CSE 521: Design and Analysis of Algorithms I **Winter 2017**

Lecture 10: Clustering, Spectral Partitioning

Lecturer: Shayan Oveis Gharan *02/13/17* *Scribe: Yihan Jiang*

10.1 Low Rank Approximation in Optimization

In this lecture we are going to study Low Rank Approximation applications in optimization. In particular, we will discuss a spectral algorithm for the maximum cut problem. Given a graph $G = (V, E)$, find $S \subseteq V$ such that $|E(S, \bar{S})|$ is maximized. As we discussed in the last lecture, given adjacency matrix A, Max Cut Problem can be formed as:

$$\max_{x \in \{0,1\}^n} x^T A(1 - x) \tag{10.1}$$

Max Cut Problem is among Karp's 21 NP-complete problems. It is also shown the problem hard to approximate with a factor better than 16/17 unless $P = NP$. We will prove the following theorem.

Theorem 10.1. *Let as $k > 1$ be an integer, for any matrix $A \in \{0,1\}^{n \times n}$, we can approximate (10.1) with an additive $\frac{n^2}{\sqrt{k}}$ error, in $O(k^k poly(n))$ time.*

Note that if G is a dense graph the above theorem gives a $(1-\epsilon)$ multiplicative approximation for a sufficiently large k.

Corollary 10.2. *Suppose for every vertex i of G, $d(i) \geq c \cdot n$ for a constant $c > 0$ and let $\epsilon > 0$. Then, for $k \leq O(\frac{1}{c^2 \epsilon^2})$, there is an algorithm that returns a cut of size at most $(1 - \epsilon)$ fraction of the optimum. The algorithm runs in time $O(k^k \operatorname{poly}(n))$.*

Proof. First of all, it is not hard to see that the optimum solution of Max Cut for any graph G is at least $|E|/2$ (and at most $|E|$). This is because a uniformly random set S cuts half of the edges in expectation.

Since every vertex has degree at least cn,

$$|E| \geq n(cn)/2 = cn^2/2.$$

Now, choose $k = O(\frac{1}{c^2 \epsilon^2})$ such that $1/\sqrt{k} = \epsilon c/2$. Then, by Theorem 10.1 there is an algorithm that finds a cut of size at least

$$OPT - n^2/\sqrt{k} \geq OPT - \epsilon cn^2/2 \geq (1 - \epsilon) OPT$$

where in the last inequality we used that $OPT \geq cn^2/2$. □

To prove Theorem 10.1, first we approximate A with rank k matrix, A_k. Secondly we solve the optimization problem (10.1) with respect to A_k. It turns out that, since A_k is a low rank matrix, we just need to solve a k-dimensional problem, as opposed to the original n dimensional problem of choosing $x \in \{0,1\}^n$. In the second step, we will show to approximately solve this problem using a technique called ϵ-net.

10.1.1 Step 1

Given a matrix $A \in \{0,1\}^n$, firstly we prove the following claim. Note that here for simplicity we assume A is a symmetric matrix; but the same proof works for non-symmetric matrices.

Claim 10.3. *For an integer $k \geq 1$, let A_k be the best rank k approximate of A with respect to the Frobenius norm. Then, for any vector $x \in \{0,1\}^n$,*

$$|x^T A(1-x) - x^T A_k(1-x)| \leq O(\frac{n^2}{\sqrt{k}}).$$

Proof. As

$$A = \sum_i \lambda_i v_i v_i^T,$$

with eigenvalues $\lambda_1 \geq \lambda_2, ...$ and corresponding orthonormal eigenvectors $v_1, \ldots, v_n$. As we discussed in the previous lecture,

$$A_k = \sum_{i=1}^{k} \lambda_i v_i v_i^T.$$

Now, we can write,

$$\begin{aligned} |x^T A(1-x) - x^T A_k(1-x)| &= |\langle x, (A - A_k(1-x)\rangle| \\ &\leq \|x\| \cdot \|(A-A_k)(1-x)\|_2 \\ &\leq \|x\| \cdot \|A - A_k\|_2 . \|1-x\| \\ &\leq \sqrt{n}(\lambda_{k+1})\sqrt{n} = n\lambda_{k+1} \end{aligned}$$

The first inequality follows by Cauchy-Schwarz inequality($|\langle a,b\rangle| \leq \|a\| \cdot \|b\|$ for any two vectors a, b). The third inequality follows from the definition of matrix operator norm. $\|A - A_k\|_2 = \max_y \frac{\|(A-A_k)y\|}{\|y\|}$. The last inequality follows from the definition of x, A_k. In particular, for any binary vector x,

$$\|x\|^2 = \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i \leq n.$$

Also by definition of A_k,

$$A - A_k = \sum_{i=k+1}^{n} \lambda_i v_i v_i^T,$$

But, as we proved in lecture 8, the operator norm is equal to the largest singular value (or largest eigenvalue for symmetric matrices), so $\|A - A_k\|_2 = \lambda_{k+1}$.

Now, to finish the proof we need to upper bound λ_{k+1}. Since $\lambda_1 \geq ... \geq \lambda_n$,

$$\lambda_{k+1}^2 \leq \frac{\lambda_1^2 + ... + \lambda_{k+1}^2}{k+1} \leq \frac{\sum_{i=1}^n \lambda_i^2}{k+1} = \frac{||A||_F^2}{k+1} = \frac{n^2}{k+1}$$

In the second to last equality we use the fact that $A_F^2 = \sum_i \lambda_i^2$. And, in the last equality we use that $A \in \{0,1\}^{n \times n}$ matrix. So we have $\lambda_{k+1} \leq \frac{n}{\sqrt{k}}$, which proved the claim. □

Note that the above upper bound on λ_{k+1} is very loose. In the worst case, if all of the first $k+1$ eigenvalues are equal and the rest are 0, then the bound is tight; there are graphs of this from but in such a case one should choose k at the point where there is a large gap between eigenvalues. This idea can be very useful in practice. Because many of the matrices that we work with in practice have large gaps between their eigenvalues or singular values. So, we can approximate them up to a very small error using a low rank approximation.

10.1.2 Step 2

In the second step we approximately solve (10.1) for a low rank matrix A_k. Recall that $A_k = \sum_{i=1}^k \lambda_i v_i v_i^T$. So, for a vector x,

$$x^T A_k(1-x) = x^T \left(\sum_{i=1}^k \lambda_i v_i v_i^T\right)(1-x) = \sum_{i=1}^k \lambda_i \langle v_i, x\rangle\langle v_i, 1-x\rangle. \tag{10.2}$$

Recall that x is supposed to be an indicator vector of a set. For a set S, let $\underline{1}^S$ is defines as follows:

$$\mathbf{1}_i^S = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise.}\end{cases}$$

Define $v_i(S) = \sum_{j\in S} v_{i,j} = \langle v_i, \mathbf{1}^S\rangle$. So, we can calculate $\underline{1}^{S^T} A_k(1-\underline{1}^{\overline{S}})$ using $v_1(S), \dots, v_k(S), v_1(\overline{S}), \dots, v_k(\overline{S})$. Note that this is an inherently $2k$ dimensional problem, so one should expect to solve it faster, i.e., in time exponential in k. This may not be clear at this point because we associate each of the 2^n possible solutions to the max cut problem with one point in a k dimensional space. So, in the worst case, one needs to brute force over all such vectors to find the best cut.

We show that it is enough to have an approximate value of $v_i(S)$ for every coordinate to aprpoximately maximize (10.1) for A_k. It turns out that if we have $v_i(S)$ only with some small error ϵ still that is enough to approximate $\mathbf{1}^{S^T} A_k(1-\mathbf{1}^{\overline{S}})$ within $n^2/\sqrt{k}$ error. In particular, supposed $|\tilde{v}_i(S) - v_i(S)| \le \epsilon$ for all i. Then,

$$\begin{aligned}
\left|\sum_{i=1}^k \lambda_i v_i(S) v_i(\overline{S}) - \sum_{i=1}^k \lambda_i \tilde{v}_i(S)\tilde{v}_i(\overline{S})\right| &\le \left|\sum_{i=1}^k \lambda_i v_i(S) v_i(\bar{S}) - \sum_{i=1}^k (v_i(S)\pm\epsilon)(v_i(\bar{S})\pm\epsilon))\right| \\
&\le \sum_{i=1}^k \lambda_i \cdot \epsilon \cdot \max_{\{1\le j\le k}\{v_j(S), v_j(\bar{S})\} \\
&\le n\sqrt{k}\frac{\sqrt{n}}{k}\sqrt{n} \\
&= O(\frac{n^2}{\sqrt{k}})
\end{aligned}$$

Let us discus the last inequality. We choose $\epsilon = O(\frac{\sqrt{n}}{k})$. For every set S,

$$v_j(S) = \langle v_j, S\rangle \le \|v_j\| \cdot \|\mathbf{1}^S\| \le \sqrt{n}.$$

Also, by Cauchy-Schwarz inequality,

$$\sum_i^k \lambda_i \le \sqrt{k} \cdot \sqrt{\sum_i^k \lambda_i^2} \le \sqrt{k} \cdot \|A\|_F = \sqrt{k|E|} \le n\sqrt{k}.$$

Now, we define $\tilde{v}_i(S)$ by rounding $v_i(S)$ to a multiple of ϵ. This way, we essentially discretize the vector $v_i(S)$ and $v_i(\overline{S})$. Note that since $-\sqrt{n} \le v_i(S) \le \sqrt{n}$,

$$\tilde{v}_i(S) \in \{-k\epsilon, -(k-1)\epsilon, ... + k\epsilon\}.$$

So, now, our search space only has $(2k)^{2k}$ discrete points. Note that the discretization argument essentially reduces our 2^n possible solutions to max cut to $(2k)^{2k}$ many solutions. So, all we need to do is to brute force over all possible $\tilde{v}$ vectors, i.e., all $(2k)^{2k}$ possibilities. For each of them we can calculate $\sum_{i=1}^{k} \lambda_i \tilde{v}_i(S)\tilde{v}_i(\overline{S})$ and choose the best point that we find. The algorithm runs in time $O((2k)^{2k}\,\mathrm{poly}(n))$.

The above idea is called an *ϵ-net.* Whenever we have an optimization problem in a k dimensional space even if there are 2^n possible solutions to our problem we can divide the space by a grid with cell size ϵ. Then, for each cell we choose a representative. We brute force over all of the cells and choose the representative of highest value among all sets as the solution to our optimization problem. Note that there is a tradeoff in choosing ϵ. On one hand, we want to choose ϵ as big as possible to decrease the size of the search space of our brute force algorithm and on the other hand we want to choose it as small as possible to make the error incurred by choosing just one representative per cell as small as possible. In practice one should choose ϵ based on the available computational power.

Finally, there is one technical point that we did not mention. For any approximate vector $x \in \{-k\epsilon, -(k-1)\epsilon, ..., +k\epsilon\}^{2k}$, we need to consider x in our brute force algorithm if there is a set S where the approximate

$$(\tilde{v}_1(S), \ldots, \tilde{v}_k(S), \tilde{v}_1(\overline{S}), \ldots, \tilde{v}_k(\overline{S})) = x.$$

To find such a set we can use tools from convex optimization, and in particular linear programming. We will discuss this in future lecture. But, the upshot is that we can find such a set S for a given vector x efficiently.

10.2 Spectral Partitioning

In the next lecture we will discuss spectral partitioning algorithms. Let us conclude this section by giving background on a very important matrix associated with graphs which is called the Laplacian matrix.

For two vertices i, j of a graph we use $i \sim j$ to denote that $(i, j) \in E$ is an edge of G.

Definition 10.4 (Laplacian Matrix)**.** *Given $G = (V, E)$ a weighted graph G, where every edge (i, j) has weight $w_{i,j}$. Consider the weighted adjacency matrix where for every edge (i, j)*

$$A_{i,j} = A_{j,i} = w_{i,j},$$

and the rest of the entries are zero. Also, let D be the diagonal matrix of vertex degrees; that is for each vertex i,

$$D_{i,i} = d_w(i) = \sum_{j \sim i} w_{i,j}$$

The Laplacian matrix of G

$$L_G = D - A.$$

In other words, it has degrees on the diagonal and the off-diagonal entries $L_{i,j}$ is $-w_{i,j}$ if there is an edge and 0 otherwise.

For unweighted graph G, the quadratic form of L_G is

$$\begin{aligned} x^T L_G x &= x^T (D - A)x \\ &= \sum_i d(i)x_i^2 - \sum_{i \sim j} x_i x_j \\ &= \sum_i d(i)x_i^2 - 2\sum_{i \sim j : i < j} x_i x_j = \sum_{i,j \in G} (x_i - x_j)^2 \end{aligned}$$

Similarly for a weighted graph:

$$x^T L_G x = \sum_{i,j \in G} w_{i,j}(x_i - x_j)^2 \geq 0 \tag{10.3}$$

Then the following fact is immediate.

Fact 10.5. *For any (weighted) graph G, L_G is a PSD matrix, $L_G \succeq 0$*

Proof. By variational characterization of eigenvalues (see Lecture 8, section 2), L_G's minimum eigenvalue is

$$\lambda_{min}(L_G) = \min_x \frac{x^T L_G x}{x^T x} = \min_x \frac{\sum_{i\sim j}(x_i - x_j)^2}{x^T x} \geq 0.$$

where we used that the numerator is always nonnegative. □

Fact 10.6. $\lambda_{min}(L_G) = 0$

Proof. Let $x = \vec{1}$ be the all ones vectors. Then,

$$\sum_{i\sim j}(x_i - x_j)^2 = \sum_{i\sim j}(1-1)^2 = 0.$$

Since by Fact 10.5, $y^T L_G y \geq 0$ for any vector y, by variational characterization of eigenvalues, the smallest eigenvalue of L_G is zero. □

Lemma 10.7. *Let λ_2 be the second smallest eigenvalue of L_G. Then, $\lambda_2 = 0$, if and only if G is disconnected.*

The above lemma shows a very nice connection between a combinatorial property of a graph and an algebraic property of its matrices. Namely, we can test whether a given graph is connected without running any graph search algorithm, include BFS or DFS. All we need to do is to test if the second eigenvalue of Laplacian matrix is nonzero. In the future lectures we see efficient algorithms to approximately compute the second eigenvalue of the Laplacian matrix.

Proof. First, we prove that if G is not connected, then $\lambda_2 = 0$. Suppose G is disconnected vertex and S and there is no edge between $\overline{S}$, i.e.,

$$|E(S, \overline{S})| = 0.$$

We construct x as:

$$x_i = \begin{cases} \frac{1}{|S|} & \text{if } i \in S \\ -\frac{1}{|\bar{S}|} & \text{if } i \in \bar{S} \end{cases}$$

Since x assign same number to all vertices of S, and all vertices of $\overline{S}$, for all $(i,j) \in E$, we have $x_i = x_j$, so

$$x^T L_G x = \sum_{i\sim j}(x_i - x_j)^2 = 0.$$

Note that in the above we also use that there is no edge between $S, \overline{S}$. Since the first eigenvector of L_G is the all ones vector, by variational characterization,

$$\lambda_2 = \min_{x:\langle x, \mathbf{1}\rangle = 0} \frac{x^T L_G x}{x^T x}.$$

Since, the above vector x is orthogonal to $\mathbf{1}$, $\lambda_2 = 0$

Conversely, suppose $\lambda_2 = 0$; we show that G is disconnected. Let x be the eigenvector associates with λ_2. Then, by variational characterization, $\langle x, \mathbf{1}\rangle = 0$ and

$$x^T L_G x = \sum_{i\sim j}(x_i - x_j)^2 = 0.$$

The RHS implies that for all $i \sim j$,

$$(x_i - x_j)^2 = 0 \Rightarrow x_i = x_j.$$

But this means that any two vertices i, j which are in the same connected component of G we have $x_i = x_j$ since equality means transitive. So to show that G has two connected components it is enough to show that x assigns two distinct values to the vertices of G, or x is not a constant vector. But, since x is orthogonal to $\vec{1}$. there is i, j such that $x_i \neq x_j$ as desired. $\square$

We will use above properties for spectral graph theory to design a spectral partitioning algorithm.