Randomized Algorithms: Lecture Notes

Fall 2025

Contents

Chapter 1

Introduction to Randomized Algorithms and Linearity of Expectation

1.1 Introduction

Randomization in algorithms is quite common these days. Thus we don't need a long introduction.

A *randomized algorithm* is one that has access to a random number or random bit generator and can use those numbers/bits to make decisions. Historically, people have used random sampling in surveys and data collection. Most people credit the Metropolis-Hastings algorithm for numerical integration and sampling as the first non-trivial use of randomness in algorithms. It is an early example of a situation where randomness was not only powerful but also, in a sense, the only way. In modern computing, randomized algorithms are ubiquitous.

The goal of this course is to expose you to randomized algorithms from a theoretical computer science perspective. We will cover:

- Some well-known algorithms
- Tools to help design and analyze randomized algorithms
- Basic notions of complexity related to randomization in computing
- Applications to a few areas

Algorithms can be quite complicated to design and analyze, and the addition of randomization makes it even more so. For this reason, one has to learn certain "templates." Thus, there will be quite a bit of emphasis on *tools*.

1.1.1 Today's Tool: Linearity of Expectation

Suppose $X = \sum_{i=1}^{n} a_i X_i$, where $X_1, X_2, \ldots, X_n$ are random variables over some probability space $(\Omega, \Pr)$ with finite expectation. Then, by the linearity of expectation:

$$\mathbb{E}[X] = \sum_{i=1}^{n} a_i \, \mathbb{E}[X_i].$$

A key advantage is that the random variables X_i can be dependent.

1.2 Randomized Quicksort

Quicksort was developed by Hoare in 1959. It works well in practice since it is in-place, does not require additional memory, and has good cache behavior if implemented properly. One can show that the deterministic version runs in $O(n^2)$ time in the worst case. A bad example that leads to $\Omega(n^2)$ comparisons is a sorted array. One can try various pivot strategies.

We will analyze Randomized Quicksort. Assume the input array A has distinct elements.

1: **procedure** QUICKSORT($A[1 \dots n]$)
2: **if** $n \leq 1$ **then**
3: **return** A
4: pivot $\leftarrow A[i]$, where i is chosen uniformly at random from $\{1, 2, \dots, n\}$
5: Use pivot to split $A[1 \dots n]$ into $A[1 \dots \ell - 1]$, $A[\ell]$, $A[\ell + 1 \dots n]$ where:

(i) $A[1 \dots \ell - 1]$ are elements less than the pivot

(ii) $A[\ell + 1 \dots n]$ are elements greater than the pivot

(iii) ℓ is the rank of $A[i]$

6: This step takes $O(n)$ time with $O(1)$ additional memory.
7: Recursively sort $A[1 \dots \ell - 1]$ and $A[\ell + 1 \dots n]$.

Theorem 1.1. *Randomized Quicksort runs in expected time $O(n \log n)$ on an array of n elements.*

In fact, we will later show that it runs in $O(n \log n)$ time "with high probability."

Intuition: A random pivot has probability $\frac{1}{2}$ of having rank between $\frac{n}{4}$ and $\frac{3n}{4}$, i.e., it is an "approximate median" which leads to good divide and conquer. We know that there are many good approximate medians, but finding them deterministically is hard in many cases, even though a random sampling algorithm can easily get one. This is one design principle of randomized algorithms: *sample to find "good" witnesses.*

1.2.1 Analysis of Expected Running Time

Recursion-Based Analysis

Let $T(n)$ be the expected number of comparisons that Randomized Quicksort makes on an array of n elements. The recurrence relation is:

$$T(n) = n - 1 + \frac{1}{n} \sum_{i=1}^{n} \big(T(i-1) + T(n-i)\big)$$

with the base case $T(1) = 0$.

Exercise 1.2. Show that $T(n) = O(n \log n)$.

This recurrence is easy to write down, but the analysis is not very intuitive or insightful.

A Slick Analysis Using Indicator Variables

Let the input array be $A = [a_1, a_2, \dots, a_n]$. Let the sorted version of the array be $\text{Sort}(A) = [a_{i_1}, a_{i_2}, \dots, a_{i_n}]$, which we rewrite as $[a'_1, a'_2, \dots, a'_n]$.

- Let X be the total number of comparisons.

- Let E_{ij} be the event that the algorithm compares a'_i and a'_j.
- Let X_{ij} be the indicator random variable for E_{ij} (i.e., $X_{ij} = 1$ if E_{ij} happens and 0 otherwise).

The total number of comparisons is $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$. By linearity of expectation:

$$\mathbb{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[E_{ij}].$$

Lemma 1.3. $\Pr[E_{ij}] = \dfrac{2}{j-i+1}$.

Informal Proof. The elements a'_i and a'_j are compared if and only if one of them is chosen as a pivot from the set $\{a'_i, a'_{i+1}, \ldots, a'_j\}$ before any other element in that set is chosen as a pivot. Since any element in this set is equally likely to be the first one chosen as a pivot, the probability that this is either a'_i or a'_j is $\frac{2}{j-i+1}$.

The above is an informal proof. It needs more care to make it rigorous, using induction and conditional probability. □

Using this lemma, the expected number of comparisons is:

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \quad (\text{letting } k = j-i) \\
&\leq \sum_{i=1}^{n-1} 2H_n \\
&< 2nH_n \approx 2n(\ln n + \ln\ln n + O(1)) = O(n \log n).
\end{aligned}$$

Comments: This analysis is (i) very precise, and (ii) intuitive and insightful.

1.2.2 Concentration Bounds: Markov's Inequality

We understand that the expected running time of Randomized Quicksort is $O(n \log n)$. Does that mean it is a good algorithm? Is it sufficient to know the expectation? Ideally, we would like to know the full distribution of the running time. However, this is complicated. So we will typically use *bounds*.

The simplest probabilistic inequality is called Markov's inequality.

Theorem 1.4 (Markov's Inequality)**.** *Suppose X is a non-negative real-valued random variable with finite $\mathbb{E}[X]$. Then for any $t > 0$:*

$$\Pr[X \geq t \cdot \mathbb{E}[X]] \leq \frac{1}{t}.$$

For Randomized Quicksort, since $\mathbb{E}[X] \approx 2nH_n$, we can say:

$$\Pr[\#\text{comparisons} \geq 10nH_n] \leq \frac{1}{5}.$$

Here $t = 5$, since $10nH_n = 5 \cdot (2nH_n) \approx 5 \cdot \mathbb{E}[X]$.

Proof of Markov's Inequality. We have

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{\omega\in\Omega} \Pr[\omega]\, X(\omega) \\
&= \sum_{\substack{\omega\in\Omega \\ X(\omega)<t\,\mathbb{E}[X]}} \Pr[\omega]\, X(\omega) \;+\; \sum_{\substack{\omega\in\Omega \\ X(\omega)\geq t\,\mathbb{E}[X]}} \Pr[\omega]\, X(\omega) \\
&\geq \sum_{\substack{\omega\in\Omega \\ X(\omega)\geq t\,\mathbb{E}[X]}} \Pr[\omega]\, X(\omega) \qquad \text{(since } X \text{ is non-negative)} \\
&\geq \sum_{\substack{\omega\in\Omega \\ X(\omega)\geq t\,\mathbb{E}[X]}} \Pr[\omega]\cdot t\,\mathbb{E}[X] \\
&= \Pr[X \geq t\,\mathbb{E}[X]]\cdot t\,\mathbb{E}[X].
\end{aligned}$$

Rearranging gives $\Pr[X \geq t\,\mathbb{E}[X]] \leq \frac{1}{t}$. □

1.3 Max-Cut

Given a graph $G = (V, E)$, the Max-Cut problem asks for a partition of the vertex set V into $(S, V \setminus S)$ to maximize the number of edges crossing the cut, i.e., $|\delta(S)|$. This problem is NP-Hard.

1.3.1 A Simple Randomized Algorithm

1. $S \leftarrow \emptyset$.
2. Add each $u \in V$ to S independently with probability $\frac{1}{2}$.
3. Output S.

Analysis

Let X be the total number of edges that cross S, i.e., $|\delta(S)|$. For each edge $e \in E$, let X_e be the indicator random variable for e crossing S. Then $X = \sum_{e\in E} X_e$.

By linearity of expectation, $\mathbb{E}[X] = \sum_{e\in E} \mathbb{E}[X_e] = \sum_{e\in E} \Pr[X_e = 1]$.

For an edge $e = (u, v)$:

$$\Pr[X_e = 1] = \Pr[u \in S \wedge v \notin S] + \Pr[u \notin S \wedge v \in S] = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}.$$

Thus $\mathbb{E}[X] = \frac{m}{2}$, where $m = |E|$. Since the optimal cut size $\mathrm{OPT} \leq m$, we get $\mathbb{E}[X] \geq \frac{\mathrm{OPT}}{2}$, a $\frac{1}{2}$-approximation in expectation.

The Probabilistic Method

Remark 1.5. We have actually shown a graph-theoretic fact: in any graph, there exists a cut of size at least $\frac{|E|}{2}$. This is useful in itself. This bound is not tight. The following is known:

Theorem 1.6 (Edwards)**.** *In any connected graph, there exists a cut of size at least* $\frac{m}{2} + \frac{n-1}{4}$.

This way of using probability to show the existence of a certain type of combinatorial object is called the **probabilistic method**.

1.3.2 Reverse Markov Inequality

Now, back to the algorithm. Is there an analogue of Markov's inequality for showing a lower bound on the value of a random variable? We would like a bound of the form $\Pr[X < t\,\mathbb{E}[X]] \leq f(t)$ where $t \in (0,1)$. But one can see that this is not feasible if X does not have any upper bound. However, we can prove the following.

Theorem 1.7 (Reverse Markov Inequality)**.** *Let X be a non-negative real-valued random variable such that $X \leq B$ almost surely, i.e., $\Pr[0 \leq X \leq B] = 1$. Then for any $d < \mathbb{E}[X]$:*

$$\Pr[X \geq d] \geq \frac{\mathbb{E}[X] - d}{B - d}.$$

Proof. Apply Markov's inequality to the non-negative random variable $Y = B - X$. □

For Max-Cut, the number of crossing edges $X \leq m$. We have $\mathbb{E}[X] = \frac{m}{2}$. The probability of getting a cut of size at least $(1-\delta)\frac{m}{2}$:

$$\Pr\Big[X \geq (1-\delta)\frac{m}{2}\Big] \geq \frac{\frac{m}{2} - (1-\delta)\frac{m}{2}}{m - (1-\delta)\frac{m}{2}} = \frac{\delta\frac{m}{2}}{\frac{m}{2}(1+\delta)} = \frac{\delta}{1+\delta}.$$

By repeating the experiment many times and taking the maximum cut found among the solutions, we can get arbitrarily close to $\frac{m}{2}$.

Exercise 1.8. Given δ and ε in $(0,1)$, how many times should you repeat the algorithm so that

$$\Pr\Big[\text{output} \geq (1-\delta)\frac{m}{2}\Big] \geq 1 - \varepsilon\ ?$$

1.4 Approximating Max-Cut via Vector Programming

The randomized algorithm guarantees a cut of value $\geq \frac{m}{2}$ in expectation, but it may output only $\frac{m}{2}$ edges even when the graph is bipartite and OPT $= m$. For a long time, $\frac{1}{2}$ was the best approximation ratio, until Goemans and Williamson came up with a deceptively simple algorithm. The goal of this part of the lecture is to highlight the use of linearity of expectation and the use of *continuous* random variables. This is a bit advanced material that is meant to be interesting but may not be accessible to all students right away.

1.4.1 Problem Formulation and Relaxation

Let us write Max-Cut as an optimization problem using a quadratic program. Let $G = (V, E)$. We assume for notational ease that $V = \{1, 2, \ldots, n\}$ and use ij to refer to an edge connecting i to j. For each vertex $i \in V$, let $x_i \in \{-1, 1\}$. An edge ij is in the cut if $x_i \neq x_j$, i.e., $x_i x_j = -1$. The size of the cut is $\sum_{ij \in E} \frac{1}{2}(1 - x_i x_j)$. So the problem is:

$$\max \sum_{ij \in E} \frac{1}{2}(1 - x_i x_j) \quad \text{s.t.} \quad x_i \in \{-1, 1\} \ \ \forall i \in V.$$

Note that we are using a quadratic function in the objective. This is the same as requiring $x_i^2 = 1$ and $x_i \in \mathbb{R}$. It is clear that solving the above is going to solve Max-Cut exactly, and hence it is NP-Hard. One can also see this difficulty by noting that $1 - x_i x_j$ is a convex function; maximizing convex objectives is hard.

Let us take a geometric view of the above. We want to assign a unit vector x_i in one dimension to each $i \in V$ to maximize $\sum_{ij \in E} \frac{1}{2}(1 - x_i x_j)$.

Is there some way to *relax* the above to find an efficiently solvable program? Turns out, yes! Suppose instead of requiring x_i to be a one-dimensional scalar, we allow it to be an n-dimensional unit vector:

$$\text{(SDP Relaxation)} \quad \max \sum_{ij \in E} \frac{1}{2}(1 - \bar{v}_i \cdot \bar{v}_j) \quad \text{s.t.} \quad \|\bar{v}_i\| = 1,\ \bar{v}_i \in \mathbb{R}^n \ \ \forall i \in V.$$

The above, by magic, turns out to be a convex optimization problem! It is a semidefinite program (SDP) and can be solved to near-optimality efficiently.

Let OPT_{SDP} be the value of this relaxation. Clearly, $\text{OPT}_{\text{SDP}} \geq \text{OPT}$.

Example 1.9. Consider the triangle graph K_3 with vertices $\{1, 2, 3\}$. Here OPT $= 2$ (any cut separating one vertex from the other two cuts exactly 2 edges). For the SDP, we can place the three vectors $\bar{v}_1, \bar{v}_2, \bar{v}_3$ at angles of 120 from each other. Then:

$$\text{OPT}_{\text{SDP}} \geq \frac{3}{2}\big(1 - \cos 120\big) = \frac{3}{2} \cdot \frac{3}{2} = \frac{9}{4} = 2.25.$$

Is this a good relaxation? Can we round the vectors into a good solution?

1.4.2 Goemans-Williamson Rounding

How do we get a cut from the vector solution $\bar{v}_1, \ldots, \bar{v}_n$?

1. Solve the SDP to get vectors $\bar{v}_1, \ldots, \bar{v}_n$.
2. Pick a random hyperplane through the origin. Equivalently, let $\bar{r}$ be a random unit vector in $\mathbb{R}^n$.
3. $S = \{i \mid \bar{r} \cdot \bar{v}_i > 0\}$.
4. Output S.

Analysis

Let X_{ij} be the indicator for edge ij being cut. Let $X = \sum_{ij \in E} X_{ij}$.

$$\mathbb{E}[X] = \sum_{ij \in E} \Pr[X_{ij} = 1].$$

Let θ_{ij} be the angle between $\bar{v}_i$ and $\bar{v}_j$. An edge ij is cut if $\bar{v}_i$ and $\bar{v}_j$ fall on opposite sides of the random hyperplane. The probability of this is:

$$\Pr[X_{ij} = 1] = \frac{\theta_{ij}}{\pi}.$$

Thus $\mathbb{E}[X] = \sum_{ij \in E} \frac{\theta_{ij}}{\pi}$.

The value of the SDP solution is $\text{OPT}_{\text{SDP}} = \sum_{ij \in E} \frac{1}{2}(1 - \cos \theta_{ij})$.

Claim 1.10. *For all $\theta \in [0, \pi]$:*

$$\frac{\theta}{\pi} \geq 0.878 \cdot \frac{1}{2}(1 - \cos \theta).$$

This implies $\mathbb{E}[X] \geq 0.878 \cdot \text{OPT}_{\text{SDP}} \geq 0.878 \cdot \text{OPT}$.

Remark 1.11. It turns out that this is the optimal approximation ratio under a conjecture called the Unique Games Conjecture (UGC).

1.4.3 Generating Random Vectors

Question: How do we generate a random unit vector in $\mathbb{R}^d$ for d large?

Consider the standard 1-d Normal distribution $N(0,1)$ with mean 0 and standard deviation 1, having density function $\frac{1}{\sqrt{2\pi}}e^{-x^2/2}$.

1. Generate a vector $\bar{z} = (X_1, X_2, \ldots, X_d)$, where each X_i is an independent random variable drawn from $N(0,1)$.

2. The joint density is:
$$f_{\bar{z}}(x_1, \ldots, x_d) = \frac{1}{(\sqrt{2\pi})^d}\, e^{-(x_1^2+\cdots+x_d^2)/2}.$$

3. Notice the density is the same for all points $(x_1, \ldots, x_d)$ with the same length, i.e., it depends only on $\|\bar{z}\|$. Therefore the distribution is centrally symmetric, and the direction of $\bar{z}$ is uniformly distributed.

4. Output the unit vector $\bar{r} = \frac{\bar{z}}{\|\bar{z}\|}$.

Chapter 2

Randomized Min-Cut (Karger's Algorithm)

2.1 Randomized Min-Cut

Given an undirected graph $G = (V, E)$, we want to find a *global min-cut*. That is, a partition of V into S and $V \setminus S$, both non-empty, to minimize the number of edges crossing the partition. This is also called the *edge-connectivity* of a graph and is usually denoted by $\lambda(G)$.

2.1.1 Alternate View

In this view, we look for a minimum number of edges whose removal creates two non-trivial components.

Observation 2.1. In a connected undirected graph, a set of edges $E' \subseteq E$ is a min-cut if and only if $G - E'$ has exactly two connected components.

2.1.2 Examples

- **Cycle graph (C_n):** For a cycle graph, $\lambda(G) = 2$. There are exactly $\binom{n}{2}$ distinct min-cuts.
- **Complete graph (K_n):** For a complete graph, the min-cuts are singletons (partitioning off a single vertex). The size of such a cut is $n - 1$.

2.2 Algorithms for Min-Cut

2.2.1 Traditional Algorithm

Use $(n-1)$ s-t cut computations. Fix s and compute the s-v min-cut for all $v \in V \setminus \{s\}$, and take the minimum.

Very recent development: Using isolating cuts, one can compute the global min-cut using $O(\log^2 n)$ s-t cut computations.

Undirected graph cuts have more structure than directed graphs. Do we need to use flow? In the early 90's, Nagamochi and Ibaraki developed an MA-ordering based $O(mn)$ time algorithm.

2.2.2 Karger's Randomized Contraction Algorithm

In the mid-90's, Karger used *random contraction*. This algorithm and its analysis and implications have been very influential.

We will work with *multigraphs* since they come up naturally even when starting with a simple graph. Let $n = |V|$ and $m = |E|$. We will later discuss graphs having capacities.

Motivation

To motivate the algorithm we start with a seemingly simple observation.

Lemma 2.2. *The average degree is $\frac{2m}{n}$, and $\lambda(G) \leq \frac{2m}{n}$.*

Proof. Let $\delta = \min_{v \in V} \deg(v)$ be the minimum degree. We have

$$\delta \leq \frac{1}{n} \sum_{v \in V} \deg(v) = \frac{2m}{n}.$$

Moreover, $\lambda(G) \leq \delta$, since the cut formed by isolating the vertex with minimum degree has size δ. □

Now fix a specific min-cut C, so $|C| = \lambda$. Suppose we pick a random edge e from the graph. Then

$$\Pr[e \in C] = \frac{|C|}{|E|} = \frac{\lambda}{m} \leq \frac{2}{n}$$

by Lemma 2.2. Thus a random edge has a very low probability of being in C. In other words, there are many witnesses for not being in C. If $e = uv \notin C$, then it is safe to "contract" e.

Edge Contraction

Given a graph $G = (V, E)$ and an edge e, we denote G/e as the new graph obtained by merging the two endpoints of e and removing any self-loops. This may create parallel edges.

Observation 2.3. For any edge e, $\lambda(G/e) \geq \lambda(G)$, assuming G is connected. (Otherwise $\lambda = 0$ and is easy to check.)

The Algorithm

```
1: procedure RandomContraction(G = (V, E))
2:     if n = 2 then
3:         Output the unique cut between the two vertices.
4:     else
5:         Pick an edge e uniformly at random from E.
6:         return RandomContraction(G/e)
```

Theorem 2.4. *Let C be a fixed min-cut of G (viewed as a set of edges). Then*

$$\Pr[\textit{algorithm outputs } C] \geq \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}.$$

Proof. By induction on n. If $n = 2$, then C is the only cut and the algorithm outputs it with probability 1.

Otherwise, let A be the event that the random edge chosen is not in C. We have

$$\Pr[\bar{A}] = \frac{|C|}{m} \leq \frac{2}{n}, \qquad \text{so} \quad \Pr[A] \geq 1 - \frac{2}{n}.$$

The algorithm outputs C if A happens and, conditioned on A, the recursive call on G/e outputs C. Let B be the event that the algorithm on G/e outputs C, conditioned on A. Since C is still a min-cut in G/e, by induction:

$$\Pr[B] \geq \frac{2}{(n-1)(n-2)}.$$

Hence the algorithm outputs C with probability

$$\Pr[A] \cdot \Pr[B] \geq \left(1 - \frac{2}{n}\right) \cdot \frac{2}{(n-1)(n-2)} = \frac{n-2}{n} \cdot \frac{2}{(n-1)(n-2)} = \frac{2}{n(n-1)}.$$

A direct proof is simply to note that C is preserved if every contraction avoids C:

$$\Pr[\text{success}] \geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{3}\right) = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdots \frac{1}{3} = \frac{2}{n(n-1)}.$$

□

Thus the RC algorithm outputs any specific min-cut with probability $\Omega(1/n^2)$. Even though it is small, it is not too small. This type of randomized algorithm is called a **Monte Carlo algorithm** since it can fail with some probability but otherwise runs efficiently.

Improving Success Probability

The RC algorithm can be easily implemented to run in $O(n^2)$ time. It can be implemented to run in $O(m)$ time with some care.

Suppose we run $\alpha\binom{n}{2}$ independent copies of the algorithm and output the cheapest cut found. Then

$$\Pr[\text{output is not a min-cut}] \leq \left(1 - \frac{1}{\binom{n}{2}}\right)^{\alpha\binom{n}{2}} \leq e^{-\alpha}.$$

By choosing α sufficiently large, we can make this small. A typical choice is $\alpha = O(1)$ or $\alpha = O(\log n)$, giving failure probability $e^{-O(\log n)} = 1/\text{poly}(n)$. But then the running time increases to $T(n, m) \cdot \alpha\binom{n}{2}$. Thus, for constant probability of success we would end up with a running time of $O(n^2 m)$, which is not very competitive. However, we will see that the running time can be improved.

2.2.3 A Key Structural Result

The main impact of the algorithm was its simplicity and the following structural fact that led to many other results.

Theorem 2.5. *In any connected graph on n nodes, there are at most $\binom{n}{2}$ distinct min-cuts.*

Proof. Recall that each fixed min-cut C is output by the RC algorithm with probability $\geq 1/\binom{n}{2}$. Since these are disjoint events (the algorithm outputs exactly one cut), there cannot be more than $\binom{n}{2}$ distinct min-cuts. □

This bound is tight, as shown by the n-cycle C_n, which has exactly $\binom{n}{2}$ distinct min-cuts.

Definition 2.6. For $\alpha \geq 1$, an *α-approximate min-cut* is a cut C such that $|C| \leq \alpha \cdot \lambda(G)$.

How many α-approximate min-cuts are there?

Exercise 2.7. Suppose C is an α-approximate min-cut. Prove that the RC algorithm outputs C with probability $\geq 1/n^{2\alpha}$. Use this to argue that the number of α-approximate min-cuts is at most $n^{2\alpha}$.

Thus the number of approximate min-cuts grows only mildly with α.

Exercise 2.8. Let G be a graph and s, t be two nodes. Give an example that shows that the number of distinct s-t cuts is $2^{\Omega(n)}$.

2.3 Faster Min-Cut: The Karger-Stein Algorithm

The RC algorithm analysis shows that the probability of making an error is small at the beginning but gets large as n decreases. In particular, we have the following.

Observation 2.9. Fix a min-cut C. The probability that C survives the contraction process down to $\lceil n/\sqrt{2} \rceil + 2$ vertices is

$$\left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{\lceil n/\sqrt{2} \rceil + 2}\right) = \frac{(\lceil n/\sqrt{2} \rceil + 1)(\lceil n/\sqrt{2} \rceil)}{n(n-1)} \geq \frac{1}{2}.$$

So we can cut the graph to half the vertices while having good probability of success. Hence it makes sense to repeat this process.

2.3.1 The Karger-Stein (KS) Algorithm

1: **procedure** KargerStein(G) ▷ G has n vertices
2: **if** $n \leq n_0$ **then** ▷ fixed constant
3: Output min-cut in $O(1)$ time.
4: **else**
5: $t \leftarrow \lceil n/\sqrt{2} \rceil + 2$
6: Run RandomContraction on G until t vertices remain. Let G_1 be the resulting graph.
7: $C_1 \leftarrow$ KargerStein(G_1)
8: Independently, run RandomContraction on G until t vertices remain. Let G_2 be the result.
9: $C_2 \leftarrow$ KargerStein(G_2)
10: Output $\min(C_1, C_2)$.

2.3.2 Analysis

Lemma 2.10. *The Karger-Stein algorithm runs in time $O(n^2 \log n)$.*

Proof. The contraction phase to get to $n/\sqrt{2}+2$ vertices takes $O(n^2)$ time. (We can do it in $O(m)$ time, but it does not help here in the overall time, so we use the simpler bound.) Hence two calls take $O(n^2)$. The recurrence is

$$T(n) = O(n^2) + 2T\left(\frac{n}{\sqrt{2}} + 2\right).$$

The depth is $O(\log n)$, and this solves to $T(n) = O(n^2 \log n)$. □

Lemma 2.11 (Main Lemma)**.** *The Karger-Stein algorithm outputs a min-cut with probability* $\Omega(1/\log n)$.

Assuming Lemma 2.11, we can repeat KS $O(\log^2 n)$ times to get a min-cut with high probability, i.e., $1 - 1/\text{poly}(n)$. The total running time is $O(n^2 \log^3 n)$.

2.3.3 Connection to Galton-Watson Processes

Now we prove the main lemma. For this we make a connection to a birth-death process.

A *Galton-Watson process* is a branching stochastic process that models population growth. Imagine a species that generates children using asexual reproduction. The number of children generated by an individual is by some distribution $\mathcal{D}$ over $\{0, 1, 2, \dots\}$. Call this $\mathcal{D}$ and let μ be the expected number of children. Starting with one root, we generate k children where k is drawn according to $\mathcal{D}$. Each child then independently generates its own children. If $k = 0$, no children are generated.

Let X_i be the number of individuals at level/generation i, with $X_0 = 1$ (the root). The process is extinct at level i if $X_i = 0$. The extinction probability is $\lim_{n\to\infty} \Pr[X_n = 0]$.

Assume $\mathcal{D}$ is not trivial (i.e., $\mathcal{D}$ is not deterministically 1). Three cases:

- $\mu < 1$: Extinction probability is 1.
- $\mu = 1$: Extinction probability is 1 (this is the *critical case*).
- $\mu > 1$: Extinction probability is < 1 (the population may survive with positive probability).

In the critical case $\mu = 1$, one can show that $\Pr[X_d > 0] = \Theta(1/d)$.

Now we make a connection between the Galton-Watson process and the KS algorithm. Imagine a binary tree of infinite depth. Each edge is deleted independently with probability $\frac{1}{2}$. Let A_d be the event that there is a path from the root to some leaf at depth d.

In the KS algorithm, we think of each call to RC as succeeding with probability $\frac{1}{2}$. The algorithm outputs the correct answer if it does not fail along some path to depth $d = O(\log n)$. By the Galton-Watson analysis, the success probability is $\Omega(1/\log n)$.

2.3.4 Direct Proof of Success Probability

Proof of Lemma 2.11. Let p_i be the probability of reaching level i (i.e., success at recursion depth i). We have $p_0 = 1$.

We can write a recurrence for p_i. A call at depth $i+1$ succeeds if at least one of its two children succeeds. A child succeeds if its contraction phase preserves the cut (probability $\geq 1/2$) and its recursive call succeeds (probability p_i). The probability of one branch failing is $\frac{1}{2} + \frac{1}{2}(1 - p_i) = 1 - \frac{p_i}{2}$. So:

$$p_{i+1} = 1 - \left(1 - \frac{p_i}{2}\right)^2 = p_i - \frac{p_i^2}{4}.$$

We check: $p_0 = 1$, $p_1 = 1 - 1/4 = 3/4$, $p_2 = 3/4 - 9/64 = 39/64$.

Claim 2.12. $p_i \geq \dfrac{1}{i+1}$ *for all* $i \geq 0$.

Proof by induction. **Base case** $(i = 1)$**:** $p_1 = 3/4 \geq 1/2$. ✓

Inductive step: Assume $p_i \geq \frac{1}{i+1}$. The function $f(x) = x - x^2/4$ is increasing for $x \in [0,1]$. So if $p_i \geq \frac{1}{i+1}$, then

$$p_{i+1} = p_i - \frac{p_i^2}{4} \geq \frac{1}{i+1} - \frac{1}{4(i+1)^2} = \frac{4i+3}{4(i+1)^2}.$$

We need to show $\frac{4i+3}{4(i+1)^2} \geq \frac{1}{i+2}$. This is equivalent to

$$(4i+3)(i+2) \geq 4(i+1)^2, \qquad \text{i.e.,} \quad 4i^2 + 11i + 6 \geq 4i^2 + 8i + 4,$$

which simplifies to $3i + 2 \geq 0$. This is always true. □

The depth of recursion is $d \approx \log_{\sqrt{2}} n = 2\log_2 n$. The success probability is $p_d \geq \frac{1}{d+1} = \Omega(1/\log n)$. □

Chapter 3

Polynomial Identity Testing and Application to Matchings

3.1 Polynomial Identity Testing (PIT)

We begin with a brief review of fields, which provide the algebraic setting for polynomial identity testing.

Definition 3.1 (Field)**.** A *field* $\mathbb{F}$ is a set equipped with two operations, addition ($+$) and multiplication ($\cdot$), satisfying:

- $(\mathbb{F}, +)$ is a commutative (abelian) group.
- $(\mathbb{F} \setminus \{0\}, \cdot)$ is a commutative group.
- The distributive law holds: $a(b + c) = ab + ac$.

Example 3.2. The reals $\mathbb{R}$, complex numbers $\mathbb{C}$, and rationals $\mathbb{Q}$ are all fields.

Definition 3.3 (Finite Fields)**.** A *finite field* is a field with a finite number of elements.

- $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ with addition and multiplication modulo a prime p is a finite field.
- Every finite field has size p^k for some prime p and integer $k \geq 1$.

3.1.1 The PIT Problem

Given a multivariate polynomial $P(x_1, x_2, \ldots, x_n)$ over a field $\mathbb{F}$, presented as a "black box," we want to determine whether P is identically zero, i.e., whether $P \equiv 0$. By "black box" we mean that we can query for values of $P(a_1, a_2, \ldots, a_n)$ for any given field elements $a_1, \ldots, a_n \in \mathbb{F}$. More formally, we are given an arithmetic circuit that evaluates P.

Example 3.4. Consider $P(x_1, x_2, x_3) = 2(x_1 - x_2)x_3 + 2x_3x_2 + x_1^2 + x_2^2 - (x_1 + x_3)^2$. Is $P \equiv 0$?

Note that checking whether two polynomials P_1 and P_2 are identical reduces to checking whether $P_1 - P_2 \equiv 0$.

Definition 3.5 (Multivariate Polynomial)**.** A *multivariate polynomial* is a sum of monomials with coefficients from $\mathbb{F}$. A *monomial* is a product of powers of the variables:

$$x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}, \qquad \text{where } i_1, i_2, \ldots, i_n \geq 0 \text{ are integers.}$$

The *degree* of a monomial is $i_1 + i_2 + \cdots + i_n$. The *degree* of a polynomial P, denoted $\deg(P)$, is the maximum degree among its monomials (with non-zero coefficients).

Example 3.6. For $P(x_1, x_2) = x_1^3 x_2^4 + x_1^{10} + x_1^9 x_2^3$, the monomial degrees are 7, 10, and 12 respectively, so $\deg(P) = 12$.

3.1.2 Algorithms for PIT

Univariate Case

We first consider the case of a single variable x.

Theorem 3.7. *A non-zero univariate polynomial $P(x)$ of degree d over a field $\mathbb{F}$ has at most d roots.*

Deterministic Algorithm. Pick any set of $d+1$ distinct field elements $a_1, a_2, \ldots, a_{d+1} \in \mathbb{F}$. Evaluate $P(a_1), \ldots, P(a_{d+1})$. If all are zero, then $P \equiv 0$; otherwise $P \not\equiv 0$.

Randomized Algorithm. Let $S \subseteq \mathbb{F}$ be a finite set.

1: Pick $a \in S$ uniformly at random.
2: **if** $P(a) = 0$ **then**
3: Output "$P \equiv 0$."
4: **else**
5: Output "$P \not\equiv 0$."

Analysis. If $P \equiv 0$, the algorithm is always correct (it always outputs "$P \equiv 0$"). If $P \not\equiv 0$, the algorithm errs only when it picks a root of P:

$$\Pr[\text{Algorithm outputs "}P \equiv 0\text{"} \mid P \not\equiv 0] \leq \frac{d}{|S|}.$$

By choosing $|S|$ large, or by repeating, we can make the error probability arbitrarily small.

Remark 3.8 (Precision Issues)**.** If $\mathbb{F}$ is the reals, we cannot truly pick a uniformly random real number, nor evaluate $P(a)$ exactly due to precision issues. A common technique is to work over a finite field. If the coefficients are integers with maximum absolute value B and the degree is d, we can choose a prime $q > \max(B, d)$ and perform all computations in $\mathbb{Z}_q$. Then $P \equiv 0$ over $\mathbb{Q}$ if and only if $P \equiv 0$ over $\mathbb{Z}_q$. The computation is polynomial in $\log q$ and the complexity of the circuit for P.

Multivariate Case

For multivariate polynomials, there is no easy deterministic algorithm known. A multivariate polynomial P may have many roots even if $P \not\equiv 0$, so we cannot simply evaluate at a few points as in the univariate case. But randomization works!

Lemma 3.9 (Schwartz–Zippel Lemma)**.** *Let $P(x_1, \ldots, x_n)$ be a non-zero multivariate polynomial of total degree d over a field $\mathbb{F}$. Let $S \subseteq \mathbb{F}$ be a finite set. If $a_1, \ldots, a_n$ are chosen uniformly and independently at random from S, then*

$$\Pr[P(a_1, \ldots, a_n) = 0] \leq \frac{d}{|S|}.$$

Proof. By induction on n.

Base case ($n = 1$): $P(x_1)$ is a univariate polynomial of degree at most d, so it has at most d roots. Thus $\Pr[P(a_1) = 0] \leq d/|S|$.

Inductive step: Assume the lemma holds for $n-1$ variables. Write P as a polynomial in x_n:

$$P(x_1, \ldots, x_n) = \sum_{j=0}^{k} Q_j(x_1, \ldots, x_{n-1}) \cdot x_n^j,$$

where k is the highest power of x_n appearing in some monomial of P. If $k = 0$, then P depends only on $x_1, \ldots, x_{n-1}$ and we apply the inductive hypothesis directly. So assume $k \geq 1$, and note that $Q_k \not\equiv 0$ and $\deg(Q_k) \leq d - k$.

Let A denote the event $P(a_1, \ldots, a_n) = 0$, and let B denote the event $Q_k(a_1, \ldots, a_{n-1}) = 0$. We bound $\Pr[A]$ using the complementary approach. We have

$$\Pr[P(a_1, \ldots, a_n) \neq 0] \geq \Pr[Q_k(a_1, \ldots, a_{n-1}) \neq 0] \cdot \Pr[P(a_1, \ldots, a_n) \neq 0 \mid Q_k(a_1, \ldots, a_{n-1}) \neq 0].$$

By the inductive hypothesis, $\Pr[B] \leq (d-k)/|S|$, so $\Pr[\neg B] \geq 1 - (d-k)/|S|$.

Conditioned on $\neg B$ (i.e., $Q_k(a_1, \ldots, a_{n-1}) \neq 0$), fixing $a_1, \ldots, a_{n-1}$ makes $P(a_1, \ldots, a_{n-1}, x_n)$ a non-zero univariate polynomial in x_n of degree at most k. By Theorem 3.7, $\Pr[P = 0 \mid \neg B] \leq k/|S|$, so $\Pr[P \neq 0 \mid \neg B] \geq 1 - k/|S|$.

Combining:

$$\Pr[P \neq 0] \geq \left(1 - \frac{d-k}{|S|}\right)\left(1 - \frac{k}{|S|}\right) \geq 1 - \frac{d-k}{|S|} - \frac{k}{|S|} = 1 - \frac{d}{|S|}.$$

Therefore $\Pr[P(a_1, \ldots, a_n) = 0] \leq d/|S|$. □

Remark 3.10 (Major Open Question)**.** Is there a deterministic polynomial-time algorithm for PIT? This is a central question in derandomization. It is known that a deterministic polynomial-time algorithm for PIT would imply fundamental arithmetic circuit lower bounds, and vice versa.

3.2 Application to Matchings

Definition 3.11 (Matching)**.** Given a graph $G = (V, E)$, a *matching* is a subset $M \subseteq E$ such that no two edges in M share a vertex (i.e., if $e_1, e_2 \in M$ then $e_1 \cap e_2 = \emptyset$, viewing edges as vertex subsets). A matching is *perfect* if every vertex is incident to exactly one edge in M, which requires $|M| = |V|/2$.

Matchings have many applications. Important algorithmic problems include:

- Does G have a perfect matching?
- Does G have a matching of size $\geq k$?
- Find a maximum cardinality matching.
- Find a maximum weight matching.
- Find a minimum cost perfect matching.

In bipartite graphs, matching problems can be reduced to network flow and min-cost flow. General graph matchings are more complex and require the idea of blossoms (Edmonds' algorithm). In this lecture we will see a completely different algebraic approach to matchings via PIT and randomized algorithms.

3.2.1 Bipartite Perfect Matching

Let $G = (U \cup V, E)$ be a bipartite graph with $|U| = |V| = n$, where $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. We want to determine whether G has a perfect matching.

Definition 3.12 (Edmonds Matrix)**.** Define the $n \times n$ matrix A by

$$A_{ij} = \begin{cases} x_{ij} & \text{if } (u_i, v_j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where each x_{ij} is a distinct formal variable.

The determinant of A is the multivariate polynomial

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)},$$

where S_n is the set of all $n!$ permutations of $\{1, 2, \ldots, n\}$ (equivalently, the set of all bijections from $\{1, \ldots, n\}$ to $\{1, \ldots, n\}$).

Definition 3.13 (Sign of a Permutation)**.** A pair (i, j) with $i < j$ is an *inversion* of σ if $\sigma(i) > \sigma(j)$. The *sign* of σ is

$$\operatorname{sgn}(\sigma) = \begin{cases} +1 & \text{if the number of inversions in } \sigma \text{ is even,} \\ -1 & \text{if the number of inversions in } \sigma \text{ is odd.} \end{cases}$$

We will see another characterization of $\operatorname{sgn}(\sigma)$ via cycle decompositions later.

Example 3.14. Consider a bipartite graph on $n = 3$ vertices per side with edges (u_1, v_1), (u_1, v_2), (u_2, v_2), (u_3, v_2), (u_3, v_3). The Edmonds matrix is

$$A = \begin{pmatrix} x_{1,1} & x_{1,2} & 0 \\ 0 & x_{2,2} & 0 \\ 0 & x_{3,2} & x_{3,3} \end{pmatrix}.$$

Here $\det(A) = x_{1,1}x_{2,2}x_{3,3}$, which is non-zero, confirming that the graph has a perfect matching (namely $\{(u_1, v_1), (u_2, v_2), (u_3, v_3)\}$).

Now consider a graph with additional edges (u_2, v_3):

$$A = \begin{pmatrix} x_{1,1} & x_{1,2} & 0 \\ 0 & x_{2,2} & x_{2,3} \\ 0 & x_{3,2} & x_{3,3} \end{pmatrix}.$$

Then $\det(A) = x_{1,1}x_{2,2}x_{3,3} - x_{1,1}x_{2,3}x_{3,2}$. This is a non-zero polynomial, reflecting two perfect matchings.

Finally, if the only edges are (u_1, v_1), (u_1, v_2), (u_2, v_3), (u_3, v_3):

$$A = \begin{pmatrix} x_{1,1} & x_{1,2} & 0 \\ 0 & 0 & x_{2,3} \\ 0 & 0 & x_{3,3} \end{pmatrix}.$$

Here $\det(A) = 0$ (identically), and indeed the graph has no perfect matching.

Note that $\det(A)$ is a multivariate polynomial of degree at most n in the variables $\{x_{ij} : (u_i, v_j) \in E\}$.

Lemma 3.15. *The bipartite graph G has a perfect matching if and only if $\det(A) \not\equiv 0$.*

Proof. Each term in $\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}$ corresponds to a permutation σ. The product $\prod_{i=1}^{n} A_{i,\sigma(i)}$ is non-zero if and only if $(u_i, v_{\sigma(i)}) \in E$ for all $i \in [n]$, in which case the edges $\{(u_i, v_{\sigma(i)}) : i \in [n]\}$ form a perfect matching.

($\Rightarrow$) If $\det(A) \not\equiv 0$, then some monomial $\prod_{i=1}^{n} x_{i,\sigma(i)}$ is non-zero for some permutation σ. This means all edges $\{(u_i, v_{\sigma(i)})\}$ are present in G, giving a perfect matching.

($\Leftarrow$) If G has a perfect matching $M = \{(u_i, v_{\sigma(i)}) : i \in [n]\}$ for some permutation σ, then the term $\operatorname{sgn}(\sigma) \prod_{i=1}^{n} x_{i,\sigma(i)}$ appears in $\det(A)$. Since each permutation produces a distinct monomial (each x_{ij} is a distinct variable), this monomial cannot be cancelled by any other term. Thus $\det(A) \not\equiv 0$. □

This reduces the bipartite perfect matching problem to PIT!

Randomized Algorithm for Bipartite Perfect Matching

1: Let q be a prime with $q > 2n$.
2: For each variable x_{ij} (where $(u_i, v_j) \in E$), pick $a_{ij} \in \{1, \ldots, q-1\}$ uniformly at random.
3: Compute $\det(A_a)$, where A_a is the matrix obtained by substituting a_{ij} for each x_{ij}.
4: **if** $\det(A_a) \neq 0$ **then**
5: Output "G has a perfect matching."
6: **else**
7: Output "G has no perfect matching."

Analysis. By the Schwartz–Zippel Lemma (Lemma 3.9), if G has a perfect matching (so $\det(A) \not\equiv 0$ and $\deg(\det(A)) \leq n$), then

$$\Pr[\text{Algorithm is correct}] \geq 1 - \frac{n}{q} > 1 - \frac{n}{2n} = \frac{1}{2}.$$

If G has no perfect matching, the algorithm is always correct.

Running Time and Advantages

The determinant can be computed via matrix multiplication in $O(n^\omega)$ time, where $\omega < 3$ is the matrix multiplication exponent (currently $\omega \leq 2.372$).

Two related questions:

1. Can we solve maximum cardinality matching (given G and k, is there a matching of size $\geq k$)? Yes, this can be reduced to the decision version of perfect matching.

2. *Search problem:* Can we actually find the matching, not just decide its existence?

A key advantage of the algebraic approach is that determinant computation is easy to *parallelize*. The only known truly fast parallel algorithm for matching is via this algebraic approach, and it requires randomization.

Remark 3.16 (Open Question)**.** Is there a fast *deterministic* parallel algorithm for matching in graphs? This remains a major open problem.

3.2.2 General Graph Perfect Matching

Let $G = (V, E)$ be a general (not necessarily bipartite) graph with $V = [n] = \{1, 2, \ldots, n\}$. Assume n is even (otherwise there is no perfect matching).

Definition 3.17 (Tutte Matrix)**.** The $n \times n$ *Tutte matrix* T of G is defined by

$$T_{ij} = \begin{cases} x_{ij} & \text{if } \{i,j\} \in E \text{ and } i < j, \\ -x_{ji} & \text{if } \{i,j\} \in E \text{ and } i > j, \\ 0 & \text{otherwise.} \end{cases}$$

Note that T is *skew-symmetric*: $T_{ij} = -T_{ji}$ for all i, j, and $T_{ii} = 0$.

Example 3.18. Consider the graph G on vertices $\{1, 2, 3, 4\}$ with edges $\{1, 2\}$, $\{2, 3\}$, $\{2, 4\}$. The Tutte matrix is

$$T = \begin{pmatrix} 0 & x_{1,2} & 0 & 0 \\ -x_{1,2} & 0 & x_{2,3} & x_{2,4} \\ 0 & -x_{2,3} & 0 & 0 \\ 0 & -x_{2,4} & 0 & 0 \end{pmatrix}.$$

Theorem 3.19 (Tutte, 1947)**.** *A graph G has a perfect matching if and only if* $\det(T) \not\equiv 0$.

The proof requires understanding permutations via their cycle decompositions.

Permutations and Cycle Decompositions

Given a permutation $\sigma \in S_n$, define a directed graph H_σ on vertex set $V = \{1, 2, \ldots, n\}$ by adding a directed arc $(i, \sigma(i))$ for each $i \in [n]$ (self-loops are allowed when $\sigma(i) = i$). The graph H_σ decomposes into a collection of disjoint directed cycles; this is called a *cycle cover*. Conversely, any collection of vertex-disjoint directed cycles covering all of $[n]$ corresponds to a unique permutation.

A vertex i with $\sigma(i) = i$ is a *fixed point* of σ, corresponding to a cycle of length 1.

Example 3.20. Let $\sigma = (1\ 2)(3\ 4)$, i.e., $\sigma(1) = 2$, $\sigma(2) = 1$, $\sigma(3) = 4$, $\sigma(4) = 3$. Then H_σ consists of two 2-cycles: $1 \to 2 \to 1$ and $3 \to 4 \to 3$.

Let $\tau(1) = 3$, $\tau(2) = 2$, $\tau(3) = 4$, $\tau(4) = 1$. Then H_τ has a 3-cycle $1 \to 3 \to 4 \to 1$ and a fixed point $2 \to 2$.

Proposition 3.21 (Cycle Characterization of Sign)**.**

$$\operatorname{sgn}(\sigma) = (-1)^{\text{number of even-length cycles in } H_\sigma}.$$

Each even-length cycle contributes -1 to the sign, and each odd-length cycle contributes $+1$.

Exercise 3.22. Prove that this characterization is equivalent to the one via inversions (Definition 3.13).

Proof of Tutte's Theorem

We now prove Theorem 3.19. Recall that

$$\det(T) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} T_{i,\sigma(i)}.$$

We organize the proof around several claims.

Claim 3.23. *If σ has a fixed point, then $\prod_{i=1}^{n} T_{i,\sigma(i)} = 0$.*

Proof. If $\sigma(i) = i$ for some i, then $T_{i,i} = 0$ (since T is skew-symmetric), so the entire product is zero. □

Therefore, only fixed-point-free permutations contribute to $\det(T)$. Furthermore, $\prod_{i=1}^{n} T_{i,\sigma(i)} \neq 0$ implies that $\{i, \sigma(i)\} \in E$ for all i, i.e., the cycle cover H_σ uses only edges of G.

Claim 3.24. *Suppose $\sigma \in S_n$ has no fixed points but has an odd-length cycle C in H_σ. Let $\tau \in S_n$ be the permutation obtained by reversing the cycle C (and keeping all other cycles the same). Then*

$$\operatorname{sgn}(\sigma) \prod_{i=1}^{n} T_{i,\sigma(i)} = -\operatorname{sgn}(\tau) \prod_{i=1}^{n} T_{i,\tau(i)}.$$

Proof. Reversing a single cycle does not change the cycle structure (it is still a cycle of the same length), so $\operatorname{sgn}(\sigma) = \operatorname{sgn}(\tau)$. However, since T is skew-symmetric, reversing a cycle of odd length ℓ introduces a factor of $(-1)^\ell = -1$ in the product (each arc (i, j) becomes (j, i), contributing a factor of -1, and there are ℓ arcs). Thus the two terms cancel. □

Claim 3.25. *Let $S' = \{\sigma \in S_n : H_\sigma$ has a fixed point or an odd-length cycle$\}$. Then*

$$\sum_{\sigma \in S'} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} T_{i,\sigma(i)} = 0.$$

Proof. Each term corresponding to a σ with a fixed point is zero by Claim 3.23. For permutations with no fixed points but an odd-length cycle, we pair σ with τ (obtained by reversing the lowest-numbered odd cycle). By Claim 3.24, these paired terms cancel. One can verify this is a well-defined matching on such permutations. □

What remains are permutations σ such that all cycles of H_σ have even length.

Claim 3.26. *If G has a perfect matching M, then there exists a permutation $\sigma \in S_n$ such that H_σ has only even-length cycles and $\prod_{i=1}^{n} T_{i,\sigma(i)} \neq 0$.*

Proof. If $\{i, j\} \in M$, set $\sigma(i) = j$ and $\sigma(j) = i$. This gives a permutation consisting of $n/2$ transpositions (cycles of length 2). All cycles are even, and the product $\prod_{i=1}^{n} T_{i,\sigma(i)} \neq 0$ since all edges of M are in G. □

Claim 3.27. *Suppose $\sigma \in S_n$ has only even-length cycles and $\prod_{i=1}^{n} T_{i,\sigma(i)} \neq 0$. Then G has a perfect matching.*

Proof. If $\prod_{i=1}^{n} T_{i,\sigma(i)} \neq 0$, then all edges $\{i, \sigma(i)\}$ are in G. The cycle cover H_σ consists of even-length cycles on the edges of G. Each even-length cycle can be decomposed into two perfect matchings on its vertices (by taking alternating edges). Combining these across all cycles gives a perfect matching of G. □

Claim 3.28. *The terms corresponding to permutations with only even-length cycles do not all cancel.*

Proof. Suppose σ has only even-length cycles and $\prod_{i=1}^{n} T_{i,\sigma(i)} \neq 0$. Let E_σ be the set of edges of G induced by σ. To cancel $\operatorname{sgn}(\sigma) \prod_{i=1}^{n} T_{i,\sigma(i)}$, one would need another permutation τ with the same monomial (up to sign) but opposite sign in $\det(T)$. The only way to get the same set of edges is by reversing one or more cycles of H_σ. But since all cycles are even, reversing an even-length cycle of length ℓ introduces a factor of $(-1)^\ell = +1$, so the sign does not change. Therefore, no cancellation occurs. □

Combining Claims 3.25–3.28: $\det(T) \not\equiv 0$ if and only if there exists a permutation with only even-length cycles and all edges present in G, which by Claims 3.26 and 3.27 is equivalent to G having a perfect matching. □

The same randomized algorithm (substitute random values for the variables, compute the determinant) works for general graphs via the Tutte matrix, just as it does for bipartite graphs via the Edmonds matrix.

3.2.3 Finding a Matching: The Isolation Lemma

The PIT-based algorithms above are *decision* algorithms: they determine whether a perfect matching exists, but do not find one. A standard reduction from search to decision (remove edges one at a time, checking if a perfect matching still exists) is inherently sequential.

To find a perfect matching quickly *in parallel*, Mulmuley, Vazirani, and Vazirani developed the *Isolation Lemma*, a very general and powerful tool.

Lemma 3.29 (Isolation Lemma (Mulmuley–Vazirani–Vazirani))**.** *Let $N = \{1, 2, \ldots, n\}$ be a finite ground set, and let $\mathcal{F} \subseteq 2^N$ be a non-empty family of subsets. Assign to each element $i \in N$ a weight w_i chosen independently and uniformly at random from $\{1, 2, \ldots, d\}$. For a set $S \subseteq N$, define its weight as $W(S) = \sum_{i \in S} w_i$. Then*

$$\Pr[\textit{there is a unique minimum-weight set in } \mathcal{F}] \geq 1 - \frac{n}{d}.$$

Note that this holds regardless of the structure of $\mathcal{F}$—we do not even need to know $\mathcal{F}$ explicitly. By choosing $d = 2n$, we get a probability of at least $1/2$.

By assigning random weights to the edges of the graph, the Isolation Lemma guarantees that with good probability, there is a *unique* minimum-weight perfect matching. This uniqueness can be exploited in a parallel algorithm: one can determine for each edge whether it belongs to the unique minimum-weight matching by checking a suitable determinant condition, and all these checks can be performed in parallel.

Chapter 4

Tail Inequalities: Markov, Chebyshev, and Chernoff Bounds

4.1 Markov's Inequality

We have already seen Markov's inequality in an earlier lecture.

Inequality 4.1 (Markov's Inequality)**.** *Suppose X is a non-negative random variable. Then for any $t > 0$,*

$$\Pr[X \geq t \cdot \mathbb{E}[X]] \leq \frac{1}{t}.$$

Equivalently, for any $\alpha > 0$,

$$\Pr[X \geq \alpha] \leq \frac{\mathbb{E}[X]}{\alpha}.$$

Exercise 4.2. For any $t \geq 1$, give an example of a random variable X where Markov's inequality is tight.

4.2 Variance

The expectation is the first moment and provides important information, but is often not sufficient to do more sophisticated analysis. Now we will discuss the second moment, or the *variance*.

Definition 4.3 (Variance)**.** The *variance* of a random variable X is

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2].$$

The *standard deviation* is $\sigma_X = \sqrt{\mathrm{Var}(X)}$.

An alternative and useful formula for variance is:

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Example 4.4 (Binary Random Variable)**.** Let $X = 1$ with probability p and $X = 0$ otherwise. Then

$$\mathbb{E}[X] = p, \qquad \mathbb{E}[X^2] = p, \qquad \mathrm{Var}(X) = p - p^2 = p(1-p).$$

Example 4.5. Let $X = 1$ with probability $1/2$ and $X = -1$ with probability $1/2$. Then

$$\mathbb{E}[X] = 0, \qquad \mathbb{E}[X^2] = 1, \qquad \mathrm{Var}(X) = 1.$$

Example 4.6 (Geometric Random Variable)**.** Let X be a geometric random variable with parameter p, representing the number of coin tosses needed to get the first head (where the probability of heads is p). Then

$$\mathbb{E}[X] = \frac{1}{p}, \qquad \mathrm{Var}(X) = \frac{1-p}{p^2}.$$

Example 4.7 (Poisson Random Variable)**.** Let X be a Poisson random variable with parameter λ, so $\Pr[X = i] = e^{-\lambda}\frac{\lambda^i}{i!}$. Then

$$\mathbb{E}[X] = \lambda, \qquad \mathrm{Var}(X) = \lambda.$$

Example 4.8 (Normal Distribution)**.** For a normal distribution with density $f(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu)^2/(2\sigma^2)}$,

$$\mathbb{E}[X] = \mu, \qquad \mathrm{Var}(X) = \sigma^2.$$

4.2.1 Sums of Independent Random Variables

In many applications we want to understand the behavior of a sum $X = X_1 + X_2 + \cdots + X_n$ where $X_1, \ldots, X_n$ are independent random variables.

Lemma 4.9. *If X_1 and X_2 are independent random variables, then* $\mathrm{Var}(X_1 + X_2) = \mathrm{Var}(X_1) + \mathrm{Var}(X_2)$.

Proof. We have

$$\mathbb{E}[(X_1 + X_2)^2] = \mathbb{E}[X_1^2] + \mathbb{E}[X_2^2] + 2\mathbb{E}[X_1X_2] = \mathbb{E}[X_1^2] + \mathbb{E}[X_2^2] + 2\mathbb{E}[X_1]\mathbb{E}[X_2],$$

where the last step uses independence. Also,

$$(\mathbb{E}[X_1 + X_2])^2 = (\mathbb{E}[X_1])^2 + (\mathbb{E}[X_2])^2 + 2\mathbb{E}[X_1]\mathbb{E}[X_2].$$

Subtracting gives $\mathrm{Var}(X_1+X_2) = (\mathbb{E}[X_1^2]-(\mathbb{E}[X_1])^2)+(\mathbb{E}[X_2^2]-(\mathbb{E}[X_2])^2) = \mathrm{Var}(X_1)+\mathrm{Var}(X_2)$. □

More generally, for a sum of n mutually independent random variables $X = \sum_{i=1}^n X_i$:

$$\mathbb{E}[X] = \sum_{i=1}^n \mathbb{E}[X_i] \quad \text{(by linearity of expectation)}, \qquad \mathrm{Var}(X) = \sum_{i=1}^n \mathrm{Var}(X_i) \quad \text{(by independence)}.$$

4.3 Chebyshev's Inequality

Inequality 4.10 (Chebyshev's Inequality)**.** *Suppose X is a random variable with variance σ_X^2. Then for any $t > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \geq t\sigma_X] \leq \frac{1}{t^2}.$$

Equivalently, for any $\gamma > 0$,

$$\Pr[|X - \mathbb{E}[X]| \geq \gamma] \leq \frac{\sigma_X^2}{\gamma^2}.$$

Proof. Let $Y = (X - \mathbb{E}[X])^2$. Then $\mathbb{E}[Y] = \sigma_X^2$ by definition, and Y is a non-negative random variable. By Markov's inequality,

$$\Pr[|X - \mathbb{E}[X]| \geq t\sigma_X] = \Pr[Y \geq t^2\sigma_X^2] \leq \frac{\mathbb{E}[Y]}{t^2\sigma_X^2} = \frac{\sigma_X^2}{t^2\sigma_X^2} = \frac{1}{t^2}.$$ □

4.4 Applications of Chebyshev's Inequality

4.4.1 Variance Reduction

Suppose we have a randomized procedure that outputs a random variable X to estimate a quantity of interest α, such that $\mathbb{E}[X] = \alpha$. This may not be adequate if $\mathrm{Var}(X)$ is large.

Suppose we run the algorithm k times independently, obtaining estimates $X_1, X_2, \ldots, X_k$. The final estimate is the average:

$$\hat{X} = \frac{1}{k}\sum_{i=1}^{k} X_i.$$

The expectation remains correct: $\mathbb{E}[\hat{X}] = \alpha$. The variance is reduced:

$$\mathrm{Var}(\hat{X}) = \mathrm{Var}\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right) = \frac{1}{k^2}\sum_{i=1}^{k}\mathrm{Var}(X_i) = \frac{\mathrm{Var}(X)}{k}.$$

So repetition and averaging reduces variance.

4.4.2 Balls and Bins

Consider throwing m balls independently and uniformly at random into n bins. This is a basic and important process that underlies many applications such as hashing.

Let Y_i be the load in bin i (the number of balls in bin i). Define indicator variables X_{ij} for ball j landing in bin i, so $\Pr[X_{ij} = 1] = 1/n$. Then $Y_i = \sum_{j=1}^{m} X_{ij}$, and the X_{ij} for $j = 1, \ldots, m$ are independent. We have

$$\mathbb{E}[Y_i] = \frac{m}{n}, \qquad \mathrm{Var}(Y_i) = \sum_{j=1}^{m}\mathrm{Var}(X_{ij}) = m \cdot \frac{1}{n}\left(1 - \frac{1}{n}\right) \approx \frac{m}{n}.$$

A key quantity of interest is the maximum load $Z = \max_{i=1}^{n} Y_i$, which measures how unevenly the balls are spread.

Let us focus on the case $m = n$. Then $\mathbb{E}[Y_i] = 1$ and $\mathrm{Var}(Y_i) = 1 - 1/n < 1$. The variables $Y_1, \ldots, Y_n$ are correlated, making direct analysis of Z difficult. A useful and recurring paradigm is to combine a *deviation inequality* with the *union bound*.

Lemma 4.11 (Union Bound)**.** *Let $A_1, \ldots, A_n$ be events. Then* $\Pr[A_1 \cup A_2 \cup \cdots \cup A_n] \le \sum_{i=1}^{n}\Pr[A_i]$.

This is a weak inequality, but like linearity of expectation, it requires no understanding of the dependencies among the events.

Applying Markov to Y_i. We have $\mathbb{E}[Y_i] = 1$, so $\Pr[Y_i \ge t] \le 1/t$. But to get anything useful via the union bound over n bins, we need $t \ge n$, which is useless.

Applying Chebyshev to Y_i. For $m = n$, Chebyshev gives

$$\Pr[|Y_i - \mathbb{E}[Y_i]| \ge t\sqrt{n}] \le \frac{\mathrm{Var}(Y_i)}{t^2 n} \le \frac{1}{t^2 n}.$$

This implies $\Pr[Y_i \ge t\sqrt{n} + 1] \le \frac{1}{t^2 n}$. By the union bound over all n bins:

$$\Pr[Z \ge t\sqrt{n} + 1] = \Pr\left[\bigcup_{i=1}^{n}\{Y_i \ge t\sqrt{n} + 1\}\right] \le n \cdot \frac{1}{t^2 n} = \frac{1}{t^2}.$$

Exercise 4.12. Show that $\mathbb{E}[Z] \le c\sqrt{n}$ for some fixed constant c.

Exercise 4.13. Generalize the above analysis to general m.

4.4.3 Random Walk on the Line

Start at the origin. At each step, walk one unit to the right or left with equal probability 1/2. Let $X_i \in \{-1, +1\}$ be the i-th step, and let $Y_n = \sum_{i=1}^{n} X_i$ be the position after n steps. Then $|Y_n|$ is the distance from the origin.

We have $\mathbb{E}[X_i] = 0$ and $\mathrm{Var}(X_i) = \mathbb{E}[X_i^2] = 1$. By independence,

$$\mathbb{E}[Y_n] = 0, \qquad \mathrm{Var}(Y_n) = n, \qquad \sigma_{Y_n} = \sqrt{n}.$$

By Chebyshev's inequality:

$$\Pr[|Y_n| \geq t\sqrt{n}] = \Pr[|Y_n - \mathbb{E}[Y_n]| \geq t\sqrt{n}] \leq \frac{1}{t^2}.$$

This implies $\mathbb{E}[|Y_n|] = O(\sqrt{n})$.

4.5 Chernoff–Hoeffding Bounds

Chernoff–Hoeffding bounds are very useful probabilistic inequalities that provide tight bounds in many settings. They are perhaps the most important tool you will learn from this course.

To motivate, they give much better results than Chebyshev's inequality for the previous examples:

- **Balls and bins ($m = n$):** $\mathbb{E}[Z] = O\big(\frac{\ln n}{\ln \ln n}\big)$, much better than $O(\sqrt{n})$ from Chebyshev.
- **Random walk:** $\Pr[|Y_n| \geq t\sqrt{n}] \leq 2e^{-t^2/2}$, exponential decay in t^2, much stronger than $1/t^2$ from Chebyshev.

4.5.1 Chernoff Bounds for Sums of Binary Variables

There are several variants. We start with the multiplicative Chernoff bound.

Theorem 4.14 (Multiplicative Chernoff Bound)**.** *Let $X = \sum_{i=1}^{n} X_i$, where $X_1, \ldots, X_n$ are independent binary random variables ($X_i \in \{0, 1\}$). Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i]$. Then:*

1. ***Upper tail:*** *For any $\delta > 0$,*

$$\Pr[X \geq (1+\delta)\mu] \leq \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}.$$

2. ***Lower tail:*** *For $\delta \in (0, 1)$,*

$$\Pr[X \leq (1-\delta)\mu] \leq \left[\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right]^{\mu}.$$

Example 4.15. Consider n independent fair coins: $X_i \in \{0, 1\}$ with probability 1/2 each. Let $X = \sum_{i=1}^{n} X_i$, so $\mathbb{E}[X] = n/2$. All 2^n bit strings are equally likely, and X counts the number of 1s. The exact distribution is

$$\Pr[X = k] = \binom{n}{k}\left(\frac{1}{2}\right)^n.$$

Using Stirling's approximation, $\Pr[X = n/2] \approx \sqrt{\pi}/\sqrt{n}$, confirming that the distribution is sharply peaked. Now suppose $p = \frac{\ln n}{n}$ for each coin. Then $\mathbb{E}[X] = \ln n$. Even though exact formulas exist, they are difficult to work with—the Chernoff bound provides a clean analytical formula depending only on $\mu = \mathbb{E}[X]$.

Proof of the Upper Tail

Proof. The key idea is to use the *moment generating function* and apply Markov's inequality. For any $t > 0$:

$$\Pr[X \geq (1+\delta)\mu] = \Pr[e^{tX} \geq e^{t(1+\delta)\mu}] \leq \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\delta)\mu}}.$$

The moment generating function is called so because $e^{tX} = 1 + tX + \frac{t^2X^2}{2!} + \frac{t^3X^3}{3!} + \cdots$, so its expectation encodes all moments of X.

Now we bound $\mathbb{E}[e^{tX}]$. By independence,

$$\mathbb{E}[e^{tX}] = \mathbb{E}\left[\prod_{i=1}^{n} e^{tX_i}\right] = \prod_{i=1}^{n} \mathbb{E}[e^{tX_i}].$$

Let $p_i = \mathbb{E}[X_i]$. Then

$$\mathbb{E}[e^{tX_i}] = p_i e^t + (1 - p_i) = 1 + p_i(e^t - 1) \leq e^{p_i(e^t-1)},$$

using the inequality $1 + x \leq e^x$. Therefore,

$$\prod_{i=1}^{n} \mathbb{E}[e^{tX_i}] \leq \prod_{i=1}^{n} e^{p_i(e^t-1)} = e^{(e^t-1)\sum_i p_i} = e^{(e^t-1)\mu}.$$

Plugging back:

$$\Pr[X \geq (1+\delta)\mu] \leq \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}} = \left[\frac{e^{e^t-1}}{e^{t(1+\delta)}}\right]^{\mu}.$$

We choose t to minimize $g(t) = e^t - 1 - t(1+\delta)$. Setting $g'(t) = e^t - (1+\delta) = 0$ gives $t = \ln(1+\delta)$. Substituting:

$$e^t - 1 - t(1+\delta) = \delta - (1+\delta)\ln(1+\delta),$$

so the bound becomes

$$\Pr[X \geq (1+\delta)\mu] \leq \left[e^{\delta-(1+\delta)\ln(1+\delta)}\right]^{\mu} = \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}. \qquad \square$$

Proof of the Lower Tail

Proof. The proof is similar. For any $t > 0$:

$$\Pr[X \leq (1-\delta)\mu] = \Pr[-X \geq -(1-\delta)\mu] = \Pr[e^{-tX} \geq e^{-t(1-\delta)\mu}] \leq \frac{\mathbb{E}[e^{-tX}]}{e^{-t(1-\delta)\mu}}.$$

By independence,

$$\mathbb{E}[e^{-tX}] = \prod_{i=1}^{n} \mathbb{E}[e^{-tX_i}] = \prod_{i=1}^{n} \left[p_i e^{-t} + (1 - p_i)\right] = \prod_{i=1}^{n} \left[1 + p_i(e^{-t} - 1)\right] \leq e^{(e^{-t}-1)\mu}.$$

Thus

$$\Pr[X \leq (1-\delta)\mu] \leq \frac{e^{(e^{-t}-1)\mu}}{e^{-t(1-\delta)\mu}}.$$

Minimizing $g(t) = e^{-t} - 1 + t(1-\delta)$ by setting $g'(t) = -e^{-t} + (1-\delta) = 0$ gives $e^{-t} = 1 - \delta$, so $t = \ln\frac{1}{1-\delta}$. Substituting yields the bound

$$\Pr[X \leq (1-\delta)\mu] \leq \left[\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right]^{\mu}. \qquad \square$$

Simplified Chernoff Bounds

The standard forms can be simplified into more convenient bounds.

Corollary 4.16 (Simplified Chernoff Bounds)**.** *Let $X = \sum_{i=1}^{n} X_i$ with $X_i \in \{0,1\}$ independent, and $\mu = \mathbb{E}[X]$.*

- ***Lower tail:*** *For $\delta \in (0,1)$,*
$$\Pr[X \le (1-\delta)\mu] \le e^{-\mu\delta^2/2}.$$
- ***Upper tail (small δ):*** *For $0 < \delta \le 1.81$,*
$$\Pr[X \ge (1+\delta)\mu] \le e^{-\mu\delta^2/3}.$$
- ***Upper tail (constant δ):*** *For $\delta \ge 1$,*
$$\Pr[X \ge (1+\delta)\mu] \le e^{-\delta\mu/3}.$$
- ***Upper tail (large δ):*** *For $\delta \ge 1$,*
$$\Pr[X \ge (1+\delta)\mu] \le e^{-(1+\delta)\ln(1+\delta)\mu/4}.$$

4.5.2 Hoeffding's Inequality (General Case)

This is an additive form of the bound for variables bounded in an interval.

Theorem 4.17 (Hoeffding's Inequality)**.** *Let $X_1, \dots, X_n$ be independent random variables with $X_i \in [a_i, b_i]$. Let $X = \sum_{i=1}^{n} X_i$. Then for any $a > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \ge a] \le 2\exp\left(-\frac{2a^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

A common special case is $X_i \in \{-1, 1\}$, where $b_i - a_i = 2$. Then

$$\Pr[X - \mathbb{E}[X] \ge a] \le e^{-a^2/(2n)}, \qquad \Pr[|X - \mathbb{E}[X]| \ge a] \le 2e^{-a^2/(2n)}.$$

Note that this bound depends on n and a, and cannot give any multiplicative guarantee relative to $\mathbb{E}[X]$, which can be 0.

4.6 Applications of Chernoff–Hoeffding Bounds

4.6.1 Balls and Bins (Revisited)

For $m = n$ balls into n bins, recall Y_i is the load on bin i with $\mathbb{E}[Y_i] = 1$. We bound the probability that $Y_i \ge 10\frac{\ln n}{\ln\ln n}$.

Here $\mu = 1$ and $(1+\delta)\mu = 10\frac{\ln n}{\ln\ln n}$, so δ is large. We use the large-deviation upper tail bound from Theorem 4.14:

$$\Pr[Y_i \ge (1+\delta)\mu] \le e^{-(1+\delta)\ln(1+\delta)\mu/4}.$$

We have $(1+\delta) = 10\frac{\ln n}{\ln\ln n}$, so

$$\ln(1+\delta) = \ln\left(\frac{10\ln n}{\ln\ln n}\right) = \ln 10 + \ln\ln n - \ln\ln\ln n \ge \frac{1}{2}\ln\ln n$$

for sufficiently large n. Therefore,

$$(1+\delta)\ln(1+\delta) \geq \frac{10\ln n}{\ln\ln n} \cdot \frac{\ln\ln n}{2} = 5\ln n,$$

giving

$$\Pr\left[Y_i \geq 10\frac{\ln n}{\ln\ln n}\right] \leq e^{-5\ln n} = \frac{1}{n^5}.$$

By the union bound over all n bins:

$$\Pr\left[Z \geq 10\frac{\ln n}{\ln\ln n}\right] \leq n \cdot \frac{1}{n^5} = \frac{1}{n^4}.$$

This shows that with high probability, the maximum load is $O\big(\frac{\ln n}{\ln\ln n}\big)$.

4.6.2 Randomized Quicksort (Revisited)

We saw that randomized quicksort runs in expected time $O(n\ln n)$ on an n-element array. We will show that the runtime is tightly concentrated around this mean. Let $Q(A)$ be the number of comparisons on an array A of size n. We will show:

$$\Pr[Q(A) > 32n\ln n] \leq \frac{1}{n^8}.$$

The constants are loose; much tighter bounds are known. The point is to illustrate the technique.

Setup. Focus on a fixed but arbitrary element $a \in A$. Let S_i be the subarray containing a at recursion level i, with $S_0 = A$ and $S_k = \{a\}$ where k is the last level for a (a random variable). The total number of comparisons is at most $k \cdot n$, so it suffices to prove that the recursion depth k is at most $32\ln n$ with high probability.

Call a "lucky" at level i if the pivot chosen at that level is an "approximate median" for S_i, meaning $|S_{i+1}| \leq \frac{3}{4}|S_i|$. The probability of a lucky split is at least $1/2$ (a pivot in the middle half of the sorted order of S_i achieves this).

Lemma 4.18. *Let n be sufficiently large and $h = 32\ln n$. Consider h independent unbiased coin tosses $X_1, \ldots, X_h$, and let A be the event that fewer than $4\ln n$ are heads. Then $\Pr[A] \leq 1/n^9$.*

Proof. Let $Y = X_1 + \cdots + X_h$. Then $\mathbb{E}[Y] = h/2 = 16\ln n$, and $4\ln n = (1 - 3/4) \cdot 16\ln n$. Using the simplified Chernoff lower tail bound with $\delta = 3/4$:

$$\Pr[Y < 4\ln n] = \Pr[Y \leq (1-\delta)\mathbb{E}[Y]] \leq e^{-\delta^2\mathbb{E}[Y]/2} = e^{-(9/16)(16\ln n)/2} = e^{-9\ln n} = \frac{1}{n^9}. \qquad \square$$

Now, element a has recursion depth $> h = 32\ln n$ only if it has fewer than $4\ln n$ lucky rounds in the first h levels. (If it has at least $4\ln n$ lucky rounds, the subarray size is reduced by a factor of $3/4$ at least $4\ln n$ times, giving size at most $n \cdot (3/4)^{4\ln n} < 1$ for large n.) By Lemma 4.18, $\Pr[k > 32\ln n] \leq 1/n^9$.

By the union bound over all n elements:

$$\Pr[\text{depth} > 32\ln n] \leq \frac{n}{n^9} = \frac{1}{n^8}.$$

Since the number of comparisons at each level is at most n, we conclude $\Pr[Q(A) > 32n\ln n] \leq 1/n^8$.

4.6.3 Random Walk (Revisited)

Using Hoeffding's inequality for $Y_n = \sum_{i=1}^{n} X_i$ where $X_i \in \{-1, 1\}$ and $\mathbb{E}[Y_n] = 0$:

$$\Pr[|Y_n| \geq t\sqrt{n}] = \Pr[|Y_n - \mathbb{E}[Y_n]| \geq t\sqrt{n}] \leq 2\exp\left(-\frac{t^2 n}{2n}\right) = 2e^{-t^2/2}.$$

This is an exponential decay in t^2, significantly stronger than the $1/t^2$ decay from Chebyshev's inequality.

Chapter 5

Applications of Chernoff-Hoeffding Bounds

5.1 An Application to Routing for Congestion Minimization

A classical problem from both theory and practice.

Given $G = (V, E)$ a directed graph with $m = |E|$ edges. Let $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$ be k source-sink pairs.

5.1.1 The Edge-Disjoint Paths (EDP) Problem

The *Edge-Disjoint Paths* (EDP) problem asks the following: can we find paths $P_1, P_2, \ldots, P_k$ such that

(i) P_i is an s_i–t_i path for each $i \in [k]$, and

(ii) $P_1, \ldots, P_k$ are edge-disjoint?

This is a fundamental but difficult problem. We will consider a *relaxation*: given G and the pairs, find paths $P_1, \ldots, P_k$ such that no edge is used in too many paths. The maximum number of paths using any single edge is called the *congestion*.

5.1.2 LP Relaxation

Let $\mathcal{P}_i$ be the set of all $s_i \to t_i$ paths—an exponentially large set. We write a path-based LP relaxation:

$$\begin{aligned} \min \quad & \lambda \\ \text{s.t.} \quad & \sum_{p \in \mathcal{P}_i} x_p = 1 && \forall\, i \in [k] \\ & \sum_{\substack{i=1 \\ p \in \mathcal{P}_i,\, p \ni e}}^{k} x_p \leq \lambda && \forall\, e \in E \\ & x_p \geq 0 && p \in \bigcup_i \mathcal{P}_i \end{aligned}$$

We will not discuss how to solve this LP, but it can be done via the Ellipsoid method or by writing a different edge-flow based formulation. In the edge-flow formulation, $x(e, i)$ denotes the

flow on edge e for pair i:

$$\begin{aligned}
\min \quad & \lambda \\
\text{s.t.} \quad & \sum_{e \in \delta^+(s_i)} x(e,i) - \sum_{e \in \delta^-(s_i)} x(e,i) = 1 && \forall\, i \in [k] \\
& \sum_{e \in \delta^+(v)} x(e,i) - \sum_{e \in \delta^-(v)} x(e,i) = 0 && \forall\, v \notin \{s_i, t_i\},\ \forall\, i \in [k] \\
& \sum_{i=1}^{k} x(e,i) \le \lambda && \forall\, e \in E \\
& x(e,i) \ge 0 && e \in E,\ i \in [k]
\end{aligned}$$

Let λ^* be the optimum value of the LP relaxation. Note that λ^* can be smaller than 1, so the true lower bound on congestion is $\max\{\lambda^*, 1\}$.

Can we convert the fractional solution to an integral solution with small congestion?

5.1.3 Randomized Rounding

The rounding algorithm is simple:

1. Solve the LP relaxation to obtain an optimal fractional solution $\bar{x}$.
2. For each pair (s_i, t_i) *independently*, pick a random path $P_i \in \mathcal{P}_i$ where path p is chosen with probability $\bar{x}_p$.

Raghavan and Thompson were the first to analyze this rounding scheme via Chernoff bounds, in 1987.

Theorem 5.1 (Raghavan–Thompson)**.** *The algorithm outputs a set of random paths $P_1, \ldots, P_k$ such that $\forall\, i \in [k]$, P_i is an s_i–t_i path. Moreover, with probability at least $1 - \frac{1}{\text{poly}(m)}$, the load on every edge is at most*

$$O\left(\frac{\log m}{\log \log m}\right) \max\{\lambda^*, 1\}.$$

Proof sketch. Consider any fixed edge e. Let Y_i be the indicator random variable for the event that P_i, the path chosen for pair (s_i, t_i), uses edge e. Let $Y = \sum_{i=1}^{k} Y_i$ be the total load on e.

We have

$$\Pr[Y_i = 1] = \sum_{\substack{p \in \mathcal{P}_i \\ p \ni e}} \bar{x}_p = x(e,i),$$

the total flow on e for commodity i.

Since the paths for different pairs are chosen *independently*, $Y_1, \ldots, Y_k$ are independent random variables. Therefore

$$\mathbb{E}[Y] = \sum_{i=1}^{k} \mathbb{E}[Y_i] = \sum_{i=1}^{k} x(e,i) \le \lambda^*.$$

By the Chernoff bound (using the form for $\mu \le 1$, since $\lambda^* \le 1$ is possible), for sufficiently large constants c and c',

$$\Pr\left[Y \ge c \cdot \frac{\log m}{\log \log m}\right] \le \frac{1}{m^{c'}}.$$

We apply the union bound over all m edges:

$$\Pr\left[\text{load on any edge} > c \cdot \frac{\log m}{\log\log m}\right] \le m \cdot \frac{1}{m^{c'}} = \frac{1}{m^{c'-1}}.$$

Choosing c' large enough (say $c' \ge 2$) gives the desired high-probability bound. □

Exercise 5.2. You can use Chernoff bounds to prove two refined bounds for the routing problem above.

(i) Show that there exist constants c, c' such that for any $\epsilon \in (0,1)$,

$$\Pr\left[Y \ge (1+\epsilon)\lambda^* + \frac{c\log m}{\epsilon^2}\right] \le \frac{1}{m^{c'}}.$$

Thus when $\lambda^* = \Omega(\log m)$, we get a $(1+\epsilon)$-approximation to the fractional congestion plus a lower-order additive term—a very good approximation.

(ii) Show that when $\lambda^* \ge c\log m$,

$$\Pr\left[Y > \lambda^* + \sqrt{c\log m \cdot \lambda^*}\right] \le \frac{1}{m^{c'}}.$$

Note that the preceding bound has *no* multiplicative factor on λ^*; the deviation is only $O(\sqrt{\lambda^* \log m})$.

5.2 Additive Chernoff Bound (Hoeffding's Inequality)

To motivate this bound, consider again the random walk on the line. We had $Y = \sum_{i=1}^n X_i$ where $X_i \in \{-1, 1\}$ and $\mathbb{E}[X_i] = 0$, and hence $\mathbb{E}[Y] = 0$. In this setting the mean is zero, so we cannot expect a *multiplicative* Chernoff bound (which gives tail bounds relative to the mean). We need an *additive* bound instead.

5.2.1 Hoeffding's Bound

Theorem 5.3 (Hoeffding's inequality)**.** *Let $X = \sum_{i=1}^n X_i$ where*

(i) $X_1, \ldots, X_n$ are independent,

(ii) $X_i \in [a_i, b_i]$ for all $i \in [n]$,

(iii) $\mathbb{E}[X_i] = 0$ for all $i \in [n]$.

Then for any $\eta > 0$,

$$\Pr[X \ge \eta] \le \exp\left(-\frac{2\eta^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

and

$$\Pr[X \le -\eta] \le \exp\left(-\frac{2\eta^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Comments.

(i) Suppose $X_i \in [-1, 1]$ for all i. Then $\sum_{i=1}^n (b_i - a_i)^2 = 4n$, and the bound becomes $\Pr[X \geq \eta] \leq e^{-\eta^2/(2n)}$.

(ii) Why do we assume $\mathbb{E}[X_i] = 0$? We can always replace X_i by $Y_i = X_i - \mathbb{E}[X_i]$; then $\mathbb{E}[Y_i] = 0$. If $X_i \in [a_i, b_i]$, then $Y_i \in [a_i - \mathbb{E}[X_i],\ b_i - \mathbb{E}[X_i]]$, and the width $b_i - a_i$ does not change.

Without assuming $\mathbb{E}[X_i] = 0$, the standard form of Hoeffding's inequality is:

$$\Pr[X - \mathbb{E}[X] \geq t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

5.2.2 Proof of Hoeffding's Inequality

Proof sketch. As with the multiplicative Chernoff bound, we use the exponential moment method. For a parameter $t > 0$,

$$\Pr[X \geq \eta] = \Pr[e^{tX} \geq e^{t\eta}] \leq \frac{\mathbb{E}[e^{tX}]}{e^{t\eta}} \quad \text{(by Markov's inequality)}.$$

So it boils down to upper bounding $\mathbb{E}[e^{tX}] = \prod_{i=1}^n \mathbb{E}[e^{tX_i}]$ (by independence) and choosing the best t.

Bounding $\mathbb{E}[e^{tX_i}]$. We sketch the key argument. The function e^{ty} is convex on the interval $[a_i, b_i]$ (indeed on the entire real line). We know $\mathbb{E}[X_i] = 0$ and $X_i \in [a_i, b_i]$. What distribution on $[a_i, b_i]$ with mean zero maximizes $\mathbb{E}[e^{tX_i}]$, since that is what gives the weakest bound?

Due to convexity, it turns out we should put all the probability mass on the extremes of the interval $[a_i, b_i]$. That is, the worst case is when $X_i \in \{a_i, b_i\}$ subject to $\mathbb{E}[X_i] = 0$ (note $a_i \leq 0$ and $b_i \geq 0$). Concretely, $X_i = b_i$ with probability $p = \frac{-a_i}{b_i - a_i}$ and $X_i = a_i$ with probability $1 - p = \frac{b_i}{b_i - a_i}$. One can prove this rigorously by convexity.

Assuming this worst case,

$$\mathbb{E}[e^{tX_i}] \leq p\, e^{tb_i} + (1 - p)\, e^{ta_i}.$$

By calculus (not completely obvious!) one can show that

$$p\, e^{tb_i} + (1 - p)\, e^{ta_i} \leq e^{t^2 (b_i - a_i)^2/8}.$$

Combining and optimizing. Assuming the bound above, we have

$$\mathbb{E}[e^{tX}] = \prod_{i=1}^n \mathbb{E}[e^{tX_i}] \leq \prod_{i=1}^n e^{t^2 (b_i - a_i)^2/8} = \exp\left(\frac{t^2}{8} \sum_{i=1}^n (b_i - a_i)^2\right).$$

Thus

$$\Pr[X \geq \eta] \leq \frac{\mathbb{E}[e^{tX}]}{e^{t\eta}} \leq \exp\left(\frac{t^2}{8} \sum_{i=1}^n (b_i - a_i)^2 - t\eta\right).$$

Minimizing over t: taking the derivative and setting it to zero,

$$\frac{t}{4} \sum_{i=1}^n (b_i - a_i)^2 = \eta \qquad \Longrightarrow \qquad t^* = \frac{4\eta}{\sum_{i=1}^n (b_i - a_i)^2}.$$

Plugging in, the exponent becomes

$$\frac{t^{*2}}{8}\sum_i (b_i-a_i)^2 - t^*\eta = \frac{16\eta^2}{8\left(\sum_i (b_i-a_i)^2\right)} - \frac{4\eta^2}{\sum_i (b_i-a_i)^2}$$
$$= \frac{2\eta^2}{\sum_i (b_i-a_i)^2} - \frac{4\eta^2}{\sum_i (b_i-a_i)^2} = -\frac{2\eta^2}{\sum_{i=1}^n (b_i-a_i)^2}.$$

Thus

$$\Pr[X \geq \eta] \leq \exp\left(-\frac{2\eta^2}{\sum_{i=1}^n (b_i-a_i)^2}\right).$$

The lower tail bound $\Pr[X \leq -\eta]$ follows by applying the same argument to $-X$. □

5.2.3 Application: Random Walk on the Line

Let $X = \sum_{i=1}^n X_i$ where $X_i \in \{-1, 1\}$ with $\mathbb{E}[X_i] = 0$. Here $b_i = 1$ and $a_i = -1$, so $\sum_{i=1}^n (b_i - a_i)^2 = 4n$. By Hoeffding's inequality,

$$\Pr[X > t\sqrt{n}] \leq e^{-2t^2 n/(4n)} = e^{-t^2/2}$$

and similarly

$$\Pr[X < -t\sqrt{n}] \leq e^{-t^2/2}.$$

This confirms that the random walk on the line is concentrated within $O(\sqrt{n})$ of the origin, with exponentially decaying tails.

Chapter 6

Johnson-Lindenstrauss Lemma and Hashing

6.1 Johnson-Lindenstrauss Lemma and Dimensionality Reduction

A fundamental yet simple result from convex geometry that has found many applications in data analysis and algorithms.

Let $\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_n$ be n vectors (points) in $\mathbb{R}^d$ where d is very large. The lemma says that one can project these vectors to a lower dimensional space $\mathbb{R}^k$ to create new vectors $\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_n$ such that for all $i, j \in [n]$:

$$(1-\epsilon)\|\overline{v}_i - \overline{v}_j\|_2 \leq \|\overline{u}_i - \overline{u}_j\|_2 \leq (1+\epsilon)\|\overline{v}_i - \overline{v}_j\|_2$$

i.e. the pairwise Euclidean distance is approximately preserved. And:

$$k = O\left(\frac{\log n}{\epsilon^2}\right)$$

Note that d does not show up.

Clearly this is useful only if d was originally larger than k.

Further the projection is via a linear map $\mathbb{R}^{k \times d}$ that is randomized and oblivious to the data.

The core of the result is about a single vector and we then use a union bound.

6.1.1 Distributional JL Lemma

Fix a vector $\overline{x} \in \mathbb{R}^d$. Let Π be a $k \times d$ matrix where $\Pi_{i,j}$ is chosen independently from a standard Gaussian distribution $N(0,1)$.

If $k = \Omega\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$ then with probability $\geq 1-\delta$:

$$(1-\epsilon)\|\overline{x}\|_2 \leq \left\|\frac{1}{\sqrt{k}}\Pi\overline{x}\right\|_2 \leq (1+\epsilon)\|\overline{x}\|_2$$

Assuming the DJL the dimensionality reduction is easy.

Choose $k = \Omega(\frac{1}{\epsilon^2}\log n)$. Then for $i \neq j$ look at vector $\overline{x} = \overline{v}_i - \overline{v}_j$. With probability $1 - \frac{1}{n^3}$, $\left\|\frac{1}{\sqrt{k}}(\Pi\overline{v}_i - \Pi\overline{v}_j)\right\|_2$ is within $(1 \pm \epsilon)\|\overline{v}_i - \overline{v}_j\|_2$. Let $\overline{u}_i = \frac{1}{\sqrt{k}}\Pi\overline{v}_i$.

So by union bound all pairs are preserved with probability $1 - \binom{n}{2}\frac{1}{n^3} \geq 1 - \frac{1}{n}$.

The projection is oblivious since Π was chosen without considering the data $\overline{v}_1, \ldots, \overline{v}_n$. This implies it can be used for "future" data.

6.1.2 Proof of the DJL Lemma

The proof relies on some nice properties of the Gaussian distribution. One can use other distributions, and there are some advantages to do so, but the analysis is more involved.

Sum of Independent Normally Distributed Random Variables

Recall $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. Its probability density function is:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Lemma 6.1. *If $Z_1 \sim N(\mu_1, \sigma_1^2)$ and $Z_2 \sim N(\mu_2, \sigma_2^2)$ and Z_1, Z_2 are independent, then $Z_1 + Z_2 \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.*

Corollary 6.2. *Suppose $\vec{z} = (z_1, z_2, \ldots, z_d)$ is a vector of independent $N(0,1)$ random variables. Let $\overline{x} \in \mathbb{R}^d$ be a unit vector, i.e., $\|\overline{x}\|_2 = 1$. Then $\sum_{i=1}^d x_i z_i$ is distributed as $N(0,1)$.*

Proof. The sum is a sum of independent Normal variables, as each $x_i z_i \sim N(0, x_i^2)$. The mean is $\sum_{i=1}^d \mathbb{E}[x_i z_i] = \sum_{i=1}^d x_i \mathbb{E}[z_i] = 0$. The variance is $\sum_{i=1}^d \mathrm{Var}(x_i z_i) = \sum_{i=1}^d x_i^2 \mathrm{Var}(z_i) = \sum_{i=1}^d x_i^2 = 1$, since $\overline{x}$ is a unit vector. □

Proof of Lemma (via Moment Generating Functions)

We will consider the moment generating function (MGF) of $X \sim N(\mu, \sigma^2)$, which is $\mathbb{E}[e^{tX}]$.

$$\mathbb{E}[e^{tX}] = \int_{-\infty}^{\infty} e^{tx} \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} dx$$

This evaluates to:

$$\mathbb{E}[e^{tX}] = e^{\mu t + \frac{\sigma^2 t^2}{2}}$$

Suppose we have k independent random variables $X_1, X_2, \ldots, X_k$. If $M_1(t), \ldots, M_k(t)$ are their respective MGFs, then the MGF of their sum $X = \sum X_i$ is:

$$M(t) = M_1(t) M_2(t) \ldots M_k(t)$$

This is because $\mathbb{E}[e^{tX}] = \mathbb{E}[e^{t\sum X_i}] = \mathbb{E}\left[\prod_{i=1}^k e^{tX_i}\right] = \prod_{i=1}^k \mathbb{E}[e^{tX_i}]$ by independence. If $X_i \sim N(\mu_i, \sigma_i^2)$, then the MGF for the sum is:

$$M(t) = \prod_{i=1}^k e^{\mu_i t + \frac{\sigma_i^2 t^2}{2}} = e^{(\sum_i \mu_i) t + \frac{(\sum_i \sigma_i^2) t^2}{2}}$$

This is the MGF of $N(\sum \mu_i, \sum \sigma_i^2)$.

Main Proof of DJL

Let Π be a $k \times d$ Gaussian matrix and let $\overline{x}$ be a unit vector (wlog). Let $\overline{y} = \Pi \overline{x} = (Y_1, Y_2, \ldots, Y_k)$. From the corollary, each $Y_i = \sum_{j=1}^d \Pi_{ij} x_j$ is distributed as $Y_i \sim N(0,1)$, and the Y_i are independent. We are interested in the squared norm $\|\overline{y}\|_2^2 = \sum_{i=1}^k Y_i^2$. The expectation is $\mathbb{E}[\|\overline{y}\|_2^2] = \sum_{i=1}^k \mathbb{E}[Y_i^2] = k$, since $\mathbb{E}[Y_i^2] = \mathrm{Var}(Y_i) + (\mathbb{E}[Y_i])^2 = 1 + 0 = 1$. This suggests that $\frac{1}{\sqrt{k}}\|\Pi\overline{x}\|_2$ should be close to 1.

Our goal is to prove that $Y = \sum_{i=1}^{k} Y_i^2$ is concentrated around its mean. The distribution of the sum of squares of t independent $N(0,1)$ random variables is called the χ^2 distribution with parameter t, denoted $\chi^2(t)$.

Lemma 6.3 (Concentration for χ^2)**.** *Let $Y_1, \ldots, Y_k$ be independent $N(0,1)$ random variables and let $Y = \sum_{i=1}^{k} Y_i^2$. Then for $\epsilon \in (0, 1/2)$:*

$$\Pr[(1-\epsilon)^2 k \leq Y \leq (1+\epsilon)^2 k] \geq 1 - 2e^{-c\epsilon^2 k}$$

where c is an absolute constant.

This concentration lemma implies the DJL lemma. By choosing $k = O(\frac{1}{\epsilon^2} \ln \frac{1}{\delta})$, we get $\Pr[(1-\epsilon) \leq \left\| \frac{1}{\sqrt{k}} \Pi \overline{x} \right\|_2 \leq (1+\epsilon)] \geq 1 - \delta$.

Proof of the Concentration Lemma

We use the exponential moment method (Chernoff bound). For $t \geq 0$:

$$\Pr[Y > \alpha] = \Pr[e^{tY} > e^{t\alpha}] \leq \frac{\mathbb{E}[e^{tY}]}{e^{t\alpha}}$$

The MGF for $Y \sim \chi^2(k)$ has a closed form for $t < 1/2$:

$$\mathbb{E}[e^{tY}] = (1-2t)^{-k/2}$$

Plugging this in, we get $\Pr[Y > \alpha] \leq (1-2t)^{-k/2} e^{-t\alpha}$ as long as $t \in (0, 1/2)$.

We choose $t = \left(1 - \frac{k}{\alpha}\right) \frac{1}{2}$ (which is in $(0, 1/2)$ when $\alpha > k$). Then:

$$\Pr[Y > \alpha] \leq \left(\frac{k}{\alpha}\right)^{k/2} e^{-(\alpha - k)/2}$$

For the upper tail, set $\alpha = (1+\epsilon)^2 k$:

$$\Pr[Y > (1+\epsilon)^2 k] \leq (1+\epsilon)^{-k} \cdot e^{-k[(1+\epsilon)^2 - 1]/2} = e^{-k\left[\frac{(1+\epsilon)^2 - 1}{2} - \ln(1+\epsilon)\right]}$$

Now using $\ln(1+\epsilon) = \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} - \cdots \leq \epsilon$, we get:

$$\frac{(1+\epsilon)^2 - 1}{2} - \ln(1+\epsilon) \geq \frac{\epsilon^2}{2} + \epsilon - \epsilon = \frac{\epsilon^2}{2}$$

Therefore $\Pr[Y > (1+\epsilon)^2 k] \leq e^{-k\epsilon^2/2}$. The lower tail is similar.

The difficult part is understanding $\mathbb{E}[e^{tY}]$. How do we get the closed form? Recall $Y = \sum_{i=1}^{k} Y_i^2$ where the Y_i are independent and each $Y_i \sim N(0,1)$. The MGF of Y_i^2 is:

$$M_{Y_i^2}(t) = \mathbb{E}[e^{tY_i^2}] = \int_{-\infty}^{\infty} e^{ty^2} \frac{e^{-y^2/2}}{\sqrt{2\pi}} dy = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}(1-2t)} dy$$

Rewriting:

$$= \frac{1}{\sqrt{1-2t}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi \cdot \frac{1}{1-2t}}} e^{-\frac{y^2}{2} \cdot \frac{1}{1/(1-2t)}} \, dy = \frac{1}{\sqrt{1-2t}} = (1-2t)^{-1/2}$$

since the integral is that of a $N(0, \frac{1}{1-2t})$ density and equals 1. This is defined for $t < 1/2$. Since the Y_i^2 are independent and identically distributed, the MGF of their sum $Y = \sum Y_i^2$ is the product of the individual MGFs:

$$\mathbb{E}[e^{tY}] = \prod_{i=1}^{k} (1-2t)^{-1/2} = (1-2t)^{-k/2}$$

6.2 New Topic: Hashing

This topic requires some background in pseudo-randomness and derandomization. How do we convert randomized algorithms into deterministic algorithms? Is it always possible? We will discuss some of these questions later, but our goal today is to introduce the notion of *limited independence*.

Two key observations motivate the study of pairwise independence:

- Pairwise independence suffices in some applications.
- One can generate many pairwise independent random bits from a small number of independent random bits.

Definition 6.4. A set of random variables $X_1, \ldots, X_n$ are *pairwise independent* (or 2-wise independent) if X_i and X_j are independent for any $i \neq j$.

Example 6.5. Let $X_1, X_2 \in \{0,1\}$ be independent random bits. Let $X_3 = X_1 \oplus X_2$ (XOR). Then X_1, X_2, X_3 are pairwise independent but not fully (mutually) independent.

Lemma 6.6. *One can construct $n = 2^k - 1$ pairwise independent random variables $X_1, \ldots, X_n$ (where each $X_i \in \{0,1\}$) from just k truly random bits $Y_1, \ldots, Y_k$.*

Construction: Let S be any non-empty subset of $\{1, 2, \ldots, k\}$. Define $X_S = \bigoplus_{i \in S} Y_i$. For any two distinct non-empty subsets S and T, the variables X_S and X_T are independent.

Proof. If $S \neq T$ then either $S - T \neq \emptyset$ or $T - S \neq \emptyset$ (or both).

Case 1: $S - T \neq \emptyset$. Let $i \in S - T$. For any choice of the bits in T, the value of $X_S = \bigoplus_{j \in S} Y_j$ depends on Y_i (which is not in T). Since Y_i is a uniformly random bit independent of all other bits, X_S is equally likely to be 0 or 1 regardless of the value of X_T. Hence X_S and X_T are independent.

Case 2: $T - S \neq \emptyset$. Same argument with the roles of S and T reversed. □

6.2.1 Application: Derandomizing the Max-Cut algorithm

Recall the simple randomized algorithm for Max-Cut on a graph $G = (V, E)$: start with $S = \emptyset$, and for each vertex $v \in V$, add v to S with probability $1/2$.

Recall the analysis. Let X_v be the indicator for $v \in S$, and let Y_e be the indicator for edge e being in the cut $\delta(S)$. For an edge $e = (u, v)$, we have $\Pr[Y_e = 1] = 1/2$ — this only requires X_u and X_v to be independent. Let $Y = \sum_e Y_e$. By linearity of expectation, $\mathbb{E}[Y] = \frac{1}{2}|E|$.

So if X_v, $v \in V$, are only pairwise independent, we still have $\mathbb{E}[Y] = \frac{1}{2}|E|$.

We can derandomize this as follows:

- We need $n = |V|$ pairwise independent random bits. We can generate these from a "seed" of $k = \lceil \log_2(n+1) \rceil$ true random bits using the construction above.
- The number of possible seeds is 2^k, which is $O(n)$.
- A deterministic algorithm can iterate through all $O(n)$ possible seeds. For each seed, it generates the n bits, defines the corresponding cut, and computes its size.
- The algorithm then outputs the largest cut found. Since the expected size is $|E|/2$, at least one of these deterministic outcomes must produce a cut of size at least $|E|/2$. This yields a deterministic polynomial-time approximation algorithm.

Chapter 7

K-wise Independence and Hash Tables

7.1 K-wise Independence and Hashing

Recall we defined pairwise independence and showed how to construct n pairwise independent bits $\{0,1\}$ from $O(\log n)$ truly random bits. We will initially focus on pairwise independent setting and now think about generating a pairwise independent random variables but from the large $\{0,1,2,\ldots,m-1\}$ instead of just bits.

Definition 7.1. $X_1, X_2, \ldots, X_n \in \{0,1,\ldots,m-1\}$ are pairwise independent if:

(i) X_i is uniformly distributed over $\{0,1,\ldots,m-1\}$ for all $i \in [n]$

(ii) For $i \neq j$, $\Pr[X_i = r \text{ and } X_j = s] = \frac{1}{m^2}$ for all $r, s \in \{0,1,\ldots,m-1\}$

Suppose $m = 2^h$. Then we can use previous bit scheme for each of the h bits. This would require $O(h \log n) = O(\log m \log n)$ bits and is not that efficient. We will see a way to get $O(\log n + \log m)$ bits.

We consider the setting $n \leq m$ and achieve a good scheme when m is a prime power p^k where p is a prime number. We recall some facts about prime numbers.

Lemma 7.2. *$\mathbb{Z}_p = \{0,1,2,\ldots,p-1\}$ under $+$ and $\times \bmod p$ is a field.*

Corollary 7.3. *Suppose $i, j \in \mathbb{Z}_p$ with $i \neq j$. Then for any $r, s \in \mathbb{Z}_p$, there exists a unique pair (a,b) that satisfy the equations:*

$$ai + b \equiv r \pmod{p} \tag{7.1}$$
$$aj + b \equiv s \pmod{p} \tag{7.2}$$

Proof. We solve these equations for a, b as we do over reals since $\mathbb{Z}_p$ is a field:

$$a(i-j) \equiv r - s \pmod{p} \tag{7.3}$$
$$a \equiv (r-s)(i-j)^{-1} \pmod{p} \tag{7.4}$$

Since $i \neq j$ and $\mathbb{Z}_p$ is a field, $(i-j)^{-1}$ exists and is unique in $\mathbb{Z}_p$. Then $b \equiv r - ai \pmod{p}$ or $b \equiv s - aj \pmod{p}$. □

7.1.1 Constructing pairwise independent random variables $X_0, X_1, \ldots, X_{p-1}$ with range $\{0, 1, 2, \ldots, p-1\}$

1. Pick $a, b \in \mathbb{Z}_p$ independently
2. For each i, set $X_i = ai + b \bmod p$

Claim 7.4. *X_i is uniformly distributed over $\mathbb{Z}_p$.*

Proof. Fix i. Then ai is fixed. Since b is uniformly random in $\mathbb{Z}_p$, $ai + b$ will be uniformly random in $\mathbb{Z}_p$. □

Claim 7.5. *For $i \neq j$, X_i and X_j are pairwise independent, i.e.,* $\Pr[X_i = r \text{ and } X_j = s] = \frac{1}{p^2}$.

Proof. There exists a unique $(a, b) \in \mathbb{Z}_p^2$ such that:

$$ai + b \equiv r \pmod{p} \tag{7.5}$$

$$aj + b \equiv s \pmod{p} \tag{7.6}$$

Therefore $\Pr[X_i = r \text{ and } X_j = s] = \frac{1}{p^2}$. □

Amount of randomness required is $2 \log p = O(\log n)$.

If one notices in the preceding construction, we only used the fact that $\mathbb{Z}_p$ is a field. Thus we could have done the same construction over any finite field. An important result in algebra is:

Theorem 7.6. *If $\mathbb{F}$ is a finite field, then $|\mathbb{F}| = p^k$ for some prime p and integer $k \geq 1$. Moreover, for every prime p and integer k, there is a finite field of order p^k, and all finite fields of the same order are isomorphic.*

Note that $+$ and $\times$ in a finite field of order p^k need to be defined, and one needs to handle the computational aspects. For our purposes, we will treat them as $O(1)$ time operations.

The advantage of using fields of size p^k is that we can use $p = 2$ and any integer k. Powers of 2 are natural for CS.

Thus we can obtain n pairwise independent random variables over $\{0, 1, \ldots, 2^k - 1\}$ when $n \leq 2^k$.

Suppose we want $n \leq m$. Say m is also a power of 2, $m = 2^h$. Then we can generate n r.v.s over $\{0, 1, \ldots, 2^h - 1\}$ and drop the first $\ell - h$ bits. It is an easy exercise to argue that this works.

$n > m$ is easy. Since we can generate n r.v.s and just use m of them.

7.2 K-wise Independence

Definition 7.7. $X_1, X_2, \ldots, X_n \in \{0, 1, \ldots, m-1\}$ are k-wise independent if:

(i) X_i for all $i \in [n]$ are uniformly distributed over $\{0, 1, \ldots, m-1\}$

(ii) Any k of the given random variables are independent

One can generalize the construction for $k = 2$ to larger k via polynomials.

Let $\mathbb{F}$ be a finite field of order m. Pick $a_0, a_1, a_2, \ldots, a_{k-1} \in \mathbb{F}$ uniformly and independently from $\mathbb{F}$.

Consider polynomial:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}$$

Now set each $X_i = p(i)$.

So we need to pick k coefficients, so total of $O(k \log m)$ bits.

7.2.1 Why does the construction work?

Suppose $i_1, i_2, \ldots, i_k$ are distinct elements of the field. Let $j_1, j_2, \ldots, j_k$ be arbitrary elements of the field. We claim $p(i_1) = j_1, p(i_2) = j_2, \ldots, p(i_k) = j_k$ is satisfied by a unique degree $\leq k-1$ polynomial.

First, there exists a degree $\leq k-1$ polynomial by Lagrange interpolation. Consider the polynomial:

$$p(x) = \sum_{\ell=1}^{k} j_\ell \prod_{h \neq \ell} \frac{(x - i_h)}{(i_\ell - i_h)}$$

which satisfies the k constraints:

$$p(i_1) = j_1, p(i_2) = j_2, \ldots, p(i_k) = j_k$$

Suppose we have two such polynomials p and q. Then consider $p - q$. This has k non-trivial roots, which is not possible for a non-zero polynomial of degree $< k$.

7.3 Hashing and Hash Tables

Dictionary data structure is perhaps the most basic and important data structure in programming. We want to store and retrieve a bunch of key-value pairs where keys are assumed to be distinct and come from some large universe U of objects. Say $|U| = N$.

However, at any time we only have a small subset $n \ll N$ of keys in our data structure.

7.3.1 Operations

- `insert(x)`: add x to stored set
- `find(x)`: is x in set?
- `delete(x)`: remove x from set if it is there

Since everything in a computer can be represented as a string, we can assume WLOG that U is ordered. Typically we deal with U being numbers, but in practice U can be complex objects such as images, tuples of strings, etc.

Dictionaries over an ordered universe U can be implemented via pointer-based tree data structures: $O(\log n)$ comparisons of unwieldy objects. Pointer-based data structures can be bad for cache access for many applications.

7.3.2 Hashing

Use array/tables so memory access is better. Hash functions allow mapping from unwieldy objects to small integers. Can get $O(1)$ on average.

If U is a finite universe of size N, $h : U \to [k]$ is a hash function. Let $\mathcal{H}_{all}$ be the set of all hash functions from U to $[k]$.

$\mathcal{H} \subseteq \mathcal{H}_{all}$ is a family of hash functions.

7.3.3 Hashing

1. Fix some family of "nice" hash functions $\mathcal{H} \subseteq \mathcal{H}_{all}$
2. Pick a random $h \in \mathcal{H}$
3. Let $S \subseteq U$ where $|S| = n$
4. Map each $x \in S$ to $h(x)$

Assumption: we have an array of size k where location i maps to location $h(x)$. Goal is to store x in location $h(x)$. But what if $h(x) = h(y)$ for some $x \neq y$? **Collision!**

Hashing data structures differ in how they handle collisions. We will see two ideas.

7.4 Hash tables with chaining

U is universe with N elements. We will assume that we want to store a set $S \subseteq U$ where $|S| = n$ and we know n. Hence we can allocate space roughly $O(n)$. If we don't know n, we can guess and double and rebuild.

Given knowledge of n, it is common to set up table size k to be cn for some small constant c, and then pick a hash family $\mathcal{H}$ of functions that map U to $[k]$, where $[k] = \{0, 1, \ldots, k-1\}$.

In chaining, all elements hashed to a bucket index $i \in [k]$ are stored in a linked list associated with i.

We hope that the hash function spreads the items nicely and that the lists at each bucket are small on average.

Operation: each of the operations `insert(x)`, `find(x)`, `delete(x)` take time proportional to $1 + \ell(x)$ where $\ell(x)$ is the length of the list of elements stored at $h(x)$.

7.4.1 What is a good hash function?

Suppose we have S of n elements and a hash table of size $k = n$. Then if we choose $\mathcal{H}_{all}$ as our family and pick $h \in \mathcal{H}_{all}$ at random, then it is like balls and bins: $h(x)$ for any x will be uniformly distributed, and for $x, y \in S$ with $x \neq y$, $h(x)$ and $h(y)$ will be independent.

We will call $\mathcal{H}_{all}$ an ideal hash family and a random h from it as an ideal hash function.

The problem is that $\mathcal{H}_{all}$ is a very large and complex family, and a random h has no structure, so computing $h(x)$ is not efficient.

So what we want are hash function families that have the following features:

1. For every $h \in \mathcal{H}$, h must be efficiently computable
2. Sampling a random h from $\mathcal{H}$ should be efficient
3. A random h from $\mathcal{H}$ should behave as closely as possible to an ideal hash function

Definition 7.8. A hash family $\mathcal{H}$ from U to $[k]$ is **strongly 2-universal** if the following properties hold for a random h chosen from $\mathcal{H}$:

(i) For all $x \in U$, $h(x)$ is uniformly distributed over $[k]$

(ii) For all $x, y \in U$ with $x \neq y$, $h(x)$ and $h(y)$ are independent, i.e., $\Pr[h(x) = i \text{ and } h(y) = j] = \frac{1}{k^2}$

We now define a weaker version:

Definition 7.9. A hash family $\mathcal{H}$ is **2-universal** if for all $x, y \in U$ with $x \neq y$:

$$\Pr[h(x) = h(y)] \leq \frac{1}{k}$$

Observation: A strongly 2-universal hash family is also 2-universal.

Suppose we had a 2-universal hash family:

Lemma 7.10. *Let $S \subseteq U$ with $|S| = n$. Fix $x \in S$. Suppose h is chosen from a 2-universal hash family $\mathcal{H} : U \to [k]$. Let S_x be the random set $\{y \in S : h(x) = h(y)\}$. Then $\mathbb{E}[|S_x|] \leq 1 + \frac{n-1}{k}$.*

Proof. Fix $y \in S$ with $y \neq x$. Let I_y be the indicator that $h(y) = h(x)$. Then:

$$|S_x| = 1 + \sum_{y \in S, y \neq x} I_y$$

$$\mathbb{E}[|S_x|] = 1 + \sum_{y \in S, y \neq x} \mathbb{E}[I_y] = 1 + \sum_{y \in S, y \neq x} \Pr[h(y) = h(x)] \leq 1 + \frac{n-1}{k}$$

□

Corollary 7.11. *For any sequence of n operations starting from an empty set, the expected cost of the operations is $O(n(1 + \frac{n}{k}))$. If $k = cn$ for constant $c > 0$, then expected cost is $O(n)$.*

7.4.2 How do we construct strongly 2-universal and 2-universal hash families?

We already did this when constructing pairwise independent random variables!

Say we create $X_0, X_1, \ldots, X_{n-1}$ pairwise independent r.v.s with range $[k]$. Then we think of X_i as $h(i)$.

To repeat: Let $N = 2^\ell$ for some ℓ and $k = 2^h$ for some $h \leq \ell$. Consider field $\mathbb{F}$ of size 2^ℓ. Associate U with $N = 2^\ell$.

$$\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{F}\}$$
$$h_{a,b}(i) = ai + b$$

For $i \neq j$, if a, b chosen independently and uniformly at random, $h_{a,b}(i)$ and $h_{a,b}(j)$ are independent and identically distributed over $\mathbb{F}$. We can then truncate the bits to get distribution over $[k]$.

7.5 2-Universal Family

Although the preceding construction works, there is a simpler family that yields a 2-universal family.

Let p be a prime $\geq N$. Define:

$$\mathcal{H} = \{h_{a,b} : a \in \{1, 2, \ldots, p-1\}, b \in \mathbb{Z}_p\}$$
$$h_{a,b}(i) = (ai + b \bmod p) \bmod k$$

Claim 7.12. *$\mathcal{H}$ is a 2-universal hash family.*

Note the difference with strongly 2-universal family: We did not allow a to be 0. This avoids the not-so-good/interesting hash function h_0 which maps all elements to 0. Second, by taking a simple mod$k$, we lose the uniformity property. Nevertheless, the family is 2-universal and is quite simple to describe and implement. Can use standard arithmetic on $\mathbb{Z}_p$.

7.5.1 Sketch of universality

Lemma 7.13. *Fix $i \neq j$ and $r \neq s \in \mathbb{Z}_p$. Exactly one pair (a, b) with $a \neq 0$ and $a, b \in \mathbb{Z}_p$ satisfies:*

$$ai + b \equiv r \pmod{p} \tag{7.7}$$

$$aj + b \equiv s \pmod{p} \tag{7.8}$$

Proof. $a(i-j) \equiv r-s \pmod{p}$, so $a \equiv (r-s)(i-j)^{-1} \pmod{p}$. If $i \neq j$, then $ai+b \equiv aj+b \bmod p$ iff $a = 0$, so no collisions before folding. □

Think of $h_{a,b}(i)$ as a two-step process:

1. $r = ai + b \bmod p$
2. $h = r \bmod k$

If $i \neq j$, $r \neq s$, but r can be equal to s after mod$k$.

Lemma 7.14. *The number of pairs $(r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$ with $r \neq s$ such that $r \bmod k = s \bmod k$ is at most $\frac{p(p-1)}{k}$.*

Proof. For any fixed r, the number of $s \in \mathbb{Z}_p$ such that $r \bmod k = s \bmod k$ is at most $\lceil p/k \rceil$, but this includes $s = r$, so the number with $s \neq r$ is at most $\lceil p/k \rceil - 1 \leq (p-1)/k$. Summing over all $r \in \mathbb{Z}_p$, the total is at most $p \cdot \frac{p-1}{k} = \frac{p(p-1)}{k}$. □

Fixing $i \neq j$, random $a \neq 0$ and $b \in \mathbb{Z}_p$ creates a random pair $(r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$ with $r \neq s$. There are $p(p-1)$ such pairs in total. The number of pairs (r, s) with $r \neq s$ that collide after folding is at most $\frac{p(p-1)}{k}$.

Hence $\Pr[h_{a,b}(i) = h_{a,b}(j)] \leq \frac{p(p-1)}{k \cdot p(p-1)} = \frac{1}{k}$.

7.6 Hash Tables: Linear Probing

Chaining is simple and easy to analyze. In terms of practice, using linked lists and dynamic memory allocation is not so great. There are other hashing techniques such as linear probing and cuckoo hashing that try to take advantage of arrays.

Linear probing is a technique that handles collisions by scanning along the array. If $h(x)$ is occupied, see Kent's notes for description.

Chapter 8

Hash Tables with Linear Probing

8.1 Hash Tables with Linear Probing

We saw hashing with chaining. Using universal hashing we get expected $O(1)$ time per operation. One disadvantage is that chaining requires a list data structure at each bucket. Today we will discuss another popular technique called linear probing. We will mostly be following Kent Quanrud's thesis, which has nice figures and more detailed explanations, including historical notes. For this reason, we will be high-level in our description.

8.1.1 Linear Probing

Let the universe be U, with $|U| = u$. The size of the hash table is $A[0 \dots m-1]$. We pick a random hash function h from a hash family H.

Algorithm 1 insert(x)

```
i ← h(x)
while A[i] is not empty do
    i ← (i + 1) (mod m)
A[i] ← x
```

Algorithm 2 find(x)

```
i ← h(x)
while A[i] is not empty do
    if A[i] = x then
        return Yes
    i ← (i + 1) (mod m)
return No
```

The `delete(x)` operation is more complicated. There are different strategies, but we want to maintain the correctness of `insert` and `find`. To delete x, we first find it. Say x is in $A[i]$. If $h(x) = i$ then we set $A[i] =$ empty and we are done. Otherwise we have inserted x into a location "after" $h(x)$ due to collisions. If we remove x from $A[i]$ we create a "hole." We try to fill it by scanning from i to the right to see if we encounter another element y such that $A[j] = y$ but $h(y) \neq j$, and move it to the hole. We repeat this process.

Algorithm 3 delete(x)

```
i ← find(x)                                   ▷ assume x is in A
A[i] ← empty
repeat
    j ← (i + 1) (mod m)
    while A[j] is not empty and h(A[j]) = j do
        j ← (j + 1) (mod m)
    if A[j] is empty then
        break
    else
        A[i] ← A[j]
        i ← j
until true
```

8.2 Analysis

Let's assume "ideal" hash functions (each key is mapped to a slot uniformly and independently). How can we upper bound the cost of the operations? Suppose we perform n operations. Let S be the set of elements that were ever inserted. We will assume $m > 2en$. We will consider the state of the hash table as if we had inserted all the elements in S.

The hash table A will be broken into "runs", where a run is a maximal contiguous interval of occupied cells. The key observation is the following: The cost of `insert(x)`, `find(x)`, and `delete(x)` are all upper-bounded by the length of the run that contains the slot $h(x)$. Let's call this run $R(x)$.

Our goal is to analyze the expected length of this run. What is $\mathbb{E}[|R(x)|]$?

Lemma 8.1. *If $m > 2en$, under the ideal hashing assumption, then for any x, $\mathbb{E}[|R(x)|] = O(1)$.*

This lemma implies that the total expected cost for n operations is $O(n)$.

Proof. Let R denote the run $R(x)$, and let $h(x) = i$. The expected length of the run is:

$$\mathbb{E}[|R|] = \sum_{l=1}^{n} l \cdot \Pr[|R| = l]$$

Consider an interval I of length l that contains the slot i. There are l such possible intervals. Let's fix one such interval, say I_j. For the run containing i to be exactly this interval I_j, two conditions must be met:

1. All l slots in I_j must be occupied by elements from S.
2. The two slots bordering I_j must be empty.

By symmetry, the probability that the run R is equal to any specific valid interval of length l is the same. Thus, $\Pr[|R| = l]$ is related to $l \cdot \Pr[R = I_j]$ for a fixed I_j.

Let's find an upper bound on the probability that a specific interval I_j of length l contains the run. For this to happen, at least l elements from S must hash into I_j. The probability of this is:

$$\Pr[\text{at least } l \text{ items of } S \text{ hash to } I_j] \leq \binom{n}{l} \left(\frac{l}{m}\right)^l$$

Using the inequality $\binom{n}{k} \leq (\frac{en}{k})^k$, we get:

$$\leq \left(\frac{en}{l}\right)^l \left(\frac{l}{m}\right)^l = \left(\frac{en}{m}\right)^l$$

Given our assumption that $m \geq 2en$, we have $\frac{en}{m} \leq \frac{1}{2}$. Therefore, the probability is at most $(\frac{1}{2})^l$. Now we can bound the expected run length.

$$\mathbb{E}[|R|] \leq \sum_{l=1}^{n} l \cdot \left(l \cdot \frac{1}{2^l}\right) = \sum_{l=1}^{n} \frac{l^2}{2^l}$$

This sum $\sum_{l=1}^{\infty} l^2 x^l$ converges for $|x| < 1$. For $x = 1/2$, the sum is a constant. Thus, $\mathbb{E}[|R|] = O(1)$. □

8.2.1 Analysis with 5-Universal Hashing

The above analysis assumed ideal hashing. It turns out that a similar result can be obtained with a weaker assumption: 5-universal hashing.

Lemma 8.2. *Suppose H is a 5-strongly universal hash family from $[u] \to [m]$ and $m \geq 8n$. Then the expected cost of each of the first n operations is $O(1)$.*

To prove this, we need a concentration inequality for 4-wise independent random variables, which generalizes Chebyshev's inequality.

Lemma 8.3 (Concentration Inequality)**.** *Suppose $X_1, \ldots, X_n \in \{0, 1\}$ are 4-wise independent random variables. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then for any $\beta > 0$:*

$$\Pr[X \geq \mu + \beta] \leq \frac{\mu + 3\mu^2}{\beta^4}$$

Proof of $O(1)$ cost using 5-universal hashing. Let's assume the concentration lemma. We want to bound $\mathbb{E}[|R|]$. We can write:

$$\mathbb{E}[|R|] = \sum_{l=1}^{\infty} \Pr[|R| \geq l] \leq \sum_{k=1}^{\lceil \log n \rceil} 2^k \Pr[|R| > 2^{k-1}]$$

Let's bound the probability $\Pr[|R| > 2^{k-1}]$. Let $h(x) = i$. Consider an interval I_k of length 2^{k+1} centered at i. If the run $R(x)$ has length greater than 2^{k-1}, then at least 2^{k-1} elements must have hashed into an interval of size at most 2^{k+1} around i.

Let X_a be the indicator random variable that element $a \in S \setminus \{x\}$ hashes into I_k. Let $X = \sum_{a \in S \setminus \{x\}} X_a$. Since our hash family is 5-strongly universal, conditioning on $h(x) = i$ leaves the hash values of other elements 4-wise independent. Thus, the variables X_a are 4-wise independent.

The expected value of X is:

$$\mu = \mathbb{E}[X] = \sum_{a \in S \setminus \{x\}} \mathbb{E}[X_a] = \sum_{a \in S \setminus \{x\}} \Pr[h(a) \in I_k] \leq n \cdot \frac{|I_k|}{m} = n \cdot \frac{2^{k+1}}{m}$$

With $m \geq 8n$, we have:

$$\mu \leq n \cdot \frac{2^{k+1}}{8n} = \frac{2^{k+1}}{8} = 2^{k-2}$$

The event $|R| > 2^{k-1}$ implies that $X \geq 2^{k-1}$. We use our concentration lemma to bound this probability. Let $\beta = 2^{k-1} - \mu \geq 2^{k-1} - 2^{k-2} = 2^{k-2}$.

$$\Pr[X \geq 2^{k-1}] = \Pr[X \geq \mu + (2^{k-1} - \mu)] \leq \Pr[X \geq \mu + 2^{k-2}]$$

Applying the lemma with $\beta = 2^{k-2}$ and $\mu \leq 2^{k-2}$:

$$\leq \frac{\mu + 3\mu^2}{\beta^4} \leq \frac{2^{k-2} + 3(2^{k-2})^2}{(2^{k-2})^4} = \frac{1 + 3 \cdot 2^{k-2}}{(2^{k-2})^3} \leq \frac{4 \cdot 2^{k-2}}{(2^{k-2})^3} = \frac{4}{(2^{k-2})^2}$$

Now we plug this back into the sum for $\mathbb{E}[|R|]$:

$$\mathbb{E}[|R|] \leq \sum_{k=1}^{\lceil \log n \rceil} 2^k \cdot \Pr[|R| > 2^{k-1}] \leq \sum_{k=1}^{\infty} 2^k \cdot \frac{4}{(2^{k-2})^2} = \sum_{k=1}^{\infty} 2^k \cdot \frac{4}{2^{2k-4}} = \sum_{k=1}^{\infty} \frac{64 \cdot 2^k}{2^{2k}} = \sum_{k=1}^{\infty} \frac{64}{2^k}$$

This is a convergent geometric series, so $\mathbb{E}[|R|] = O(1)$. □

Proof of Concentration Lemma. We want to prove $\Pr[X \geq \mu + \beta] \leq \frac{\mu + 3\mu^2}{\beta^4}$. Using Markov's inequality on the non-negative random variable $(X - \mu)^4$:

$$\Pr[X - \mu \geq \beta] = \Pr[(X - \mu)^4 \geq \beta^4] \leq \frac{\mathbb{E}[(X - \mu)^4]}{\beta^4}$$

Our task is to bound $\mathbb{E}[(X - \mu)^4]$. Let $p_i = \mathbb{E}[X_i]$, so $\mu = \sum p_i$. Let $Y_i = X_i - p_i$. Note that $\mathbb{E}[Y_i] = 0$. Then $X - \mu = \sum Y_i$.

$$\mathbb{E}[(X - \mu)^4] = \mathbb{E}\left[\left(\sum_{i=1}^{n} Y_i\right)^4\right] = \mathbb{E}\left[\sum_{i,j,k,l \in [n]} Y_i Y_j Y_k Y_l\right]$$

By linearity of expectation, this is $\sum_{i,j,k,l} \mathbb{E}[Y_i Y_j Y_k Y_l]$. Because the variables are 4-wise independent and $\mathbb{E}[Y_i] = 0$, any term where an index appears only once will be zero. For example, $\mathbb{E}[Y_i Y_j^3] = \mathbb{E}[Y_i]\mathbb{E}[Y_j^3] = 0$ for $i \neq j$. The only non-zero terms are of the form $\mathbb{E}[Y_i^4]$ and $\mathbb{E}[Y_i^2 Y_j^2]$ for $i \neq j$. The expansion of $(\sum Y_i)^4$ gives:

- Terms of type Y_i^4: There are n such terms. The term is $\sum_i \mathbb{E}[Y_i^4]$.
- Terms of type $Y_i^2 Y_j^2$: There are $\binom{n}{2}$ pairs $\{i, j\}$, and the coefficient for each is $\binom{4}{2} = 6$. The term is $6 \sum_{i<j} \mathbb{E}[Y_i^2]\mathbb{E}[Y_j^2]$.

So, $\mathbb{E}[(X - \mu)^4] = \sum_{i=1}^{n} \mathbb{E}[Y_i^4] + 6 \sum_{i<j} \mathbb{E}[Y_i^2]\mathbb{E}[Y_j^2]$. Let's bound these expectations. Since $X_i \in \{0, 1\}$, $X_i^k = X_i$ for $k \geq 1$. $\mathbb{E}[Y_i^2] = \mathbb{E}[(X_i - p_i)^2] = \mathbb{E}[X_i^2 - 2p_i X_i + p_i^2] = p_i - 2p_i^2 + p_i^2 = p_i - p_i^2 = p_i(1 - p_i) \leq p_i$. $\mathbb{E}[Y_i^4] = \mathbb{E}[(X_i - p_i)^4] = p_i(1 - p_i)^4 + (1 - p_i)(-p_i)^4 = p_i(1 - p_i)^4 + (1 - p_i)p_i^4$. Since $(1 - p_i) \leq 1$ and $p_i \leq 1$, we have $\mathbb{E}[Y_i^4] \leq p_i(1 - p_i) + p_i^2(1 - p_i) \leq p_i$. So, $\sum_i \mathbb{E}[Y_i^4] \leq \sum_i p_i = \mu$. And $6 \sum_{i<j} \mathbb{E}[Y_i^2]\mathbb{E}[Y_j^2] = 6 \sum_{i<j} p_i(1 - p_i)p_j(1 - p_j) \leq 6 \sum_{i<j} p_i p_j$. We know that $2 \sum_{i<j} p_i p_j = (\sum_i p_i)^2 - \sum_i p_i^2 \leq (\sum_i p_i)^2 = \mu^2$. So, $6 \sum_{i<j} p_i p_j \leq 3\mu^2$. Putting it all together:

$$\mathbb{E}[(X - \mu)^4] \leq \mu + 3\mu^2$$

Finally, substituting this back into the Markov inequality expression gives the desired result:

$$\Pr[X - \mu \geq \beta] \leq \frac{\mu + 3\mu^2}{\beta^4}$$

□

Chapter 9

Streaming Algorithms

9.1 Introduction to Streaming Algorithms

This lecture provides an introduction to streaming, sampling, and estimation algorithms.

9.1.1 Setting

We are given a set of objects or tokens that arrive one by one in a stream (online fashion):

$$e_1, e_2, \ldots, e_m$$

Examples of stream elements e_i:

- A number from a set $[n] = \{1, \ldots, n\}$.
- An edge (u, v) in a graph.
- A vector or point in $\mathbb{R}^d$.
- A row or column of a matrix.
- A matrix itself.

Assumption: The stream length m is very large and possibly unknown. We cannot afford to store all elements $e_1, \ldots, e_m$ in memory. We have a limited memory space, say B units (where one token takes one unit of space).

Question: What functions of the input stream can we compute? This depends on the available space B. There are various trade-offs between efficiency, accuracy, etc.

9.1.2 Frequency Vectors

A simple but powerful setting is when each element e_i is an integer from a large range, say $[n] = \{0, 1, \ldots, n-1\}$. For example, consider a stream: $5, 3, 2, 2, 10, 5, 90, \ldots$ This stream implicitly defines a frequency vector $f \in \mathbb{N}^n$ that starts at all zeros. For each incoming element e_i, we effectively perform an update $f_{e_i} \leftarrow f_{e_i} + 1$. After the stream $5, 3, 2, 2, 10, 5, 90$, the frequency vector would have $f_2 = 2, f_3 = 1, f_5 = 2, f_{10} = 1, f_{90} = 1$, and all other entries would be zero.

9.2 Frequency Moment Estimation

Given a stream $e_1, \ldots, e_m$, let f_i be the frequency of element $i \in [n]$ at the end of the stream. We want to estimate or compute the k^{th} frequency moment, denoted by F_k.

$$F_k = \sum_{i=1}^{n} f_i^k$$

(Note: Sometimes this is defined as the L_k-norm of the frequency vector, i.e., $(\sum_{i=1}^{n} f_i^k)^{1/k}$).

- **k=0:** $F_0 = \sum_{f_i>0} 1$. This is the number of **distinct elements** in the stream.
- **k=1:** $F_1 = \sum f_i = m$. This is the total length of the stream.
- **k=2:** $F_2 = \sum f_i^2$. This is related to the squared Euclidean norm (L_2 norm) of the frequency vector and is very important in many applications.
- **k=∞:** $F_\infty = \max_i f_i$. This is the frequency of the most frequent element, used in finding "heavy hitters".

If we can store the entire stream or the frequency vector f, these problems are trivial. The main question is: can we compute or estimate these moments with memory $B \ll n$? The answer is yes, often with $B = \tilde{O}(1)$ (polylogarithmic space), but this requires randomization and approximation. It can be shown that without one of these, it's not feasible with sub-linear ($o(n)$) memory.

9.3 Reservoir Sampling

9.3.1 Sampling One Element

Given a stream $e_1, \ldots, e_m$ where m is unknown, we want to pick one sample uniformly at random.

Algorithm 4 Reservoir Sampling (1 element)

```
m ← 0
s ← null
while stream has next element e do
    m ← m + 1
    With probability 1/m, set s ← e.
output s
```

9.3.2 Sampling k Elements

To get k samples without replacement, we can extend the algorithm.

Algorithm 5 Reservoir Sampling (k elements)

$S \leftarrow \emptyset$
$m \leftarrow 0$
while stream has next element e_m **do**
    $m \leftarrow m + 1$
    if $m \leq k$ **then**
        Add e_m to S.
    else
        With probability k/m:
        Replace a random element of S with e_m.
output S

Exercise 9.1. Prove the correctness of the k-element Reservoir Sampling algorithm.

9.4 Distinct Element Estimation (F_0)

The problem is to find the number of distinct elements in a stream $e_1, \ldots, e_m$ where each $e_i \in [n]$.

- **Example:** In the stream $1, 2, 5, 2, 3, 5, 5, 1$, the distinct elements are $\{1, 2, 3, 5\}$, so $F_0 = 4$.
- **Application:** In a high-speed internet switch, packets are tuples of (source, destination, content). Source and destination are IP addresses (e.g., 128-bit values, so $n = 2^{128}$). A key question is: how many distinct source addresses have sent packets?

An offline solution could use a hash table or dictionary to store distinct elements seen so far. This would require $\Theta(F_0)$ space, which can be very large and is not suitable for a streaming model where F_0 is unknown and potentially massive.

It can be shown that any deterministic algorithm that gives a $(1 \pm \epsilon)$ approximation requires $\Omega(n)$ space. However, with randomization, we can get a $(1 \pm \epsilon)$ estimate with probability at least $(1 - \delta)$ in $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \cdot \text{polylog}(n))$ space.

9.4.1 Hashing-Based Algorithm (Flajolet-Martin)

Assume we have access to an "ideal" hash function $h : [n] \to [0, 1]$ that maps each item to a uniformly random real number.

Algorithm 6 Basic Distinct Elements Hashing

Pick a random hash function $h : [n] \to [0, 1]$.
$z \leftarrow \infty$.
while stream has next element e_i **do**
    $z \leftarrow \min(h(e_i), z)$.
output $\frac{1}{z} - 1$.

This algorithm uses memory for only one number, z.

Lemma 9.2. *Suppose $X_1, X_2, \ldots, X_k$ are independent random variables uniformly distributed in $[0, 1]$. Let $Y = \min_{i=1,\ldots,k} X_i$. Then:*

1. $\mathbb{E}[Y] = \frac{1}{k+1}$

2. $\text{Var}(Y) = \frac{k}{(k+1)^2(k+2)}$

Proof Sketch. Let $F_Y(t)$ and $f_Y(t)$ be the CDF and PDF of Y. For $t \in (0,1)$:

$$F_Y(t) = P(Y \leq t) = 1 - P(Y > t) = 1 - P(X_1 > t, \ldots, X_k > t) = 1 - (1-t)^k$$

The PDF is the derivative:

$$f_Y(t) = k(1-t)^{k-1}$$

The expectation is:

$$\mathbb{E}[Y] = \int_0^1 t \cdot f_Y(t)dt = \int_0^1 tk(1-t)^{k-1}dt = \frac{1}{k+1}$$

Similarly, one can calculate $\mathbb{E}[Y^2] = \frac{2}{(k+1)(k+2)}$, which gives:

$$\text{Var}(Y) = \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2 = \frac{2}{(k+1)(k+2)} - \frac{1}{(k+1)^2} = \frac{k}{(k+1)^2(k+2)}$$

□

If $F_0 = k$, our estimator Z is the minimum of k uniform random variables. The algorithm's output $\frac{1}{Z} - 1$ is an unbiased estimator for k, since $\mathbb{E}[\frac{1}{Z} - 1] \approx \frac{1}{\mathbb{E}[Z]} - 1 = \frac{1}{1/(k+1)} - 1 = k$. However, the variance is too high for a good approximation with a single estimate. Using Chebyshev's inequality shows that the probability of being close to the mean is not high enough.

9.4.2 Variance Reduction

Averaging

To reduce variance, we can average multiple independent estimators. Let $Y_1, \ldots, Y_h$ be i.i.d. random variables with mean μ and variance σ^2. Let their average be $\bar{Y} = \frac{1}{h}\sum_{i=1}^h Y_i$. Then:

$$\mathbb{E}[\bar{Y}] = \mu \quad \text{and} \quad \text{Var}(\bar{Y}) = \frac{\sigma^2}{h}$$

The variance is reduced by a factor of h.

We apply this to our problem. We run the basic algorithm h times in parallel, each with an independent hash function h_j. This gives us h minimum values $z_1, \ldots, z_h$. Let $Z = \frac{1}{h}\sum_{j=1}^h z_j$. Our final output is $Y = \frac{1}{Z} - 1$.

- $\mathbb{E}[Z] = \frac{1}{k+1}$
- $\text{Var}(Z) = \frac{\text{Var}(z_j)}{h} \leq \frac{1}{h(k+1)(k+2)}$

Using Chebyshev's inequality on Z:

$$\Pr\left[|Z - \mathbb{E}[Z]| \geq \delta \cdot \mathbb{E}[Z]\right] \leq \frac{\text{Var}(Z)}{(\delta\mathbb{E}[Z])^2}$$

For $\frac{1}{Z} - 1$ to be within $(1 \pm \epsilon)$ of k, we need Z to be within $O(\epsilon)/(k+1)$ of its true value $1/(k+1)$. Specifically:

$$\Pr\left[\left|Z - \frac{1}{k+1}\right| > \frac{\epsilon}{3(k+1)}\right] \leq \frac{\text{Var}(Z)}{\epsilon^2/(9(k+1)^2)} \leq \frac{1/(h(k+1)(k+2))}{\epsilon^2/(9(k+1)^2)} = \frac{9(k+1)}{h(k+2)\epsilon^2} \leq \frac{18}{h\epsilon^2}$$

Thus if $\frac{18}{h\epsilon^2} \leq \frac{1}{10}$, then with probability at least $9/10$, $Z \in (1 \pm \epsilon/3)\frac{1}{k+1}$, which implies $\frac{1}{Z} - 1 \in (1 \pm \epsilon)k$. This requires $h \geq 180/\epsilon^2$.

Exercise 9.3. To get a $(1 \pm \epsilon)$ approximation with probability at least $(1 - \delta)$ via Chebyshev's inequality alone (without the median trick), argue that $h = \Omega\left(\frac{1}{\epsilon^2} \cdot \frac{1}{\delta}\right)$ suffices. Note the worse dependence on δ compared to the median trick below.

Median Trick

To boost the success probability from a constant to $(1 - \delta)$, we can use the median trick. This provides a better dependency on δ than simple averaging.

1. Create a "base estimator" that is a $(1 \pm \epsilon)$-approximation with probability at least $3/4$. We can do this by averaging $h = O(1/\epsilon^2)$ basic estimators as described above.

2. Run this base estimator l times independently to get estimates $Y_1, Y_2, \ldots, Y_l$.

3. Output $Y = \text{median}(Y_1, \ldots, Y_l)$.

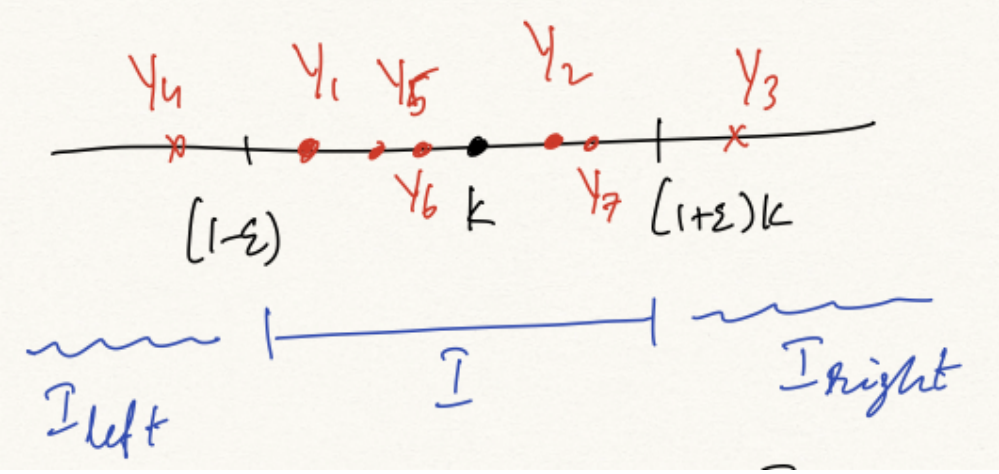

Figure 9.1: Illustration of the median trick from the lecture notes. The true value is k. The interval $[(1 - \epsilon)k, (1 + \epsilon)k]$ contains most estimates $(Y_1, Y_5, Y_6, Y_2, Y_7)$. Outliers like Y_3 and Y_4 are ignored by taking the median, which is much more likely to fall within the desired interval.

Lemma 9.4. *Let Y_i be independent estimators such that $\Pr[(1 - \epsilon)k \leq Y_i \leq (1 + \epsilon)k] \geq 3/4$. If we take $l = O(\log(1/\delta))$, then the median Y is a $(1 \pm \epsilon)$ approximation of k with probability at least $1 - \delta$.*

Proof Sketch using Chernoff Bounds. Let A_i be the event that Y_i is a "good" estimate (within the desired interval). $P(A_i) \geq 3/4$. For the median to be "bad" (outside the interval), at least $l/2$ of the estimators Y_i must be "bad". Let B_i be an indicator variable for Y_i being "bad". $P(B_i = 1) \leq 1/4$. Let $B = \sum B_i$. $\mathbb{E}[B] \leq l/4$. We want to bound the probability $P(B \geq l/2)$. By Chernoff bounds, this probability is exponentially small in l.

$$P(B \geq l/2) = P(B \geq 2\mathbb{E}[B]) \leq e^{-c \cdot l/4}$$

By setting $l = O(\log(1/\delta))$, we can make this probability less than δ. □

9.4.3 Final Hashing Algorithm and Analysis

The full algorithm combines both techniques.

Algorithm 7 Full Distinct Elements Algorithm

Given ϵ, δ. Let $h = O(1/\epsilon^2)$ and $l = O(\log(1/\delta))$.
for $i = 1$ to l **do**
    Run h parallel instances of the basic hashing algorithm with independent hash functions $h_{i,1}, \ldots, h_{i,h}$ to get minimums $z_{i,1}, \ldots, z_{i,h}$.
    $X_i \leftarrow \frac{1}{h} \sum_{j=1}^{h} z_{i,j}$.
    $Y_i \leftarrow \frac{1}{X_i} - 1$.
output median$(Y_1, \ldots, Y_l)$.

This algorithm uses $O(h \cdot l) = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ space (for storing the minimums). It outputs a $(1 \pm \epsilon)$ approximation with probability at least $(1 - \delta)$.

We abstract out the technique used above.

Definition 9.5. An (ϵ, δ) *randomized scheme* for a non-negative quantity α is a randomized algorithm that outputs a value X such that

$$\Pr\left[(1-\epsilon)\alpha \leq X \leq (1+\epsilon)\alpha\right] \geq 1 - \delta.$$

Lemma 9.6. *Suppose we have a randomized algorithm that outputs $X \geq 0$ with $\mathbb{E}[X] = \alpha$ and $\mathrm{Var}(X) \leq \beta \cdot \alpha^2$. Then one can use $O\left(\frac{\beta}{\epsilon^2} \ln \frac{1}{\delta}\right)$ independent runs of the algorithm to obtain an (ϵ, δ)-scheme.*

Practical considerations:

- The "ideal" hash function to $[0, 1]$ can be approximated by hashing to a large integer range. We can discretize $[0, 1]$ by $\{0, \frac{1}{n^3}, \frac{2}{n^3}, \ldots, 1\}$, so a hash function $h : [n] \to [n^3]$ suffices.
- Pairwise independent hash functions are sufficient for the basic estimator.
- Some small tweaks to the algorithm are needed when using limited independence.

9.5 A Simpler Sampling-Based Algorithm

A more recent and surprisingly simple algorithm due to [CMV22] is based on sampling. **Idea:** Suppose we sample each of the F_0 distinct elements with a fixed probability p. If we end up with a sample B, then $\mathbb{E}[|B|] = p \cdot F_0$. So, $|B|/p$ is an unbiased estimator for F_0.

Challenges:

1. How do we sample from the set of *distinct* elements when the stream contains duplicates?
2. How do we choose p? If p is too small, $|B|$ will be small and the estimate inaccurate. If p is too large, the sample set B might be too large to store in memory.

Solution to (1): We can adapt the sampling procedure. Maintain a sample set B. For each new element e_m:

1. If e_m is already in B, remove it.

2. Add e_m to B with probability p.

This procedure ensures that at the end of the stream, each distinct element is present in the final set B with probability p.

Solution to (2): The key idea is to adapt the sampling probability p dynamically. Start with $p = 1$ and decrease it whenever the sample set B grows too large.

Algorithm 8 Adaptive Distinct Element Sampling

Given ϵ. Choose a threshold $\tau = O(\frac{\log n}{\epsilon^2})$.
$p \leftarrow 1$, $B \leftarrow \emptyset$.
while stream has next element e_m **do**
    $B \leftarrow B \setminus \{e_m\}$ *(remove if present)*
    With probability p, add e_m to B.
    if $|B| \geq \tau$ **then**
        $p \leftarrow p/2$.
        Subsample B: for each element in B, discard it with probability $1/2$.
output $|B|/p$.

Theorem 9.7. *The adaptive sampling algorithm uses $O(\frac{\log n}{\epsilon^2})$ words of memory and outputs an estimate that is within $(1 \pm \epsilon)F_0$ with high probability.*

Chapter 10

Frequency Estimation in Streams

10.1 Lecture 10

10.1.1 AMS estimator and estimation F_2

Recall we want to estimate frequency moment F_k. Let $\sigma = e_1, \ldots, e_m \in [n]^m$.

$$F_k = \sum_{i=1}^{n} (f_i)^k$$

We will consider a more abstract and general estimation problem. Let $g_i : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ be a real valued function with $g_i(0) = 0$. Say we want to estimate

$$g(\sigma) = \sum_{i=1}^{n} g_i(f_i)$$

Note that we are allowing different functions for different i.

$$F_k = \sum_{i=1}^{n} g(f_i) \text{ where } g(x) = x^k.$$

Alon-Matias-Szegedy (AMS) in their influential paper provided an unbiased estimator for computing $g(\sigma)$.

10.1.2 AMS-Estimator (g)

- Sample e_J uniformly at random from stream. Say $e_J = i$ where $i \in [n]$.
- Let $R = |\{j \mid J \leq j \leq m, e_j = i\}|$.
- Output $m \cdot (g_i(R) - g_i(R-1))$.

10.1.3 Implementation via Reservoir Sampling

$s \leftarrow$ null
$m \leftarrow 0$
$R \leftarrow 0$
while (stream is not empty)

$m \leftarrow m + 1$
$e_m \leftarrow$ current item
If $(s == e_m)$ $R \leftarrow R + 1$
with prob $\frac{1}{m}$
$s \leftarrow e_m$
$R \leftarrow 1$
end while
Output $m \cdot (g_i(R) - g_i(R-1))$ where $s = i$.

10.1.4 Analysis

Lemma: Let Y be the output of the algorithm. Then $\mathbb{E}[Y] = g(\sigma)$.

Proof.

$$\begin{aligned}
\mathbb{E}[Y] &= m \sum_{i=1}^{n} \mathbb{E}[Y | e_J = i] \Pr[e_J = i] \\
&= m \sum_{i=1}^{n} \frac{f_i}{m} \mathbb{E}[Y | e_J = i] \\
&= m \sum_{i=1}^{n} \frac{f_i}{m} \cdot \sum_{j=1}^{f_i} \frac{g_i(j) - g_i(j-1)}{f_i} \\
&= \sum_{i=1}^{n} g_i(f_i)
\end{aligned}$$

Thus we can use AMS estimator for F_k as well. But we need to understand the variance of the estimator. □

Lemma: For $g(x) = x^k$, we have

$$\mathrm{Var}(Y) \leq k F_1 F_{2k-1} \leq k n^{1-\frac{1}{k}} F_k^2$$

Proof. $\mathrm{Var}(Y) \leq \mathbb{E}[Y^2]$

$$\begin{aligned}
\mathbb{E}[Y^2] &= \sum_{i=1}^{n} \Pr[e_J = i] \cdot \sum_{l=1}^{f_i} \frac{m^2}{f_i} (l^k - (l-1)^k)^2 \\
&\approx \sum_{i=1}^{n} \frac{f_i}{m} \cdot \frac{m^2}{f_i} \sum_{l=1}^{f_i} (l^k - (l-1)^k)^2 \\
&\leq F_1 \sum_{i=1}^{n} \sum_{l=1}^{f_i} k l^{k-1} (l^k - (l-1)^k) \\
&\leq k F_1 \sum_{i=1}^{n} f_i^{k-1} f_i^k \\
&\leq k F_1 F_{2k-1}
\end{aligned}$$

$F_1 F_{2k-1} = (\sum_{i=1}^n f_i)(\sum_{i=1}^n f_i^{2k-1}) \leq n^{1-\frac{1}{k}} F_k^2$
since $\text{Var}(Y) \leq k n^{1-\frac{1}{k}} F_k^2$
and $\mathbb{E}[Y] = F_k$.

Using median trick we can get an (ϵ, δ)-approximation in space $O(k \frac{1}{\epsilon^2} n^{1-\frac{1}{k}} \log \frac{1}{\delta})$. □

10.1.5 F_k / F_2 estimation

For $k = 2$, we get (ϵ, δ)-approx in $\tilde{O}(\frac{\sqrt{n}}{\epsilon^2} \log \frac{1}{\delta})$ space. Can we do better? For F_2, AMS showed that $polylog(n)$ suffices! For $k > 2$ the right bound is $\tilde{O}(n^{1-2/k})$. For $0 < k \leq 2$ we can get $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} polylog(n))$.

10.1.6 AMS F_2 Estimation

- Choose $h : [n] \to \{-1, 1\}$ from a 4-wise independent hash family H.
- $z \leftarrow 0$
- **while** (stream is not empty)
  e_m is current element
  $z \leftarrow z + h(e_m)$
- **end while**
- Output z^2.

A 4-wise independent family can be stored via $O(1) \log n$ bit numbers.

10.1.7 Analysis

Let $Y_i = h(i)$. $\mathbb{E}[Y_i] = 0$, $\mathbb{E}[Y_i^2] = 1$. $Y_i \in \{-1, 1\}$. $Y_1, \ldots, Y_n$ are 4-wise independent. $Z = \sum_{i=1}^n f_i Y_i$. Output is Z^2.

$$\begin{aligned}
\mathbb{E}[Z^2] &= \mathbb{E}\left[\left(\sum_{i=1}^n f_i Y_i\right)^2\right] = \mathbb{E}\left[\sum_i f_i^2 Y_i^2 + \sum_{i \neq j} f_i f_j Y_i Y_j\right] \\
&= \sum_{i=1}^n f_i^2 \mathbb{E}[Y_i^2] + \sum_{i \neq j} f_i f_j \mathbb{E}[Y_i Y_j] \\
&= \sum_{i=1}^n f_i^2 = F_2
\end{aligned}$$

$\text{Var}(Z^2) = \mathbb{E}[Z^4] - (\mathbb{E}[Z^2])^2 = \mathbb{E}[Z^4] - F_2^2$.

$$\mathbb{E}[Z^4] = \mathbb{E}\left[\sum_i \sum_j \sum_k \sum_l f_i f_j f_k f_l Y_i Y_j Y_k Y_l\right]$$

Any term with only one occurrence of a term Y_i becomes 0.

$$\mathbb{E}[Z^4] = \sum_{i=1}^n f_i^4 \mathbb{E}[Y_i^4] + 6 \sum_{i=1}^n \sum_{j=i+1}^n f_i^2 f_j^2 \mathbb{E}[Y_i^2 Y_j^2] = \sum_{i=1}^n f_i^4 + 6 \sum_{i<j} f_i^2 f_j^2$$

$$\begin{aligned}\mathrm{Var}(Z^2) &= \sum_{i=1}^{n} f_i^4 + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2 f_j^2 - \left(\sum_{i=1}^{n} f_i^2\right)^2 \\ &= \sum_{i=1}^{n} f_i^4 + 6\sum_{i<j} f_i^2 f_j^2 - \left(\sum_{i=1}^{n} f_i^4 + 2\sum_{i<j} f_i^2 f_j^2\right) \\ &= 4\sum_{i<j} f_i^2 f_j^2 \leq 2\left(\sum f_i^2\right)^2 = 2F_2^2\end{aligned}$$

An (ϵ, δ)-approximation requires $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ counters. We make an observation that non-negativity of f_i did not play a role in the proof. This will lead us to a generalization of the streaming model. Recall we had $e_t \in [n]$ for each time t. Now we have $e_t = (i_t, \Delta_t)$ where $i_t \in [n]$ and Δ_t is an update to coordinate i_t.

We use $x \in \mathbb{R}^n$ that starts at $\vec{0}$ and is updated as

$$x_{i_t} = x_{i_t} + \Delta_t$$

after e_t. Now F_2 becomes $||x||_2^2$. We see that the AMS-F_2 estimator works in this model.

10.1.8 AMS F_2 Estimation (Generalized)

- Choose $h : [n] \to \{-1, 1\}$ from a 4-wise independent hash family H.
- $z \leftarrow 0$
- **while** (stream is not empty)
  $e_t \leftarrow (i_t, \Delta_t)$
  $z \leftarrow z + \Delta_t h(i_t)$
- **end while**
- Output z^2.

Exercise: Show $\mathbb{E}[Z^2] = ||x||_2^2$ and $\mathrm{Var}(Z^2) \leq 2||x||_2^4$. $||x||_2^2$ is the length of the vector x.

Should remind you of dimensionality reduction! Interpreting AMS-F_2 estimator as a linear sketch. We can view the streaming computation as

$$\begin{pmatrix}+1 & -1 & +1 & \dots & -1\end{pmatrix}\begin{pmatrix}x_1\\x_2\\\vdots\\x_n\end{pmatrix} = z$$

where the row vector is obtained from h.

Recall that in dimensionality reduction we picked a $k \times n$ matrix A where we chose A_{ij} as an independent Gaussian. In F_2 estimation we are picking A where each row of A is obtained from a 4-wise independent hash function with entries in $\{-1, +1\}$. In dimensionality reduction we chose $k = \Theta(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ and showed that $||\frac{1}{\sqrt{k}}Ax||_2$ is an (ϵ, δ)-approximation to $||x||_2$. But in F_2 estimation we seem to be getting the same! $\Theta(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ rows suffice to get an ϵ-approximation with probability $(1-\delta)$. But we are only using 4-wise independence in each row. Why not use this for dimensionality reduction?

The difference is the following. Ax with $k = \Theta(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ has sufficient information to recover a $(1 \pm \epsilon)$-approximation for $||x||_2$ but $||Ax||_2$ is itself not the way we compute the approximation. We use a median estimator which is not a linear function. Nevertheless the information that the algorithm computes is a linear sketch, Ax.

A **sketch** of a data stream σ is some function $C(\sigma)$ that is a compact representation of σ. We want sketches to have **comparability**. Given σ_1 and σ_2 and sketches $C(\sigma_1)$ and $C(\sigma_2)$, we would like to compute $C(\sigma_1 \cdot \sigma_2)$ from $C(\sigma_1)$ and $C(\sigma_2)$. A particularly nice sketch is a **linear sketch**, where $C(\sigma_1 \cdot \sigma_2) = C(\sigma_1) + C(\sigma_2)$.

The F_2 estimation can be seen as a linear sketch.

$$Ax = \begin{pmatrix} \longleftarrow & h_1 & \longrightarrow \\ \longleftarrow & h_2 & \longrightarrow \\ & \vdots & \\ \longleftarrow & h_k & \longrightarrow \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Each row corresponds to a hash function. Note that the way we use the output of the sketch to compute some information about the data can be some non-linear function of the sketch itself. Linear sketches naturally allow for deletions.

Chapter 11

Heavy Hitters and Sketch Algorithms

11.1 Introduction to Heavy Hitters

F_0 is the most frequently occurring item in the stream. It is a brittle measure. In most applications we want to know the **heavy hitters**: items that occur very frequently.

11.1.1 Definition

From a theoretical perspective, we call an index $i \in [n]$ a heavy hitter if

$$F_i \geq \frac{m}{c}$$

for some sufficiently large constant c.

Alternatively, fix $F_i \geq \frac{m}{k}$ for some integer k.

11.2 Misra-Gries Algorithm

A classical algorithm shows that one can identify items i with $F_i \geq \frac{m}{k}$.

11.2.1 Algorithm Description

We have a data structure D that stores k items along with a counter for each. D is initialized to the empty set.

Implicitly, it defines an estimate $\hat{F}_i$ for each i:

- If $i \in D$ at the end, then $\hat{F}_i$ is the counter value
- Otherwise, it is 0

Theorem 11.1. *For all $i \in [n]$:*

$$f_i - \frac{m}{k+1} \leq \hat{f}_i \leq f_i.$$

In particular, $\hat{f}_i$ never overestimates f_i, and if i is a heavy hitter (i.e., $f_i \geq m/k$), then i will be in D. Space usage is $O(k)$.

Although Misra-Gries is nice, it does not allow deletion and also does not provide a sketch.

Algorithm 9 Misra-Gries-k

$D \leftarrow \emptyset$
while stream is not empty **do**
  e is current item
  if $e \in D$ **then**
    increment counter for e in D
  else
    if $|D| < k$ **then**
      add e to D with counter value 1
    else
      decrease counter value by 1 for all current elements
      delete from D any element with counter set to 0
Output: values stored in D and the counter values

11.3 Count-Min Sketch

Count-Min and Count sketches use hashing to identify heavy hitters and they have led to many applications.

11.3.1 Basic Idea

Suppose we use a hash function $h : [n] \to [ck]$ for some sufficiently large constant c. Then h spreads the n items into ck buckets. Suppose the heavy hitters are $i_1, \ldots, i_k$. We expect that they will not collide and we can use separate counts in each bucket.

We will use amplification as usual by considering multiple hash functions rather than a single one.

11.3.2 Algorithm (Cormode-Muthukrishnan)

Let $h_1, h_2, \ldots, h_d$ be d independent (pairwise independent) hash functions from $[n]$ to $[w]$.

Algorithm 10 Count-Min Sketch

Initialize $C[\ell][j] = 0$ for all $\ell \in [d], j \in [w]$
while stream is not empty **do**
  let (i, Δ) be current item
  for $\ell = 1$ to d **do**
    $C[\ell][h_\ell(i)] \leftarrow C[\ell][h_\ell(i)] + \Delta$
for $i = 1$ to n **do**
  $\hat{x}_i = \min_{\ell=1}^{d} C[\ell][h_\ell(i)]$

where w is the width of the sketch and d is the number of independent hash functions we use. We use $w = \frac{e}{\varepsilon}$ and $d = \log \frac{1}{\delta}$.

Lemma 11.2. *Consider the strict turnstile model ($\bar{x} \geq 0$). Let $d = \log \frac{1}{\delta}$ and $w = \frac{e}{\varepsilon}$. Then:*

1. $\hat{x}_i \geq x_i$
2. $\Pr[\hat{x}_i - x_i \geq \varepsilon \|\bar{x}\|_1] \leq \delta$

Proof. Fix i. For $\ell \in [d]$:

$$Z_\ell = C[\ell][h_\ell(i)] = x_i + \sum_{i' \neq i:\, h_\ell(i') = h_\ell(i)} x_{i'}$$

Since $\bar{x} \geq 0$, we have $Z_\ell \geq x_i$, which gives part (i). For the excess:

$$Z_\ell - x_i = \sum_{i' \neq i:\, h_\ell(i') = h_\ell(i)} x_{i'}$$

By pairwise independence, $\Pr[h_\ell(i') = h_\ell(i)] = \frac{1}{w}$, so:

$$\mathbb{E}[Z_\ell - x_i] = \frac{1}{w} \sum_{i' \neq i} x_{i'} \leq \frac{\|\bar{x}\|_1}{w} = \frac{\varepsilon}{e} \|\bar{x}\|_1$$

By Markov's inequality:

$$\Pr[Z_\ell - x_i \geq \varepsilon \|\bar{x}\|_1] \leq \frac{1}{e}$$

Since the d hash functions are independent:

$$\Pr[\hat{x}_i - x_i \geq \varepsilon \|\bar{x}\|_1] = \Pr[\min_\ell (Z_\ell - x_i) \geq \varepsilon \|\bar{x}\|_1] \leq \left(\frac{1}{e}\right)^d \leq \delta$$

Choosing $d = \lceil \log \frac{1}{\delta} \rceil$ gives part (ii). Setting $d = \Omega(\log n)$, the bound holds simultaneously for all $i \in [n]$ with high probability. □

Count-Min gives overestimates. Total space is $O(dw)$ counters $= O\left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \log n\right)$.
Advantages: Simple, handles dependencies.
Disadvantages: Only handles $\bar{x} \geq 0$.
Exercise: Show that Count-Min is a linear sketch.

11.4 Count Sketch

Count Sketch is similar to Count-Min in using d independent hash functions but uses F_2 estimation ideas and median estimator instead of min.

11.4.1 Algorithm (Charikar-Chen-Farach-Colton)

Let $h_1, h_2, \ldots, h_d$ be independent hash functions from $[n]$ to $[w]$.
Let $g_1, g_2, \ldots, g_d$ be independent functions from $[n]$ to $\{-1, +1\}$.
$\hat{x}_i$ can be negative even if $\bar{x} \geq 0$. Cancellation can happen like in F_2 estimation.

Lemma 11.3. *Let $d = O(\log \frac{1}{\delta})$ and $w = O\left(\frac{1}{\varepsilon^2}\right)$. Then for any $i \in [n]$:*

1. $\mathbb{E}[\hat{x}_i] = x_i$
2. $\Pr[|\hat{x}_i - x_i| \geq \varepsilon \|\bar{x}\|_2] \leq \delta$

Algorithm 11 Count Sketch

Initialize $C[\ell][j] = 0$ for all ℓ, j
while stream is not empty **do**
  let (i, Δ) be current item
  for $\ell = 1$ to d **do**
    $C[\ell][h_\ell(i)] \leftarrow C[\ell][h_\ell(i)] + g_\ell(i) \cdot \Delta$
for $i \in [n]$ **do**
  $\hat{x}_i = \text{median}_{\ell=1}^{d}\{g_\ell(i) \cdot C[\ell][h_\ell(i)]\}$

Proof. Fix i. For $\ell \in [d]$, to make analysis easier, let $Y_{i'} = \mathbb{1}_{h_\ell(i')=h_\ell(i)}$ be the indicator for $h_\ell(i') = h_\ell(i)$.

$$\begin{aligned} Z_\ell &= g_\ell(i) \cdot C[\ell][h_\ell(i)] \\ &= g_\ell(i) \cdot \left(\sum_{i':h_\ell(i')=h_\ell(i)} g_\ell(i') \cdot Y_{i'} \cdot x_{i'} \right) \\ &= \sum_{i'} g_\ell(i) \cdot g_\ell(i') \cdot Y_{i'} \cdot x_{i'} \end{aligned}$$

$\mathbb{E}[Z_\ell] = x_i$ by pairwise independence of g_ℓ.
We note that $\mathbb{E}[Y_{i'}] = \frac{1}{w}$ and $\mathbb{E}[g_\ell(i) \cdot g_\ell(i')] = 0$ for $i' \neq i$ by pairwise independence of h_ℓ.

$$\begin{aligned} \text{Var}(Z_\ell) &= \mathbb{E}[Z_\ell^2] - (\mathbb{E}[Z_\ell])^2 \\ &= \mathbb{E}\left[\sum_{i'} g_\ell(i) \cdot g_\ell(i') \cdot Y_{i'} \cdot x_{i'} \right]^2 - x_i^2 \\ &= \mathbb{E}\left[\sum_{i'} Y_{i'} \cdot x_{i'}^2 \right] - x_i^2 \\ &\leq \frac{1}{w} \|\bar{x}\|_2^2 \end{aligned}$$

Hence, using Chebyshev's inequality:

$$\Pr[|Z_\ell - x_i| \geq \varepsilon \|\bar{x}\|_2] \leq \frac{1}{w\varepsilon^2}$$

Via Chernoff bounds:

$$\Pr[\text{median}_\ell Z_\ell - x_i \geq \varepsilon \|\bar{x}\|_2] \leq \delta$$

□

11.5 Finding Heavy Hitters

Important: Sketches do not store directly the identity of the heavy hitters. Given $i \in [n]$, we can estimate $\hat{x}_i$ from the sketch. But outputting all i such that $\hat{x}_i$ is high requires a linear scan through $[n]$. Can maintain multiple data structures and use additional information to find the heavy hitters in $O(k)$ space and time.

11.6 Sparse Recovery

One nice and powerful application of Count Sketch is for sparse recovery. Suppose $\bar{x} \in \mathbb{R}^n$ is sparse or close to sparse, meaning that only k of the coordinates are non-zero. Can we recover $\bar{x}$ without knowing which of the coordinates are going to be important? Want to use only $O(k)$ space.

11.6.1 Definition

Given $\bar{x} \in \mathbb{R}^n$, let

$$\text{error}_k(\bar{x}) = \min_{\bar{z}:\|\bar{z}\|_0 \leq k} \|\bar{x} - \bar{z}\|_2$$

That is, what is the best k-sparse approximation to $\bar{x}$?

Offline, it is easy to compute:

$$\bar{z}_i^* = \begin{cases} x_i & \text{if } i \text{ is among the largest absolute value } k \text{ coordinates of } \bar{x} \\ 0 & \text{otherwise} \end{cases}$$

Can we find $\bar{z}^*$ in the streaming setting?

There exists a Count Sketch with $w = O\left(\frac{k}{\varepsilon^2}\right)$ and $d = O(\log n)$ that allows us to find a $\bar{z}$ such that $\|\bar{z}\|_0 \leq O(k)$ and with high probability

$$\|\bar{x} - \bar{z}\|_2 \leq C \cdot \text{error}_k(\bar{x})$$

In particular, if $\bar{x}$ is k-sparse, then we get exact recovery.

11.7 Compressed Sensing and RIP Matrices

Count Sketch guarantees that we can recover any sparse $\bar{x}$ with high probability. Can we guarantee probability 1 with a linear sketch? Yes!

There exist $\ell \times n$ matrices M for $\ell = O(k \log \frac{n}{k})$ such that given any k-sparse $\bar{x} \in \mathbb{R}^n$, one can recover $\bar{x}$ from $M\bar{x}$.

Note that $M\bar{x}$ takes $O(\ell)$ space, and since $\ell = O(k \log n)$, we are not storing much more than what we want to recover.

Such matrices are called **RIP matrices** (Restricted Isometry Property).

It turns out that a random $\ell \times n$ matrix with each entry chosen independently from a $\mathcal{N}(0, 1)$ Gaussian distribution satisfies the RIP. But we cannot easily verify that a given matrix is RIP.

This area is called **Compressed Sensing** and has several applications in signal processing.

Chapter 12

DNF Counting and Network Unreliability

12.1 Basic Estimation Framework

Suppose we have a finite universe U of N elements and $S \subseteq U$ of n elements. We know N and can sample uniformly at random from U. We want to estimate $|S|$.

We also assume that given $x \in U$, we can efficiently check if $x \in S$.

Basic Algorithm: We can estimate $|S|$ by taking a sample $X \in U$ and outputting N if $X \in S$ and 0 otherwise. It is easy to see that this is an unbiased estimator for $|S|$, but the variance is $N^2 \cdot \frac{|S|}{|U|}\left(1 - \frac{|S|}{|U|}\right)$.

Thus, using our standard tricks, we can obtain an (ϵ, δ)-approximation of $|S|$ using $O\left(\frac{|U|}{|S|} \cdot \frac{1}{\epsilon^2} \cdot \ln \frac{1}{\delta}\right)$ samples.

Therefore, if n is not too large and we can sample from U uniformly, we can estimate $|S|$. The goal is to see two nice applications of this simple idea and also introduce briefly the counting complexity class #P defined by Valiant in his influential work.

Note: We can apply the above estimator even in the continuous setting where we have a probability measure μ on U and we want to estimate $\mu(S)$.

12.2 #P and DNF Counting

A **DNF formula** over n boolean variables $X_1, X_2, \ldots, X_n$ is a formula

$$\phi = C_1 \vee C_2 \vee \cdots \vee C_m$$

which is a disjunction (OR) of several clauses, each of which is a conjunction of a set of literals.

Example:

$$(X_1 \wedge \bar{X}_3 \wedge X_5) \vee (X_2 \wedge \bar{X}_4 \wedge X_1 \wedge X_6)$$

Clearly any DNF formula is satisfiable. We want to count the number of satisfying assignments to ϕ. We will denote it by $|\phi|$. Exact counting is likely to be hard since it is Complete for the counting class #P.

However, we can get an (ϵ, δ) approximation in poly $\left(m, n, \frac{1}{\epsilon}, \log \frac{1}{\delta}\right)$ time.

12.2.1 Naive Approach

Let U be the set of 2^n possible Boolean assignments to the variables. Let S be those that satisfy ϕ.

We can sample from U and it is easy to check if $X \in S$. Hence we can apply the basic scheme. Unfortunately, $\frac{|S|}{|U|}$ can be exponentially small. Hence this naive scheme does not work.

12.2.2 Refined Idea

Let S_i be the set of all assignments that satisfy clause C_i.

We know $|S_i|$ and can also generate a uniform sample from S_i easily.

How? If $C_i = X_1 \wedge \bar{X}_2 \wedge X_5$, then an assignment to the n variables satisfies C_i iff $X_1 = 1$, $X_2 = 0$, $X_5 = 1$. Thus $|S_i| = 2^{n-3}$. We can sample from S_i uniformly as well.

We want to estimate $|S| = |S_1 \cup S_2 \cup \cdots \cup S_m|$. A satisfying truth assignment may appear in multiple sets.

To use our basic set up of U and S, we do the following:

i. Let $\beta \in S$ be a satisfying truth assignment. We let

$$U(\beta) = \{(\beta, i) : \beta \in S_i\}.$$

In other words, we create a copy of β for each S_i it belongs to.

ii. We map (β, i) to S if i is the smallest index such that $\beta \in S_i$.

Now $|U| = \sum_i |S_i|$ and $|S| = |\phi|$, and thus

$$\frac{|S|}{|U|} = \frac{|\phi|}{\sum_i |S_i|} \geq \frac{1}{m}.$$

Sampling from U: We pick $i \in [m]$ where probability of i appearing is $\frac{|S_i|}{|U|}$. Once i is picked, we pick a uniformly random β from S_i. It is easy to see that this process generates a uniform element from U.

Checking membership in S: Given (β, i), we check if i is the smallest index such that $\beta \in S_i$. If it is, then we output it as an element from S.

Thus we can estimate $|\phi|$ to within a $(1 \pm \epsilon)$ factor with high probability using $O\left(\frac{m \log n}{\epsilon^2}\right)$ samples. Each sample can be generated and checked in poly time.

12.2.3 A Natural Example

Given shapes, want to estimate area of their union. If shapes are well behaved, can do exact computation but expensive. Can obtain fast approximation generically. Only need to have information for each shape.

12.3 Unreliability of a Graph

Given undirected graph $G = (V, E)$ and for each edge $e \in E$ a value $p_e \in [0, 1]$, where p_e is the probability that e fails.

Suppose each edge fails independently with probability p_e.

Let α be the probability that G is disconnected. This is the **unreliability estimation problem**.

We can also be interested in estimating $1 - \alpha = \beta$. Note that approximating α is not same as approximating β.

We will discuss estimating α.

12.3.1 Two Regimes of Interest

(i) Easy Case: $\alpha \geq \frac{1}{n^c}$ for some constant c.

This is easy because we can run the Monte Carlo Simulation with $O\left(\frac{1}{\alpha} \cdot \frac{1}{\epsilon^2} \cdot \log n\right)$ experiments. We can estimate α to within $1 \pm \epsilon$ with high probability.

(ii) Difficult Case: When $\alpha < \frac{1}{n^c}$. This is when α is really small. Plain simulation will not work since we will rarely if ever see G being disconnected.

We will address the above case in the rest of the lecture.

12.3.2 Analysis for Small α

For simplicity, we will assume $p_e = p$ for all e. One can reduce the general case to this but requires some work.

Let K be the size of the min cut of G in terms of number of edges.

Let $C_1, C_2, \ldots, C_h$ be the cuts of G ordered in non-decreasing value. We treat each C_i as a set of edges.

Let A_i be the event that cut C_i fails. That is, all edges in C_i fail: $\prod_{e \in C_i} X_e$ where $X_e = 1$ with prob p. Then $\Pr[A_i] = p^{|C_i|}$.

G is disconnected iff $\bigcup_i A_i$.

If h was small (polynomially bounded) we could use previous ideas, but there are an exponential number of cuts.

Since $p_e = p$ for all e and $K \leq |C_i|$, we have $\Pr[A_i] \leq \Pr[A_j]$ if $i \leq j$.

12.3.3 Main Observation

We are in the setting that p is quite small (failure prob).

Also $\alpha = \Pr\left[\bigcup_i A_i\right] \geq \Pr[A_1] = p^K$.

Consider any cut C_j such that $|C_j| \geq 4K$. Then

$$\Pr[A_j] = p^{|C_j|} \leq p^{4K} = (p^K)^4.$$

So large cuts are very unlikely to fail. But there are a lot of large cuts.

However, as we saw earlier, the number of β-approximate cuts is only growing as $n^{2\beta}$.

Thus we can use union bound over all large cuts and still the total probability of them failing will be tiny compared to $\Pr[A_1]$. So we can ignore all cuts j such that $|C_j| \geq 4K$ if we choose c appropriately.

This implies we can focus on cuts j such that $K \leq |C_j| \leq 4K$, but there are only $O(n^8)$ of those cuts and we can run the basic estimation algorithm on these polynomially many cuts. Karger's algorithm can be used to enumerate all these cuts efficiently with high probability.

12.3.4 Formal Analysis

Lemma 12.1. $\displaystyle\sum_{j:|C_j| \geq cK} \Pr[A_j] \leq \frac{\alpha}{100}$

Proof. Consider all cuts j such that $K \leq |C_j| \leq cK$ for $c \geq 2$. Let l_i be the number of such cuts with $|C_j| = i$.

By Karger's theorem, $l_i \leq n^{2i/K}$.

Also $\Pr[A_j] = p^i$ for such C_j.

$$\begin{aligned}
\sum_{j:|C_j|\geq cK} \Pr[A_j] &\leq \sum_{i=cK}^{|E|} l_i \cdot p^i \\
&\leq \sum_{i=cK}^{|E|} n^{2i/K} \cdot p^i \\
&= \sum_{i=cK}^{|E|} \left(n^{2/K} \cdot p\right)^i \\
&\leq \frac{\left(n^{2/K} \cdot p\right)^{cK}}{1 - n^{2/K} \cdot p} \\
&\leq \frac{\alpha}{100}
\end{aligned}$$

for sufficiently large c. □

Thus we can ignore all large cuts and do the estimation with the polynomial number of small cuts.

Note: The above analysis is quite loose. See cited notes for better calculations.

Chapter 13

Markov Chains and Random Walks

13.1 Introduction

A **stochastic process** is a time-evolving sequence of random variables: $X_0, X_1, X_2, \ldots, X_t, \ldots$ where X_t is the state of the system at time t and X_0 is the initial state (which can itself be a random one). One can view it also as an evolving randomized algorithm. A Markovian process or a Markov chain is a particular type of stochastic process where X_t depends only on X_{t-1}. To formalize this, we assume that $X_0, X_1, \ldots$ are random variables over some state space Ω.

Definition 13.1. A stochastic process is **Markovian** if:

$$\Pr[X_t = u_t \mid X_{t-1} = u_{i-1}, X_{t-2} = u_{t-2}, \ldots, X_0 = u_0] = \Pr[X_t = u_t \mid X_{t-1} = u_{t-1}]$$

We will be mostly concerned with finite state Markov chains, although countable state chains are also quite relevant. A finite state Markov chain is easy to visualize as a directed graph $G = (V, E)$ with non-negative edge-weights. Note that we allow self-loops.

- V represents the states. We usually use n for $|V|$ and assume the states are numbered 1 to n.
- For state (vertex) i, the weighted outgoing edges from i represent the probability with which state j is reached from state i in one step. Thus the weight of the edge $p(i, j) \in [0, 1]$. If no edge (i, j) is present, then the probability is implicitly 0.
- Self-loops are allowed since we want to allow the process to remain in the same state.

We associate an $n \times n$ probability transition matrix P with an n-state Markov chain:

$$P_{ij} = \Pr[X_t = j \mid X_{t-1} = i]$$

Clearly we need:

$$\sum_{j \in [n]} P_{ij} = 1 \quad \forall i \in [n]$$

and $P_{ij} \geq 0$.

The advantage of the graphical representation is that it allows us to understand the properties of the chain via graph-theoretical aspects.

Evolution of the Process: Suppose $\pi(0)$ is an initial probability distribution over the state space V. How does the process evolve? This is the central question in Markov chains. We represent probability distributions over the states by n-dimensional *row* vectors. We let π_i denote the probability of being in state i. Suppose we start with $\pi(0)$ as the starting distribution (can be deterministic in that we have $\pi(0)_i = 1$ for some fixed vertex i). Then it is not difficult to see that after one step the distribution is:

$$\pi(1) = \pi(0)P$$

since $\Pr[X_1 = j] = \sum_i \Pr[X_0 = i]P_{ij}$ due to the Markovian property. After t steps, we see by induction that:

$$\pi(t) = \pi(0)P^t$$

We want to understand properties of the chain as it evolves and in particular the long-term behaviour of the chain. A central result in the theory is that the process converges to a stable stationary distribution under reasonable conditions. These conditions are natural and relatively easy to understand from a graph-theoretic viewpoint.

Transient states and irreducibility: Suppose G is not strongly connected. Then it is not hard to see that the process will get stuck in a sink component of the underlying meta-graph (i.e., the strongly connected component graph). There could be multiple such sink components and the process will reach one of them and will not leave. Thus any state that is not in one of those sink components is going to be *transient*. Thus, for understanding long-term behaviour it suffices to focus on strongly connected chains (sometimes we simply say connected). This motivates the following:

Definition 13.2. A Markov chain is **irreducible** if the underlying graph is strongly connected.

If a chain is irreducible then any state i can reach any other state j with some probability $\epsilon > 0$ since there is an i to j path with non-zero probabilities on each of the edges. We can then prove the following.

Lemma 13.3. *Let h_{ij} be the expected time to hit state j for the first time starting in state i; if $j = i$ we think of the first time when i is revisited. Then $h_{ij} < \infty$.*

Periodicity: A second issue that comes up is periodicity or oscillatory behaviour.

Definition 13.4 (Period)**.** For a finite state Markov chain defined by a graph $G = (V, E)$ and a state $i \in V$, period(i) is the largest non-negative integer d such that d divides the length of any closed walk containing i:

$$\text{period}(i) = \gcd\{|W| : W \text{ is a closed walk containing } i\}$$

Lemma 13.5. *Suppose G is strongly connected. Then period(i) = period(j) for all $i, j \in V$.*

Proof. Exercise. □

The preceding lemma implies that there is a period d for a strongly connected graph.

Lemma 13.6. *Suppose period is $d \geq 1$. Then V can be partitioned into $V_0, V_1, \ldots, V_{d-1}$ such that $(u, v) \in E$ implies $u \in V_i$ and $v \in V_{(i+1) \bmod d}$.*

Definition 13.7. An irreducible finite state Markov chain is **aperiodic** if period $d = 1$.

Definition 13.8. An irreducible and aperiodic Markov chain is called **ergodic**.

Lazy Random Walk Suppose we have a Markov chain which is irreducible but periodic. We can make it aperiodic by adding a self-loop to each i and making i stay in state i with probability $p > 0$ (say $\frac{1}{2}$) and take the original transition with probability $(1-p)$. In other words, we are changing the transition matrix from P to $(1-p)P + pI$ where I is the identity matrix. The new walk is called a **lazy version** of the original walk and retains the essential properties.

An important and fundamental notion is the following.

Definition 13.9. A distribution π is a **stationary distribution** of a Markov chain with transition matrix P if $\pi P = \pi$.

A stationary distribution is a stable distribution.

13.2 Fundamental Theorem of Markov Chains

Theorem 13.10. *Suppose P corresponds to a finite state irreducible Markov chain. Then there exists a unique stationary distribution π for it.*

1. *For any i, $\pi_i = \frac{1}{h_{ii}}$ where h_{ii} is the expected time for the chain to revisit i if started in i.*
2. *Let $N_i(t)$ be the number of times the chain visits i in t steps. Then $\lim_{t\to\infty} \frac{N_i(t)}{t} = \pi_i$.*

Moreover if P is also aperiodic, and hence the chain is ergodic, for any starting distribution $\pi(0)$, $\lim_{t\to\infty} \pi(0)P^t = \pi$.

13.2.1 Proof via Perron-Frobenius Theorem

Connection to eigenvalues: $\pi P = \pi$ implies π is a left eigenvector of P with eigenvalue 1. Thus, to prove existence of π we can try to do it via linear algebra. We are typically used to right eigenvectors: If A is an $n \times n$ matrix, $Ax = \lambda x$ has a non-zero solution x iff λ is an eigenvalue and x is a corresponding eigenvector. $\det(A - \lambda I)$ is the characteristic polynomial; its roots are eigenvalues. In general, eigenvalues need not be real. There are two well-known situations when A has real eigenvalues.

- If A is symmetric, then all eigenvalues are real.
- If A is a symmetric positive semi-definite matrix then all eigenvalues are ≥ 0.

But P is not symmetric in the general case. However P is non-negative. Note that $P\mathbf{1} = \mathbf{1}$ since P is a stochastic matrix and hence P has an eigenvalue of 1 and a right eigenvector which is the all ones vector but we are looking for a left eigen vector. Note that for a matrix A the eigenvalues of A and its transpose A^T are the same (due to the characteristic polynomial characterization) and the right eigenvectors of A are the left eigenvectors of A^T.

Theorem 13.11 (Perron)**.** *Let A be a non-negative matrix (i.e., $A_{ij} \geq 0$ for all i, j). Then:*

1. *A has a real positive eigenvalue $\lambda_0 > 0$ and corresponding positive eigenvector $v > 0$ such that any other eigenvalue λ (can be complex) satisfies $|\lambda| < \lambda_0$; hence λ_0 is the unique largest eigenvalue.*
2. *v is the unique non-negative vector (up to scaling) such that $Av = \lambda_0 v$.*
3. *Every other eigenvector has at least one non-positive coordinate.*

Perron's theorem requires $A > 0$ (strictly positive) while P for a Markov chain satisfies $P \geq 0$. Although one can derive properties for non-negative matrices via Perron's theorem from limits of positive matrices, there are subtelties for general non-negative matrices. Frobenius generalized Perron's theorem to a class of matrices including ones relevant to us.

Definition 13.12. A is an $n \times n$ matrix with $A \geq 0$ (i.e., $A_{ij} \geq 0$ for all i, j). We say A is **irreducible** if the corresponding weighted directed graph is strongly connected.

Theorem 13.13 (Perron-Frobenius)**.** *Let $A \geq 0$ and irreducible. Then A has a positive eigenvalue $\lambda_0 > 0$ and all other eigenvalues λ satisfy $|\lambda| \leq \lambda_0$. There is a positive eigenvector $v > 0$ such that $Av = \lambda_0 v$, and the following hold for λ_0 and v:*

1. *v is the unique eigenvector associated with λ_0. That is, if $Ax = \lambda_0 x$, then $x = \alpha v$ for some scalar $\alpha \geq 0$.*

2. *v is the only eigenvector with strictly positive coordinates.*

Corollary 13.14. *The largest real eigenvalue of an irreducible matrix $A \geq 0$ has a positive left eigenvector π. π is unique up to scaling and is the only non-zero vector that satisfies $\pi A = \lambda_0 \pi$.*

Proof. Consider A^T. $A^T \geq 0$ and irreducible. A^T has same eigenvalues as A. π is the right eigenvector of A^T. □

Now we can apply the preceding to Markov chains. Consider stochastic matrix P from an irreducible Markov chain: We saw that $\lambda = 1$ is a eigenvalue since $P\mathbf{1} = \mathbf{1}$. Since P is stochastic we can see that $Ax \leq x$ and hence 1 is the largest positive real eigenvalue. Thus the left eigenvector $\pi > 0$ corresponding to the eigenvalue 1 is unique by the preceding theorem/corollary. If $\pi > 0$ we can noramalize it to be a probability distribution. Uniqueness of π follows from the fact that the eigenvector for $\lambda = 1$ is unique up to scaling.

Periodic and aperiodic chains: We established existence of a stationary distribution via irreducability. When we have aperiodicity we can prove a stronger property. For this we note that if P is aperiodic and connected then for some sufficiently large t, $P^t > 0$. This is because if the gcd of the walk lengths in G is 1 then there is some integer K such that for all $t \geq K$, there is a walk of length t from i to j for any i, j. This implies that $P^t > 0$. Such matrices are called primitive. When P^t is strictly positive we can use Perron's theorem which is stronger and guarantees that all eigen values are strictly smaller than λ_0 (they could be complex). One can then use this gap to show that $\pi(0)P^t$ converges to π as $t \to \infty$.

Perron's theorem is classical and there are many sources. See Kents notes for one. It is not that long.

13.2.2 A second proof

This is from the book by Blum, Hopcroft, Kannan.

Let π_t be the distribution after t steps starting with $\pi(0)$. We have $\pi(t) = \pi(0)P^t$

Define $a(t) = \frac{1}{t}(\pi(0) + \pi(1) + \cdots + \pi(t-1))$ be the time-averaged distribution. Note that this is also a probability distribution.

Theorem 13.15. *Let G be an irreducible Markov chain. There is a unique probability distribution π such that $\pi P = \pi$. Moreover, for any starting distribution, $\lim_{t\to\infty} a(t)$ exists and equals π.*

Lemma 13.16. *Let P be the transition matrix of a connected Markov chain. The matrix $A = [P - I \mid \mathbf{1}]$ obtained by augmenting $P - I$ by an all-ones column has rank n.*

Proof. Note that A is a $n \times (n+1)$ matrix. Suppose $\text{rank}(A) < n$. Then let S be the null space of A, i.e., $S = \{y \in R^{n+1} : Ay = 0\}$. Then $\dim(S) \geq 2$. Each row of P sums to 1, so each row of $P - I$ sums to 0. Thus $(\mathbf{1}, 0) \in S$. If $\dim(S) \geq 2$, there exists a vector $(x, \alpha) \in S$ orthogonal to $(\mathbf{1}, 0)$.

Orthogonality implies $\sum_i x_i = 0$. Since $A \cdot (x, \alpha) = 0$, we have $(P - I)x = \alpha\mathbf{1}$. This means $x_i = \sum_j P_{ij}x_j + \alpha$ for each $i \in [n]$.

Let x_k have the max value among $x_1, x_2, \ldots, x_n$. Some $\ell \neq k$ has $x_\ell < x_k$ since $\sum_i x_i = 0$ and $x \neq 0$. By connectedness of G, there exists some ℓ such that (k, ℓ) is an edge and $x_\ell < x_k$. But we have $x_k = \sum_j P_{kj}x_j + \alpha$ which implies that $\alpha > 0$. Similarly, by considering the min value among $x_1, x_2, \ldots, x_n$ we can derive that $\alpha < 0$. This is a contradiction. □

Consider $a(t)P - a(t)$.

$$\begin{aligned} a(t)P - a(t) &= \frac{1}{t}(\pi(0)P + \pi(1)P + \ldots + \pi(t-1)P) - a(t) \\ &= \frac{1}{t}(\pi(1) + \pi(2) + \ldots + \pi(t)) - \frac{1}{t}(\pi(0) + \ldots + \pi(t-1)) \\ &= \frac{1}{t}(\pi(t) - \pi(0)) \end{aligned}$$

Define $b(t) = a(t)(P - I) = \frac{1}{t}(\pi(t) - \pi(0))$. Then $\|b(t)\|_\infty \leq \frac{2}{t} \to 0$ as $t \to \infty$.

By the preceding lemma, $A = [P - I \mid \mathbf{1}]$ has rank n. Since $a(t)(P - I) = b(t)$, we have $a(t)[P - I \mid \mathbf{1}] = [b(t) \mid 1]$. Consider the $n \times n$ sub-matrix B of A obtained by ignoring the first column of A; B has rank n. Let $c(t)$ be obtained from $b(t)$ by removing the first entry. Then $a(t)B = [c(t, 1]$. Since B is invertible, we have $a(t) = [c(t), 1]B^{-1}$. Since $b(t) \to 0$, we have $c(t) \to 0$ and hence $a(t) \to [\mathbf{0}, 1]B^{-}1$.

Thus, $\lim_{t\to\infty} a(t) = \pi$ where $\pi = [\mathbf{0}, 1]B^{-}1$. π is unique because we showed that starting with any distribution π_0, $a(t)$ converges to π. If there existed another stationary distribution π', then starting at π' we would have $a(t) = \pi'$ for all t.

A useful lemma is the following.

Lemma 13.17. *Suppose P corresponds to an irreducible chain. If we have a distribution π such that $\pi_i P_{ij} = \pi_j P_{ji}$ for all i, j, then $\pi P = \pi$.*

Proof. Exercise. □

13.2.3 Another Proof

This is from the Levin-Peres book.

For $i \in V$, define:

$$\tau_i = \min\{t \geq 0 : X_t = i\}$$

as the first hitting time for i, and

$$\tau_i^+ = \min\{t \geq 1 : X_t = i\}$$

which is the first hitting time not counting the initial state.

When $X_0 = i$, we call τ_i^+ the first return time. The expected value $h_{ij} = E_i[\tau_j^+]$ is the expected time to reach j starting at $X_0 = i$. In the rest of this section we will be interested in various quantities conditioned on $X_0 = i$. We use E_i and $\Pr_i$ to denote these quantities.

Lemma 13.18. *For any states i, j of an irreducible chain, $h_{ij} = E_i[\tau_j^+] < \infty$.*

Proof. Since the chain is strongly connected and $P_{ij} > 0$ for $(i,j) \in E$, there exists $\epsilon > 0$ and an integer r such that for any two states u, v, $\Pr_u[X_\ell = v] \geq \epsilon$. This is because we can take a path of length at most n from u to v and multiply the probabilities along that path to see that it is some non-zero value.

Thus, for any value of X_t, the probability of hitting state j between t and $t+r$ is at least ϵ.

Hence, for $k \geq 0$, we have:

$$\Pr_i[\tau_j > kr] \leq (1-\epsilon) \Pr_i[\tau_j > (k-1)r]$$

Therefore:

$$\Pr_i[\tau_j > kr] \leq (1-\epsilon)^k$$

Now, if Z is a non-negative random variable:

$$\mathbb{E}[Z] = \sum_{t=0}^{\infty} \Pr[Z > t]$$

We have $\Pr[\tau_j > t]$ is a decreasing function of t. Hence:

$$\begin{aligned} E_i[\tau_j] &= \sum_{t=0}^{\infty} \Pr_i[\tau_j > t] \\ &\leq \sum_{k=0}^{\infty} r \Pr_i[\tau_j > kr] \\ &\leq r \sum_{k=0}^{\infty} (1-\epsilon)^k < \infty \end{aligned}$$

□

Existence of a stationary distribution: This proof has the nice feature that we can construct the stationary distribution somewhat explicitly, which also implies that $\pi_i = \frac{1}{h_{ii}}$ for each i which is intuitive.

Let k be an *arbitrary state* of the irreducible chain. For any $i \in V$, define:

$$\pi'_i = E_k[\text{number of visits to } i \text{ before returning to } k] = \sum_{t=0}^{\infty} \Pr_k[X_t = i, \tau_k^+ > t]$$

Note that $\pi'_k = 1$ by the above definition.

Lemma 13.19. *Let π' be defined as above. Then π' satisfies $\pi' P = \pi'$ and π'/h_{kk} is a stationary distribution.*

In particular it shows that $\pi'_k/h_{kk} = 1/h_{kk}$. Note that the lemma applies for every k. It does not directly prove that we get the same stationary distribution if we use different states k but if you knew that the stationary distribution is unique then you would be able to conclude that $\pi_i = 1/h_{ii}$ for all i.

Now we prove the lemma. For any i we have $\pi'_i \leq h_{kk} < \infty$ (which we have established previously). We will check that π' is stationary. Fix state j. From definition of π'_i,

We have

$$\sum_i \pi'_i P(i,j) = \sum_i \sum_{t=0}^{\infty} \Pr_k[X_t = i, \tau_k^+ > t] P(i,j)$$

Since the event $\{\tau_k^+ \geq t+1\} = \{\tau_k^+ > t\}$ is determined by $X_0, \ldots, X_t$,

$$\Pr_k[X_t = i, X_{t+1} = j, \tau_k^+ \geq t+1] = \Pr_k[X_t = i, \tau_k^+ \geq t+1] P(i,j)$$

Reversing the order of summation in the first equality and using the preceding identity we get

$$\sum_i \pi'_i P(i,j) = \sum_{t=0}^{\infty} \Pr_k[X_{t+1} = j, \tau_k^+ \geq t+1] = \sum_{t=1}^{\infty} \Pr_k[X_t = j, \tau_k^+ \geq t]$$

The expression $\sum_{t=1}^{\infty} \Pr_k[X_t = j, \tau_k^+ \geq t]$ is very similar to the definition of π'_j and our goal is to show that it is indeed the same which would verify the stationarity of π'.

$$\begin{aligned}
\sum_{t=1}^{\infty} \Pr_k[X_t = j, \tau_k^+ \geq t] &= \pi'_j - \Pr_k[X_0 = j, \tau_k^+ \geq 0] + \sum_{t=1}^{\infty} \Pr_k[X_t = j, \tau_k^+ = t] \\
&= \pi'_j - \Pr_k[X_0 = j] + \Pr_k[X_{\tau_k^+} = j] \\
&= \pi'_j
\end{aligned}$$

We want to justify the last inequality by considering two cases.

- $j = k$. Since $X_0 = k$ and $X_{\tau_k^+} = k$, the terms $\Pr_k[X_0 = j]$ and $\Pr_k[X_{\tau_k^+} = j]$ are 1 and cancel out.
- $j \neq k$. Both terms are 0.

Finally, to get a probability measure we normalize by $\sum_i \pi'_i = E_k[\tau_k^+] = h_{kk}$. Thus $\pi = \pi'/h_{kk}$ is a stationary distribution.

13.3 Application to PageRank

The early approach of Google to rank web pages was based on using the link information that was in the pages. This was done to avoid the deficiencies of previous approaches that were based on manually classifying (Yahoo and AltaVista and others) and deficiencies of keyword search due to spam and other reasons.

The web graph is a directed graph where each web page corresponds to a node in a graph (this is a somewhat crude approximation) and a link in a page to another page creates a natural arc. Links encode information that gives information on how important pages are. The goal is to create a ranking of webpages globally; use ranking + query words later for personalized ranking.

How should one rank webpages in terms of importance? A simple approach is assign a score based on how many other links point to a webpage. $\text{score}(u) = \sum_{v:(v,u)\in E} 1$ This is easy to spam. A better approach is to score based on the importance of incoming pages:

$$\text{score}(u) = \sum_{v:(v,u)\in E} \frac{\text{score}(v)}{\text{out-deg}(v)}$$

This is a recursive definition. Main question: Does score(v) exist? Can normalize scores (since scaling does not violate the equation). Assume $\sum_{u \in V} x_u = 1$, so x is a probability distribution. This looks like the stationary distribution of a random walk on the web graph! But the web graph may not be ergodic.

The Brin-Page trick is to create an ergodic chain by considering a new graph H which is a convex combination of the webgraph and a complete bipartite graph. Thus, if P is the matrix corresponding to G and Q is the matrix corresponding to the complete directed graph on V we let

$$P' = (1 - \epsilon)P + \epsilon Q$$

This corresponding to a random walk where with probability ϵ, jump to a random webpage; with probability $1 - \epsilon$, follow a random outgoing link of the web graph. P' corresponds to an ergodic chain, so there exists a unique stationary distribution π for any fixed $\epsilon > 0$. This π is the ranking.

Computing PageRank: How to compute π? Use the power method: $\pi(0)P^t \to \pi$ for any $\pi(0)$. Start with $\pi(0) = \frac{1}{n}\mathbf{1}$. Graph is sparse (average out-degree is 8-10), hence computation is not onerous. Mostly matrix-vector multiplication and numerical linear algebra and the process converges after a few iterations to a reasonable vector - note that the goal is not to actually compute a stationary distribution but only to find a ranking.

Chapter 14

Random Walks on Undirected Graphs

14.1 Random Walks in Undirected Graphs

A finite state Markov chain corresponds to a random walk in a weighted directed graph. Random walks in undirected graphs have many nice properties and a number of applications. They are also closely related to reversible Markov chains.

Suppose $G = (V, E)$ is an undirected graph. We let $\vec{G} = (V, \vec{E})$ be the corresponding bidirected graph.

We can consider weights on the edges, but for simplicity we assume all are 1 (we allow multi-graphs).

14.1.1 Definition of Random Walk

A random walk on G is the following stochastic process: Start at some random vertex given by a probability distribution π_0 on V. In each step, if we are at vertex v, pick a uniform random edge in $\delta(v)$ and go to the endpoint of v.

Note that if the edge is a self-loop, we stay at v.

We can think of this random walk as a Markov chain on V where each edge (v, u) is given probability $\frac{1}{d(v)}$.

Lemma 14.1. *Suppose G is a loopless connected graph. Then G is aperiodic iff G is not bipartite.*

Proof. If G is bipartite, the underlying chain has period 2, since all cycles and closed walks have even length.

If G is not bipartite, G has an odd length cycle. In $\vec{G}$ we have that each vertex is in a closed walk of even length and one with odd length. By gcd, the period is 1. □

We can either assume G is not bipartite or add self-loops on each vertex and make the walk lazy. This will ensure the walk is aperiodic (ergodic).

Lemma 14.2. *A random walk on G converges to a stationary distribution π where $\pi(v) = \frac{d(v)}{2m}$.*

Proof. Exercise: Verify that this satisfies $\pi P = \pi$ for the underlying Markov chain. □

14.1.2 Hitting Times and Commute Times

Let $h_{u,v}$ be the expected time to reach state v when starting at u.

Hitting time is not necessarily symmetric.

Example: Lollipop graph L_n: $h_{u,v} = \Theta(n^3)$ and $h_{v,u} = \Theta(n^2)$.

Also, L_n shows that adding edges can increase $h_{u,v}$ and $C_{u,v}$.

Commute time is $C_{u,v} = h_{u,v} + h_{v,u}$, which is symmetric.

14.1.3 Basic Results

We will prove two basic results using elementary methods.

Lemma 14.3. *For any edge $uv \in E$: $h_{u,v} + h_{v,u} \leq 2m$.*

Proof. Consider: We can view the random walk on G as a random walk on $\vec{E}$. That is, the state space is $\vec{E}$. Consider this claim.

Consider the transition matrix Q for this chain. It turns out to be doubly stochastic.

For a normal transition matrix, row sum is 1, but here column sum is also 1. Easy to verify: $(1, 1, \ldots, 1)^T$ is a left eigenvector of Q. By normalizing, the stationary distribution of Q is $\frac{1}{2m}$, the uniform distribution.

$h_{u,v} + h_{v,u} \leq 2m$ where $h_{(u,v),(u,v)}$ is the expected time in the edge walk chain to start on edge (u, v) and revisit (u, v). We can interpret such a walk as giving an upper bound on $h_{u,v} + h_{v,u}$.

Claim: $h_{u,v} + h_{v,u} \leq 2m$. If the original random walk traversed the edge (u, v), then the expected time to traverse (u, v) again is $2m$.

Note: Since the original walk is memoryless, once it reaches v, it shows that the expected time to visit u and take edge (u, v) is at most $2m$.

But this walk is only one way to start at v and reach u and back to v: $h_{v,u} + h_{u,v} \leq 2m$. □

Caveat: Note the above holds only for $u, v \in E$. We will later see a more refined version when (u, v) is not necessarily an edge.

14.1.4 Cover Time

Definition 14.4. The cover time of a graph $G = (V, E)$ is the max over all $v \in V$ of the expected time to visit all the vertices. $C(v)$ is cover time starting at v. $C(G) = \max_v C(v)$.

Theorem 14.5. $C(G) \leq 2m(n - 1)$.

Proof. Consider a spanning tree T of G.

We can consider an Eulerian walk on T. Say it is $v_1, v_2, v_3, \ldots, v_{2n-2}, v_1$.

We can upper bound $C(v_1)$ by:

$$h_{v_1,v_2} + h_{v_2,v_3} + \cdots + h_{v_{2n-2},v_1} = \sum_{uv \in E(T)} (h_{u,v} + h_{v,u}) \leq 2m(n - 1).$$

□

One can prove another interesting upper bound on cover time:

Theorem 14.6. *$C(G) \leq H(n - 1) \cdot \max_{u,v \in V} h_{u,v}$, where $H(n - 1) = \sum_{i=1}^{n-1} \frac{1}{i}$ is the $(n - 1)$-th harmonic number.*

14.2 Applications

14.2.1 s-t Connectivity in $O(\log n)$ Space

Suppose we are given an undirected graph written on read-only memory in adjacency list/matrix format. We want to use very little extra memory to decide if some given s can reach t. We can easily do this using $O(n)$ space by using graph search (BFS/DFS).

Can we do this with $O(\log n)$ space? Note that writing s or t takes $O(\log n)$ bits.

Yes, if we allow randomization!

How: Start a random walk at s. Because $C(G) = O(mn)$, if we don't see t after $O(mn \log n)$ steps, we know w.h.p. that s is not connected to t.

Can implement random walk in $O(\log n)$ space.

G can be bipartite, so need to use lazy random walk. Doesn't change details too much.

14.2.2 2-SAT

2-SAT: Given a Boolean formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ where each clause has exactly 2 variables. Can check if ϕ is satisfiable, e.g., $\phi = (x_5 \vee x_3) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee x_7)$.

2-SAT is solvable in P. How? One nice way to see it is via random walks.

Algorithm:

1. Let $a = (a_1, a_2, \ldots, a_n) \in \{0,1\}^n$ be an arbitrary assignment to x_i.
2. While a does not satisfy ϕ do:
   - Let C_i be an arbitrary clause that is not satisfied by a.
   - Pick a literal of C_i uniformly at random.
   - Flip the assignment for the chosen literal and update a.

Lemma 14.7. *If ϕ is satisfiable, the algorithm terminates in $O(n^2)$ steps.*

Proof. Suppose s is a fixed satisfying assignment. Let a^t be the assignment after t steps. Let $d_t = \text{dist}(s, a^t)$ be the Hamming distance between s and a^t. That is, the number of variables in which a^t differs from s.

If $d_t = 0$ then algorithm terminates.

The algorithm can be viewed as doing a random walk on state space $\{0, 1, 2, \ldots, n\}$ and starting at position $\text{dist}(s, a^0)$.

Since only one variable is changed, distance changes by $+1$ or -1.

Since C_i is picked as an unsatisfied clause, at least one literal is incorrect, and hence with probability at least $\frac{1}{2}$ we will reduce distance.

Thus we can view this as a walk on $\{0, 1, 2, \ldots, n\}$. In the worst case, it starts at n on each side. Can view it as a random walk on a finite line.

Cover time of line is $O(n^2)$; will visit 0 in $O(n^2)$ in expectation. □

14.3 Electrical Networks and Random Walks

Ohm's law: $V = IR$ (voltage = current × resistance).

For resistors in series: effective resistance is $R = R_1 + R_2$.

For resistors in parallel: effective resistance is $R = \frac{R_1 R_2}{R_1 + R_2}$.

Let $R_{u,v}$ be effective resistance between u and v.

Theorem 14.8. $C_{u,v} = h_{u,v} + h_{v,u} = 2m \cdot R_{u,v}$.

Corollary 14.9. *If* $uv \in E$, *then* $C_{u,v} \leq 2m$.

Chapter 15

Electrical Networks and Random Walks

15.1 Ohm's Law

Ohm's Law: $V = IR$

where V is voltage, I is current, and R is resistance.

For effective resistance:

$$R_{\text{series}}: \quad R = R_1 + R_2$$
$$R_{\text{parallel}}: \quad R = \frac{R_1 R_2}{R_1 + R_2}$$

Let R_{uv} be the effective resistance between vertices u and v, defined as the resistance between u and v when a voltage difference of 1 is applied, with current i_{uv} flowing from u to v.

Corollary 15.1. *If $(u, v) \in E$, then $C_{uv} \leq 2m$.*

Proof. Because $R_{uv} \leq 1$ when $(u, v) \in E$ (the direct edge has resistance 1, and parallel paths can only reduce effective resistance). □

15.2 Example: Lollipop Graph

For a lollipop graph, we can compute the effective resistance using series and parallel resistance formulas.

15.3 Electrical Flow

Let $G = (V, E)$ be an undirected graph. We work with electrical flow, as opposed to standard flows in directed graphs.

15.3.1 Standard Flow (Directed Graphs)

Basic Setup: Suppose $d : V \to \mathbb{R}$ is a demand vector, where $d(v)$ represents the demand at v.

- $d(v) > 0$: flow coming into v

- $d(v) < 0$: flow leaving v

For a directed graph $G = (V, E)$, flow conservation requires:

$$\sum_{e \text{ out of } u} f_e - \sum_{e \text{ into } u} f_e = d(u)$$

This requires $\sum_{u \in V} d(u) = 0$.

A special case is when $d(v) = 0$ except at source s and sink t.

Capacity Constraints: Typically we have capacity constraints C_e for $e \in E$ and want $0 \leq f_e \leq C_e$.

Note: If there exists a feasible flow, then there exists an acyclic feasible flow.

Feasibility as LP: Feasibility of flow can be written as a linear program:

$$\sum_{e \text{ out of } u} f_e - \sum_{e \text{ into } u} f_e = d(u) \quad \forall u \in V$$
$$0 \leq f_e \leq C_e \quad \forall e \in E$$

15.3.2 Electrical Flow (Undirected Graphs)

We now consider electrical flows in undirected graphs. For each edge e, we orient it arbitrarily and fix this orientation. Let f_e be positive or negative:

- If $f_e > 0$: flow goes along the chosen orientation
- If $f_e < 0$: flow goes in the reverse direction

Flow Conservation: With this notation:

$$\sum_{v \in N(u)} f_{u,v} = d(u) \quad \forall u \in V$$

where $d(u)$ is the current injected into u (can be positive or negative).

Energy Minimization: Each edge $e \in E$ has a resistance $r_e \geq 0$. An electrical flow minimizes energy:

The energy consumed by flow f_e over edge with resistance r_e is $r_e f_e^2$ (voltage $\times$ current).

Thus, an electrical flow minimizes:

$$E(f) = \sum_{e \in E} r_e f_e^2$$

Optimization Problem:

$$\begin{aligned} \min \quad & \sum_{e \in E} r_e f_e^2 \\ \text{s.t.} \quad & \sum_{v \in N(u)} f_{u,v} = d(u) \quad \forall u \in V \end{aligned}$$

where f_e are unrestricted (no capacity constraints). The objective is convex quadratic and the constraint is a set of linear constraints.

15.4 Constrained Optimization and Lagrange Multipliers

Abstract Problem:

$$\min g(\bar{x}) \quad \text{s.t.} \quad A\bar{x} = b$$

where g is convex and $\bar{x} \in \mathbb{R}^n$, A is $m \times n$.

Theorem 15.2. *Let $\bar{x}^*$ be an optimal solution. Then $\nabla g(\bar{x}^*) = A^T y$ for some $y \in \mathbb{R}^m$, where $\nabla g(\bar{x}^*)$ is the gradient of g.*

Proof. Let $\text{Ker}(A) = \{\bar{x} : A\bar{x} = 0\}$.

We claim that $\nabla g(\bar{x}^*)$ is orthogonal to $\text{Ker}(A)$.

Suppose not. Then there exists $\bar{z} \in \text{Ker}(A)$ such that $\nabla g(\bar{x}^*) \cdot \bar{z} > 0$.

For small $\epsilon > 0$, let $\bar{x} = \bar{x}^* + \epsilon\bar{z}$. Then:

- $A\bar{x} = A(\bar{x}^* + \epsilon\bar{z}) = A\bar{x}^* + \epsilon A\bar{z} = b + 0 = b$
- $g(\bar{x}) = g(\bar{x}^* + \epsilon\bar{z}) \approx g(\bar{x}^*) + \epsilon\nabla g(\bar{x}^*) \cdot \bar{z} < g(\bar{x}^*)$

This contradicts optimality of $\bar{x}^*$.

Geometrically: If $a_1, \ldots, a_m \in \mathbb{R}^n$ are the rows of A, then $\text{Ker}(A)$ is the set of all vectors orthogonal to the space spanned by $a_1, \ldots, a_m$.

Any vector orthogonal to $\text{Ker}(A)$ is in the span of $a_1, \ldots, a_m$, and is of the form:

$$\sum_{i=1}^{m} y_i a_i = A^T y \quad \text{for some } y \in \mathbb{R}^m$$

Therefore, $\nabla g(\bar{x}^*) = A^T y$ for some $y \in \mathbb{R}^m$. □

15.5 Application to Electrical Flow

Theorem: Suppose $f^* = (f_e^*)_{e \in E}$ is an optimum solution to the electrical flow problem. Then there exists a potential function $p : V \to \mathbb{R}$ such that:

$$f_e^* = \frac{p(u) - p(v)}{r_e}$$

for each edge $e = (u, v)$.

In other words, there exist voltages that induce the electrical current via Ohm's law.

Note: Shifting all potentials by a constant Δ does not change currents. Hence, we can assume $p(t) = 0$ for some vertex t.

15.6 Connecting Electrical Flow and Hitting Times

Fix a vertex $t \in V$. We want to find $h(u)$ for $u \in V$, where $h(u)$ is the expected time for a random walk to hit t starting at u. Let $h(t) = 0$.

These values satisfy the recurrence:

$$h(u) = 1 + \sum_{v \in N(u)} \frac{h(v)}{\deg(u)} \quad \forall u \neq t$$

Rearranging:

$$\deg(u) \cdot h(u) - \sum_{v \in N(u)} h(v) = \deg(u)$$

We consider potentials on V where $p(u) = h(u)$ with $p(t) = 0$. This induces flow:

$$f_{u,v} = \frac{p(u) - p(v)}{r_{u,v}} = \frac{h(u) - h(v)}{1} = h(u) - h(v)$$

(setting edge resistances $r_e = 1$).
The outflow from u is:

$$\sum_{v \in N(u)} f_{u,v} = \sum_{v \in N(u)} (h(u) - h(v)) = \deg(u) \cdot h(u) - \sum_{v \in N(u)} h(v)$$

If $u \neq t$, this equals $\deg(u)$.
By flow conservation, the flow entering t is:

$$\sum_{u \neq t} d(u) = 2m - \deg(t)$$

where $d(u) = \deg(u)$ for $u \neq t$ and $d(t) = -(2m - \deg(t))$.
Thus, $h(u, t)$ values correspond to vertex potentials induced by:

- Injecting $\deg(u)$ current at each u
- Draining $2m - \deg(t)$ units of current at t

15.7 Lemma: Hitting Times and Effective Resistance

Lemma 15.3. *Let $G = (V, E)$ be an undirected graph and $s, t \in V$. Then:*

$$h(s) - h(t) = 2m \cdot R_{s,t}$$

where $R_{s,t}$ is the effective resistance between s and t.

Proof. We have seen that $h(t)$ corresponds to potentials that route $2m \cdot d(t)$ flow current into t and $\deg(u)$ flow out at each u.

Similarly, $h(s)$ corresponds to potentials that drain $2m \cdot d(s)$ units of current at s and $\deg(u)$ flow out from each $u \neq s$.

Let $p(u) = h(u)$ and $q(u) = h(s)$. Consider $p(u) - q(u)$.

The net flow at u is:

$$\begin{aligned} d(u) &= d(u)|_s - d(u)|_t \\ &= \deg(u) - (2m - \deg(t)) - [\deg(u) - (2m - \deg(s))] \\ &= \deg(s) - \deg(t) + 2m - 2m \\ &= 0 \quad \text{except at } s, t \end{aligned}$$

At s: $d(s) = 2m - \deg(s) - [-(2m - \deg(s))] = 2m$
At t: $d(t) = -(2m - \deg(t)) - 0 = -(2m - \deg(t))$
Actually: At s, $d(s) = -2m + \deg(s) + (2m - \deg(s)) = 0$... [recalculating]

More directly: The potential difference is:

$$p(s) - p(t) = h(s) - h(t)$$

This induces current $2m$ flowing from s to t:

$$h(s) - h(t) = 2m \cdot R_{\text{eff}}(s,t)$$

□

15.8 Laplacian Matrix

Given an undirected graph $G = (V, E)$, we associate the **Laplacian matrix** L_G.

15.8.1 Adjacency Matrix

The adjacency matrix is:

$$A_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

For undirected graphs, A is symmetric.

For directed graphs, we set A_{ij} to indicate presence of edge (i, j), and A is not necessarily symmetric.

15.8.2 Edge-Vertex Incidence Matrix

The edge-vertex incidence matrix B is an $m \times n$ matrix (where $m = |E|$, $n = |V|$):

$$B_{ei} = \begin{cases} 1 & \text{if vertex } i \text{ is incident to edge } e \\ 0 & \text{otherwise} \end{cases}$$

In a directed graph:

$$B_{ei} = \begin{cases} 1 & \text{if } e \text{ leaves } u \\ -1 & \text{if } e \text{ enters } u \end{cases}$$

The set of flows satisfying a demand vector d can be written as:

$$Bf = d$$

Note that f is not restricted to be non-negative.

When working with electrical flows in undirected graphs, we orient the edges arbitrarily and use B for the resulting edge-vertex incidence matrix.

15.8.3 Laplacian Definition

Recall that when solving the electrical flow optimization problem:

$$\min \sum_e r_e f_e^2 \quad \text{s.t.} \quad Bf = d$$

the optimal solution f^* satisfies:

$$2rf^* = B^T p$$

where p is the set of potentials.

We can directly write a linear system to find the potentials that satisfy demands via Ohm's law.

Flow Conservation:

$$\sum_{v \in N(u)} f_{u,v} = d(u)$$

But $f_{u,v} = \frac{p(u)-p(v)}{r_e}$.

Let $w(u,v) = 1/r_e$ be the conductance. Then:

$$\sum_{v \in N(u)} w(u,v)[p(u) - p(v)] = d(u)$$

Expanding:

$$\sum_{v \in N(u)} w(u,v)p(u) - \sum_{v \in N(u)} w(u,v)p(v) = d(u)$$

This is a linear system:

$$L_G p = d$$

where L_G is the **Laplacian** of G, defined as:

$$L_G(u,u) = \sum_{v \in N(u)} w(u,v) \quad \text{(weighted degree of } u\text{)}$$

$$L_G(u,v) = -w(u,v) \quad \text{for } u \neq v,\ (u,v) \in E$$

$$L_G(u,v) = 0 \quad \text{for } u \neq v,\ (u,v) \notin E$$

15.8.4 Properties of the Laplacian

L_G is a symmetric, diagonally dominant matrix:

$$|L_{ii}| \geq \sum_{i \neq j} |L_{ij}|$$

We can solve $L_G p = d$ to compute effective resistance and $h(u)$ values in polynomial time. Near-linear time algorithms are known.

L_G is singular, but for a connected graph, its rank is $n - 1$ (the null space is spanned by the all-ones vector).

15.9 Computational Considerations

The $h(u)$ values can be computed by solving:

$$L_G h = b$$

where:

$$b(u) = \deg(u) \quad \text{for } u \neq t$$

$$b(t) = -(2m - \deg(t))$$

for a target vertex t.

Chapter 16

Convergence and Mixing Times

16.1 Introduction

Let $G = (V, E)$ be a connected graph. We have seen various properties of random walks on connected graphs:

1. The stationary distribution depends only on degrees: $\pi_i = \frac{\deg(i)}{2m}$.

2. Cover time is $\leq 2m(n-1)$.

3. $h_{s,t} + h_{t,s} = 2m \cdot R^{\text{eff}}_{s,t}$.

4. Connection to electrical networks.

We are interested in the convergence of the random walk to its stationary distribution. In order to make this formal, we define the following.

Definition 16.1. Given two distributions π and π' on V, their *total variational distance* is

$$\|\pi - \pi'\|_{TV} = \frac{1}{2} \sum_v |\pi(v) - \pi'(v)|.$$

We say that a Markov chain converges to its stationary distribution if $\|\pi^t - \pi\|_{TV} \to 0$ as $t \to \infty$, where π is the stationary distribution.

We will be interested in the rate of convergence: starting from some arbitrary π^0, how long does it take to get ϵ-close to π?

Note: A random walk may be periodic, in which case it does not converge. We will then work with the lazy walk.

16.2 Random Walks and Spectral Analysis

We will focus on random walks in undirected graphs and spectral analysis based convergence. And then relate spectral analysis with combinatorial properties of a graph.

16.2.1 Setting up the Matrices

So far we have been working with distributions over states as row vectors. Since we will work with eigenvalues and eigenvectors a lot, it is useful to switch to $p^{(0)}, p^{(1)}, \ldots, p^{(t)}$ as column vectors. Recall P was the probability transition matrix of the chain. In order to work with column vectors we will use P^T. Thus $p^{(t+1)} = P^T p^{(t)}$ where $p^{(t)}$ is the distribution of the chain at time step t.

Consider a random walk on an undirected graph $G = (V, E)$ with $V = [n]$. Let A be the adjacency matrix of G. Let D be the $n \times n$ diagonal matrix with $D_{ii} = \deg(i)$. Then D^{-1} will be the matrix with $D^{-1}_{ii} = \frac{1}{\deg(i)}$.

Recall the transition matrix was defined as $P_{ij} = \frac{1}{\deg(i)}$ for $(i, j) \in E$, so $P = D^{-1}A$. Hence $P^T_{ij} = \frac{1}{\deg(j)}$, and we can see that $P^T = AD^{-1}$.

We call $W = AD^{-1}$ the *walk matrix*.

For the lazy random walk, the walk matrix is $W = \frac{1}{2}(I + AD^{-1})$.

Suppose G is d-regular. Then $W = AD^{-1} = \frac{1}{d}A$ is symmetric. Symmetric matrices have substantial structure and there is a beautiful and powerful spectral theory for them with many applications.

16.3 Review of Linear Algebra

16.3.1 Eigenvalues and Eigenvectors

For an $n \times n$ matrix $A \in \mathbb{R}^{n \times n}$, a vector $\bar{v} \in \mathbb{R}^n$ is an eigenvector of A if there exists $\lambda \in \mathbb{R}$ such that $A\bar{v} = \lambda\bar{v}$, where λ is an eigenvalue.

Eigenvalues are solutions to the polynomial $\det(A - \lambda I) = 0$. In general this polynomial may not have real roots and hence A may not have real eigenvalues/eigenvectors. However, if A is viewed as a $\mathbb{C}^{n \times n}$ matrix, then we have complex eigenvalues/eigenvectors since every univariate polynomial over $\mathbb{C}$ can be factorized.

Symmetric matrices are however special and have substantial structure. This is captured by the Spectral Theorem which has many applications.

16.3.2 Spectral Theorem

Theorem 16.2 (Spectral Theorem)**.** *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix. Then A has n real eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $\bar{x}_1, \ldots, \bar{x}_n$ which form an orthonormal basis of $\mathbb{R}^n$. Moreover, $A = V^T D V$ where D is diagonal with $D_{ii} = \lambda_i$ and $V = [\bar{x}_1 \cdots \bar{x}_n]$.*

16.3.3 Rayleigh Quotient

A useful characterization of the eigenvalues is obtained via the Rayleigh quotient. Given A and $\bar{x} \in \mathbb{R}^n$, consider $x^T Ax$ which is the quadratic form induced by A. Note that

$$x^T Ax = \sum_{1 \leq i,j \leq n} A_{ij} x_i x_j.$$

When A is a symmetric matrix we get $\sum_{i=1}^n A_{ii} x_i^2 + 2\sum_{i<j} A_{ij} x_i x_j$.

Consider the problem of $\max / \min_{\|x\|=1} x^T Ax$, which is the same as $\max / \min_{\bar{x} \neq 0} \frac{x^T Ax}{x^T x}$.

If A is symmetric then

$$\lambda_1 = \min_{\substack{\bar{v} \in \mathbb{R}^n \\ \|\bar{v}\|=1}} v^T Av, \quad \lambda_n = \max_{\substack{\bar{v} \in \mathbb{R}^n \\ \|\bar{v}\|=1}} v^T Av.$$

In fact one can characterize all eigenvalues:

$$\lambda_k = \min_{V_k} \max_{\substack{\bar{x} \in V_k \\ \bar{x} \neq 0}} \frac{x^T A x}{x^T x}$$

where V_k is a k-dimensional subspace of $\mathbb{R}^n$.

One can derive this characterization from the spectral theorem or directly. Suppose $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ are orthonormal eigenvectors of A and let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Let $\bar{v} \in \mathbb{R}^n$ with $\|\bar{v}\| = 1$ be any unit vector. Then $\bar{v} = \sum_{i=1}^n d_i \bar{x}_i$ where $\sum_{i=1}^n d_i^2 = 1$. We have $v^T A v = \sum_{i=1}^n \lambda_i d_i^2$. Hence $\max_{\|\bar{v}\|=1} v^T A v = \lambda_n$ and similarly $\min_{\|\bar{v}\|=1} v^T A v = \lambda_1$.

16.3.4 Positive Semidefinite Matrices

Definition 16.3. A real symmetric matrix A is positive semidefinite (PSD), written $A \succeq 0$, if $\bar{v}^T A \bar{v} \geq 0$ for all $\bar{v} \in \mathbb{R}^n$.

Theorem 16.4. *A symmetric matrix A is PSD if and only if one of the following equivalent conditions holds:*

1. $\bar{v}^T A \bar{v} \geq 0$ *for all* $\bar{v} \in \mathbb{R}^n$.
2. *All eigenvalues of A are non-negative:* $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
3. $A = W^T W$ *for some* $W \in \mathbb{R}^{n \times n}$.

Proof. (3) $\Rightarrow$ (1): $v^T A v = v^T W^T W v = u^T u \geq 0$ where $u = Wv$.

(1) $\Rightarrow$ (2): Recall for symmetric A, $\lambda_1 = \min_{\|\bar{v}\|=1} v^T A v$. Hence $\lambda_1 \geq 0$.

(2) $\Rightarrow$ (3): A is symmetric, so $A = V^T D V$ where $D_{ii} = \lambda_i$. If $\lambda_i \geq 0$ for all i, then $D = (D^{1/2})^T D^{1/2}$ where $D_{ii}^{1/2} = \sqrt{\lambda_i}$. Hence $A = V^T (D^{1/2})^T D^{1/2} V = W^T W$. □

16.4 Convergence Analysis for Regular Graphs

We will first consider d-regular graphs. $W = AD^{-1} = \frac{1}{d}A$ is symmetric. Note that W is doubly stochastic. By the spectral theorem all eigenvalues are real. Let $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$ be the eigenvalues of W.

Claim 16.5. $\alpha_1 = 1$ *and* $\alpha_n \geq -1$.

Proof. Exercise. □

Exercise 16.6. Show that $\alpha_n = -1$ implies G is bipartite. Otherwise $|\alpha_n| < 1$.

For the lazy walk $W = \frac{1}{2}(I + \frac{1}{d}A)$, the eigenvalues are $\alpha'_i = \frac{1+\alpha_i}{2}$ for $i = 1, \ldots, n$. So $\alpha'_1 = 1$ and $\alpha'_n = \frac{1+\alpha_n}{2} > 0$ (since $\alpha_n > -1$ for connected non-bipartite graphs, and $\alpha_n \geq -1$ always).

16.4.1 Spectral Decomposition and Convergence

By the spectral theorem,

$$W = \sum_{i=1}^{n} \alpha_i v_i v_i^T$$

where $v_1, \ldots, v_n$ are the orthonormal eigenvectors of W and $\otimes$ denotes the outer product.

We want to know $W^t p^{(0)}$ where $p^{(0)}$ is the starting distribution. Since $v_1, \ldots, v_n$ are orthonormal, we can write $p^{(0)} = \sum_{i=1}^{n} c_i v_i$ where $c_i = \langle p^{(0)}, v_i \rangle$.

Then

$$W^t p^{(0)} = \sum_{i=1}^{n} c_i \alpha_i^t v_i.$$

Recall $\alpha_1 = 1$ and $v_1 = \frac{1}{\sqrt{n}}\bar{1}$ (since we normalize to unit vectors).

Spectral gap. Define $\beta = \min(1 - \alpha_2, 1 - |\alpha_n|)$. Note $0 \le \beta \le 1$.

Suppose $\beta > 0$. Then $\alpha_2 < 1$ and $|\alpha_n| < 1$, so $|\alpha_i| < 1$ for all $i \ge 2$. Thus $\alpha_i^t \to 0$ as $t \to \infty$ for $i \ge 2$, and

$$W^t p^{(0)} \to c_1 v_1.$$

Now $c_1 = \langle p^{(0)}, v_1 \rangle = \langle p^{(0)}, \frac{1}{\sqrt{n}}\bar{1} \rangle = \frac{1}{\sqrt{n}}$ since $\sum_i p_i^{(0)} = 1$. Hence $c_1 v_1 = \frac{1}{n}\bar{1}$, which is the uniform distribution—and the stationary distribution for a regular graph.

16.4.2 Rate of Convergence

Claim 16.7. *For d-regular graphs, the mixing time is $O\left(\frac{\ln n}{\beta}\right)$.*

Proof. We have

$$W^t p^{(0)} = \bar{\pi} + \sum_{i=2}^{n} c_i \alpha_i^t v_i.$$

Hence

$$d_{TV}(W^t p^{(0)}, \bar{\pi}) = \|W^t p^{(0)} - \bar{\pi}\|_1 = \left\| \sum_{i=2}^{n} c_i \alpha_i^t v_i \right\|_1 \le \sqrt{n} \left\| \sum_{i=2}^{n} c_i \alpha_i^t v_i \right\|_2$$

by Cauchy–Schwarz ($\|\bar{u}\|_1 \le \sqrt{n}\|\bar{u}\|_2$).

Since v_i are orthonormal:

$$\left\| \sum_{i=2}^{n} c_i \alpha_i^t v_i \right\|_2^2 = \sum_{i=2}^{n} c_i^2 \alpha_i^{2t} \le (1-\beta)^{2t} \sum_{i=2}^{n} c_i^2.$$

Now $\sum_{i=2}^{n} c_i^2 \le \sum_{i=1}^{n} c_i^2 = \|p^{(0)}\|_2^2 \le \|p^{(0)}\|_1^2 = 1$.

Hence

$$d_{TV}(W^t p^{(0)}, \bar{\pi}) \le (1-\beta)^t \sqrt{n}.$$

We want $(1-\beta)^t \sqrt{n} \le \frac{1}{4}$. Taking logarithms: $t \ln(1-\beta) \le -\ln(4\sqrt{n})$, which gives $t \ge \frac{\ln(4\sqrt{n})}{-\ln(1-\beta)}$. Since $-\ln(1-\beta) \ge \beta$, it suffices to take $t = \Omega\left(\frac{\ln n}{\beta}\right)$. □

16.4.3 Lazy Random Walk

For the lazy walk $W = \frac{1}{2}(I + \frac{1}{d}A)$, recall the eigenvalues are $\alpha_i' = \frac{1+\alpha_i}{2}$. The spectral gap becomes

$$\beta = \min(1 - \alpha_2', 1 - \alpha_n') = 1 - \alpha_2' = \frac{1 - \alpha_2}{2}$$

since $\alpha_n' = \frac{1+\alpha_n}{2} > 0$ and $\alpha_2' = \frac{1+\alpha_2}{2}$ is the second largest eigenvalue.

16.5 Example: Cycle Graph

Let $G = C_n$ (the n-cycle). What are the eigenvalues of $AD^{-1} = \frac{1}{2}A$? One can show they are

$$\alpha_i = \cos\left(\frac{2\pi(i-1)}{n}\right), \quad i = 1, \ldots, n.$$

If n is even, $\alpha_n = -1$, so the spectral gap is 0. Hence we need to use the lazy walk.

What about α_2? We have $\alpha_2 = \cos\left(\frac{2\pi}{n}\right)$. Using $\cos x \approx 1 - \frac{x^2}{2}$ as $x \to 0$:

$$\alpha_2 \approx 1 - \frac{2\pi^2}{n^2}.$$

So $\beta \leq \frac{1}{n^2}$, giving convergence time $\Omega(n^2)$. Not surprising.

16.6 General Non-Regular Graphs

For graphs that are not necessarily regular, the lazy walk matrix is $W = \frac{1}{2}(I + AD^{-1})$. W is not symmetric, so we cannot use the spectral theorem directly.

16.6.1 Normalized Adjacency Matrix

Consider the normalized adjacency matrix $\tilde{A} = D^{-1/2}AD^{-1/2}$, which has entries $\tilde{A}_{ij} = \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(j)}}$ for $(i, j) \in E$. This is symmetric.

We can write

$$W = D^{1/2}\left(\frac{1}{2}(I + D^{-1/2}AD^{-1/2})\right)D^{-1/2}.$$

So W is similar to the symmetric matrix $\frac{1}{2}(I + \tilde{A})$.

16.6.2 Similar Matrices

Definition 16.8. Two $n \times n$ matrices X and Y are *similar* if there exists a non-singular matrix B such that $X = BYB^{-1}$.

The action of X can be understood via the action of Y.

Claim 16.9. *Similar matrices have the same eigenvalues. Eigenvectors may be different.*

Proof. Suppose $Y\bar{v} = \lambda\bar{v}$. Let $\bar{u} = B\bar{v}$, so $B^{-1}\bar{u} = \bar{v}$. Then

$$X\bar{u} = (BYB^{-1})\bar{u} = B(Y\bar{v}) = B(\lambda\bar{v}) = \lambda\bar{u}.$$

So λ is an eigenvalue of X. □

Corollary 16.10. *If a matrix X is similar to a symmetric matrix, then all its eigenvalues are real. Its eigenvectors span $\mathbb{R}^n$, even though they may not be orthonormal.*

16.6.3 Eigenvalues and Eigenvectors of W

Now back to W. We have $W = D^{1/2}\left(\frac{1}{2}I + \frac{1}{2}D^{-1/2}AD^{-1/2}\right)D^{-1/2}$. So W is similar to $\frac{1}{2}(I + \tilde{A})$, which is symmetric (the "normalized adjacency matrix" part $\tilde{A} = D^{-1/2}AD^{-1/2}$ is symmetric).

The eigenvalues of $\tilde{A}$ are $1 = \alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n \geq -1$. Hence the eigenvalues of W are

$$\alpha_i' = \frac{1 + \alpha_i}{2}, \quad i = 1, \ldots, n.$$

If $\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n$ are orthonormal eigenvectors of $\frac{1}{2}(I + \tilde{A})$, then the eigenvectors of W are $D^{1/2}\bar{v}_i$ for $i = 1, \ldots, n$. Note that $\bar{v}_1, \ldots, \bar{v}_n$ are orthonormal, so $D^{1/2}\bar{v}_1, \ldots, D^{1/2}\bar{v}_n$ are linearly independent and span $\mathbb{R}^n$ (but are not orthonormal in general).

16.6.4 Convergence for General Graphs

Now we want to understand $W^t\bar{p}_0$ where $\bar{p}_0$ is the starting distribution. Since the eigenvectors of W span $\mathbb{R}^n$:

$$\bar{p}_0 = \sum_{i=1}^{n} c_i D^{1/2}\bar{v}_i$$

for some $c_1, \ldots, c_n$.

Therefore

$$W^t\bar{p}_0 = \sum_{i=1}^{n} c_i W^t(D^{1/2}\bar{v}_i) = \sum_{i=1}^{n} c_i \left(\frac{1 + \alpha_i}{2}\right)^t D^{1/2}\bar{v}_i.$$

Recall $1 = \alpha_1 > \alpha_2 \geq \cdots \geq \alpha_n > -1$ (for a connected graph). So $\alpha_1' = 1$ and $|\alpha_i'| < 1$ for $i \geq 2$. If $\beta_2 < 1$, then

$$W^t\bar{p}_0 = c_1 D^{1/2}\bar{v}_1 + \sum_{i=2}^{n} c_i \beta_i^t D^{1/2}\bar{v}_i$$

where $\beta_i = \frac{1+\alpha_i}{2}$. Since $|\beta_i| < 1$ for $i \geq 2$:

$$W^t\bar{p}_0 \to c_1 D^{1/2}\bar{v}_1 \quad \text{as } t \to \infty.$$

This converges to the stationary distribution. What is $c_1 D^{1/2}\bar{v}_1$?

Computing $\bar{v}_1$: The first eigenvector of $\frac{1}{2}(I + \tilde{A})$ with eigenvalue 1 is

$$\bar{v}_1 = \frac{D^{1/2}\bar{1}}{\|D^{1/2}\bar{1}\|} = \frac{1}{\sqrt{2m}} D^{1/2}\bar{1}$$

since $\|D^{1/2}\bar{1}\|^2 = \sum_i \deg(i) = 2m$.

Computing c_1: Recall $\bar{p}_0 = D^{1/2}\sum_{i=1}^{n} c_i\bar{v}_i$, so $D^{-1/2}\bar{p}_0 = \sum_{i=1}^{n} c_i\bar{v}_i$. Since $\bar{v}_1, \ldots, \bar{v}_n$ are orthonormal:

$$c_1 = \langle D^{-1/2}\bar{p}_0, \bar{v}_1\rangle = \left\langle D^{-1/2}\bar{p}_0, \frac{1}{\sqrt{2m}} D^{1/2}\bar{1}\right\rangle = \frac{1}{\sqrt{2m}}$$

since $\sum_i p_0(i) = 1$.

Therefore

$$W^t\bar{p}_0 \to c_1 D^{1/2}\bar{v}_1 = \frac{1}{\sqrt{2m}} \cdot D^{1/2}\left(\frac{1}{\sqrt{2m}} D^{1/2}\bar{1}\right) = \frac{1}{2m} D\bar{1} = \frac{1}{2m}\begin{bmatrix}\deg(1)\\ \deg(2)\\ \vdots \\ \deg(n)\end{bmatrix}.$$

Hence the lazy walk converges to the stationary distribution $\bar{\pi} = \frac{1}{2m} D\bar{1}$ whatever the starting distribution, provided G is connected (since $\beta_2 < 1$ when G is connected).

16.6.5 Mixing Time for General Graphs

We want to know how long before $\|\bar{p}_t - \bar{\pi}\|_1 \le \frac{1}{4}$.

Theorem 16.11 (Mixing Time Bound)**.** *For a connected graph with spectral gap $\beta = 1 - \alpha_2' = \frac{1-\alpha_2}{2}$ and minimum degree $d_{\min}$,*

$$\|\bar{p}_t - \bar{\pi}\|_{TV} \le \sqrt{\frac{d_{\max}}{d_{\min}}} \cdot (1-\beta)^t.$$

To achieve $\|\bar{p}_t - \bar{\pi}\|_{TV} \le \frac{1}{4}$, it suffices to take $t = O\Big(\frac{1}{\beta} \log \frac{d_{\max}}{d_{\min}}\Big)$.

Proof. We have

$$\bar{p}_t - \bar{\pi} = \sum_{i=2}^{n} c_i \left(\frac{1+\alpha_i}{2}\right)^t D^{1/2} \bar{v}_i.$$

Multiplying by $D^{-1/2}$:

$$D^{-1/2}(\bar{p}_t - \bar{\pi}) = \sum_{i=2}^{n} c_i \left(\frac{1+\alpha_i}{2}\right)^t \bar{v}_i.$$

By Cauchy–Schwarz, $\|\bar{u}\|_1 \le \sqrt{n}\|\bar{u}\|_2$. Also, for any vector $\bar{z}$:

$$\|D^{-1/2}\bar{z}\|_2^2 = \sum_i \frac{z_i^2}{\deg(i)} \ge \frac{1}{d_{\max}} \|\bar{z}\|_2^2.$$

Since $\bar{v}_1, \ldots, \bar{v}_n$ are orthonormal:

$$\|D^{-1/2}(\bar{p}_t - \bar{\pi})\|_2^2 = \sum_{i=2}^{n} c_i^2 \left(\frac{1+\alpha_i}{2}\right)^{2t} \le (1-\beta)^{2t} \sum_{i=2}^{n} c_i^2.$$

Now $\sum_{i=2}^n c_i^2 \le \sum_{i=1}^n c_i^2 = \|D^{-1/2}\bar{p}_0\|_2^2 = \sum_i \frac{p_0(i)^2}{\deg(i)} \le \frac{1}{d_{\min}}$ (since $p_0(i) \le 1$ and $\sum_i p_0(i) = 1$, so $\sum_i p_0(i)^2 \le 1$).

Combining:

$$\frac{1}{d_{\max}} \|\bar{p}_t - \bar{\pi}\|_2^2 \le \|D^{-1/2}(\bar{p}_t - \bar{\pi})\|_2^2 \le \frac{(1-\beta)^{2t}}{d_{\min}}.$$

Hence $\|\bar{p}_t - \bar{\pi}\|_2 \le \sqrt{\frac{d_{\max}}{d_{\min}}}(1-\beta)^t$, and since $\|\cdot\|_{TV} \le \|\cdot\|_2$, the result follows. □

16.7 Next Lecture

We will investigate which graphs have constant spectral gap $\beta = \Omega(1)$. This ensures that random walks converge in $O(\log n)$ steps.

Chapter 17

Expander Graphs

17.1 Expander Graphs

Expander graphs are almost magical graphs that have many applications in mathematics and computer science.

Definition 17.1. A multigraph $G = (V, E)$ is an α-edge expander if

$$|E(S, \bar{S})| \geq \alpha |S|$$

for all $S \subseteq V$ with $|S| \leq \frac{|V|}{2}$.

A related concept is vertex expansion.

Definition 17.2. A graph $G = (V, E)$ is a c-vertex expander if for all $S \subseteq V$ with $|S| \leq \frac{|V|}{2}$, the set of neighbors $N(S)$ (excluding S itself) satisfies

$$|N(S) \setminus S| \geq c|S|$$

The surprising fact is that random d-regular graphs are good expanders with high probability.

Theorem 17.3 (Bollobás)**.** *Let $d \geq 3$ be an integer and $\epsilon \in (0, 1)$. If n is sufficiently large (depending on ϵ and d), then a random d-regular graph on n vertices has edge expansion $\geq \frac{d-2}{d} - \epsilon$ with high probability.*

Corollary 17.4. *As $n \to \infty$, a random d-regular graph has expansion $\to \frac{d-2}{d}$ with high probability. In particular, for $d = 3$ we can obtain expansion $\to \frac{1}{3}$.*

The proof is via the probabilistic method and not very difficult, though somewhat technical. Simpler proofs often give weaker expansion bounds.

17.1.1 Explicit Constructions

Although random regular graphs are expanders, it is hard to compute (or prove) the expansion of a specific randomly generated graph. Therefore, there have been several works that construct expanders explicitly. Many of these are based on group theoretic constructions.

One of the early explicit constructions is given by Margulis and analyzed by Gabber and Galil.

Construction: Fix integer m and let $n = 2m^2$. We construct a 5-regular bipartite graph on n vertices. The parts are A and B, with $|A| = |B| = m^2$.

We can identify the vertices in each part with $\mathbb{Z}_m \times \mathbb{Z}_m$.

$$A = \mathbb{Z}_m \times \mathbb{Z}_m$$
$$B = \mathbb{Z}_m \times \mathbb{Z}_m$$

For each vertex $(x, y) \in A$, we add 5 edges to the following vertices in B:

$$(x, y), \quad (x, x + y), \quad (x, x + y + 1), \quad (x + y, y), \quad (x + y + 1, y)$$

Here, all addition is done mod m. It can be shown that this graph has expansion $\frac{2-\sqrt{3}}{4}$.

17.1.2 Conductance

A notion related to expansion is conductance.

Definition 17.5. The conductance ϕ of a graph is

$$\phi = \min_{S:\mathrm{Vol}(S)\leq\frac{\mathrm{Vol}(V)}{2}} \frac{|E(S, \bar{S})|}{\mathrm{Vol}(S)}$$

where $\mathrm{Vol}(S) = \sum_{u\in S} \deg(u)$ is the volume of the set S.

Note that if G is d-regular then $\mathrm{Vol}(S) = d|S|$ and $\mathrm{Vol}(V) = d|V|$. Hence for d-regular graphs, $\phi = \frac{\alpha}{d}$.

17.2 Cheeger's Inequality

Cheeger's inequality is a fundamental result that connects the combinatorial expansion property of a graph (its conductance) to its algebraic properties (the eigenvalues of its Laplacian matrix).

Recall that we did a spectral analysis of the convergence of a random walk in undirected graphs. Let A be the adjacency matrix of G. Let D be the diagonal matrix of degrees, $D_{ii} = d_i$. The random walk matrix is

$$W = AD^{-1}$$

The lazy random walk matrix is $\frac{1}{2}(I + AD^{-1})$.

If G is d-regular then $W = \frac{1}{d}A$, which is symmetric. Otherwise, W is not symmetric. We instead consider the normalized adjacency matrix

$$\mathcal{A} = D^{-1/2}AD^{-1/2}$$

which is symmetric. W is similar to $\mathcal{A}$ since $W = D^{-1}A = D^{-1/2}\mathcal{A}D^{1/2}$.

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the real eigenvalues of $\mathcal{A}$. It is known that $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq -1$. If G is connected, we have $\lambda_2 < 1$.

The rate of convergence of a random walk to its stationary distribution is governed by the spectral gap. For the lazy random walk, the convergence rate is determined by $1 - \lambda_2$. (The lazy walk is used to handle bipartite graphs, where $\lambda_n = -1$).

How do we know when $1 - \lambda_2$ is large? Cheeger's inequality relates this gap to the conductance ϕ.

We use another important matrix, the normalized Laplacian:

$$\mathcal{L} = I - \mathcal{A}$$

(The unnormalized Laplacian is $L = D - A$). $\mathcal{L}$ is positive semi-definite.

Observation 17.6. Let $0 = \gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_n$ be the eigenvalues of $\mathcal{L}$. Since $\mathcal{L} = I - \mathcal{A}$, we have $\gamma_i = 1 - \lambda_i$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of $\mathcal{A}$. Thus $\gamma_1 = 1 - \lambda_1 = 0$, $\gamma_2 = 1 - \lambda_2$, and so on.

The spectral gap $\gamma_2 = 1 - \lambda_2$ is the smallest non-zero eigenvalue of the normalized Laplacian.

Theorem 17.7 (Cheeger's Inequality)**.** *For any graph G, let ϕ be its conductance and γ_2 be the second smallest eigenvalue of its normalized Laplacian $\mathcal{L}$. Then*

$$\frac{\phi^2}{2} \leq \gamma_2 \leq 2\phi$$

This theorem is powerful because it means a graph has good expansion (constant ϕ) if and only if it has a large spectral gap (constant γ_2).

Corollary 17.8. *Suppose G is d-regular. Let $1 = \alpha_1 > \alpha_2 \geq \cdots \geq \alpha_n$ be eigenvalues of $W = AD^{-1} = \frac{1}{d}A$. Then $1 - \alpha_2 = \gamma_2$, and*

$$d \cdot \frac{1 - \alpha_2}{2} \leq \alpha(G) \leq d\sqrt{2(1 - \alpha_2)}.$$

Proof. For d-regular graphs, $\phi(G) = \alpha(G)/d$ and $1 - \alpha_2 = \gamma_2$. The result follows from Cheeger's inequality. □

Thus if we have a constant-degree expander, $1 - \alpha_2 \geq (\alpha(G)/d)^2$, and $\alpha(G) = \Omega(1)$ implies $1 - \alpha_2 = \Omega(1)$. This large spectral gap implies that the (lazy) random walk on G mixes rapidly, in $O(\log n)$ steps.

17.3 Randomized Complexity Classes

Recall **P** is the set of decision problems that have a deterministic poly-time algorithm.

17.3.1 RP (Randomized Polynomial Time)

$L \in \mathbf{RP}$ if there exists a poly-time randomized algorithm A such that for all inputs $x \in \Sigma^*$:

(i) If $x \notin L$, $A(x)$ always says No.

(ii) If $x \in L$, $A(x)$ says YES with probability $\geq \frac{1}{2}$.

This is called one-sided error.

17.3.2 co-RP

co-RP is the set of languages L such that $\bar{L} \in \mathbf{RP}$. (One-sided error for $x \notin L$).

17.3.3 BPP (Bounded-Error Probabilistic Polynomial Time)

BPP is the set of languages that admit poly-time randomized algorithms that can make 2-sided errors. $L \in \mathbf{BPP}$ if there exists a randomized poly-time algorithm A such that for all $x \in \Sigma^*$:

(i) If $x \in L$, $A(x)$ outputs YES with probability $\geq \frac{3}{4}$.

(ii) If $x \notin L$, $A(x)$ outputs YES with probability $\leq \frac{1}{4}$.

RP and **BPP** algorithms are called Monte Carlo algorithms: they are always fast (poly-time) but can make a mistake.

It is clear that $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{BPP}$.

17.3.4 ZPP (Zero-Error Probabilistic Polynomial Time)

ZPP is the set of problems for which there is a randomized algorithm A such that for all x:

(i) If $x \in L$, $A(x)$ returns YES.

(ii) If $x \notin L$, $A(x)$ returns No.

(iii) The expected run time of A on x is $p(|x|)$ for some fixed polynomial p.

These are called Las Vegas algorithms: they are always correct, but their runtime is a random variable (though polynomial on average).

Claim 17.9. ***ZPP*** $=$ ***RP*** $\cap$ ***co-RP***

Proof Sketch. ($\subseteq$) If $L \in$ **ZPP**, the ZPP algorithm itself works as an RP (and co-RP) algorithm, as it's always correct.

($\supseteq$) Suppose $L \in$ **RP** (with algorithm A_{RP}) and $L \in$ **co-RP** (with algorithm A_{coRP}).

Consider the following ZPP algorithm A_{ZPP}:

1. Run $A_{RP}(x)$. If it outputs YES, we know $x \in L$. Return YES.

2. Run $A_{coRP}(x)$. If it outputs NO, we know $x \notin L$. Return NO.

3. If neither of the above happened (i.e., A_{RP} said NO and A_{coRP} said YES), repeat from step 1.

This algorithm never errs. If $x \in L$, A_{RP} says YES with p $\geq 1/2$. If $x \notin L$, A_{coRP} says NO with p $\geq 1/2$. In either case, the loop terminates in any given iteration with probability $\geq 1/2$. Thus, the expected number of iterations is ≤ 2, and the expected runtime is polynomial. □

17.4 Error Reduction in Randomized Algorithms

For both **RP** and **BPP**, we can reduce the error probability exponentially by repetition.

Lemma 17.10. *Suppose $L \in$ **RP** with algorithm A. By running A k times independently, we can reduce the error probability to $\leq 2^{-k}$. (If $x \in L$, the probability that all k runs output NO is $\leq (1/2)^k$).*

Lemma 17.11. *Suppose $L \in$ **BPP** with algorithm A. By running A k times independently and taking the majority vote, the error probability is reduced to $2^{-\Omega(k)}$. (This is a standard application of Chernoff bounds).*

This standard repetition requires $k \cdot n$ random bits if one run of A uses n random bits. Can we do better? Yes, by using expander graphs. It turns out that one can reduce the error to 2^{-k} by using only $O(n + k)$ random bits.

17.4.1 Setup

Let $N = 2^n$. We view the set of all n-bit random strings as the N vertices of a graph. We will assume that we can implicitly construct a constant-degree expander $G = (V, E)$ on these N vertices. Let G have degree d and expansion α. We assume $d = O(1)$ and $\alpha = \Omega(1)$.

Each vertex $v \in V$ corresponds to an n-bit binary string r_v. We assume that given a vertex v, we can find its d neighbors in $\text{poly}(n)$ time. (This is true for explicit constructions like the Margulis-Gabber-Galil expander).

We can implement a random walk on G for t steps in $\text{poly}(t, n)$ time. The number of random bits required is only $O(t \log d) = O(t)$, since d is constant.

17.4.2 Algorithm for Amplification

For both **RP** and **BPP** amplification, we do the following:

1. Pick a uniformly random starting vertex $v_1 \in V$. (This requires n random bits).
2. Perform a random walk for t steps: $v_1, v_2, \ldots, v_t$. (This requires $O(t)$ additional random bits).
3. Let $r_1, r_2, \ldots, r_t$ be the n-bit random strings associated with $v_1, \ldots, v_t$.
4. Let $b_i = A(x, r_i)$ be the output of A on input x with random string r_i.
5. Aggregate the results $b_1, \ldots, b_t$ (details below).

The total number of random bits used is $n + O(t)$. If we set $t = O(k)$, the total is $O(n + k)$.

The correctness relies on the following key lemma.

Lemma 17.12. *Let $G = (V, E)$ be an undirected d-regular graph whose random walk matrix $W = AD^{-1} = \frac{1}{d}A$ has spectral gap $\beta = \min(1 - \alpha_2, 1 - |\alpha_n|)$.*

Consider a t-step random walk $v_1, v_2, \ldots, v_t \in V$ where $v_1 \in V$ is chosen uniformly at random. For any set $B \subseteq V$, let $\mu = \frac{|B|}{|V|}$.

$$\Pr[\{v_1, v_2, \ldots, v_t\} \subseteq B] \leq (\mu + (1 - \beta))^t.$$

Proof. Let P be the $|V| \times |V|$ diagonal matrix with $P_{vv} = 1$ if $v \in B$ and 0 otherwise. For any vector $\bar{x} \in \mathbb{R}^{|V|}$, $P\bar{x}$ zeros out the coordinates outside B.

Since v_1 is chosen uniformly, $p^{(0)} = \frac{1}{n}\bar{1}$ where $n = |V|$. The probability that $v_1, \ldots, v_t \in B$ equals $\|(PWP)^t p^{(0)}\|_1$.

We need to bound the spectral norm of PWP. Since W is symmetric (for d-regular graphs), PWP is also symmetric.

Claim: For any vector $\bar{y}$, $\|PWP\bar{y}\|_2 \leq (\mu + (1 - \beta))\|\bar{y}\|_2$.

Proof of Claim. We can assume $y_i = 0$ for $i \notin B$ (since P projects onto B, this can only help), $\bar{y} \geq 0$, and $\sum_i y_i = 1$ (by scaling). Write $\bar{y} = \bar{u} + \bar{z}$ where $\bar{u} = \frac{1}{n}\bar{1}$ is the uniform distribution and $\bar{z}$ is orthogonal to $\bar{u}$. Then:

$$PWP\bar{y} = PW(\bar{u} + \bar{z}) = PW\bar{u} + PW\bar{z} = P\bar{u} + PW\bar{z}.$$

By triangle inequality: $\|PWP\bar{y}\|_2 \leq \|P\bar{u}\|_2 + \|PW\bar{z}\|_2$.

For the first term: $\|P\bar{u}\|_2 = \sqrt{\mu n \cdot \frac{1}{n^2}} = \sqrt{\mu/n}$. Since $\bar{y}$ has support at most μn and $1 = \sum y_i \leq \sqrt{\mu n}\|\bar{y}\|_2$ by Cauchy–Schwarz, we get $\|\bar{y}\|_2 \geq 1/\sqrt{\mu n}$. Hence $\|P\bar{u}\|_2 \leq \mu\|\bar{y}\|_2$.

For the second term: $\|PW\bar{z}\|_2 \leq \|W\bar{z}\|_2 \leq (1-\beta)\|\bar{z}\|_2 \leq (1-\beta)\|\bar{y}\|_2$, since $\bar{z}$ is orthogonal to the first eigenvector of W.

Thus $\|PWP\bar{y}\|_2 \leq (\mu + (1-\beta))\|\bar{y}\|_2$, so the max eigenvalue of PWP is $\leq \mu + (1-\beta)$. □

Now we prove the main lemma. We have:

$$\|(PAP)^t\bar{u}\|_1 \leq \sqrt{n}\|(PAP)^t\bar{u}\|_2 \leq \sqrt{n}(\mu+(1-\beta))^t\|\bar{u}\|_2 = \sqrt{n}(\mu+(1-\beta))^t \cdot \frac{1}{\sqrt{n}} = (\mu+(1-\beta))^t.$$

□

17.4.3 RP Amplification

Suppose $L \in \mathbf{RP}$. If $x \notin L$, A always outputs No, so our expander walk algorithm will also always output No.

Suppose $x \in L$. Let B be the set of "bad" random strings r such that $A(x,r)$ outputs No. By the **RP** definition, $\mu = |B|/|V| \leq 1/2$. (We can first use standard repetition a few times to ensure $\mu \leq 1/4$).

The algorithm is:

1. Run the t-step random walk as described, getting outputs $b_1, \ldots, b_t$.
2. Output YES if any b_i is YES.
3. Otherwise, output NO.

The algorithm fails (outputs NO) only if $v_1, \ldots, v_t$ are all in the bad set B. By the lemma, the failure probability is:

$$\Pr[\text{Output NO}] = \Pr[\{v_1, \ldots, v_t\} \subseteq B] \leq (\mu + (1-\beta))^t.$$

Since G is an expander, β is a fixed constant. By basic repetition (a constant number of times) we can ensure that $\mu \leq \beta/2$. Then $\mu + 1 - \beta \leq 1 - \beta/2 < 1$.

If we choose t such that $(1-\beta/2)^t \leq 1/2^k$, we will have failure probability $\leq 1/2^k$. Thus $t = O(k)$ suffices (the $O(\cdot)$ hides a $1/\beta$ dependence, which is a fixed constant). The total random bits used is $O(n+k)$.

17.4.4 BPP Amplification

For **BPP**, we use the same t-step random walk but take a majority vote of the outputs $b_1, \ldots, b_t$. The analysis is more involved and relies on a Chernoff-like bound for random walks on expanders.

Theorem 17.13 (Expander Chernoff Bound)**.** *Let $G = (V, E)$ be a d-regular graph with spectral gap $\gamma_2 = 1 - \lambda_2 > 0$. Let $v_1, \ldots, v_t$ be a t-step random walk starting from a uniform v_1. Let $f : V \to [0,1]$ be any function, and let $\mathbb{E}[f] = \frac{1}{|V|}\sum_v f(v)$. Then*

$$\Pr\left[\left|\frac{1}{t}\sum_{i=1}^{t} f(v_i) - \mathbb{E}[f]\right| \geq \delta\right] \leq 2e^{-\Omega(\gamma_2\delta^2 t)}$$

For **BPP**, we define $f(r) = 1$ if $A(x,r)$ is correct, and 0 otherwise. By definition, $\mathbb{E}[f] \geq 2/3$. The theorem shows that the average correctness $\frac{1}{t}\sum f(v_i)$ will be concentrated around $\mathbb{E}[f]$ with high probability. This means the majority vote will be correct with probability $1 - 2e^{-\Omega(t)}$. Setting $t = O(k)$ gives error $2^{-\Omega(k)}$, using only $O(n+k)$ random bits.

Chapter 18

Negative Correlation and Applications

18.1 Introduction

We have seen Chernoff-Hoeffding bounds for sums of independent random variables. However, there are several situations where we have dependent random variables and we need to reason about them. In some situations, we can get concentration even with dependence.

18.1.1 Example: Balls and Bins

Suppose we throw n balls into n bins. Let X_i be the indicator for bin i to be empty. We see that:

$$\Pr[X_i = 1] = \left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e}$$

Let $X = \sum_{i=1}^{n} X_i$ be the number of empty bins, so $\mathbb{E}[X] = \frac{n}{e}$.

Note that $X_1, X_2, \ldots, X_n$ are **not independent**, so we cannot use the Chernoff bound directly. However, it turns out that the Chernoff bound holds for the upper tail.

18.2 Negative Correlation

Definition 18.1. A collection $X_1, \ldots, X_n$ of random variables is **negatively correlated** if:

$$\mathbb{E}\left[\prod_{i \in S} X_i\right] \leq \prod_{i \in S} \mathbb{E}[X_i]$$

for all subsets $S \subseteq \{1, 2, \ldots, n\}$.

Claim 18.2. *In the example we saw with balls and bins, $X_1, \ldots, X_n$ are negatively correlated.*

Proof. We have $\mathbb{E}[X_i] = \left(1 - \frac{1}{n}\right)^n$.

$\mathbb{E}\left[\prod_{i \in S} X_i\right]$ is the probability that all bins in S are empty.

Let $|S| = k$. Then:

$$\mathbb{E}\left[\prod_{i \in S} X_i\right] = \left(1 - \frac{k}{n}\right)^n$$

One can check that $\left(1 - \frac{k}{n}\right)^n \leq \left(1 - \frac{1}{n}\right)^{kn} = \left[\left(1 - \frac{1}{n}\right)^n\right]^k$. □

18.3 Chernoff Bound for Negatively Correlated Variables

Theorem 18.3. *Suppose $X_1, \ldots, X_n$ are binary random variables and are negatively correlated. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then:*

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$$

Note: The bound is exactly the same as for the standard upper tail in the multiplicative Chernoff bound. The reason for that is that in a certain formal sense, the whole moment generating function-based proof goes through.

18.3.1 Proof Sketch

Let $\bar{X}_1, \ldots, \bar{X}_n$ be independent binary random variables where $\mathbb{E}[\bar{X}_i] = \mathbb{E}[X_i]$.

Let $\bar{X} = \sum_{i=1}^{n} \bar{X}_i$, so $\mathbb{E}[\bar{X}] = \mathbb{E}[X] = \mu$.

The MGF proof for Chernoff bound proceeds as follows:

$$\Pr[\bar{X} \geq (1+\delta)\mu] = \Pr[e^{t\bar{X}} \geq e^{t(1+\delta)\mu}] \leq \frac{\mathbb{E}[e^{t\bar{X}}]}{e^{t(1+\delta)\mu}}$$

The key step where independence is used is in expanding:

$$e^{t\bar{X}} = \prod_{i=1}^{n} e^{t\bar{X}_i}$$

After this step, we only work with bounds on $\mathbb{E}[e^{t\bar{X}_i}]$, etc.

Now consider $X = \sum X_i$ where X_i are negatively correlated and $\mathbb{E}[X_i] = \mathbb{E}[\bar{X}_i]$ for all i.

If we can show that:

$$\mathbb{E}[e^{tX}] \leq \mathbb{E}[e^{t\bar{X}}]$$

then we are done. For this, we expand e^{tX} as $\sum_{j=0}^{\infty} \frac{t^j X^j}{j!}$ and use Taylor series expansion to each term with expectation outside:

$$e^{tX} = \prod_{i=1}^{n} (1 + (e^t - 1)X_i)$$

and similarly:

$$e^{t\bar{X}} = \prod_{i=1}^{n} (1 + (e^t - 1)\bar{X}_i)$$

In the product, we have terms which are polynomials in t and in the variables. Consider a term $X_{i_1}^{a_1} X_{i_2}^{a_2} \cdots X_{i_k}^{a_k}$ and the corresponding term $\bar{X}_{i_1}^{a_1} \bar{X}_{i_2}^{a_2} \cdots \bar{X}_{i_k}^{a_k}$.

Since variables are binary, we can drop the exponents, so we have $X_{i_1} X_{i_2} \cdots X_{i_k}$ and $\bar{X}_{i_1} \bar{X}_{i_2} \cdots \bar{X}_{i_k}$.

Now by negative correlation assumption:

$$\mathbb{E}[X_{i_1} X_{i_2} \cdots X_{i_k}] \leq \mathbb{E}[\bar{X}_{i_1}]\mathbb{E}[\bar{X}_{i_2}] \cdots \mathbb{E}[\bar{X}_{i_k}] = \mathbb{E}[\bar{X}_{i_1} \bar{X}_{i_2} \cdots \bar{X}_{i_k}]$$

Thus, term by term, we have $\mathbb{E}[e^{tX}] \leq \mathbb{E}[e^{t\bar{X}}]$, and we can proceed with the rest of the proof with $\bar{X}$ and $\bar{X}_i$.

18.3.2 Lower Tail Bound

Sometimes people define binary random variables $X_1, \ldots, X_n$ to be negatively correlated if for all $S \subseteq \{1, \ldots, n\}$:

$$\mathbb{E}\left[\prod_{i \in S} X_i\right] \leq \prod_{i \in S} \mathbb{E}[X_i]$$

and:

$$\mathbb{E}\left[\prod_{i \in S} (1 - X_i)\right] \leq \prod_{i \in S} (1 - \mathbb{E}[X_i])$$

If both conditions are satisfied, then we also get the lower tail bound for $X = \sum X_i$:

$$\Pr[X \leq (1 - \delta)\mu] \leq e^{-\frac{\delta^2 \mu}{2}}$$

18.4 Application: Max Coverage Problem

Consider the **Max Coverage** problem, which is a problem related to Set Cover.

Problem: Given a universe U of n elements and m sets $S_1, \ldots, S_m \subseteq U$, and an integer k, pick k of the given sets to maximize the size of their union. In other words, pick k sets to cover as many elements as possible.

A simple greedy algorithm gives a $(1 - 1/e)$ approximation. However, it does not give the same ratio for a slightly more general constraint, so we will instead consider an LP relaxation-based approach.

18.4.1 LP Relaxation

Variables:

- x_i for set S_i (chosen or not)
- z_j for whether element j is covered

$$\begin{aligned}
\max \quad & \sum_{j=1}^{n} z_j \\
\text{s.t.} \quad & \sum_{i=1}^{m} x_i \leq k \\
& \sum_{i : j \in S_i} x_i \geq z_j \quad \forall j \in [n] \\
& z_j \leq 1 \quad \forall j \in [n] \\
& x_i \geq 0 \quad \forall i \in [m]
\end{aligned}$$

Suppose we solve the above LP relaxation. Let OPT_{LP} be the value of the optimal fractional solution (x^*, z^*).

18.4.2 Randomized Rounding (Naive Approach)

A simple strategy is to pick each set S_i independently with probability x_i^*.

Let us evaluate the expected number of elements covered. Let $Y_j = 1$ if element j is covered.

$$\Pr[Y_j = 1] = 1 - \prod_{i:j \in S_i} (1 - x_i^*)$$

Since $1 - x \leq e^{-x}$ for all x:

$$\Pr[Y_j = 1] \geq 1 - \prod_{i:j \in S_i} e^{-x_i^*} = 1 - e^{-\sum_{i:j \in S_i} x_i^*} \geq 1 - e^{-z_j^*} \geq \left(1 - \frac{1}{e}\right) z_j^*$$

(Using the concavity of $1 - e^{-z}$: for $z \in [0, 1]$, $1 - e^{-z} \geq (1 - 1/e)z$.)

Thus, by linearity of expectation, the expected number of elements covered is:

$$\geq \sum_{j=1}^{n} \left(1 - \frac{1}{e}\right) z_j^* = \left(1 - \frac{1}{e}\right) \cdot \mathrm{OPT}_{\mathrm{LP}}$$

Problem: We may not satisfy the constraint that we pick at most k sets.

18.5 Pipage Rounding

How can we ensure that we satisfy the constraint and still get a good approximation for covering elements? We will discuss a rounding strategy called **pipage rounding**. This is a dependent rounding strategy.

18.5.1 Algorithm

1. Solve LP to obtain fractional solution $\bar{x} \in [0, 1]^m$
2. While $\bar{x}$ has fractional variables:
   (a) Let x_i, x_j be such that $0 < x_i, x_j < 1$
   (b) Let $\epsilon = \min\{x_i, 1 - x_j, 1 - x_i, x_j\}$
   (c) Toss a coin. If heads:
   - $x_i \leftarrow x_i + \epsilon$
   - $x_j \leftarrow x_j - \epsilon$

   Else:
   - $x_i \leftarrow x_i - \epsilon$
   - $x_j \leftarrow x_j + \epsilon$
3. Output sets with $x_i = 1$

Claim 18.4. *After the while loop terminates, $\bar{x}$ is integral and $\sum_{i=1}^{m} x_i = k$.*

Lemma 18.5. *Let X_i be the value of x_i at end of algorithm. Then $\mathbb{E}[X_i] = x_i^*$.*

Proof. In each step, it is easy to see that $\mathbb{E}[x_i]$ does not change. By induction on steps. □

Lemma 18.6. *The algorithm terminates in T steps where $\mathbb{E}[T] \leq$ poly(m).*

Proof. In each iteration with probability $1/2$, at least one variable becomes 0 or 1. If a variable is 0 or 1, it is not touched again. Initially, at most m fractional variables implies in expectation $T \leq 2m$. Can also prove high probability bound using Chernoff bounds. □

18.5.2 Main Technical Lemma

Lemma 18.7. *$X_1, \ldots, X_m$ are negatively correlated.*

The proof relies on the fact that expectations are preserved and only two variables are modified at each step. It is not difficult but we omit details.

18.5.3 Analysis

Thus, the rounding ensures that $\sum_{i=1}^{m} X_i = k$ deterministically and $X_1, \ldots, X_m$ are negatively correlated.

Now consider an element j. What is $\Pr[j \text{ is covered}]$?

$$\Pr[j \text{ is covered}] = 1 - \prod_{i:j \in S_i} (1 - X_i)$$

By negative correlation:

$$\prod_{i:j \in S_i} (1 - X_i) \leq \prod_{i:j \in S_i} (1 - \mathbb{E}[X_i]) = \prod_{i:j \in S_i} (1 - x_i^*)$$

Therefore:

$$\Pr[j \text{ is covered}] \geq 1 - \prod_{i:j \in S_i} (1 - x_i^*)$$

and hence we can use the same analysis as before: expected number of elements covered is $(1 - 1/e) \cdot \mathrm{OPT}_{\mathrm{LP}}$.

Thus, we maintain the constraint and obtain a $(1 - 1/e)$ approximation.

18.6 Generalization: Matroid Constraints

The above approach generalizes quite a bit to submodular function maximization subject to an arbitrary matroid constraint. We will not go into details but consider the following extension of Max K-Coverage.

As before, we have U and sets $S_1, \ldots, S_m$. Now the sets are colored. In other words, we partition the sets into ℓ groups $A_1, \ldots, A_\ell$.

Each group h has a bound k_h, and this implies that at most k_h sets can be chosen from A_h.

We can write a natural LP with this more complicated constraint:

$$\begin{aligned}
\max \quad & \sum_{j=1}^{n} z_j \\
\text{s.t.} \quad & \sum_{i \in A_h} x_i \leq k_h \quad \forall h \in [\ell] \\
& \sum_{i:j \in S_i} x_i \geq z_j \quad \forall j \in [n] \\
& z_j \leq 1 \quad \forall j \in [n] \\
& x_i \geq 0 \quad \forall i \in [m]
\end{aligned}$$

Now, as before, we can see that if we randomly round by picking each set S_i independently with probability x_i^*, we get expected coverage $(1 - 1/e) \cdot \mathrm{OPT}_{\mathrm{LP}}$.

It is not hard to generalize pipage rounding to this slightly more complex constraint. This yields a $(1 - 1/e)$ approximation.

Note: The natural greedy algorithm yields only a $1/2$ approximation for this generalization.

Chapter 19

Martingales and Concentration Inequalities

19.1 Introduction and Background

We saw Chernoff-Hoeffding bounds for sum of independent random variables and applications. Last lecture we saw concentration bounds also hold for negatively correlated random variables and saw an application. However, there are other settings where we don't have independence or negative correlation and still concentration holds.

Martingales provide a powerful framework for such bounds and are also somewhat natural for algorithms. We saw an example of a martingale type process in the last lecture on rounding a fractional solution for the Max K-Cover problem.

19.1.1 Background on Conditional Probability

Suppose Ω is a probability space and $X : \Omega \to \mathbb{R}$ is a real-valued random variable.

If $A \subseteq \Omega$ is an event, then $\mathbb{E}[X|A]$ is a real value, which, in the setting when X is discrete, is defined as:

$$\mathbb{E}[X|A] = \frac{1}{\Pr[A]} \sum_{\omega \in A} \Pr[\omega] X(\omega).$$

In the continuous setting it is:

$$\frac{1}{\Pr[A]} \int_A f(x)dx$$

where f is the probability density function (or similar function depending on the context).

Given random variables X and Y on Ω, the random variable $\mathbb{E}[X|Y]$ is defined as follows. If $Z = \mathbb{E}[X|Y]$, then $Z(\omega) = \mathbb{E}[X|Y = Y(\omega)]$ (this is in the discrete setting). The meaning is the following: Y partitions Ω into parts where Y is constant in each part. $Y = b$ for some fixed b constitutes a part which can be alternatively thought as an event $A_b = \{\omega | Y(\omega) = b\}$. $\mathbb{E}[X|Y]$ assigns a value to each ω equal to $\mathbb{E}[X|A_b]$.

Claim 19.1. $\mathbb{E}[\mathbb{E}[X|Y]] = \mathbb{E}[X]$.

Proof. Exercise. □

When we write $\mathbb{E}[X|Y_1, Y_2, ..., Y_n]$ where $Y_1, ..., Y_n$ are several random variables, then we are looking at the partition induced by $Y_1, ..., Y_n$ taking on different values in a joint way. The events

correspond to:

$$A_{b_1,b_2,\ldots,b_n} = \{\omega \in \Omega | Y_i(\omega) = b_i\}.$$

Lemma 19.2 (Tower Property)**.** $\mathbb{E}[\mathbb{E}[X|Y]|Z] = \mathbb{E}[X|Z]$ *(assuming Z is a "coarser" partition of Ω than Y, or more formally, $\sigma(Z) \subseteq \sigma(Y)$).*

Proof. Exercise. □

Technically, a more formal way to describe this is via σ-algebras and filtrations but that requires more background.

19.2 Martingales

Definition 19.3. A sequence of random variables $X_0, X_1, X_2, \ldots$ is a **martingale sequence with respect to another sequence of random variables** $Y_0, Y_1, \ldots$ if for $n \geq 0$:

1. X_n is determined by $Y_0, \ldots, Y_n$.
2. $\mathbb{E}[|X_n|] < \infty$ for all n.
3. $\mathbb{E}[X_{n+1}|Y_0, \ldots, Y_n] = X_n$.

A sequence $X_0, X_1, \ldots$ is a **martingale** if it is a martingale w.r.t. itself. That is:

1. $\mathbb{E}[|X_n|] < \infty$ and
2. $\mathbb{E}[X_{n+1}|X_0, \ldots, X_n] = X_n$.

Example 19.4. Suppose a gambler starts with a random amount X_0 of money on day 0 and goes to the casino every day and plays some slot machine which is fair. Let Y_i be the winnings on day i (≤ 0 if he/she loses). Let X_i be the total amount that player has at the end of the i-th game. Because each game is fair, $\mathbb{E}[Y_{i+1}] = 0$.

$$X_{n+1} = X_n + Y_{n+1}$$

$$\mathbb{E}[X_{n+1}|Y_1, \ldots, Y_n] = X_n + \mathbb{E}[Y_{n+1}|Y_1, \ldots, Y_n]$$

Since Y_{n+1} is independent of $Y_1, \ldots, Y_n$ and $\mathbb{E}[Y_{n+1}] = 0$:

$$\mathbb{E}[X_{n+1}|Y_1, \ldots, Y_n] = X_n + \mathbb{E}[Y_{n+1}] = X_n + 0 = X_n.$$

The main thing that martingales allow one to capture is that the choice of how much to bet and which slot machine to bet on can be arbitrarily dependent on all the information/choices up to the previous step.

19.2.1 Doob Martingale

Martingales allow one to capture a particular type of phenomenon where we are interested in a function $f : U \to \mathbb{R}$ for some object X, and we have a random variable X that takes values in U. In other words, X is a random object chosen from U according to some process.

This process can be defined by a sequence of random variables $Y_0, Y_1, \ldots, Y_n$. And X is determined by $Y_0, \ldots, Y_n$. $Y_0, \ldots, Y_i$ reveal partial information about X.

Let us define

$$X_i = \mathbb{E}[f(X)|Y_0, \ldots, Y_i].$$

We will assume that $\mathbb{E}[|f(X)|] < \infty$.

Claim 19.5. *$X_0, \ldots, X_n$ is a martingale sequence with respect to $Y_0, \ldots, Y_n$.*

Proof.

$$\mathbb{E}[X_{i+1}|Y_0, \ldots, Y_i] = \mathbb{E}[\mathbb{E}[f(X)|Y_0, \ldots, Y_{i+1}]|Y_0, \ldots, Y_i]$$

(by definition of X_{i+1}). Using the Tower Rule ($\mathbb{E}[\mathbb{E}[A|B]|C] = \mathbb{E}[A|C]$ if $\sigma(C) \subseteq \sigma(B)$):

$$\mathbb{E}[\mathbb{E}[f(X)|Y_0, \ldots, Y_{i+1}]|Y_0, \ldots, Y_i] = \mathbb{E}[f(X)|Y_0, \ldots, Y_i] = X_i$$

by definition. □

Remark 19.6. In some settings it is easier to view $f(X)$ as another random variable Z and define $X_i = \mathbb{E}[Z|Y_1, \ldots, Y_i]$.

Example 19.7 (Empty bins in balls and bins)**.** We throw m balls into n bins independently. We think of this as a sequential process where we place ball i in the i-th step and Y_i is the random choice of the i-th ball (i.e., which bin it falls into). Let X be the number of empty bins after all m balls are thrown. Then:

$$X_i = \mathbb{E}[X|Y_1, \ldots, Y_i]$$

$$X_0 = \mathbb{E}[X] = n\left(1 - \frac{1}{n}\right)^m$$

Example 19.8 (Chromatic number of $G(n,p)$)**.** Let $G(n,p)$ be a graph on n vertices where each potential edge in the graph is chosen to be added independently with probability p. Let X be the chromatic number of the random graph.

We can define a Doob martingale called the **edge exposure martingale** where $Y_1, Y_2, \ldots, Y_{\binom{n}{2}}$ correspond to the binary random variables for picking the edges in some fixed order.

$$X_0 = \mathbb{E}[\chi(G_{n,p})]$$

and $X_i = \mathbb{E}[X|Y_1, \ldots, Y_i]$.

We can define another Doob martingale called the **vertex exposure martingale** where $V_1, V_2, \ldots, V_n$ is an ordering of vertices and Y_i reveals information about the edges of vertex i to vertices 1 to $i-1$ in the random process.

19.3 Azuma-Hoeffding Inequality

Recall the additive change Hoeffding inequality. Let $X = \sum_{i=1}^n X_i$ where:

1. X_i are independent.
2. $X_i \in [a_i, b_i]$.

Then:

$$\Pr[X - \mathbb{E}[X] \geq \lambda] \leq e^{-\frac{\lambda^2}{2\sum_i (b_i - a_i)^2}}$$

and $\Pr[X - \mathbb{E}[X] \leq -\lambda] \leq e^{-\frac{\lambda^2}{2\sum_i (b_i - a_i)^2}}$.

A simpler form is when $[a_i, b_i] = [-c_i, c_i]$, in which case we have:

$$\Pr[X - \mathbb{E}[X] > \lambda] \leq e^{\frac{-\lambda^2}{2\sum_{i=1}^n c_i^2}}$$

Similarly for the lower tail.

The Azuma-Hoeffding bound extends this to the martingale setting.

Theorem 19.9 (Azuma-Hoeffding)**.** *Let $X_0, X_1, X_2, \ldots$ be a martingale sequence where*

$$|X_i - X_{i-1}| \leq c_i \quad \forall i \geq 1.$$

Then:

$$\Pr[X_n - X_0 \geq \lambda] \leq e^{\frac{-\lambda^2}{2\sum_{i=1}^n c_i^2}}$$

The theorem also holds for the lower tail: $\Pr[X_n - X_0 \leq -\lambda] \leq e^{\frac{-\lambda^2}{2\sum_{i=1}^n c_i^2}}$.

19.3.1 Proof of Azuma-Hoeffding

We need an auxiliary lemma.

Lemma 19.10 (Hoeffding's Lemma)**.** *Let X be a random variable in $[-1, 1]$ with $\mathbb{E}[X] = 0$. Then $\mathbb{E}[e^{aX}] \leq e^{a^2/2}$.*

Proof. The function e^{ax} is convex on $[-1, 1]$. For any $X \in [-1, 1]$, we can write $X = \frac{1+X}{2}(+1) + \frac{1-X}{2}(-1)$. Hence by convexity (Jensen's inequality):

$$e^{aX} \leq \frac{1+X}{2}e^a + \frac{1-X}{2}e^{-a} = \frac{e^a + e^{-a}}{2} + \frac{e^a - e^{-a}}{2}X.$$

If $X \in [-1, 1]$ with $\mathbb{E}[X] = 0$, taking expectation on both sides:

$$\mathbb{E}[e^{aX}] \leq \frac{e^a + e^{-a}}{2} + \frac{e^a - e^{-a}}{2}\mathbb{E}[X] = \frac{e^a + e^{-a}}{2}.$$

By Taylor expansion:

$$\frac{e^a + e^{-a}}{2} = 1 + \frac{a^2}{2!} + \frac{a^4}{4!} + \cdots \leq 1 + \frac{a^2}{2} + \frac{(a^2/2)^2}{2!} + \cdots = e^{a^2/2}.$$

□

Corollary 19.11. *If $\mathbb{E}[X] = 0$ and $X \in [-c, c]$, then $\mathbb{E}[e^{aX}] \leq e^{\frac{a^2c^2}{2}}$.*

Proof. Consider $X' = \frac{X}{c}$ and apply the previous Lemma. □

Proof of Azuma-Hoeffding. Let $t > 0$ be a parameter to be chosen later. By Markov's inequality:

$$\Pr[X_n - X_0 \geq \lambda] \leq \Pr[e^{t(X_n - X_0)} \geq e^{t\lambda}] \leq \frac{\mathbb{E}[e^{t(X_n - X_0)}]}{e^{t\lambda}}.$$

So it boils down to estimating $\mathbb{E}[e^{t(X_n - X_0)}]$.

Consider the differences $Z_i = X_i - X_{i-1}$. Recall that $|Z_i| \leq c_i$ and:

$$\mathbb{E}[Z_i | X_0, \ldots, X_{i-1}] = \mathbb{E}[X_i - X_{i-1} | X_0, \ldots, X_{i-1}]$$

$$= \mathbb{E}[X_i | X_0, \ldots, X_{i-1}] - X_{i-1} = X_{i-1} - X_{i-1} = 0.$$

Consider $\mathbb{E}[e^{tZ_i} | X_0, \ldots, X_{i-1}]$. By the Corollary with $a = t$ and $c = c_i$:

$$\mathbb{E}[e^{tZ_i} | X_0, \ldots, X_{i-1}] \leq e^{\frac{t^2 c_i^2}{2}}.$$

Now:

$$\mathbb{E}[e^{t(X_n - X_0)}] = \mathbb{E}[e^{t(Z_n + Z_{n-1} + \cdots + Z_1)}] = \mathbb{E}[e^{t(Z_{n-1} + \cdots + Z_1)} \cdot e^{tZ_n}]$$

Using the property $\mathbb{E}[AB] = \mathbb{E}[A \cdot \mathbb{E}[B|\text{information about } A]]$ and the fact that $e^{t(Z_{n-1}+\cdots+Z_1)}$ is determined by $X_0, \ldots, X_{n-1}$:

$$\mathbb{E}[e^{t(X_n - X_0)}] = \mathbb{E}[e^{t(Z_{n-1} + \cdots + Z_1)} \cdot \mathbb{E}[e^{tZ_n} | X_0, \ldots, X_{n-1}]]$$

Applying the bound:

$$\leq \mathbb{E}[e^{t(Z_{n-1} + \cdots + Z_1)}] e^{\frac{t^2 c_n^2}{2}}$$

Repeating this recursively:

$$\leq \mathbb{E}[e^{t(Z_{n-2} + \cdots + Z_1)}] \cdot e^{\frac{t^2 c_{n-1}^2}{2}} \cdot e^{\frac{t^2 c_n^2}{2}}$$

$$\leq e^{(t^2 \sum_{i=1}^n c_i^2)/2}$$

Thus:

$$\mathbb{E}[e^{t(X_n - X_0)}] \leq e^{\frac{\sum_{i=1}^n c_i^2}{2} \cdot t^2}.$$

Putting together with Markov's inequality:

$$\Pr[X_n - X_0 \geq \lambda] \leq \frac{\mathbb{E}[e^{t(X_n - X_0)}]}{e^{t\lambda}} \leq \frac{e^{t^2 \frac{\sum c_i^2}{2}}}{e^{t\lambda}} = e^{t^2 \left(\frac{\sum c_i^2}{2}\right) - t\lambda}$$

Choosing $t = \frac{\lambda}{\sum c_i^2}$ to minimize the exponent:

$$t^2 \left(\frac{\sum c_i^2}{2}\right) - t\lambda = \frac{\lambda^2}{(\sum c_i^2)^2} \frac{\sum c_i^2}{2} - \frac{\lambda^2}{\sum c_i^2} = \frac{\lambda^2}{2 \sum c_i^2} - \frac{\lambda^2}{\sum c_i^2} = -\frac{\lambda^2}{2 \sum c_i^2}.$$

$$\Pr[X_n - X_0 \geq \lambda] \leq e^{-\frac{\lambda^2}{2 \sum_{i=1}^n c_i^2}}.$$

□

Corollary 19.12. *If $c_i \leq C$ for all i and $\lambda = C \cdot \alpha\sqrt{n}$, then:*

$$\Pr[|X_n - X_0| \geq C \cdot \alpha\sqrt{n}] \leq 2e^{-\frac{\alpha^2}{2}}.$$

(The 2 comes from bounding both tails).

19.4 McDiarmid's Inequality and Application

Chernoff-Hoeffding bound suggests that if $X = X_1 + X_2 + \cdots + X_n$ where $X_1, \ldots, X_n$ are independent and in a bounded range, then X has concentration.

Now consider the setting where we have an arbitrary function $f : U_1 \times U_2 \times \cdots \times U_n \to \mathbb{R}$. In other words f is a function of n "variables," each of which has domain U_i.

Definition 19.13. $f : U_1 \times U_2 \times \cdots \times U_n \to \mathbb{R}$ is **c-Lipschitz** for some $c > 0$ if:

$$|f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_n)| \leq c$$

for any $y, z \in U_i$, and any fixed x_j. In other words, changing one coordinate does not change the value of the function by more than c in absolute value.

A more refined definition is that f is $(c_1, \ldots, c_n)$-**Lipschitz** if:

$$|f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_n)| \leq c_i$$

for all i, all $y, z \in U_i$, and all x_j.

Theorem 19.14 (McDiarmid's Inequality)**.** *Suppose f is $(c_1, \ldots, c_n)$-Lipschitz. Let $X_1, X_2, \ldots, X_n$ be independent random variables where $X_i \in U_i$. Then:*

$$\Pr[|f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)]| \geq \lambda] \leq 2e^{\frac{-\lambda^2}{2\sum_i c_i^2}}$$

Note that independence is required. In some settings it can be relaxed.

19.4.1 Proof Sketch (Reduction to Azuma's Inequality)

We reduce this to Azuma's inequality via the Doob martingale. The main thing to note is where independence is used.

Let $\overline{X}$ denote $(X_1, \ldots, X_n)$ and $Z_i = \mathbb{E}[f(\overline{X})|X_1, \ldots, X_i]$. $Z_0 = \mathbb{E}[f(\overline{X})]$.

Recall that $Z_0, \ldots, Z_n$ is a Doob martingale sequence. We wish to apply Azuma's inequality to $Z_0, \ldots, Z_n$. For this we need to bound $Z_i - Z_{i-1}$.

$$Z_i - Z_{i-1} = \mathbb{E}[f(\overline{X})|X_1, \ldots, X_i] - \mathbb{E}[f(\overline{X})|X_1, \ldots, X_{i-1}].$$

Since $X_1, \ldots, X_n$ are independent, $X_{i+1}, \ldots, X_n$ are independent of $X_1, \ldots, X_i$.

It can be shown (using c_i-Lipschitz property and independence):

$$|Z_i - Z_{i-1}| \leq \sup_{u_1, \ldots, u_{i-1}} |\mathbb{E}[f(u_1, \ldots, u_{i-1}, a, X_{i+1}, \ldots, X_n)] - \mathbb{E}[f(u_1, \ldots, u_{i-1}, b, X_{i+1}, \ldots, X_n)]| \leq c_i$$

where $a, b \in U_i$. (The intermediate step involves sup over $X_1 = u_1, \ldots, X_{i-1} = u_{i-1}$ of the difference of conditional expectations, which is bounded by c_i when applying the expectation over $X_{i+1}, \ldots, X_n$). We used c_i-Lipschitzness on f and independence of $X_{i+1}, \ldots, X_n$ from $X_1, \ldots, X_i$.

Now we can apply Azuma's inequality to $Z_0, \ldots, Z_n$ and conclude that

$$\Pr[Z_n - Z_0 > \lambda] \leq e^{-\frac{\lambda^2}{2\sum c_i^2}}.$$

We have $Z_0 = \mathbb{E}[f(\overline{X})]$ and $Z_n = f(\overline{X})$.

19.4.2 Application: Balls and Bins

We throw m balls into n bins independently. Let $X_1, \ldots, X_m$ be the random choices of the m balls. $X_i \in [n]$ for all $i \in [m]$. Define $f(X_1, X_2, \ldots, X_m)$ to be the number of empty bins. $\mathbb{E}[f(X_1, \ldots, X_m)]$ is easy to calculate and is equal to $n\left(1 - \frac{1}{n}\right)^m$.

We claim f is 1-Lipschitz (specifically, $c_i = 1$ for all i). This is easy to verify. Changing the assignment of one ball can change the number of empty bins by at most 1.

Thus we get concentration (using $\sum c_i^2 = \sum_{i=1}^m 1^2 = m$):

$$\Pr[|f(X_1, \ldots, X_m) - n\left(1 - \frac{1}{n}\right)^m| > \lambda] \leq 2e^{\frac{-\lambda^2}{2m}}.$$

Hence if $\lambda = \alpha\sqrt{m}$, the probability is $\leq 2e^{-\frac{\alpha^2}{2}}$.

19.4.3 Application: Chromatic number of Random Graphs

Consider Random graph $G(n,p)$. Let $\chi(G)$ be the chromatic number. For instance, it is known that $\mathbb{E}[\chi(G(n,\frac{1}{2}))] \approx \frac{n}{2\log_2 n}$. What about concentration?

We consider the vertex exposure martingale. Fix an ordering of vertices and let Y_i for vertex i denote the random vector of edges to vertices 1 to $i-1$. Let $f(Y_1,\ldots,Y_n)$ be the chromatic number of $G(n,p)$.

We claim f is 1-Lipschitz. Changing the edges incident to a single vertex can change the chromatic number by at most 1. Why? If the chromatic number is k with the original edges, then after changing the edges of one vertex, we can still color all other vertices with k colors and use at most one additional color for the changed vertex, so the chromatic number is at most $k+1$. Similarly, the chromatic number can decrease by at most 1.

Thus the chromatic number of $G(n,p)$ is concentrated around its mean.

Chapter 20

Swap Rounding for Spanning Trees

20.1 Review: Pipage Rounding

We previously saw pipage rounding as a way to convert a fractional solution $x_1, x_2, \ldots, x_n \in [0,1]^n$ such that $\sum_i x_i = k$ into a random integer solution $X_1, X_2, \ldots, X_n$ such that:

(i) $X_i \in \{0,1\}$ and $\mathbb{E}[X_i] = x_i$

(ii) $\sum_i X_i = k$

(iii) $X_1, X_2, \ldots, X_n$ are negatively correlated

20.1.1 Pipage Rounding Algorithm

while $\bar{x}$ is not fully integral **do**
    Let $x_i, x_j \in (0,1)$ both fractional
    Let $\epsilon = \min\{x_i, x_j, 1-x_i, 1-x_j\}$
    Toss a coin with probability $\frac{1}{2}$
    if coin is heads **then**
        $x_i \leftarrow x_i + \epsilon$
        $x_j \leftarrow x_j - \epsilon$
    else
        $x_i \leftarrow x_i - \epsilon$
        $x_j \leftarrow x_j + \epsilon$
Output $\bar{x}$

20.2 Swap Rounding

Today the goal is to show a related but different scheme called **swap rounding** that works for fractional points in any matroid polytope. It is easier to see the combinatorics of this than the pipage rounding. We will explain this via spanning trees since they are familiar.

Definition 20.1. Let $G = (V, E)$ be a connected graph with m edges (can be a multigraph). Let $E = \{e_1, \ldots, e_m\}$.

A vector $\bar{x} \in [0,1]^m$ is a **fractional spanning tree** of G if there exist spanning trees $T_1, T_2, \ldots, T_h$ of G and coefficients $\lambda_1, \lambda_2, \ldots, \lambda_h \in [0,1]$ such that:

(i) $\sum_{i=1}^{h} \lambda_i = 1$, and

(ii) for each edge e: $\bar{x}(e) = \sum_{i=1}^{h} \lambda_i \mathbb{1}_{T_i}(e)$

Remark 20.2. Note that $\bar{x}$ may correspond to different decompositions. Not necessarily unique.

20.2.1 Example

Consider a graph with edges forming a tree structure. Different spanning trees T_1, T_2 can be combined with coefficients to form a fractional spanning tree.

20.2.2 Rounding a Fractional Spanning Tree

Suppose we are given a fractional spanning tree $\bar{x}$ along with a decomposition as $\sum_{i=1}^{h} \lambda_i T_i$.

We want to round $\bar{x}$ into a random spanning tree $X_1, X_2, \ldots, X_m$.

A basic property we want is that $\forall i : \mathbb{E}[X_i] = x_i$.

This is easy to accomplish: Simply pick a random tree from $T_1, \ldots, T_h$ where the probability that T_i is picked is equal to λ_i.

However, $X_1, X_2, \ldots, X_m$ can be very correlated. We will now describe a way to do this in a different way. For this we will use a nice exchange property of spanning trees. This is more generally true of bases of any matroid.

Lemma 20.3 (Exchange Property)**.** *Let T_1 and T_2 be two spanning trees of a graph $G = (V, E)$. Suppose $E(T_1) \neq E(T_2)$. Then for any edge $e \in E(T_1) \setminus E(T_2)$, there is an edge $e' \in E(T_2) \setminus E(T_1)$ such that $T_1 \cup \{e'\} \setminus \{e\}$ and $T_2 \cup \{e\} \setminus \{e'\}$ are both spanning trees.*

Proof. Exercise. □

The idea in swap rounding is to use the exchange property to merge trees in a step-by-step way.

20.2.3 Merge Algorithm

function MERGE($\lambda_1, T_1, \lambda_2, T_2$) where $\lambda_1, \lambda_2 \in [0, 1]$
  while $E(T_1) \neq E(T_2)$ **do**
    Let $e \in E(T_1) \setminus E(T_2)$
    Find $e' \in E(T_2) \setminus E(T_1)$ such that $T_1 \cup \{e'\} \setminus \{e\}$ and $T_2 \cup \{e\} \setminus \{e'\}$ are spanning trees
    With probability $\frac{\lambda_1}{\lambda_1 + \lambda_2}$:
      $T_1 \leftarrow T_1 \cup \{e'\} \setminus \{e\}$
    Else:
      $T_2 \leftarrow T_2 \cup \{e\} \setminus \{e'\}$
  Output T_1 $(= T_2)$

Properties:

- Merge finishes in $n - 1$ steps
- Easy to implement with tree data structures

Swap rounding uses Merge to iteratively merge the trees one step at a time.

20.2.4 Swap Rounding Algorithm

1: **function** SwapRounding($\lambda_1, T_1, \ldots, \lambda_h, T_h$)
2: **if** $h = 1$ **then**
3: **return** T_1
4: **else**
5: $T \leftarrow$ Merge($\lambda_1, T_1, \lambda_2, T_2$)
6: **return** SwapRounding($\lambda_1 + \lambda_2, T, \lambda_3, T_3, \ldots, \lambda_h, T_h$)

The process can be visualized as:

$$\bar{x} = \sum_{i=1}^{h} \lambda_i T_i \to \sum_{i=1}^{h-1} \lambda_i' T_i' \to \cdots \to T$$

It is easy to prove by induction that this forms a martingale sequence. If $\bar{x}^{(t)}$ is the vector after t steps, then:

(i) $\mathbb{E}[\bar{x}^{(t+1)} \mid \bar{x}^{(t)}] = \bar{x}^{(t)}$

(ii) Only two coordinates of $\bar{x}^{(t)}$ change in each step

Using the above, one can show that if the final vector is $X_1, \ldots, X_m$, then $X_1, \ldots, X_m$ are negatively correlated.

20.2.5 Negative Correlation

Lemma 20.4 (Chawla–Mattheus–Vondrák)**.** *Let $\tau \in \mathbb{N}$ and let $X^t = (X_{1,t}, \ldots, X_{n,t})$ for $t \in \{0, \ldots, \tau\}$ be a non-negative vector-valued random process with initial distribution given by $X_{i,0} = x_i$ with probability 1 $\forall i \in [n]$, and satisfying the following properties:*

1. *$\mathbb{E}[X_{i,t+1} \mid X^t] = X_{i,t}$ for $t \in \{0, \ldots, \tau\}$ and $i \in [n]$.*
2. *X^t and X^{t+1} differ in at most two components for $t \in \{0, \ldots, \tau - 1\}$.*
3. *For $t \in \{0, \ldots, \tau\}$, if two components $i, j \in [n]$ change between X^t and X^{t+1}, then their sum is preserved: $X_{i,t+1} + X_{j,t+1} = X_{i,t} + X_{j,t}$.*

Then for any $t \in \{0, \ldots, \tau\}$, the components of X^t satisfy $\mathbb{E}\left[\prod_{i \in S} X_{i,t}\right] \leq \prod_{i \in S} x_i$ for all $S \subseteq [n]$.

Proof. We are interested in the quantity $Y_t = \prod_{i \in S} X_{i,t}$. At the beginning of the process, we have $\mathbb{E}[Y_0] = \prod_{i \in S} x_i$. The main claim is that for each t, we have $\mathbb{E}[Y_{t+1} \mid X^t] \leq Y_t$.

Let us condition on a particular configuration of variables at time t, $X^t = (X_{1,t}, \ldots, X_{n,t})$. We consider three cases:

- If no variable X_i, $i \in S$, is modified in step t, we have $Y_{t+1} = \prod_{i \in S} X_{i,t+1} = \prod_{i \in S} X_{i,t} = Y_t$.
- If exactly one variable X_i, $i \in S$, is modified in step t, then by property 1 of the lemma:

$$\mathbb{E}[Y_{t+1} \mid X^t] = \mathbb{E}[X_{i,t+1} \mid X^t] \cdot \prod_{j \in S \setminus \{i\}} X_{j,t} = \prod_{j \in S} X_{j,t} = Y_t.$$

- If two variables X_i, X_j, $i, j \in S$, are modified in step t, we use the property that their sum is preserved: $X_{i,t+1} + X_{j,t+1} = X_{i,t} + X_{j,t}$. This also implies that

$$\mathbb{E}[(X_{i,t+1} + X_{j,t+1})^2 \mid X^t] = (X_{i,t} + X_{j,t})^2. \tag{20.1}$$

  On the other hand, the value of each variable is preserved in expectation. Applying this to their difference, we get $\mathbb{E}[X_{i,t+1} - X_{j,t+1} \mid X^t] = X_{i,t} - X_{j,t}$. Since $\mathbb{E}[Z^2] \geq (\mathbb{E}[Z])^2$ holds for any random variable, we get

$$\mathbb{E}[(X_{i,t+1} - X_{j,t+1})^2 \mid X^t] \geq (X_{i,t} - X_{j,t})^2. \tag{20.2}$$

  Combining (1) and (2), and using the formula $XY = \frac{1}{4}((X+Y)^2 - (X-Y)^2)$, we get

$$\mathbb{E}[X_{i,t+1}X_{j,t+1} \mid X^t] \leq X_{i,t}X_{j,t}.$$

  Therefore,

$$\mathbb{E}[Y_{t+1} \mid X^t] = \mathbb{E}[X_{i,t+1}X_{j,t+1} \mid X^t] \cdot \prod_{k \in S \setminus \{i,j\}} X_{k,t} \leq \prod_{k \in S} X_{k,t} = Y_t,$$

  as claimed.

By taking expectation over all configurations X^t we obtain $\mathbb{E}[Y_{t+1}] \leq \mathbb{E}[Y_t]$. Consequently,

$$\mathbb{E}\left[\prod_{i \in S} X_{i,t}\right] = \mathbb{E}[Y_t] \leq \mathbb{E}[Y_{t-1}] \leq \cdots \leq \mathbb{E}[Y_0] = \prod_{i \in S} x_i,$$

as claimed by the lemma. □

20.3 Continuous Extensions of Set Functions

Suppose we have a set function $f : 2^N \to \mathbb{R}$ that is real valued. For each subset $S \subseteq N$, $f(S)$ is the value of S.

Many times it is useful to extend f to fractional values:

$$f(x_1, \ldots, x_n) \text{ where } x_i \in [0,1],\ i \in [n]$$

How should we define it? We know $f(x_1, \ldots, x_n)$ if $\bar{x} \in \{0,1\}^n$.

20.3.1 Probability Distribution View

We can use a probability distribution view for interpolation. Given $\bar{x} \in [0,1]^n$, there are multiple ways of expressing $\bar{x}$ as a convex combination of the corners $\{0,1\}^n$.

Let

$$\mathcal{P}(\bar{x}) = \left\{ \alpha : 2^N \to [0,1] \mid \sum_{S \subseteq N} \alpha_S \mathbb{1}_S(i) = x_i \,\forall i \in N,\ \sum_{S \subseteq N} \alpha_S = 1 \right\}$$

This is the space of all convex combinations of sets that can express $\bar{x}$. We can think of α as a probability distribution (over subsets).

20.3.2 Convex Closure

We obtain one extension, called the **convex closure**, by choosing the distribution that minimizes the expected value of f:

$$f^-(\bar{x}) = \min \left\{ \sum_{S \subseteq N} \alpha_S f(S) \mid \sum_{S \subseteq N} \alpha_S = 1, \sum_{S \subseteq N} \alpha_S \mathbb{1}_S(i) = x_i \, \forall i \in N, \, \alpha_S \geq 0 \, \forall S \subseteq N \right\}$$

Claim: $f^-(\bar{x})$ is a convex function for any set function f.

Proof. Exercise. □

20.3.3 Concave Closure

Similarly, we can define the **concave closure** f^+ if we change min to max in the above LP.

20.3.4 Other Extensions

Two other extensions are the:

- Multilinear extension
- Lovász extension

Chapter 21

Lovász Local Lemma

21.1 Lovász Local Lemma

LLL is a powerful tool in the probabilistic method and has found several highly non-trivial applications in algorithms.

In the probabilistic method we prove the existence of some object by running a probabilistic experiment and arguing that the object property holds with non-zero probability.

21.1.1 First Moment Method

We use expectation analysis. As an example, we showed that in any graph $G = (V, E)$ there exists a max cut of value $\frac{|E|}{2}$ by picking a random cut whose expectation is $\frac{|E|}{2}$.

21.1.2 Second Moment Method

Here we use variance analysis plus something like the Chebyshev bound.

Example 21.1. Let $G(n, p)$ be a random graph on n vertices where each edge is chosen independently with probability p. At what value of p will $G(n, p)$ have a clique of size 4?

Clearly when $p = 0$, the graph will be very sparse and there will not be a 4-clique. When $p = 1$ there will be a 4-clique with high probability since the graph becomes dense.

Turns out that $p = n^{-2/3}$ is the threshold.

To see one direction we use first moment method. Let X be the number of 4-cliques.

$$\mathbb{E}[X] = \binom{n}{4} p^6$$

since if we fix a set of 4 vertices, it will be a clique iff all six edges are chosen.

If $p \leq \frac{c}{n^{2/3}}$ for sufficiently small constant c, then $\mathbb{E}[X] \leq 0.1$.

For a non-negative integer random variable:

$$\Pr[X \geq 1] \leq \mathbb{E}[X]$$

So $\Pr[X = 0] \geq 1 - \mathbb{E}[X] \geq 0.9$.

We would like to compute the variance of X. Note that $X = \sum_{S \in \mathcal{S}} X_S$ where S ranges over all $\binom{n}{4}$ subsets of four vertices and X_S is an indicator for S being a clique. $\Pr[X_S = 1] = p^6$.

If $S_i, S_j \in \mathcal{S}$, then X_{S_i} and X_{S_j} are independent if S_i and S_j do not share any edges. Otherwise they are dependent.

To estimate $\text{Var}(X)$ we write:

$$\mathbb{E}[X^2] = \mathbb{E}\left[\left(\sum_S X_S\right)^2\right] = \sum_S \mathbb{E}[X_S^2] + 2\sum_{S\sim S'} \mathbb{E}[X_S \cdot X_{S'}]$$

where $S \sim S'$ means S and S' are dependent and $S \perp S'$ means independent.

Suppose we pretend all S and S' are independent. Then there are roughly n^8 such pairs and in that case:

$$\text{Var}(X) = \sum_{S\in\mathcal{S}} \text{Var}(X_S) = \binom{n}{4} p^6(1-p^6)$$

Then by Chebyshev:

$$\Pr[X = 0] \leq \Pr[|X - \mathbb{E}[X]| \geq \mathbb{E}[X]] \leq \frac{\text{Var}(X)}{(\mathbb{E}[X])^2} \leq \frac{n^4p^6}{(n^4p^6)^2} = \frac{1}{n^4p^6}$$

One can check that if $p \geq \frac{C}{n^{2/3}}$ for sufficiently large C, then $\Pr[X = 0] \leq 0.1$.

However, X_S and $X_{S'}$ are not independent for all S_i and $S_j \in \mathcal{S}$. But if we calculate $\text{Var}(X)$ more carefully we see that:

$$\begin{aligned}\mathbb{E}[X^2] &= \sum_{S_i\in\mathcal{S}} \mathbb{E}[X_{S_i}^2] + \sum_{S_i\sim S_j} \mathbb{E}[X_{S_i}X_{S_j}] \\ &= \sum_{S\in\mathcal{S}} \mathbb{E}[X_S] + \sum_{S\perp S'} \mathbb{E}[X_S]\mathbb{E}[X_{S'}]\end{aligned}$$

There are $\binom{n}{4}^2 = O(n^8)$ total pairs. How many are not independent? $S \sim S'$ if they share at least one edge. But the number of dependent pairs is $O(n^7)$ while the number of independent pairs is $\Theta(n^8)$, which is close to all pairs. So $\text{Var}(X)$ still behaves as if not all pairs are independent and one can show that $p \geq \frac{C}{n^{2/3}}$ for sufficiently large constant ensures $G(n,p)$ has a 4-clique with probability ≥ 0.9.

21.1.3 Concentration plus Union Bound

We saw several examples of using concentration bounds plus union bound. The general strategy is to show that for some events $A_1, A_2, \ldots, A_n$:

$$\Pr[A_i] \leq \frac{1}{n}$$

where A_i is a bad event.

Then by the union bound:

$$\Pr[A_1 \cup A_2 \cup \cdots \cup A_n] \leq n \cdot \frac{1}{n} = 1$$

$$\Pr[\overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_n}] > 0$$

Thus we have all good events happening with non-zero probability, and if we can express the property we want in those terms then we are done.

Example 21.2 (Routing paths)**.** We saw that we can convert fractional solution to integral solution by randomized rounding. We get $O(\log n/\log\log n)$ congestion by using Chernoff bounds on each single edge and then union bound over all edges.

21.1.4 Local Phenomena

There are many situations where we cannot use union bound because the individual bad event probability is not that small.

If A_i are independent this does not matter because we have:

$$\Pr[\overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_n}] = \prod_{i=1}^{n} (1 - \Pr[A_i])$$

and hence all we need is $\Pr[A_i] < 1$.

However, independence is rarely possible in complex events.

LLL considers a local setting where the events $A_1, \ldots, A_n$ are not completely independent but there is some limited dependence.

How can one capture such a scenario? For that we use a **dependency graph** on the events. The vertices are the events and we have an edge (A_i, A_j) if A_i and A_j are dependent. No edge means that A_i and A_j are conditionally independent. That is, $\Pr[A_i \mid A_j] = \Pr[A_i]$.

Note that A_i is conditionally independent of all events that it has no edges to. We define it formally:

Definition 21.3. An event A is conditionally independent with respect to $B_1, B_2, \ldots, B_k$ if:

$$\forall S \subseteq \{1, 2, \ldots, k\} : \Pr\left[A \mid \bigcap_{i \in S} B_i\right] = \Pr[A]$$

When can we easily identify conditional independence?

Claim 21.4. *Suppose $X_1, X_2, \ldots, X_n$ are independent random variables. Suppose event A_i is completely determined by a subset $S_i \subseteq \{X_1, X_2, \ldots, X_n\}$. If $S_i \cap S_j = \emptyset$ for $j \in \{j_1, j_2, \ldots, j_k\}$ then A_i is mutually independent of $A_{j_1}, A_{j_2}, \ldots, A_{j_k}$.*

With this in place we state the Symmetric version of the LLL:

Theorem 21.5 (Symmetric LLL)**.** *Suppose $A_1, A_2, \ldots, A_n$ are events in an underlying probability space and let d be the max degree of the dependency graph and $\Pr[A_i] \leq p$ for all i. Then:*

(i) If $pd \leq \frac{1}{4}$, then $\Pr\left[\bigcap_{i=1}^{n} \overline{A_i}\right] \geq (1-2p)^n > 0$.

(ii) If $p(d+1) \leq \frac{1}{e}$, then $\Pr\left[\bigcap_{i=1}^{n} \overline{A_i}\right] \geq \left(1 - \frac{1}{d+1}\right)^n > 0$.

A more general version of the LLL called asymmetric or lopsided LLL is the following:

Theorem 21.6 (LLL)**.** *Suppose $A_1, \ldots, A_n$ are events in a probability space and let H be the dependency graph. Suppose there exist numbers $x_1, x_2, \ldots, x_n \in (0, 1)$ such that:*

$$\Pr[A_i] \leq x_i \prod_{j \in N(i)} (1 - x_j)$$

Then:

$$\Pr\left[\bigcap_{i} \overline{A_i}\right] \geq \prod_{i=1}^{n} (1 - x_i)$$

Here $N(i)$ is the set of dependent neighbors of A_i.

21.1.5 Proof of Symmetric Version

The heart of the proof is the following lemma:

Lemma 21.7. *For any $S \subseteq \{1, 2, \ldots, n\}$ and $i \notin S$:*

$$\Pr\left[A_i \mid \bigcap_{j \in S} \overline{A_j}\right] \leq 2p$$

Assuming the lemma above, the symmetric LLL follows as below:

$$\begin{aligned}
\Pr[\overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_n}] &= \Pr[\overline{A_1}] \cdot \Pr[\overline{A_2} \mid \overline{A_1}] \cdot \Pr[\overline{A_3} \mid \overline{A_1} \cap \overline{A_2}] \cdots \\
&\quad \cdot \Pr[\overline{A_n} \mid \overline{A_1} \cap \overline{A_2} \cap \cdots \cap \overline{A_{n-1}}] \\
&= (1 - \Pr[A_1]) \cdot (1 - \Pr[A_2 \mid \overline{A_1}]) \\
&\quad \cdot (1 - \Pr[A_3 \mid \overline{A_1} \cap \overline{A_2}]) \cdots \\
&\geq (1 - 2p)^n > 0
\end{aligned}$$

Now we prove the lemma by induction on $|S|$.

Suppose $|S| = 0$. Then $\Pr[A_i] \leq p \leq 2p$.

Assume true for $|S| = k$. Consider $|S| = k + 1$, $i \notin S$. Want to prove $\Pr[A_i \mid \bigcap_{j \in S} \overline{A_j}] \leq 2p$.

Let $S_{dep} = S \cap N(i)$ be the set of events in S that A_i is dependent on, and $S_{ind} = S \setminus S_{dep}$. Thus $S = S_{dep} \cup S_{ind}$.

If $|S_{ind}| = k + 1$ then $S_{dep} = \emptyset$ and:

$$\Pr[A_i \mid \bigcap_{j \in S} \overline{A_j}] = \Pr[A_i] \leq p \leq 2p$$

Thus we now consider $|S_{ind}| \leq k$.

We now use conditional probability or Bayes theorem. For events X, Y:

$$\Pr[X \mid Y] = \frac{\Pr[X \cap Y]}{\Pr[Y]}$$

and for events X, Y, Z:

$$\Pr[X \mid Y \cap Z] = \frac{\Pr[X \cap Y \mid Z]}{\Pr[Y \mid Z]}$$

Applying with $X = A_i$, $Y = \bigcap_{j \in S_{dep}} \overline{A_j}$ and $Z = \bigcap_{j \in S_{ind}} \overline{A_j}$, we have:

$$\Pr\left[A_i \mid \bigcap_{j \in S} \overline{A_j}\right] = \frac{\Pr\left[A_i \cap \left(\bigcap_{j \in S_{dep}} \overline{A_j}\right) \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right]}{\Pr\left[\bigcap_{j \in S_{dep}} \overline{A_j} \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right]}$$

Consider denominator. Via union bound:

$$\begin{aligned}
\Pr\left[\bigcap_{j \in S_{dep}} \overline{A_j} \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right] &= 1 - \Pr\left[\bigcup_{j \in S_{dep}} A_j \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right] \\
&\geq 1 - \sum_{j \in S_{dep}} \Pr\left[A_j \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right]
\end{aligned}$$

By induction hypothesis, since $|S_{ind}| \leq k$:

$$\Pr\left[A_j \mid \bigcap_{\ell \in S_{ind}} \overline{A_\ell}\right] \leq 2p$$

and $|S_{dep}| \leq d$.

Hence:

$$\Pr\left[\bigcap_{j \in S_{dep}} \overline{A_j} \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right] \geq 1 - 2pd \geq 1 - 2 \cdot \frac{1}{4} = \frac{1}{2}$$

where we used $pd \leq 1/4$.

Numerator is:

$$\begin{aligned}\Pr\left[A_i \cap \left(\bigcap_{j \in S_{dep}} \overline{A_j}\right) \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right] &\leq \Pr\left[A_i \mid \bigcap_{j \in S_{ind}} \overline{A_j}\right] \\ &= \Pr[A_i] \leq p\end{aligned}$$

Hence:

$$\Pr\left[A_i \mid \bigcap_{j \in S} \overline{A_j}\right] \leq \frac{p}{1 - 2pd} \leq 2p$$

21.2 Application of LLL

21.2.1 k-SAT

Recall that a k-SAT formula is a Boolean formula in CNF form with each clause having exactly k literals over distinct variables.

Theorem 21.8. *Let Φ be a k-SAT formula in which each variable occurs in at most $\frac{2^{k-2}}{k}$ clauses. Then Φ is satisfiable.*

Note that there is no limitation on number of variables or clauses.

Example 21.9. If $k = 10$ then it is requiring each variable to be in at most $\frac{2^8}{10} \approx 26$ clauses.

The theorem may not be interesting from a SAT perspective but is mainly to showcase the power of LLL and a setting in which it applies.

We prove this by considering a random assignment to the variables. Let A_i be the event that a clause C_i is not satisfiable (bad event), then $\Pr[A_i] = \frac{1}{2^k}$.

What does A_i depend on? C_i has k variables. Each variable that is in C_i is in at most $\frac{2^{k-2}}{k} - 1$ other clauses. So C_i shares a variable with at most $k \cdot \left(\frac{2^{k-2}}{k} - 1\right) \leq 2^{k-2}$ other clauses.

If C_i and C_j do not share variables then A_i and A_j are mutually independent.

Thus we can apply symmetric LLL (condition (i)) with $p = \frac{1}{2^k}$ and $d \leq 2^{k-2}$.

Since $pd \leq \frac{2^{k-2}}{2^k} = \frac{1}{4}$:

$$\Pr\left[\bigcap_i \overline{A_i}\right] > 0$$

where m is the number of clauses.

Therefore $\Pr[\Phi \text{ is satisfiable}] > 0$.

21.2.2 Routing for Congestion Minimization

Recall that we saw the congestion minimization problem:

- $G = (V, E)$ directed graph
- $(s_1, t_1), \ldots, (s_k, t_k)$: k pairs that we want to connect by paths $P_1, P_2, \ldots, P_k$
- Minimize $\max_{e \in E} |\{i : e \in P_i\}|$ (congestion on e)

We used an LP relaxation and found a fractional routing that minimizes max fractional congestion.

A fractional routing for a pair (s_i, t_i) is a probability distribution over paths $p \in \mathcal{P}_i$ where $\mathcal{P}_i$ is the set of all s_i-t_i paths. We let x_p, $p \in \mathcal{P}$, be the amount of flow routed along p. We have $\sum_{p \in \mathcal{P}_i} x_p = 1$.

Suppose:

$$\sum_{i: p \in \mathcal{P}_i} \sum_{p \in \mathcal{P}_i : e \in p} x_p \leq 1$$

i.e., the max fractional congestion is at most 1.

Randomized rounding picks a path for each i independently according to the distribution $\{x_p : p \in \mathcal{P}_i\}$.

Then we used Chernoff bounds to show that $\Pr\left[\ell_e \geq c\frac{\log m}{\log \log m}\right]$ is at most $\frac{1}{m^2}$ for some sufficiently large constant c. Here ℓ_e is the load on e (the number of paths that use e).

Then via the union bound we see that with high probability $\max_{e \in E} \ell_e = O\Big(\frac{\log m}{\log \log m}\Big)$.

Better bound when paths are short

Now we will prove a better bound when paths are short.

Suppose $x_p = 0$ or $|p| \leq h$ where h is some parameter. In many applications h is a small constant independent of m, n. This implies locality because even if the graph is large, paths along which flow is routed are short.

Theorem 21.10. *There exists an integral routing where max congestion is* $O\Big(\frac{\log h}{\log \log h}\Big)$.

Note: the bound does not depend on graph size.

In order to apply LLL we do some preprocessing. By discretization tricks we will assume that all x_p values that are non-zero have same value $\frac{1}{L}$ for some L. We may duplicate paths to achieve this. Thus each pair now has exactly L paths.

Then we do randomized rounding as before but with the discretized paths. So we pick one of the L paths for each pair.

How can we apply LLL here? Need to set up the events carefully.

Let C be the threshold of congestion we want to avoid. For edge e, let S_e be the set of all paths that use e.

Define $A_{e,S}$ for $e \in E$, $S \subseteq S_e$, $|S| = C$ to be the event that S is the set of paths chosen.

Claim 21.11. *If S contains two paths from path collection of same pair, then* $\Pr[A_{e,S}] = 0$. *Otherwise it is equal to* $\frac{1}{L^C}$.

We use notation $S = \{(i_1, j_1), (i_2, j_2), \ldots, (i_C, j_C)\}$ where $i_\ell \in \{1, 2, \ldots, K\}$ indicates the pair and $j_\ell \in \{1, 2, \ldots, L\}$ to denote the index of the path in the L paths for the pair. We order the paths in some fashion for each pair. We let P_i^j denote the j-th path for pair i.

Dependencies: Fix two events $A_{e,S}$ and $A_{e',S'}$ where S has paths from distinct pairs and similarly S'.

Suppose $S = \{(i_1, j_1), (i_2, j_2), \ldots, (i_C, j_C)\}$ and $S' = \{(i'_1, j'_1), (i'_2, j'_2), \ldots, (i'_C, j'_C)\}$.

If $\{i_1, i_2, \ldots, i_C\} \cap \{i'_1, i'_2, \ldots, i'_C\} = \emptyset$, i.e., the pairs don't overlap, then $A_{e,S}$ is independent of $A_{e',S'}$.

We need to understand how many other events does $A_{e,S}$ depend on. Suppose $e' \in E$, $S' \subseteq S_{e'}$, and $|S'| = C$ with $\{i_1, i_2, \ldots, i_C\} \cap \{i'_1, i'_2, \ldots, i'_C\} \neq \emptyset$. Say $i_1 = i'_1$.

How many choices of $e', \{i'_2, \ldots, i'_C\}$ and $\{j'_1, j'_2, \ldots, j'_C\}$ do we have?

L choices for j'_1. Fix one such j'_1.

Then we have path $P_{i_1}^{j'_1}$ and this path has at most h edges.

So e' has h choices. For each such edge, at most L paths use the edge (since total flow on each edge ≤ 1). And we can choose any $C - 1$ paths from those L.

Thus for fixed choice of j'_1 there are $h \cdot \binom{L}{C-1}$ choices.

Hence total is $L \cdot h \cdot \binom{L}{C-1}$.

There are C choices of pair overlap between S and S'.

Hence the neighborhood size of $A_{e,S}$ in the dependency graph is at most $C \cdot L \cdot h \cdot \binom{L}{C-1}$.

To apply LLL we have $\Pr[A_{e,S}] = \frac{1}{L^C} = p$ and $d \leq C \cdot L \cdot h \cdot \binom{L}{C-1} \leq \frac{C \cdot h \cdot L^C}{(C-1)!}$.

Hence:

$$pd \leq \frac{1}{L^C} \cdot \frac{C \cdot h \cdot L^C}{(C-1)!} = \frac{Ch}{(C-1)!}$$

To ensure $pd \leq \frac{1}{4}$ we need $\frac{Ch}{(C-1)!} \leq \frac{1}{4}$. By Stirling-type estimates, $C = \Omega\left(\frac{\log h}{\log \log h}\right)$ suffices.

Note that if no bad event $A_{e,S}$ happens, then congestion is at most $C - 1$.

Chapter 22

VC Dimension and Geometric Range Spaces

22.1 Set Systems and Range Spaces

Set systems arise in many applications. A set system consists of a pair (P, R) where P is a set and R is a collection of subsets of P. When P is finite, we have a finite set system. Sometimes finite set systems are thought of as hypergraphs with P as vertices and each $r \in R$ as a hyperedge.

Here we will be concerned with set systems that arise in geometric settings where P is typically all of $\mathbb{R}^d$ for some dimension d, or a finite subset of $\mathbb{R}^d$. We also consider R to be sets that are induced by structured shapes that intersect with P.

22.1.1 Examples of Shapes

Examples of shapes include:

- Intervals
- Disks
- Half-spaces
- Convex polygons

In the geometric setting, (P, R) are often called **range spaces** and each $r \in R$ is called a **range**. Typically we associate r with a shape such as an interval I, and then

$$r = I \cap P$$

Geometric range spaces have additional properties that lead to a number of applications, and the notion of VC dimension, ε-sample theorem, ε-net theorem and others have had striking influence on many areas, in particular machine learning where the notion of VC dimension arose.

22.2 VC Dimension

VC dimension of a set system is one important measure of the complexity of a set system.

Definition 22.1. Let (P, R) be a range space. A finite subset $Q \subseteq P$ is said to be **shattered** by R if $\{Q \cap r : r \in R\} = 2^Q$. In other words, $R|_Q = 2^Q$, the powerset of Q.

Example 22.2. Suppose P is the real line and R is the collection of all closed intervals.

It can be seen from the figure that $Q = \{a, b\}$ can be shattered by the collection of intervals:

- $\emptyset$
- $\{a\}$
- $\{b\}$
- $\{a, b\}$

Definition 22.3. The **VC dimension** of a set system (P, R) is the maximum cardinality of a finite set $Q \subseteq P$ such that Q is shattered by R.

Example 22.4. Let $P = \mathbb{R}$ and R be the collection of intervals. Then VC-dim $= 2$.

Why? We saw that it is at least 2. Can it be ≥ 3?

Suppose $Q = \{a, b, c\}$ where $a < b < c$. Can we get the set $\{a, c\}$ as an intersection of $\{a, b, c\}$ and an interval? No.

Example 22.5. $P = \mathbb{R}^2$ (the 2D plane) and $R = \{D : D \text{ is a closed disk in the plane}\}$.

VC dimension is 3. Three points can be shattered but not 4.

Example 22.6. $P = \mathbb{R}^d$ and $R =$ set of half-spaces.

Recall: a half-space is defined by an inequality $\sum a_i x_i \leq b$ for some $a_1, a_2, \ldots, a_d, b \in \mathbb{R}$.

Claim: VC-dim $= d + 1$.

It is easy to see that VC-dim $\geq d + 1$. Take the $d + 1$ points: $(0, 0, \ldots, 0)$ and $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0)$, …, $(0, 0, \ldots, 0, 1)$.

This set can be shattered. Why?

However, $d + 2$ points cannot be shattered, and this follows from Radon's theorem.

Theorem 22.7 (Radon's Theorem)**.** *Let Q be a set of $d + 2$ points in $\mathbb{R}^d$. Then one can partition Q into S_1 and S_2 such that*

$$\textit{convex-hull}(S_1) \cap \textit{convex-hull}(S_2) \neq \emptyset$$

The preceding theorem shows Q cannot be shattered by half-spaces.

22.3 Sauer's Lemma

Now that we have seen the definition of VC dimension, we state and prove a key technical lemma about set systems with bounded VC dimension.

Theorem 22.8 (Sauer's Lemma)**.** *Suppose a set system (P, R) has VC dimension at most d. Let $Q \subseteq P$ be a finite set of cardinality n. Then*

$$|R|_Q| \leq \sum_{i=0}^{d} \binom{n}{i} \leq n^d$$

Proof. By induction on n.

If $n = 0$, it is trivial.

Let Q be a set of n points, $n > 0$. We can restrict attention to $R|_Q = \{r \cap Q : r \in R\}$. Hence we can work with a finite range space. Now all we need to do is count $|R|_Q|$.

Fix some $p \in Q$.

Let $R_1 = \{r \setminus \{p\} : r \in R\}$ be the set of all ranges obtained by removing p from the original ranges.

Suppose r is such that $p \in r$ and also $r \setminus \{p\} \in R$. Then both r and $r \setminus \{p\}$ project to the same range in R_1. So to count $|R|_Q|$ we create a separate range space.

Let $R_2 = \{r \setminus \{p\} : r \cup \{p\} \in R \text{ and } r \setminus \{p\} \in R\}$.

From this explanation we have:

Claim 22.9.

$$|R|_Q| = |R_1| + |R_2|$$

Now we consider the two range spaces $(Q \setminus \{p\}, R_1)$ and $(Q \setminus \{p\}, R_2)$.

Claim 22.10. *VC-dim*$(Q \setminus \{p\}, R_1) \leq d$.

Proof. Removing a point does not increase VC-dim. □

Claim 22.11. *VC-dim of* $(Q \setminus \{p\}, R_2) \leq d - 1$.

Proof. If $Q' \subseteq Q \setminus \{p\}$ is shattered by R_2, then since every range $r \in R_2$ satisfies the property that $r \cup \{p\}$ and $r \setminus \{p\} \in R$, we would have $Q' \cup \{p\}$ is shattered by R. Thus $|Q'| \leq d - 1$. □

Now by induction:

$$|R_1| \leq \sum_{i=0}^{d} \binom{n-1}{i}$$

and

$$|R_2| \leq \sum_{i=0}^{d-1} \binom{n-1}{i}$$

Thus,

$$\begin{aligned} |R|_Q| &= |R_1| + |R_2| \\ &\leq \sum_{i=0}^{d} \binom{n-1}{i} + \sum_{i=0}^{d-1} \binom{n-1}{i} \\ &= \sum_{i=0}^{d} \binom{n}{i} \end{aligned}$$

□

22.3.1 Shattering Dimension

In many settings, the only way VC-dim is used is via the bound given by Sauer's lemma. So it makes sense to define the following:

Definition 22.12. The **shattering dimension** of a range space (P, R) is d if $\forall Q \subseteq P$ with $|Q| = n$, the size of $R|_Q \leq O(n^d)$.

VC-dim$(P, R) = d \Rightarrow$ shattering-dim$(P, R) = d$.

Converse is also true with weaker parameters:

Shattering-dim$(P, R) = d \Rightarrow$ VC-dim$(P, R) = O(d \log d)$.

22.3.2 Closure Properties

One important aspect of VC-dim is a kind of closure when combining range spaces.

Theorem 22.13. *Suppose (P, R_1) and (P, R_2) are range spaces with VC-dim d_1 and d_2 respectively. Then:*

- *VC-dim of (P, R) where $R = \{r_1 \cap r_2 : r_1 \in R_1, r_2 \in R_2\}$ is $O(d_1 + d_2)$.*
- *Similarly for (P, R) where $R = \{r_1 \cup r_2 : r_1 \in R_1, r_2 \in R_2\}$.*

22.4 ε-Sample and ε-Net Theorem

We now discuss two theorems about how a random sample of a set from a set system (P, R) can approximate it.

For the following discussion, it is useful to think of P as a finite set. Some of the concepts can be lifted to infinite sets with appropriate generalizations.

For a given system (P, R), let $\mu(r)$ denote the measure of r, i.e., $\mu(r) = \frac{|r|}{|P|}$.

Suppose we take a small random sample Q from P. Does Q preserve the measure of all $r \in R$?

For this, define $\mu_Q(r) = \frac{|r \cap Q|}{|Q|}$.

Clearly a small sample cannot touch all ranges, so we need to allow some additive error.

Definition 22.14. A subset $Q \subseteq P$ is an **ε-sample** for (P, R) if

$$|\mu_Q(r) - \mu(r)| \leq \varepsilon \quad \forall r \in R$$

A related notion is the following:

Definition 22.15. Q is an **ε-net** for (P, R) if $|Q \cap r| \geq 1$ for all $r \in R$ where $\mu(r) \geq \varepsilon$.

Note that an ε-sample is automatically an ε-net.

Theorem 22.16. *Let (P, R) be a range space with VC dimension d. Let $\ell \geq \frac{c}{\varepsilon^2}\left(d \log \frac{d}{\varepsilon} + \log \frac{1}{\delta}\right)$.*
Then a random sample of ℓ points with repetition from P is an ε-sample with probability $\geq 1 - \delta$.

Note: The sample size does not depend on $|P|$. Could be infinite.

Note: The theorem relies only on the growth rate of the number of distinct ranges of a given size that follows from Sauer's lemma. Hence the proof is not so tied to VC dimension itself.

A stronger bound is known for ε-nets:

Theorem 22.17. *Let (P, R) be a range space with VC-dim $\leq d$. Let $\ell \geq \frac{c}{\varepsilon}\left(d \log \frac{d}{\varepsilon} + \log \frac{1}{\delta}\right)$.*
Then a random sample of ℓ points with repetition from P is an ε-net with probability $\geq 1 - \delta$.

22.5 Proof of ε-Sample Theorem

It is a clever argument using a double sample argument.

First we recall the additive Chernoff bound:

Theorem 22.18 (Chernoff Bound)**.** *Let $X_1, X_2, \ldots, X_n \in \{0, 1\}$ and independent. Let $Y = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[Y]$. Then*

$$\Pr[|Y - \mu| \geq t] \leq 2e^{-\frac{2t^2}{n}}$$

Equivalently, $\Pr[|Y - \mu| \geq \varepsilon n] \leq 2e^{-2\varepsilon^2 n}$.

To see how to use the above theorem in our setting: Fix a range r with $\mu(r) = p \in [0,1]$. Suppose we take an ℓ-sample Q with repetition. What is $\mathbb{E}[\mu_Q(r)]$?

Let X_i be indicator random variable for sample i being in r.

Let $Y = \sum_{i=1}^{\ell} X_i$. Then $\mu_Q(r) = \frac{Y}{\ell}$ and $\mathbb{E}[Y] = \mu(r) \cdot \ell$.

Hence $\mathbb{E}[\mu_Q(r)] = \mu(r)$.

Therefore, $\Pr[|\mu_Q(r) - \mu(r)| \geq \varepsilon] \leq 2e^{-2\varepsilon^2 \ell}$.

22.5.1 Full Proof of ε-Sample Theorem

It is a clever argument using a double sample argument.

Let Q be a sample of size ℓ. Let Q' be an independent sample also of size ℓ.

Let A be the event that $\exists$ some range r such that $|\mu_Q(r) - \mu(r)| > \varepsilon$. Here $\mu(r) = \frac{|r|}{|P|}$ and $\mu_Q(r) = \frac{|r \cap Q|}{|Q|}$ for short.

Let B be the event that $\exists r$ such that $|\mu_Q(r) - \mu_{Q'}(r)| > \varepsilon/2$.

Claim 22.19. $\Pr[A] \leq 2\Pr[B]$.

Proof. Let D be the event that $\exists r$ such that $|\mu_Q(r) - \mu(r)| > \varepsilon$ and $|\mu_{Q'}(r) - \mu(r)| \leq \varepsilon/2$.

We have $\Pr[B] \geq \Pr[D] \geq \Pr[D|A] \cdot \Pr[A]$.

We claim that $\Pr[D|A] \geq \frac{1}{2}$, which would imply that $\Pr[A] \leq 2\Pr[B]$.

To see this, suppose event A happens: $\exists r$ such that $|\mu_Q(r) - \mu(r)| > \varepsilon$. For this r, via the additive Chernoff bound,

$$\Pr[|\mu_{Q'}(r) - \mu(r)| > \varepsilon/2] \leq 2e^{-2(\varepsilon/2)^2 \ell} \leq \frac{1}{2}$$

This is because r is fixed and the sample is big enough and Q' is independent of Q.

So with probability $\geq 1/2$, we have $|\mu_{Q'}(r) - \mu(r)| \leq \varepsilon/2$.

If $|\mu_Q(r) - \mu(r)| > \varepsilon$ and $|\mu_{Q'}(r) - \mu(r)| \leq \varepsilon/2$, then by triangle inequality:

$$\begin{aligned} |\mu_Q(r) - \mu_{Q'}(r)| &\geq |\mu_Q(r) - \mu(r)| - |\mu_{Q'}(r) - \mu(r)| \\ &> \varepsilon - \varepsilon/2 = \varepsilon/2 \end{aligned}$$

Thus event B occurs. Hence $\Pr[D|A] \geq \frac{1}{2}$. □

Thus we can focus on event B, which is $\Pr[\exists r : |\mu_Q(r) - \mu_{Q'}(r)| > \varepsilon/2]$ for some range r. Within factor of 2 we will get $\Pr[A]$.

To analyze B, we think of the process differently. Instead of picking Q and Q' independently as two separate steps, we think of picking 2ℓ elements Q_0 and then splitting Q_0 into two halves Q and Q'.

$$\begin{aligned} \Pr[B] &= \sum_{Q_0} \Pr[Q_0] \cdot \Pr[B|Q_0] \\ &\leq \max_{Q_0} \Pr[B|Q_0] \end{aligned}$$

What is $\max_{Q_0} \Pr[B|Q_0]$?

We think of Q_0 as an arbitrary multiset of 2ℓ points and Q and Q' are obtained by evenly splitting of Q_0 into ℓ points each.

What is the advantage of this? If we fix Q_0, then $R|_{Q_0}$ has $\leq (2\ell)^d$ ranges.

Thus we need to only worry about a small number of ranges.

For any fixed range in $R|_{Q_0}$, via the additive Chernoff bound:

$$\Pr[|\mu_Q(r) - \mu_{Q_0}(r)| \geq \varepsilon/4] \leq 2e^{-2(\varepsilon/4)^2\ell}$$

$$\Pr[|\mu_{Q'}(r) - \mu_{Q_0}(r)| \geq \varepsilon/4] \leq 2e^{-2(\varepsilon/4)^2\ell}$$

Since $|\mu_Q(r) - \mu_{Q'}(r)| \leq |\mu_Q(r) - \mu_{Q_0}(r)| + |\mu_{Q_0}(r) - \mu_{Q'}(r)|$:

$$\Pr[|\mu_Q(r) - \mu_{Q'}(r)| > \varepsilon/2] \leq 4e^{-2(\varepsilon/4)^2\ell} = 4e^{-\varepsilon^2\ell/8}$$

By the union bound over all $(2\ell)^d$ ranges:

$$\Pr[B|Q_0] \leq (2\ell)^d \cdot 4e^{-\varepsilon^2\ell/8} \leq \frac{\delta}{2}$$

for $\ell \geq \frac{c}{\varepsilon^2}\left(d\log\frac{d}{\varepsilon} + \log\frac{1}{\delta}\right)$ with appropriate constant c.

Therefore $\Pr[B] \leq \delta/2$ and $\Pr[A] \leq 2\Pr[B] \leq \delta$, so $\Pr[\bar{A}] \geq 1 - \delta$.

Chapter 23

PAC Learning

23.1 PAC Learning

How do we generalize machine learning? Broadly speaking, **Supervised Learning** is about coming up with an algorithm or suite of algorithms that, given some initial training (labeled data), outputs a "prediction algorithm" that does well on future data that is unlabeled. The most basic but still very useful and important problem is **binary classification** (Yes or No).

23.1.1 Set Up

1. **Data** is formalized as vectors $\mathbf{x} \in X$.
2. **Labeled data**: pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where y_i is the correct label for data $\mathbf{x}_i$.
3. Want to obtain a **prediction algorithm** (could be randomized) $A(\mathbf{x})$ that, given a future data point $\mathbf{x}$, outputs a label.

Since arbitrary algorithms are complicated and hard to reason about and interpret and optimize over, traditionally the set of prediction algorithms (or **hypotheses** $\mathcal{H}$) were simple. The term **concept class** $\mathcal{C}$ is used to indicate that we are only interested in finding a "prediction" algorithm from this class.

23.1.2 The PAC Model

The PAC (Probably Approximately Correct) model, introduced by Valiant, is a nice theoretical model that addresses the issues of generalization, sample complexity, and efficiency of learning.

4. In the full consistency model we will assume that $\exists$ some $c^* : X \to \{0, 1\}$ which gives the true label for each example. This c is called a **concept**.
5. The examples are drawn from a **distribution** D.

Given a hypothesis $h \in \mathcal{H}$, we define its **error** with respect to c^* and D as:

$$\text{err}_D(h) = \Pr_{\mathbf{x} \sim D}[h(\mathbf{x}) \neq c^*(\mathbf{x})]$$

PAC Learnability Definition

Definition 23.1. A pair $(\mathcal{C}, \mathcal{H})$ is **PAC-Learnable** if $\exists$ a learning algorithm that, given $m = \text{poly}(1/\epsilon, 1/\delta)$ random samples from D, outputs (efficiently) a hypothesis h such that $\Pr_{\mathcal{S}}[\text{err}_D(h) > \epsilon] \leq \delta$.

(i) The algorithm is allowed to output an imperfect hypothesis even when there exists a perfect concept.

(ii) The algorithm is allowed to fail with some small probability δ.

23.1.3 Consistency and Generalization Bounds for Finite $\mathcal{H}$

Consistency

Definition 23.2. Given a set of correctly labeled examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ (where $y_i = c^*(\mathbf{x}_i)$), a hypothesis $h : X \to \{0, 1\}$ is **consistent** if $h(\mathbf{x}_i) = y_i$ for all i.

Theorem for Finite Hypothesis Classes (Consistent Case)

Theorem 23.3. *Let $\mathcal{H}$ be a finite hypothesis class and let $\epsilon, \delta \in (0, 1)$. Suppose the sample size is:*

$$n \geq \frac{1}{\epsilon}\left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$$

Then, if h is consistent with S, then $err_D(h) \leq \epsilon$ with probability $1 - \delta$.

Proof sketch. Say hypothesis h is **bad** if $\text{err}_D(h) > \epsilon$. Let $h_1, h_2, \ldots, h_l$ be the bad hypotheses. The probability that h_i is consistent with S is at most $(1 - \epsilon)^n$. By the Union Bound, if $l(1 - \epsilon)^n < \delta$, then no bad hypothesis will be consistent, so the output of the algorithm will be a good hypothesis. This condition is satisfied by the required n since $l \leq |\mathcal{H}|$. □

Theorem for Finite Hypothesis Classes (Agnostic Case)

We may only have a "good" hypothesis $h^* \in \mathcal{H}$ that makes a small error $\text{err}_D(h^*)$. We find a hypothesis h that makes the sample error $\text{err}_S(h)$ as small as possible.

Theorem 23.4. *Let $\mathcal{H}$ be a finite hypothesis class and let $\epsilon, \delta \in (0, 1)$. Suppose the sample size is:*

$$n \geq \frac{1}{2\epsilon^2}\left(\ln|\mathcal{H}| + \ln\frac{2}{\delta}\right)$$

Then, with probability $1 - \delta$, for all $h \in \mathcal{H}$:

$$|err_S(h) - err_D(h)| \leq \epsilon$$

Proof. Needs additive Chernoff bound. □

23.1.4 VC-Dimension Based Bounds

Consider the setting where $X \subseteq \mathbb{R}^d$ and $\mathcal{H}$ is the class of half-spaces in $\mathbb{R}^d$. Here, $|\mathcal{H}|$ is infinite.

Theorem 23.5. *Let $\mathcal{H}$ be a family of hypotheses with VC-dimension d_{VC}. Let $\epsilon, \delta \in (0, 1)$.*

1. *If $n \geq \frac{c}{\epsilon}\left(d_{VC}\ln\frac{d_{VC}}{\epsilon} + \ln\frac{1}{\delta}\right)$ for some constant c, then, with probability $1 - \delta$, any consistent hypothesis h has $err_D(h) \leq \epsilon$.*
2. *If $n \geq \frac{c}{\epsilon^2}\left(d_{VC}\ln\frac{d_{VC}}{\epsilon} + \ln\frac{1}{\delta}\right)$ for some constant c, then $\forall h \in \mathcal{H}, |err_D(h) - err_S(h)| \leq \epsilon$.*

23.1.5 Examples

Disjunctions

Let $X = \{0,1\}^n$. $\mathcal{C}$ is the class of **disjunctions** of Boolean literals (e.g., $z_1 + z_5 + \overline{z_7}$). The size of the concept class is $|\mathcal{C}| = 3^n$.

Claim 23.6. *Given a set of examples $S = \{(\mathbf{x}_i, y_i)\}$, there is an **efficient algorithm** that outputs a disjunction $c \in \mathcal{C}$ that is consistent with S if one exists.*

Algorithm Outline:

1. Start with $c = (z_1 + \overline{z_1}) + \cdots + (z_n + \overline{z_n})$.
2. Let S_0 be examples with $y_i = 0$ and S_1 be examples with $y_i = 1$.
3. **Prune based on S_0 (Negative Examples):** For each $i \in S_0$, remove literals from c that are satisfied by $\mathbf{x}_i$.
4. **Check S_1 (Positive Examples):** For each $i \in S_1$, check if at least one literal in c is satisfied by $\mathbf{x}_i$. If not, output "no consistent hypothesis".

Figure 23.1: Example Data Table

Example	**Data (x)**	**Label (y)**
$\mathbf{x}_1$	$(0,1,1,0,1)$	$y_1 = 0$
$\mathbf{x}_2$	$(0,0,1,0,1)$	$y_2 = 1$
$\mathbf{x}_3$	$(1,1,0,0,1)$	$y_3 = 0$
$\mathbf{x}_4$	$(1,1,0,0,0)$	$y_4 = 1$
$\mathbf{x}_5$	$(1,1,1,0,0)$	$y_5 = 1$

Half-Spaces in $\mathbb{R}^d$

$\mathcal{C}$ is the class of half-spaces in $\mathbb{R}^d$. Finding a consistent hypothesis is equivalent to finding a hyperplane $\mathbf{a} \cdot \mathbf{x} - b = 0$ that separates positive examples (S_1) from negative examples (S_0).

This can be formulated as a **Linear Program (LP)** to find the coefficients $a_1, \ldots, a_d, b$:

- Constraints for $i \in S_1$ (positive examples):

$$\sum_{j=1}^{d} a_j x_{i,j} - b > 0$$

- Constraints for $i \in S_0$ (negative examples):

$$\sum_{j=1}^{d} a_j x_{i,j} - b \leq 0$$

The LP is feasible if and only if a separating half-space exists.

Chapter 24

Primality Testing

24.1 Primality Testing

Problem: Given an integer N, check if N is a prime number.

Note that the representation size is $\log N$ bits. So an efficient algorithm runs in poly($\log N$) time.

- PRIMES $= \{x \in \{0,1\}^* : x$ interpreted as a binary integer is prime$\}$
- COMPOSITE $= \{0,1\}^* \setminus$ PRIMES

It is easy to see that COMPOSITE is in NP since one can prove that N is COMPOSITE by exhibiting x, y such that $N = x \cdot y$.

Not obvious that PRIMES is in NP. Vaughn Pratt in 1975 showed that PRIMES is in NP.

Question: Is there a poly time algorithm to check if N is a prime?

It had to wait till 2002 for a deterministic poly time algorithm due to Agarwal, Kayal, and Saxena.

However, a randomized poly time algorithm was known since 1977 due to Solovay and Strassen and Miller and Rabin.

We need some number theoretic and group theoretic background.

24.2 Background

Claim 24.1. *Given non-negative integers a, k, n, can compute $a^k \bmod n$ efficiently.*

Proof. Exercise. □

24.2.1 Euclid's Algorithm

Given integers $a, b > 0$, Euclid's algorithm can be used to obtain the following:

Theorem 24.2. *Given integers $a, b > 0$, $\exists$ integers x, y such that $\gcd(a, b) = ax + by$. Moreover, x, y can be computed in poly time.*

24.2.2 Groups

Let $n > 0$ be a positive integer. $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$ defines an additive abelian group under the operation mod$n$.

Let $\mathbb{Z}_n^* = \{a > 0 : a \in \mathbb{Z}_n, \gcd(a, n) = 1\}$ be the set of numbers that are relatively prime to n and in $\mathbb{Z}_n$.

Claim 24.3. *$\mathbb{Z}_n^*$ is a group under multiplication* mod$n$.

Proof. Recall Euclid's gcd algorithm. Given a, b, it returns $\gcd(a, b)$ and can be implemented to run in poly time. As a byproduct, it also returns x, y such that $ax + by = \gcd(a, b)$.

Now consider $a \in \mathbb{Z}_n^*$ and $b = n$. $\exists x, y$ such that $ax + yn = 1$, thus $ax \equiv 1 \pmod{n}$. $x \bmod n$ is a candidate for the inverse of a in $\mathbb{Z}_n^*$.

Cannot have x, x' such that $xa \equiv 1$ and $x'a \equiv 1 \pmod{n}$ because $(x - x')a \equiv 0 \pmod{n}$ but $\gcd(a, n) = 1 \Rightarrow x = x'$. □

Corollary 24.4. *$\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ is a group. $\mathbb{Z}_p$ is a field.*

Remark 24.5. Proof also shows that given $a \in \mathbb{Z}_n^*$, one can find a^{-1} efficiently.

24.2.3 Euler's Totient Function

Definition 24.6 (Euler totient function)**.** $\phi(m)$ for integer $m > 0$ is $|\mathbb{Z}_m^*|$.

Properties of ϕ:

- $\phi(1) = 1$
- For prime p: $\phi(p) = p - 1$
- For prime p and $k > 0$: $\phi(p^k) = p^k - p^{k-1}$
- For relatively prime n, m: $\phi(nm) = \phi(n)\phi(m)$

Via above properties:

Theorem 24.7. *If n has prime factorization $p_1^{k_1} p_2^{k_2} \cdots p_t^{k_t}$, then*

$$\phi(n) = n \prod_{i=1}^{t} \left(1 - \frac{1}{p_i}\right)$$

24.2.4 Lagrange's Theorem

Theorem 24.8 (Lagrange)**.** *Let G be a finite group and let $H \subseteq G$ be a subgroup. Then $|H|$ divides $|G|$.*

Definition 24.9. A group G is *cyclic* if $\exists$ an element $g \in G$ such that $\forall a \in G$, $\exists$ integer k such that $g^k = a$. g is called a *generator* for the group.

Definition 24.10. Given group G and $a \in G$, $\text{ord}(a)$ is smallest integer k such that $a^k = 1$ (identity).

For any $a \in G$, $H(a) = \{1, a, a^2, \ldots, a^{\text{ord}(a)-1}\}$ forms a cyclic subgroup of G. Therefore we have: $\text{ord}(a)$ divides $|G|$ for $a \in G$.

This implies:

Theorem 24.11 (Euler)**.** *For any $a \in \mathbb{Z}_m^*$, $a^{\phi(m)} \equiv 1 \pmod{m}$.*

Corollary 24.12 (Fermat)**.** *For any prime p: $a^{p-1} \equiv 1 \pmod{p}$ or $a^p \equiv a \pmod{p}$.*

Another useful lemma:

Lemma 24.13. *For any $n > 0$: $\sum_{d|n} \phi(d) = n$.*

Proof. Consider integer $d \in \{1, 2, \ldots, n\}$. Let $A_d = \{1 \le x \le n : \gcd(x, n) = d\}$. $A_d = \emptyset$ if d is not a divisor of n. $\{A_d : d|n\}$ partition $\{1, 2, \ldots, n\}$. Hence $\sum_d |A_d| = n$.

Not difficult to see $|A_d| = \phi(n/d)$. Hence

$$\sum_{d|n} \phi(d) = \sum_{d|n} \phi(n/d) = \sum_{d|n} |A_d| = n$$

□

24.2.5 Cyclic Groups

It is easy to see that $\mathbb{Z}_n$ for any n is cyclic. However, $\mathbb{Z}_n^*$ need not be cyclic in general.

However:

Theorem 24.14. *Let p be prime. Then $\mathbb{Z}_p^*$ is cyclic.*

Proof. Recall $|\mathbb{Z}_p^*| = p - 1$. For $k|(p-1)$, let $O_k = \{a \in \mathbb{Z}_p^* : \text{ord}(a) = k\}$ be the set of elements with order k.

From previous lemma: $\sum_{k|(p-1)} \phi(k) = p - 1$.

We will show later that $|O_k| = 0$ or $\phi(k)$.

$$\sum_k |O_k| = \sum_{k|(p-1)} \phi(k) = p - 1$$

and $|O_k| = \phi(k)$ for $k|(p-1)$. Thus $|O_{p-1}| = \phi(p-1) \ge 1$, so $\exists g \in \mathbb{Z}_p^*$ with order $p - 1$.

Claim 24.15. *$|O_k| = 0$ or $\phi(k)$.*

All elements in O_k are roots of the polynomial $X^k \equiv 1 \pmod{p}$ over the field $\mathbb{Z}_p$. Suppose $|O_k| \ge 1$. Then $\exists a$ root for the above polynomial, and further all the roots are $\{1, a, a^2, \ldots, a^{k-1}\}$. Since these are distinct, $\text{ord}(a) = k$. Note that $a^i \in O_k$ if $\gcd(i, k) = 1$. Thus $|O_k| = \phi(k)$ if $|O_k| \neq 0$. □

A number theoretic theorem:

Theorem 24.16. *$\mathbb{Z}_n^*$ is cyclic iff $n \in \{1, 2, 4, p^k, 2p^k\}$ for integer k and odd prime p.*

24.2.6 Chinese Remainder Theorem

Theorem 24.17 (Chinese Remainder Theorem)**.** *In a ring where $n = n_1 n_2 \cdots n_k$ are pairwise coprime, i.e., $\gcd(n_i, n_j) = 1$ for $i \neq j$. For any sequence $r_1, \ldots, r_k$ where $r_i \in \mathbb{Z}_{n_i}$, there exists a unique $r \in \mathbb{Z}_n$ s.t. $r \equiv r_i \pmod{n_i}$ for $i = 1$ to k. Moreover, given $\{r_i\}_i$, r can be computed efficiently.*

Proof. First we consider showing one r exists. Since n/n_i is coprime to n_i, a multiplicative inverse m_i in $\mathbb{Z}_{n_i}$ to n/n_i exists. Let m_i be that inverse: $(n/n_i) \cdot m_i \equiv 1 \pmod{n_i}$.

Also $(n/n_i) \cdot m_i \equiv 0 \pmod{n_j}$ for $j \neq i$.

Thus $r = \sum_i r_i m_i (n/n_i) \bmod n$ satisfies the desired congruences.

To see uniqueness, we do a counting argument. How many distinct $(r_1, \ldots, r_k)$ are there? $n_1 \cdot n_2 \cdots n_k$. For each $r \in \mathbb{Z}_n$, but there are only n elements in $\mathbb{Z}_n$, and for a given r we have only one $(r_1, \ldots, r_k)$. □

$$\mathbb{Z}_n \text{ is isomorphic to } \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k}$$

24.2.7 Quadratic Residues

Definition 24.18. A residue $a \in \mathbb{Z}_m^*$ is a *quadratic residue* if $\exists$ a number x such that $x^2 \equiv a \pmod{m}$. In other words, a is a quadratic residue if it has a square root.

Lemma 24.19. *Let p be a prime and consider generator g for $\mathbb{Z}_p^*$. Then g^k is a quadratic residue iff k is even.*

Proof. It is easy to see that if k is even then $g^k = (g^{k/2})^2$ is a square root of g^k. Suppose k is odd. If g^k is a quadratic residue then $g^k = (a)^2 = g^{2h}$ for some h since g is a generator. Thus $k = 2h$, contradiction. □

Corollary 24.20 (Euler)**.** *For $a \in \mathbb{Z}_p^*$, a is a quadratic residue iff $a^{(p-1)/2} \equiv 1 \pmod{p}$.*

Proof. $a = g^k$ where g is a generator, so $g^{p-1} = 1$ gives $g^{(p-1)/2} = \pm 1$. For a generator g, $g^{(p-1)/2} \equiv -1 \pmod{p}$. □

Definition 24.21 (Legendre symbol)**.** For prime p and $a \in \mathbb{Z}_p^*$ we define $\left(\frac{a}{p}\right)$ where

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is quadratic residue} \\ -1 & \text{if } a \text{ is not a quadratic residue} \end{cases}$$

Equivalently, $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}$ where we interpret $p-1$ as -1.

Now we consider a not necessarily prime odd number.

Definition 24.22 (Jacobi symbol)**.** Let n be an odd number with prime factorization $n = p_1^{k_1} p_2^{k_2} \cdots p_t^{k_t}$. For $a \in \mathbb{Z}_n^*$:

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{t} \left(\frac{a}{p_i}\right)^{k_i}$$

Note that $\left(\frac{a}{n}\right)$ is the same as the Legendre symbol when n is an odd prime. It is also ± 1.

Even though the definition of the Jacobi symbol involves the prime factorization, given a, n one can compute $\left(\frac{a}{n}\right)$ in poly time.

Theorem 24.23. *Given a, n where n is odd and $\gcd(a, n) = 1$, there is a poly time alg to compute $\left(\frac{a}{n}\right)$.*

One can derive the above from properties of the Jacobi symbol (see Motwani, Raghavan).

Definition 24.24. For an odd number n, define

$$J_n = \left\{ a \in \mathbb{Z}_n^* : \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n} \right\}$$

Note that $|J_n| = |\mathbb{Z}_n^*|$ when n is prime (by Euler's criterion).

A key observation is:

Lemma 24.25. *J_n is a subgroup of $\mathbb{Z}_n^*$ and is a proper subgroup if n is composite. Hence if n is composite, $|J_n| \leq |\mathbb{Z}_n^*|/2$.*

24.3 Solovay-Strassen Algorithm

The preceding observation leads to first RP algorithm for COMPOSITE due to Solovay and Strassen.

Algorithm: Input $n \geq 2$

1. If n is even, output Composite
2. Pick a random $a \in \{2, \ldots, n-1\}$
3. If $\gcd(a, n) \neq 1$, output Composite
4. Compute $a^{(n-1)/2} \bmod n$
5. Compute $\left(\frac{a}{n}\right)$
6. If $a^{(n-1)/2} \not\equiv \left(\frac{a}{n}\right) \pmod{n}$, then output Composite
7. Else output Prime

It is clear that algorithm outputs Prime for a prime, but for a composite it will err with prob $\leq 1/2$.

Theorem 24.26. *COMPOSITE $\in$ RP.*

By repeating we can reduce the error.

24.4 Miller-Rabin Test

We will assume n is odd and ≥ 5.

Simplest randomized test is to pick a number $a \in \{2, \ldots, n-2\}$ and check if $\gcd(a, n) \neq 1$. Test will succeed with prob $\geq 1 - \phi(n)/n$ but we can have $\phi(n) \approx n$. For instance, $n = pq$ where p, q are prime, then $\phi(n) = (p-1)(q-1)$. So we need some property.

24.4.1 Fermat Test

Pick $a \in \{2, \ldots, n-2\}$ randomly. If $a^{n-1} \not\equiv 1 \pmod{n}$, output Composite; else Prime.

If n is prime then correctly says it is prime, and if it says it is composite then it is correct, but it may say prime even when n is composite.

What is the probability?

Let $F_n = \{a \in \mathbb{Z}_n^* : a^{n-1} \equiv 1 \pmod{n}\}$.

Claim 24.27. *F_n is a subgroup of $\mathbb{Z}_n^*$.*

Proof. If $a, b \in F_n$, then $ab \in F_n$. Also a^{-1} is inverse of a and $a^{-1} \in F_n$. □

Suppose F_n is a proper subgroup of $\mathbb{Z}_n^*$. Then $|F_n| \leq |\mathbb{Z}_n^*|/2$ by Lagrange's theorem, and algorithm will have constant probability of success.

But what if $F_n = \mathbb{Z}_n^*$? Are there any such composite n?

Yes, they are called *Carmichael numbers*. There are infinitely many. Smallest is $561 = 3 \cdot 11 \cdot 17$.

Thus we need a test that handles Carmichael numbers.

Another property of primes is that $\mathbb{Z}_p$ is a field, and $x^2 \equiv 1 \pmod{p}$ has only two roots: $x = 1$ and $x = p - 1 \equiv -1 \pmod{p}$.

If n is composite then there can be non-trivial square roots. Ex: $n = 91$, then $1, 27, 64, 90$ are all square roots of 1.

If we find a non-trivial square root of 1 mod n, then n is composite. If $n = pq$, then by CRT only 4 non-trivial square roots.

24.4.2 Euler Test

Pick $a \in \{2, \ldots, n-2\}$. If $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$, output Prime; else Composite.

At least as good as Fermat test. But why only $a^{(n-1)/2}$ and not $a^{(n-1)/4}$? First may not be even, but if $(n-1)/2^i$ is odd then we can try.

24.4.3 Rabin-Miller Test

n is odd. Write $n - 1 = 2^u \cdot k$ where k is odd.

Algorithm:

1. Pick $a \in \{2, \ldots, n-1\}$
2. If $\gcd(a, n) \neq 1$, output Composite
3. $b_0 = a^k \bmod n$
4. If $b_0 \equiv \pm 1 \pmod{n}$, output Prime
5. For $i = 1$ to $u - 1$ do:
   - $b_i = b_{i-1}^2 \bmod n$
   - If $b_i = -1$, output Prime
   - If $b_i = 1$, return Composite
6. Return Composite since either $b_u \not\equiv 1 \pmod{n}$ or $b_u = 1$ and $b_{u-1} \neq -1$

Theorem 24.28. *If n is prime, alg outputs Prime with prob 1. If n is composite, alg outputs Prime with prob $\leq 1/4$.*

Sketch of the analysis:

Fix $a \in \mathbb{Z}_n^*$. Let $b_0 = a^k \bmod n$, $b_1 = a^{k \cdot 2} = a^{2k} \bmod n$, $b_2 = a^{k \cdot 2^2} = a^{4k} \bmod n$, $\ldots$, $b_u = a^{k \cdot 2^u} = a^{n-1} \bmod n$.

If b_i becomes 1, then $b_{i+1}, \ldots$ are all 1. When do we find a non-trivial square root of 1? If $b_i = 1$ and $b_{i-1} \neq \pm 1$. This is precisely where the alg outputs Composite. Also if $b_u = a^{n-1} \not\equiv 1 \pmod{n}$.

Example: $n = 325 = 5^2 \cdot 13$, $n - 1 = 324 = 81 \cdot 2^2$.

Definition 24.29. Let $n \geq 3$ be an odd composite. Let $n-1 = 2^k \cdot q$ where q is odd and $k \geq 1$. A number $a \in \mathbb{Z}_n^*$ is an *RM witness* for n if $a^q \not\equiv 1 \pmod{n}$ and $a^{2^i q} \not\equiv -1 \pmod{n}$ for $0 \leq i < k$.

If n is composite and a is not an RM witness for n, then a is an *RM liar* for n.

Lemma 24.30. *If a is an RM witness for n, then n is composite.*

24.4.4 Analysis

We will prove a weaker theorem:

Theorem 24.31. *Let $n > 3$ be odd composite, and let $L(n)$ be the set of RM liars for n. Then $|L(n)| \leq |\mathbb{Z}_n^*|/2$.*

The difficulty is that $L(n)$ is not a subgroup of $\mathbb{Z}_n^*$. Thus to prove the theorem we identify a proper subgroup S of $\mathbb{Z}_n^*$ and argue that $L(n) \subseteq S$.

We consider two cases:

Case 1: n is not a Carmichael number. If a is a RM liar then it is also a Fermat liar, and we argued that $|F_n| \leq |\mathbb{Z}_n^*|/2$ when n is not a Carmichael number.

Case 2: n is a Carmichael number.

However, this requires us to understand Carmichael numbers. We will state and not prove:

Theorem 24.32. *Suppose $n = p_1^{k_1} p_2^{k_2} \cdots p_t^{k_t}$ where each p_i is an odd prime.*

(a) n is Carmichael iff $\phi(p_i^{k_i}) | (n-1)$ for $i = 1$ to t.

(b) If n is Carmichael, then $n = p_1 p_2 \cdots p_t$ and $(p_i - 1) | (n-1)$ for $i = 1$ to t. In particular, $t \geq 3$.

Our goal is to find a proper subgroup S of $\mathbb{Z}_n^*$ s.t. $L(n) \subseteq S$.

Let i_0 be maximal $i > 0$ such that there is some RM liar a_0 with $a_0^{2^{i_0} q} \equiv -1 \pmod{n}$. Since n is odd composite, $i_0 \geq 1$, so i_0 exists.

Since n is a Carmichael number, $a^{n-1} \equiv 1 \pmod{n}$ and hence $0 \leq i_0 \leq k$.

Let $B_n = \{a \in \mathbb{Z}_n^* : a^{2^{i_0} q} \equiv \pm 1 \pmod{n}\}$.

Lemma 24.33.

(i) $L(n) \subseteq B_n$

(ii) B_n is a subgroup of $\mathbb{Z}_n^*$

(iii) $\mathbb{Z}_n^* \setminus B_n \neq \emptyset$

Proof. (i) Let $a \in L(n)$. Case 1: $a^q \equiv 1 \pmod{n}$. Then $a^{2^{i_0} q} \equiv 1 \pmod{n}$, hence $a \in B_n$. Case 2: $a^{2^i q} \equiv -1 \pmod{n}$ for some i. By defn of i_0: $0 \leq i \leq i_0$. If $i = i_0$ then $a \in B_n$. If $i < i_0$, then $a^{2^{i_0} q} \equiv 1 \pmod{n}$ and hence $a \in B_n$.

(ii) B_n is a subgroup: $1 \in B_n$ trivially. If $a, b \in B_n$, then $ab \in B_n$ (easy).

(iii) We know that n has at least 3 prime factors, hence $n = n_1 n_2$ where n_1, n_2 are odd and $\gcd(n_1, n_2) = 1$. Recall a_0 is a RM liar with $a_0^{2^{i_0} q} \equiv -1 \pmod{n}$. Let $\bar{a} = a_0 \bmod n_1$. By CRT, $\exists$ unique $a \in \mathbb{Z}_n$ s.t. $a \equiv \bar{a} \pmod{n_1}$ and $a \equiv 1 \pmod{n_2}$.

We claim $a \in \mathbb{Z}_n^* \setminus B_n$.

Computing modn_1: since $\bar{a} = a_0 \bmod n_1$, $a^{2^{i_0} q} \equiv -1 \pmod{n_1}$.

Computing modn_2: since $a \equiv 1 \pmod{n_2}$, $a^{2^{i_0} q} \equiv 1 \pmod{n_2}$.

Thus $a^{2^{i_0} q} \equiv -1 \pmod{n_1}$ and $a^{2^{i_0} q} \equiv 1 \pmod{n_2}$, so $a \notin B_n$.

Further, $a^{n-1} \equiv 1 \pmod{n_1}$ and $a^{n-1} \equiv 1 \pmod{n_2}$. By CRT, $a^{n-1} \equiv 1 \pmod{n}$, so $\gcd(a, n) = 1$ and $a \in \mathbb{Z}_n^*$. $\square$

24.5 Alternate Proof (Keith Conrad)

The preceding proof used characterization of Carmichael numbers and is from Dietzfelbinger's book. We give an alternate proof from the notes of Keith Conrad that avoids the use of Carmichael numbers.

We again consider 2 cases:

Case 1: $n = p^d$ for $d \geq 2$, p prime. We claim n is not Carmichael; we can use the fact that $|F_n| \leq |\mathbb{Z}_n^*|/2$.

To see this, it suffices to exhibit some $a \in \mathbb{Z}_n^*$ such that $a^{n-1} \not\equiv 1 \pmod{n}$.

Consider $a = 1 + p^{d-1}$. Note $\gcd(a, n) = 1$ since $p \nmid 1$. By Binomial expansion:

$$(1 + p^{d-1})^{n-1} = 1 + (n-1)p^{d-1} + \binom{n-1}{2} p^{2(d-1)} + \cdots$$

For $d \geq 2$, the terms with $p^{2(d-1)}$ and higher are divisible by p^d (since $2(d-1) \geq d$). Hence $(1 + p^{d-1})^{n-1} \equiv 1 + (n-1)p^{d-1} \pmod{p^d}$. Since $(n-1)p^{d-1} = (p^d - 1)p^{d-1}$ and $p^d \nmid (p^d - 1)p^{d-1}$ (because $p \nmid (p^d - 1)$), we get $(1 + p^{d-1})^{n-1} \not\equiv 1 \pmod{p^d}$.

Case 2: n is not a prime power. Write $n = p^s n_2$ where p does not divide n_2, so $n = n_1 n_2$ where n_1, n_2 are odd and n_1, n_2 relatively prime.

Let i_0 be maximal in $\{0, \ldots, u-1\}$ such that there is some $a \in \mathbb{Z}_n^*$ such that $a^{2^{i_0} q} \equiv -1 \pmod{n}$. Since $(-1)^2 = 1 \bmod n$, i_0 exists and $a \in \mathbb{Z}_n^*$.

Let $B_n = \{a \in \mathbb{Z}_n^* : a^{2^{i_0} q} \equiv \pm 1 \pmod{n}\}$.

Lemma 24.34.

(i) $L(n) \subseteq B_n$

(ii) B_n *is a subgroup of* $\mathbb{Z}_n^*$

(iii) $\mathbb{Z}_n^* \setminus B_n \neq \emptyset$

Proof. (i) Let $a \in L(n)$. Case 1: $a^q \equiv 1 \pmod{n}$. Then $a^{2^{i_0} q} \equiv 1 \pmod{n}$, hence $a \in B_n$. Case 2: $a^{2^i q} \equiv -1 \pmod{n}$ for some i. By defn of i_0: $0 \leq i \leq i_0$. If $i = i_0$ then $a \in B_n$. If $i < i_0$, then $a^{2^{i_0} q} \equiv 1 \pmod{n}$ and hence $a \in B_n$.

(ii) B_n is a subgroup: $1 \in B_n$ trivially. If $a, b \in B_n$, then $ab \in B_n$ (easy).

(iii) $n = n_1 n_2$ where n_1, n_2 are odd and $\gcd(n_1, n_2) = 1$. Recall a_0 is a RM liar with $a_0^{2^{i_0} q} \equiv -1 \pmod{n}$. Let $\bar{a} = a_0 \bmod n_1$. By CRT, $\exists$ unique $a \in \mathbb{Z}_n$ s.t. $a \equiv \bar{a} \pmod{n_1}$ and $a \equiv 1 \pmod{n_2}$.

We claim $a \in \mathbb{Z}_n^* \setminus B_n$.

Computing modn_1: since $\bar{a} = a_0 \bmod n_1$, $a^{2^{i_0} q} \equiv -1 \pmod{n_1}$.

Computing modn_2: since $a \equiv 1 \pmod{n_2}$, $a^{2^{i_0} q} \equiv 1^{2^{i_0} q} \equiv 1 \pmod{n_2}$.

Thus $a^{2^{i_0} q} \equiv -1 \pmod{n_1}$ and $a^{2^{i_0} q} \equiv 1 \pmod{n_2}$, so $a \notin B_n$.

Further, $a^{n-1} \equiv a_0^{n-1} \pmod{n_1}$ and $a^{n-1} \equiv 1 \pmod{n_2}$. By CRT, $a^{n-1} \equiv 1 \pmod{n}$ (need to verify $a_0^{n-1} \equiv 1 \pmod{n_1}$, which follows from Euler's theorem), so $\gcd(a, n) = 1$ and $a \in \mathbb{Z}_n^*$. $\square$

Chapter 25

Online Algorithms and Yao's Min-Max Principle

25.1 Introduction: Stairs vs. Elevators Problem

Consider the "Stairs vs. Elevators" problem. Let:

- S: Time to take stairs.
- L: Time to take elevator.
- Assume $S > L$.

Problem: How long should one wait for the elevator? If one knows the distribution of the arrival time of the elevator, then one can compute various quantities. We will work with the worst-case model, which has merits and cons. The **Adversary** decides the arrival time.

25.2 Deterministic Strategy

Is there a good deterministic strategy? Let $T = S - L$.

Let $\text{OPT}(t)$ be the optimum total time if the elevator arrives at time t.

- If $t \leq T$, then $\text{OPT}(t) = t + L$ (Wait for elevator).
- If $t > T$, then $\text{OPT}(t) = S$ (Take stairs immediately/early).

For an algorithm A, let $A(t)$ be the total time if the elevator arrives at time t. The **Competitive Ratio** of algorithm A is:

$$C(A) = \sup_t \frac{A(t)}{\text{OPT}(t)} \tag{25.1}$$

25.2.1 Characterizing Deterministic Algorithms

Any deterministic algorithm for this problem can be characterized by a single parameter a: the time it will wait before taking the stairs. (It will take the elevator if it comes before a).

Let $Wait_a$ be the algorithm. What is its competitive ratio?

Case $Wait_0$: This strategy is to always take the stairs. If $t = 0^+$ (elevator arrives immediately), then $Wait_0$ takes S time, while $\text{OPT}(0^+) = L$. The competitive ratio is $\frac{S}{L}$, which can be arbitrarily large.

General $Wait_a$: One can see that the worst case for $Wait_a$ is if the elevator comes at time $a + \epsilon$ (just after we give up and take the stairs). In this case, the competitive ratio is:

$$\frac{a + S}{\min(a + L, S)} \tag{25.2}$$

It is not hard to see that $a = S - L = T$ is the minimizing value (intuitively, it does not make sense to wait longer than T). Substituting $a = S - L$:

$$\text{Competitive Ratio} = \frac{S - L + S}{S} = 2 - \frac{L}{S} \tag{25.3}$$

Theorem: The optimum competitive ratio for the deterministic problem is $2 - \frac{L}{S}$.

25.3 Randomized Algorithms

What if we allow randomization in the algorithm? Can we improve the ratio?

25.3.1 The Model

Need to be careful about the model. We will consider an **Oblivious Adversary** that cannot see the randomness of the algorithm.

Alternatively, we let the adversary know the algorithm, pick the worst-case input σ, and then we allow the algorithm to randomize. We define the competitive ratio as:

$$\max_{\sigma} \frac{\mathbb{E}[A(\sigma)]}{\text{OPT}(\sigma)} \tag{25.4}$$

Note that $A(\sigma)$ is a random variable.

25.3.2 Optimal Randomized Strategy

For the elevator problem, we view a randomized algorithm as picking the waiting time according to a distribution. What distribution? It turns out the optimal distribution is to pick $w \in [0, T]$ according to the density function:

$$p(t) = \frac{1}{(e-1)T} e^{t/T} \tag{25.5}$$

Note that $\int_0^T p(t)dt = 1$. And $p(t) = 0$ for $t > T$.

Theorem: The competitive ratio of the randomized algorithm that picks waiting time according to $p(t)$ is $\frac{e}{e-1} \approx 1.58$.

25.3.3 Proof Sketch

Fix any arrival time $a \in [0, T]$. Suppose the elevator arrives at time a. Then $\text{OPT}(a) = a + L$.

What about the algorithm? Its total time is:

- $a + L$ if the waiting time picked is $\geq a$.
- $t + S$ if the waiting time picked is $t < a$.

Thus, the expected cost is:

$$\mathbb{E}[A(a)] = \int_0^a (S+t)p(t)dt + \left(\int_a^T p(t)dt\right)[a+L] \tag{25.6}$$

The worst case happens when $a = T$ (as in the deterministic case). Then we have:

$$\begin{aligned}\mathbb{E}[A(T)] &= S + \int_0^T tp(t)dt \\ &= S + \int_0^T \frac{t}{(e-1)T} e^{t/T} dt\end{aligned}$$

Using integration by parts:

$$\begin{aligned}&= S + \frac{1}{(e-1)T}\left[Tte^{t/T} - T^2 e^{t/T}\right]_0^T \\ &= S + \frac{1}{e-1}T\end{aligned}$$

While $\mathrm{OPT}(T) = S$.

Hence, the competitive ratio is:

$$\frac{S + \frac{1}{e-1}T}{S} = \frac{S + \frac{S-L}{e-1}}{S} = 1 + \frac{1}{e-1} - \frac{L}{S(e-1)} \tag{25.7}$$

If we let $L/S \to 0$, the ratio tends to:

$$1 + \frac{1}{e-1} = \frac{e-1+1}{e-1} = \frac{e}{e-1} \tag{25.8}$$

25.4 Lower Bounds: Yao's Min-Max Principle

How do we prove that the above ratio is optimal? In general, how do we prove lower bounds on randomized algorithms?

Typically, we use what is called **Yao's Lemma**. Yao recognized early on that one can use the Von-Neumann Min-Max characterization of two-player games to prove lower bounds.

25.4.1 The Principle

Instead of explaining the general setup, we illustrate it in the context of this discrete problem. To prove a lower bound on randomized algorithms, we focus on a lower bound on **deterministic algorithms against inputs drawn from a distribution**.

The game is as follows:

1. Come up with a probability distribution on inputs, say D.
2. Now, given knowledge of D, what is the best deterministic algorithm?
3. Suppose no deterministic algorithm can do better than C (even with knowledge of D).
4. Then C is a lower bound on randomized algorithms.

25.4.2 Application to Elevator Problem

For our problem, we design/pick a distribution on the arrival time of the elevator. Suppose we pick the distribution e^{-t} for the elevator arrival time. Note that we are allowing arbitrarily large times, but that is ok. To simplify analysis we assume $S = 1, L = 0$.

What is the expected value of OPT?

$$\mathbb{E}[\text{OPT}] = \int_0^\infty \min(1,t)p(t)dt = \int_0^1 te^{-t}dt + \int_1^\infty 1e^{-t}dt = 1 - e^{-1} \tag{25.9}$$

Fix any deterministic algorithm. It is a threshold algorithm with parameter a. Its expected cost is:

$$\mathbb{E}[A(a)] = \int_0^a te^{-t}dt + \int_a^\infty (a+1)e^{-t}dt = 1 \tag{25.10}$$

(Result doesn't depend on the threshold a!).

Thus, the competitive ratio is:

$$\geq \frac{1}{1-\frac{1}{e}} = \frac{e}{e-1} \tag{25.11}$$

25.5 Ski Rental Problem

A closely related problem, but in some sense a discrete problem. More well known. **Setup:** Costs $\$B$ to buy skis and \$1 to rent for 1 day ($B > 1$). Every day one has to decide whether to rent or buy if not already bought. Adversary decides when to break one's leg and finish the ski season. What is the best strategy to minimize total cost?

Deterministic: Rent for $\lfloor B \rfloor$ days and then buy. One can show 2-competitive and optimum for deterministic. Can obtain $(1-\frac{1}{e})^{-1}$ for randomized. Somewhat more technical than elevator/stairs problem because of discrete days.

25.6 Yao's Min-Max Principle (Formal View)

Consider a discrete setting where we enumerate all inputs of a particular size and all deterministic algorithms. Say m algorithms ($\mathcal{A}_i$) and n inputs (I_j).

	I_1	$\dots$	I_n
$\mathcal{A}_1$	c_{11}	$\dots$	c_{1n}
$\vdots$		$\ddots$	
$\mathcal{A}_m$	c_{m1}	$\dots$	c_{mn}

Let p be a distribution over inputs. Let q be a distribution over algorithms.

Let q^* be an optimal randomized algorithm. This corresponds to choosing q^* to minimize:

$$\max_{I_j} \frac{\sum_{i=1}^m q_i^* A_i(I_j)}{\text{OPT}(I_j)} = c^* \tag{25.12}$$

By Min-Max Theorem (swapping min and max over distributions):

$$\begin{aligned}\min_q \max_p \mathbb{E}[\text{cost}] &= \max_p \min_q \mathbb{E}[\text{cost}] \\ &= \min_{A_i} \sum_{j=1}^n p_j \frac{A_i(I_j)}{\text{OPT}(I_j)}\end{aligned}$$

25.7 Probabilistic Tree Embedding

Let $G = (V, E)$ be a graph. Want to approximate distances in G by a tree. Why? Trees are simple. Can one approximate distances by a single spanning tree?

Consider an n-cycle. For any spanning tree T, there is an edge $e = uv$ removed. Then $d_T(u, v) = n - 1$, while $d_G(u, v) = 1$. This is a large distortion.

Can we use randomization to improve this? Yes!

Theorem: There exists a probability distribution p over spanning trees $ST(G)$ such that for all u, v:

$$\mathbb{E}_{T \sim p}[d_T(u, v)] \leq O(\log n \log \log n) \cdot d_G(u, v) \tag{25.13}$$

Moreover, one can sample a tree from p efficiently.

Many applications to algorithms. The bound $O(\log n \log \log n)$ can be improved to $O(\log n)$ if G is a complete graph that corresponds to a metric space. This suffices in many applications. Lower bound is $\Omega(\log n)$ even for planar graphs (or general graphs as per context).