CS 466/666 Algorithm Design and Analysis . Spring 2020

Lecture 3: Applications of tail inequalities

We will see two interesting and important applications of tail inequalities.

The first is about graph sparsification and the second is about dimension reduction.

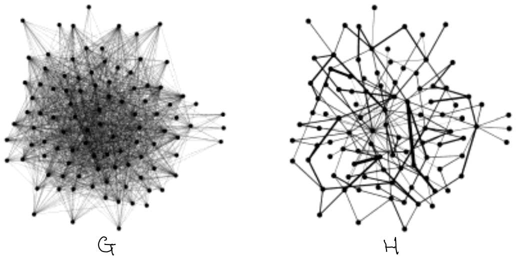
Graph sparsification

Given an undirected graph $G=(V,E)$ with a weight $w(e)$ on each edge $e\in E$, for a subset of vertices $S\subseteq V$, let $\delta_G(S)$ be the set of edges with one endpoint in S and one endpoint in $V-S$, and let $w(\delta_G(S)) = \sum_{e\in\delta_G(S)} w(e)$ be the total weight of the edges in $\delta_G(S)$.

We are interested in finding a "sparse" graph H that approximates the cut structures of G well.

Definition (ε-cut approximator) We say $H=(V,F)$ is an ε-cut approximator of $G=(V,E)$ if for all $S\subseteq V$ we have $(1-\varepsilon)\,w(\delta_G(S)) \le w(\delta_H(S)) \le (1+\varepsilon)\,w(\delta_G(S))$.

Note that G and H are defined on the same vertex set but could have different edge sets.



(picture from Nick Harvey's notes)

This turns out to be an important and beautiful problem with many interesting and surprising results.

Today we will see the first and simplest result in this direction, and mention one surprising application in designing near-linear time algorithm for minimum cut.

This first result needs a somewhat strong assumption on the input graph.

Assumption: The input graph $G=(V,E)$ is unweighted and has min-cut value $\Omega(\log|V|)$.

With this assumption, there is a very simple uniform sampling algorithm to sparsify a dense input graph.

Algorithm: Set a sampling probability p.

For every edge $e\in E(G)$, with probability p, put e in H with edge weight $w_e=\frac{1}{p}$.

The idea is to set the expectation right, so that we expect to choose p-fraction of edges, and make their weight to be $\frac{1}{p}$, so that the expected total weight in each cut in H is the same as that in G.

But, of course, it is not enough to have the expected values to be correct, as we need to ensure that all cuts in H have approximately the same weight as that in G simultaneously.

But, of course, it is not enough to have the expected values to be correct, as we need to ensure that all cuts in H have approximately the Same weight as that in G Simultaneously.

For this, we will use Chernoff bound and crucially the assumption that the min-cut value is $\Omega(\log n)$. where we recall that expected value $\Omega(\log n)$ is the regime that we can achieve tight concentration.

Theorem (Karger) Set $p = \frac{15 \ln n}{\varepsilon^2 c}$, where c is the min-cut value of G.

Then H is an ε-cut approximator of G with $O(p \cdot |E(G)|)$ edges with probability $\geq 1 - \frac{4}{n}$.

proof Consider a subset $S \subseteq V$. Say $\delta_G(S)$ has k edges. Note that $k \geq c$ by definition.

By linearity of expectation, $\mathbb{E}[|\delta_H(S)|] = \sum_{e \in \delta_G(S)} \mathbb{E}[x_e] = \sum_{e \in \delta_G(S)} (p \cdot 1 + (1-p) \cdot 0) = p|\delta_G(S)| = pk$ where x_e is an indicator variable where $x_e = 1$ if the edge e is added to H and $x_e = 0$ otherwise.

Similarly, $\mathbb{E}[w(\delta_H(S))] = \frac{1}{p} \cdot p \cdot |\delta_G(S)| = |\delta_G(S)| = k$, and so the expected values are what we want.

Next, we consider the probability that the actual value of $|\delta_H(S)|$ is "far" from the expectation.

Since $|\delta_H(S)|$ can be written as a sum of independent 0-1 variables (i.e. $\sum_{e \in \delta_G(S)} x_e$), we can apply Chernoff bound and get $\Pr(||\delta_H(S)| - p \cdot k| \geq \varepsilon pk) \leq 2e^{-\varepsilon^2 pk/3} = 2e^{-\frac{5k}{c}\ln n} = 2n^{-\frac{5k}{c}}$, where we used our choice of $p = 15 \ln n / (\varepsilon^2 c)$.

Recall that $k \geq c$ by definition, so the probability that $\delta_H(S)$ violates the requirement of an ε-cut approximator is at most n^{-5}, which is pretty small.

But there are exponentially many subsets of vertices $S \subseteq V$, and so a naive union bound won't work.

The important observation is that the probability that a large cut (i.e. k large) is violated is much smaller (i.e. $n^{-\frac{5k}{c}}$), and there are not many small cuts!

Lemma The number of cuts with at most αc edges for $\alpha \geq 1$ is at most $n^{2\alpha}$.

proof We have done the case when $\alpha = 1$. The general case is similar with a modification. The proof is left as an exercise or a homework problem. □

With this lemma, we can do a more careful union bound based on the size of the cuts:

Pr(some cut is violated)

$\leq \sum_{S \subseteq V} \Pr(\text{cut } S \text{ is violated})$ // union bound

$= \sum_{\alpha = 1, 2, 4, 8, \ldots} \sum_{S \subseteq V:\, \alpha c \leq |\delta_G(S)| \leq 2\alpha c} \Pr(\text{cut } S \text{ is violated})$ // dividing into cases based on the cut sizes grouped into $2^i c \leq |\delta_G(S)| \leq 2^{i+1} c$

$\leq \sum_{\alpha = 2^i:\, 0 \leq i \leq \infty} n^{4\alpha} \cdot \Pr(\text{cut } S \text{ is violated} \mid \alpha c \leq |\delta_G(S)| \leq 2\alpha c)$ // by the lemma

$\leq \sum_{\alpha = 2^i:\, 0 \leq i \leq \infty} n^{4\alpha} \cdot 2n^{-5\alpha c/c}$ // by the Chernoff bound above

$$= \sum_{\alpha = 2^i : 0 \le i \le \infty} 2n^{-\alpha}$$

$$\le 4/n \qquad // \text{ first term dominated}$$

Therefore, with probability at least $1-\frac{4}{n}$, H is an ε-cut approximator of G.

Finally, by a simple application of Chernoff bound, H has $O(p \cdot |E(G)|)$ edges with high probability.

This completes the proof. □

Remark Using Chernoff bound and union bound can already solve many interesting problems.

Example: This result is restrictive, since it assumes the graph has a somewhat large min-cut value.

But for graphs that are essentially complete (i.e. $c = \Omega(n)$), the theorem says that there is an ε-cut approximator with $O(n\log n / \varepsilon^2)$ edges, a significant sparsification from $\Omega(n^2)$ edges in the input graph.

Applications: One natural application is to design fast approximation algorithms for cut problems.

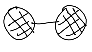
Take our favorite problem about cuts, say the minimum s-t cut problem.

The running time of the algorithms usually depend on $|E|$, which could be $\Omega(|V|^2)$.

To speedup, we first sparsify G by constructing an ε-cut approximator H with fewer edges.

Then, run an (exact) algorithm on H to find a minimum s-t cut $S \subseteq V$, and return S as our approximate min s-t cut on G, and it can be shown that it is a $(1+3\varepsilon)$-approximation.

This gives us a tradeoff between runtime and approximation guarantee.

Improvement by Benczur and Karger: Without the minimum cut assumption, it is easy to see that the same uniform sampling algorithm won't work, e.g in , it is very likely the cut edge is not chosen.

Benczur and Karger designed a very clever non-uniform sampling algorithm, where each edge is sampled with probability inversely proportional to the "connectivity" of the two endpoints.

They defined a notion called "strong connectivity" and proved that sampling with probability inversely proportional to it will result in an ε-cut approximator with $O(n\log n / \varepsilon^2)$ edges for any graph.

Furthermore, they showed how to compute this ε-cut sparsifier in near linear time, and this for example gives the first $\tilde{O}(n^2)$-time algorithm for computing approximate min s-t cut.

We will not discuss this result in more detail. Instead, we will study a stronger result on spectral sparsification when we come to the linear algebraic part of the course.

Minimum cuts in near linear time (optional, sketch)

An amazing application of Karger's sparsification result is a near-linear time algorithm for minimum cuts.

Let c be the min-cut value of the input graph G. So, G is c-edge-connected.

Karger cleverly used a classical result about spanning tree packing.

Theorem If G is c-edge-connected, then G has at least $\lfloor c/2 \rfloor$ edge-disjoint spanning trees.

Suppose we have edge-disjoint spanning trees $T_1, T_2, T_3, \dots, T_{\lfloor c/2 \rfloor}$.

Then, we know that there exists some tree T_i that crosses a minimum cut S at most 2 times.

Karger observed that having such a tree T_i would be very helpful in finding that minimum cut S, because one just needs to search over all cuts formed by removing two edges from T_i.

He came up with a sophisticated dynamic programming $\tilde{O}(m)$ time algorithm to find a minimum cut formed by removing at most two edges from T_i (the case of removing one edge is easier)

Okay, but how to find such a tree T_i?

There is an $\tilde{O}(m \cdot c)$ time algorithm to find c edge-disjoint spanning trees.

But this is too slow for our purpose.

The idea here, of course, is to do graph sparsification.

Using karger's graph sparsification result, we can sparsify the graph so that its min-cut value is $O(\log n)$ while preserving the value of each cut approximately.

Now, we can compute $T_1, T_2, \dots, T_{O(\log n)}$ in the sparsifier, and one of these trees will cross a minimum cut at most two times.

So, doing dynamic programming on each of these $O(\log n)$ trees would work, with total complexity $\tilde{O}(m)$!

This is just a sketch of the main ideas, with many details missing.

Dimension reduction

Given n points in the Euclidean space, we can always represent the vectors in n-dimensions.

In general, we cannot do better if no distortion is allowed (e.g. the standard basis).

Surprisingly, if we allow just a little distortion, then the number of dimensions can be significantly reduced.

Theorem (Johnson-Lindenstrauss Lemma) Given any $\varepsilon \in (0, \frac{1}{2})$ and any set of points $X = \{x_1, x_2, \dots, x_n\}$, there exists a map $A : X \to \mathbb{R}^k$ for $k = O(\frac{\log n}{\varepsilon^2})$ such that

$$1 - \varepsilon \le \frac{\| A(x_i) - A(x_j) \|_2^2}{\| x_i - x_j \|_2^2} \le 1 + \varepsilon.$$

Algorithm : The construction is very simple. It just projects the points in a random k-dimensional subspace.

Let d be the dimension of the original points.

Let M be a $k \times d$ matrix, such that each entry of M is drawn from the normal $N(0,1)$ distribution (Gaussian random variable with mean 0 and variance 1, with density $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.)

Define $A(x) = \frac{1}{\sqrt{k}} Mx$. This is efficiently computable.

Since A is a linear transformation (e.g. $A(x) - A(y) = A(x-y)$), the theorem can be reduced to the following.

Lemma If A is constructed by the above algorithm with $k = \Theta\left(\frac{1}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right)$,

then $\Pr\left(1-\varepsilon \le \|A(x)\|_2^2 \le 1+\varepsilon\right) \ge 1-\delta$ for any unit vector $x \in \mathbb{R}^d$ and any $\varepsilon \in (0, \frac{1}{2})$.

First, we see how the lemma implies the theorem.

We set $\delta = \frac{1}{n^2}$ and thus $k = \Theta\left(\frac{\log n}{\varepsilon^2}\right)$.

For any pair $i \ne j$, the squared length of $x_i - x_j$ is maintained to within $1 \pm \varepsilon$ with probability $\ge 1 - \frac{1}{n^2}$.

By the union bound, the distances of all pairs are maintained to within $1 \pm \varepsilon$ with probability $\ge \frac{1}{2}$.

Henceforth, we focus on proving the lemma.

Proof idea: Consider the elementary unit vector $e_1 = (1, 0, \dots, 0)$.

Then, Me_1 is just the first column of M, with independent and identical Gaussian values.

We are interested in the length of this column, which is the sum of squares of these Gaussians.

Note that $E\left[\sum_{i=1}^{k} M_{i,1}^2\right] = \sum_{i=1}^{k} E[M_{i,1}^2] = k$ as the variance of each $M_{i,j}$ is one, and so the expected length of Ae_1 is one as $E\left[\|Ae_1\|_2^2\right] = \frac{1}{k} E\left[\|Me_1\|_2^2\right] = 1$.

By setting k to be large enough, we expect that the length is highly concentrated around its expectation.

From our intuition of Chernoff bound, if we set $k = O\left(\frac{\log n}{\varepsilon^2}\right)$, then the error probability is at most $2e^{-\mu\varepsilon^2/3} \le \frac{1}{n^2}$.

Proof The actual proof is similar to the above idea. There are two issues to handle.

① We cannot assume $x = e_1$, and we need to deal with any x.

② The standard Chernoff-Hoeffding bound in L02 cannot be directly applied, because the random variables are unbounded (although with a small tail).

The first issue can be taken care of by the nice properties of Gaussian random variables.

Consider an arbitrary entry y_j of the vector Mx for an arbitrary unit vector x.

Then $y_j = \sum_{i=1}^{d} M_{ji} x_i$ where M_{ji} is an $N(0,1)$ random variable.

So, y_j is a sum of Gaussian variable, and it is a well-known fact that y_j is an $N(0, \sum_{i=1}^{d} x_i^2)$

random variable. Since x is a unit vector, y_j is just an $N(0,1)$ random variable.

So, each of the k coordinates of Mx is just independent Gaussian.

By the same argument as in the proof idea, the expected length of $\frac{1}{\sqrt{k}}Mx$ is one.

The second issue requires some work, to derive a Chernoff bound for Gaussian random variables.

By elementary calculus, we can compute the moment generating function of the sum of squares of independent Gaussians (i.e. $E[e^{tX^2}] = \frac{1}{\sqrt{1-2t}}$ for $t < \frac{1}{2}$ for $X \sim N(0,1)$) and use it to prove that $\Pr(\|Ax\|_2^2 \geq 1+\varepsilon) \leq e^{-k\varepsilon^2/8}$. The details are left as a (harder) exercise.

Similarly, we can bound the lower tail and get a similar result.

So, by setting $k = O(\frac{1}{\varepsilon^2}\ln(\frac{1}{\delta}))$, we have $\Pr(\left|\|Ax\|_2^2 - 1\right| > \varepsilon) \leq \delta$, proving the lemma. □

Remarks: The same result is true even when M is a random ± 1 matrix [Achiloptas].

The proof is more difficult but the algorithm is much easier to implement.

Actually, if we use a random ± 1 matrix, this is similar to what we will see in data streaming.

Applications: One immediate and important application is to do approximate near neighbor search.

A linear scan takes $O(n^2)$ time, but only $O(n\log n)$ time after dimension reduction.

Note that it works for Euclidean distances only (e.g. not for say L_1-distances).

Another application is approximate matrix multiplication.

Given two $n\times n$ matrices A and B, we do dimension reductions on the rows of A and the columns of B to $O(\log n)$ dimensions, so that the product can be done in $O(n^2\log n)$ time, while each entry is approximately the same as the inner products are approximated with high prob.

Project ideas: There are already many interesting project topics that we can start:

(1) concentration inequalities for randomized algorithms (2) graph sparsification and applications

(3) minimum cuts (4) metric embedding (5) randomized numerical linear algebra.

References

- Karger, Random sampling in cut, flow and network design problems, 1994.
- Benczur and Karger, Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time, 1996.
- Karger. Minimum cuts in near linear time, 2000.
- Lecture notes in discrete geometry, chapter 15, by Matousek. (Highly recommended.)