Lecture 13: Semidefinite programming

We define semidefinite programming abstractly – as optimizing a linear function over the PSD cone. Then define the viewpoint as a vector program with constraints being between inner products of the variables.

Disclaimer: These lecture notes are informal in nature and are not thoroughly proofread. In case you find a serious error, please send email to the instructor pointing it out.

Optimization on the PSD cone

In the last class, we defined the PSD cone as a subset of the space of $n \times n$ matrices (i.e., of $\mathbb{R}^{n^2}$), which consists of all matrices $M$ of the form

$$ M = \sum_{i} \alpha_i v_i v_i^T, $$

for some non-negative reals $\alpha_i$, and vectors $v_i \in \mathbb{R}^n$. We also mentioned that this is precisely the set of symmetric matrices with all non-negative eigenvalues. (It is a simple exercise to prove this.)

Such matrices are called positive semidefinite (or PSD) matrices. We also write this as $M \succeq 0$.

An alternate definition of a PSD matrix is that $x^T M x \geq 0$ for all $x \in \mathbb{R}^n$. It is easy to see that this is equivalent to saying that all eigenvalues are non-negative (why? recall the min-max characterization of eigenvalues). We can also write the constraint $x^T M x \geq 0$ as $M \cdot X \succeq 0$, where $X$ is the matrix $xx^T$. Thus, a matrix is PSD iff $M \cdot X \succeq 0$ for all $X = xx^T$, for $x \in \mathbb{R}^n$.

Now, for a given $x$, the constraint $M \cdot X \succeq 0$ is a linear constraint on $M$. Thus the PSD cone can also be defined by the infinite set of linear constraints $M \cdot X \succeq 0$.

The natural question is now, is there a separation oracle? I.e., given an $M$ which is not in PSD, is it easy to find a violated constraint? This is equivalent to asking, can we find an $x \in \mathbb{R}^n$ such that $x^T M x < 0$?

To do this, it suffices to compute $\min_x x^T M x$, which is precisely the problem of finding the bottom eigenvalue of the matrix $M$! This can easily be done efficiently, as we saw earlier. Thus, the separation oracle for the PSD cone is an algorithm that computes the minimum eigenvalue.

This implies that optimizing linear functions over PSD$_n$ (with possibly other linear constraints) can be done efficiently via the ellipsoid algorithm. Such an optimization problem is called a semidefinite program, or an SDP. Formally, an SDP is an optimization problem of the following form:

$$ \max C \cdot X \quad \text{subject to} $$

$$ A_1 \cdot X \leq b_1 $$

$$ A_2 \cdot X \leq b_2 $$

$$ \vdots $$

$$ A_m \cdot X \leq b_m $$

$$ X \succeq 0 $$

As we will see, SDP’s are a powerful class of optimization problems, and can be used to design several very interesting approximation algorithms. Before we go into the details, we make a simple observation that helps us define an SDP in a simpler way.
Vector program formulation

A simple fact is that any PSD matrix $M$ can be written as $M = V^T V$, for some matrix $V$ (this is called the Cholesky decomposition of $M$). To see this, recall that we can write $M = U^T D U$, where $U$ is the matrix with eigenvectors as the columns, and $D$ is the matrix of eigenvalues (which we know are all non-negative); now simply define $V = D^{1/2} U$. Furthermore, any matrix of the form $V^T V$ is PSD.

What this means is that the $(i,j)$th entry $M_{ij} = \langle v_i, v_j \rangle$, where $v_i$ is the $i$th column of $V$. We can rewrite the SDP above as the problem of maximizing $\sum_{i,j} C_{ij} \langle v_i, v_j \rangle$, subject to an arbitrary set of linear constraints on the inner products $\langle v_i, v_j \rangle$. The optimization is over all vectors $v_i$ (technically, without any restriction on the dimension, but we can assume that it is at most $n$, the number of vectors).

This way of viewing an SDP is called the vector programming formulation, which could sometimes give more intuition.

Example: SDP for the Max CUT problem

SDPs have been used to come up with novel approximation algorithms for problems. We see an example for the max cut problem, which we encountered before. We have a graph $G = (V,E)$, and the goal is to divide the vertices into two sets, such that the number of edges going across (from one set to the other) is maximized.

As we have seen, we can write it as the following optimization problem:

$$\max_{x_i \in \pm 1} \frac{1}{4} \sum_{i,j \in E} |x_i - x_j|^2 = \sum_{i,j \in E} \frac{1 - x_i x_j}{2}.$$ 

Now the key idea is, suppose we replaced the condition $x_i \in \pm 1$ with the constraint “$v_i$ is a unit vector”, and replaced the term $x_i x_j$ with $\langle v_i, v_j \rangle$. What would the resultant optimization problem look like? Firstly, it is clearly a relaxation of the original formulation (because if $v_i$ were one-dimensional vectors, we would get the original formulation). Secondly, this is a semidefinite program, because the constraints $\|v_i\| = 1$ can be written as the constraints $\langle v_i, v_i \rangle = 1$, and the objective is already a linear function of $\langle v_i, v_j \rangle$. Thus, this is a relaxation that can be solved efficiently!

The key question is now, does a solution to the relaxation give us a good way solve max cut? I.e., we would like to round the solution to the relaxation, as we did earlier for LP relaxations.

What does it mean to round a solution for the max cut SDP? Recall that a solution is simply a set of unit vectors $v_i$, one for each vertex in the graph. The goal is to come up with an $x_i = \pm 1$, such that $\sum_{i,j \in E} \frac{1 - x_i x_j}{2}$ is not too small compared to $\sum_{i,j \in E} \frac{1 - \langle v_i, v_j \rangle}{2}$.

Abstractly, the question is, how do we map points $v_i$ on a sphere to $x_i \in \pm 1$? We need to divide the surface of the unit sphere (in $n$ dimensions, where $n = |V|$ into two regions, one of which maps to $+1$ and the other to $-1$. One natural way to do it is to break up the surface into two hemispheres (using a hyperplane through the origin).

In the next class, we show that this idea works! In fact, we show that if we pick a random hyperplane through the origin, and assign $x_i$ as above, the expected size of the obtained cut is not too small.