Semidefinite Programming[1]

- In this lecture, we introduce another "big hammer" in the algorithmic toolbox which has had many successes in the design and analysis of approximation algorithm. This tool is ***semidefinite programming*** which is a generalization of linear programming. The main characters in this story are $n \times n$ *symmetric* matrices $A \in \mathbb{R}^{n\times n}$ which have *non-negative* eigenvalues. Such matrices are called ***positive semidefinite***, or simply PSD matrices, and are denotes as $A \succcurlyeq 0$.

- ***A Linear Algebra Refresher.*** Before moving further, it is a good time to refresh the reader's memory of a few facts about eigenvalues and vectors. Fix a square matrix $A \in \mathbb{R}^{n\times n}$. A non-zero n-dimensional vector $\mathbf{v}$ is an *eigenvector* of A with *eigenvalue* λ if $A\mathbf{v} = \lambda\mathbf{v}$. Geometrically, if one thinks of A as a *linear transform* then the eigenvectors are precisely the ones which get scaled by the action of A.

  Fact 1 (Eigenvalues and Eigenvectors)**.** a. Every $n \times n$ matrix has n eigenvalues and corresponding eigenvectors. These eigenvectors and eigenvalues may be *complex*.

  b. If $A \in \mathbb{R}^{n\times n}$ is symmetric, then all its eigenvalues are real, and therefore can be ordered as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

  c. If $A \in \mathbb{R}^{n\times n}$ is symmetric, then there exists an *orthonormal basis* of eigenvectors. More precisely, there exists eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ such that (a) $A\mathbf{u}_i = \lambda_i\mathbf{u}_i$, (b) $\|\mathbf{u}_i\|_2 = 1$, (c) $\mathbf{u}_i^\top\mathbf{u}_j = 0$ for $i \neq j$, and (d) $\mathbf{u}_i$'s span $\mathbb{R}^n$. Such a basis is called an eigenbasis of A.

  d. (Spectral Decomposition.) If $A \in \mathbb{R}^{n\times n}$ is symmetric with real eigenvalues $\{\lambda_i\}_{i=1,\ldots,n}$ and with a real orthonormal basis $\{\mathbf{u}_i\}_{i=1,\ldots,n}$, then $A = \sum_{i=1}^n \lambda_i\mathbf{u}_i\mathbf{u}_i^\top$.

  e. Given real symmetric A, the λ_i's and $\mathbf{u}_i$'s can be found in polynomial time.

  As noted above, an $n \times n$ symmetric matrix $A \succcurlyeq 0$ is PSD if $\lambda_i \geq 0$. There are various equivalent ways to think of PSD matrices, and we will keep introducing them as needed. The first useful fact is the following. It states that A is PSD if the dot-product of *any* vector and its transform is always non-negative (the angle between them is not obtuse). This dot-product is the *quadratic form* defined by A.

  Fact 2 (Quadratic Forms and PSD matrices)**.** A $n \times n$ symmetric matrix is PSD if and only if $\mathbf{v}^\top A\mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathbb{R}^n$.

  Proof. Consider the vector $\mathbf{v}$ written in the eigenbasis $\{\mathbf{u}_i\}_{i=1,\ldots,n}$ as $\mathbf{v} = \sum_{i=1}^n \alpha_i\mathbf{u}_i$ for some reals $\alpha_i \in \mathbb{R}$. Now observe that $\mathbf{v}^\top A\mathbf{v} = \sum_{i=1}^n \lambda_i\alpha_i^2$. Therefore, if $A \succcurlyeq 0$ then since $\lambda_i \geq 0$, we get $\mathbf{v}^\top A\mathbf{v} \geq 0$. On the other hand, if $A \not\succcurlyeq 0$, then if $\lambda_n < 0$ and $\mathbf{u}_n$ is the corresponding eigenvector, then $\mathbf{u}_n^\top A\mathbf{u}_n = \lambda_n < 0$. □

  The above characterization is very useful. In fact, it allows us to establish an extremely important fact one which leads to the tractability of semidefinite programming (which we have not defined yet).

[1]*Lecture notes by Deeparnab Chakrabarty. Last modified : 21st Feb, 2022*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

Fact 3 (Convexity of the PSD cone)**.** Consider the set $\mathbf{S}_+^n := \{A \in \mathbb{R}^{n\times n} \; : \; A \succcurlyeq 0\}$. This set is a convex set.

Proof. Let A and B two members of $\mathbf{S}_+^n$ and consider $M := \theta A + (1-\theta)B$ for some $\theta \in [0,1]$. We claim that $M \succcurlyeq 0$ as well. To show this, fix $\mathbf{v} \in \mathbb{R}^n$, and due to Fact 2 it suffices to show $\mathbf{v}^\top M\mathbf{v} \geq 0$. However, the LHS is simply $\theta\mathbf{v}^\top A\mathbf{v} + (1-\theta)\mathbf{v}^\top B\mathbf{v}$, which is ≥ 0 again due to Fact 2 since both A and B are PSD. □

- ***Examples of PSD matrices.*** Let's mention some examples of PSD matrices.

  a. *Diagonal matrices $D \geq 0$ with non-negative entries.* Indeed, the eigenvectors of such a matrix are the unit vectors $\mathbf{e}_i$ which has a 1 in the ith coordinate and 0 everywhere else, and the corresponding D_{ii} entry is the eigen value.

  b. *Outer-Product of vector and matrices.* Let $\mathbf{z} \in \mathbb{R}^n$ be any vector. The *outer product* of $\mathbf{z}$ with itself is the $n \times n$ symmetric matrix $\mathbf{z}\mathbf{z}^\top$, whose ijth entry is $\mathbf{z}_i\mathbf{z}_j$. This matrix is PSD. To see this, for any vector $\mathbf{v} \in \mathbb{R}^n$, note $\mathbf{v}^\top \left(\mathbf{z}\mathbf{z}^\top\right)\mathbf{v} = \left(\mathbf{v}^\top\mathbf{z}\right)^2 \geq 0$. PSDness follows from Fact 2. For a more general example, let B be an $n \times m$ matrix. Then the matrix $BB^\top$ is PSD for exactly the same reason as above.

  c. *Moment matrix and Covariance Matrix.* Let $X_1, \ldots, X_n$ be n *random variables* generated by some distribution $\mathcal{D}$. These random variables may not be independent. The $n \times n$ *second moment* matrix induced by these n random variables is

$$M := \begin{bmatrix} \mathbf{Exp}[X_1^2] & \mathbf{Exp}[X_1X_2] & \cdots & \mathbf{Exp}[X_1X_n] \\ \mathbf{Exp}[X_2X_1] & \mathbf{Exp}[X_2^2] & \cdots & \mathbf{Exp}[X_2X_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Exp}[X_nX_1] & \mathbf{Exp}[X_nX_2] & \cdots & \mathbf{Exp}[X_n^2] \end{bmatrix}$$

  This matrix $M \succcurlyeq 0$. The reason is similar to the previous bullet point: for any $\mathbf{v} \in \mathbb{R}^n$ one notices using linearity of expectation that $\mathbf{v}^\top M\mathbf{v} = \mathbf{Exp}[\mathbf{v}^\top\left(\mathbf{x}\mathbf{x}^\top\right)\mathbf{v}]$ where $\mathbf{x} := (X_1, \ldots, X_n)^\top$ is the (random) vector with these random variables stacked one over the other. And thus $\mathbf{v}^\top M\mathbf{v}$ is the expectation of a non-negative random variable.

  d. *Laplacian of a Graph.* Another important example which we may not use in this course but is, I think, important to know is a matrix associated with a graph. Given an undirected simple (no loops or parallel edges) graph $G = (V, E)$ on n vertices and m edges, the Laplacian $\mathcal{L}_G$ is an $n \times n$ symmetric matrix defined as

$$\mathcal{L}_G(u,v) = \begin{cases} \deg(v) & \text{if } u = v \\ -1 & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$

  This matrix $\mathcal{L}_G \succcurlyeq 0$. To see this, observe that $\mathcal{L}_G$ can be written as a sum of m matrices $\{\mathcal{L}_{G,e}\}_{e\in E}$, one for each edge $e = (u,v)$ where $\mathcal{L}_{G,e}(u,u) = \mathcal{L}_{G,e}(v,v) = 1$ and $\mathcal{L}_{G,e}(u,v) = \mathcal{L}_{G,e}(v,u) = -1$. The matrix $\mathcal{L}_{G,e} \succcurlyeq 0$; indeed, for any $\mathbf{x} \in \mathbb{R}^n$ the quadratic form $\mathbf{x}^\top\mathcal{L}_{G,e}\mathbf{x} = (\mathbf{x}_u - \mathbf{x}_v)^2$. And the proof of Fact 3 shows that the sum of PSD matrices is PSD.

- ***Semidefinite Programs.*** We can now state what a semidefinite program (SDP) is. In such a program the set of variables are arranged in the form of an $n \times n$ symmetric matrix $\mathbf{X}$. More precisely, there are $\binom{n}{2}$ variables $\mathbf{X}_{ij} = \mathbf{X}_{ji}$ corresponding to the entries of the matrix $\mathbf{X}$. The objective is a linear function $c(\mathbf{X})$ of these variables. This could be minimized/maximized. There are two kinds of constraints. One is a *linear* constraints on these variables; so these can be of the form $a_i(\mathbf{X}) \leq b_i$. But the most crucial of them all is the PSD constraint: it constraints that $\mathbf{X} \succcurlyeq 0$.

$$\texttt{sdp} := \max \; c(\mathbf{X}) \tag{A General SDP}$$
$$a_i(\mathbf{X}) \leq b_i, \quad 1 \leq i \leq m \tag{1}$$
$$\mathbf{X} \succcurlyeq 0, \tag{2}$$

Oftentimes, one writes the linear functions $c(\mathbf{X})$ and the $a_i(\mathbf{X})$'s as a "matrix dot product". Note that any linear function $c(\mathbf{X})$ is simply $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\mathbf{X}_{ij}$. If one considers the c_{ij}'s as an $n \times n$ matrix $\mathbf{C}$ with $\mathbf{C}_{ij} = c_{ij}$, then this summation is often written as $\mathbf{C} \bullet \mathbf{X}$, but we avoid this notation.

At this point, a reader perhaps wonders *why* would having a PSD constraint (a) make sense, and (b) be helpful. We will address both these questions with an example soon. However, the main theorem which one needs to know is that SDPs are "solvable", and for this course we just assume they can be solved exactly.

Theorem 1. For (A General SDP) there exists a polynomial time algorithm which can return an $(1+\varepsilon)$-approximate solution in time at most a polynomial in n and $\log(1/\varepsilon)$. This works even if the access to (1) is via a separation oracle.

We won't say much about the proof of this except that a variation of the ellipsoid algorithm solves this. As mentioned above, in these lecture notes we will assume they can be solved exactly.

- ***Why SDPs? The Maximum Cut Problem.*** Recall the maximum cut problem. We are given an undirected graph $G = (V, E)$ and every edge e has a non-negative weight $w(e)$. The objective is to find a subset $S \subseteq V$ such that $w(\partial S) = \sum_{e \in \partial S} w(e)$ is maximized. We saw a $\frac{1}{2}$-approximation via local search. We now see how to write an SDP relaxation for the maximum cut problem.

Before we write an SDP relaxation, let us first write an *exact* "quadratic" program for the maximum cut problem. Indeed, here it is. We think of assigning each vertex $v \in V$ a variable $\mathbf{x}_v \in \{-1, +1\}$ with -1 being in S and $+1$ not being in S (we could have used $0, 1$ but $-1, 1$ makes life easier). We ensure that $\mathbf{x}_v$ can indeed take only these values by asserting $\mathbf{x}_v^2 = 1$, the quadratic constraint. Thus, we get

$$\texttt{opt} := \max \; \frac{1}{2} \cdot \sum_{(u,v) \in E} w(u,v) \cdot (1 - \mathbf{x}_u \mathbf{x}_v) \tag{Max Cut QP}$$
$$\mathbf{x}_v^2 = 1, \quad \forall v \in V \tag{3}$$

Now consider rewriting the above as a matrix where $\mathbf{X}_{uv}$ is supposed to capture $\mathbf{x}_u\mathbf{x}_v$. This "lifting" the product of two variables to $\binom{n}{2}$ variables makes both the objective and (6), linear constraints in the entries of $\mathbf{X}$. Of course, we have only transferred the hardness to the constraint that the matrix $\mathbf{X}$

must look like $\mathbf{X}_{uv} = \mathbf{x}_u\mathbf{x}_v$ for all u, v. Or in other words, if we think $\mathbf{x}$ as an n-dimensional vector, $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$.

$$\begin{aligned} \mathtt{opt} := \max \quad & \frac{1}{2} \cdot \sum_{(u,v)\in E} w(u,v) \cdot (1 - \mathbf{X}_{uv}) && \text{(Max Cut QP, restated)} \\ & \mathbf{X}_{vv} = 1, \quad \forall v \in V && (4) \\ & \mathbf{X} \text{ is an outer product } \mathbf{x}\mathbf{x}^\top \text{ for some vector } \mathbf{x} && (5) \end{aligned}$$

It is the constraint (5) that we ***relax*** to a PSD constraint; it is a valid relaxation since outer products are PSD matrices. And so, the SDP relaxation for the maximum cut problem is the following.

$$\begin{aligned} \mathtt{opt} \le \mathsf{sdp} := \max \quad & \frac{1}{2} \cdot \sum_{(u,v)\in E} w(u,v) \cdot (1 - \mathbf{X}_{uv}) && \text{(Max Cut SDP)} \\ & \mathbf{X}_{vv} = 1, \quad \forall v \in V && (6) \\ & \mathbf{X} \succcurlyeq 0, && (7) \end{aligned}$$

- ***How to use an SDP Solution?*** Okay, so given an instance of the maximum cut problem, namely an undirected graph with non-negative weights on edges, using semidefinite programming we can obtain an $n \times n$ PSD matrix $\mathbf{X}$ whose diagonal entries are 1 and $\frac{1}{2}\sum_{(u,v)\in E} w(u,v)(1 - \mathbf{X}_{uv})$ is an upper bound on `opt`. How does it help us in obtaining a cut of comparable value? How does one "round" this, at first glance bizarre object, into a subset of vertices? Time to state another fact about PSD matrices.

Fact 4 (Vector Dot-Product Representation)**.** Let $A \succcurlyeq 0$ be an $n \times n$ matrix. Then one can efficiently find an $r \times n$ real matrix V such that $A = V^\top V$, where $r = \mathsf{rank}(A)$. That is, for any $1 \le i, j \le n$, $A_{ij} = \mathbf{v}_i^\top \mathbf{v}_j$ where $\mathbf{v}_i$'s are the r-dimensional columns of V.

Proof. By the spectral decomposition fact in Fact 1, $A = \sum_{i=1}^n \lambda_i \mathbf{u}_i\mathbf{u}_i^\top$. If A is PSD, then $\lambda_i \ge 0$, and therefore $\sqrt{\lambda_i}$ is a real number. Furthermore, the number of non-zero summands above is precisely $r = \mathsf{rank}(A)$. Therefore, $A = QQ^\top$ where $Q = \sum_{i=1}^r \sqrt{\lambda_i}\mathbf{u}_i$ is an $n \times r$ matrix. The matrix V asserted in the fact is $Q^\top$. □

Coming back to the matrix $\mathbf{X}$ from the SDP solution. Fact 4 gives us n vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ which live in $\mathbb{R}^r$ such that $\mathbf{X}_{ij} = \mathbf{v}_i^\top \mathbf{v}_j$. Note that if $r = 1$, then each $\mathbf{v}_i \in \{-1, +1\}$ and that would indeed point us towards the cut: $S = \{i : \mathbf{v}_i = +1\}$ say. Instead, for each vertex $i \in V$ we get a "high-dimensional" vector $\mathbf{v}_i$ and we need to "project" them down to $\{-1, +1\}$ to obtain our cut. We know $\|\mathbf{v}_i\|_2 = 1$ since $\mathbf{X}_{ii} = 1$. And to rewrite the objective, we know

$$\mathtt{opt} \le \mathsf{sdp} = \frac{1}{2}\sum_{(i,j)\in E} w(i,j) \cdot \left(1 - \mathbf{v}_i^\top \mathbf{v}_j\right)$$

Many SDP "rounding" (maybe they should be called projecting) algorithms follow the above "vector-relaxation" approach.

Notes

Semidefinite Programming is perhaps the most sophisticated tool in the algorithm designer's arsenal. Note that it generalizes linear programming. To see this, note that non-negativity constraints can be cast as a PSD constraint when the variables are on the diagonals. In practice, SDP solvers are still slower than LP solvers, but there is active research in this area. There are many beautiful surveys written on this subject. We point to this one [3] by Lovász for a perspective on applications to CS and math, and to this slightly older one [5] from an optimization perspective. The first and most famous application of SDPs to approximation algorithms is the paper [1] by Goemans and Williamson. More recently, a much deeper connection between SDPs, approximability, and the so-called Unique Games Conjecture have been uncovered in the papers [2, 4] by Khot, Kindler, Mossel, and O'Donnell, and Raghavendra, respectively.

References

[1] M. X. Goemans and D. P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM*, pages 1115–1145, 1995.

[2] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing (SICOMP)*, 37(1):319–357, 2007.

[3] L. Lovász. Semidefinite programs and combinatorial optimization. In *Recent advances in algorithms and combinatorics*, pages 137–194. Springer, 2003.

[4] P. Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 245–254, 2008.

[5] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.