Optimization and Approximation ENS Lyon

Lecture 10

Lecturer: Alantha Newman *December 6, 2016*

1 Semidefinite Programming and Graph Partitioning

In previous lectures, we saw how linear programs can be deployed to design approximation algorithms for various $\mathcal{NP}$-hard optimization problems. In this lecture, we will introduce a more general class of relaxations, *vector programs*, which allow for variables to be vectors instead of scalars. In particular, vector programs are a relaxation for problems that can be formulated as *strict quadratic programs.*

Definition 1 (Quadratic program)**.** *A quadratic program (QP) is an optimization problem whose objective function is quadratic in terms of integer valued variables, subject to quadratic constraints. If in addition a quadratic program has all its monomials degree 0 or 2, then it is a* ***strict quadratic program****.*

As we will show later, vector programs are equivalent to semidefinite programs, which can be viewed as a generalization of linear programs. We can solve semidefinite programs by the ellipsoid algorithm up to an arbitrary small additive error ϵ, in time polynomial in the input size and $\log(1/\epsilon)$.

We will illustrate the concept of semidefinite programming by examining some graph partitioning problems. We start with the well-known maximum cut problem.

1.1 Maximum Cut

Given an undirected graph $G = (V, E)$, the goal of the maximum cut problem (max-cut) is to find a partition of the vertices, $(S, \bar{S})$, that maximizes the number of edges crossing the cut, i.e. edges with one endpoint in S and the other endpoint in $\bar{S}$. We denote the number of edges crossing the maximum cut by $\mathrm{OPT}_{\text{max-cut}}$. Max-cut is known to be $\mathcal{NP}$-Hard, so our goal is to find a good polynomial time approximation algorithm for it. Note that $|E|$ is an upper bound on $\mathrm{OPT}_{\text{max-cut}}$. Thus, we can achieve a $\frac{1}{2}$-approximation for max-cut simply by placing each vertex in S or $\bar{S}$ with probability 1/2.

The maximum cut problem can be formulated as a quadratic program:

$$\begin{aligned} \max \sum_{ij\in E} \big(x_i(1-x_j) + x_j(1-x_i)\big) \qquad (1)\\ \text{s.t } x_i \in \{0,1\} \quad \forall i \in V, \end{aligned}$$

where a variable $x_i \to 1$ if vertex i is in set S, and $x_i \to 0$ if vertex i is in set $\bar{S}$. Note that an edge with both ends in the same set will not contribute to the objective function.

If we relax the integer constraint in this QP, we have the following formulation:

$$\begin{aligned} \max \sum_{ij\in E} \big(x_i(1-x_j) + x_j(1-x_i)\big) \qquad (2)\\ \text{s.t } x_i \in [0,1] \quad \forall i \in V. \end{aligned}$$

If we can solve this relaxed version optimally, we will still be able to find a maximum cut. Consider the solution for the relaxed QP, $\mathrm{OPT}_{\text{rel}}$. For any vertex $h \in V$, with fractional value assigned to x_h, we can rewrite the objective function (1) as follows. Let $\delta(h)$ denote all edges adjacent to vertex h.

$$\sum_{ij\in E\setminus\delta(h)} (x_i(1-x_j) + x_j(1-x_i)) + x_h \overbrace{\sum_{j:hj\in\delta(h)} (1-x_j)}^{A} + (1-x_h) \overbrace{\sum_{j:hj\in\delta(h)} x_j}^{B}.$$

Then if $A \geq B$, we round x_h to one, otherwise we round it to zero. Let's denote by OPT_{rd} the solution we get after rounding $\mathrm{OPT}_{\text{rel}}$. Note that $\mathrm{OPT}_{\text{rel}} \leq \mathrm{OPT}_{\text{rd}}$, so by solving the relaxed QP for max-cut and rounding the solution, we obtain an integral solution that is at least as good as OPT_{rel}, which is at least as good as the true optimal value $\text{OPT}_{\text{max-cut}}$. We can deduce from this that solving this particular relaxed version is also $\mathcal{NP}$-hard, since it boils down to solving exactly the $\mathcal{NP}$-hard maximum cut problem in polynomial time. So relaxing the integrality constraint does not help here. Instead, we'll relax max-cut to a semidefinite program.

Definition 2 (Semidefinite program (SDP))**.** *A semidefinite program is a convex optimization problem whose objective function is linear in terms of its variables x_{ij} and which is subject to linear constraints over the variables, with the additional constraint that the symmetric matrix $X = [x_{ij}]$ should be positive semidefinite.*

We recall the definition of positive semidefinite matrix:

Definition 3. *A symmetric matrix $X \in \mathbb{R}^{n\times n}$ is positive semidefinite ($X \succeq 0$) iff for all $y \in \mathbb{R}^n$, $y^T X y \geq 0$.*

Theorem 4. *If $X \in \mathbb{R}^{n\times n}$ is a symmetric matrix, then the following are equivalent:*

1. X is positive semidefinite.
2. All eigenvalues of X are non negative.
3. $\exists V \in \mathbb{R}^{m\times n}$, $m \leq n$, s.t $X = V^T V$.

(Note that we are abusing notation here by using V to refer to a matrix, rather than the previously defined vertex set V of a graph G. However, this notation will be useful since we will eventually use the vector corresponding to a column of V to represent a vertex in G.) Since we can compute the eigendecomposition of a symmetric matrix $X = Q\Lambda Q^T$ in polynomial time, we can test for positive definiteness in polynomial time. However, the decomposition $V^T V$ is not polynomial time computable, since taking the square root of the diagonal matrix Λ can lead to irrational values. We can get an arbitrarily good approximation of this decomposition, so we can assume we have the exact decomposition in polynomial time, given that this inaccuracy can be included in the approximation factor.

By replacing the variable x_{ij} in an SDP by the inner product of the two vectors v_i and v_j in the decomposition $X = V^T V$ corresponding to entry x_{ij}, we obtain an equivalent vector program:

$$\begin{aligned} \max & \sum_{i,j} c_{ij} x_{ij} \\ \text{s.t} & \sum_{i,j} a_{ijk} x_{ij} = b_k \\ & x_{ij} = x_{ji} \\ & X \succeq 0 \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} \max & \sum_{i,j} c_{ij} (v_i \cdot v_j) \\ \text{s.t} & \sum_{i,j} a_{ijk} (v_i \cdot v_j) = b_k \\ & v_i \in \mathbb{R}^n \end{aligned}$$

The maximum cut problem admits a strict quadratic program formulation, which can be relaxed to a vector program:

$$\begin{aligned} \max & \sum_{ij\in E} \frac{1 - v_i \cdot v_j}{2} \\ \text{s.t} & \ v_i \in \{-1, 1\} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \max & \sum_{ij\in E} \frac{1 - v_i \cdot v_j}{2} \\ \text{s.t} & \ v_i \cdot v_i = 1 \\ & v_i \in \mathbb{R}^n \end{aligned}$$

1.2 Random Hyperplane Rounding

Given a solution $\{v_i |\ \forall i \in V\}$ to the vector program of max-cut with value OPT_v, we round our solution as follows:

- Pick a vector g uniformly at random from the unit sphere S_{n-1}.
- For all $i \in V$: $\begin{cases} g \cdot v_i \geq 0 & \Rightarrow i \to S, \\ g \cdot v_i < 0 & \Rightarrow i \to \bar{S}. \end{cases}$

This rounding procedure is called **random hyperplane rounding**.

Theorem 5. *There exists a polynomial time algorithm that achieves a 0.878-approximation of the maximum cut with high probability.*

To prove Theorem 5, we first show that there exists a method to generate a vector g uniformly at random from the unit sphere S_{n-1}.

Lemma 6. *Let $r = (r_1, r_2, \ldots, r_n)$ be random variables such that $r_i \in \mathcal{N}(0,1)$, i.e. each entry is chosen according to the normal distribution with mean 0 and variance 1. Let $d = \sqrt{\sum_{i=1}^n r_i^2}$ and let $g = (r_1/d, r_2/d, \ldots, r_n/d)$. Then g is uniformly distributed on the unit sphere S_{n-1}.*

Proof. Since each of the random variables $r_1, r_2, \ldots, r_n$ is chosen according to the normal distribution $\mathcal{N}(0,1)$, the probability density function of r is:

$$\begin{aligned} \Pr(r_1, r_2, \ldots, r_n) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{r_i^2}{2}} \\ &= \frac{1}{(2\pi)^{\frac{n}{2}}} e^{-\frac{\sum_{i=1}^n r_i^2}{2}} = \frac{1}{(2\pi)^{\frac{n}{2}}} e^{-\frac{d^2}{2}}. \end{aligned}$$

We see that the probability density function only depends on the length of the vector. Therefore all vectors of the same length have an equal probability of being chosen. Thus, the normalized vector g will be uniformly distributed on the unit sphere S_{n-1}. $\square$

We state (but do not prove) the following useful fact related to the distribution of a weighted sum of normally distributed random variables.

Fact 7. *Let $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ denote a normally distributed random variable with mean μ_i and variance σ_i^2. Suppose $X_1, X_2, \ldots, X_m$ are independent random variables. Then:*

$$\sum_{i=1}^{m} a_i X_i \sim \mathcal{N}\left(\sum_{i=1}^{m} a_i \mu_i, \sum_{i=1}^{m} (a_i \sigma_i)^2\right).$$

Let $r = (r_1, r_2, \ldots, r_n)$, where $r_i \in \mathcal{N}(0,1)$. The following two lemmas provide useful facts about the projection of r to lower dimensions. Let $u \in \mathbb{R}^n$ denote a unit vector.

Lemma 8. *The projection of r onto unit vector u is normally distributed with mean 0 and variance 1.*

Proof. Let $r' = r \cdot u$ denote the projection of r onto u. Fact 7 immediately implies that $r' = \sum_{i=1}^n u_i \cdot r_i \sim \mathcal{N}(0,1)$. $\square$

Lemma 9. *The projections of r onto unit vectors e_1 and e_2 are independent iff e_1 and e_2 are orthogonal.*

Proof. Let r_1 and r_2 denote the projections of r onto e_1 and e_2, respectively. By Fact 7, r_1 and r_2 are both normally distributed random variables with mean 0 and variance 1. In this case, it is sufficient to prove that the covariance of r_1 and r_2 is zero, which is the case if $\mathbb{E}[r_1 r_2] = 0$. If e_1 and e_2 are orthogonal, then:

$$\mathbb{E}[r_1 r_2] = \mathbb{E}[(e_1^\intercal r)(r^\intercal e_2)] = e_1^\intercal \mathbb{E}[r r^\intercal] e_2 = e_1^\intercal e_2 = 0,$$

implying that r_1 and r_2 are independent. If r_1 and r_2 are independent, then $\mathbb{E}[r_1 r_2] = \mathbb{E}[r_1]\mathbb{E}[r_2] = 0$, which implies that $e_1^\intercal e_2 = 0$. $\square$

Lemma 10. *Let r' be the projection of r onto a 2-dimensional plane, then $\frac{r'}{\|r'\|}$ is uniformly distributed on a unit circle in the plane.*

Proof. Consider the 2-dimensional plane defined by unit vectors e_1 and e_2 and let r_1 and r_2 denote the projections of r onto e_1 and e_2, respectively. Then $r' = (r_1, r_2)$.

By Lemma 8, each projection (i.e. r_1 and r_2) has mean 0 and variance 1, and by Lemma 9, the two projections are independent random variables. Thus, r_1 and r_2 are independent normally distributed random variables, and we can apply Lemma 6 to conclude that $\frac{r'}{\|r'\|}$ is uniformly distributed on the unit circle. $\square$

Lemma 11. *The probability that edge ij is cut is $\frac{\arccos(v_i \cdot v_j)}{\pi} = \frac{\theta_{ij}}{\pi}$, where θ_{ij} is the angle between vectors v_i and v_j.*

Proof. Project vector r onto the plane containing v_i and v_j. It is easy to see that edge ij is cut iff r falls within the area formed by the vectors perpendicular to v_i and v_j, which has area equal to $\frac{2\theta_{ij}}{2\pi}$. $\square$

Lemma 12. *For $x \in [-1, 1]$, one can show that:* $\frac{\arccos(x)}{\pi} \geq 0.878\left(\frac{1-x}{2}\right)$.

Now we can compute the expected weight of the edges crossing the cut produced by random hyperplane rounding.

$$\begin{aligned}
E[\text{Weight of cut}] &= E[\sum_{ij \in E} \Pr(\text{edge } ij \text{ is cut})] \\
&= \sum_{ij \in E} \frac{\theta_{ij}}{\pi} && \text{(by Lemma 11)} \\
&= \sum_{ij \in E} \frac{\arccos(v_i \cdot v_j)}{\pi} \\
&\geq 0.878 \sum_{ij \in E} \frac{1 - (v_i \cdot v_j)}{2} && \text{(by Lemma 12)} \\
&= 0.878 \ \mathrm{OPT}_v \\
&\geq 0.878 \ \mathrm{OPT}_{\text{max-cut}}
\end{aligned}$$

Given this expected value, one can show the existence of an algorithm that achieves a 0.878-approximation of the maximum cut in polynomial time, with high probability. This concludes the proof of Theorem 5.

Finally, we give an example to show that this approximation factor is almost tight. Consider a 5-cycle graph. $\mathrm{OPT}_{\text{max-cut}} = 4$, while the optimal solution for the SDP is placing the 5 vector in a 2-dimensional plane with an angle $\frac{2\pi}{5}$ between each two vectors. The approximation factor achieved in this case is 0.884.

1.3 Correlation Clustering

In the *correlation clustering problem*, we are given an undirected graph $G = (V, E)$ with edge weights w_{ij}^+ and w_{ij}^-, which denote how similar or dissimilar, respectively, the endpoints of an edge are. (In our analysis we'll assume that each edge has only one of these two types of weights, but the approximation algorithm will still work in general.) Our goal is to partition the vertex set into clusters of similar vertices. In other words, we aim to maximize the following objective function:

$$\max \sum_{i,j \text{ are in the same cluster}} w_{ij}^+ + \sum_{i,j \text{ are in different clusters}} w_{ij}^-.$$

We denote the optimal value by OPT_{cc}.

The correlation clustering problem also admits a simple $\frac{1}{2}$-approximation algorithm by picking the best of the following two procedures:

1. Form one cluster: $S = V$.
2. Set each vertex to be in its own cluster.

Note that if the total sum of w^+_{ij} in the graph is greater than the total sum of w^-_{ij}, choice 1 will guarantee at least half OPT_{cc}, otherwise choice 2 will.

The correlation clustering problem admits an exact formulation that can be relaxed to a vector program:

$$\begin{aligned} &\max \sum_{ij \in E} \left(w^+_{ij}(v_i \cdot v_j) + w^-_{ij}(1 - v_i \cdot v_j) \right) \\ &\text{s.t } v_i \in \{e_1, e_2, \cdots, e_n\} \quad \forall i \in V \end{aligned} \quad \Longrightarrow \quad \begin{aligned} &\max \sum_{ij \in E} \left(w^+_{ij}(v_i \cdot v_j) + w^-_{ij}(1 - v_i \cdot v_j) \right) \\ &\text{s.t } v_i \cdot v_i = 1 \\ &\quad\ \ v_i \cdot v_j \geq 0 \\ &\quad\ \ v_i \in \mathbb{R}^n \end{aligned}$$

where e_k denotes the unit vector with the k^{th} entry set to one. In the exact formulation, v_i is set to e_k if vertex i belongs to cluster k.

1.4 Rounding Algorithm for Correlation Clustering

Given a solution $\{v_i | \ \forall i \in V\}$ to the vector program of correlation clustering with value OPT_v, we round our solution as follows:

1. Pick two random vectors g and h s.t each entry in each vector is drawn from the standard normal distribution $\mathcal{N}(0, 1)$.
2. Form four clusters as follows:

$$\begin{aligned} C_1 &= \{i \in V : \ g \cdot v_i \geq 0 \text{ and } h \cdot v_i \geq 0\}, \\ C_2 &= \{i \in V : \ g \cdot v_i \geq 0 \text{ and } h \cdot v_i < 0\}, \\ C_3 &= \{i \in V : \ g \cdot v_i < 0 \text{ and } h \cdot v_i \geq 0\}, \\ C_4 &= \{i \in V : \ g \cdot v_i < 0 \text{ and } h \cdot v_i < 0\}. \end{aligned}$$

Theorem 13. *There exists a polynomial time algorithm that achieves a $\frac{3}{4}$-approximation of the correlation clustering problem with high probability.*

Proof. We start with a useful lemma:

Lemma 14. *For $x \in [0, 1]$, one can show that $\frac{(1 - \frac{\arccos(x)}{\pi})^2}{x} \geq 0.75$ and $\frac{1 - (1 - \frac{\arccos(x)}{\pi})^2}{1 - x} \geq 0.75$.*

Let x_{ij} be a random variable that takes the value one if i and j are in the same cluster. Note that the probability that v_i and v_j are not separated by either g or h is $(1 - \frac{\arccos(v_i \cdot v_j)}{\pi})^2$. Thus,

$$E[x_{ij}] = (1 - \frac{\arccos(v_i \cdot v_j)}{\pi})^2.$$

Now we can compute the expected weight of the clustering we obtain from solving the vector program for correlation clustering and applying the random 2-hyperplane rounding.

$$\begin{aligned}
E[\text{Weight of clustering}] &= E[\sum_{ij\in E} w_{ij}^{+}x_{ij} + w_{ij}^{-}(1-x_{ij})] \\
&= \sum_{ij\in E} w_{ij}^{+}(1-\frac{\arccos(v_i\cdot v_j)}{\pi})^2 + w_{ij}^{-}(1-(1-\frac{\arccos(v_i\cdot v_j)}{\pi})^2) \\
&\geq 0.75 \sum_{ij\in E} w_{ij}^{+}(v_i\cdot v_j) + w_{ij}^{-}(1-v_i\cdot v_j) && \text{(by Lemma 14)} \\
&= 0.75\ \text{OPT}_v \\
&\geq 0.75\ \text{OPT}_{cc}.
\end{aligned}$$

Given this expected value, one can show the existence of an algorithm that achieves a 0.75-approximation for the correlation clustering problem in polynomial time, with high probability. □

References

[CGW05] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[Swa04] Chaitanya Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the fifteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 526–527, 2004.

[Vaz13] Vijay V. Vazirani. *Approximation Algorithms.* Springer, 2013.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.

These lecture notes are mainly based on Chapter 6 of [WS11] and Chapter 26 of [Vaz13]. The algorithm for the maximum cut is due to Goemans and Williamson [GW95] and is the first use of semidefinite programming in approximation algorithms. The algorithm for correlation clustering is due to Swamy [Swa04]. A similiar algorithm for correlation clustering with a slightly better approximation guarantee can be found in [CGW05]. A previous version of these lecture notes scribed by Marwa El Halabi was used in a course on approximation algorithms (Lecture 14) at EPFL (`http://theory.epfl.ch/osven/courses/Approx13`).