonging to a hierarchy, where as we go higher up, we obtain a better solution, but at the cost of a higher runtime.

We have the luck that many algorithms to which SDP has been applied can be handled by just the bottom two levels of this hierarchy:

$\deg n$ SoS	$\longrightarrow$	$\forall \{0,1\}^n / \{\pm 1\}^n$, exact solution
	$\vdots$	
$\deg 2d$ SoS	$\longrightarrow$	n^{2d} variables, $n^{O(d)}$ time
	$\vdots$	
$\deg 4$ SoS		
$\deg 2$ SoS	$\longleftrightarrow$	0.878 approximation for max cut, for example

Any SDP relaxation gives us a collection of numbers corresponding to the moments of the true distribution over solutions. We now try to make use of this information to actually solve the original problem. We return to the vertex cover problem. Suppose we have the following graph



We obtain the following linear constraints in the degree 1 case:

$$\begin{cases} x_1 + x_2 \geq 1 \\ x_2 + x_3 \geq 1 \\ x_3 + x_1 \geq 1 \end{cases}$$

The deficiencies of this formulation are especially clear when we deal with a triangle, because we can achieve all of the above constraints by setting

$$x_1 = x_2 = x_3 = \frac{1}{2} \qquad x_1 + x_2 + x_3 = \frac{3}{2}$$

when we know that the optimal vertex cover for a triangle requires $2 > \frac{3}{2}$ vertices. One immediate way to make our constraints tighter is to make use of additional structure in the problem. In any triangle, we must select at least 2 vertices, so we can add the constraint

$$x_1 + x_2 + x_3 \geq 2$$

We may similarly note that in a pentagon, we must select at least 3 vertices, whence we have

$$x_1 + x_2 + x_3 + x_4 + x_5 \geq 3$$

This procedure is very ad hoc and doesn't seem like a feasible way to address the structural issues in our degree-2 SDP formulation. As it turns out, these additional constraints nicely fall out when we go higher in the SDP hierarchy.

Theorem 20.4.1. *Suppose $x \in \{0,1\}^n$ - or equivalently, $x_i^2 - x_i = 0 \forall i$ - is among the constraints of our polynomial system. Then, the degree $2d$ SoS SDP captures the "true constraints" on a set of $2d$ variables.*

What we mean by "true constraint" here is a constraint that is true of any integer solution to the SDP (for example, the triangle and pentagon conditions we mentioned above hold for integer solutions).

Proof. We show that for every subset $S \subset \{x_1, \ldots, x_n\}$ where $|S| = 2d$, there exists a probability distribution μ_S over assignments $\{0,1\}^S$ to those variables such that for $x \sim \mu_S$

$$\tilde{E}[x_\sigma] = \mathbb{E}[x_\sigma]$$

for all $\sigma \subseteq S$ (this will show that anything achievable by a solution to the order-d SDP is also achievable by integer solutions, meaning that the SDP satisfies true constraints). That is, knowledge of the moments give an actual local distribution. To see this, we compute the probabilities of this distribution. We have that

$$\mathbf{1}[x_S = \alpha] = \prod_{\substack{i \in S \\ \alpha_i = 0}} (1 - x_i) \prod_{\substack{i \in S \\ \alpha_i = 1}} x_i$$

so we have converted the indicator function of S into a polynomial. Thus, we have that

$$\begin{aligned} \mathbb{P}[x_S = \alpha] &= \mathbb{E}[\mathbf{1}[x_S = \alpha]] \\ &= \tilde{E}\left[\prod_{\substack{i \in S \\ \alpha_i = 0}} (1 - x_i) \prod_{\substack{i \in S \\ \alpha_i = 1}} x_i\right] \end{aligned}$$

This reduces simply to

$$\tilde{E}[x_\sigma] = \mathbb{E}[x_\sigma]$$

where we identify x_S above as being drawn from a distribution μ_S, and the expectation here is over μ_S as well. It thus follows that

$$\sum_{\sigma \subseteq S} c_\sigma x_\sigma \geq 0 \iff \mathbb{E}\left[\sum c_\sigma x_\sigma\right] \geq 0 \iff \tilde{E}\left[\sum c_\sigma x_\sigma\right] \geq 0$$

□

We can see now that this hierarchy of SDP relaxations is in fact quite powerful. The full significance of this theorem may be clearer with a bit more context about program hierarchies, which we discuss now.

The idea of SDP hierarchies came up only after many people had already considered linear program hierarchies. There are a number of different LP hierarchies, such as the Lovasz-Schrijver and Sherali-Adams hierarchies. Unfortunately, for most problems, the LP hierarchy does not appear to help; moving higher in the hierarchy does not improve the solution quality. For instance, in Vertex Cover, a degree 2 LP gives a approximation that cannot be improved upon.

The SoS SDP hierarchy, however, is much more powerful than the previous LP hierarchies. Although we do not know the lower bounds of approximations for higher degree SoS programs for problems such as MaxCut (we have not been able to determine whether or not a degree-4 relaxation improves on the 0.878 constant from before), moving up in the hierarchy provably improves performance on certain statics problems.

As we move forward, we will see how the sum-of-squares SDP hierarchy can be used to devise algorithms for other problems. In some sense, the SoS SDP hierarchy gives us a compiler that turns proofs into algorithms. In this way, it turns the problem of devising an algorithm into the problem of writing a proof.