CSE 521: Design and Analysis of Algorithms I **Spring 2016**

Lecture 10: Applications of Low Rank Approximation in Optimization

Lecturer: Shayan Oveis Gharan *April 27th* *Scribe:*

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

In this lecture we describe applications of low rank approximation in optimization. Firstly, let us give a short overview of the last lecture. We defined the operator norm of a matrix $\|.\|_2$ and the Frobenius norm $\|.\|_F$ and we showed that the best rank k approximation of a given matrix M is the matrix that chooses singular vectors corresponding to the largest k singular value of M.

Let us give a geometric view of SVD and low rank approximation. Let $M = \sum_{i=1}^{n} \lambda_i v_i v_i^T$ be a symmetric matrix. Geometrically, we can view M as follows. Suppose we write the coordinate of each point in terms of the basis vectors $v_1, \ldots, v_n$. The circle in at the left of Figure 10.1 represents points of norm 1. If $M = 1.7v1v_1^T + 0.75v_2v_2^T$, the points on this circle map to the ellipse at the right. Note that since $\lambda_1 >> \lambda_2$, points which have a larger inner product with v_1 gets stretched. The operator norm of such a M is the longest amount that a point gets stretched under this linear map. So, in our example, a point along the v_1 direction gets stretched by 1.7 factor.

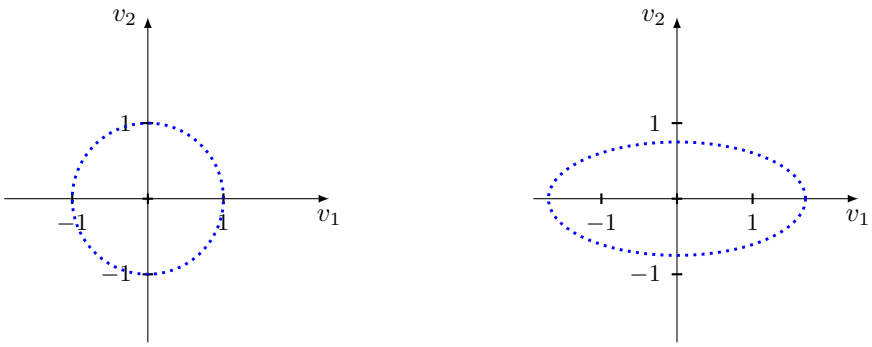

Figure 10.1: Consider a matrix $M = 1.7v_1v_1^T + 0.75v_2v_2^T$. This matrix maps the points (at distance 1 of the origin) on the left circle to the ellipse in the right.

In Figure 10.2, we can see how a nonsymmetric matrix act. Say $M = 1.7u_1v_1^T + 0.75u_2v_2^T$ and $\langle u_i, v_i \rangle = \sqrt{2}/2$. Note that a singular value of a nonsymmetric matrix satisfies the following identity:

$$Mv_i = \sigma_i u_i.$$

So, in this case we get a rotated ellipse and the rotation depends on the inner products of u_i, v_i's.

When we study low rank approximation of matrices, we are trying to approximate the above linear maps with a simpler mapping. For example, if we do a rank 1 approximation for the mappings in Figure 10.1 and Figure 10.2, we should only choose $1.7v_1v_1^T$ and $1.7u_1v_1^T$ respectively. With such a rank-1 mapping any point will be projected along v_1 and the projection value will be stretched by 1.7 factor. In particular, if we consider points which have 0 projection on v_1, they map to 0. The latter corresponds to the largest possible error in this rank-1 approximation.

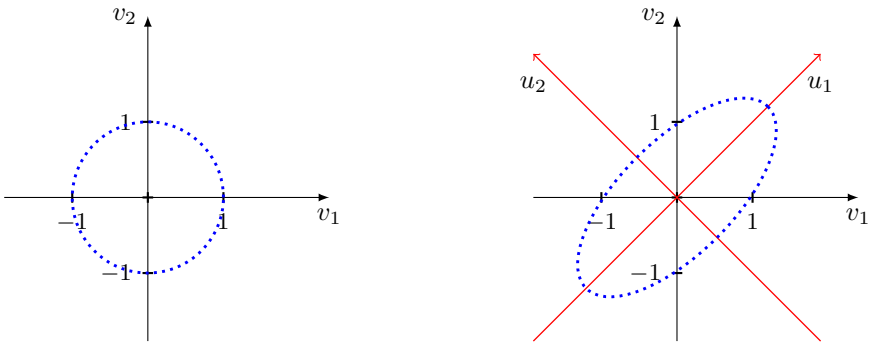

Figure 10.2: Consider a matrix $M = 1.7u_1v_1^T + 0.75u_2v_2^T$. This matrix maps the points (at distance 1 of the origin) on the left circle to the rotated ellipse in the right.

Other Norms. Low rank approximation can be also studied for other families of norms on matrices. A famous one is the weighted Frobenius norm,

$$\|M\|_{WF} = \sum_{i,j} W_{i,j} M_{i,j}^2.$$

where $W_{i,j} \geq 0$ for all i, j. Obtaining the best rank k approximation with respect to an arbitrary given W is an NP-hard problem in general, but it is possible to approximate it if W has low rank [RSW16].

Computation. Computing the SVD of a $m \times n$ matrix takes time $O(mn \min\{m, n\})$. So it is very time consuming in practice. Recently, there have been a number of results which manage to approximate the best rank-k approximation in almost linear time [CW13].

10.1 Applications of Low Rank Approximation in Optimization

In this section we see how one can use low rank approximation of a matrix to design an approximation algorithm for NP-hard optimization problems. We describe an algorithm for Max-cut, but as we will see the approach is quite general and can be extended to a wide range of optimization problem.

In an instance of the max-cut problem we are given a graph $G = (V, E)$ and we want to find a set S which maximizes $|E(S, \overline{S})|$.

Although the min-cut problem can be solved optimally, as we saw in the first lecture, max-cut is an NP-hard problem. The best approximation algorithm that we know for max-cut is by an algorithm of Goemans and Williamson [GW95]. They showed that there is a polynomial time algorithm which always return a cut $(T, \overline{T})$ such that

$$|E(T, \overline{T})| \geq 0.87 \max_S |E(S, \overline{S})|.$$

Let G be a graph with n vertices. It is well known that max-cut can be solved up to very small errors in dense graphs, i.e., graphs with average degree $c \cdot n$ for some constant $c > 0$. In this section we describe one such algorithm which exploits the low rank approximation of the adjacency matrix of G. Recall that the adjacency matrix of G, A is a symmetric matrix defined as follows:

$$A_{i,j} = \begin{cases} 1 & \text{if} i \sim j \\ 0 & \text{otherwise.} \end{cases}$$

We sketch the proof of the following theorem. There is an algorithm that for any given graph G and integer $k > 1$ returns a cut $(T, \overline{T})$ such that

$$|E(T, \overline{T})| \geq \max_S |E(S, \overline{S})| - \frac{n^2}{\sqrt{k}},$$

in time $O(n \cdot k^k)$.

So, the above algorithms gives an additive $n^2/\sqrt{k}$ approximation to the maximum cut. This approximation factor is useless of the size of the optimum cut is less than $n^2/\sqrt{k}$. Note that k is a parameter that we can use. By choosing a larger value for k, we improve the approximation factor of our algorithm at the cost of having a slower algorithm. Also, observe that the running time of the algorithm exponentially depends on k; this is unavoidable because max-cut problem is NP-hard in sparse graphs.

Firstly, we formulate the max-cut problem algebraically. For a set $S \subseteq V$, let

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

be the indicator vector of the set S. We claim that for any set $S \subseteq V$,

$$|E(S, \overline{S})| = \mathbf{1}^{S^T} A \mathbf{1}^{\overline{S}}. \tag{10.1}$$

To see the above, first observe that for any two vectors x, y,

$$x^T A y = \sum_{i,j} x_i A_{i,j} y_j.$$

So,

$$\mathbf{1}^{S^T} A \mathbf{1}^{\overline{S}} = \sum_{i,j} \mathbf{1}_i^S A_{i,j} \mathbf{1}_j^{\overline{S}} = \sum_{i,j} \mathbb{I}\left[i \in S\right] \mathbb{I}\left[i \sim j\right] \mathbb{I}\left[j \in \overline{S}\right] = |E(S, \overline{S})|.$$

This proves (10.1). In addition, observe that $\mathbf{1}^{\overline{S}} = 1 - \mathbf{1}^S$. Therefore, we can rewrite the max-cut problem as the following algebraic question:

$$\max_S |E(S, \overline{S})| = \max_{x \in \{0,1\}^n} x^T A (1 - x). \tag{10.2}$$

Now, let us describe the high-level strategy. First, we approximate A by a rank-k matrix A_k and we show that for any vector $x \in \{0,1\}^n$,

$$|x^T A(1-x) - x^T A_k (1-x)| = |x^T (A - A_k)(1-x)| \leq n^2/\sqrt{k}. \tag{10.3}$$

This shows that all we need to do is $\max_{x \in \{0,1\}^n} x^T A_k (1-x)$. In the second step we use the fact that A_k has rank k to solve the latter problem in time $O(n \cdot k^k)$.

Step 1. Let $A = \sum_{i=1}^m \sigma_i u_i v_i^T$ where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$. Note that although A is a symmetric matrix, we do not use that fact to emphasize the general proof idea. Also, let $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$. We can write,

$$\begin{aligned} |x^T (A - A_k)(1-x)| &= |\langle x, (A - A_k)(1-x) \rangle| \\ &\leq \|x\| \cdot \|(A - A_k)(1-x)\|_2 \\ &\leq \|x\| \cdot \|A - A_k\|_2 \cdot \|1 - x\| \leq n \sigma_{k+1} \end{aligned} \tag{10.4}$$

The first inequality uses the fact that the inner product of any two vectors is no larger than the product of their norms and the second inequality follows from the definition of the operator norm. In particular, recall that

$$\|A - A_k\|_2 = \max_y \frac{\|(A-A_k)y\|}{\|y\|} \Rightarrow \forall y : \|(A-A_k)y\| \leq \|A - A_k\|_2 \cdot \|y\|.$$

Also, note that (10.4) follows from the facts that $\|x\|, \|1-x\| \leq \sqrt{n}$ because $x, 1-x \in \{0,1\}^n$.

To prove (10.3), we need to upper bound σ_{k+1} by $n/\sqrt{k}$. We use the Frobenius norm of A to prove this inequality. Observe that

$$\begin{aligned}
\sigma_{k+1}^2 &\leq \frac{\sigma_1^2 + \cdots + \sigma_{k+1}^2}{k+1} \\
&\leq \frac{\sigma_1^2 + \cdots + \sigma_m^2}{k+1} \\
&= \frac{\|A\|_F^2}{k+1} \leq \frac{n^2}{k+1}.
\end{aligned}$$

The last inequality uses that $A \in \{0,1\}^{n \times n}$ matrix. Putting the above inequality together with (10.4) proves (10.3).

Step 2. In this step we want to approximate

$$\max_{x \in \{0,1\}^n} x^T A_k (1-x).$$

First, by the definition of A_k, we can write

$$x^T A(1-x) = x^T \left(\sum_{i=1}^k \sigma_i u_i v_i^T \right) (1-x) = \sum_{i=1}^k \sigma_i \langle u_i, x \rangle \langle v_i, 1-x \rangle.$$

So, it is enough to approximate $2k$ numbers, the inner products of x with all u_i, v_i's. Recall that we are thinking of k as a small number, say 3 or 4. For a set $S \subseteq V$, let

$$u_i(S) = \sum_{j \in S} u_i(j),$$

and let

$$w(S) = \left(u_1(S), v_1(\overline{S}), u_2(S), v_2(\overline{S}), \ldots, u_k(S), v_k(\overline{S}) \right).$$

All we need to do is to approximate $w(S)$ vectors and find the one where $\sum_{i=1}^k \sigma_i u_i(S) v_i(\overline{S})$ is maximized. Let $\epsilon = O(\sqrt{n}/k)$. For each i, S, we round $u_i(S)$ to the nearest multiple of ϵ and we let $\tilde{u}_i(S)$ be that approximate value. It is not hard to see that for any set S,

$$\left| \sum_{i=1}^k \sigma_i u_i(S) v_i(S) - \sum_{i=1}^k \sigma_i \tilde{u}_i(S) \tilde{v}_i(S) \right| \leq O\left((\epsilon \cdot \sum \sigma_i \cdot \max_{1 \leq i \leq k} \{u_i(S), v_i(S)\} \right) = O(n^2/\sqrt{k}),$$

where we used that $u_i(S) \leq \|u_i\| \cdot \|\mathbf{1}^S\| \leq \sqrt{n}$.

So, it is enough to find

$$\max_S \sum_{i=1}^k \sigma_i \tilde{u}_i(S) \tilde{v}_i(S).$$

Now, note that for each i, S, $|\tilde{u}_i(S)| \leq \sqrt{n}$. So, each $\tilde{u}_i(S)$ can take one of $2k$ possible values (recall $\epsilon = O(\sqrt{n}/k)$. And, the vector

$$\tilde{w}(S) = (\tilde{u}_1(S), \tilde{v}_1(S), \ldots, \tilde{u}_k(S), \tilde{v}_k(S)),$$

can take no more than $O(k^k)$ different values. Note that some of these k^k possible values may not be achievable by any of the 2^n sets. So, all we need to do is to go over all of these k^k possibilities and see if there is a set S, where $\tilde{w}(S)$ is the point we are looking for. The naïve way of doing that takes time which is exponential in n.

Instead, we can use dynamic programming. Here is the inductive step of the dynamic program. Say

$$P(j) = \{\tilde{w}(S) : S \subseteq \{1, 2, \ldots, j\}\}$$

be the set of points attainable with sets which are subsets of $[j]$. To compute $P(j+1)$ we need to go over all vectors in $P(j)$ and add the $j+1$ points to them; that leads to a new set points points, say

$$P'(j) = \{\tilde{w}(S \cup \{j+1\}) : S \subseteq \{1, 2, \ldots, j\}\}.$$

We let $P(j+1) = P(j) \cup P'(j)$. Since for each j, $|P(j)| \leq k^k$ the update takes time $O(n \cdot k^k)$. At the end of the day, we compute

$$\max_{S : \tilde{w}(S) \in P(n)} \sum_{i=1}^{k} \sigma_i \tilde{u}_i(S) \tilde{v}_i(\overline{S}).$$

This gives an $O(n^2/\sqrt{k})$ additive approximation of the max-cut. The running time of the algorithm is $O(n \cdot k^k)$.

References

[CW13] K. L. Clarkson and D. P. Woodruff. "Low rank approximation and regression in input sparsity time". In: *STOC*. ACM. 2013, pp. 81–90 (cit. on p. 10-2).

[GW95] M. X. Goemans and D. P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *J. ACM* 42.6 (Nov. 1995), pp. 1115–1145 (cit. on p. 10-2).

[RSW16] I. Razenshteyn, Z. Song, and D. P. Woodruff. "Weighted Low Rank Approximations with Provable Guarantees". In: *STOC*. 2016 (cit. on p. 10-2).