E0 249: Approximation Algorithms **Spring 2015**

Lecture 11: March 20

Lecturer: Arnab Bhattacharyya *Scribe: Amleshwar Kumar and Nitin Singh*

11.1 Preliminaries

An $n \times n$ real matrix M is called *symmetric* if $M_{ij} = M_{ji}$ for all $i, j \in [n]$. Let SYM_n denote the set of all $n \times n$ real symmetric matrices. For matrices $A, X \in \text{SYM}_n$, we define the product $A \cdot X$ to be $\sum_{i,j \in [n]} a_{ij} x_{ij}$, where a_{ij} and x_{ij} denote the $(i,j)^{th}$ entries of A and X respectively. A real symmetric matrix X is called *positive semidefinite* if all its eigen values are non-negative (note that all eigen values are real in any case). From the standard linear algebra (which we will not go into here), we have the following equivalent characterizations of positive semidefinite matrices:

Proposition 11.1. *Let X be a real symmetric $n \times n$ matrix. Then the following are equivalent:*

(i) All eigen values of X are non-negative.

(ii) For all $a \in \mathbb{R}^n$, $a^T X a \geq 0$.

(iii) $X = U^T U$ for some $n \times n$ matrix U.

A semidefinite program is an optimization problem of the form:

$$\begin{array}{lrcl} \textbf{Maximize:} & C.X & & \\ \text{s.t.} & A_1 \cdot X & = & b_1 \\ & A_2 \cdot X & = & b_2 \\ & \vdots & & \\ & A_n \cdot X & = & b_n \\ & X & \succeq & 0. \end{array}$$

where $C, A_1, \ldots, A_n$ are real symmetric matrices and $b_1, \ldots, b_n$ are real numbers. The constraint $X \succeq 0$ in the above denotes that X is positive semidefinite matrix.

11.2 MAX-CUT Problem

In this section we illustrate an application of semidefinite programming to obtain an approximation algorithm for the MAX-CUT problem. Following is the MAX-CUT problem.

MAX-CUT: Given a graph $G = (V, E)$, find $S \subseteq V$ such that the cardinality of the set $\{e : e = (u, v) \text{ with } u \in S, v \notin S\}$ is maximized.

As a first approach we consider the following program:

$MC1$:

$$\begin{array}{ll} \max & \displaystyle\sum_{(i,j) \in E} \frac{1 - x_i x_j}{2} \\ \text{s.t} & x_i \in \{\pm 1\} \ \forall i \in [n]. \end{array}$$

Algorithm 1 MAX-CUT

Require: Graph $G = (V, E)$.
Solve $SDP\text{-}A$ for vectors $u_1, \ldots, u_n$.
Infer set S from vectors $u_1, \ldots, u_n$ using GW-rounding.
return S.

The solution to the above corresponds to the set $S = \{i : x_i = 1\}$. It can be seen through this correspondence that $OPT \leq MC1$. Below we consider a semidefinite program, where instead of reals x_i we use vectors $u_i \in \mathbb{R}^n$.

$SDP\text{-}A$:

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} \frac{1 - u_i \cdot u_j}{2} \\ \text{s.t} \quad & u_i \cdot u_i = 1 \ \forall i \in [n]. \end{aligned}$$

Before we describe how a solution to $SDP\text{-}A$ corresponds to a solution to MAX-CUT, let us observe that $OPT \leq SDP\text{-}A$. Why ? (Hint: $MC1 \leq SDP\text{-}A$ via embedding scalars into vectors in $\mathbb{R}^n$). We call the vector program $SDP\text{-}A$ to be the *vector relaxation* of the program $MC1$.

From Proposition 11.1, it follows that $SDP\text{-}A$ is equivalent to the following semidefinite program:

$SDP\text{-}B$:

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} \frac{1 - y_{ij}}{2} \\ \text{s.t} \quad & y_{ii} = 1 \ \forall i \in [n] \\ & Y \succeq 0. \end{aligned}$$

Once we have a solution to $SDP\text{-}B$, we can obtain a solution to $SDP\text{-}A$ by determining U such that $Y = U^T U$ (Cholesky's decomposition [1]). We still have not specified how to obtain a cut from a solution to $SDP\text{-}A$. This is accomplished by a procedure called GW-rounding, introduced by Goemans and Williamson.

GW-rounding:

(i) Pick a random vector $s \in \mathbb{R}^n$ with $|s| = 1$.

(ii) Put i in S if $u_i.s \geq 0$.

(iii) Let $x_i = 1$ if $u_i.s \geq 0$, and $x_i = -1$ if $u_i.s < 0$.

Algorithm 1 contains the complete algorithm for MAX-CUT.

Analysis: Clearly, the size of the cut returned by the above algorithm is equal to the number of pairs $(i, j) : i < j$ such that $x_i \neq x_j$. Let p_{ij} denote the probability that $x_i \neq x_j$ for $i < j$. Note that the randomness comes from randomly choosing the reference vector s (It is worth pondering, how do we randomly choose a unit vector in $\mathbb{R}^n$). Let $C(s)$ denote the size of cut obtained for a choice of s. Then we have,

$$E[C(s)] = \sum_{(i,j)\in E} p_{ij} \tag{11.1}$$

[1] Note that since U may have irrational entries, we may not get $Y = U^T U$ exactly. But we can get arbitrarily accurate, and the error is absorbed in the approximation factor.

Now let α be the angle between vectors u_i and u_j. When do $u_i.s$ and $u_j.s$ have different signs ? Since $u_i \cdot s = u_i \cdot \hat{s}$ and $u_j \cdot s = u_j \cdot \hat{s}$ where $\hat{s}$ is the projection of s onto the plane determined by u_i and u_j, we essentially need the probability that for a random vector v in the plane of u_i and u_j, the dot products $u_i \cdot v$ and $u_j \cdot v$ have different signs. We leave it to the reader to show that it is α/π where $\cos\alpha = u_i.u_j$. Then from (11.1) we have,

$$\begin{aligned} E[C(s)] &= \sum_{(i,j)\in E} \frac{\cos^{-1}(u_i \cdot u_j)}{\pi} \\ &\geq 0.878567 \sum_{(i,j)\in E} \frac{1 - u_i \cdot u_j}{2} \\ &\geq 0.878567 \cdot SDP\text{-}A \\ &\geq 0.878567 \cdot OPT \end{aligned}$$

In the above, we have used the fact that $\dfrac{2\cos^{-1} z}{\pi(1-z)} \geq 0.878567$ for $z \in [-1, 1]$.

11.3 How do we solve SDPs

In this section we look at the question of solving SDPs. One way is to use the following property of the ellipsoid algorithm.

Given a convex set C contained in a ball of radius R, and a polynomial time seperation oracle (to decide if a solution is feasible, or produce a certificate of infeasibility), ellipsoid algorithm minimizes any linear function over C. *(*)*

We need to verify two things:

(i) Is the set of all feasible solutions to an SDP convex ?

(ii) Is there a polynomial time seperation oracle to check feasibility of a solution ?

Convexity of the set of feasible solution follows from Proposition 11.1. Let us look at the question of having a polynomial time seperation oracle. If a solution X is infeasible, one of the following must happen.

(i) X is not symmetric. In this case there exist i, j such that $X_{ij} > X_{ji}$. Then the seperating hyperplane is $Y_{ij} \leq Y_{ji}$.

(ii) X is not positive semidefinite. In this case, there is a negative eigen value, say λ. Let a be the corresponding eigen vector. Then we see that $a^T X a < 0$. The seperating hyperplane is then $a^T y a \geq 0$.

(iii) X violates one of the constraints. In this case, the constraint itself yields the seperating hyperplane.

One also needs to check that the feasible solutions of the SDP have bounded Frobenius norm (ℓ_2 norm of the matrix viewed as an n^2-length vector). This is not true for arbitrary polynomial size SDP's, but for most applications, such as MAX-CUT above, it is true, because each entry of a feasible solution matrix corresponds to the inner product between bounded length vectors.

11.4 MAX-2SAT Problem

The following is the MAX-2SAT problem: Given a boolean clause $C = X_1 \wedge X_2 \wedge \cdots \wedge X_k$ where each X_i is an 'or' of atmost two variables. The goal is to find a $\{T, F\}$ assignment of variables which maximizes the number of clauses X_i evaluating to T.

We consider the following SDP formulation to the above. Let $v_1, \ldots, v_n$ be the variables appearing in the clauses. Let $y_0 \in \{\pm 1\}$ be a reference variable, and $y_1, \ldots, y_n \in \{\pm 1\}$ be "SDP" variables corresponding to boolean variables $v_1, \ldots, v_n$. We infer v_i to be true when $y_i = y_0$ and false otherwise. Define:

$$\tau(v_i) := \frac{1 + y_0 y_i}{2} \text{ and } \tau(\bar{v}_i) := \frac{1 - y_0 y_i}{2}. \tag{11.2}$$

Notice that for single variable clause V, $\tau(V) = 1$ when V is true and $\tau(V) = 0$ when V is false. Similarly for a two variable clause $v_i \vee v_j$, define:

$$\begin{aligned}
\tau(v_i \vee v_j) &:= 1 - \tau(\bar{v}_i)\tau(\bar{v}_j) \\
&:= 1 - \frac{1 - y_0 y_i}{2} \cdot \frac{1 - y_0 y_j}{2} \\
&:= \frac{1 + y_0 y_i}{4} + \frac{1 + y_0 y_j}{4} - \frac{1 - y_i y_j}{4}.
\end{aligned}$$

In the above, we use the fact that $y_0^2 = 1$. Again, $\tau(v_i \vee v_j)$ is 1 or 0 according to whether $v_i \vee v_j$ is true or false. We can come up with similar expressions for other two variable clauses, in terms of $(1 - y_i y_j)$ or $(1 + y_i y_j)$. Thus the number of satisfied clauses is given by $\sum_{i=1}^{k} \tau(X_i)$. We thus have the following SDP:

$$\begin{aligned}
\max \quad & \sum_{0 \le i < j \le n} a_{ij}(1 + y_i y_j) + b_{ij}(1 - y_i y_j) \\
\text{s.t} \quad & y_i \in \{\pm 1\} \; \forall \; 0 \le i \le n.
\end{aligned}$$

Again, relaxing the above program to a vector program, and using the same algorithm as MAX-CUT, we get a 0.87856-approximate algorithm for MAX-2SAT (exercise).