PRINCETON UNIV. F'14 COS 521: ADVANCED ALGORITHM DESIGN

Lecture 15: Semidefinite Programs (SDPs) and Approximation Algorithms

Lecturer: *Sanjeev Arora* Scribe:

Recall that a set of points K is *convex* if for every two $x, y \in K$ the line joining x, y, i.e., $\{\lambda x + (1-\lambda)y : \lambda \in [0,1]\}$ lies entirely inside K. A function $f : \Re^n \to \Re$ is *convex* if $f(\frac{x+y}{2}) \leq \frac{1}{2}(f(x)+f(y))$. It is called *concave* if the previous inequality goes theother way. A linear function is both convex and concave. A *convex program* consists of a convex function f and a convex body K and the goal is to minimize $f(x)$ subject to $x \in K$. Is is a vast generalization of linear programming and like LP, can be solved in polynomial time under fairly general conditions on f, K. Today's lecture is about a special type of convex program called *semidefinite programs.*

Recall that a symmetric $n \times n$ matrix M is *positive semidefinite* (PSD for short) iff it can be written as $M = AA^T$ for some real-valued matrix A (need not be square). It is a simple exercise that this happens iff every eigenvalue is nonnegative. Another equivalent characterization is that there are n vectors $u_1, u_2, \ldots, u_n$ such that $M_{ij} = \langle u_i, u_j \rangle$. Given a PSD matrix M one can compute such n vectors in polynomial time using a procedure called *Cholesky decomposition.*

LEMMA 1
The set of all $n \times n$ PSD matrices is a convex set in $\Re^{n^2}$.

PROOF: It is easily checked that if M_1 and M_2 are PSD then so is $M_1 + M_2$ and hence so is $\frac{1}{2}(M_1 + M_2)$. $\square$

Now we are ready to define semidefinite programs. These are very useful in a variety of optimization settings as well as control theory. We will use them for combinatorial optimization, specifically to compute approximations to some NP-hard problems. In this respect SDPs are more powerful than LPs.

View 1: A linear program in n^2 real valued variables Y_{ij} where $1 \leq i, j \leq n$, with the additional constraint "*Y is a PSD matrix.*"

View 2: A *vector program* where we are seeking n vectors $u_1, u_2, \ldots, u_n \in \Re^n$ such that their inner products $\langle u_i, u_j \rangle$ satisfy some set of linear constraints.

Clearly, these views are equivalent.

Exercise: Show that every LP can be rewritten as a (slightly larger) SDP. The idea is that a diagonal matrix, i.e., a matrix whose offdiagonal entries are 0, is PSD iff the entries are nonnegative.

Question: Can the vectors $u_1, \ldots, u_n$ in View 2 be required to be in $\Re^d$ for $d < n$? Answer: This is not known and imposing such a constraint makes the program nonconvex. (The reason is that the sum of two matrices of rank d can have rank higher than d.)

1 Max Cut

Given an n-vertex graph $G = (V, E)$ find a cut $(S, \overline{S})$ such that you maximise $E(S, \overline{S})$.

The exact characterization of this problem is to find $x_1, x_2, \ldots, x_n \in \{-1, 1\}$ (which thus represent a cut) so as to maximise

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

This works since an edge contributes 1 to the objective iff the endpoints have opposite signs.

The SDP relaxation is to find vectors $u_1, u_2, \ldots, u_n$ such that $|u_i|_2^2 = 1$ for all i and so as to maximise

$$\sum_{\{i,j\} \in E} \frac{1}{4} |v_i - v_j|^2.$$

This is a relaxation since every ± 1 solution to the problem is also a vector solution where every u_i is $\pm v_0$ for some fixed unite vector v_0.

Thus when we solve this SDP we get n vectors, then the value of the objective OPT_{SDP} is at least as large as the capacity of the max cut. How do we get a cut out of these vectors? The following is the simplest rounding one can think of. Pick a random vector z. If $\langle u_i, z \rangle$ is positive, put it in S and otherwise in $\overline{S}$. Note that this is the same as picking a random hyperplane passing through the origin and partitioning the vertices according to which side of the hyperplane they lie on.

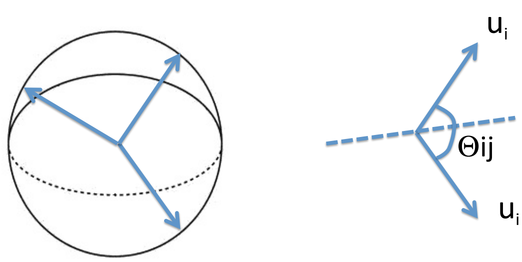

Figure 1: SDP solutions are unit vectors and they are rounded to ± 1 by using a random hyperplane through the origin. The probability that i, j end up on opposite sides of the cut is proportional to Θ_{ij}, the angle between them.

Theorem 2 (Goemans-Williamson'94)
The expected number of edges in the cut produced by this rounding is at least 0.878.. *times* OPT_{SDP}.

Proof: The rounding is essentially picking a random hyperplane through the origin and vertices i, j fall on opposite sides of the cut iff u_i, u_j lie on opposite sides of the hyperplane. Let's estimate the probability they end up on opposite sides. This may seem a difficult n-dimensional calculation, until we realize that there is a 2-dimensional subspace defined by u_i, u_j, and all that matters is the intercept of the random hyperplane with this 2-dimensional subspace, which is a random line in this subspace. Specifically θ_{ij} be the angle between u_i

and u_j. Then the probability that they fall on opposite sides of this random line is θ_{ij}/π. Thus by linearity of expectations,

$$\mathbf{E}[\text{Number of edges in cut}] = \sum_{\{i,j\}\in E} \frac{\theta_{ij}}{\pi}. \tag{1}$$

How do we relate this to OPT_{SDP}? We use the fact that $\langle u_i, u_j \rangle = \cos\theta_{ij}$ to rewrite the objective as

$$\sum_{\{i,j\}\in E} \frac{1}{4}|v_i - v_j|^2 = \sum_{\{i,j\}\in E} \frac{1}{4}(|v_i|^2 + |v_j|^2 - 2\langle v_i, v_j\rangle) = \sum_{\{i,j\}\in E} \frac{1}{2}(1 - \cos\theta_{ij}). \tag{2}$$

This seems hopeless to analyse for us mortals: we know almost nothing about the graph or the set of vectors. Luckily Goemans and Williamson had the presence of mind to verify the following in Matlab: each term of (1) is at least 0.878.. times the corresponding term of (2)! Specifically, Matlab shows that for all

$$\frac{2\theta}{\pi(1-\cos\theta)} \geq 0.878 \qquad \forall \theta \in [0, \pi]. \tag{3}$$

QED □

The saga of 0.878**...** The GW paper came on the heels of the PCP Theorem (1992) which established that there is a constant $\epsilon > 0$ such that $(1+\epsilon)$-approximation to

2 0.878-approximation for MAX-2SAT

We earlier designed approximation algorithms for MAX-2SAT using LP. The SDP relaxation gives much tighter approximation than the 3/4 we achieved back then. Given a 2CNF formula on n variables with m clauses, we can express MAX-2SAT as a quadratic optimization problem. We want $x_i^2 = 1$ for all i (hence x_i is ± 1; where $+1$ corresponds to setting the variable y_i to true) and we can write a quadratic expression for each clause expressing that it is satisfied. For instance if the clause is $y_i \vee y_j$ then the expression is $1 - \frac{1}{4}(1-x_i)(1-x_j)$. It is 1 if either of x_i, x_j is 1 and 0 else.

Representing this expression directly as we did for MAX-CUT is tricky because of the "1" appearing in it. Instead we are going to look for $n+1$ vectors $u_0, u_1, \ldots, u_n$. The first vector u_0 is a dummy vector that stands for "1". If $u_i = u_0$ then we think of this variable being set to True and if $u_i = -u_0$ we think of the variable being set to False. Of course, in general $\langle u_i, u_0\rangle$ need not be ± 1 in the optimum solution.

So the SDP is to find these vectors satisfying $|u_i|^2 = 1$ for all i so as to maximize $\sum_{clausel} v_l$ where v_l is the expression for lth clause. For instance if the clause is $y_i \vee y_j$ then the expression is

$$1 - \frac{1}{4}(u_0 - u_i)(u_0 - u_j) = \frac{1}{4}(1 + u_0 \cdot u_j) + \frac{1}{4}(1 + u_0 \cdot u_i) + \frac{1}{4}(1 - u_i \cdot u_j).$$

This is a very Goemans-Williamson like expression, except we have expressions like $1 + u_0 \cdot u_i$ whereas in MAX-CUT we have $1 - u_i \cdot u_j$. Now we do Goemans-Williamson rounding. The key insight is that since we round to ± 1 each term $1 + u_i \cdot u_j$ becomes 2 with probability $1 - \frac{\theta_{ij}}{\pi} = \frac{\pi - \theta_{ij}}{\pi}$ and is 0 otherwise. Similarly, $1 - u_i \cdot u_j$ becomes 2 with probability θ_{ij}/π and 0 else.

Now the term-by-term analysis used for MAX-CUT works again once we realize that (3) also implies (by substituting $\pi - \theta$ for θ in the expression) that $\frac{2(\pi-\theta)}{\pi(1+\cos\theta)} \geq 0.878$ for $\theta \in [0, \pi]$. We conclude that the expected number of satisfied clauses is at least 0.878 times OPT_{SDP}.