CS 761 : Randomized Algorithms , Spring 2018 , Waterloo

Lecture 20 : Randomized linear algebra

We study a generalization of the cut sparsification problem, called spectral sparsification, and see how this problem can be solved nicely using random sampling and a matrix Chernoff bound.

Then we discuss how related ideas can be used to design fast algorithms for the linear regression problem, a basic problem in numerical linear algebra.

Spectral sparsification

Recall that a graph H is a $(1\pm\varepsilon)$-cut approximator of G if $(1-\varepsilon)w_G(\delta(S)) \le w_H(\delta(S)) \le (1+\varepsilon)w_G(\delta(S))$ for all $S \subseteq V$, where $w_G(\delta(S))$ is the total weight of the edges crossing S.

We mentioned the result by Benczur and Karger that for any G, there exists a $(1\pm\varepsilon)$-cut approximator with $O(n\log n/\varepsilon^2)$ edges.

Today we will prove a spectral generalization of this result.

To state the result, first we recall some background from linear algebra.

Positive semidefinite matrix

A real symmetric matrix M is positive semidefinite if all its eigenvalues are non-negative.

We use the notation $M \succeq 0$ to denote that M is positive semidefinite.

Fact The following are equivalent. Let M be a real symmetric $n \times n$ matrix.

1. $M \succeq 0$
2. $x^T M x \ge 0 \quad \forall x \in \mathbb{R}^n$
3. $M = UU^T$ for some matrix $U \in \mathbb{R}^{n \times n}$.

For two positive semidefinite matrices A, B, we write $A \succeq B$ if $A - B \succeq 0$, or equivalently $x^T A x \ge x^T B x \ \forall x \in \mathbb{R}^n$.

This defines a partial ordering of symmetric matrices, called the Löwner ordering.

Laplacian matrix

Given an undirected graph with edge weight $w(e) \ge 0$ on each edge e, the (weighted) Laplacian matrix L is defined as $D - A$, where D is the diagonal (weighted) degree matrix where

$D_{ii} = \deg_w(i) = \sum_{j:ij \in E} w(i,j)$ and A is the (weighted) adjacency matrix where $A_{ij} = w(ij)$

Let L_e be the Laplacian matrix of an edge $e = ij$, i.e. $L_{ii} = L_{jj} = w(ij)$ and $L_{ij} = L_{ji} = -w(ij)$ and zero otherwise.

We can write $L_e = b_e b_e^T$, where $b_e \in \mathbb{R}^n$ is a vector with $\sqrt{w(ij)}$ in the i-th entry and $-\sqrt{w(ij)}$ in the j-th entry and zero otherwise.

It is easy to check that $L = \sum_{e \in E} L_e = \sum_{e \in E} b_e b_e^T$.

Note that the Laplacian matrix has a nice quadratic form:

$$x^T L x = x^T \left(\sum_{e \in E} L_e \right) x = x^T \left(\sum_{e \in E} b_e b_e^T \right) x = \sum_{e \in E} x^T b_e b_e^T x = \sum_{e \in E} \langle x, b_e \rangle^2 = \sum_{ij \in E} w(ij) (x_i - x_j)^2 \geq 0.$$

This implies that the Laplacian matrix of a graph is positive semidefinite.

Spectral approximator

We say a graph H is a $(1 \pm \varepsilon)$-spectral approximator of G if $(1-\varepsilon) L_G \preceq L_H \preceq (1+\varepsilon) L_G$, or equivalently $(1-\varepsilon) x^T L_G x \leq x^T L_H x \leq (1+\varepsilon) x^T L_G x \quad \forall x \in \mathbb{R}^n$ where n is the number of vertices.

Claim A $(1 \pm \varepsilon)$-spectral approximator is a $(1 \pm \varepsilon)$-cut approximator.

proof For $S \subseteq V$, let $x_S \in \mathbb{R}^n$ be the vector with $x_S(i) = 1$ if $i \in S$ and zero otherwise.

Then, $x_S^T L_G x_S = \sum_{ij \in E} w(ij) (x_S(i) - x_S(j))^2 = w_G(\delta(S))$ and similarly $x_S^T L_H x_S = w_H(\delta(S))$.

Since H is a $(1 \pm \varepsilon)$-spectral approximator of G, we have

$(1-\varepsilon) x_S^T L_G x_S \leq x_S^T L_H x_S \leq (1+\varepsilon) x_S^T L_G x_S \quad \forall S \subseteq V$ and thus $(1-\varepsilon) w_G(\delta(S)) \leq w_H(\delta(S)) \leq (1+\varepsilon) w_G(\delta(S)) \quad \forall S \subseteq V$. □

The following theorem by Spielman and Srivastava thus generalizes the result of Benczur and Karger.

Theorem Any graph has a $(1 \pm \varepsilon)$-spectral approximator with $O(n \log n / \varepsilon^2)$ edges.

Reduction

The spectral sparsification result can be reduced to the following purely linear algebraic result.

Theorem Suppose $v_1, \ldots, v_m \in \mathbb{R}^n$ are given with $\sum_{i=1}^m v_i v_i^T = I_n$.

There exist scalars $s_1, \ldots, s_m$ with at most $O(n \log n / \varepsilon^2)$ non-zeros such that

$$(1-\varepsilon) I_n \leq \sum_{i=1}^m s_i v_i v_i^T \leq (1+\varepsilon) I_n.$$

We sketch the proof of the reduction of the spectral sparsification result to the above result.

The idea is to apply a linear transformation so that the Laplacian matrix becomes the identity matrix.

Let M be a positive semidefinite matrix with eigen-decomposition $M = \sum_{i=1}^n \lambda_i u_i u_i^T$.

The pseudo-inverse of M is defined as $M^{\dagger} = \sum \frac{1}{\lambda_i} u_i u_i^T$ and $M^{\dagger/2} = \sum \frac{1}{\sqrt{\lambda_i}} u_i u_i^T$

Let M be a positive semidefinite matrix with eigen-decomposition $M = \sum_{i=1}^{n} \lambda_i u_i u_i^T$.

The pseudo-inverse of M is defined as $M^{\dagger} = \sum_{i:\lambda_i > 0} \frac{1}{\lambda_i} u_i u_i^T$, and $M^{\dagger/2} = \sum_{i:\lambda_i>0} \frac{1}{\sqrt{\lambda_i}} u_i u_i^T$.

Given $L_G = \sum_{e \in E} L_e = \sum_{e \in E} b_e b_e^T$, we consider $I = L_G^{\dagger/2} L_G L_G^{\dagger/2} = \sum_{e \in E} (L_G^{\dagger/2} b_e)(b_e^T L_G^{\dagger/2}) = \sum_{e \in E} v_e v_e^T$,

where we define $v_e = L_G^{\dagger/2} b_e \quad \forall e \in E$.

Apply the above theorem gives us s_e with at most $O(n \log n / \varepsilon^2)$ non-zeros so that

$$(1-\varepsilon) I \leq \sum_{e \in E} s_e v_e v_e^T \leq (1+\varepsilon) I .$$

Now, multipling $L_G^{1/2}$ on the left and right gives us $(1-\varepsilon) L_G \leq \sum_{e \in E} s_e b_e b_e^T \leq (1+\varepsilon) L_G$, so by scaling the weight of each edge by a factor of s_e, we get our spectral sparsifier.

The above "proof" is not precise as we are dealing with the pseudo-inverse (but not the inverse), but the missing details are rather routine and is not the important part of the proof, and so omitted.

Sampling algorithm

Now, our focus is to prove the linear algebraic result, by random sampling.

First, we get some intuition about the condition $\sum_{i=1}^{m} v_i v_i^T = I_n$.

When $m=n$, then $v_1, \ldots, v_n$ must be an orthonormal basis.

When $m>n$, we can also think of it as an "overcomplete" basis, as we can write any $x \in \mathbb{R}^n$ as

$$x = I_n x = \left(\sum_{i=1}^{m} v_i v_i^T\right) x = \sum_{i=1}^{m} \langle x, v_i \rangle v_i .$$

Similarly, for any unit vector $y \in \mathbb{R}^n$, we have $1 = y^T y = y^T I y = y^T \left(\sum_{i=1}^{m} v_i v_i^T\right) y = \sum_{i=1}^{m} y^T v_i v_i^T y = \sum_{i=1}^{m} \langle v_i, y \rangle^2$.

Intuitively, the vectors are "evenly spread out", so that the projection of any direction y to these vectors are the same.

Idea: Given $\sum_{i=1}^{m} v_i v_i^T = I_n$, we would like to find a small subset of vectors $S \subseteq \{1, \ldots, m\}$ and some scaling factors so that $\sum_{i \in S} s_i v_i v_i^T \approx I_n$.

So, the subsets should still be "evenly spread out", with contributions in each direction about the same.

As in the graph sparsification case, uniform sampling won't work. For example, if some v_j has $\|v_j\| = 1$, then we must include v_j in the solution, as otherwise that direction will not be covered in the solution and so it won't be a spectral sparsifier. The analogy in the graph sparsification result is that a cut edge must be included in any sparsifier.

So, as in the graph sparsification case, we need to do non-uniform sampling (if we do random sampling).

The idea is similar: for longer vectors, the sampling probability is higher; for shorter vectors, we can be more aggressive in setting the sampling probability to be smaller, and when we choose them, we reweight the vector so that it has the correct expected value.

More concretely, we sample each vector v_i with probability $\|v_i\|_2^2$, and if it is chosen, we set the scaler $s_i = \frac{1}{\|v_i\|_2^2}$, so that $E[s_i v_i v_i^T] = \frac{v_i v_i^T}{\|v_i\|_2^2} \cdot \Pr(v_i \text{ is chosen}) = \frac{v_i v_i^T}{\|v_i\|_2^2} \cdot \|v_i\|_2^2 = v_i v_i^T$.

Algorithm

The actual algorithm is basically the same as described above, but we need to repeat this experiment $C = \Theta(\log n)$ times and take the average, so that we can prove concentration.

- Initially, $F \leftarrow \phi$, $s \leftarrow 0$, $C = \frac{6 \log n}{\varepsilon^2}$
- For $1 \le t \le C$ do

  For each $e \in E$, with probability $p_i = \|v_i\|_2^2$, update $F \leftarrow F \cup \{i\}$ and $s_i \leftarrow s_i + \frac{1}{C p_e}$.
- Return $\sum_{i \in F} s_i v_i v_i^T$ as our spectral approximator.

Analysis

There are two steps in the analysis.

One is to show that there are $O(n \log n / \varepsilon^2)$ non-zeros scalars, i.e. $|F| = O(n \log n / \varepsilon^2)$.

Another is to show that the returned solution is a $(1 \pm \varepsilon)$-spectral sparsifier.

We first bound the number of non-zeros scalars.

Claim With probability at least 0.9, $|F| = O(n \log n / \varepsilon^2)$.

Proof The expected value is $E[|F|] = \sum_{i=1}^m \Pr(\text{vector } i \text{ is in } F) = \sum_{i=1}^m (1 - (1-p_i)^C) \le \sum_{i=1}^m (1 - (1 - C p_i)) = C \cdot \sum_{i=1}^m p_i$,

which can also be seen by a union bound.

Note that $\sum_{i=1}^m p_i = \sum_{i=1}^m \|v_i\|_2^2 = \sum_{i=1}^m v_i^T v_i = \sum_{i=1}^m \mathrm{tr}(v_i^T v_i) = \sum_{i=1}^m \mathrm{tr}(v_i v_i^T) = \mathrm{tr}\left(\sum_{i=1}^m v_i v_i^T\right) = \mathrm{tr}(I_n) = n$, where

$\mathrm{tr}(A) = \sum_j A_{jj}$ and we use the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ (or directly check that $v^T v = \mathrm{tr}(v v^T)$).

Therefore, $E[|F|] \le C \sum_{i=1}^m p_i = Cn = 6 n \log n / \varepsilon^2$. The result follows from Markov's inequality. □

Matrix Chernoff bound

There is an elegant generalization of the Chernoff-Hoeffding bound to the matrix setting.

Theorem (Tropp) Let $X_1, \dots, X_k$ be independent, $n \times n$ symmetric matrices with $0 \preceq X_i \preceq RI$.

Let $\mu_{min} I \preceq \sum_{i=1}^{k} E[X_i] \preceq \mu_{max} I$. For any $\varepsilon \in [0,1]$,

- $Pr\left(\lambda_{max}\left(\sum_{i=1}^{k} X_i\right) \geq (1+\varepsilon)\mu_{max}\right) \leq n e^{-\frac{\varepsilon^2 \cdot \mu_{max}}{2R}}$
- $Pr\left(\lambda_{min}\left(\sum_{i=1}^{k} X_i\right) \leq (1-\varepsilon)\mu_{min}\right) \leq n e^{-\frac{\varepsilon^2 \cdot \mu_{min}}{2R}}$.

Note that it is almost an exact analog of the Chernoff-Hoeffing bound in the scalar case, by using the maximum eigenvalue and minimum eigenvalue to measure the "size" of a matrix.

It says that if we consider the sum of independent random matrices, where each matrix is not too "big/influential", then the sum is concentrated around the expectation in terms of the eigenvalues.

Concentration

The proof that our solution is a $(1\pm\varepsilon)$-spectral sparsifier is a direct application of the matrix Chernoff bound.

The random variables are $X_{i,t} = \begin{cases} \frac{v_i v_i^T}{C p_i} & \text{with probability } p_i = \|v_i\|_2^2, \text{ for vector } i \text{ in iteration } t. \\ 0 & \text{otherwise} \end{cases}$

Note that the output of the algorithm is $S := \sum_{t=1}^{C} \sum_{i=1}^{m} X_{i,t}$.

As discussed before, $E[S] = \sum_{t=1}^{C} \sum_{i=1}^{m} E[X_{i,t}] = \sum_{t=1}^{C} \sum_{i=1}^{m} \frac{v_i v_i^T}{C p_i} \cdot p_i = \sum_{t=1}^{C} \sum_{i=1}^{m} \frac{v_i v_i^T}{C} = \sum_{i=1}^{m} v_i v_i^T = I$.

So, the expected value is correct, with $\mu_{max} = \mu_{min} = 1$ in this problem.

To apply the matrix Chernoff bound, we just need to find a bound for R so that $X_{i,t} \preceq RI$.

Note that $X_{i,t} = \frac{v_i v_i^T}{C p_i} = \frac{v_i v_i^T}{C \|v_i\|_2^2} = \frac{1}{C}\left(\frac{v_i}{\|v_i\|}\right)\left(\frac{v_i}{\|v_i\|}\right)^T$. This is a rank one matrix of a unit vector, and so the maximum eigenvalue is just $\frac{1}{C}$ (with the only eigenvector being $\frac{v_i}{\|v_i\|}$). So, $R = \frac{1}{C}$.

By Tropp's theorem, we get $Pr(\lambda_{max}(S) \geq 1+\varepsilon) \leq n e^{-\varepsilon^2 C/3} = n e^{-2\log n} = \frac{1}{n}$, as $C = 6\log n / \varepsilon^2$.

The lower tail follows similarly.

So, with probability at least $1 - \frac{2}{n}$, we have $\lambda_{max}(S) \geq 1+\varepsilon$ and $\lambda_{min}(S) \leq 1-\varepsilon$, and so $(1-\varepsilon) I \preceq S \preceq (1+\varepsilon) I$, proving that our solution S is a $(1\pm\varepsilon)$ spectral sparsifier of I_n.

By a union bound, we know that a $(1\pm\varepsilon)$-spectral sparsifier with $O(n \log n / \varepsilon^2)$ edges exist, and indeed the random sampling algorithm will succeed with high probability, proving the theorem.

Discussions

There are a few things to discuss about.

① By considering this linear algebraic generalization of cut sparsification, we have a clean and arguably simple proof of the result of Benczur and Karger.

A subsequent amazing result by Batson, Spielman and Srivastava proves that every graph has a $(1\pm\varepsilon)$-spectral sparsifier with $O(n/\varepsilon^2)$ edges, which is best possible.

We don't know of an alternative (combinatorial) proof to achieve the same bound even for cut approximator (a special case).

This linear algebraic perspective seems to be the correct way to look at the problem.

② The sampling probability p_e is directly proportional to the effective resistance of the edge e.

Recall that $p_e = \|v_e\|_2^2 = \|L_G^{\dagger} b_e\|_2^2 = b_e^T L_G^{\dagger} b_e$. Let $e=uv$.

Note that $L_G^{\dagger} b_e$ is a solution x to $L_G x = b$, which is the potential vector $\vec{\phi}$ of the electrical flow problem when one unit of electrical flow is sent from u to v.

Then, $b_e^T L_G^{\dagger} b_e = b_e^T \vec{\phi} = \phi(u) - \phi(v)$ is just the definition of $R_{eff}(u,v)$.

So, the sampling algorithm works by sampling each edge with probability proportional to its effective resistance, a somewhat surprising application of this concept (though it makes sense).

③ There is a nearly linear time algorithm to estimate the effective resistances of all edges.

The main tools are a near linear time algorithm to solve a Laplacian system of equations (another breakthrough result by Spielman and Teng), and also dimension reduction.

So, we have a near linear time (randomized) algorithm for constructing spectral sparsifiers.

These results are also within the scope of this lecture, but we don't have time.

④ The analysis of the random sampling algorithm is tight.

In a complete graph, the effective resistance of every edge is the same, as the graph is symmetric.

So, the random sampling algorithm on a complete graph is just the uniform sampling algorithm.

We know from homework 1 (although informally) that it won't work with $o(n\log n/\varepsilon^2)$ edges.

Fast linear regression

Random sampling and dimension reduction are widely used in designing fast algorithms for numerical linear algebra problems.

We illustrate these ideas in a basic problem, the least square problem.

In the least square problem, we are given an $n \times d$ matrix A and $b \in \mathbb{R}^n$, and the objective is to find an $x \in \mathbb{R}^d$ to minimize $\|Ax-b\|_2$.

We think of $n \gg d$, so the problem is over-constrained.

To solve it exactly, the runtime is $\Omega(n\,\text{poly}(d))$, which is too slow for large n.

We would like to find an approximation algorithm with $\|Ax'-b\|_2 \le (1+\varepsilon) \min_x \|Ax-b\|_2$ in $\tilde{O}(nd + \text{poly}(d/\varepsilon))$ time, which is near linear when $n \gg d$.

The idea is to use a near-linear time algorithm to compress the matrix A into a $k \times d$ matrix A' with $k = \text{poly}(d/\varepsilon)$, and then solve $\|A'x-b\|_2$ exactly as our approximate solution.

This is in a similar spirit to the fast matrix rank algorithm that we have seen in L09.

There are two approaches to do the compression, both of which we have learnt.

In the following we just sketch the main ideas.

Dimension reduction

The following is an important concept useful in various numerical linear algebra problem.

Definition (Subspace embedding) A $(1\pm\varepsilon)$ ℓ_2-Subspace embedding for the column space of an $n \times d$ matrix is a matrix S for which $(1-\varepsilon)\|Ax\|_2^2 \le \|SAx\|_2^2 \le (1+\varepsilon)\|Ax\|_2^2 \quad \forall x \in \mathbb{R}^d$.

Suppose we have such a matrix $k \times n$ matrix S with $k = \text{poly}(d/\varepsilon)$.

Then we just solve $\min_x \|SAx - Sb\|_2$ instead in $\text{poly}(d/\varepsilon)$ time and use this solution as our approximate solution, as $\|Ax-b\|_2^2 \le (1+\varepsilon)\|S(Ax-b)\|_2^2 \le (1+\varepsilon)\|S(Ax^*-b)\|_2^2 \le \frac{1+\varepsilon}{1-\varepsilon}\|Ax^*-b\|_2^2$

where x and x^* are the minimizers for the compressed and the original problem respectively.

As you may imagine, the Johnson-Lindenstrauss theorem will be useful here, i.e. $S = \frac{1}{\sqrt{k}} G$ where each entry is a standard normal random variable.

One technical detail is that in the Johnson-Lindenstrauss transform, for $k = O(\frac{1}{\varepsilon^2}\log(\frac{1}{\delta}))$, it works for one specific vector with probability $\ge 1-\delta$.

The subspace embedding requires that it works for all vectors in $\mathbb{R}^d$, and there are infinitely many.

The analysis is to show that if it works for an ε-net for some constant ε (i.e. a discretization of the unit sphere $\mathbb{S}^{d-1}$), then it works for all vectors.

And a standard "volume" argument shows that the ε-net is of size at most c^d for some constant c.

Therefore, the Johnson-Lindenstrauss transform will work if $k = \Theta(d/\varepsilon^2)$, by setting $\delta = \frac{1}{c^d}$ and union bound.

Another detail is to compute SA and Sb quickly - because matrix multiplication is too slow and so the compression itself took too much time already.

There are much research in fast dimension reduction, and it is possible to do the compression in near linear time in terms of the number of nonzeros of A.

One particularly nice result of Clarkson and Woodruff proves that a very sparse matrix S works: set $k = \Theta(\frac{d^2}{\varepsilon^2}\,\text{polylog}(\frac{d}{\varepsilon}))$, for each column, choose a random location, set it to be $+1$ with probability $\frac{1}{2}$ and -1 with probability $\frac{1}{2}$.

So, each column has only one non-zero entry, and the compression can be done very efficiently.

Row sampling

Another approach is similar to what we have seen in spectral sparsification.

Given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, we first reduce the problem to the case when the columns of A are orthonormal.

This is reminiscent to the reduction to the identity matrix in spectral sparsification, so that $A^TA = I_d$, or equivalently $\sum_{i=1}^{n} a_i a_i^T = I_d$ where a_i is the i-th row of A.

Then, we construct a matrix B by sampling and rescaling each row proportional to its length, so that $\sum_{i=1}^{n} s_i a_i a_i^T \approx I_d$ with only $O(d\log d/\varepsilon^2)$ nonzeros, i.e. B has $O(d\log d/\varepsilon^2)$ rows. where each row of B is $\sqrt{s_i}\,a_i$ so that $(1-\varepsilon)A^TA \le B^TB \le (1+\varepsilon)A^TA$.

It is a good subspace embedding as $\|Ax\|_2^2 \approx \|Bx\|_2^2$ because $x^TA^TAx \approx x^TB^TBx$.

All the technical details are very similar to those in spectral sparsification, e.g. matrix Chernoff bound.

The sampling probability is called the leverage score of a row, a generalization of effective resistance.

These ideas are very useful in numerical linear algebra, which have further applications in other areas such as optimization (e.g. fast interior point algorithms for linear programming crucially used many of these tricks).

References

- Spielman and Srivastava, Graph sparsification by effective resistance, 2008.
- Batson, Spielman, and Srivastava, Twice-Ramanujan Sparsifiers, 2009.
- Lecture notes by Nick Harvey on Tropp's inequality and spectral sparsification.
- Woodruff, Sketching as a tool for numerical linear algebra (chapter 2), a good survey.

Concluding remarks

In this course, we have learnt many of the basic concepts and techniques in the design and analyis of randomized algorithms, and also seen many important results in different domains.

Probably the most important part is that we have a fair bit of training in the low level technical details, so that we have developed some intuition and experience, which hopefully will be useful in our research.

There are still many topics that we have not covered, such as martingales, negative correlation, more concentration inequalities, derandomization, more randomized linear algebra (Laplacian solver), compressed sensing, more hashing (cuckoo hashing), complexity theory (PCP theorem), random matrix, etc.

But you should be able to pick up a new topic on your own now. :)

Thank you for a great term!