CMPUT 675: Approximation Algorithms **Fall 2014**

Lecture 20 (Oct 22): Semidefinite Programming and Max Cut

Lecturer: Zachary Friggstad *Scribe: Zachary Friggstad*

20.1 Semidefinite Programming

Semidefinite programming is a generalization of linear programming that we can still effectively solve in polynomial time. First, we review a key concept.

Definition 1 *A matrix $X \in \mathbb{R}^{n \times n}$ is* **positive semidefinite** *if for every vector $\mathbf{v} \in \mathbb{R}^n$ we have $\mathbf{v}^T \cdot \mathbf{X} \cdot \mathbf{v} \geq 0$. We write $\mathbf{X} \succeq 0$ to indicate X is positive semidefinite.*

For symmetric $\mathbf{X} \succeq 0$, there are many equivalent ways to define this. We mention just a couple below.

Theorem 1 *Let $\mathbf{X} \in \mathbb{R}^{n \times n}$ be a symmetric matrix. Then the following statements are equivalent:*

- $\mathbf{X} \succeq 0$ *(according to the above definition).*
- *Every eigenvalue λ of $\mathbf{X}$ is satisfies $\lambda \geq 0$ (recall any symmetric matrix has only real-valued eigenvalues).*
- *There are vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ that satisfy $v_i \circ v_j = X_{i,j}$ where $\circ$ denotes the dot product.*

Finally, for two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$, the *Schur product* of $\mathbf{A}$ and $\mathbf{B}$ is $\mathbf{A} \circ \mathbf{B} = \sum_{i,j} A_{i,j} \cdot B_{i,j}$.

A *semidefinite program* (SDP) consists of a dimension n, matrices $\mathbf{C}, \mathbf{A_1}, \ldots, \mathbf{A_k} \in \mathbb{R}^{n \times n}$ and values $b_1, \ldots, b_k \in \mathbb{R}$. It is the mathematical program

$$\begin{array}{rrcll} \text{minimize :} & \mathbf{C} \circ \mathbf{X} & & & \\ \text{subject to :} & \mathbf{A_i} \circ \mathbf{X} & = & b_i & \text{for each } 1 \leq i \leq k \\ & \mathbf{X} & = & \mathbf{X}^T & \\ & \mathbf{X} & \succeq & 0 & \end{array} \tag{SDP}$$

As with linear programming, we can instead maximize the objective function and/or have linear inequalities like $\mathbf{A_i} \circ \mathbf{X} \leq b_i$. Of course, for the sake of computability we assume all $\mathbf{C}, \mathbf{A_i}, b_i$ values are rational. Similar to how we denote the optimal value of an LP, we will use OPT_{SDP} to denote the optimum solution value of a given semidefinite program.

One barrier to solving SDPs exactly is that even if the input data consists only of integers it might be that the unique optimum is in fact irrational. An example of this is in the next lecture (the integrality gap example for Max Cut).

The following summarizes what we can expect from polynomial-time solvers for semidefinite programming. We let Δ denote the "size" of the input: the number of bits used to record $\mathbf{C}$ and each of the $\mathbf{A_i}$ and b_i constraints.

Theorem 2 *Under certain "niceness" assumptions (that will be satisfied by the SDPs in this course), for any $\epsilon > 0$ we can find a feasible solution $\overline{\mathbf{X}}$ in* poly $\left(\Delta, \log \frac{1}{\epsilon}\right)$ *time that satisfies $\mathbf{C} \circ \overline{\mathbf{X}} \leq OPT_{SDP} + \epsilon$. Furthermore, we can also find vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in* poly $\left(\Delta, \log \frac{1}{\epsilon}\right)$ *time where $|\mathbf{v}_i \circ \mathbf{v}_j - \overline{X}_{i,j}| \leq \epsilon$.*

From now on, we will sweep this ϵ under the rug and simply assume that we can solve the SDPs optimally and obtain an exact vector decomposition.

20.2 Quadratic Programming and Vector Programming

A *(Strict) Quadratic Program* is presented much like a linear program, except that each term is either a constant or the product of two variables. Generally, it is presented as follows where $v_1, \ldots, v_n$ are the $\mathbb{R}$-valued variables:

$$\begin{array}{lrcll} \text{minimize}: & \sum_{i,j} c_{i,j} \cdot v_i \cdot v_j & & & \\ \text{subject}_{\text{t}}\text{o}: & \sum_{i,j} a^k_{i,j} \cdot v_i \cdot v_j & = & b_k & \text{for each } k \\ & v_i & \in & \mathbb{R} & \text{for each } i \end{array} \tag{QP}$$

Here, k indexes the various constraints.

Solving quadratic programs, even up to just a constant $\epsilon > 0$, is NP-hard. However, if we substitute each real-valued variable v_i with a vector-valued variable $\mathbf{v}_i \in \mathbb{R}^n$ and the multiplication of variables with the dot product, we obtain a *Vector Programming* relaxation.

$$\begin{array}{lrcll} \text{minimize}: & \sum_{i,j} c_{i,j} \cdot \mathbf{v}_i \circ \mathbf{v}_j & & & \\ \text{subject}_{\text{t}}\text{o}: & \sum_{i,j} a^k_{i,j} \cdot \mathbf{v}_i \circ \mathbf{v}_j & = & b_k & \text{for each } k \\ & \mathbf{v}_i & \in & \mathbb{R}^n & \text{for each } i \end{array} \tag{VP}$$

The claim is that this is just another view of semidefinite programming. For example, we could replace each $\mathbf{v}_i \circ \mathbf{v}_j$ with the variable $X_{i,j}$ and then assert that $\mathbf{X} = \mathbf{X}^T$ and $\mathbf{X} \succeq 0$ to get an SDP. This process reverses, given an SDP we can replace each occurrence of $X_{i,j}$ with $\mathbf{v}_i \circ \mathbf{v}_j$. Given the equivalence (for symmetric $\mathbf{X}$) of $\mathbf{X} \succeq 0$ and $\mathbf{X}$ having a vector decomposition, these operations produce equivalent mathematical programs.

The point is, we often write a semidefinite program as a vector program with the understanding that it can be solved using algorithms for solving SDPs. In this way, we can regard SDPs as being useful for finding efficiently solvable relaxations of quadratic programs.

20.3 Max Cut

In the Max Cut problem, we have a graph $G = (V, E)$ with edge weights $w(e) \geq 0, e \in E$. The goal is to find a maximum weight cut: a set $S \subseteq V$ with maximum possible $\sum_{(i,j)\in\delta(S)} w(i,j)$.

An easy extension of a problem in Assignment 0 shows that it is always possible to find a cut with at least half of the *total* edge weight. In fact, adding each $v \in V$ to S independently and uniformly at random results in the expected weight of $\delta(S)$ being exactly half of the total edge weight. These are 1/2-approximations. Here, we will see a much better approximation based on semidefinite programming.

We start with a quadratic program.

$$\begin{array}{lrcll} \text{maximize}: & \displaystyle\sum_{(i,j)\in E} w(i,j) \cdot \frac{1 - v_i \cdot v_j}{2} & & & \\ \text{subject to}: & v_i \cdot v_i & = & 1 & \text{for each } i \in V \\ & v_i & \in & \mathbb{R} & \text{for each } i \in V \end{array}$$

Here, we let $v_i = 1$ indicate $i \in S$ and $v_i = -1$ indicate $i \notin S$. This way, in any feasible solution the term $\frac{1-v_i \cdot v_j}{2}$ in the objective function is 1 if i and j lie on different sides of the cut and is 0 if i and j lie on the same side of the cut.

The vector programming/SDP relaxation is

$$\begin{aligned} \text{maximize :} \quad & \sum_{(i,j)\in E} w(i,j) \cdot \frac{1 - \mathbf{v}_i \cdot \mathbf{v}_j}{2} \\ \text{subject to :} \quad & \mathbf{v}_i \cdot \mathbf{v}_i = 1 \quad \text{for each } i \in V \\ & \mathbf{v}_i \in \mathbb{R}^n \quad \text{for each } i \in V \end{aligned} \qquad \textbf{(SDP-Max-Cut)}$$

Let $\alpha := \min_{0 \le \theta \le \pi} \frac{2}{\pi} \cdot \frac{\theta}{1-\cos(\theta)} > 0.87856$. The numerical estimate can be obtained by asking Wolfram Alpha :)

```
minimize 2/Pi * t/(1-cos(t)) for t in [0,pi]
```

Theorem 3 *The integrality gap of (***SDP-Max-Cut***) is at least α. Moreover, there is a polynomial-time randomized algorithm that finds a cut with expected weight at least $\alpha \cdot OPT_{SDP}$.*

Proof. Let $\mathbf{v}_i^*$ denote the vectors in an optimal solution to (**SDP-Max-Cut**). Sample a random unit vector $\mathbf{r}$ (i.e. $\mathbf{r}$ is uniformly chosen from the set $\{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x}||_2 = 1\}$). See the comments at the end of this set of notes for some details on how to do this.

Let $S = \{i \in V : \mathbf{r} \circ \mathbf{v} \ge 0\}$. We show $\Pr[(i,j) \in \delta(S)] \ge \alpha \cdot \frac{1-\mathbf{v}_i^* \circ \mathbf{v}_j^*}{2}$. As usual, this implies by linearity of expectation that the expected weight of $\delta(S)$ is at least $\alpha \cdot OPT_{\text{SDP}}$.

If $\mathbf{v_i^*}$ and $\mathbf{v_j^*}$ lie on a line through the origin then $\mathbf{v_i^*} \circ \mathbf{v_j^*} = \pm 1$ because they are unit vectors. In this case, $\Pr[(i,j) \in \delta(S)] = \frac{1-\mathbf{v_i^*} \circ \mathbf{v_j^*}}{2}$. So, suppose they do not line on a line and consider the 2-dimensional plane that passes through $\mathbf{v_i^*}$, $\mathbf{v_j^*}$ and the origin.

We claim that $\Pr[(i,j) \in \delta(S)]$ remains unchanged whether we sample $\mathbf{r}$ uniformly at random from all unit vectors in $\mathbb{R}^n$ or uniformly at random from all unit vectors on this plane. Intuitively, there is no reason that the uniform distribution over unit vectors in $\mathbb{R}^n$ should have any bias to some particular region on the unit circle in this plane. This can be proven formally, but we skip it.

So, we must show that if $\mathbf{v_i^*}, \mathbf{v_j^*}$ lie on the plane, then $\Pr[(i,j) \in \delta(S)] \ge \alpha \frac{1-\mathbf{v_i^*} \circ \mathbf{v_j^*}}{2}$ when we sample $\mathbf{r}$ uniformly at random from the unit circle. Let $\theta_{i,j}$ denote the angle between $\mathbf{v_i^*}$ and $\mathbf{v_j^*}$. Then $\mathbf{v_i^*} \circ \mathbf{r}$ and $\mathbf{v_j^*} \circ \mathbf{r}$ have different signs if and only if the line η that is normal to $\mathbf{r}$ passes between $\mathbf{v_i^*}$ and $\mathbf{v_j^*}$. Because $\mathbf{r}$ is uniformly distributed over the circle, this happens with probability at exactly $\theta_{i,j}/\pi$ (see Figure 20.1).

We have shown $\Pr[(i,j) \in \delta(S)] = \theta_{i,j}/\pi$. By definition of α, this is at least $(1 - \cos(\theta_{i,j}))/2$. We conclude by recalling that for any two vectors $\mathbf{u}, \mathbf{v}$ that $\mathbf{u} \circ \mathbf{v} = ||\mathbf{u}|| \cdot ||\mathbf{v}|| \cdot \cos(\theta)$. The constraints of (**SDP-Max-Cut**) ensure the vectors $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$ are unit vectors. Thus,

$$\Pr[(i,j) \in \delta(S)] \ge \alpha \cdot \frac{1 - \cos(\theta_{i,j})}{2} = \alpha \cdot \frac{1 - \mathbf{v}_i^* \circ \mathbf{v}_j^*}{2}.$$

■

20.4 Sampling A Random Vector

The rounding algorithm for MAX CUT required us to sample a random unit vector $\mathbf{r}$. We can do this simply by sampling from the normal distribution $\mathcal{N}(0,1)$. That is, for each coordinate $1 \le i \le n$, first sample $x_i \sim \mathcal{N}(0,1)$

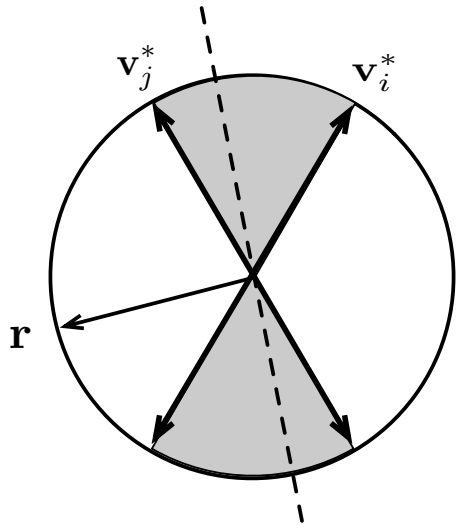

Figure 20.1: Illustration of a vector $\mathbf{r}$ such that $\mathbf{v}_j^* \circ \mathbf{r} > 0$ and $\mathbf{v}_i^* \circ \mathbf{r} < 0$. The dashed line is $\mathbf{r}$'s normal line, note that it passes between $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$.

where the distribution $\mathcal{N}(0,1)$ is given by the probability density function $p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$. This is done independently for each coordinate. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be this sampled vector. Return $\mathbf{r} = \mathbf{x}/||\mathbf{x}||$.

First we examine the probability density function for the random vector $(x_1, \ldots, x_n)$. Since the coordinates were sampled independently, the density of a particular vector $(x_1, \ldots, x_n)$ is precisely

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} = (2\pi)^{-n/2} e^{-\sum_{i=1}^{n} x_i^2/2} = (2\pi)^{-n/2} e^{-||\mathbf{x}||/2}.$$

In particular, any two vectors with the same length have the same density. So for every $d \geq 0$ the conditional distribution over $\mathbf{x}$ with $||\mathbf{x}|| = d$ is the uniform distribution. Therefore, the distribution over unit the vectors $\mathbf{r}$ is also uniform.

Sampling from $\mathcal{N}(0,1)$ is a bit outside of the scope of this course, but it can be done approximately well using independent, unbiased random bits. At the very least, even knowing that the expected weight is at least $\alpha \cdot OPT_{\text{SDP}}$ when $\mathbf{r}$ is sampled uniformly from the set of unit vectors is enough to prove that the integrality gap is at least α.

20.5 Discussion

This groundbreaking algorithm was discovered by Goemans and Williamson [GW95]. Surprisingly, there is evidence suggesting that this is the best possible approximation. That is, if the *Unique Games Conjecture* holds (we will discuss this later) there is no c-approximation for Max Cut for any constant $c > \alpha$ (where $\alpha \approx 0.87856$ is the constant defined above) [K+07]. Under the more widely believed conjecture that P $\neq$ NP, there is no $16/17 \approx 0.941176$ approximation for Max Cut [H01].

References

GW05 M. X. Goemans and D. P. Williamson, Improved algorithms for maximum cut and satisfiability problems using semidefinite programming, *Journal of the ACM*, 42:1115–1145, 1995.

H01 J. Håstad, Some optimal inapproximability results, *Journal of the ACM*, 48:798–869, 2001.

K+07 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*, 37(1):319–357, 2007.