CSE 521: Design and Analysis of Algorithms I **Fall 2018**

Lecture 12: Clustering, Spectral Partitioning

Lecturer: Shayan Oveis Gharan *11/07/18*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

12.1 Importance Sampling

Say we have a set of numbers of $A_1, \ldots, A_n$ and we want to construct an unbiased estimator of $\sum_i A_i$. Here is a general framework to construct an unbiased estimator: Choose a probability vector $p_1, \ldots, p_n$ of nonnegative numbers such that

$$\sum_i p_i = 1.$$

Then, let a random variable $X = A_i/p_i$ with probability p_i. It follows that

$$\mathbb{E}[X] = \sum_i p_i \cdot \frac{A_i}{p_i} = \sum_i A_i$$

So, for every choice of the probability vector we will obtain an unbiased estimator. Ideally, we would like to choose the probability vector to minimize the variance of X. Let us first calculate the second moment:

$$\mathbb{E}\left[X^2\right] = \sum_i p_i \cdot \frac{A_i^2}{p_i^2} = \sum_i \frac{A_i^2}{p_i}$$

Perhaps, the first idea is to let $p_1, \ldots, p_n$ be a uniform distribution, i.e., $p_i = 1/n$. But in such a case, we get

$$\mathbb{E}\left[X^2\right] = n \sum_i A_i^2$$

Note that this can be much larger than $\mathbb{E}[X]$ in the regime where A_i's can be negative and we have cancellations. Now, suppose instead we let $p_i \propto A_i^2$, i.e., $p_i = \frac{A_i^2}{\sum_j A_j^2}$. The intuition is that we choose the i-th number with probability related to its magnitude. If a number is more important it is more likely to be chosen. Then, we get

$$\mathbb{E}\left[X^2\right] = \sum_i \frac{A_i^2}{A_i^2/\sum_j A_j^2} = \sum_i A_i^2.$$

So, we improve the second moment of X by a factor of n. We will see many applications of this idea in PS3.

12.2 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:

$$\max_{x \in \{0,1\}^n} x^T A(1 - x) \tag{12.1}$$

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 12.1. *Let as $k > 1$ be an integer, for any matrix $A \in \{0,1\}^{n\times n}$, we can approximate (12.1) 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 12.2. *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 $O(k^k \,\mathrm{poly}(n))$.*

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 12.1 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 12.1, first we approximate A with rank k matrix, A_k. Secondly we solve the optimization problem (12.1) 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.

12.2.1 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 12.3. *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 $\lambda_1 \geq \lambda_2, \ldots$ 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.

12.2.2 Step 2

In the second step we approximately solve (12.1) 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,

$$x^T A_k(1-x) = x^T \left(\sum_{i=1}^{k} \lambda_i v_i v_i^T \right) (1-x) = \sum_{i=1}^{k} \lambda_i \langle v_i, x\rangle \langle v_i, 1-x\rangle. \tag{12.2}$$

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 $v_1(S), \ldots, v_k(S), v_1(\overline{S}), \ldots, v_k(\overline{S})$. 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 (12.1) 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 $|\tilde{v}_i(S) - v_i(S)| \leq \epsilon$ for all i. Then,

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

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

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

Also, by Cauchy-Schwarz inequality,

$$\sum_{i}^{k} \lambda_i \leq \sqrt{k} \cdot \sqrt{\sum_{i}^{k} \lambda_i^2} \leq \sqrt{k} \cdot \|A\|_F = \sqrt{k|E|} \leq n\sqrt{k}.$$

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 $-\sqrt{n} \leq v_i(S) \leq \sqrt{n}$,

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

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.