PRINCETON UNIVERSITY F'02 COS 597D: A THEORIST'S TOOLKIT

Lecture 12: Semidefinite Programming

Lecturer: *Sanjeev Arora* Scribe:*Edith Elkind*

1 Introduction

Semidefinite programming (SDP) is an extension of linear programming that can be applied to integer optimization problems even when linear programming itself is of little help.

In this method, in addition to linear constraints, the matrix formed by variables is required to be positive semidefinite. As the feasible solution space becomes smaller, the solution to the semidefinite program might be closer to the solution to the original integer problem, as compared to the corresponding LP solution.

Since SDP is a special case of convex programming, a semidefinite program can be solved to an arbitrary precision in polynomial time using the ellipsoid method or a version of Karmarkar's algorithm.

2 Basic definitions

DEFINITION 1 *A symmetric matrix $M \in \mathbb{R}^{n\times n}$ is* positive semidefinite *(note: this is often written as $A \succcurlyeq 0$) if any of the following conditions holds:*

(i) M can be represented as AA^T for some $A \in \mathbb{R}^{n\times n}$;

(ii) M has no negative eigenvalues;

(iii) $x^T Mx \geq 0$ for any $x \in \mathbb{R}^n$;

(iv) $\exists \mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ such that $M_{ij} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle$.

THEOREM 1
Conditions (i)–(iv) are equivalent.

PROOF: An easy exercise. □

LEMMA 2
The set $\{M \in \mathbb{R}^{n\times n} \mid M$ is a symmetric positive semidefinite matrix $\}$ is a convex cone.

PROOF: From (ii), it follows that if M is positive semidefinite, then so is λM for any $\lambda > 0$. From (iii), it follows that if M_1, M_2 are positive semidefinite, then so is $M_1 + M_2$. Obviously, symmetry is preserved, too. □

DEFINITION 2 *A* semidefinite program *is formulated as follows:*

$$\begin{aligned} \min & C \cdot Y \\ & A_1 \cdot Y \geq b_1, \\ & \ldots \\ & A_m \cdot Y \geq b_m, \\ & Y \succcurlyeq 0 \end{aligned}$$

where Y is an $n \cdot n$-matrix of variables, $C, A_1, \ldots, A_m \in \mathbb{R}^{n\cdot n}$, $b_1, \ldots, b_m \in \mathbb{R}$ and $X \cdot Y$ is interpreted as $\sum_{i,j=1}^{n} X_{ij} Y_{ij}$.

Such a convex program can be solved to an arbitrary precision (modulo some technical requirements we do not go into) using the Ellipsoid method. As mentioned earlier this method can solve a linear program to an arbitrary degree of precision in polynomial time even if the number of constraints is non-polynomial (or infinite), so long as there is a polynomial-time separation oracle for the set of constraints. In case of the above convex program, we can replace the "$Y \succcurlyeq 0$" by the infinite family of linear constraints $x^T Y x \geq 0$ for every $x \in \Re^n$. Furthermore, given any Y that is not psd, there is a simple algorithm (derived from the Cholesky decomposition) that gives an $x \in \Re^n$ such that $x^T Y x < 0$; this is the separation oracle.

3 An SDP for Graph Expansion

Last time, we wrote a linear program for Graph Expansion. Given a graph $G = (V, E)$, for every $i, j \in V$, we introduce a variable x_{ij} and consider the following LP:

$$\begin{aligned} \min & \frac{\sum_{\{i,j\} \in E} x_{ij}}{\sum_{i<j} x_{ij}} \\ x_{ij} + x_{jk} \leq x_{ik} & \text{ for all } i, j, k \text{ (triangle inequality)} \\ x_{ij} \geq 0 & \\ x_{ij} = x_{ji} & \end{aligned}$$

In an effort to tighten the relaxation, we require the following constraint in addition:

$$\exists\, v_1, v_2, \ldots, v_n \in \Re^n \text{ such that } x_{ij} = (v_i - v_j) \cdot (v_i - v_j).$$

We show that this constraint is not an unfair one, by shoing that the optimum integer solution (and, more generally, any cut metric) satisfies it: simply pick an arbitrary vector $\mathbf{u}$ and set $\mathbf{v}_i = \mathbf{u}$ if $v_i \in S$ and $\mathbf{v}_i = -\mathbf{u}$ if $v_i \in \bar{S}$, so the optimum integer solution is still feasible.

How can we optimize under this constraint using SDP? Consider the matrix M where $M_{ii} = 1$ and $M_{ij} = 1 - \frac{x_{ij}}{2}$ if $i \neq j$. The constraint amounts to saying this matrix to be psd (see Theorem 1 part (iv)).

We leave it as an exercise to show that this SDP provides a lower bound that is at least as good as the eigenvalue bound for d-regular graphs and no worse than Leighton–Rao bound for the general case. Thus this approach combines the best of both worlds.

It is conjectured that the linearity gap for this program is $O(1)$ (it is known to be $O(\log n)$). This would follow from another conjecture (by Goemans and Linial), namely, that any metric $d(i, j)$ that can be represented as $\|\mathbf{v}_i - \mathbf{v}_j\|^2$ for some $\{\mathbf{v}_i\}_{i=1}^n$, can be embedded into l^1 with $O(1)$ distortion.

4 0.878-Approximation for Max Cut

Let us present the celebrated 0.878 approximation algorithm for Max Cut by Goemans and Williamson (1993). Here the goal is to partition the vertices of a graph $G = (V, E)$ into two sets S and $\bar{S}$ so as to *maximize* $E(S, \bar{S})$.

The setup is almost identical to the previous case; the only difference is that now we seek to maximize $\sum_{\{i,j\} \in E} x_{ij} = \sum_{\{i,j\} \in E} 1/4 \|\mathbf{v}_i - \mathbf{v}_j\|^2$, where $\|\mathbf{v}_i\| = 1$, $i = 1, \ldots, n$. (The factor $1/4$ is needed for renormalization: if $\mathbf{v}_i = -\mathbf{v}_j$, then $\|\mathbf{v}_i - \mathbf{v}_j\|^2 = 4$.)

Denote by OPT_f the optimal fractional solution; the corresponding vectors are $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Note that for any pair of vectors we can draw a 2-dimensional plane through them, so

$$\|\mathbf{v}_i - \mathbf{v}_j\|^2 = \langle \mathbf{v}_i - \mathbf{v}_j, \mathbf{v}_i - \mathbf{v}_j \rangle = 2 - 2\cos\theta_{ij}, \tag{1}$$

where θ_{ij} is the angle between these two vectors in the plane.

This suggests the following algorithm: pick a random hyperplane H passing through the origin. It divides the endpoints of $\{\mathbf{v}_i\}_{i=1}^n$ into two sets; assign the vertices of the graph to S or $\bar{S}$ accordingly.

For any $\{i,j\} \in E$, the probability that $\{i,j\}$ crosses the cut is θ_{ij}/π, so the expected value of $E(S,\bar{S})/\mathrm{OPT}_f$ is

$$\frac{4\sum_{\{i,j\}\in E}\theta_{ij}}{\pi \times 2\sum_{\{i,j\}\in E}(1-\cos\theta_{ij})} \geq \min \frac{\theta_{ij}}{1-\cos\theta_{ij}} \times \frac{2}{\pi} = 0.878 \tag{2}$$

Hence, this algorithm gives a 0.878-approximation.

5 Spectral Partitioning of Random Graphs

This part is based on a FOCS 2001 paper by Frank McSherry, which considers the task of finding a solution to an NP-hard graph problem when the input graph is drawn from a distribution over acceptable graphs.

For example, suppose that a graph on n vertices is constructed as follows: include each edge with probability p; then pick k vertices out of n (uniformly over all sets of size k), and complete the clique on these vertices.

Given a graph constructed according to this procedure, can we find the planted clique with high probability (over the coin tosses of the generating algorithm as well as our own random choices)?

More generally, consider the following setup: for a graph on n vertices, there is a mapping $\psi : [n] \mapsto [k]$ that partitions the vertices into k classes, and a matrix $P_{k\times k}$, $0 \leq P_{ij} \leq 1$. To generate a random instance $\hat{G}$ on a vertex set V, $|V| = n$, include an edge between v_i and v_j with probability $P_{\psi(i)\psi(j)}$. We would like to construct an algorithm that given $\hat{G}$ can reconstruct ψ with high probability. (Usually, after that it is possible to (approximately) reconstruct P).

Note that for the clique problem described above, $k = 2$, $P_{00} = 1$, $P_{01} = P_{10} = P_{11} = p$ and the planted clique can be described as $\psi^{-1}(0)$.

Denote the adjacency matrix of $\hat{G}$ by $\hat{M}$. Note that $M = \mathbf{E}\hat{M}$ gives us the corresponding edge probabilities. Hence, for any two vertices that are in the same class with respect to ψ, the corresponding columns are identical, so the rank of M is at most k. Obviously, if we knew M, we could find ψ by grouping the columns; since $\hat{M}$ is obtained from M by randomized rounding, let us try to use clustering.

Let E be the error matrix, $E = \hat{M} - M$. Suppose that P projects the space spanned by the columns of $\hat{M}$ onto k dimensions. If both $\|P(E)\|$ and $\|P(M) - M\|$ are small, then by triangle inequality, $P(\hat{M}) \approx M$. Now, our goal is to find a suitable P.

Since M is a low-rank symmetric matrix, we can use singular value decomposition (SVD).

It is known that an $m \times n$-matrix A of rank r can be represented as $U\Sigma V^T$, where U is an $m \times r$ orthogonal matrix, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, and V is an $n \times r$ orthogonal matrix.

This decomposition is unique up to a permutation of σ_i's (and respective permutations of rows and columns of U and V), so we can assume $\sigma_1 \geq \ldots \geq \sigma_r$; for a symmetric A, σ_i's are the eigenvalues.

By truncating U to k columns, Σ to k values, and V^T to k rows, we obtain a matrix A_k, which is known to be the best rank k approximation of A in the spectral norm. The columns of this matrix are projections of the corresponding columns of A onto the first k eigenvectors of A. In what follows, we denote the projection $X \mapsto X_k$ by P_{X_k}.

Instead of applying SVD directly, we will use a minor modification of this approach. Namely, consider the following algorithm **Partition**(τ):

- Randomly divide $[n]$ into two parts; let us say that this division splits the columns of $\hat{G}$ as $[\hat{A} \mid \hat{B}]$.
- Compute $P_1 = P_{B_k}$, $P_2 = P_{A_k}$.
- Let $H = [P_1(\hat{A})|P_2(\hat{B})]$.
- While there are unclustered nodes, choose one of them (say, u) and put all unclustered vs such that $|\hat{H_u} - \hat{H_v}| < \tau$ into u's cluster; repaet.
- Return the resulting partition.

Here τ is a parameter (intercluster distance). Intuitively, since our partition is random, any class is likely to be split evenly between A and B, so the projection computed for A should work for B and vice versa. By splitting the matrix into two parts we avoid some tricky interdependency issues.

Suppose that for all u we have

$$|P_1(A_u) - A_u| \le \gamma_1 \text{ and } |P_1(A_u - \hat{A_u})| \le \gamma_2$$
$$|P_2(B_u) - B_u| \le \gamma_1 \text{ and } |P_2(B_u - \hat{B_u})| \le \gamma_2$$

and $|G_u - G_v| \ge 4(\gamma_1 + \gamma_2)$ whenever $\psi(u) \ne \psi(v)$. Then our algorithm finds ψ correctly if $\tau = 2(\gamma_1 + \gamma_2)$. To see that, suppose that u and v ended up in the same cluster. As $|G_u - \hat{H_u}| \le \gamma_1 + \gamma_2$, $|G_v - \hat{H_v}| \le \gamma_1 + \gamma_2$, and $|\hat{H_u} - \hat{H_v}| < 2(\gamma_1 + \gamma_2)$, it must be that $G_u = G_v$. Conversely, if $G_u = G_v$, $\hat{H_u}$ and $\hat{H_v}$ cannot be too far apart.

To make use of this algorithm, we need a bound on γ_1 and γ_2.

THEOREM 3
With probability at least $1 - \delta$,

$$|P_{\hat{A}}(B_u) - B_u| \le 8\sigma\sqrt{nk/s_u}$$
$$|P_{\hat{A}}(B_u - \hat{B_u})| \le \sqrt{2k\log(n/\delta)},$$

where σ *is the largest deviation of an entry in* $\hat{A}$ *and* s_u *is the size of the part containing* u*, i.e.,* $s_u = |\{v \mid \psi(v) = \psi(u)\}|$.

For the proof, the reader is referred to the paper.