

Semidefinite Programming Approach to Randomized Approximation Algorithms

1 Introduction

In this lecture we will demonstrate the use of semidefinite programming in randomized approximation algorithms. In particular, we will discuss groundbreaking work of Goemans and Williamson in [2] who introduce this approach to obtain approximate solutions for MAXCUT. First, we will introduce an integer quadratic program for MAX CUT. Then we will consider a relaxation that can be solved using semidefinite programming, as an approximation for the integer program. Lastly, we examine an additional application of semidefinite programming to graph coloring.

As motivation for the topic, techniques seen in this lecture have been used to find the only known polynomial time algorithm for finding maximum independent sets in perfect graphs. Lovász related the concept to Shannon capacities. Furthermore, the α -approximation in [2] is the first significant progress beyond a $(1/2)$ -approximation, which can be obtained quite easily. Thus, the techniques developed next are not only novel and interesting, but also fundamentally important and widely applicable.

2 MAX-CUT

The MAX CUT problem is the classical example of an optimization problem that can be approximated using a Semidefinite Programming relaxation. The goal is, given a graph $G = (V, E)$ and nonnegative weights w_{ij} on the edges $(i, j) \in E$, to find a subset S of the vertices of G that maximizes the sum of the weights of the edges from S to \bar{S} . Note that MAX CUT is NP-complete (even for the unweighted version). Thus we hope to find a ρ -approximation algorithm that will produce a valid cut whose weight is at least ρ times the value of the optimal cut.

In [2], Goemans and Williamson produce an $(\alpha - \epsilon)$ -approximation, where

$$\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \left(\frac{\theta}{1 - \cos \theta} \right) > 0.87856.$$

2.1 Integer Quadratic Program (Q)

Given a graph G with vertex set $V = \{1, 2, \dots, n\}$ and nonnegative edge weights w_{ij} as above, the maximum cut of G can be found by solving the following program:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i < j} w_{ij} (1 - x_i x_j) \\ \text{s.t.} \quad & x_i \in \{-1, 1\} \quad \forall i \in V \end{aligned}$$

Let $S = \{i : x_i = 1\}$. Then clearly an edge weight w_{ij} is counted if and only if exactly one of i, j is in S . That is, the objective function counts the weight of a cut.

2.2 Relaxation (P)

A natural relaxation that we often see of an integer program is to allow x_i to be any real number in the interval $[-1, 1]$. Let's consider a different relaxation. Notice that the $x_i \in \{-1, 1\}$ are unit vectors in one dimension. Suppose we allow them to be unit vectors v_i in many dimensions. Then we can replace the product $x_i x_j$ by the dot product $v_i \cdot v_j$. Then we have the following program:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i < j} w_{ij} (1 - v_i \cdot v_j) \\ \text{s.t.} \quad & v_i \in S_n \quad \forall i \in V \end{aligned}$$

where S_n is the unit sphere in \mathbb{R}^n . This is indeed a relaxation, since any solution to (Q) is a solution to (P), and thus

$$\text{OPT}(Q) \leq \text{OPT}(P).$$

2.3 Semidefinite Program (SDP)

Consider the set of vectors $v_i \in S_n$ from the program (P). Let B be the matrix whose i^{th} column is v_i , and define $Y = V^T V$. Then notice that $y_{ij} = v_i \cdot v_j$ for all $i, j \in V(G)$. Since the v_i are unit vectors, we also have $y_{ii} = v_i \cdot v_i = 1$. So we can replace the set of vectors $\{v_i\}$ by Y and write a new program. As we will see later, one way to define a positive semidefinite matrix is that it can be decomposed into a product $B^T B$ for some matrix B , and thus Y is positive semidefinite. Y is called the Gram matrix of the v_i .

Then (P) is equivalent to

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i < j} w_{ij} (1 - y_{ij}) \\ \text{s.t.} \quad & y_{ii} = 1 \quad \forall i \in V(G) \\ & Y \text{ is positive semidefinite} \end{aligned}$$

where $Y = (y_{ij})$. We will explore semidefinite matrices in detail in the next section.

Notice that since this is merely a reformulation of (P) we have

$$\text{OPT}(\text{SDP}) = \text{OPT}(P).$$

3 Semidefinite Programming

Definition. An $n \times n$ matrix A is called **positive semidefinite**(PSD) if for all $x \in \mathbb{R}^n$

$$x^T A x \geq 0.$$

In this case we denote $A \succcurlyeq 0$. In general, if $X - Y$ is PSD we will write $X \succcurlyeq Y$.

It can be shown that for a symmetric matrix A , the following are equivalent:

- $A \succcurlyeq 0$.
- All eigenvalues of A are nonnegative.
- There exists a matrix B such that $A = B^T B$.

Note that the set of $n \times n$ PSD matrices form a cone; that is, the set is closed under addition and multiplication by nonnegative scalars. To see this, we notice that if $A, B \succcurlyeq 0$ then for all $x \in \mathbb{R}^n$,

$$x^T(A + B)x = x^T Ax + x^T Bx \geq 0,$$

and for all $\lambda \geq 0$, we have $x^T \lambda Ax \geq 0$.

3.1 General form

A semidefinite program (SDP) is an optimization program with constraints restricting the variables to be entries of a positive semidefinite matrix. That is, for symmetric matrices C, A_1, \dots, A_m and the vector $b \in \mathbb{R}^m$ a semidefinite program has the form

$$\begin{aligned} \min \quad & C \bullet X \\ \text{s.t.} \quad & A_i \bullet X = b_i \quad \forall i = 1, 2, \dots, m \\ & X \succcurlyeq 0 \end{aligned}$$

where the product $A \bullet B = \sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{ij}$ is the Frobenius product.

3.2 LP \subset SDP

Consider the general form of a linear program:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & a_i^T x = b_i \quad \forall i = 1, 2, \dots, m \\ & x \in \mathbb{R}_+^n \end{aligned}$$

where c and a_i are vectors in \mathbb{R}^n . We will show that this linear program can be written as a semidefinite program. Let

$$C = \begin{bmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_n \end{bmatrix}, \quad A_i = \begin{bmatrix} a_{i1} & 0 & \dots & 0 \\ 0 & a_{i2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{in} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n \end{bmatrix}.$$

Notice that $X \succcurlyeq 0$ since each of the $x_i \geq 0$ and these x_i are the eigenvalues of X . Then clearly

$$C \bullet X, \text{ s.t. } A_i \bullet X = b_i$$

is in the form of a semidefinite program.

3.3 Properties of SDP

As semidefinite programming is a strict relaxation of linear programming, we do not have all of the same nice properties we had in linear programming, but we can still say a good bit. Since the solution of an SDP may be irrational, we are not guaranteed to be able to find an optimal solution in polynomial time. However, using interior point methods it is possible to find, for any $\epsilon > 0$, a solution whose objective value is within ϵ of optimality in time polynomial in the input and $\log(\frac{1}{\epsilon})$.

Other important properties include:

- Weak duality holds.
- Strong duality does not necessarily hold, and the duality gap may even be infinite. However, for all but some pathological cases, strong duality does hold.
- There is no finite algorithm for solving SDP. There is a simplex algorithm, but it need not terminate.

4 An Algorithm of Goemans and Williamson

Now that we've formulated the programs, we are ready to discuss a randomized approximation algorithm for MAXCUT. The algorithm due to Goemans and Williamson from the 1995 paper [2] is as follows.

- Solve (SD) and obtain the matrix \hat{Y} .
- Decompose \hat{Y} into $V^T V$, and let v_i denote the i^{th} column vector of V for $i = 1, \dots, n$.
- Choose a vector $r \in \mathbb{R}^n$ uniformly at random on the unit sphere S_n .
- Let $S = \{i : v_i \cdot r \geq 0\}$.

This is equivalent to choosing a random hyperplane in \mathbb{R}^n through the origin and letting S be the set of vertices of G that correspond to the vectors v_i that lie above this hyperplane. The other side of the cut, \bar{S} , corresponds to the remaining vectors from the solution to (SD). We call the cut $W = \sum_{i \in S, j \in \bar{S}} w_{ij}$.

Remark: Step two of this algorithm is possible since \hat{Y} is positive semidefinite. We may find V simply by diagonalizing $\hat{Y} = Q^T D Q$, where D is a diagonal matrix and Q is orthonormal (since \hat{Y} is symmetric). Then let $V = D^{1/2} Q$. Alternatively, we may obtain V of relatively low rank using an Incomplete Cholesky Decomposition, which allows faster computation.

4.1 Analysis of the Algorithm

We want to show that the cut W returned by this algorithm satisfies $E[W] \geq \alpha \cdot \text{OPT}(\text{SD})$ so that we have the fortunate inequality

$$\text{OPT}(\text{Q}) \geq E[W] \geq \alpha \cdot \text{OPT}(\text{SD}) = \alpha \cdot \text{OPT}(\text{P}) \geq \alpha \cdot \text{OPT}(\text{Q}).$$

That is, given an $\epsilon > 0$, we want to show that this is an $(\alpha - \epsilon)$ -approximation algorithm for MAXCUT. Recall that

$$\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \left(\frac{\theta}{1 - \cos \theta} \right) > 0.87856.$$

The analysis relies on the following facts.

Lemma 1.

$$\Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)] = \frac{1}{\pi} \cos^{-1}(v_i \cdot v_j),$$

where $\text{sgn}(x) = 1$ if $x \geq 0$ and -1 otherwise.

Since the v_i are unit vectors, $\cos^{-1}(v_i \cdot v_j)$ is just the angle between v_i and v_j . In other words, the probability that v_i and v_j are separated by the hyperplane $\{x : x \cdot r = 0\}$ is proportional to the angle between them.

Theorem 1.

$$E[W] = \frac{1}{\pi} \sum_{i < j} w_{ij} \cos^{-1}(v_i \cdot v_j).$$

Proof. By linearity of expectation,

$$E[W] = \sum_{i < j} w_{ij} Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)],$$

so the lemma implies the theorem. \square

Using basic calculus and by the definition of α , we get the following:

Theorem 2.

$$E[W] \geq \frac{\alpha}{2} \sum_{i < j} w_{ij} (1 - v_i \cdot v_j) = \alpha \cdot \text{OPT}(P)$$

Proof. Let $\theta_{ij} = \cos^{-1}(v_i \cdot v_j)$ be the angle between v_i and v_j .

$$\begin{aligned} E[W] &\geq \frac{1}{\pi} \sum_{i < j} w_{ij} \cos^{-1}(v_i \cdot v_j) \\ &= \frac{1}{\pi} \sum_{i < j} w_{ij} \frac{\theta_{ij}}{1 - \cos \theta_{ij}} (1 - \cos \theta_{ij}) \\ &\geq \min_{0 \leq \theta \leq \pi} \frac{1}{\pi} \sum_{i < j} w_{ij} \frac{\theta}{1 - \cos \theta} (1 - \cos \theta_{ij}) \\ &= \frac{\alpha}{2} \sum_{i < j} w_{ij} (1 - v_i \cdot v_j) \end{aligned}$$

4.2 Results

Computational results show that in practice this algorithm for MAXCUT gives a significantly better approximation than the worst-case bound of α . In roughly 300 trials on four types of random graph models, Goemans and Williamson [2] found that, on average, their algorithm produced a .96-approximation.

However, the bound of α is believed to be tight. In the 2005 paper, [1], Khot et al. use the Unique Games Conjecture and the Majority is Stablest Theorem to show that α is best possible. Hence if this conjecture is true, no other algorithm can give a better approximation for MAXCUT.

In [?], Goemans and Williamson show that the analysis above provides the best bounds for this algorithm; that is, there exist graphs for which $E[W]/\text{OPT}(P)$ is arbitrarily close to α . The worst-case graphs correspond to strongly self-dual polytopes.

Definition. (Lovász) A polytope $P \subset \mathbb{R}^n$ is said to be **strongly self-dual** if

- (i) P is inscribed in the unit sphere.
- (ii) P is circumscribed around the sphere with the origin as its center, and radius $0 < r < 1$.
- (iii) There is a bijection σ between vertices and facets of P such that for every vertex v of P , the facet $\sigma(v)$ is orthogonal to the vector v .

Given a strongly self-dual polytope P , we associate it with the graph $G = (V, E)$, where V is the set of vertices of P and $(u, v) \in E$ if v is a vertex on the facet $\sigma(u)$. For example, in \mathbb{R}^2 , the strongly self-dual polytopes are precisely the regular odd polygons. The graphs associated with them are odd cycles.

The Petersen graph is another graph that gives a particularly bad approximation ratio ($\approx .8787$).

5 Semidefinite Programming Applied to k -Coloring

The technique introduced by Goemans and Williamson has inspired efforts to use semidefinite programming to approximate other combinatorial problems. One such application, by Karger, Motwani and Sudan in [3], is to properly color a k -colorable graph $G = (V, E)$ with a small number of colors. For example, their randomized algorithm colors a 3-colorable graph with

$$\min\{O(\Delta^{1/3} \log^{1/2} \Delta \log n), O(n^{1/4} \log^{1/2} n)\}$$

colors, where $n = |V|$ and Δ is the maximum degree of a vertex in G .

While initially this result may not seem impressive, consider this fact: the existence of an algorithm that, when given a graph G that is known to be 3-chromatic, colors G with four colors, implies that $P=NP$. Furthermore, this result represents an improvement on any existing algorithm (at the time of publication) both in terms of Δ and n .

5.1 Vector Relaxation

Recall that a proper k -coloring of a graph G is an assignment of k colors to the vertices of G such that no two neighboring vertices have the same color. Suppose that instead of assigning colors to vertices, we assign an n -dimensional unit vector v_i to each vertex. We call this a **vector k -coloring** if adjacent vertices i, j have corresponding vectors v_i, v_j such that

$$v_i \cdot v_j \leq -\frac{1}{k-1}.$$

Notice that this requires that the angle between v_i and v_j is greater than $\pi/2$. We call an $n \times n$ PSD matrix M a **matrix k -coloring** if $m_{ii} = 1$ for all i , and $m_{ij} \leq \frac{-1}{k-1}$ for all $ij \in E$.

Theorem 3. A graph G has a matrix k -coloring if and only if G has a vector k -coloring.

Theorem 4. If a graph has a vector k -coloring, then one can construct a vector $(k + \epsilon)$ -coloring in polynomial time in $k, n, \log(\frac{1}{\epsilon})$.

Proof. The optimal solution to the following SDP gives a matrix k -coloring of G .

$$\begin{array}{ll} \min & \beta \\ \text{s.t.} & M = (m_{ij}) \text{ is PSD} \\ & m_{ij} \leq \beta \quad \text{if } ij \in E \\ & m_{ij} = m_{ji} \quad \forall i, j \\ & m_{ii} = 1 \quad \forall i \end{array}$$

Since there exists a matrix k -coloring of G , the optimal solution has $\hat{\beta} \leq \frac{-1}{k-1}$. Using an Incomplete Cholesky Decomposition, we can write $\hat{M} = V^T V$, where the columns of V are the vectors in the desired vector k -coloring. \square

Theorem 5. Every k -colorable graph has a vector k -coloring.

5.2 Coloring Algorithm

Given a 3-colorable graph, we can perform the following polynomial time randomized approximation algorithm:

- Use the above SDP to find a vector $(3 + \epsilon)$ -coloring.
- Use randomized rounding to create a semicoloring with $O(\Delta^{\log_3 2})$ colors.
- Create a proper coloring of the graph with desired number of colors.

In step 2, the randomized rounding procedure is similar to the rounding approach presented in section 4 with repetition. That is, we choose r_1, r_2, \dots, r_l random vectors, and consider the regions of the sphere S_n defined by the hyperplanes. Then vertices will receive the same color if and only if they correspond to vectors lying in the same region of this partition. A semicoloring is a coloring of a graph G such that at most $|V(G)|/4$ edges have ends of the same color. Note that finding a proper coloring of G from a semicoloring, as in step 3 above, can be performed in polynomial time.

6 Conclusion

In this lecture we have introduced the concept of semidefinite programming. We have seen that this technique can be applied to several NP-hard problems with striking success. Some problems not discussed here for which SDP has been applied successfully include MAX 2-SAT, MAXSAT, and MAX DICUT, each of which appear in [2]. An area of current interest is the relationship between SDP and Max-CSPs (constraint satisfaction problems).

References

- [1] Khot, Kindler, Mossel, and O’Donnell. Optimal inapproximability Results for MAXCUT and other 2-variable CSPs? *FOCS Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 00:146-154, 2004.
- [2] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115-1145, 1995.
- [3] D. Karger, R. Motwani and M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45:246-265, 1998.