Lecture Notes: Randomized Algorithm Design

Palash Dey
Indian Institute of Technology, Kharagpur
palash.dey@cse.iitkgp.ac.in

Statutory warning: This is a draft version and may contain errors. If you find any error, please send an email to the author.

Contents

Notation: $\mathbb{N} = \{0, 1, 2, ...\}$ denotes the set of natural numbers, $\mathbb{R}$ denotes the set of real numbers. For a set $\mathcal{X}$, its power set is denoted by $2^{\mathcal{X}}$.

Chapter 1

Review of Basic Probability

In any probability experiment, the set of all possible outcomes is often called sample space and denoted by Ω. We typically wish to study the probability of certain subsets of the sample space; informally speaking, these subsets are called events. It is important to note that it may not be possible to "talk about" probability of every subset of the sample space if the sample space is uncountably infinite! The way-around is simple – give up the idea of being able to assign probability to every subset of the sample space. Let us now formalize this.

1.1 σ-Algebra and Probability Space:

To formalize the setting of probability, we introduce the notion of σ-algebra (also known as σ-field). It is defined as follows.

Definition 1.1.1 (σ-Algebra)**.** *A σ-algebra over a set Ω, is a set $\mathcal{F} \subseteq 2^{\Omega}$ of some subsets of Ω which satisfies the following properties.*

(i) $\Omega \in \mathcal{F}$

(ii) (closed under complements) for every $A \in \mathcal{F}$, we have $\Omega \setminus A \in \mathcal{F}$

(iii) (closed under countable unions) for every countable collection $A_i \in \mathcal{F}$ for every $i \in \mathbb{N}$, we have $\bigcup_{i \in \mathbb{N}} A_i \in \mathcal{F}$

A σ-algebra is often denoted by $(\Omega, \mathcal{F})$. Typically, $\mathcal{F}$ in a σ-algebra would be much smaller than 2^{Ω} but large enough to contain all sets of interest to us. We observe that the second and third properties σ-algebra together imply *closeness under countable intersections* due to De Morgan's Law: for every countable collection $A_i \in \mathcal{F}$ for every $i \in \mathbb{N}$, we have $\bigcap_{i \in \mathbb{N}} A_i \in \mathcal{F}$. We also observe that the first and second properties of σ-algebra together imply that $\emptyset \in \mathcal{F}$. In a σ-algebra, Ω is called the *sample space* and elements of $\mathcal{F}$ are called *events*. Below are some examples of σ-algebra.

Example 1.1.1 (Examples of σ-algebra)**.** *Following are few examples of σ-algebra.*

1. $(\{\textit{Head, Tail}\}, \{\{\textit{Head}\}, \{\textit{Tail}\}, \{\textit{Head,Tail}\}, \emptyset\})$ *is a σ-algebra.*
2. *More generally, let Ω be any set. Then $(\Omega, 2^{\Omega})$ is a σ-algebra.*

3. *Let Ω be any set. Then $(\Omega, \{\Omega, \emptyset\})$ is a σ-algebra.*

4. *Let $\Omega = \mathbb{R}$, $\mathcal{O}$ be the set of all open sets in $\mathbb{R}$, and $\mathcal{B}$ be the smallest set of sets that contains $\mathcal{O}$ and satisfies all the properties of σ-algebra (for example, the power set $2^{\mathbb{R}}$ of $\mathbb{R}$ contains $\mathcal{O}$ and satisfies all the properties of σ-algebra but it is not the smallest set). Then $(\mathbb{R}, \mathcal{B})$ is a σ-algebra which is popularly known as the* Borel σ-algebra. *It is important to note that $\mathcal{B} \neq 2^{\mathbb{R}}$. However, $\mathcal{B}$ is so rich that one often struggles to find a set which does not belong to $\mathcal{B}$*[1].

A probability distribution (also called probability measure or probability function) is defined on a σ-algebra as follows.

Definition 1.1.2 (Probability distribution)**.** *Given a σ-algebra $(\Omega, \mathcal{F})$, a probability distribution $\mathcal{P}$ on it is a function $\mathcal{P} : \mathcal{F} \longrightarrow [0, 1]$ which satisfies the following properties.*

(i) $\mathcal{P}[\Omega] = 1$

(ii) ($\mathcal{P}$ is countably σ-additive) for every countable collection $A_i \in \mathcal{F}, i \in \mathbb{N}$ of pairwise disjoint sets in $\mathcal{F}$, we have $\mathcal{P}\left[\bigcup_{i\in\mathbb{N}} A_i\right] = \sum_{i\in\mathbb{N}} \mathcal{P}[A_i]$

A probability distribution is often denoted as $(\Omega, \mathcal{F}, \mathcal{P})$.

1.2 Discrete and Continuous Probability Distributions:

A probability distribution $\mathcal{P}$ on a σ-algebra $(\Omega, \mathcal{F})$ is called a *discrete probability distribution* if Ω is finite or countably infinite. Otherwise $\mathcal{P}$ is called a *continuous probability distribution*. A discrete probability distribution on a σ-algebra $(\Omega, 2^{\Omega})$ is often described by what is called a *probability mass function (pmf for short)* $p : \Omega \longrightarrow [0, 1]$ which specifies the probability of every element $\omega \in \Omega$. The probability of any event $\mathcal{E} \in 2^{\Omega}$ is defined as $\mathcal{P}(\mathcal{E}) = \sum_{\omega\in\mathcal{E}} p(\{\omega\})$. On the other hand, a continuous probability distribution on $(\mathbb{R}, \mathcal{B})$ is often specified by what is called a *probability density function (pdf for short)* $f : \mathbb{R} \longrightarrow \mathbb{R}_{\geqslant 0}$ such that $\int_{-\infty}^{+\infty} f(x)dx = 1$. The probability of any event $\mathcal{E} \in \mathcal{B}$ is defined as $\mathcal{P}(\mathcal{E}) = \int_{\mathcal{E}} f(x)dx$[2]. The following corollary follows from the definition of a probability density function.

Corollary 1.2.1. *Let f be a pdf on $(\mathbb{R}, \mathcal{B})$. Then, for every $\mathcal{E} \in \mathcal{B}$, we have $\int_{\mathcal{B}} f(x)dx \in [0, 1]$.*

Proof. Since the function f never takes negative value, we have $\int_{\mathcal{B}} f(x)dx \geqslant 0$. On the other hand, we have the following chain of inequalities: $\int_{\mathcal{B}} f(x)dx \leqslant \int_{-\infty}^{+\infty} f(x)dx = 1$. Again the inequality follows from the fact that f never takes any negative value. □

Let us now look into some important examples of probability distribution.

▷ The function $p : \{0, 1\} \longrightarrow \mathbb{R}_{\geqslant 0}$ defined as $p(\{0\}) = \lambda, p(\{1\}) = 1 - \lambda$ for any $\lambda \in [0, 1]$ on the σ-algebra $(\{0, 1\}, 2^{\{0,1\}})$ is a probability distribution. This distribution is called the *Bernoulli distribution*.

[1] For an example of a set which is not a Borel set, refer https://en.wikipedia.org/wiki/Borel_set#Non-Borel_sets

[2] Can you see another critical use of the fact that $\mathcal{B}$ does not contain every subset of $\mathbb{R}$? The integral is defined for every subset of $\mathcal{B}$ but not defined for some subsets outside $\mathcal{B}$. The proof of these claims are out of scope of the course. Interested people are advised to take a course on measure theory and probability theory.

- ▷ Let $n \in \mathbb{N}$ be any natural number. Let us define $\Omega = \{x \in \mathbb{N} : x \leqslant n\}$. Then the function $p : \Omega \longrightarrow \mathbb{R}_{\geqslant 0}$ defines as $p(k) = \binom{n}{k}\lambda^k(1-\lambda)^{n-k}$ for any $\lambda \in [0,1]$ on the σ-algebra $(\Omega, 2^\Omega)$ is a probability distribution. This probability distribution is called the *Binomial distribution*.
- ▷ The function $p : \mathbb{N} \longrightarrow \mathbb{R}_{\geqslant 0}$ defined as $p(n) = \frac{e^{-\lambda}\lambda^n}{n!}$ for any $\lambda \in \mathbb{R}$ for $n \in \mathbb{N}$ on the σ-algebra $(\mathbb{N}, 2^{\mathbb{N}})$ is a probability distribution. This distribution is called the *Poisson distribution*.
- ▷ The function $p : \mathbb{N} \longrightarrow \mathbb{R}_{\geqslant 0}$ defined as $p(n) = \lambda(1-\lambda)^n$ for any $\lambda \in (0,1)$ for $n \in \mathbb{N}$ on the σ-algebra $(\mathbb{N}, 2^{\mathbb{N}})$ is a probability distribution. This distribution is called the *Geometric distribution*.
- ▷ Let Ω be any finite set. The function $p : \Omega \longrightarrow \mathbb{R}_{\geqslant 0}$ defined as $p(\omega) = 1/|\Omega|$ on the σ-algebra $(\Omega, 2^\Omega)$ is a probability distribution. This distribution is called the *discrete uniform distribution*.
- ▷ Let $a, b \in \mathbb{R}$ with $a < b$. Then the function $f : \mathbb{R} \longrightarrow \mathbb{R}_{\geqslant 0}$ defined as $f(x) = 1/(b-a)$ for every $x \in [a,b]$ and $f(x) = 0$ for every other x on the σ-algebra $(\mathbb{R}, \mathcal{B})$ is a probability density function. This distribution is called the *continuous uniform distribution*.
- ▷ The function $f : \mathbb{R} \longrightarrow \mathbb{R}_{\geqslant 0}$ defined as $f(x) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$ for any $\mu \in \mathbb{R}, \sigma \in \mathbb{R}_{>0}$ for every $x \in \mathbb{R}$ on the σ-algebra $(\mathbb{R}, \mathcal{B})$ is a probability density function. This distribution is called the *normal distribution or Gaussian distribution*. If $\mu = 0$ and $\sigma = 1$, then $f(x) = \frac{e^{-(x^2/2)}}{\sqrt{2\pi}}$ and the corresponding distribution is called the *standard normal distribution*.
- ▷ The function $f : \mathbb{R} \longrightarrow \mathbb{R}_{\geqslant 0}$ defined as $f(x) = \lambda e^{-\lambda x}$ for $x \geqslant 0$ and $f(x) = 0$ otherwise for any $\lambda \in (0, \infty)$ on the σ-algebra $(\Omega, 2^\Omega)$ is a probability distribution. This distribution is called the *exponential distribution*.

1.3 Cumulative Distribution Function:

Let $\mathcal{P}$ be any probability distribution on a σ-algebra $(\Omega, \mathcal{F})$ with $\Omega \subseteq \mathbb{R}$ and for every $x \in \mathbb{R}$, we have $\{\omega \in \Omega : \omega \leqslant x\} \in \mathcal{F}$. Then the cumulative distribution function (often abbreviated as CDF) or simply distribution function $\mathcal{F} : \mathbb{R} \longrightarrow [0,1]$ is defined as $\mathcal{F}(x) = \mathcal{P}[\{\omega \in \Omega : \omega \leqslant x\}]$. The following properties of CDF are easy to prove from the definition itself (hence left as an exercise). Let $\mathcal{F}$ be a CDF of a probability distribution $\mathcal{P}$.

- ▷ $\mathcal{F}$ is a non-decreasing function.
- ▷ $\mathcal{F}$ is right-continuous. Can you find an example of a probability distribution where the corresponding CDF is not left-continuous? (Hint: very easy!)
- ▷ $\lim_{x\to-\infty} \mathcal{F}(x) = 0$ and $\lim_{x\to\infty} \mathcal{F}(x) = 1$.
- ▷ Let $\mathcal{P}$ be a continuous distribution on the σ-algebra $(\mathbb{R}, \mathcal{B})$ with pdf f. Then prove that $\frac{d}{dx}(F) = f$.

1.4 Conditional Distribution:

Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probabilities distribution and $\mathcal{E} \in \mathcal{F}$ be an event such that $\mathcal{P}(\mathcal{E}) \neq 0$. For any event $\mathcal{A} \in \mathcal{F}$, the conditional probability of $\mathcal{A}$ given $\mathcal{E}$, denoted by $\mathcal{P}(\mathcal{A}|\mathcal{E})$, is defined as $\frac{\mathcal{P}(\mathcal{A}\cap\mathcal{E})}{\mathcal{P}(\mathcal{E})}$.

1.5 Independence:

Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probabilities distribution and $\mathcal{A}, \mathcal{B} \in \mathcal{F}$ be any two events. Intuitively speaking, we say that the events $\mathcal{A}$ and $\mathcal{B}$ are independent if the conditional probability of $\mathcal{A}$ given $\mathcal{B}$ or vice a versa is the same as the probability of $\mathcal{B}$ given $\mathcal{A}$ or vise a versa. That is $\mathcal{P}(\mathcal{A}|\mathcal{B}) = \mathcal{P}(\mathcal{A})$. The drawback of taking this as a definition of independence is that it does not work if $\mathcal{P}(\mathcal{B})$ is 0 (in this case, $\mathcal{P}(\mathcal{A}|\mathcal{B})$ is undefined). We observe that $\mathcal{P}(\mathcal{A}|\mathcal{B}) = \mathcal{P}(\mathcal{A})$ implies $\mathcal{P}(\mathcal{A} \cap \mathcal{B}) = \mathcal{P}(\mathcal{A})\mathcal{P}(\mathcal{B})$ and we take the later expression as the definition of independence since it does not need to assume $\mathcal{P}(\mathcal{A})$ or $\mathcal{P}(\mathcal{B})$ to be non-zero. That is, we would say that any two events $\mathcal{A}$ and $\mathcal{B}$ are independent if we have $\mathcal{P}(\mathcal{A} \cap \mathcal{B}) = \mathcal{P}(\mathcal{A})\mathcal{P}(\mathcal{B})$.

1.6 Random Variable:

Intuitively speaking, a random variable is a function which maps the outcomes of a random experiment to numerical values which helps us understand the random experiment easier. Obviously, it cannot be any arbitrary function since it needs to "respect" the properties of the probability distribution and σ-algebra discussed above. A random variable is called a real random variable if it maps to the set of real numbers. In this course, we need only real random variables. It is formally defined as follows.

Definition 1.6.1 ((Real) Random Variable)**.** *Let $\mathcal{P}$ be a probability distribution on a σ-algebra $(\Omega, \mathcal{F})$. A random variable $\mathcal{X} : \Omega \longrightarrow \mathbb{R}$ is a function such that for every Borel set $\mathcal{A} \subseteq \mathbb{R}$, the set $\{\omega \in \Omega : \mathcal{X}(\omega) \in \mathcal{A}\}$ belongs to $\mathcal{F}$; that is, the inverse image of every Borel set is an event (so that we can talk about the probability of it).*

Given a random variable $\mathcal{X}$ defined on a probability distribution $(\Omega, \mathcal{F}, \mathcal{P})$ and a Borel set $\mathcal{A} \subseteq \mathbb{R}$, the probability that $\mathcal{X}$ takes its value in $\mathcal{A}$ is $\Pr[\mathcal{X} \in \mathcal{A}] = \mathcal{P}(\{\omega \in \Omega : \mathcal{X}(\omega) \in \mathcal{A}\})$. We observe that if the probability of every Borel set in $\mathcal{R}$ for a random variable $\mathcal{X}$ is somehow specified, then we do not need to bother about the underlying probability distribution $(\Omega, \mathcal{F}, \mathcal{P})$ and work exclusively with the random variable $\mathcal{X}$ along with the Borel σ-algebra $(\mathbb{R}, \mathcal{B})$ since all the information about $\mathcal{P}$ is already available. In this case, the distribution and CDF of $\mathcal{P}$ induces a corresponding distribution and CDF on the Borel σ-algebra $(\mathbb{R}, \mathcal{B})$ which are called respectively the distribution and CDF of the random variable $\mathcal{X}$. A random variable is called discrete if the underlying probability distribution is discrete. For a discrete random variable $\mathcal{X}$, the set $\{x \in \mathbb{R} : \Pr[\mathcal{X} = x] \neq 0\}$ is countable and is called the *support of* $\mathcal{X}$. Similarly, a random variable is called continuous if the underlying probability distribution is continuous. The proof of the following facts are left as exercises.

Fact 1.6.1. ▷ *Let $\mathcal{X}$ be a random variable and $\phi : \mathbb{R} \longrightarrow \mathbb{R}$ a continuous real valued function. Then $\phi(\mathcal{X})$ (or more formally, $\phi \circ \mathcal{X}$) is also a random variable.*

▷ *Let $\mathcal{X}$ and $\mathcal{Y}$ be two random variables. Then $\mathcal{X} + \mathcal{Y}, \mathcal{X} - \mathcal{Y}, \mathcal{X}\mathcal{Y}$ are also random variables.*

1.7 Expectation of Random Variable:

Intuitively, the expectation of a random variable is its average value weighted by corresponding probabilities. Concretely, let $\mathcal{X}$ be a discrete random variable with support $\mathcal{S}$. Then we say that the expectation $\mathbb{E}[\mathcal{X}]$ of $\mathcal{X}$

exists if the sum $\sum_{x\in S} |x| \Pr[\mathcal{X} = x]$ converges[3]. If $\mathbb{E}[\mathcal{X}]$ exists, then we define $\mathbb{E}[\mathcal{X}] = \sum_{x\in S} x \Pr[\mathcal{X} = x]$. The following fact is easy to prove from the definition of convergent series.

Fact 1.7.1. *Let $\mathcal{X}$ be a discrete random variable. If expectation of $\mathcal{X}$ exists, then $\sum_{x\in S} x \Pr[\mathcal{X} = x]$ converges to a unique real number.*

Similarly, if $\mathcal{X}$ is a continuous random variable, then we say that the expectation $\mathbb{E}[\mathcal{X}]$ of $\mathcal{X}$ exists if $\int_{-\infty}^{+\infty} |x| f(x) dx$ converges and If $\mathbb{E}[\mathcal{X}]$ exists, then we define $\mathbb{E}[\mathcal{X}] = \int_{-\infty}^{+\infty} x f(x) dx$. The following fact is easy to prove again from the definition of convergence.

Fact 1.7.2. *Let $\mathcal{X}$ be a continuous random variable. If expectation of $\mathcal{X}$ exists, then $\int_{-\infty}^{+\infty} x f(x) dx$ converges to a unique real number.*

We now prove the most important property of expectation: it is a linear function.

Lemma 1.7.1 (Linearity of Expectation)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two random variables both having their expectations and c any real number. Then we have $\mathbb{E}[c\mathcal{X} + \mathcal{Y}] = c\mathbb{E}[\mathcal{X}] + \mathbb{E}[\mathcal{Y}]$.*

Proof. We prove the result for the case when both $\mathcal{X}$ and $\mathcal{Y}$ are discrete random variables. The same technique can be adopted to prove the result for the cases when both $\mathcal{X}$ and $\mathcal{Y}$ are either discrete and continuous random variable. The proof for the general case is out of scope of the course. Let $\mathcal{Z} = c\mathcal{X} + \mathcal{Y}$. Since $\mathcal{X}$ and $\mathcal{Y}$ are discrete random variable, the support $S(\mathcal{Z})$ of the random variable $c\mathcal{X} + \mathcal{Y}$ is countable (since it is a subset of the union of the supports of $\mathcal{X}$ and $\mathcal{Y}$). The existence of $\mathbb{E}[\mathcal{Z}]$ follows from the chain of inequalities below.

$$\begin{aligned}
\sum_{z\in S(\mathcal{Z})} |z| \Pr[\mathcal{Z} = z] &= \sum_{x\in S(\mathcal{X})} \sum_{y\in S(\mathcal{Y})} |(cx + y)| \Pr[\mathcal{X} = x, \mathcal{Y} = y] \\
&\leqslant \sum_{x\in S(\mathcal{X})} \sum_{y\in S(\mathcal{Y})} (|cx| \Pr[\mathcal{X} = x, \mathcal{Y} = y] + |y| \Pr[\mathcal{X} = x, \mathcal{Y} = y]) \\
&= |c| \sum_{x\in S(\mathcal{X})} \sum_{y\in S(\mathcal{Y})} |x| \Pr[\mathcal{X} = x, \mathcal{Y} = y] + \sum_{x\in S(\mathcal{X})} \sum_{y\in S(\mathcal{Y})} |y| \Pr[\mathcal{X} = x, \mathcal{Y} = y] \\
&= |c| \sum_{x\in S(\mathcal{X})} |x| \sum_{y\in S(\mathcal{Y})} \Pr[\mathcal{X} = x, \mathcal{Y} = y] + \sum_{y\in S(\mathcal{Y})} |y| \sum_{x\in S(\mathcal{X})} \Pr[\mathcal{X} = x, \mathcal{Y} = y] \\
&= |c| \sum_{x\in S(\mathcal{X})} |x| \Pr[\mathcal{X} = x] + \sum_{y\in S(\mathcal{Y})} |y| \Pr[\mathcal{Y} = y]
\end{aligned}$$

The inequality above follows from triangle inequality. Hence $\sum_{z\in S(\mathcal{Z})} |z| \Pr[\mathcal{Z} = z]$ converges since both $\sum_{x\in S(\mathcal{X})} |x| \Pr[\mathcal{X} = x]$ and $\sum_{y\in S(\mathcal{Y})} |y| \Pr[\mathcal{Y} = y]$ converge. From the definition of expectation of $\mathcal{Z}$, we now have the following.

$$\mathbb{E}[c\mathcal{X} + \mathcal{Y}] = \sum_{x\in S(\mathcal{X})} \sum_{y\in S(\mathcal{Y})} (cx + y) \Pr[\mathcal{X} = x, \mathcal{Y} = y]$$

[3]Pay special attention to the fact that we demand convergence of $\sum_{x\in S} |x| \Pr[\mathcal{X} = x]$ instead of $\sum_{x\in S} x \Pr[\mathcal{X} = x]$. In other words, we demand that the series $\sum_{x\in S} x \Pr[\mathcal{X} = x]$ must converge *absolutely* for expectation to exist. Without absolute convergence, it may be possible to rearrange the terms in the series $\sum_{x\in S} x \Pr[\mathcal{X} = x]$ to obtain different answers (this is called the Riemann Rearrangement Theorem or the Reimann Series Theorem)! To see an example, consider the series $1 - 1 + 1/2 - 1/2 + 1/3 - 1/3 + \ldots$ converges to 0 but does not absolutely converge (the series $1 + 1 + 1/2 + 1/2 + 1/3 + 1/3 + \ldots$ diverges). Let us rearrange the above series as follows: $1 + 1/2 - 1 + 1/3 + 1/4 - 1/2 + \ldots$ converges to $\ln 2$. A series which converges but does not absolutely converge is called a *conditionally convergence series*.

$$
\begin{aligned}
&= \sum_{x \in S(\mathcal{X})} \sum_{y \in S(\mathcal{Y})} (cx \Pr[\mathcal{X} = x, \mathcal{Y} = y] + y \Pr[\mathcal{X} = x, \mathcal{Y} = y]) \\
&= c \sum_{x \in S(\mathcal{X})} \sum_{y \in S(\mathcal{Y})} x \Pr[\mathcal{X} = x, \mathcal{Y} = y] + \sum_{x \in S(\mathcal{X})} \sum_{y \in S(\mathcal{Y})} y \Pr[\mathcal{X} = x, \mathcal{Y} = y] \\
&= c \sum_{x \in S(\mathcal{X})} x \sum_{y \in S(\mathcal{Y})} \Pr[\mathcal{X} = x, \mathcal{Y} = y] + \sum_{y \in S(\mathcal{Y})} y \sum_{x \in S(\mathcal{X})} \Pr[\mathcal{X} = x, \mathcal{Y} = y] \\
&= c \sum_{x \in S(\mathcal{X})} x \Pr[\mathcal{X} = x] + \sum_{y \in S(\mathcal{Y})} y \Pr[\mathcal{Y} = y] \\
&= c\mathbb{E}[\mathcal{X}] + \mathbb{E}[\mathcal{Y}]
\end{aligned}
$$

The second equality follows from the definition of support (of $\mathcal{Z}$); the fourth, fifth, and sixth equality follows from the existence of expectation of $\mathcal{Z}$, $c\mathcal{X}$, and $\mathcal{Y}$. □

Repeated application of Lemma 1.7.1 gives us the following.

Corollary 1.7.1. *Let $\mathcal{X}_i, i \in [n]$ be n random variables having individual expectations for any natural number $n \geqslant 1$. Then we have $\mathbb{E}[\mathcal{X}_1 + \cdots + \mathcal{X}_n] = \mathbb{E}[\mathcal{X}_1] + \cdots + \mathbb{E}[\mathcal{X}_n]$.*

1.8 Variance of Random Variable:

Let $\mathcal{X}$ be a random variable whose expectation exists and equal to $\mu \in \mathbb{R}$. If the expectation of $(\mathcal{X}-\mu)^2$ exists, then we say that the variance of $\mathcal{X}$, denoted by $\text{var}(\mathcal{X})$ exists and it is equal to $\mathbb{E}[(\mathcal{X} - \mu)^2]$. The following lemma proves an often useful formula for the variance.

Lemma 1.8.1. *Let $\mathcal{X}$ be a random variable whose both expectation and variance exist. Then we have var$(\mathcal{X}) = \mathbb{E}[\mathcal{X}^2] - (\mathbb{E}[\mathcal{X}])^2$.*

Proof. We have the following chain of equalities.

$$
\begin{aligned}
\text{var}(\mathcal{X}) &= \mathbb{E}[(\mathcal{X} - \mu)^2] \\
&= \mathbb{E}[\mathcal{X}^2 - 2\mu\mathcal{X} + \mu^2] \\
&= \mathbb{E}[\mathcal{X}^2] - 2\mu\mathbb{E}[\mathcal{X}] + \mu^2 \\
&= \mathbb{E}[\mathcal{X}^2] - 2\mu^2 + \mu^2 \\
&= \mathbb{E}[\mathcal{X}^2] - (\mathbb{E}[\mathcal{X}])^2
\end{aligned}
$$

The third inequality follows from the linearity of expectation and the fact that μ is a constant. □

For any random variable $\mathcal{X}$, we observe that $\text{var}(\mathcal{X})$ is the expectation of a non-negative random variable, namely $(\mathcal{X} - \mu)^2$. Hence, $\text{var}(\mathcal{X})$, if it exists, is always non-negative. This proves the following result.

Corollary 1.8.1. *Let $\mathcal{X}$ be a random variable whose both expectation and variance exist. Then we have $\mathbb{E}[\mathcal{X}^2] \geqslant (\mathbb{E}[\mathcal{X}])^2$.*

Later in this course, we may see a generalization of the above Corollary to any convex function; this generalization is known as Jensen's inequality.

1.9 Conditional Expectation:

Conditional Expectation given an Event:

Let $\mathcal{X}$ be a random variable and $\mathcal{A} \in \mathcal{B}$ be a Borel set with $\Pr[\mathcal{X} \in \mathcal{A}] \neq 0$. Then the conditional expectation of $\mathcal{X}$ given $\mathcal{A}$ is the expectation of the random variable $\mathcal{X}|\mathcal{A}$. For any Borel set $\mathcal{C} \in \mathcal{B}$, the probability $\Pr[\mathcal{X} \in \mathcal{C}|\mathcal{A}]$ that $\mathcal{X}$ belongs to $\mathcal{C}$ given $\mathcal{A}$ is defined as $\frac{\Pr[\mathcal{X} \cap \mathcal{A}]}{\Pr[\mathcal{A}]}$. Hence the formula for condition expectation of $\mathcal{X}$ given $\mathcal{A}$ is as follows. Below, $\mathbb{1}_{\mathcal{A}}(x)$ is the indicator random variable for the event $\mathcal{A}$: it takes value 1 if $x \in \mathcal{A}$ and 0 otherwise.

$$\mathbb{E}[\mathcal{X}|\mathcal{A}] = \begin{cases} \sum_{x \in \text{Sup}(\mathcal{X}) \cap \mathcal{A}} x \frac{\Pr[\mathcal{X}=x]}{\Pr[\mathcal{A}]} & \text{if } \mathcal{X} \text{ is a discrete random variable} \\ \int_{-\infty}^{+\infty} x \mathbb{1}_{\mathcal{A}}(x) \frac{f(x)}{\Pr[\mathcal{A}]} dx & \text{if } \mathcal{X} \text{ is a continuous random variable} \end{cases}$$

Conditional Expectation given another Random Variable:

Let $\mathcal{X}$ and $\mathcal{Y}$ be two random variables defined on same underlying probability space $(\Omega, \mathcal{F}, \mathcal{P})$. In this subsection, we will restrict ourselves to the case where $\mathcal{Y}$ is a discrete random variable[4] (observe how many things will break down in the following lines without this assumption). For every y in the support of $\mathcal{Y}$, let $\mathcal{E}_y$ be the event that the random variable $\mathcal{Y}$ takes value y. Then the expectation $\mathbb{E}[\mathcal{X}|\mathcal{Y} = y]$ of the random variable $\mathcal{X}$ given $\mathcal{Y} = y$ is defined as $\mathbb{E}[\mathcal{X}|\mathcal{Y} = y] = \mathbb{E}[\mathcal{X}|\mathcal{E}_y]$. Observe that, for every y in the support of $\mathcal{Y}$, $\mathbb{E}[\mathcal{X}|\mathcal{Y} = y]$ is some real number. Hence, $\mathbb{E}[\mathcal{X}|\mathcal{Y} = \cdot]$ is a discrete random variable (can you see why $\mathbb{E}[\mathcal{X}|\mathcal{Y} = \cdot]$ is a discrete random variable even when $\mathcal{X}$ is a continuous random variable?) taking value $\mathbb{E}[\mathcal{X}|\mathcal{Y} = y]$ with probability $\sum_{y' \in \text{Sup}(\mathcal{Y}) : \mathbb{E}[\mathcal{X}|\mathcal{Y}=y] = \mathbb{E}[\mathcal{X}|\mathcal{Y}=y']} \Pr[\mathcal{Y} = y']$. The following result is useful.

Lemma 1.9.1. *Let X, Y, X be three random variables such that $\mathbb{E}[X|Y, Z]$ and $\mathbb{E}[X|Z]$ exist. Then $\mathbb{E}[\mathbb{E}[X|Y, Z]|Z] = \mathbb{E}[X|Z]$.*

Proof. As usual, we will prove this statement when X, Y, X are three discrete random variables. We have

$$\begin{aligned} \mathbb{E}[\mathbb{E}[X|Y, Z]|Z] &= \mathbb{E}\left[\sum_{x \in Supp(X)} x \Pr[X = x|Y, Z] \middle| Z \right] \\ &= \sum_{y \in Supp(Y)} \left(\sum_{x \in Supp(X)} x \Pr[X = x|Y = y, Z] \right) \Pr[Y = y] \\ &= \sum_{x \in Supp(X)} \left(x \sum_{y \in Supp(Y)} \Pr[X = x|Y = y, Z] \Pr[Y = y] \right) \\ &= \sum_{x \in Supp(X)} x \Pr[X = x|Z] \\ &= \mathbb{E}[X|Z] \end{aligned}$$

□

[4] Conditional expectation given some continuous random variable is more non-trivial. Interested readers are referred to Section 1.6 of the book "Probability: Theory and Examples" by Durrett, R.

Chapter 2

First Few Examples Randomized Algorithm

2.1 Types of Randomized Algorithms

A randomized algorithm has access to coins with probabilities of head being any real number $p \in (0,1)$. Randomized algorithms can broadly be classified into two types: (i) *Las Vegas type randomized algorithm:* this type of randomized algorithms always output the right answer on all instances; however the running time depends on the outcome of the coin tosses, (ii) *Monte Carlo type randomized algorithm:* this type of randomized algorithm takes similar time irrespective of the outcomes of the coin tosses; however it may sometime answer wrongly. For Las Vegas type randomized algorithms, we study the expected time complexity (the expectation of the random variable which denotes the number of steps the algorithm takes). For Monte Carlo type randomized algorithm, we study the probability that the algorithm outputs wrongly. We now see some examples of both types of algorithms. We will now see a Monte Carlo type randomized algorithm for our first problem which is popularly known as polynomial identity testing.

2.2 Polynomial Identity Testing (PIT)

Our first problem is the Polynomial Identity Testing problem. In this problem, we are given two polynomials p and q in $\mathbb{F}[X_1, X_2, \ldots, X_n]$ (that is, p and q are polynomials in variables $X_1, X_2, \ldots, X_n$ over a field $\mathbb{F}$) and we need to compute whether p and q are the same polynomials. That is whether $p - q$ is a 0 polynomial. Before going forward, let us first define what is a polynomial.

Definition 2.2.1 (Polynomial over Fields)**.** *A polynomial $p(X_1, X_2, \ldots, X_n)$ in n variables $X_1, X_2, \ldots, X_n$ over a field $\mathbb{F}$[1] is an expression of the form $\sum_{(i_1,\ldots,i_n)\in\mathbb{N}^n} a_{(i_1,\ldots,i_n)} X_1^{i_1} \ldots X_n^{i_n}$ where $a_{(i_1,\ldots,i_n)} \in \mathbb{F}$ and all but finitely many of $a_{(i_1,\ldots,i_n)}$s are zero. More formally, a polynomial $p(X_1, X_2, \ldots, X_n)$ in n variables $X_1, X_2, \ldots, X_n$ over a field $\mathbb{F}$ is a function $f_p : \mathbb{N}^n \longrightarrow \mathbb{F}$ such that the inverse image of $\mathbb{F}\setminus\{0\}$ under f_p is a finite set. Each individual*

[1] A field, informally speaking, is an algebraic structure which has addition and multiplication as two basic operations and allows division by non-zero elements. Common examples of field are the field of rational numbers, the field of real numbers. These are examples of infinite fields (contain infinitely many elements). However, fields can as well be finite. For example, for any prime number p, there is a field, denoted by $\mathbb{F}_p$, which contains $\{0, 1, \ldots, p-1\}$ and additions and multiplications are modulo p.

term $a_{(i_1,\ldots,i_n)}X_1^{i_1}\ldots X_n^{i_n}$ *where* $a_{(i_1,\ldots,i_n)} \neq 0$ *is called a monomial and* $a_{(i_1,\ldots,i_n)}$ *is called the coefficient of the monomial.*

For any field $\mathbb{F}$, we denote the set[2] of all polynomials over the field $\mathbb{F}$ in variables $X_1,\ldots,X_n$ by $\mathbb{F}[X_1,\ldots,X_n]$. Observe that polynomials, as defined Definition 2.2.1, can as well be treated as formal objects or formal expressions. We say two polynomials p and q are *identical* if $p-q$ is the zero polynomial. Let us see few examples of polynomials.

Example 2.2.1 (Polynomials)**.** $4X_1^3X_2^2 - 10.3X_1X_2 \in \mathbb{R}[X_1,X_2], X_1^3 + X_1^2X_2 + X_2X_3 \in \mathbb{F}_2[X_1,X_2,X_3]$*, etc.*

One invariant of any polynomial is its degree. For polynomials in more than one variable, the are many notion of degree of a polynomial (all of them coincides with our usual notion of degree for polynomials in single variable). In our context, *total degree* will be relevant which is defined as follows.

Definition 2.2.2 (Total Degree of Polynomial)**.** *Let* $p(X_1,X_2,\ldots,X_n) = \sum_{(i_1,\ldots,i_n)\in\mathbb{N}^n} a_{(i_1,\ldots,i_n)}X_1^{i_1}\ldots X_n^{i_n}$ *be a polynomial in* n *variables* $X_1,X_2,\ldots,X_n$ *over a field* $\mathbb{F}$*. The total degree of a monomial* $a_{(i_1,\ldots,i_n)}X_1^{i_1}\ldots X_n^{i_n}$ *is defined as* $\sum_{j=1}^{n} i_j$*. The total degree of a polynomial is the highest total degree of its monomials.*

Example 2.2.2 (Total degree of polynomials)**.** *Total degree of* $4X_1^3X_2^2 - 10.3X_1X_2$ *is* 5 *and the total degree of* $X_1^3 + X_1^2X_2 + X_2X_3$ *is* 3*, etc.*

Any polynomial naturally defines a polynomial function as follows.

Definition 2.2.3 (Polynomial Function)**.** *Let* $p(X_1,X_2,\ldots,X_n) = \sum_{(i_1,\ldots,i_n)\in\mathbb{N}^n} a_{(i_1,\ldots,i_n)}X_1^{i_1}\ldots X_n^{i_n}$ *be a polynomial in* n *variables* $X_1,X_2,\ldots,X_n$ *over a field* $\mathbb{F}$*. Then the function* $f_p : \mathbb{F}^n \longrightarrow \mathbb{F}$ *induced by the polynomial* p *is defined as* $f_p(x_1,x_2,\ldots,x_n) = \sum_{(i_1,\ldots,i_n)\in\mathbb{N}^n} a_{(i_1,\ldots,i_n)}x_1^{i_1}\ldots x_n^{i_n}$ *where* $x_1,x_2,\ldots,x_n \in \mathbb{F}$*.*[3]

It is immediate that if two polynomials p and q are identical, then their corresponding polynomial functions f_p and f_q are also identical. However, *the converse is not true* as the following example shows.

Example 2.2.3 (Example of two polynomials with same induced function)**.** *Let us consider* $p = X(X-1) \in \mathbb{F}_2[X]$ *and* $q = 0 \in \mathbb{F}_2[X]$*. We observe that* $f_p(0) = 0$ *and* $f_p(1) = 0$*. We also have* $f_q(0) = f_q(1) = 0$*. Hence* f_p *and* f_q *are identical functions from* $\mathbb{F}_2$ *to* $\mathbb{F}_2$*. On the other hand, clearly,* p *and* q *are not identical polynomials.*

We now formally define our problem.

Definition 2.2.4 (Polynomial Identity Testing (PIT))**.** *Given a polynomial* $p(X_1,\ldots,X_n) \in \mathbb{F}[X_1,\ldots,X_n]$*, compute whether* p *is a* 0 *polynomial or not.*

We observe that the problem of computing whether two given polynomials p and q in $\mathbb{F}[X_1,\ldots,X_n]$ are identical polynomial reduces to the problem of deciding whether the polynomial $p-q$ is identically 0.

Input Format: There are various possibilities for specifying the input polynomial. In our context, we assume that the polynomial has been specified as a formula; an individual monomial $aX_1^{i_1}\ldots X_n^{i_n}$ is specified by the tuple $(a,i_1,i_2,\ldots,i_n)$.

[2] It actually posses much richer structure than simply a set. For example, $\mathbb{F}[X_1,\ldots,X_n]$ forms an $\mathbb{F}$-algebra.

[3] Why are we not bothered about convergence in the definition of $f_p(x_1,x_2,\ldots,x_n)$?

Obvious Algorithm is Inefficient: The obvious algorithm for the polynomial identity testing problem may be to simply expand the polynomial as sum of monomials, perform necessary cancellations, and output that the input polynomial is identically 0 if and only if all the monomials cancel. This algorithm, although correct, does not run in polynomial time since expanding the input polynomial as a sum of monomials may require writing exponentially many terms. For example, try expanding the polynomial $(X_1 + Y_1)\dots(X_n + Y_n) \in \mathbb{R}[X_1, \dots, X_n, Y_1, \dots, Y_n]$ and see how many terms it involves.

As of the time of writing, we do not know any deterministic polynomial time algorithm for the polynomial identity testing problem (we also do not know any proof of non-existence). Indeed, finding a deterministic polynomial time algorithm for this problem has been a challenging research question for many years. However, there exists a simple randomized algorithm. To see the intuition behind the algorithm, let us assume for the moment that the input polynomial p is a real polynomial in one variable. Let d be the degree of p. Then p has at most d zeros (roots of the equation $p = 0$). Hence, if we randomly pick a real number x and evaluate p at x, then with very high probability, $p(x) \neq 0$ which allows us to conclude that p is not a 0 polynomial. Sadly, this simple degree argument breaks down for polynomials in more than one variable. For example, consider the real polynomial $q = X_1X_2 \in \mathbb{R}[X_1X_2]$. The set of zeros of q is $\{(x_1, x_2) \in \mathbb{R}^2 : x_1 = 0 \text{ or } x_2 = 0\}$ which is an infinite set although the total degree of q is only 2. Interestingly, although the number of zeros of a polynomial in at least two variables can be infinite, the probability argument (that we used for univariate polynomials) still holds as the famous Schwartz-Zippel Lemma proves.

2.3 Schwartz-Zippel Lemma

Lemma 2.3.1. *Let $p(X_1, X_2, \dots, X_n)$ be a non-zero polynomial in n variables $X_1, X_2, \dots, X_n$ of total degree d over a field $\mathbb{F}$ and $S \subseteq \mathbb{F}$ be any finite set. Let $x_1, x_2, \dots, x_n$ be drawn independently and uniformly from S. Then we have the following.*

$$\Pr\Big[p(x_1, x_2, \dots, x_n) = 0\Big] \leqslant \frac{d}{|S|}$$

Proof. We will prove it by induction on n. For $n = 1$, the polynomial p is a univariate polynomial of degree d over the field $\mathbb{F}$. Then the result follows from the fact that p has at most d roots.

Let us now assume the statement for polynomials on $n - 1$ variables. Let $p(X_1, X_2 \dots, X_n)$ be any polynomial. Without loss of generality, let us assume that there exists at least one monomial in $p(X_1, X_2 \dots, X_n)$ where X_1 appears; if there exist no such monomial, then p is also a polynomial in $X_2, X_3, \dots, X_n$ and the result follows from the induction hypothesis. We sample $x_2, x_3, \dots, x_n$ independently and uniformly from S. Let $\mathcal{E}_1$ be the event that the univariate polynomial $p(X_1, x_2, x_3, \dots, x_n)$ is a non-zero polynomial. Let us assume that the event $\mathcal{E}_1$ has happened. We write the univariate polynomial $p(X_1, x_2, x_3, \dots, x_n)$ in reduced form as follows. Let us call the polynomial $p(X_1, x_2, x_3, \dots, x_n)$ as $f(X_1)$; suppose the degree of $f(X_1)$ be $k \leqslant d$.

$$f(X_1) = p(X_1, x_2, \dots, x_n) = \sum_{i=1}^{k} X_1^i q_i(x_2, x_3, \dots, x_n)$$

We now sample x_1 uniformly from S. Let $\mathcal{E}_2$ be the event that $f(x_1) = 0$. Then from induction base case, we have $\Pr[\mathcal{E}_2|\mathcal{E}_1] \leqslant k/|S|$. We now bound $\Pr[\overline{\mathcal{E}_1}]$. We first observe that there exists a monomial in p where the degree of X_1 is at least k (why?); let that monomial be $aX_1^{k'}r(x_2, x_3, \dots, x_n)$ where $k' \geqslant k$ and thus the total degree of the polynomial $r(x_2, x_3, \dots, x_n)$ is at most $d - k$ (why?). Hence, for $p(X_1, x_2, \dots, x_n)$ to be

zero, a necessary condition is that $r(x_2, x_3, \ldots, x_n) = 0$ which happens with probabilities at most $(d-k)/|S|$ thanks to induction hypothesis. Hence, we have $\Pr[\overline{\mathcal{E}_1}] \leqslant (d-k)/|S|$. We now bound the probability that $p(x_1, x_2, \ldots, x_n) = 0$ as follows.

$$\begin{aligned}
\Pr\left[p(x_1, x_2, \ldots, x_n) = 0\right] &= \Pr[\overline{\mathcal{E}_1}] + \Pr[\mathcal{E}_2|\mathcal{E}_1]\Pr[\mathcal{E}_1] \\
&\leqslant \Pr[\overline{\mathcal{E}_1}] + \Pr[\mathcal{E}_2|\mathcal{E}_1] \\
&\leqslant \frac{d-k}{|S|} + \frac{k}{|S|} \\
&= \frac{d}{|S|}
\end{aligned}$$

The first inequality follows from the fact that $\Pr[\mathcal{E}_1] \leqslant 1$. This concludes the proof of the lemma. □

With Schwartz-Zippel Lemma at hand, our randomized algorithm is quite straight forward. Let us assume that the degree d of the input polynomial $p(X_1, \ldots, X_n) \in \mathbb{F}[X_1, \ldots, X_n]$ be strictly less than $|\mathbb{F}|$, that is $d < |\mathbb{F}|$. The algorithm samples $x_1, \ldots, x_n$ uniformly and independently from $\mathbb{F}$ and outputs that p is identically 0 if and only if $p(x_1, \ldots, x_n) = 0$. Observe that the algorithm never makes an error if the input polynomial is indeed identically 0. It can of course sometimes wrongly declare a non-zero polynomial as identically 0. These type of algorithms are said to have one sided error. It follows immediately from Schwartz-Zippel Lemma that the error probability of the algorithm is at most $\frac{d}{|\mathbb{F}|}$ (the error probability is 0 if the field $\mathbb{F}$ is an infinite filed, say $\mathbb{Q}$, $\mathbb{R}$, etc.; however is it possible to draw uniform sample from an infinite set? If not, then how we avoid drawing uniform samples from an infinite set?). We can improve the error probability by running the algorithm $\ell = \ln(|\mathbb{F}|/d)$ times (this is necessary when $|\mathbb{F}| < \infty$) and outputting that p is identically 0 if and only if p evaluates to 0 for every random points. Then the error probability of the algorithm becomes at most $(\frac{d}{|\mathbb{F}|})^{\ln(|\mathbb{F}|/d)} = \frac{1}{e}$.

2.4 Application of PIT: Perfect Bipartite Matching

We now see an application of PIT to the perfect bipartite matching problem. In the perfect bipartite matching problem, the input is a bipartite graph $G = (V = L \cup R, E)$ with its bi-partition of vertices being L and R and we need to compute if there exists a perfect matching in G. A matching $M \subseteq E$ is a set of edges with no two of them sharing any end point (vertex). A matching M is called perfect if, for every vertex $v \in V$ in the graph, there exists an edge $e \in M$ in the matching which is incident on v. Let us assume that we have $|L| = |R|$ since otherwise G clearly has no perfect matching.

There are many polynomial time algorithms for solving the perfect bipartite matching problem. Here we will see a randomized algorithm for it. The idea is to reduce the perfect bipartite matching problem to PIT and use the randomized algorithm for PIT.

One popular representation of any bipartite graph is by its *bi-adjacency matrix*. The bi-adjacency matrix A of the bipartite graph G is a matrix whose rows are indexed by L and columns are indexed by R. For $\ell \in L$ and $r \in R$, we have $A[\ell][r] = 1$ if $\{\ell, r\} \in E$ (that is there is an edge between ℓ and r); otherwise we have $A[\ell][r] = 0$.

We now reduce the perfect bipartite matching problem to PIT. From A, let us construct another matrix A' as $A'[\ell][r] = A[\ell][r]X_{\ell,r}$ for $\ell \in L$ and $r \in R$ where $X_{\ell,r}, \ell \in L, r \in R$ are indeterminates. Hence A' is a

matrix whose entries are either 0 or some indeterminates. We now define a polynomial $p(\{X_{\ell,r}, \ell \in \mathcal{L}, r \in \mathcal{R}\}) \in \mathbb{R}[\{X_{\ell,r}, \ell \in \mathcal{L}, r \in \mathcal{R}\}]$ to be the determinant of $\mathcal{A}'$. That is,

$$p = \sum_{\pi:\mathcal{L} \xrightarrow{\approx} \mathcal{R}} \text{sign}(\pi) \prod_{\ell \in \mathcal{L}} \mathcal{A}'_{\ell,\pi(\ell)}$$

The following easy observation is central to our reduction.

Observation 2.4.1. *p is a non-zero polynomial if and only if there is a perfect bipartite matching in $\mathcal{G}$.*

Proof. (if part) Let $\mathcal{M}$ be a perfect matching in $\mathcal{G}$. The perfect matching $\mathcal{M}$ naturally defines a bijection π from $\mathcal{L}$ to $\mathcal{R}$ as follows: $\pi(\ell) = r$ if $\{\ell, r\} \in \mathcal{M}$, for $\ell \in \mathcal{L}$. It follows from the definition of a perfect bipartite matching that π is well defined and it is a bijection. We observe that the coefficient of the monomial $\prod_{\ell \in \mathcal{L}} \mathcal{A}'_{\ell,\pi(\ell)}$ is 1 in p. Hence p is a non-zero polynomial.

(Only if part) Suppose p is a non-zero polynomial. We observe that monomials of p does not cancel. So, there exists a bijection π from $\mathcal{L}$ to $\mathcal{R}$ such that $\mathcal{A}'_{\ell,\pi(\ell)} \neq 0$ for every $\ell \in \mathcal{L}$. Hence the set of edges $\{\{\ell, \pi(\ell)\} : \ell \in \mathcal{L}\}$ forms a perfect matching in $\mathcal{G}$. □

Hence our algorithm for computing whether a bipartite graph has a perfect matching is to use our randomized algorithm for PIT to test whether p is a zero polynomial. Note that, we do not need to explicitly write p as a sum of monomials (actually doing so may take exponential time and thus would not be efficient). For our randomized algorithm for PIT to work, we only need to be able to evaluate the polynomial p at some given points which boils down to computing the determinant of a $|\mathcal{L}| \times |\mathcal{R}|$ matrix which can be done in time $O(|\mathcal{L}|^\omega)$ where $O(n^\omega)$ is the time complexity for matrix multiplication.

2.5 Karger's Randomized Min-Cut Algorithm

A cut of an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is a non-empty subset $\mathcal{U} \subsetneq \mathcal{V}, \mathcal{U} \neq \emptyset$ of vertices. The size of a cut $\mathcal{U}$, denoted by $w(\mathcal{U})$ is the number of the edges whose exactly one end-point belongs to $\mathcal{U}$, that is $w(\mathcal{U}) = |\{\{x, y\} \in \mathcal{E} : x \in \mathcal{U}, y \neq \mathcal{U}\}|$. We can have parallel edges in $\mathcal{G}$ but not self-loops since self-loops do not contribute to cut sizes. In the min-cut problem, we are given an unweighted undirected graph and the goal is to compute a cut of minimum size. There is an easy flow-based algorithm for this problem which runs in polynomial-time. We will now see another simple randomized algorithm for this problem due to David Karger. This algorithm uses the operation called edge-contraction in a graph. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be any graph and $e = \{x, y\} \in \mathcal{E}$ any edge. We now delete the vertices x and y from $\mathcal{G}$ (which in turn removes all edges incident on x or y), and add another vertex, say v_{xy}, and make it adjacent to every other vertex who was adjacent to either x or y in $\mathcal{G}$. We denote the resulting graph by $\mathcal{G}/e$. Note that $\mathcal{G}/e$ can have parallel edges even when $\mathcal{G}$ does not have any parallel edge. We now describe the Karger's min-cut algorithm.

Pick an edge e uniformly randomly from the set of edges, contract it, and repeat until we have only two vertices left. Let $\mathcal{U}$ and $\mathcal{V} \setminus \mathcal{U}$ be the set of input vertices which eventually contracted to the final two vertices by the algorithm. Output $\mathcal{U}$ as a minimum-cut.

We now analyze the error probability of this algorithm. Let $\mathcal{X} \subsetneq \mathcal{V}$ be a minimum cut, $\mathcal{F} \subseteq \mathcal{E}$ the set of edges crossing $\mathcal{X}$, and $|\mathcal{F}| = k$. Let p_n be the probability that the algorithm does not pick any of the edges in $\mathcal{F}$ in the first iteration where n is the number of vertices of the graph. Then we have

$$p_n = 1 - \frac{|\mathcal{F}|}{|\mathcal{E}|} = 1 - \frac{k}{|\mathcal{E}|}.$$

Let d be the minimum degree of $\mathcal{G}$. Then we have $k \leqslant d$ (why?) and $|\mathcal{E}| \geqslant \frac{nd}{2}$. Hence, we have

$$p_n = 1 - \frac{k}{|\mathcal{E}|} \geqslant 1 - \frac{2d}{nd} = 1 - \frac{2}{n} = \frac{n-2}{n}.$$

Now the probability that the algorithm outputs $\mathcal{F}$ is

$$\prod_{i=3}^{n} p_i = \prod_{i=3}^{n} \frac{i-2}{i} = \frac{2}{n(n-1)}.$$

By repeating the algorithm $\frac{n(n-1)}{2}$ times, the probability that the algorithm outputs $\mathcal{F}$ is at least

$$\left(1 - \frac{2}{n(n-1)}\right)^{\frac{n(n-1)}{2}} \geqslant \frac{1}{e}.$$

One "run" of Karger's algorithm can be "simulated" using one run of Kruskal's algorithm for finding a minimum spanning tree. Hence, one run of Karger's algorithm can be implemented in $\mathcal{O}(m \log n)$ time. Since we need to repeate $\binom{n}{2}$ times, the overall running time of the algorithm is $\mathcal{O}(n^2 m \log n)$. David Karger and Clifford Stein build upon this algorithm to design a faster algorithm which we see in the next section.

We can observe an important corollary from the error bound of Kerger's algorithm — the maximum number of minimum cuts in an undirected graph can be at most $\binom{n}{2}$ (how?). Notice that this bound is indeed tight; a cycle graph on n vertices has $\binom{n}{2}$ minimum cuts.

2.6 Karger-Stein Algorithm for Min-Cut

Consider a "run" of Karger's algorithm. Let $\mathcal{U}$ be any min-cut and $\mathcal{F}$ its edges. The probability that the algorithm does not pick any edge from $\mathcal{F}$ in the first t iterations is at least

$$\frac{(n-t)(n-t-1)}{n(n-1)}$$

which is at least $1/2$ for $t \leqslant n - \frac{n}{\sqrt{2}} - 1$. That is, the probability that the Karger's algorithm does not output the cut $\mathcal{U}$ is small (at least some constant) till the first $n - \frac{n}{\sqrt{2}} - 1$ iterations and grows rapidly after that. The idea of Karger and Stein is to run Karger's algorithm for $n - \frac{n}{\sqrt{2}} - 1$ iterations, repeat the last $\frac{n}{\sqrt{2}} - 1$ iterations twice, and the best solution. Its pseudo-code is as follows.

1. If $\mathcal{G}$ has two (super)nodes, then output the cut.
2. Run contraction algorithm for $n - \frac{n}{\sqrt{2}} - 1$ iterations. Let $\mathcal{G}'$ be the resulting graph.
3. $\mathcal{U}_1$: recursively compute a min-cut of $\mathcal{G}'$.
4. $\mathcal{U}_2$: recursively compute a min-cut of $\mathcal{G}'$.
5. Output best of $\mathcal{U}_1$ and $\mathcal{U}_2$.

Let us first analyze the running time $T(n)$ of the algorithm (excluding the time to output which takes $\mathcal{O}(n)$). Using adjacency matrix, we can perform each contraction in $\mathcal{O}(n)$ time. We can also pick a random edge in $\mathcal{O}(\log n)$ time. Hence, we have the following recurrence.

$$T(n) = \begin{cases} 2T\left(\frac{n}{\sqrt{2}}\right) + \mathcal{O}(n^2 \log n) & \text{when } n \geqslant 3 \\ \mathcal{O}(1) & \text{otherwise} \end{cases}$$

The solution of the above recurrence, using master method, is $T(n) = \mathcal{O}(n^2 \log^2 n)$. We now analyze the error probability of this algorithm.

The recursion tree has depth $\log_{\sqrt{2}} n = \mathcal{O}(\log n)$. We also observe that the recusrion tree is a complete binary tree. The algorithm does nto make an error (that is, it outputs $\mathcal{U}$) if and only if there exists a path from the root to any leaf node where no node makes an error (that is, no node picks any edge from $\mathcal{F}$). Let x be any node in the recursion tree at height d (height of leaf nodes are zero) and the probability that there is a path from x to any leaf in the sub-tree rooted at x where no node makes an error, be $p(d)$. The success probability of the algorithm is $p(\log_{\sqrt{2}} n)$. We have $p(0) = 1$. We now have the following recurrence for $p(d)$.

$$p(d) \geqslant \frac{1}{2}(1 - (1 - p(d-1))^2) = p(d-1) - \frac{1}{2}p(d-1)^2$$

We claim that $p(d) \geqslant \frac{1}{d+2}$ for every $d \geqslant 0$. We prove this using induction. The base case of $d = 0$ holds since we have $p(0) = 1 \geqslant 1/2$. Inductively, let us assume that $p(d-1) \geqslant 1/(d+1)$. We now have

$$p(d) \geqslant p(d-1) - \frac{1}{2}p(d-1)^2 \geqslant \frac{1}{d+1} - \frac{1}{2(d+1)^2} \geqslant \frac{1}{d+1} - \frac{1}{(d+1)(d+2)} = \frac{1}{(d+2)}.$$

Hence, the success probability of the algorithm is $p(\log_{\sqrt{2}} n) = \Omega\left(\frac{1}{\log n}\right)$. We repeat the algorithm $\Theta(\log n)$ times to boost the success probability to $\Omega(1)$. The overall running time of the algorithm is $\mathcal{O}(n^2 \log^3 n)$.

2.7 Analysis of Randomized Quick Sort

In the sorting problem, the input is a sequence of n numbers, and our goal is to arrange these n numbers in non-decreasing order. We know many efficient sorting algorithms like merge sort which makes $\mathcal{O}(n \log n)$ comparisons. Another popular and practical sorting algorithm is the randomized quick sort. On a high level, the randomized quick sort algorithm picks a random number x from the set of numbers given as input, partition the input numbers with x as pivot, and recursively sorts the set of numbers less than x and the set of numbers greater than x. We refer to any standard text book on algorithms for detailed description of the randomized quick sort algorithm. Observe that the quick sort algorithm is a Las Vegas type randomized algorithm. We will now prove that the expected number of comparisons that the randomized quick sort algorithm makes on any input is $\mathcal{O}(n \log n)$. Let the input sequence of numbers be $a_1, a_2, \ldots, a_n$. Let a non-decreasing sequence of these numbers be $a'_1, a'_2, \ldots, a'_n$; that is, $a'_1 \leqslant a'_2 \leqslant \cdots \leqslant a'_n$. We make the following observation about the randomized quick sort algorithm.

Observation 2.7.1. *Comparisons between numbers are made only in the partition sub-routine of the quick sort algorithm. Moreover, for any $1 \leqslant i < j \leqslant n$, the numbers a'_i and a'_j are compared (exactly once) if and only if the first pivot selected from the set $\{a_\ell : i \leqslant \ell \leqslant j\}$ is either a'_i or a'_j.*

Proof. Immediate from the algorithm for partition. □

For $1 \leqslant i < j \leqslant n$, we define a Bernoulli random variable $X_{i,j}$ which takes value 1 if the numbers a'_i and a'_j are compared and 0 otherwise. Since pivot is selected uniformly at random, the probability that $X_{i,j}$ takes value 1 is $2/(j-i+1)$. Then we have $\mathbb{E}[X_{i,j}] = 2/(j-i+1)$. Let us denote by X the number of comparisons that the quick sort algorithm makes on the above input. Then we have the following.

$$
\begin{aligned}
\mathcal{X} &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j} \\
\Rightarrow \mathbb{E}[\mathcal{X}] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{i,j}] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\
&\leqslant \sum_{i=1}^{n-1} 2 \ln n \\
&\leqslant 2n \ln n
\end{aligned}
$$

Hence, we have proved the following result.

Theorem 2.7.1. *On any input, the expected number of comparisons that the randomized quick sort algorithm makes is at most* $2n \ln n = \mathcal{O}(n \log n)$.

We show next that $\text{var}(\mathcal{X})$ is $\mathcal{O}(n^2)$. We have

$$
\text{var}(\mathcal{X}) \leqslant \sum_{1 \leqslant i,j \leqslant n} \textit{var}(X_{ij}) + \sum_{1 \leqslant i,i',j,j' \leqslant n, i<j, i'<j'} \text{cov}(X_{ij}, X_{i'j'}).
$$

Observe that $\text{cov}(X_{ij}, X_{i'j'})$ is zero if $[i,j] \cap [i',j']$ is empty. Also observe that $\text{cov}(X_{ij}, X_{i'j'})$ is negative (that is, they are negatively correlated) if $[i,j] \cap [i',j']$ is non-empty and i, j, i', and j' are all different: if one of them take value one, then the probability that the other variable also takes value one reduces. Hence, we have

$$
\begin{aligned}
\text{var}(\mathcal{X}) &\leqslant \sum_{1 \leqslant i,j \leqslant n} \text{var}(X_{ij}) + \sum_{1 \leqslant i,i',j,j' \leqslant n, i<j, i'<j'} \text{cov}(X_{ij}, X_{i'j'}) \\
&\leqslant n^2 + \sum_{1 \leqslant i,i',j,j' \leqslant n, i<j, i'<j'} \text{cov}(X_{ij}, X_{i'j'}) && \text{[Since variance is at most 1 here.]} \\
&\leqslant n^2 + 2 \sum_{i=1}^{n} \sum_{n \geqslant j' > j > i} \text{cov}(X_{ij}, X_{ij'}) && \text{[Follows from argument above.]} \\
&\leqslant n^2 + 2 \sum_{i=1}^{n} \sum_{n \geqslant j' > j > i} \mathbb{E}[X_{ij} X_{ij'}] && \text{[Follows from definition of covariance.]} \\
&= n^2 + 2 \sum_{i=1}^{n} \sum_{n \geqslant j' > j > i} \left[\frac{1}{j'-i+1} + \frac{1}{(j'-i+1)(j-i+1)} \right] \\
&\leqslant n^2 + 2 \sum_{i=1}^{n} \sum_{n \geqslant j' > j > i} \frac{2}{j'-i+1}
\end{aligned}
$$

$$\leqslant n^2 + 2\sum_{i=1}^{n}\sum_{n\geqslant j'>i} 2$$
$$\leqslant 3n^2$$

2.8 Color Coding Technique

Color coding is an algorithm design technique. The high-level idea is to randomly color "the instance" to induce more structure on the problem instance which may help us design the algorithm easily. Let us understand this with a couple of examples.

2.8.1 Color Coding Based Algorithm for Longest Path

In the Longest Path problem, the input is an unweighted (directed or undirected) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an integer k. Here we need to compute if there exists a simple path in $\mathcal{G}$ of length at least k integers. Let us define a "colorful" version of the Longest Path problem. Suppose each vertex of the input graph $\mathcal{G}$ is colored with one of k colors (this coloring is nothing to do with vertex coloring where every edge must see different colors). The goal is to compute a "colorful" path of length (if it exists). A colorful path is a path consisting of one vertex of each color.

We now see that finding a colorful path is much simpler job.

Lemma 2.8.1. *There is a deterministic algorithm for the Colorful Longest Path problem with running time* $\mathcal{O}\left(2^k n^{\mathcal{O}(1)}\right)$.

Proof. Let $\chi : \mathcal{V} \longrightarrow [k]$ be a coloring of $\mathcal{G}$; the color classes be $(\mathcal{V}_1, \ldots, \mathcal{V}_k)$. We now explain a dynamic programming based algorithm. For a subset $\mathcal{S} \subseteq [k]$ and a vertex $u \in \cup_{i\in\mathcal{S}}\mathcal{V}_i$, we define a Boolean value $P[\mathcal{S}, u]$ to be true if and only if there is a colorful path of length $|\mathcal{S}|$ consisting exactly one vertex of color i for every $i \in \mathcal{S}$ and the path ends at vertex u. We update $P[\mathcal{S}, u]$ as follows.

$$P[\mathcal{S}, u] = \begin{cases} \bigvee\{P[\mathcal{S} \setminus \{\chi(u)\}, v] : vu \in \mathcal{E}\} & \text{if } \chi(u) \in \mathcal{S} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

The correctness of the above update rule follows from the fact that if there indeed exists a colorful path with colors in $\mathcal{S}$ ending at u then there also exists a colorful path with colors in $\mathcal{S} \setminus \chi(u)$ ending at some neighbor of u. Clearly there is a colorful path of length k is there exists a vertex v such that $P([k], u)$ is TRUE. □

Lemma 2.8.1 provides an algorithm with running time $\mathcal{O}\left(2^k n^{\mathcal{O}(1)}\right)$ for the Colorful Longest Path problem. However, how do we color the vertices — recall, our original goal is to design an algorithm for the Longest Path problem. What coloring do we desire? We want a coloring so that, if the graph indeed has a path of length k, then the colored graph also has a colorful path of length k. However how do we color vertices ensuring this? Here comes the magic of randomization as the following lemma shows.

Lemma 2.8.2. *Let* $\mathcal{U}$ *be a universe of size* n *and* $\mathcal{X} \subseteq \mathcal{U}$ *a subset of* $\mathcal{U}$ *of size* k. *Let* $\chi : \mathcal{U} \longrightarrow [k]$ *be a coloring where the color of each element is chosen uniformly at random from* $[k]$ *independent of everything else. Then the probability that the elements of* $\mathcal{X}$ *receive all the distinct* k *colors is at least* e^{-k}.

Proof. Among k^n possible colorings, $k!k^{n-k}$ of the colorings assign pairwise different colors to X. Since $k! > (k/e)^k$, the result follows. □

Using Lemma 2.8.2 and Lemma 2.8.1, we get the following.

Theorem 2.8.1. *There is a randomized algorithm for the Longest Path problem with running time* $O\left((2e)^k\right)$. *The algorithm never makes mistake for* NO *instances.*

Proof. We randomly color the vertices and run the algorithm in Lemma 2.8.1. If the input graph indeed has a path of length k, then we find one such path in time $O\left(2^k n^{O(1)}\right)$ with probability at least e^{-k}. We repeat the above step e^k times. The probability that the input graph has a path of length k and still we do not find any such path is at most

$$\left(1 - e^{-k}\right)^{e^k} < \frac{1}{e}$$

□

Chapter 3

Standard Concentration Bounds

We have seen that the expected number of comparisons that the randomized quick sort algorithm makes on any input is $\mathcal{O}(n \log n)$. However, this statement does not say anything about the probability that the randomized quick sort algorithm will make say 10 times the expected number of comparisons. In general, we would like to have the expected cost of our algorithm low and there should be significant probability mass around the expectation. This would allow us making statements like the cost of our algorithm is at most something with high probability. We will now see an array to probabilistic tools which would allow us to make these kind of statements. Our first tool is the classical Markov inequality.

3.1 Markov Inequality

Theorem 3.1.1. *Let X be a* non-negative *random variable (that is, X never takes any negative value) with finite expectation $\mathbb{E}[X]$. Then for any positive real number c, we have the following.*

$$\Pr\left[X \geqslant c\right] \leqslant \frac{\mathbb{E}[X]}{c}$$

Equivalently, we have the following

$$\Pr\left[X \geqslant c\mathbb{E}\left[X\right]\right] \leqslant \frac{1}{c}$$

Proof. Let us prove the result when X is a discrete random variable. Since X is a non-negative random variable, we have the following.

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{i \in \mathrm{Sup}(X)} i \Pr[X = i] \\
&\geqslant \sum_{i \in \mathrm{Sup}(X), i \geqslant c} i \Pr[X = i] \\
&\geqslant \sum_{i \in \mathrm{Sup}(X), i \geqslant c} c \Pr[X = i] \\
&= c \sum_{i \in \mathrm{Sup}(X), i \geqslant c} \Pr[X = i] \\
&= c \Pr[X \geqslant c]
\end{aligned}$$

□

An important point to note is that the Markov inequality is applicable only to non-negative random variables (can you find an example of a random variable which takes negative values with non-zero probability where Markov inequality fails?) Using only the knowledge of the expectation of a random variable, Markov inequality is tight as the following example shows. Let $c > 1$ be any real number. Consider a random variable X which takes value 0 with probability $(c-1)/c$ and 1 with probability $1/c$.

We observe that, by applying Markov inequality on Theorem 2.7.1, we get that the probability that, on any input, the randomized quick sort algorithm makes more than $100n \ln n$ number of queries is $1/50$.

3.2 Chebyshev Inequality

We may significantly improve the guarantee that Markov inequality provides if we know the variance of the random variable. This is popularly known as Chebyshev inequality. Interestingly, Chebyshev inequality can be easily proved by applying Markov inequality.

Theorem 3.2.1 (Chebyshev Inequality)**.** *Let X be a random variable with finite expectation μ and variance σ^2. Then for any any positive real number c, we have the following.*

$$\Pr[|X - \mu| \geqslant c] \leqslant \frac{\sigma^2}{c^2}$$

Proof. We have the following chain of inequalities.

$$\begin{aligned}
\Pr[|X - \mu| \geqslant c] &= \Pr[(X-\mu)^2 \geqslant c^2] \\
&\leqslant \frac{\mathbb{E}[(X-\mu)^2]}{c^2} && \text{[Applying Markov inequality to the r.v. } (X-\mu)^2\text{]} \\
&= \frac{\sigma^2}{c^2} && \text{[From the definition of variance]}
\end{aligned}$$

□

Let X be a random variable denoting the number of comparisons that the randomized quick sort algorithm makes on an input of size n. It can be proved (easily) that the variance of X is $\Theta(n^2)$. Then, by using Chebyshev inequality, we bound the probability that the randomized quick sort algorithm performs more than $2.1n \ln n$ comparisons as follows.

$$\Pr[X \geqslant 2.1n \ln n] \leqslant \Pr[|X - \mathbb{E}[X]| \geqslant 0.1n \ln n] \leqslant \frac{\text{var}(X)}{\Theta(n^2 \ln^2 n)} = \frac{1}{\Theta(\ln^2 n)}$$

Hence, we have just proved the following result. Observe that the Chebyshev inequality gives much stronger bound for the number of comparisons of the randomized quick sort algorithm than the Markov inequality.

Theorem 3.2.2. *On any input, the probability that the randomized quick sort algorithm makes more than $(2+\varepsilon)n \ln n$ comparisons is at most $\frac{1}{\varepsilon^2 \ln^2 n}$*[1].

[1]The probability can actually be proved to be equal to $n^{-(2+o(1))\varepsilon \ln \ln n}$ [MH96] using more sophisticated analysis.

3.3 Chernoff Bounds

Note that, although Chebyshev inequality provides stronger guarantee than the Markov inequality, it makes stronger assumption (existence of variance which is equivalent to assuming $\mathbb{E}[X^2] < \infty$) than Markov inequality. One may as well make even stronger assumption that $\mathbb{E}[|X|^n] < \infty$ for every $n \in \mathbb{N}$ (that is the random variable under consideration has all the moments) and prove even stronger concentration bound. This idea is realized in Chernoff bounds (note the plurality here; Chernoff bound, unlike Markov and Chebyshev inequalities, refer to a type of bounds). Interesting, proof of Chernoff bounds also uses Markov inequality.

Let $X_i, i \in [n]$ be n pairwise independent random variables each taking values in $\{0, 1\}$ with expectation μ_i and $S = \sum_{i=1}^{n} X_i$. This basic set up can be generalized in some ways like the range of the random variable X_i could be $[a_i, b_i]$ with $b_i - a_i$ is finite for every $i \in [n]$ and/or the expectation of the random variable X_i is μ_i. However, for sake of simplicity, let us restrict ourselves to the basic setting above. The techniques can be straightforwardly extended to more general set ups.

Using linearity of expectation, we have that $\mathbb{E}[S] = n\mu$. The general Chernoff bound is as follows.

Theorem 3.3.1 (Chernoff Bound (general form))**.** *Let $X_i, i \in [n]$ be n independent random variables each taking values in $\{0, 1\}$ and $S = \sum_{i=1}^{n} X_i$; let $\mathbb{E}[S] = \mu$. Then for any positive real number δ we have the following.*

$$\Pr\left[S \geqslant (1+\delta)\mu\right] \leqslant \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

and

$$\Pr\left[S \leqslant (1-\delta)\mu\right] \leqslant \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu}$$

Proof. Let us first prove the upper bound. Let α be any positive real number.

$$\begin{aligned}
\Pr\left[S \geqslant (1+\delta)\mu\right] &= \Pr\left[e^{\alpha S} \geqslant e^{\alpha(1+\delta)\mu}\right] && \text{[Since } e^{\alpha x} \text{ is an increasing function]} \\
&\leqslant \frac{\mathbb{E}[e^{\alpha S}]}{e^{\alpha(1+\delta)\mu}} && \text{[by Markov inequality]} \\
&= \frac{\mathbb{E}[e^{\alpha \sum_{i=1}^{n} X_i}]}{e^{\alpha(1+\delta)\mu}} \\
&= \frac{\prod_{i=1}^{n} \mathbb{E}[e^{\alpha X_i}]}{e^{\alpha(1+\delta)\mu}} && \text{[independence]} \\
&= \frac{\prod_{i=1}^{n} (e^{\alpha} p_i + (1 - p_i))}{e^{\alpha(1+\delta)\mu}} && \text{[let } \Pr[X_i = 1] = p_i\text{]} \\
&= \frac{\prod_{i=1}^{n} ((e^{\alpha} - 1)p_i + 1)}{e^{\alpha(1+\delta)\mu}} \\
&\leqslant \frac{\prod_{i=1}^{n} e^{(e^{\alpha}-1)p_i}}{e^{\alpha(1+\delta)\mu}} && [1 + x \leqslant e^{x}, x \in \mathbb{R}] \\
&= \frac{e^{(e^{\alpha}-1)\sum_{i=1}^{n} p_i}}{e^{\alpha(1+\delta)\mu}} \\
&= e^{((e^{\alpha}-1) - \alpha(1+\delta))\mu} && \left[\mu = \sum_{i=1}^{n} p_i\right]
\end{aligned}$$

The above inequality holds for any real number $\alpha > 0$. Hence, to make the above inequality tightest, we would like to pick some α which minimizes $f(\alpha) = (e^{\alpha} - 1) - \alpha(1+\delta)$. We have $f'(\alpha) = e^{\alpha} - (1+\delta), f''(\alpha) =$

$e^{\alpha} > 0$ for every real number $\alpha > 0$. Hence, the function $f(\alpha)$ is minimized for $\alpha = \ln(1+\delta)$. By putting $\alpha = \ln(1+\delta)$ in the above inequality, we have the following.

$$\Pr\left[S \geqslant (1+\delta)\mu\right] \leqslant \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

Performing similar calculation for the lower bound, we can obtain the desired lower bound (Home work). □

As we can see, the general form of the Chernoff bound is not very intuitive and may be harder to apply/remember in practice. To address that, we now see various simplified useful (of course less general and weaker) forms of the general Chernoff bound. All these forms can be proved by elementary calculus (and can be observed by simply plotting relevant functions in any software).

▷ **[Multiplicative form (useful for small δ)]** For $\delta \in [0,1]$

$$\Pr\left[S \geqslant (1+\delta)\mu\right] \leqslant e^{-\frac{\delta^2\mu}{3}}, \Pr\left[S \leqslant (1-\delta)\mu\right] \leqslant e^{-\frac{\delta^2\mu}{2}}$$

▷ **[Additive form for large deviation only (not applicable for small deviation)]** For any $R \geqslant 2e\mu$,

$$\Pr\left[S \geqslant R\right] \leqslant 2^{-R}$$

▷ **[Two-sided form]** For $\delta \in [0,1]$

$$\Pr\left[|S-\mu| \geqslant \delta\mu\right] \leqslant 2e^{-\frac{\delta^2\mu}{3}}$$

▷ **[Large deviation]** For any $k > 1$

$$\Pr\left[S \geqslant k\mu\right] \leqslant \left(\frac{e^{k-1}}{k^k}\right)^{\mu}$$

3.4 Application

We will now see few applications of these bounds.

3.5 Flipping Coin

Suppose we toss a coin n times which comes up head with probability $p \in (0,1)$. Let X_i be a indicator random variable for the event that the i-th coin toss comes up head for $i \in [n]$. Let X be a random variable denoting the number of times the coin comes up head. Then we have, $\mathbb{E}[X_i] = p, X = \sum_{i=1}^{n} X_i, \mathbb{E}[X_i] = pn$. Then, using Chernoff bound, we have that $\Pr\left[|X - pn| \geqslant \delta pn\right] \leqslant 2e^{-\delta^2 pn/2}$. In particular, for $\delta = c/(p\sqrt{n})$, we have $\Pr\left[|X - pn| \geqslant c\sqrt{n}\right] \leqslant 2e^{-c^2/2p} \leqslant 2e^{-c^2/2}$.

3.6 Coupon Collector's Problem and Union Bound

Suppose we have a bag containing n different coupons. At every step, we draw a random coupon from the bag. We would like to know how many times we need to draw coupons to observe all the n different coupons. For $i \in [n]$, let X_i be a random variable denoting the number of coupons we draw after observing

$i-1$ different coupons till we observe another new coupon. Let X be a random variable which denotes the number of time we draw coupons till we observe all the coupons. Then we have $X = \sum_{i=1}^{n} X_i$. We observe that X_i is a geometric random variable with parameter $(n-i+1)/n$ (in this case, it is the probability of observing a new coupon in next draw). Then, we have $\mathbb{E}[X_i] = n/(n-i+1)$[2] and, by linearity of expectation, $\mathbb{E}[X] = n\sum_{i=1}^{n} 1/(n-i+1) = n\sum_{i=1}^{n} 1/i = nH_n \leqslant n(1+\ln n) = O(n \ln n)$.

Using Markov inequality, we have that $\Pr\left[X \geqslant 2nH_n\right] \leqslant 1/2$. We now see how we will get much stronger bound by using Chebyshev inequality. For that, let us first compute the variance of X. We first observe that the random variables $X_i, i \in [n]$ are pairwise independent. Hence, we have the following.

$$\begin{aligned}
\text{var}(X) &= \sum_{i=1}^{n} \text{var}(X_i) && \text{[Since } X_i, i \in [n] \text{ are independent]} \\
&\leqslant \sum_{i=1}^{n} \left(\frac{n}{n-i+1}\right)^2 && \left[\text{variance of a geometric r.v. with parameter } p \text{ is } \frac{(1-p)}{p^2}\right] \\
&\leqslant n^2 \sum_{i=1}^{\infty} \frac{1}{i^2} \\
&= \frac{n^2\pi^2}{6}
\end{aligned}$$

Now by Chebyshev inequality, we have $\Pr\left[X \geqslant 2nH_n\right] \leqslant \Pr\left[|X - \mathbb{E}[X]| \geqslant nH_n\right] \leqslant O(1/\ln^2 n)$. However, we can significantly improve the bound above by using simple union bound as follows. Let us call the n different coupons as $C_i, i \in [n]$. Suppose we draw k times. Let $\mathcal{E}_i$ be the event that the coupon C_i is never been observed. Then we have the following.

$$\Pr\left[\mathcal{E}_i\right] = \left(1 - \frac{1}{n}\right)^k \leqslant e^{-k/n}$$

Now using union bound, we have the following.

$$\Pr\left[\cup_{i=1}^{n} \mathcal{E}_i\right] \leqslant \sum_{i=1}^{n} \Pr[\mathcal{E}_i] = ne^{-k/n}$$

Hence, for $k = 2nH_n$, we have the following.

$$\Pr\left[\cup_{i=1}^{n} \mathcal{E}_i\right] \leqslant ne^{-2H_n} \leqslant \frac{2}{n}$$

Union Bound: For any finite collection of events $A_i, i \in [n]$, the union bound states that $\Pr\left[\bigcup_{i=1}^{n} A_i\right] \leqslant \sum_{i=1}^{n} \Pr\left[A_i\right]$.

3.7 Balls and Bins, Birthday Paradox

Suppose we randomly put n balls into m bins. That is, we assign each ball to one of the m bins uniformly randomly. Observe that this simple setting models important problems like hashing a universe of size n to

[2]Expectation of a geometric random variable with parameter p is $1/p$.

another universe of size m, etc. We would like to analyze various events of this experiment. For example, how many balls we need to throw to m bins so that the probability of collision is at most $1/2$? With fix n and m, how many balls the heaviest bin contains?

3.7.1 Probability of Collision: Birthday Paradox

Let us begin with the first example which would lead us to the famous *birthday paradox*. Let $\mathcal{C}$ be the event that there is a collision. Then using elementary probabilistic argument, we have the following.

$$\Pr\left[\mathcal{C}\right] = \left(1 - \frac{1}{m}\right)\left(1 - \frac{2}{m}\right)\cdots\left(1 - \frac{n-1}{m}\right)$$

Using the inequality $1 + x \leqslant e^x$ for every $x \in \mathbb{R}$, we bound $\Pr\left[\mathcal{C}\right]$ as below.

$$\Pr\left[\mathcal{C}\right] \leqslant \exp\left\{-\sum_{i=1}^{n-1}\frac{i}{m}\right\} = \exp\left\{-\frac{n(n-1)}{2m}\right\} \leqslant \exp\left\{-\frac{n^2}{2m}\right\}$$

From above bound, we observe that for $n = \sqrt{2m}$, we have $\Pr\left[\mathcal{C}\right] \leqslant 1/e$. Consider in a party, n people have gathered. Suppose that their birthdays are uniformly distributed over the years. Then, for $n \geqslant 23$, we have that $\Pr\left[\mathcal{C}\right] \leqslant 1/2$. This fact is known as birthday paradox.

3.7.2 Expected Maximum Load

Let us assume that $n = m$. Let $X_i, i \in [n]$ be the random variable denoting the number of balls in the i-th bin. We now bound the probabilities that $X_i \geqslant k$ for any positive integer k with $1 \leqslant k \leqslant n$; that is the i-th bin contains at least k balls.

$$\Pr\left[X_i \geqslant k\right] \leqslant \binom{n}{k}\left(\frac{1}{n}\right)^k \leqslant \left(\frac{ne}{k}\right)^k\left(\frac{1}{n}\right)^k = \left(\frac{e}{k}\right)^k$$

The first inequality follows from union bound and the second inequality follows from that fact that $\binom{n}{k} \leqslant \left(\frac{ne}{k}\right)^k$ which can be proved by Starling's inequality. Using the above inequality, we now bound maximum load on any bin.

Theorem 3.7.1. *With probability at least $1 - \frac{1}{n}$, every bin has at most $\frac{3\ln n}{\ln\ln n}$ balls.*

Proof. By putting $k = \frac{3\ln n}{\ln\ln n}$ in the above inequality.

$$\begin{aligned}
\Pr\left[X_i \geqslant \frac{3\ln n}{\ln\ln n}\right] &\leqslant \left(\frac{e\ln\ln n}{3\ln n}\right)^{\frac{3\ln n}{\ln\ln n}} \\
&\leqslant \left(\frac{e\ln\ln n}{\ln n}\right)^{\frac{3\ln n}{\ln\ln n}} \\
&= \exp\left\{\frac{3\ln n}{\ln\ln n}\left(\ln\ln\ln n - \ln\ln n\right)\right\} \\
&= \exp\left\{-3\ln n + \frac{3\ln n\ln\ln\ln n}{\ln\ln n}\right\} \\
&\leqslant \exp\{-2\ln n\}
\end{aligned}$$

$$= \frac{1}{n^2}$$

Now using union bound, we have the following.

$$\Pr\left[\exists i \in [n], X_i \geqslant \frac{3\ln n}{\ln\ln n}\right] \leqslant \frac{1}{n}$$

□

Theorem 3.7.1 allows us to upper bound the expected value of maximum load.

Corollary 3.7.1. *Let X be the random variable denoting the maximum load of any bin in the balls and bin experiment with n balls and n bins. Then we have $\mathbb{E}[X] \leqslant \frac{3\ln n}{\ln\ln n} + 1$.*

Proof. Theorem 3.7.1 proves the following.

$$\Pr\left[X \leqslant \frac{3\ln n}{\ln\ln n}\right] \geqslant 1 - \frac{1}{n}$$

On the other hand, since there are only n balls, we have the following.

$$\Pr\left[n \geqslant X > \frac{3\ln n}{\ln\ln n}\right] \leqslant \frac{1}{n}$$

We now bound $\mathbb{E}[X]$ as follows.

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{x=1}^{n} x \Pr[X = x] \\
&= \sum_{x=1}^{\frac{3\ln n}{\ln\ln n}} x \Pr[X = x] + \sum_{x=\frac{3\ln n}{\ln\ln n}+1}^{n} x \Pr[X = x] \\
&\leqslant \frac{3\ln n}{\ln\ln n} \sum_{x=1}^{\frac{3\ln n}{\ln\ln n}} \Pr[X = x] + n \sum_{x=\frac{3\ln n}{\ln\ln n}+1}^{n} \Pr[X = x] \\
&= \frac{3\ln n}{\ln\ln n} \Pr\left[X \leqslant \frac{3\ln n}{\ln\ln n}\right] + n \Pr\left[n \geqslant X > \frac{3\ln n}{\ln\ln n}\right] \\
&\leqslant \frac{3\ln n}{\ln\ln n} + 1
\end{aligned}$$

□

We now show that the bound in Corollary 3.7.1 is tight up to constant factors. We will prove this by what is called the "second moment method." The second moment method is used to show that some non-negative random variable X takes non-zero values with high probability using the inequality that $\Pr[X = 0] \leqslant \frac{\mathrm{var}(X)}{\mathbb{E}[X]^2}$ which can be proved using Chebyshev inequality.

Lemma 3.7.1 (Second Moment Method)**.** *Let X be a non-negative random variable with finite variance. Then*

$$\Pr[X = 0] \leqslant \frac{var(X)}{\mathbb{E}[X]^2}$$

Proof. Using Chebyshev inequality, we have

$$\Pr[\mathcal{X} = 0] = \Pr[|\mathcal{X} - \mathbb{E}[\mathcal{X}]| \geqslant \mathbb{E}[\mathcal{X}]] \leqslant \frac{\mathrm{var}(\mathcal{X})}{\mathbb{E}[\mathcal{X}]^2}$$

□

On the other hand, the "first moment method" is used to prove that some non-negative random variable $\mathcal{X}$ takes the value zero with high probability using the inequality that $\Pr[\mathcal{X} = 0] \geqslant 1 - \mathbb{E}[\mathcal{X}]$ which can be proves using Markov inequality.

Theorem 3.7.2. *Let $\mathcal{X}$ be the random variable denoting the maximum load of any bin in the balls and bin experiment with n balls and n bins. Then we have $\mathbb{E}[\mathcal{X}] \geqslant \frac{\ln n}{3 \ln \ln n}$.*

Proof. Let $\mathcal{X}_i$ be the random variable denoting the number of balls in bin i. Then we have the following for any non-negative integer k.

$$\Pr[\mathcal{X}_i \geqslant k] \geqslant \binom{n}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \geqslant \left(\frac{n}{k}\right)^k \left(\frac{1}{n}\right)^k \frac{1}{e} = \frac{1}{ek^k}$$

The first inequality follows from elementary probability and the second inequality follows from the fact that $\binom{n}{k} \geqslant \left(\frac{n}{k}\right)^k$. For $k = \frac{\ln n}{3 \ln \ln n}$, the above inequality gives us the following.

$$\Pr\left[\mathcal{X}_i \geqslant \frac{\ln n}{3 \ln \ln n}\right] \geqslant \frac{1}{(\ln n)^{\frac{\ln n}{3 \ln \ln n}}} = n^{-\frac{1}{3}}$$

Let $\mathcal{Y}_i$ be the indicator random variable that the i-th bin contains at least $\frac{\ln n}{3 \ln \ln n}$ balls. Let $\mathcal{Y} = \sum_{i=1}^{n} \mathcal{Y}_i$. Then we have $\mathbb{E}[\mathcal{Y}_i] \geqslant n^{-\frac{1}{3}}$ and thus $\mathbb{E}[\mathcal{Y}] \geqslant n^{\frac{2}{3}}$ by linearity of expectation. We now upper bound the probability that $\mathcal{Y}$ takes value zero using the inequality $\Pr[\mathcal{Y} = 0] \leqslant \frac{\mathrm{var}(\mathcal{Y})}{\mathbb{E}[\mathcal{Y}]^2}$. For that, we now bound $\mathrm{var}(\mathcal{Y})$. We have $\mathrm{var}(\mathcal{Y}) = \sum_{i=1}^{n} \mathrm{var}(\mathcal{Y}_i) + \sum_{i \neq j} \mathrm{cov}(\mathcal{Y}_i, \mathcal{Y}_j)$. Observe that the variables are negatively correlated since the fact that some bins have more balls reduces the probability that some other bin also has more balls. Hence, we have $\mathrm{var}(\mathcal{Y}) \geqslant \sum_{i=1}^{n} \mathrm{var}(\mathcal{Y}_i)$. Since $\mathcal{Y}_i, i \in [n]$ are Bernoulli random variables, we have $\mathrm{var}(\mathcal{Y}_i) \leqslant 1$. Hence, we have $\mathrm{var}(\mathcal{Y}) \leqslant n$ and thus $\Pr[\mathcal{X} < \frac{\ln n}{3 \ln \ln n}] = \Pr[\mathcal{Y} = 0] \leqslant \frac{n}{n^{4/3}} = n^{-\frac{1}{3}}$. We now lower bound $\mathbb{E}[\mathcal{X}]$ as follows.

$$\begin{aligned}
\mathbb{E}[\mathcal{X}] &= \sum_{i=1}^{n} x \Pr[\mathcal{X} = x] \\
&\geqslant \sum_{i=\frac{3 \ln n}{\ln \ln n}+1}^{n} x \Pr[\mathcal{X} = x] \\
&\geqslant \left(\frac{3 \ln n}{\ln \ln n} + 1\right) \sum_{i=\frac{3 \ln n}{\ln \ln n}+1}^{n} \Pr[\mathcal{X} = x] \\
&\geqslant \left(\frac{3 \ln n}{\ln \ln n} + 1\right) \left(1 - n^{-\frac{1}{3}}\right) \\
&\geqslant \frac{3 \ln n}{\ln \ln n}
\end{aligned}$$

The last inequality holds for large enough n. □

3.8 Boosting Success Probability with Few Random Bits: Two Point Sampling

Suppose we have a randomized algorithm $\mathcal{A}$ for some decision problem Π that has one sided error. That is, for every YES instance x, the algorithm always answers correctly. On the other hand, if x is a NO instance, then the algorithm outputs correctly with probability at least $\frac{1}{2}$ (for example, our algorithm for PIT has this kind of one sided error). That is, whenever the algorithm outputs NO, the input instance is indeed a NO instance. These type of randomized algorithms are sometimes called zero error randomized algorithm. Suppose the algorithm $\mathcal{A}$ uses d random bits. The randomized algorithm $\mathcal{A}$ can be equivalently viewed as a deterministic algorithm which, along with usual input, also takes a bit string $r \in \{0,1\}^d$ as input (any randomized algorithm can be viewed like this). Suppose we wish to decrease the success probability of our algorithm from $\frac{1}{2}$ to $\frac{1}{\text{poly}(n)}$.

Since $\mathcal{A}$ always correctly answers YES instances, let us focus on NO instances only. Let x be any NO instance. Since the success probability of $\mathcal{A}$ is at least $1/2$, there exist at least 2^{d-1} possible $r \in \{0,1\}^d$ such that $\Pr[\mathcal{A}(x,r)] = \text{YES}$; we call these values of r as the *witnesses* of x. An immediate (obvious) approach to boost the success probability of $\mathcal{A}$ is to run $\mathcal{A}(x,r_k)$ for $r_k \in_R \{0,1\}^d, k \in [\lg n]$ and output YES if and only if $\mathcal{A}(x,r_k)$ outputs YES for every $k \in [\lg n]$. The probability that the algorithm outputs YES on a NO instance x is at most $\frac{1}{2^{\lg n}} = \frac{1}{n}$. This approach works perfectly unless randomness is scarce. This approach uses $d \lg n$ random bits. Can we bring down the error probability to $\frac{1}{n}$ by using less number of random bits? One way to do this is 2 point sampling.

The above boosting approach uses every random string $r_i, i \in [\lg n]$ only to compute $\mathcal{A}(x,r_i)$. The idea is to first sample two random strings $r_1, r_2 \in \{0,1\}^d$ and generate many "pseudo-random" strings $s_j, j \in [t]$ and compute $\mathcal{A}(x,s_j)$. However, since every $s_j, j \in [t]$ is generated from r_1 and r_2, they (loosely speaking) will not be "completely" independent[3]; the 2 point sampling will ensure that the random variables $s_j, j \in [t]$ are pairwise independent. Before we proceed further, let us prove an important lemma which is the central machinery of 2 point sampling.

Lemma 3.8.1. *Let $p \in \mathbb{N}$ be a prime number and $a, b \in_R \mathbb{F}_p$ be chosen independently and uniformly randomly from $\mathbb{F}_p$ (formally, a and b are random variables distributed uniformly and independently over $\mathbb{F}_p$). Then the random variables in $\{ai + b \pmod p : i \in [t]\}$ for some $t \leqslant p-1$ are uniformly distributed and pairwise independent. That is, for every $i \in [t]$ and $k \in \mathbb{F}_p$, we have the following (uniform distribution).*

$$\Pr[ai + b \equiv k \pmod p] = \frac{1}{p}$$

And, for every $i, j \in [t], i \neq j$, and $k, k' \in \mathbb{F}_p$, we have the following (pairwise independence).

$$\Pr[ai + b \equiv k \pmod p, aj + b \equiv k' \pmod p] = \Pr[ai + b \equiv k \pmod p] \Pr[aj + b \equiv k' \pmod p] = \frac{1}{p^2}$$

Proof. We first prove the first claim as follows.

$$\begin{aligned}
\Pr[ai + b \equiv k \pmod p] &= \sum_{b' \in \mathbb{F}_p} \Pr[ai + b' \equiv k \pmod p | b = b'] \Pr[b = b'] && \text{[Law of total probability]} \\
&= \sum_{b' \in \mathbb{F}_p} \Pr\left[a \equiv \frac{k - b'}{i} \pmod p | b = b'\right] \frac{1}{p} && \text{[b is chosen uniformly randomly]}
\end{aligned}$$

[3] A set of events $\mathcal{A}_i, i \in [\ell]$ are called independent if, for every subset $J \subseteq [\ell]$, we have $\Pr\left[\cap_{i \in [J]} \mathcal{A}_i\right] = \prod_{i \in [J]} \Pr[\mathcal{A}_i]$.

$$= \sum_{b' \in \mathbb{F}_p} \frac{1}{p^2} \qquad [a \text{ is chosen uniformly randomly}]$$
$$= \frac{1}{p} \qquad [|\mathbb{F}_p| = p]$$

We now prove pairwise independence.

$$\begin{aligned}\Pr[ai + b \equiv k \pmod p, aj + b \equiv k' \pmod p] &= \Pr\left[a \equiv \frac{k-k'}{i-j} \pmod p, b \equiv \frac{kj-k'i}{j-i} \pmod p\right] \\ &= \frac{1}{p^2} \\ &= \Pr[ai + b \equiv k \pmod p] \Pr[aj + b \equiv k' \pmod p]\end{aligned}$$

The first equality follows from simply solving two equations, the second equality follows from the fact that a and b are chosen independently uniformly from $\mathbb{F}_p$, and the third equality follows from the first part of the claim. We have assumed p to be a prime number in the statement. Can you see where we use it in the above proof? □

Let us now explain how we generate $s_j, j \in [t]$ from r_1 and r_2. We define $s_j = r_1 j + r_2 \pmod p$ for $j \in [t]$. Our algorithm first samples two random numbers $r_1, r_2 \in \mathbb{F}_p$ where p is a random number such that $p \geqslant 2^d$, computes $\mathcal{A}(x, s_j)$ for every $j \in [t]$, and outputs YES if and only if $\mathcal{A}(x, s_j)$ outputs YES for every $j \in [t]$. Let us now analyze the error probability of our algorithm. The random variables $s_j, j \in [t]$ are pairwise independent due to Lemma 3.8.1. Hence, the random variables $X_j = \mathcal{A}(x, s_j), j \in [t]$ are also pairwise independent where X_j is 1 if $\mathcal{A}(x, s_j)$ outputs NO and 0 otherwise. Let us define another random variable $\mathcal{X} = \sum_{j=1}^{t} X_j$. Since the random variables $X_j = \mathcal{A}(x, s_j), j \in [t]$ are pairwise independent, we have $\text{var}(\mathcal{X}) = \sum_{j=1}^{t} \text{var}(X_j) \leqslant t/4$ where the inequality follows from the fact that X_j is a Bernoulli random variable for $j \in [t]$. To bound the error probability of our algorithm, we observe that our algorithm makes an error for a NO instance x if and only if the random variable $\mathcal{X}$ takes the value 0. By linearity of expectation, we have $\mathbb{E}[\mathcal{X}] \geqslant t/2$ since $\mathbb{E}[X_j] \geqslant 1/2$ for every $j \in [t]$. By Chebyshev inequality, we have the following.

$$\Pr\left[\mathcal{X} = 0\right] \leqslant \Pr\left[|\mathcal{X} - \mathbb{E}[\mathcal{X}]| \geqslant t/2\right] \leqslant \frac{1}{t}$$

Compare the above bound with the $1/4$ bound what we would have got by directly feeding r_1 and r_2 in $\mathcal{A}$. So, it is excellent that we have reduced the error probability from $1/2$ to $1/t$ by sampling only 2 random strings. However, this comes at a cost of running time! This phenomenon is called *time-randomness trade off*.

3.9 Randomized Routing/Rounding: Multi-commodity Flow

Minimizing congestion in a network is a fundamental problem with plenty of applications from managing road traffics to packet routing in computer networks. An abstraction of these applications is as follows: Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a set $\{(s_i, t_i) : i \in [k]\}$ of k source destination pairs, and an integer $\mathcal{C}$, compute if there exist, for every $i \in [k]$, a path p_i from s_i to t_i such that, for every edge $e \in \mathcal{E}$, the number of paths $p_i, i \in [k]$ that uses the edge e is at most $\mathcal{C}$.

The problem above is NP-complete. We will now see an approximation algorithm for the congestion minimization problem. We begin with formulating our problem as an integer linear program (ILP). For that, we view our problem equivalently as the multi-commodity flow problem where our goal is route 1

$$
\begin{array}{|ll|l|}
\hline
 & \min C & \\
\text{subject to} & X_e = \sum_{i=1}^{k} f_{e,i} & \forall e \in \mathcal{E} \\
 & \sum_{(u,v)\in\mathcal{E}} f_{(u,v),i} = \sum_{(v,w)\in\mathcal{E}} f_{(v,w),i} & \forall v \in \mathcal{V}, i \in [k] \\
\text{(ILP)} & \sum_{(s_i,v)\in\mathcal{E}} f_{(u,v),i} = 1 = \sum_{(v,t_i)\in\mathcal{E}} f_{(v,w),i} & \forall i \in [k] \\
 & X_e \leqslant C & \forall e \in \mathcal{E} \\
 & C \geqslant 1 & \\
 & f_{e,i} \in \{0,1\}, \forall e \in \mathcal{E}, i \in [k] & \\
\hline
\end{array}
\qquad
\begin{array}{|ll|l|}
\hline
 & \min C & \\
\text{subject to} & X_e = \sum_{i=1}^{k} f_{e,i} & \forall e \in \mathcal{E} \\
 & \sum_{(u,v)\in\mathcal{E}} f_{(u,v),i} = \sum_{(v,w)\in\mathcal{E}} f_{(v,w),i} & \forall v \in \mathcal{V}, i \in [k] \\
\text{(LP)} & \sum_{(s_i,v)\in\mathcal{E}} f_{(u,v),i} = 1 = \sum_{(v,t_i)\in\mathcal{E}} f_{(v,w),i} & \forall i \in [k] \\
 & X_e \leqslant C & \forall e \in \mathcal{E} \\
 & C \geqslant 1 & \\
 & 0 \leqslant f_{e,i} \leqslant 1, \forall e \in \mathcal{E}, i \in [k] & \\
\hline
\end{array}
$$

Figure 3.1: On the left, we have ILP formulation and on the right side, we have LP relaxation.

unit of flow from s_i to t_i where every edge carries integral unit of flow (can you prove it formally?). We introduce variables $f_{e,i}$ which takes value 1 if the 1 unit of flow from s_i to t_i passes through the edge and 0 otherwise. The variable X_e denotes the congestion of edge $e \in \mathcal{E}$. The ILP formulation of our problem is on the left side of Figure 3.1 and on the right side, we have LP relaxation. Do you see the reason for adding the constraint $C \geqslant 1$? Without this, the integrality gap would be large and we cannot hope to find sub-polynomial approximation ratio.

Now the idea is that, for every $i \in [k]$, we will pick a path p_i from s_i to t_i randomly in such a way that, for every edge $e \in \mathcal{E}$, the edge belongs to the path p_i with probability $f_{e,i}$. How to pick such a path? greedily! Concretely, for $i \in [k]$, we pick the path p_i one hop at a time as follows. The first vertex of p_i is (obviously) s_i. We pick the second vertex of p_i as v with probability $\frac{f_{(s_i,v),i}}{\sum_{w\in\mathcal{V}} f_{(s_i,v),i}} = f_{(s_i,v),i}$. Inductively, suppose we have picked the first ℓ vertices of p_i and yet not reached t_i. We now prove that the probability that any edge $e = (u,v) \in \mathcal{E}$ belongs to the path p_i is $f_{e,i}$ by induction of the distance of u from s_i. Let the (random) path p_i be $(s_i =)u_1, u_2, \ldots, u_r(= t_i)$. We may assume without loss of generality that there is no repetition of vertices in p_i by working with an extreme point solution (not any optimal solution)[4]. It follows immediately from the description of our algorithm that $\Pr[(s_i, v) \text{ belongs to } p_i] = f_{(s_i,v),i}$ for every $(s_i, v) \in \mathcal{E}$. Hence the claim holds for every vertices which are one hop away from s_i and thus the induction starts. Suppose the claim is true for every vertices which are ℓ hops away from s_i. Let $(u,v) \in \mathcal{E}$ be an edge such that u is ℓ hops away from s_i (and thus v is $\ell+1$ hops away from s_i). By induction hypothesis, we have $\Pr[u \in p_i] = \sum_{(w,u)\in\mathcal{E}} f_{(w,u),i}$ (the probability that we pick u by picking any of its incoming edges). Hence, we have $\Pr[(u,v) \in p_i] = \frac{f_{(u,v),i}}{\sum_{(u,x)\in\mathcal{E}} f_{(u,x),i}} \sum_{(w,u)\in\mathcal{E}} f_{(w,u),i} = f_{(u,v),i}$ due to conservation of flow property. Hence, by induction principal, we have proved the claim for every edge reachable from s_i. If some edge e is not reachable from s_i, then we have $f_{e,i} = 0$ and our algorithm can never pick it. So the claim holds for every edge in the graph.

The above claim can also be proved from the flow to path decomposition theorem which states that any $s-t$ flows can be decomposed into at most m many flow paths p_j carrying a flow f_j. Moreover such a decomposition can be computed in polynomial time. Assuming the above result, let us see another algorithm to pick a path p_i from s_i to t_i randomly in such a way that, for every edge $e \in \mathcal{E}$, the edge belongs to the path p_i with probability $f_{e,i}$. Let $\mathcal{P}_i$ be the set of flow paths corresponding to the $s_i - t_t$ flow. Since, the flow

[4]Refer any standard book on Linear Programming for extreme point solution. Informally speaking, an extreme point solution is an optimal solution with maximum number of variables taking value 0.

value from s_i to t_i is 1, we can view the flows along the paths in $\mathcal{P}_i$ as probabilities. We pick a path p_i from the set $\mathcal{P}_i$ with probability being equal to the $s_i - t_i$ flow value in the path p_i. Then, it immediately follows that, for every edge e, the probability that the edge belongs to p_i is the total amount of $s_i - t_i$ flow that the edge e is carrying.

We now prove the approximation guarantee of our algorithm. For an edge $e \in \mathcal{E}$ and $i \in [k]$, we define an indicator random variable $X_{e,i}$ for the event that the edge e belongs to the path p_i. We first observe that, for every edge $e \in \mathcal{E}$, the random variables $X_{e,i}, i \in [k]$ are pairwise independent (can you prove it formally?). Obviously, for any $i \in [k]$, the random variables $X_{e,i}, e \in \mathcal{E}$ are not independent but we do not need that. For $e \in \mathcal{E}$, let $X_e = \sum_{i=1}^{k} X_{e,i}$ denotes the congestion of the edge e. Let C^* be the optimal value of ILP. Then, $\mathbb{E}[X_e] \leqslant C^*$ since LP is a relaxation of the ILP. Since $X_{e,i}, i \in [k]$ are Bernoulli random variables, by Chernoff bound, we have the following.

$$\begin{aligned}
\Pr[X_e \geqslant (1+\delta)C^*] &= \Pr\left[X_e \geqslant (1+\delta)\frac{C^*}{\mathbb{E}[X_e]}\mathbb{E}[X_e]\right] \\
&\leqslant \left(\frac{e^{(1+\delta)\frac{C^*}{\mathbb{E}[X_e]}-1}}{\left((1+\delta)\frac{C^*}{\mathbb{E}[X_e]}\right)^{(1+\delta)\frac{C^*}{\mathbb{E}[X_e]}}}\right)^{\mathbb{E}[X_e]} \\
&\leqslant \left(\frac{e^{(1+\delta)\frac{C^*}{\mathbb{E}[X_e]}-1}}{(1+\delta)^{(1+\delta)\frac{C^*}{\mathbb{E}[X_e]}}}\right)^{\mathbb{E}[X_e]} && [\text{Since } \mathbb{E}[X_e] \leqslant C^*] \\
&= \left(\frac{e^{1+\delta}}{(1+\delta)^{1+\delta}}\right)^{C^*} e^{-\mathbb{E}[X_e]} \\
&\leqslant \left(\frac{e^{1+\delta}}{(1+\delta)^{1+\delta}}\right)^{C^*}
\end{aligned}$$

Now using union bound, we have the following if we have $m(\leqslant n^2)$ edges in the graph.

$$\Pr[\exists e \in \mathcal{E}, X_e \geqslant (1+\delta)C^*] \leqslant m\left(\frac{e^{1+\delta}}{(1+\delta)^{(1+\delta)}}\right)^{C^*}$$

For $\delta = 1 + \frac{2\ln m}{\ln\ln m}$ we have $\Pr[\exists e \in \mathcal{E}, X_e \geqslant (1+\delta)C^*] \leqslant \frac{1}{m^{C^*}}\frac{1}{m}$ since $C^* \geqslant 1$. Hence we have a Monte Carlo randomized algorithm with approximation ratio $\mathcal{O}(\frac{\ln m}{\ln\ln m})$.

Chapter 4

Markov Chain

A *stochastic process* is a set $\mathcal{X} = \{X(t) : t \in \mathcal{T}\}$ of random variables where $\mathcal{T}$ is any index set. The index set $\mathcal{T}$ often denote time and, intuitively speaking, the random variable $X(t)$ denote the *state* of the process at time t. If $\mathcal{T}$ is a countable set, then $\mathcal{X}$ is called a *discrete time* stochastic process. If $X(t)$ is a discrete random variable for every $t \in \mathcal{T}$, then $\mathcal{X}$ is called a *discrete space* stochastic process. Here we focus on only discrete time and discrete space stochastic process. Whenever we do not mention explicitly, we assume that the Markov chain under consideration is discrete time and discrete space. So, we can assume without loss of generality that we have $\mathcal{T} = \mathbb{N}$.

A discrete time stochastic process is called a Markov chain if the random variable X_t depends only on X_{t-1} for every $t \in \mathcal{T}$. This property is popularly known as *memoryless property or Markov property*.

Definition 4.0.1. *A sequence of random variables $X_0, X_1, X_2, \ldots$ is called a Markov chain if the following holds for every $x_i \in \mathbb{R}, i \geqslant 1$.*

$$\Pr[X_i = x_i | X_{i-1} = x_{i-1}, X_{i-2} = x_{i-2}, \ldots, X_1 = x_1] = \Pr[X_i = x_i | X_{i-1} = x_{i-1}]$$

Without loss of generality, let us assume that the state space of a Markov chain be $\{0, 1, \ldots, n\}$ (or $\mathbb{N}$). The Markov property implies that any Markov chain can be specified uniquely by the one-step transition matrix $\mathcal{P} = (P_{i,j})_{i,j \in \{0,1,\ldots,n\}} \in [0,1]^{(n+1)\times(n+1)}$ where $P_{i,j}$ denote the probability of moving to state j from state i. Hence, we have $\sum_{j=0}^{n} P_{i,j} = 1$ for every $i \in \{0, 1, \ldots, n\}$. One of the most popular use of Markov chains is to model random walks on graphs. Once we have a Markov chain, we are often interested to answer following questions:

1. Given a start state (which may be a random state too), what is the expected number of steps the Markov chain takes to reach some state in the Markov chain? This is called *hitting time*.

2. Given a start state (which may be a random state too), does there exist any limiting distribution (called stationary distribution) of the Markov chain? Is it unique? If yes, then how many steps the Markov chain takes to reach this unique stationary distribution. This is called *mixing time*.

Let us now see a randomized algorithm for the 2SAT problem where we are interested to know the answer of the first question (of course for a specific Markov chain).

4.1 Randomized Algorithm for 2SAT

In the 2SAT problem, we are given a set $\{C_j : j \in [m]\}$ of m clauses each of which is an OR of two literals over n Boolean variables $\{x_i : i \in [n]\}$ and we need to compute if there exists a Boolean assignment to these n variables which satisfies all the clauses.

Here is a simple randomized algorithm for 2SAT. Start with an arbitrary assignment for the variables. If this assignment satisfies all the clauses, then we have found an assignment which satisfies all the clauses. Otherwise there exists a clause C_j for some $j \in [m]$ which the assignment does not satisfy. We randomly pick a variable which appears in C_j and complement it – this would ensure that the new assignment satisfies C_j (of course it may fail to satisfy some other clause which it was satisfying before). If the new assignment satisfies all the clauses, then we are done. Otherwise we repeat the above step. If we do not find any assignment which satisfies all the clauses after repeating $\ell = \frac{n^2}{\varepsilon}$ times, then we output that there does not exist any assignment to the variables which satisfies all the clauses. Of course if we ever find a satisfying assignment, we immediately output it and the algorithm terminates there itself. We now show that the probability of error of the algorithm above is at most ε.

Our randomized algorithm for 2SAT is a Monte Carlo randomized algorithm — if a 2SAT instance is not satisfiable, then the algorithm always outputs NO (never makes error): no matter what it does, it can never find any satisfying assingment. So, let us focus on a YES instance $\mathcal{I}$ — the set of clauses be $\{C_j : j \in [m]\}$ and the set of variables $\{x_i : i \in [n]\}$. Suppose $f : \{x_i : i \in [n]\} \longrightarrow \{\text{TRUE}, \text{FALSE}\}$ be a satisfying assignment for $\mathcal{I}$. Let us consider the stochastic process $X_0, X_1, X_2, \ldots$ where X_i denotes the number of variables in $\{x_i : i \in [n]\}$ where the assignment after the i-th iteration of the algorithm agree with f. Observe that each X_i is a discrete random variable taking values in $[0, n]$. If any random variable X_ℓ takes the value n for any $\ell \leqslant 2n^2$, then the algorithm has successfully discovered a satisfying assignment namely f. Hence, to bound the error probability, we need to bound the probability that no random variable X_ℓ takes the value n for every $\ell \leqslant 2n^2$ which we do now.

Let us see the distribution of $X_0, X_1, \ldots$ If $X_i = 0$, then X_{i+1} takes value 1 with probability 1 – if the current assignment does not agree with f on any variable, then the assignment in the next step will agree with f on exactly one variable. That is, we have the following.

$$\Pr[X_{i+1} = 1 | X_i = 0] = 1$$

If X_i takes the value n, then the algorithm stops (and outputs YES). For other $1 \leqslant j \leqslant n-1$, we have the following.

$$\Pr\left[X_i = j + 1 | X_{i-1} = j\right] \geqslant \frac{1}{2}$$

$$\Pr\left[X_i = j - 1 | X_{i-1} = j\right] \leqslant \frac{1}{2}$$

Suppose $X_{i-1} = j$ for any $1 \leqslant j \leqslant n-1$ (the assignment in the $(i-1)$-th iteration agrees with f on j variables), and the algorithm picks a clause C_t which the assignment in the $(i-1)$-th iteration does not satisfy. Then we observe that at least one literal between the two literals in C_t is set TRUE by f (since f satisfies it) and our algorithm picks this variable with probability $\frac{1}{2}$.

Is the stochastic process $X_0, X_1, X_2, \ldots$ a Markov chain? No! To see this, it is possible that the algorithm picks a clause C_t in the $(i-1)$-th iteration and f sets both the literals to TRUE. In this case, X_i will be $X_{i-1}+1$. So the distribution of the random variable X_i depends on the fraction of clauses of the unsatisfied clauses

(by the assignment in the $(i-1)$-th iteration) for which f satisfies both its literals. So X_i not only depends only on X_{i-1} but also on $X_0, X_1, \ldots, X_{i-2}$. However, there is an easy way to "fix" it.

Consider another stochastic process $Y_0, Y_1, \ldots$ defined as follows.

$$\Pr[Y_{i+1} = 1 | Y_i = 0] = 1$$

$$\Pr[Y_i = j+1 | Y_{i-1} = j] = \frac{1}{2}$$

$$\Pr[Y_i = j-1 | Y_{i-1} = j] = \frac{1}{2}$$

The stochastic process $Y_0, Y_1, \ldots$ is clearly a Markov chain (follows directly from the definition of distribution). Intuitively speaking, $Y_0, Y_1, \ldots$ is a pessimistic version of $X_0, X_1, X_2, \ldots$ – $Y_0, Y_1, \ldots$ moves slower to n than $X_0, X_1, X_2, \ldots$. Hence intuitively, if $\mathcal{T}_X$ and $\mathcal{T}_Y$ are the random variables denoting the number of steps that the stochastic process $X_0, X_1, \ldots$ and the stochastic process $Y_0, Y_1, \ldots$ respectively take to reach n (the first time some variable takes value n), then we have $\mathbb{E}[\mathcal{T}_X] \leqslant \mathbb{E}[\mathcal{T}_Y]$. One can make the above argument formal by using "coupling technique" (will see soon). Now we are done if we show that $\mathbb{E}[\mathcal{T}_y] \leqslant n^2$ (Markov inequality will give the claimed probability guarantee).

To compute $\mathbb{E}[\mathcal{T}_y]$, let us define a random variable Z_i for $0 \leqslant i \leqslant n$ which denotes the number of steps the Markov chain $Y_0, Y_1, \ldots$ takes to reach n. Then, we have the following.

$$\mathbb{E}[Z_0] = 1 + \mathbb{E}[Z_1]$$
$$\mathbb{E}[Z_n] = 0$$

For any $i \in [n-1]$, we have the following.

$$\begin{aligned}
\mathbb{E}[Z_i] &= \frac{1}{2}\left(1 + \mathbb{E}[Z_{i-1}]\right) + \frac{1}{2}\left(1 + \mathbb{E}[Z_{i+1}]\right) \\
&= 1 + \frac{\mathbb{E}[Z_{i-1}] + \mathbb{E}[Z_{i+1}]}{2} \\
\Rightarrow 2\mathbb{E}[Z_i] &= 2 + \mathbb{E}[Z_{i-1}] + \mathbb{E}[Z_{i+1}] \\
\Rightarrow 2\sum_{i=1}^{n-1} \mathbb{E}[Z_i] &= 2(n-1) + \sum_{i=1}^{n-1} \mathbb{E}[Z_{i-1}] + \sum_{i=1}^{n-1} \mathbb{E}[Z_{i+1}] \\
\Rightarrow \mathbb{E}[Z_{n-1}] &= \mathbb{E}[Z_n] - \mathbb{E}[Z_1] + \mathbb{E}[Z_0] + 2(n-1) \\
&= 2n - 1
\end{aligned}$$

The last equality follows from the fact that $\mathbb{E}[Z_0] = 1 + \mathbb{E}[Z_1]$. Using $\mathbb{E}[Z_{n-1}]$ to compute $\mathbb{E}[Z_{n-2}]$ and so on, we get the following.

$$\mathbb{E}[Z_{n-2}] = (2n-1) + (2n-3), \ldots, \mathbb{E}[Z_0] = (2n-1) + (2n-3) + \cdots + 1 = n^2$$

Since $\mathbb{E}[\mathcal{T}_y]$ is a weighted average of $\mathbb{E}[Z_{n-1}], \mathbb{E}[Z_{n-2}], \ldots, \mathbb{E}[Z_0]$, we have $\mathbb{E}[\mathcal{T}_y] \leqslant n^2$.

We next study the long term behavior of Markov chain – letting Markov chain to run forever, does the distribution of n-th state converge as n tends to ∞?

4.2 Stationary Distribution

Given a finite Markov chain with transition matrix $\mathcal{P}$, a distribution π on the set of states is called a *stationary distribution* if $\pi\mathcal{P} = \pi$. Observe that if the distribution of the current state is π, then the distribution of next state is $\pi\mathcal{P}$. Hence, if somehow the distribution of the current state reaches π, it remains there forever. To find a stationary distribution (or to check whether any exists), one can try to solve the system of linear equations $\pi\mathcal{P} = \pi$ with $\sum_{i=1}^{n} \pi_i = 1$ and $\pi_i \geqslant 0$ for every $i \in [n]$ where n is the number of states in Markov chain. It can be proved using elementary linear algebra that, for every finite Markov chain, there indeed exists at least one stationary distribution. If there exists an unique stationary distribution π for a Markov chain, then the probability π_i is often called the stationary probability of state i.

Obviously, if any Markov chain starts at a stationary distribution, it stays there forever. However, we would often like our Markov chain to converge to some stationary distribution even if it does not start from there. For certain Markov chain, this convergence is obviously not guaranteed from any start state. This is the case for example for the Markov chain in Figure 4.1. Its stationary distribution is $(1/2, 1/2)$. However, it can be proved easily that if the Markov chain starts at any other distribution, it never converges. A closer inspection at Figure 4.1 reveals the intuitive reason – the Markov chain is periodic.

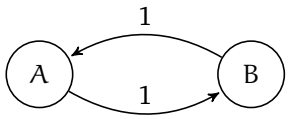

Figure 4.1: Markov chain which does not converge if the starting distribution is anything other than the stationary distribution $(1/2, 1/2)$.

Definition 4.2.1 (Periodicity of a Markov Chain)**.** *A state a in a discrete Markov chain is called periodic if there exists a positive integer Δ such that $\Pr[X_{i+s} = a | X_i = a] = 0$ unless s is division by Δ. Equivalently, for any state a, if $S(a) = \{s \in \mathbb{N} : \Pr[X_s = a | X_0 = a] \neq 0\}$, then the state a is called aperiodic if and only if we have $gcd(S(a)) = 1$. A Markov chain is called periodic if any of its states is periodic.*

A standard technique to convert a periodic Markov chain to an aperiodicity one is to make it *lazy*. Formally, suppose we are given a Markov chain with state transition matrix $\mathcal{P} = (p_{i,j})_{i,j\in[n]}$. We construct another Markov chain $\mathcal{Q}$ whose state transition matrix $\mathcal{Q} = (q_{i,j})_{i,j\in[n]}$ is defined as $q_{i,i} = \frac{1+p_{i,i}}{2}$ and $q_{i,j} = p_{i,j}/2$ for every $i \neq j$ – that is, $\mathcal{Q} = \frac{\mathcal{P}+\mathcal{I}}{2}$ where $\mathcal{I}$ is the identity matrix. If π is a stationary distribution of $\mathcal{P}$, then we have the following which shows that π is a stationary distribution for $\mathcal{Q}$ too.

$$\pi\mathcal{Q} = \pi\frac{\mathcal{P}+\mathcal{I}}{2} = \pi$$

Now, it actually can be proved that the new Markov chain always converges to the stationary distribution $(1/2, 1/2)$ irrespective of its starting distribution (we will prove a more general result soon). Does aperiodicity guarantee existence of unique stationary distribution and convergence? Not yet! On a high level, aperiodicity guarantees convergence. However, an aperiodic Markov chain can have multiple stationary distributions and the stationary distribution the Markov chain converges may depend on its starting distribution. For example, we can simply take "union" of two copies of the modified Markov chain (with added self loop) and we will have infinitely many stationary distributions! To guarantee uniqueness of stationary distribution, we need another property which is called *irreducibility*. A finite Markov chain is called *irreducible* is for any two states

a and b, the probability of reaching b from a is non-zero. In graph theoretic terminology, a Markov chain is irreducible if the corresponding state transition graph is strongly connected. It turns out that irreducibility guarantees uniqueness of stationary distribution. However irreducibility does not guarantee convergence to stationary distribution. For guaranteeing convergence, we need our Markov chain to be aperiodic. The following theorem is one of the fundamental results for Markov chains.

Theorem 4.2.1. *Any finite, irreducibile, and aperiodic Markov chain has a unique stationary distribution. Moreover, the Markov chain converges to this unique stationary distribution irrespective of the start state.*

We will prove Theorem 4.2.1 using coupling techniques which we present next. When we say convergence, there must be some notion of distance for convergence to make sense. In this context, the distance of choice is the total variation distance which is defined as follows (part of the reason for choosing this distance measure is if one proves some result on convergence or rate of convergence under total variation distance, then the result immediately holds for other popular distance measures as well).

Definition 4.2.2 (Total variation distance)**.** *Given two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ over some σ-algebra $(\Omega, \mathcal{F})$, the total variation distance of $\mathcal{P}$ and $\mathcal{Q}$ is defined as $d_{TV}(\mathcal{P}, \mathcal{Q}) = \sup_{A \in \mathcal{F}} |\mathcal{P}(A) - \mathcal{Q}(A)|$. For any two random variables $\mathcal{X}$ and $\mathcal{Y}$, we define the total variation distance between them is defined as $d_{TV}(\mathcal{X}, \mathcal{Y}) = \sup_{A \in \mathcal{B}} |\Pr[X \in A] - \Pr[Y \in A]|$ where $\mathcal{B}$ is the Borel σ-algebra.*

It can be proved using elementary combinatorics that, for discrete probability distributions $\mathcal{P}$ and $\mathcal{Q}$, $2d_{TV}(\mathcal{P}, \mathcal{Q}) = \|\mathcal{P} - \mathcal{Q}\|_1$ where $\|\mathcal{P} - \mathcal{Q}\|_1$ denotes ℓ_1 norm of $\mathcal{P} - \mathcal{Q}$.

4.2.1 Mixing Time and Coupling

We will show that any finite, irreducible, and aperiodic Markov chain converges to unique stationary distribution in total variation distance. That is, for any $\varepsilon > 0$, there exists a *mixing time* $t_{mix}(\varepsilon) \in \mathbb{N}$ such that, for any initial distribution x over the states of the Markov chain, $d_{TV}(x\mathcal{P}^t, \pi) \leqslant \varepsilon$ for every $t \geqslant t_{mix}(\varepsilon)$. To prove convergence and mixing time of Markov chains, one of the most useful tools is coupling which is defined as below.

Definition 4.2.3 (Coupling)**.** *A coupling of two random variables $\mathcal{X}$ and $\mathcal{Y}$ with distributions $\mu_{\mathcal{X}}$ and $\mu_{\mathcal{Y}}$ is a joint distribution $\mu_{(\mathcal{X},\mathcal{Y})}$ on $(\mathcal{X}, \mathcal{Y})$ such that the marginal distribution of $\mu_{(\mathcal{X},\mathcal{Y})}$ on $\mathcal{X}$ and $\mathcal{Y}$ are respectively $\mu_{\mathcal{X}}$ and $\mu_{\mathcal{Y}}$ (that is the joint distribution ensures correct marginals).*

The following result, known as *coupling lemma*, is the central tool of many coupling applications.

Lemma 4.2.1 (Coupling Lemma)**.** *For any two discrete random variables $\mathcal{X}$ and $\mathcal{Y}$, we have the following.*

$$d_{TV}(\mathcal{X}, \mathcal{Y}) \leqslant \Pr[\mathcal{X} \neq \mathcal{Y}]$$

Proof. For any $A \in \mathcal{F}$, we have the following.

$$\Pr[\mathcal{X} \in A] = \Pr[\mathcal{X} \in A \wedge \mathcal{Y} \in A] + \Pr[\mathcal{X} \in A \wedge \mathcal{Y} \notin A]$$

$$\Pr[\mathcal{Y} \in A] = \Pr[\mathcal{X} \in A \wedge \mathcal{Y} \in A] + \Pr[\mathcal{X} \notin A \wedge \mathcal{Y} \in A]$$

Thus we have the following.

$$|\Pr[\mathcal{X} \in A] - \Pr[\mathcal{Y} \in A]| = |\Pr[\mathcal{X} \in A \wedge \mathcal{Y} \notin A] - \Pr[\mathcal{X} \notin A \wedge \mathcal{Y} \in A]|$$

$$\leqslant \Pr[\mathcal{X} \neq \mathcal{Y}]$$
$$\Rightarrow \sup_{A \in \mathcal{F}} |\Pr[\mathcal{X} \in A] - \Pr[\mathcal{Y} \in A]| \leqslant \Pr[\mathcal{X} \neq \mathcal{Y}]$$
$$\Rightarrow d_{TV}(\mathcal{X}, \mathcal{Y}) \leqslant \Pr[\mathcal{X} \neq \mathcal{Y}]$$

□

Hence, for two random variables $\mathcal{X}$ and $\mathcal{Y}$, if we can show that $\Pr[\mathcal{X} \neq \mathcal{Y}]$ is small, then by coupling lemma, it follows that $d_{TV}(\mathcal{X}, \mathcal{Y})$ is also small.

We need the following result for aperiodic Markov chain to prove Theorem 4.2.1.

Lemma 4.2.2. *For every state a of a finite, irreducible, and aperiodic Markov chain $\mathcal{X}(= X_0, X_1, \ldots)$, there exists a positive integer i_a such that we have $\Pr[X_s = a | X_0 = a] \neq 0$ for every $s > i_a$.*

Proof. Let $S(a) = \{s \in \mathbb{N} : \Pr[X_s = a | X_0 = a] \neq 0\}$. Since the Markov chain is finite and irreducible the set $S(a)$ is an infinite set. Since the Markov chain $\mathcal{X}$ is aperiodic, we have $\gcd(S(a)) = 1$. This implies that there exists a finite subset $S' \subset S(a)$ such that $\gcd(S') = 1$. Let $S' = \{m_1, \ldots, m_k\}$ and $M = \prod_{i=1}^{k} m_i$. From extended Euclidean algorithm, there exists integers $a_1, \ldots, a_k$ with $|a_i| \leqslant M/m_i$ for every $i \in [k]$ such that $\sum_{i=1}^{k} a_i m_i = 1$. To prove the result, it is enough to show that there exists an integer i_a such that, for every integer $s > i_a$, there exists a path from state a to a in the corresponding state transition diagram.

Ideally, had all $a_1, \ldots, a_k$ were positive integers (which is certainly not the case) we could use these numbers to loop around the state a to get desired path length. To get around the possible negativity problem with $a_1, \ldots, a_k$, we define $b_i = a_i + M/m_i$ for $i \in [k]$ which gives $\sum_{i=1}^{k} b_i m_i = kM + 1$. We observe that, for every $\ell \in \{0, 1, \ldots, M-1\}$, there exists a loop from state a to itself of length $\ell(kM+1)$ (traverse the loop of length m_i ℓb_i times). This implies that for every $\ell \in \{0, 1, \ldots, M-1\}$, there is a loop from a to itself of length $(kM)^2 + \ell$ (traverse the loop of length $\ell(kM+1)$ and traverse the loop of length M $kM - \ell$ times). For any $\ell \geqslant M$, we can simply traverse the loop of length M more times. Hence with $t_a = (kM)^2$, we have proved the result. □

We now prove Theorem 4.2.1. For easy reference, we write Theorem 4.2.1 once again.

Theorem 4.2.2. *Any finite, irreducibile, and aperiodic Markov chain has a unique stationary distribution. Moreover, the Markov chain converges to this unique stationary distribution irrespective of the start state.*

Proof. Let $\mathcal{P} = (p_{i,j})_{i,j \in [n]}$ be the transition matrix of any finite, irreducible, and aperiodic Markov chain. We consider two copies $\{X_i\}_{i \in \mathbb{N}}$ and $\{Y_i\}_{i \in \mathbb{N}}$ of this Markov chain where the first copy $\{X_i\}_{i \in \mathbb{N}}$ starts at any arbitrary state x and the second copy $\{Y_i\}_{i \in \mathbb{N}}$ starts at some stationary distribution π. We now define a coupling between $\{X_i\}_{i \in \mathbb{N}}$ and $\{Y_i\}_{i \in \mathbb{N}}$ (which is noting but a joint distribution on $\{X_i\}_{i \in \mathbb{N}}$ and $\{Y_i\}_{i \in \mathbb{N}}$ which respect individual marginals) as follows: for any $t \in \mathbb{N}$, if $X_t \neq Y_t$, then we define $\Pr[X_{t+1} = j \wedge Y_{t+1} = j' | X_t = i, Y_t = i'] = p_{i,j} p_{i',j'}$; if $X_t = Y_t$, then we define $\Pr[X_{t+1} = Y_{t+1} = j | X_t = Y_t = i] = p_{i,j}$ (verify that the joint distribution indeed defines a coupling). The proof idea on an intuitive level is as follows. Since the Markov chain $\{Y_i\}_{i \in \mathbb{N}}$ starts at a stationary distribution, it remains there forever. On the other hand, the Markov chain $\{X_i\}_{i \in \mathbb{N}}$ runs independently until it collides with $\{Y_i\}_{i \in \mathbb{N}}$ and from there onward it simply follows $\{Y_i\}_{i \in \mathbb{N}}$. What we will show below is that the probability of that $\{X_i\}_{i \in \mathbb{N}}$ collides with $\{Y_i\}_{i \in \mathbb{N}}$ within the first k steps goes to 1 as k goes to ∞. Then the result will from using Lemma 4.2.2. We now formally prove it.

Let us fix any state $i \in [n]$ and r be the maximum over all other states $j \in [n] \setminus \{i\}$ the *first passage time* $f_{j,i}$: $f_{j,i} = \min\{t \in \mathbb{N} : \Pr[X_t = i | X_0 = j] \neq 0\}$. That is, the first passage time is the first time when the probability of reaching i from j is non-zero. Let $s \in \mathbb{N}$ be an integer such that $\Pr[X_t = i_0 | X_0 = i_0] \neq 0$ for every state i_0 and every time $t \geqslant s$ (such an s exists due to Lemma 4.2.2).

For $\ell \in \mathbb{N} \setminus \{0\}$, suppose $\{X_i\}_{i \in \mathbb{N}}$ and $\{Y_i\}_{i \in \mathbb{N}}$ do not collide till time $\ell(r+s)$; let $X_{\ell(r+s)} = j, Y_{\ell(r+s)} = j', j \neq j'$. We now bound the probability that $X_{(\ell+1)(r+s)} \neq Y_{(\ell+1)(r+s)}$. By the definition of r, there exists non-negative integers $u, u' \leqslant r$ such that there is non-zero probability that that $X_{\ell(r+s)+u} = i$ and $Y_{\ell(r+s)+u'} = i$. Now by the definition of s, since $r + s - u \geqslant s$ and $r + s - u' \geqslant s$, there is non-zero probability that $X_{(\ell+1)(r+s)} = i = Y_{(\ell+1)(r+s)}$; let this probability be ε. That is, we have $\Pr[X_{(\ell+1)(r+s)} \neq Y_{(\ell+1)(r+s)}] = (1 - \varepsilon) \Pr[X_{\ell(r+s)} \neq Y_{\ell(r+s)}] \leqslant (1 - \varepsilon)$. Applying this argument inductively, we obtain $\Pr[X_t \neq Y_t] \leqslant (1 - \varepsilon)^{\lfloor \frac{t}{r+s} \rfloor}$. Hence $\Pr[X_t \neq Y_t]$ goes to 0 as t goes to ∞. The distribution of X_t is $x\mathcal{P}^t$. Hence, by Lemma 4.2.2 we get $d_{TV}(c\mathcal{P}^t, \pi)$ goes to 0 as t goes to ∞.

Since $\{X_i\}_{i \in \mathbb{N}}$ converges to π, the stationary distribution π is unique. □

Note: In the proof of Theorem 4.2.1, we show that, for any stationary distribution π of a finite, irreducibility, and aperiodic Markov chain, the chain converges to π from any starting distribution thereby proving uniqueness of π as a by product. However, one can first prove using linear algebra that any finite and irreducible (we do not need aperiodicity for this part) has unique stationary distribution (which is equivalent to showing that the corresponding system of linear equations has a unique solution) and then follow the same approach in the proof of Theorem 4.2.1 to prove convergence.

4.3 Reversible Markov Chain

Suppose we have a finite, irreducible, and aperiodicity Markov chain whose stationary distribution we wish to compute. By Theorem 4.2.1, we know that there exists a unique stationary distribution of the Markov chain. To compute the stationary distribution, we need to solve a system of linear equations. However, in many applications, it may not be an easy task. However, many important Markov chains satisfy the following properties which are called *detailed balanced equations*.

$$\pi_i p_{i,j} = \pi_j p_{j,i}, \text{ for any two states } i, j$$

A Markov chain that satisfy detailed balanced equation for any distribution π over its states is called a *reversible* Markov chain. For a reversible Markov chain, we have $\sum_{i=1}^{n} \pi_i p_{i,j} = \sum_{i=1}^{n} \pi_j p_{j,i} = \pi_j$. Hence π is the stationary distribution. Detailed balanced equations often provides us a quick way to compute the stationary distribution of a Markov chain. Let's see an example.

4.3.1 Random Walk on Undirected Graph

A model example of a reversible Markov chain is a random walk on any (finite) undirected graph $\mathcal{G} = (\mathcal{V} = \{1, 2, \ldots, n\}, \mathcal{E})$. The state space of the Markov chain is $\mathcal{V}$ and the transition probabilities are defined as follows.

$$p_{i,j} = \begin{cases} \frac{1}{d_i} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

Let us check that the distribution $\pi_i = \frac{d_i}{2|\mathcal{E}|}$ satisfies the detailed balanced equation.

$$\pi_i p_{i,j} = \begin{cases} \frac{d_i}{2|\mathcal{E}|}\frac{1}{d_i} = \frac{1}{2|\mathcal{E}|} = \frac{d_j}{2|\mathcal{E}|}\frac{1}{d_j} = \pi_j p_{j,i} & \text{if } (i,j) \in \mathcal{E} \\ 0 = \pi_i p_{j,i} & \text{otherwise} \end{cases}$$

Since the graph is connected, the corresponding Markov chain is irreducible and thus the stationary distribution of the Markov chain is unique. Thus the distribution π is the unique stationary distribution of the Markov chain.

Given a connected undirected graph $\mathcal{G}$, how to construct a corresponding Markov chain where uniform distribution is the stationary distribution? The above result indicates the following idea — add enough number of self loops so that the "degree" of every vertex becomes same. Formally, from a graph $\mathcal{G} = (\mathcal{V} = \{1, 2, \ldots, n\}, \mathcal{E})$, we define a Markov chain on the state space $[n]$ with following transition probabilities. Let M be any at least as large as the maximum degree of $\mathcal{G}$.

$$p_{i,j} = \begin{cases} \frac{1}{M} & \text{if } (i,j) \in \mathcal{E} \\ 1 - \frac{\deg(x)}{M} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

The uniform distribution is the stationary distribution as the following calculation proves.

$$\frac{1}{n} p_{i,j} = \begin{cases} \frac{1}{n}\frac{1}{M} = \frac{1}{n} p_{j,i} & \text{if } (i,j) \in \mathcal{E} \\ \frac{1}{n}\left(1 - \frac{\deg(x)}{M}\right) = \frac{1}{n} p_{j,i} & \text{if } i = j \\ 0 = \frac{1}{n} p_{j,i} & \text{otherwise} \end{cases}$$

Actually the above idea of enforcing uniform distribution as the stationary distribution can be generalized to enforce any distribution of our choice which is popularly known as the Metropolis algorithm.

4.3.2 The Metropolis Algorithm

Lemma 4.3.1. *Let $\mathcal{G} = (\mathcal{V} = \{1, 2, \ldots, n\}, \mathcal{E})$ be a connected undirected graph and $\pi \in \Delta_n$ be any probability distribution over $\mathcal{V}$ with $\pi_i > 0$ for every $i \in \mathcal{V}$. Consider the Markov chain $\mathcal{P}$ on $\mathcal{V}$ defined as follows. Let M be any integer at least the maximum degree of $\mathcal{G}$.*

$$p_{i,j} = \begin{cases} \frac{1}{M}\min\{1, \frac{\pi_j}{\pi_i}\} & \textit{if } (i,j) \in \mathcal{E} \\ 1 - \sum_{j \neq i} p_{i,j} & \textit{if } i = j \\ 0 & \textit{otherwise} \end{cases}$$

Proof. We will show that π satisfies detailed balanced equations. Let $i, j \in \mathcal{V}$ be any two states. If $i = j$ or $\{i,j\} \notin \mathcal{E}$, then obviously we have $\pi_i p_{i,j} = \pi_j p_{j,i}$. Let us assume $i \neq j$ and $\{i,j\} \in \mathcal{E}$. Without loss of generality, let us assume that $\pi_i \geqslant \pi_j$. Then we have the following.

$$\pi_i p_{i,j} = \pi_i 1/M = \pi_j \frac{1}{M}\frac{\pi_i}{\pi_j} = \pi_j p_{j,i}$$

□

4.4 Examples

Let us now see few examples of Markov chain.

4.4.1 Markov Chain with Independent Sets as State Space

Let be $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ any graph with at least one edge. Let us define the following Markov chain with the set of all independent sets of $\mathcal{G}$ being its state space. Let X_t be the current state of the Markov chain. The next state X_{t+1} is defined as follows. We first pick a vertex $v \in \mathcal{V}$ uniformly at random. If

$$X_{t+1} = \begin{cases} X_t \setminus \{v\} & \text{if } v \in X_t \\ X_t \cup \{v\} & \text{if } v \notin X_t \text{ and } X_t \cup \{v\} \text{ is an independent set of } \mathcal{G} \\ X_t & \text{otherwise} \end{cases}$$

Since every state is reachable from the empty set (which is an independent set) and the empty set is reachable from every state, the Markov chain is irreducibile. Since $\mathcal{G}$ has at least one edge say (u, v), there exists at least one state in the Markov chain which has a self-loop and since the Markov chain is irreducibile, it is aperiodic. For any two states i, j with $i \neq j$, if there is a transition between i and j, then we have $p_{i,j} = p_{j,i} = 1/n$ where $n = |\mathcal{V}|$. Hence, the stationary distribution of the chain is the uniform distribution over the independent sets.

4.4.2 Random Walk on Cycle

A cycle is a graph $\mathcal{G} = (\mathbb{Z}_n, \mathcal{E})$ where $\{i, j\} \in \mathcal{E}$ if and only if $i = j \pm 1 \pmod n$. To ensure aperiodicity, we introduce laziness — we stay at the current state with probability $\frac{1}{2}$ and move to its neighbor each with probability $\frac{1}{4}$. The unique stationary distribution of the Markov chain is $(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$. We now study mixing time, the time needed to be close to stationary distribution for this Markov chain, using coupling technique.

As usual we take two copies $\mathcal{X} = \{X_i\}_{i \in \mathbb{N}}$ and $\mathcal{Y} = \{Y_i\}_{i \in \mathbb{N}}$ of the Markov chain. The Markov chain $\{Y_i\}_{i \in \mathbb{N}}$ starts with stationary distribution π. We now define a coupling between $\mathcal{X}$ and $\mathcal{Y}$: if $X_t = Y_t$ for any $t \in \mathbb{N}$, then $\mathcal{X}$ and $\mathcal{Y}$ move together from thereon. Otherwise, at every step, we toss a fair coin. If it comes head, then $\mathcal{X}$ stays at the same state whereas $\mathcal{Y}$ moves to one of its neighbor with equal probability. If the coin comes tail, then $\mathcal{Y}$ stays at the same state whereas $\mathcal{X}$ moves to one of its neighbor with equal probability.

Let us define another stochastic process $\mathcal{Z} = \{Z_i\}_{i \in \mathbb{N}}$ as $Z_i = X_i - Y_i \pmod n$. Observe that $\mathcal{Z}$ is a random walk on integers $[n-1] \cup \{0\}$ with 0 as an absorbing state (there is no outgoing transition from 0). We have seen in the analysis of the randomized 2SAT algorithm that the expected number of steps to reach state 0 is at most n^2. That is, if τ denotes the first time we have $X_t = Y_t$, then we have $\mathbb{E}[\tau] \leqslant n^2$. Using Markov inequality, we have the following:

$$\Pr[\tau \geqslant 2n^2] \leqslant \frac{1}{2}$$

So the probability that for $t \geqslant 2\lg(1/\varepsilon)n^2$, we have $X_t \neq Y_t$ is at most ε. Now using coupling lemma, we have $d_{TV}(X_t, \pi) \leqslant \varepsilon$ for $t \geqslant 2\lg(1/\varepsilon)n^2$. Hence the mixing time $t_{mix}(\varepsilon)$ for this Markov chain is at most $2\lg(1/\varepsilon)n^2$.

4.4.3 Shuffling Cards

In many games, we often shuffle a deck of cards with the hope that the permutation of the cards will be uniformly distributed over all permutations of the cards after shuffling is done. Here, a fundamental question is how many times shall we shuffle the deck to ensure that the cards are uniformly distributed with high probability. To answer this question, let us work with a simple model of shuffling – at every step, we pick a random card from the deck of n cards and put it on the top of every cards. It is immediately clear that the process is memoryless and thus can be modelled by a Markov chain. The state space of the Markov chain is the set of all $n!$ permutations of the cards. The shuffling process is simply a random walk on this Markov chain. Since every vertex has a self loop, the Markov chain is aperiodic. Also, it is clear that the chain is irreducible. From symmetry, it follows that the unique stationary distribution of the Markov chain is the uniform distribution over its states. To bound the mixing time, we consider the following coupling.

We take two copies $\mathcal{X} = \{X_i\}_{i\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_i\}_{i\in\mathbb{N}}$ of the Markov chain. We let $\mathcal{Y}$ start at stationary distribution and $\mathcal{X}$ start at any arbitrary distribution x. We define the following coupling between $\mathcal{X}$ and $\mathcal{Y}$– We pick a position j from $[n]$ uniformly at random. We move the j-th card C in X_t to the top position to arrive next state X_{t+1}. We also move the card C in X_{t+1} to the top position to arrive next state Y_{t+1}. It is clear that the coupling is valid – the each individual Markov chain evolves according to its own transition matrix. It can be easily proved using induction that, if some card C is chosen to be placed at the top position at t-th step, its position in both the processes $\mathcal{X}$ and $\mathcal{Y}$ remains same at every step from t onward. Hence, two process will become coupled once every card has been chosen at least once. This is exactly the coupon collector problem. Hence the probability that some specific card, C' say, has never been chosen after the Markov chain runs for $n\ln n + cn$ steps is

$$\left(1-\frac{1}{n}\right)^{n\ln n+cn} \leqslant e^{-(\ln n+c)} \leqslant \frac{e^{-c}}{n}$$

Hence, by union bound, the probability that all the cards have not been chosen at least once after $n\ln n + cn$ steps is at most e^{-c}. Hence, for $t \geqslant n\ln n + n\ln(1/\varepsilon)$, we have $\Pr[X_t \neq Y_t] \leqslant 1/\varepsilon$. If $\mathcal{P}$ is the transition matrix of the Markov chain, then by Coupling lemma, we have $d_{TV}(x\mathcal{P}^t, \pi) \leqslant 1/\varepsilon$ for all $t \geqslant n\ln n + n\ln(1/\varepsilon)$. Hence, the mixing time of the card shuffling Markov chain is $n\ln n + n\ln(1/\varepsilon)$ (compare this number with the number of states which is $n!$).

4.5 Hitting Time, Commute Time, and Cover Time

Let us now return to our first question – Given a start state i, what is the expected number of steps the Markov chain takes to reach some state j in the Markov chain? or given a start state i, what is the expected number of steps the Markov chain takes to reach some state j in the Markov chain and come back to i? The first time is called the hitting time and the second one is called the commute time.

Definition 4.5.1 (Hitting time, commute time, cover time)**.** *Given two states i and j of a Markov chain, the hitting time, denoted by $h_{i,j}$, is the expected number of steps that the Markov chain takes to reach the state j for the first time starting from state i. The commute time between i and j, denoted by $C_{i,j}$, is defined to be the expected number of steps that the Markov chain takes to reach j starting from i and then come back to i. That is, $C_{i,j} = h_{i,j} + h_{j,i}$. Let C_i denote the expected number of steps that the Markov chain takes to visit every state at least once starting from state i and ending at state i. The cover time of a Markov chain is defined as $\max_{i\in[n]} C_i$.*

We will restrict our attention to compute hitting time, commute time, cover time of random walks only. Note that for two states i and j, $h_{i,j}$ may not be equal to $h_{j,i}$ for every random walk. Can you find an example?

We now bound the cover time of any random walk. For that, we will use the following useful fact whose proof is out of scope of the course.

Fact 4.5.1. *Let $\mathcal{P}$ be the transition matrix of a finite, irreducibile, and aperiodic Markov chain with stationary distribution π. Then we have the following.*

1. *For every state $i \in [n]$, we have $h_{i,i} = \frac{1}{\pi_i}$*
2. *Let $N(i,t)$ denote the number of times the Markov chain visits the state i in first t steps. Then we have*

$$\lim_{t\to\infty} \frac{N(i,t)}{t} = \pi_i$$

Alternatively, we have $\lim_{t\to\infty} \Pr[X_t = i | X_0 = i] = \pi_i$.

An immediate corollary of Fact 4.5.1 is the following.

Corollary 4.5.1. *Consider a random walk on a undirected connected non-bipartite graph. Then we have $h_{i,i} = \frac{2|\mathcal{E}|}{deg(i)}$ for every state i.*

For a pair of states connected by an edge, we prove the following simple and useful bound.

Lemma 4.5.1. *Consider a random walk on a undirected connected non-bipartite graph. For $\{i,j\} \in \mathcal{E}$, we have $h_{j,i} < 2|\mathcal{E}|$.*

Proof. We have the following recursion.

$$\frac{2|\mathcal{E}|}{\deg(i)} = h_{i,i} = \frac{1}{\deg(i)} \sum_{j\in N(i)} (1 + h_{j,i})$$
$$\Rightarrow 2|\mathcal{E}| = \sum_{j\in N(i)} (1 + h_{j,i})$$

Hence we have $h_{j,i} < 2|\mathcal{E}|$. □

We now prove the central result for cover time for random walks.

Theorem 4.5.1. *Consider a random walk on a undirected connected non-bipartite graph $\mathcal{G}$. Then the cover time is at most $4mn$ where n and m denote the number of vertices and edges in the graph.*

Proof. Let $\mathcal{T}$ be any spanning tree of $\mathcal{G}$. Hence, there exists a tour (visiting all the vertices at least once) of length at most $2(n-1)$ (by simply traversing the edges of the spanning tree in both directions). Let such a tour be $v_0, v_1, \ldots, v_k (= v_0)$ for some $k \leqslant 2(n-1)$. Then by Lemma 4.5.1, the cover time C can be bounded as follow.

$$C \leqslant \sum_{i=0}^{k} h_{v_i, v_{i+1}} < 2km \leqslant 4mn$$

□

Chapter 5

Monte Carlo Methods

We have seen how one can estimate the probability of a coin coming head by tossing the coin some number of time (in the application of Chernoff bound). The guarantee we achieved was that, by tossing the coin $\mathcal{O}(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ times, our estimate is an ε factor approximation of the true probability with error probability at most δ. This kind of guarantee is called an (ε,δ)-approximation and this kind of technique is called Monte Carlo methods.

Definition 5.0.1 ((ε,δ)-Approximation)**.** *A randomized algorithm is said to provide an (ε,δ) approximation for some value $V(x)$ if the output $A(x)$ of the algorithm satisfies the following for every input instance x.*

$$\Pr[|A(x) - V(x)| \geqslant \varepsilon V(x)] \leqslant \delta$$

An (ε,δ) approximation algorithm is called a *fully polynomial randomized approximation scheme (FPRAS)* if it runs in time polynomial in $|x|, \frac{1}{\varepsilon}, \log\frac{1}{\delta}$. Hence, our algorithm for estimating the probability of the coin coming head is actually a FPRAS. We now see some more Monte Carlo methods which are actually an FPRAS.

5.1 Estimating π

Our first example of a Monte Carlo is to estimate the value of π and the idea is exactly same as estimating the probability of a coin coming head. We take a 2×2 square $\mathcal{S}$ in $\mathbb{R}^2$ and put a circle $\mathcal{C}$ of radius 1 inside it. We now draw ℓ uniformly random points $(x_i, y_i), i \in [\ell]$ from the square $\mathcal{S}$. Let $Z_i, i \in [\ell]$ be an indicator random variable for the event that $(x_i, y_i) \in \mathcal{C}, i \in [\ell]$. Then the probability that $Z_i = 1$ is exactly $\pi/4$ for $i \in [\ell]$. Let $W = \frac{4}{\ell}\sum_{i=1}^{\ell} Z_i$. Then we have $\mathbb{E}[W] = \pi$. We now have the following.

$$\Pr\left[|W-\pi| \geqslant \varepsilon\pi\right] = \Pr\left[\left|\sum_{i=1}^{\ell} Z_i - \pi\ell/4\right| \geqslant \varepsilon\pi\ell/4\right] \leqslant 2e^{-\frac{\varepsilon^2\ell\pi}{12}}$$

The inequality above follows from Chernoff bound. By choosing $\ell = \frac{12}{\pi\varepsilon^2}\ln\frac{2}{\delta}$, we get that $\Pr\left[|W-\pi| \geqslant \varepsilon\pi\right] \leqslant \delta$.

5.2 DNF Counting

In our examples of Monte Carlo methods so far, we have assumed that we can draw sample from some specific distribution. This may not be clear for some applications. One such scenario is the DNF counting problem. In the DNF counting problem, we are given a Boolean formula in disjunctive normal form (DNF). That is the formula is an OR of some m DNF clauses and each DNF clause is an AND of some literals over n variables. In the DNF counting problem, the goal is to compute the number of satisfying assignments of a formula. One can easily prove that the DNF counting problem is NP-hard (by obvious reduction from CNF-SAT). Hence, one cannot hope to have a polynomial time algorithm for solving the DNF counting problem exactly (assuming $P \neq NP$). We will now see a Monte Carlo based FPRAS for the DNF counting problem.

The obvious Monte Carlo method of checking how many uniformly random assignment of the variables satisfy the DNF formula does not work. One can actually prove that there exist DNF formulas with satisfying assignments for which one needs to check exponential number of random assignments to find any satisfying assignments. Intuitively, the problem with this approach is that we may be searching for needle in a haystack since the number of satisfying assignments may be few. Also this approach uses the formula only for checking whether any assignment is a satisfying assignment. We now see an FPRAS for DNF counting and the idea is to use the formula to substantially reduce the search space.

Let us assume without loss of generality that no clause contain both some variable and its negation; if there is any such clause, we simply remove it since that clause is never satisfiable. We observe that any clause containing t literals has exactly 2^{n-t} satisfying assignments. Let $C_j, j \in [m]$ be the clauses where the clause C_j contains t_j literals for $j \in [m]$ and S_j be the set of all satisfying assignments of the clause C_j. Then, by above observation, we have $|S_j| = 2^{n-t_j}$. Let us define $\mathcal{U} = \{(j, a) : j \in [m], a \in S_j\}$ and $\mathcal{S} = \{(j, a) \in S_j : j \in [m], (i, a) \notin S_i \text{ for every } i < j\}$. It is immediate that the set of satisfying assignments for the formula is in one to one correspondence with $\mathcal{S}$. Hence, our goal is to estimate $|\mathcal{S}|$. We observe that $|\mathcal{S}|/|\mathcal{U}| \geqslant 1/m$ (bounded away from 0). This follows from the observation that there exists a $j \in [m]$ such that $m|S_j| \geqslant |\mathcal{U}|$ (for example, pick the j with $\max|S_j|$). Hence, if we can sample uniformly at random from $\mathcal{U}$ such and check whether the sampled element belongs to $\mathcal{S}$, then we are done. We do this next.

We sample an element $(j, a) \in \mathcal{U}$ in the following manner. We first choose j with probability $|S_j|/\sum_{i=1}^{m}|S_i|$ and then choose a uniformly at random from S_j. Hence, the probability that an element $(j, a) \in \mathcal{U}$ is chosen by the above sampling procedure is $\frac{|S_j|}{|\mathcal{U}|}\frac{1}{|S_j|} = \frac{1}{|\mathcal{U}|}$. Also, given a sample $(j, a) \in S_j$ for some $j \in [m]$, one can easily check whether (j, a) belongs to $\mathcal{S}$. So our algorithm is as follows. We draw some ℓ number of samples from $\mathcal{U}$ uniformly randomly and find that k of them actually belong to $\mathcal{U}$. Then our algorithm outputs $\frac{k|\mathcal{U}|}{\ell}$. Now routine application of Chernoff bound shows that, for $\ell = \frac{3m}{\varepsilon^2}\ln\frac{2}{\delta}$, the above algorithm is an (ε, δ)-approximation algorithm for the DNF counting problem (and thus a FPRAS). Actually, we can prove the following more general and useful result.

Theorem 5.2.1 (Estimator Theorem)**.** *Suppose we wish to estimate $|\mathcal{S}|$ and we can draw samples from some underlying probability space $(\Omega, 2^{\Omega}, \mathcal{P})$ such that the probability $\frac{|\mathcal{S}|}{|\Omega|}$ that a sample belongs to $\mathcal{S}$ is at least ρ. Then the corresponding Monte Carlo method provides a (ε, δ) approximation if it draws $\frac{3}{\rho\varepsilon^2}\ln\frac{2}{\delta}$ samples.*

Proof. Let $\ell = \frac{3}{\rho\varepsilon^2}\ln\frac{2}{\delta}$, Y_i be an indicator random variable for the event that the i-th sample belongs to $\mathcal{S}$, and $Y = \sum_{i=1}^{\ell} Y_i$. Then, we have $\mathbb{E}[Y_i] \geqslant \rho$ and thus $\mathbb{E}[Y] \geqslant \rho\ell$. We now have the following.

$$\Pr\left[\left|\frac{|\Omega|Y}{\ell} - |\mathcal{S}|\right| \geqslant \varepsilon|\mathcal{S}|\right] = \Pr\left[\left|Y - \frac{\ell|\mathcal{S}|}{\Omega}\right| \geqslant \frac{\varepsilon\ell|\mathcal{S}|}{\Omega}\right]$$

$$
\begin{aligned}
&= \Pr\left[|Y - \mathbb{E}[Y]| \geqslant \varepsilon \mathbb{E}[Y]\right] \\
&\leqslant 2e^{-\varepsilon^2 \mathbb{E}[Y]/3} && \text{[Chernoff Bound]} \\
&\leqslant 2e^{-\varepsilon^2 \rho \ell/3} && [\mathbb{E}[Y] \geqslant \rho\ell] \\
&= \delta
\end{aligned}
$$

□

5.3 Approximate Sampling: FPAUS

Our examples of Monte Carlo method shows that there is a fundamental connection between sampling and counting – if we can sample from an appropriate space approximately uniformly, then we can approximately count. This motivates the following definition.

Definition 5.3.1 (Fully Polynomial Almost Uniform Sampler (FPAUS))**.** *A sampling distribution $(\Omega, 2^\Omega, \mathcal{P})$ is called an ε-uniform sample if $d_{TV}(\mathcal{P}, \mu_u) \leqslant \varepsilon$ where μ_u is the uniform distribution over $(\Omega, 2^\Omega)$. A sampling algorithm is called a fully polynomial almost uniform sampler if, for every input x and ε, it outputs an ε-uniform sample in time polynomial in x and $\ln \frac{1}{\varepsilon}$.*

Although we insist the running time to be polynomial in $\ln \frac{1}{\varepsilon}$ for FPAUS, often it is enough for the running time to be polynomial in $\frac{1}{\varepsilon}$ to obtain an FPRAS.

5.4 Markov Chain Monte Carlo Method: Counting Number of Independent Sets

Suppose we wish to count the number of independent sets of a given graph. A Monte Carlo based approach for this problem would need an almost uniform sampler as the first step. Now, it is not clear how we can sample an independent set uniformly or almost uniformly and Markov chain comes to rescue exactly here. We have seen a Markov chain on the set of all independent sets of a graph whose stationary distribution is the uniform distribution (exactly what we want). So, the idea is as follows. We let Markov chain run for some t steps. Then, for some large enough t, the fundamental theorem of Markov chains state that the distribution of X_t is close to its stationary distribution (which is the uniform distribution over all independent sets) irrespective of the start step. So, we can use X_t as our sample. The running time of the above sampler is dominated by t if every state has polynomial number of outgoing edges which is the case for the Markov chain on independent sets. How large should t be so that the distribution of X_t is close to the uniform distribution (that is, the total variation distance is at most ε)? This is exactly the question that mixing time answers. Let us postpone the issue of mixing time for the time being and assume that $t_{mix}(\varepsilon)$ is small enough which we will prove later assuming the maximum degree of the graph being at most 4.

Let $G = (V, E)$ be any graph with the set of edges being $E = \{e_1, e_2, \ldots, e_m\}$. We define $E_i = \{e_j : j \in [i]\}$ and $G_i = (V, E_i)$ for $i \in \{0, 1, \ldots, m\}$. Hence we have $G_m = G$. Let us denote by $\mathcal{I}(G_i)$ the set of independent sets of G_i. Our goal is to estimate $|\mathcal{I}(G_m)|$. On the other hand, we know $|\mathcal{I}(G_0)| = 2^n$. We express $|\mathcal{I}(G_m)|$ as follows.

$$
|\mathcal{I}(G)| = |\mathcal{I}(G_m)| = |\mathcal{I}(G_0)| \prod_{i=1}^{m} \frac{|\mathcal{I}(G_i)|}{|\mathcal{I}(G_{i-1})|}
$$

Let us define $r_i = \frac{|\mathcal{I}(\mathcal{G}_i)|}{|\mathcal{I}(\mathcal{G}_{i-1})|}$ for $i \in [m]$. Then we have the following.

$$|\mathcal{I}(\mathcal{G})| = 2^n \prod_{i=1}^{m} r_i$$

Hence to estimate $|\mathcal{I}(\mathcal{G})|$, it is enough to estimate r_i for every $i \in [m]$. For this, we assume that we can draw $\frac{\varepsilon}{6m}$-uniform sample from $\mathcal{I}(\mathcal{G}_i)$. The idea is to draw a of almost uniform samples from $\mathcal{I}(\mathcal{G}_{i-1})$ and consequently it will belong to $\mathcal{I}(\mathcal{G}_i)$ with probability (approximately) r_i. Intuitively speaking, if we show that $r_i \geqslant 1/2$, then Theorem 5.2.1 will ensure that small number of samples will be enough. The above statement would have been perfectly immediate if our samples were uniform. However, our samples are only approximately uniform. But, this does not change the result much as we show the following.

Lemma 5.4.1. *There exists a constant c such that the Monte Carlo method for estimating r_i using $c\frac{m^2}{\varepsilon^2}\ln\frac{2m}{\delta}$ many $\frac{\varepsilon}{6m}$-uniform samples provides a $\left(\frac{\varepsilon}{2m}, \frac{\delta}{m}\right)$-approximation for r_i.*

Proof. To have any hope to use Theorem 5.2.1, we first need to who that the probability r_i that a uniform sample from $\mathcal{I}(\mathcal{G}_{i-1})$ belongs to $\mathcal{I}(\mathcal{G}_i)$ is not too small. We observe that $\mathcal{G}_i$ and $\mathcal{G}_{i-1}$ differs only in one edge $e_i = \{u, v\}$. Hence an independent set $\mathcal{W}$ of $\mathcal{G}_{i-1}$ is also an independent set of $\mathcal{G}_i$ unless $u, v \in \mathcal{W}$. So the map from $\mathcal{I}(\mathcal{G}_{i-1}) \setminus \mathcal{I}(\mathcal{G}_i)$ to $\mathcal{I}(\mathcal{G}_i)$ where an independent set $\mathcal{W} \in \mathcal{I}(\mathcal{G}_{i-1}) \setminus \mathcal{I}(\mathcal{G}_i)$ goes to $\mathcal{W} \setminus \{v\}$ is injective. This implies that $|\mathcal{I}(\mathcal{G}_{i-1})| \leqslant 2|\mathcal{I}(\mathcal{G}_i)|$ and thus $r_i \geqslant 1/2$.

We next show that we can estimate r_i even with almost uniform samples from $\mathcal{I}(\mathcal{G}_{i-1})$. Suppose we draw ℓ samples. Let X_i be the indicator random variable for the event that the i-th sample from $\mathcal{I}(\mathcal{G}_{i-1})$ belongs to $\mathcal{I}(\mathcal{G}_i)$. Then, by definition of almost uniform sample, we have the following for every $i \in [\ell]$ where $\tilde{r}_i = \sum_{i=1}^{\ell} X_i/\ell$ is the estimate for r_i.

$$\begin{aligned}
&\left|\Pr[X_i = 1] - \frac{|\mathcal{I}(\mathcal{G}_i)|}{|\mathcal{I}(\mathcal{G}_{i-1})|}\right| \leqslant \frac{\varepsilon}{6m} \\
&\Rightarrow \left|\mathbb{E}[X_i] - \frac{|\mathcal{I}(\mathcal{G}_i)|}{|\mathcal{I}(\mathcal{G}_{i-1})|}\right| \leqslant \frac{\varepsilon}{6m} && \text{[Since } X_i \text{ is indicator r.v.]} \\
&\Rightarrow \left|\mathbb{E}\left[\frac{\sum_{i=1}^{\ell} X_i}{\ell}\right] - \frac{|\mathcal{I}(\mathcal{G}_i)|}{|\mathcal{I}(\mathcal{G}_{i-1})|}\right| \leqslant \frac{\varepsilon}{6m} && \text{[Linearity of expectation]} \\
&\Rightarrow |\mathbb{E}[\tilde{r}_i] - r_i| \leqslant \frac{\varepsilon}{6m} \\
&\Rightarrow \left|\frac{\mathbb{E}[\tilde{r}_i]}{r_i} - 1\right| \leqslant \frac{\varepsilon}{6mr_i} \\
&\leqslant \frac{\varepsilon}{3m} && [r_i \geqslant 1/2]
\end{aligned}$$

Since $r_i \geqslant 1/2$, for large enough m, we have $\mathbb{E}[\tilde{r}_i] \geqslant r_i - \varepsilon/(6m) \geqslant 1/3$. Now by standard application of (multiplicative version of) Chernoff bound, there exists a constant c such that for $\ell = c\frac{m^2}{\varepsilon^2}\ln\frac{2m}{\delta}$, we have the following.

$$\Pr\left[|\tilde{r}_i - \mathbb{E}[\tilde{r}_i]| \geqslant \frac{\varepsilon}{12m}\mathbb{E}[\tilde{r}_i]\right] = \Pr\left[\left|\frac{\tilde{r}_i}{\mathbb{E}[\tilde{r}_i]} - 1\right| \geqslant \frac{\varepsilon}{12m}\right] \leqslant \frac{\delta}{m}$$

The above inequality bounds $\tilde{r}_i$ with respect to $\mathbb{E}[\tilde{r}_i]$. However, we need to compare $\tilde{r}_i$ with r_i. Loosely speaking, $\frac{\tilde{r}_i}{\mathbb{E}[\tilde{r}_i]}$ should be close to $\frac{\tilde{r}_i}{r_i}$ since r_i is close to $\mathbb{E}[\tilde{r}_i]$ with high probability. Formally, we have the following with probability at least $1 - \frac{\delta}{m}$.

$$1 - \frac{\varepsilon}{2m} \leqslant \left(1 - \frac{\varepsilon}{3m}\right)\left(1 - \frac{\varepsilon}{12m}\right) \leqslant \frac{\tilde{r}_i}{r_i} \leqslant \left(1 + \frac{\varepsilon}{3m}\right)\left(1 + \frac{\varepsilon}{12m}\right) \leqslant 1 + \frac{\varepsilon}{2m}$$

□

The Monte Carlo method outputs ALG $= 2^n \prod_{i=1}^{m} \tilde{r}_i$. By union bound, we have $\frac{\tilde{r}_i}{r_i} \in [1 - \frac{\varepsilon}{2m}, 1 + \frac{\varepsilon}{2m}]$ for every $i \in [m]$ with probability at least $1 - \delta$. Then we have the following with probability at least $1 - \delta$.

$$\frac{\text{ALG}}{|\mathcal{I}(\mathcal{G})|} = \prod_{i=1}^{m} \frac{\tilde{r}_i}{r_i} \in \left[\left(1 - \frac{\varepsilon}{2m}\right)^m, \left(1 + \frac{\varepsilon}{2m}\right)^m\right] = [1 - \varepsilon, 1 + \varepsilon]$$

This concludes the proof that the above Monte Carlo based method provides an (ε, δ)-approximation for counting the number of independent sets of a graph.

5.5 The Path Coupling Technique

The following write up on path coupling is due to Prof. Jaikumar Radhakrishnan from TIFR Mumbai. I sincerely thank him for this.

Let $P = (p_{ij})$ be the $n \times n$ transition matrix of a Markov chain $\{\mathbf{X}_t\}$ on the set of states $[n]$. Here p_{ij} denotes $p_{j|i}$, the probability of moving to state j at time $t + 1$ given that the state is i at time t, that is, $p_{ij} = \Pr[\mathbf{X}_{t+1} = j | \mathbf{X}_t = i]$. We say that a transition matrix $Q = (q_{ij,i'j'})$ for a Makov chain with states in $[n] \times [n]$ is a coupling for the original Markov chain if for all $i, j \in [n]$:

$$\sum_{j'} q_{ij,i'j'} = p_{ii'}, \text{ for all } i' \in [n]\,; \tag{5.1}$$

$$\sum_{i'} q_{ij,i'j'} = p_{jj'}, \text{ for all } j' \in [n]\,. \tag{5.2}$$

It follows from this that if we consider the Markov chain $\{(\mathbf{X}_t, \mathbf{Y}_t)\}$ with transition matrix Q, then $\{\mathbf{X}_t\}$ and $\{\mathbf{Y}_t\}$ are both Markov chains with transition matrix P.

Theorem 5.5.1. *Suppose P is the transition matrix for a random walk on a connected graph G. Suppose that for all edges $(i, j) \in E(G)$, we have a distribution D_{ij} on $[n] \times [n]$ such that if $(\mathbf{X}, \mathbf{Y}) \sim D_{ij}$, then*

$$\Pr[\mathbf{X} = i'] = p_{ii'} \text{ for all } i'\,; \tag{5.3}$$

$$\Pr[\mathbf{Y} = j'] = p_{jj'} \text{ for all } j'\,; \tag{5.4}$$

$$\mathbb{E}\,[[]\, d(\mathbf{X}, \mathbf{Y})] \leqslant \alpha d(i, j). \tag{5.5}$$

(Note that $d(i, j) = 1$, when i and j are adjacent.) Then, for all pairs $(i, j) \in [n] \times [n]$ (not necessarily adjacent in G), there is a distribution D_{ij} on $[n] \times [n]$ such that if $(\mathbf{X}, \mathbf{Y}) \in D_{ij}$, then eqs. (5.3) to (5.5) hold.

Remark: The above theorem says that if suitable coupling distributions D_{ij} are available when i and j are adjacent, then this coupling can be extended to all pairs of states. Note that then the matrix $Q = (q_{ij,i'j'})$ with $q_{ij,i'j'} = D_{ij}(i'j')$ will be a valid coupling for the original transition matrix P of the random walk.

Notation: In the following, we will use the following notation.

1. For $i \in [n]$, we let D_i be the distribution on $[n]$ defined by $D_i(j) = p_{ij}$.

2. Similarly, for $(i,j) \in E(G)$, and $i' \in [n]$, let $D_{iji'}(j') = \Pr[\mathbf{Y} = j' | \mathbf{X} = i;]$ where $(\mathbf{X}, \mathbf{Y}) \sim D_{ij}$; that is,

$$D_{iji'}(j') = D_{ij}(i'j') / \sum_{\ell} D_{ij}(i'\ell).$$

3. For $(i,j) \in E(G)$ and $i' \in [n]$, let $\mathbf{X}_{iji'} \sim D_{iji'}$. We may visualize these as arranged in a three-dimensional array of random variables; we assume that these random variables are independent.

Claim 5.5.1. *Assume the coupling distributions D_{ij} for $(i,j) \in E(G)$ given in the theorem. Let $(i,j) \in E(G)$ and $\mathbf{X} \sim D_i$ be generated independently of the the variables $(\mathbf{X}_{iji'} : i' \in [n])$. Then, $(X, X_{ijX}) \sim D_{ij}$. In particular,*

$$\mathbf{X}_{ijX} \sim D_j\,; \tag{5.6}$$

$$d(\mathbf{X}, \mathbf{X}_{ijX}) \leqslant \alpha\,. \tag{5.7}$$

Proof. This follows from the definition of $\mathbf{X}$ and properties of the coupling distribution D_{ij}. □

Proof of Theorem 5.5.1

The following randomized procedure, called PathCoupling(i,j), outputs a pair of random variables $(X,Y) \in [n] \times [n]$; the distribution of this pair of random variables will be the D_{ij} we seek in the theorem.

Algorithm 1 PathCoupling(i,j) {The procedures outputs $(\mathbf{X}, \mathbf{Y}) \sim D_{ij}$}

1: Fix a shortest (i,j)-path in G: $i = k_0, k_1, k_2, \ldots, k_d = j$.
2: **for** $i, j, i' \in [n]$ **do**
3: **generate:** $\mathbf{X}_{iji'} \sim D_{iji'}$ (a three dimensional array of independent random variables)
4: **end for**
5: **generate:** $\mathbf{X}_0 \sim D_i$
6: **for** $\ell = 1, 2, \ldots, d$ **do**
7: $\mathbf{X}_\ell \leftarrow \mathbf{X}_{k_{\ell-1}, k_\ell, X_{\ell-1}}$
8: **end for**
9: **output:** $(\mathbf{X}_0, \mathbf{X}_d)$

Claim 5.5.2. *Suppose $d(i,j) = d$. Consider a run of* PathCoupling(i,j) *with output $(\mathbf{X}, \mathbf{Y})$. Then,*

(a) $\mathbf{X} \sim D_i$*;*

(b) $\mathbf{Y} \sim D_j$*;*

(c) for $\ell = 0, 1, \ldots, d$, we have $\mathbf{X}_\ell \sim D_{k_\ell}$;

(d) for $\ell = 0, 1, \ldots, d-1$, we have $(\mathbf{X}_\ell, \mathbf{X}_{\ell+1}) \sim D_{k_\ell, k_{\ell+1}}$

Proof. For $i = j$, the output is $(\mathbf{X}_0, \mathbf{X}_0)$, and the claim is straightforward from line 5; so we assume $i \neq j$ and $d \geqslant 1$. We first show parts (c) and (d) by induction; parts (a) and (b) follow from part (c) by taking $\ell = 0$ and $\ell = d$, respectively. The base case of part (c), corresponding to $\ell = 0$, follows from line 5; then part (d) follows for $\ell = 0$ from Claim 5.5.1. In general, assuming $X_\ell \sim D_{k_\ell}$, we conclude from Claim 5.5.1 that $(\mathbf{X}_\ell, \mathbf{X}_{\ell+1}) \sim D_{k_\ell, k_{\ell+1}}$, and then use eq. (5.6) to conclude that $\mathbf{X}_{\ell+1} \sim D_{k_{\ell+1}}$. (Note that for Claim 5.5.1, we need that $\mathbf{X}_\ell$ is generated independently of the variables $(\mathbf{X}_{k_\ell, k_{\ell+1}, i'} : i' \in [n])$; this holds because the k_i are distinct.) □

Claim 5.5.3. *Let* $(\mathbf{X}, \mathbf{Y}) \sim D_{i,j}$. *Then,* $\mathbb{E}[[] d(\mathbf{X}, \mathbf{Y})] \leqslant \alpha d(i,j)$.

Proof. Let $d = d(i,j)$. Then, for $(\mathbf{X}, \mathbf{Y})$ generated by $\mathsf{PathCoupling}(i,j)$, we have

$$\begin{aligned}
\mathbb{E}[[] d(\mathbf{X}, \mathbf{Y})] = \mathbb{E}[[] d(\mathbf{X}_0, \mathbf{X}_d)] &\leqslant \mathbb{E}[[] d(\mathbf{X}_0, \mathbf{X}_1) + d(\mathbf{X}_1, \mathbf{X}_2) + \cdots + d(\mathbf{X}_{d-1}, \mathbf{X}_d)] && (5.8)\\
&= \mathbb{E}[[] d(\mathbf{X}_0, \mathbf{X}_1)] + \mathbb{E}[[] d(\mathbf{X}_1, \mathbf{X}_2)] + \cdots + \mathbb{E}[[] d(\mathbf{X}_{d-1}, \mathbf{X}_d)] && (5.9)\\
&\leqslant \alpha + \alpha + \cdots + \alpha \quad \text{(by part (c) of the above claim and eq. (5.5))} && (5.10)\\
&= \alpha d(i,j). && (5.11)
\end{aligned}$$

□

Hence what we have essentially argued above is that, to prove polynomial mixing time, it is enough to give an edge coupling and show that the expected distance between X_t ***and*** Y_t ***does not increase from*** X_{t-1} ***and*** Y_{t-1} ***and the distance remains same with probability at least*** $1 - \frac{1}{\mathit{poly}(n)}$.

Chapter 6

Probabilistic Method

On a high level, we have so far seen how randomization could be used to design efficient algorithms. Before continuing to do so, let us take a brief detour and see how randomization could be elegantly used as a proof technique. The probabilistic method is one of the most powerful tools for proving existence of objects with certain properties. On a very high level, suppose we wish to prove that some kind of objects (graph for example) exist with some properties (nonexistence of any short cycle for example). A straightforward way to prove such claim is to explicitly construct such an object. However, in many scenarios, explicitly constructing such objects may be non-trivial. In such scenarios probabilistic method can help us prove such claim without explicitly constructing such objects. The idea, on a very high level, is as follows. You prove that a random object (say a random graph) has the desired property (say nonexistence of any short cycle) with non-zero probability. It then follows from basic probability that there indeed exists such an object with desired properties (otherwise the probability would have been zero). Let us now see some concrete examples.

6.1 Basic Method

6.1.1 Ramsey Number

Our first example is bounding Ramsey number. For any two positive integers k, ℓ, the corresponding Ramsey number $R(k, \ell)$ is defined as follows.

$$R(k, \ell) = \min\{n \in \mathbb{N} : \text{any graph on } n \text{ vertices either contains a clique of size } k \text{ or an independent set of size } \ell\}$$

It is not even clear from the definition whether the Ramsey number is finite for any given k and ℓ. The Ramsey theorem says that $R(k, \ell)$ is finite for every k and ℓ. However, computing exact values of $R(k, \ell)$ is almost always non-trivial except for small values of k and ℓ. We prove the following lower bound for $R(k, k)$.

Theorem 6.1.1. *For any $k \geqslant 3$, we have $R(k, k) > 2^{\frac{k}{2}-1}$.*

Proof. Let $\mathcal{G}$ be a random graph on n vertices – between every pair of vertices $i, j \in [n], i \neq j$, $\mathcal{G}$ has an edge between i and j with probability $1/2$ independent of everything else. Let $\mathcal{A} \subseteq [n]$ be any subset of k vertices. Then the probability that $\mathcal{A}$ forms either a clique or an independent set is $2^{-\binom{k}{2}+1}$. Now, by union bound,

the probability that there exists a clique or an independent set of size k is at most $\binom{n}{k}2^{-\binom{k}{2}+1}$. We have the following for $n = 2^{\frac{k}{2}-1}$ which proves the theorem.

$$\binom{n}{k}2^{-\binom{k}{2}+1} \leqslant n^k 2^{-\frac{k^2}{2}+\frac{k}{2}+1} \leqslant 2^{\frac{k^2}{2}-k}2^{-\frac{k^2}{2}+\frac{k}{2}+1} \leqslant 2^{1-\frac{k}{2}} \leqslant \frac{1}{2}$$

□

6.2 Argument Using Expectation

Sometimes, directly showing that the probability of some event is non-zero may not be obvious. One way out is such cases is to argue using expectation. The idea is as follows. Any random variable X takes values at least $\mathbb{E}[X]$ with non-zero probability (otherwise the expectation would have been less). Let us see an example. A graph is called a *tournament* if it is directed and for every pair $i, j, i \neq j$ of vertices either we have an edge from i to j or from j to i.

Theorem 6.2.1. *There exists a tournament graph on n vertices which has at least $\frac{n!}{2^{n-1}}$ Hamiltonian paths.*

Proof. Let G be a random tournament on n vertices – between every pair of vertices $i, j \in [n], i \neq j$, G has an edge from i to j with probability $1/2$ or from j to i with probability $\frac{1}{2}$ independent of everything else. Let p be a any sequence of vertices of G. The probability that p is a Hamiltonian path is 2^{1-n}. Let X_p denote the indicator random variable for the event that p is a Hamiltonian path in G. Then, by linearity of expectation, the expected number of Hamiltonian paths in G is $\frac{n!}{2^{n-1}}$. Hence, there exists a tournament which contains $\frac{n!}{2^{n-1}}$ Hamiltonian paths. □

6.3 Alteration

Sometimes, it is not straightforward to show that the probability of good event is non-zero. The idea of alternation is to show the probability of an almost good event to be non-zero and use the existence of almost good object to conclude existence of a good object. Again, let us see this trick with an example.

Theorem 6.3.1 (Weak Turán Theorem)**.** *Let G be any graph on n vertices and m edges. Its average degree d is $\frac{2m}{n} \geqslant 1$. Then there exists an independent set in G of size at least $\frac{n}{2d}$*[1]*.*

Proof. Let us construct a random subset S of vertices as follows. Every vertex $i \in [n]$ belongs to S with probability p (we will fix the value of p later). Let V_S and E_S denote the number of vertices and the number of edges in the induced graph on S. Then by linearity of expectation, we have the following.

$$\mathbb{E}[V_S] = np, \mathbb{E}[E_S] = mp^2 = \frac{1}{2}dnp^2$$

By choosing $p = 1/d$, we have $\mathbb{E}[V_S] = n/d, \mathbb{E}[E_S] = mp^2 = n/2d$. Hence the induced graph on S has n/d vertices and $n/2d$ edges. We now remove one end point of every edge to have a subset S' of S which contains at least $n/2d$ vertices and no edges (and thus an independent set). □

[1] For Turán Theorem, this bound is $\frac{n}{d+1}$ which is tight in general.

6.4 Lovász Local Lemma

On a high level, in applications of probabilistic methods, we wish to prove that the probability of something bad to happen is less than 1. Typically, we have a collection of bad events $\mathcal{A}_i, i \in [n]$ with $\Pr[\mathcal{A}_i] < 1$ for every $i \in [n]$ and we wish to claim that $\Pr\left[\bigcap_{i=1}^{n} \overline{\mathcal{A}}_i\right] > 0$. Without any assumption on the events, we can use union bound to have $\Pr\left[\bigcap_{i=1}^{n} \overline{\mathcal{A}}_i\right] = 1 - \Pr\left[\bigcup_{i=1}^{n} \mathcal{A}_i\right] \geqslant 1 - \sum_{i=1}^{n} \Pr[\mathcal{A}_i]$. The above bound can be too lose in many cases. If we have independence of $\mathcal{A}_i, i \in [n]$, then we can have $\Pr\left[\bigcap_{i=1}^{n} \overline{\mathcal{A}}_i\right] = \prod_{i=1}^{n} \Pr[\overline{\mathcal{A}}_i] > 0$. So, intuitively, if we have "limited independence," one could expect to have something which is stronger than union bound but weaker than product formula. Lovász local lemma makes this intuition concrete. The dependency relation among the set of events $\mathcal{A}_i, i \in [n]$ can be captured by what is called a *dependency graph*.

Definition 6.4.1 (Dependency Graph)**.** *Let $\mathcal{A}_i, i \in [n]$ be any n events in a common underlying probability space. A directed graph $\mathcal{G} = (\mathcal{V} = [n], \mathcal{E})$ is called a dependency graph for $\mathcal{A}_i, i \in [n]$ if, for every $i \in [n]$, the event $\mathcal{A}_i$ is independent of every event $\mathcal{A}_j$ with $(i, j) \notin \mathcal{E}, j \in [n], j \neq i$.*

The general version of Lovász local lemma is the following.

Lemma 6.4.1 (Lovász Local Lemma)**.** *Let $\mathcal{A}_i, i \in [n]$ be any n events in a common underlying probability space with $\mathcal{G} = (\mathcal{V} = [n], \mathcal{E})$ being their dependency graph. Let $x_i \in [0, 1), i \in [n]$ be real numbers which satisfy the following for every $i \in [n]$.*

$$\Pr[\mathcal{A}_i] \leqslant x_i \prod_{(i,j) \in \mathcal{E}} (1 - x_j)$$

Then we have the following.

$$\Pr\left[\bigcap_{i=1}^{n} \overline{\mathcal{A}}_i\right] \geqslant \prod_{i=1}^{n} (1 - x_i) > 0$$

Proof. The high level idea of the proof is the following. If simultaneous occurrence of some subset $\overline{\mathcal{A}}_j, j \in \mathcal{S} \subseteq [n]$ of events guarantee that some other event $\mathcal{A}_k$ happens with probability 1, then we do not have any hope proving the statement. So we need to bound (from above) the probability of any event $\mathcal{A}_k$ happens given the events $\overline{\mathcal{A}}_j, j \in \mathcal{S} \subseteq [n]$ happen simultaneously. We prove this by induction on $|\mathcal{S}|$. Formally, we claim that, for any $\mathcal{S} \subseteq [n]$ and $k \in [n] \setminus \mathcal{S}$, $\Pr[\mathcal{A}_k | \bigcap_{i \in \mathcal{S}} \overline{\mathcal{A}}_i] \leqslant x_k$. We prove it by induction on $|\mathcal{S}|$.

(Base case) For $|\mathcal{S}| = 0$, the result follows from the assumption since $\Pr[\mathcal{A}_i] \leqslant x_i \prod_{(i,j) \in \mathcal{E}} (1 - x_j) \leqslant x_i$.

In the inductive step, we assume for any $|\mathcal{S}| \leqslant \ell$ and we need to prove for any $|\mathcal{S}| = \ell + 1$. Let $k \in [n] \setminus \mathcal{S}$ be any integer. Let $\mathcal{S}_1 = \{i \in \mathcal{S} : (k, i) \in \mathcal{E}\}$. If, $\mathcal{S}_1 = \emptyset$, then the result follows from the base case. So let us assume without loss of generality that $\mathcal{S}_1 \neq \emptyset$. We now have the following.

$$\Pr\left[\mathcal{A}_k \middle| \bigcap_{i \in \mathcal{S}} \overline{\mathcal{A}}_i\right] = \frac{\Pr\left[\mathcal{A}_k \bigcap_{i \in \mathcal{S}_1} \overline{\mathcal{A}}_i \middle| \bigcap_{i \in \mathcal{S} \setminus \mathcal{S}_1} \overline{\mathcal{A}}_i\right]}{\Pr\left[\bigcap_{i \in \mathcal{S}_1} \overline{\mathcal{A}}_i \middle| \bigcap_{i \in \mathcal{S} \setminus \mathcal{S}_1} \overline{\mathcal{A}}_i\right]}$$

We now bound the numerator as below.

$$\Pr\left[\mathcal{A}_k \bigcap_{i \in \mathcal{S}_1} \overline{\mathcal{A}}_i \middle| \bigcap_{i \in \mathcal{S} \setminus \mathcal{S}_1} \overline{\mathcal{A}}_i\right] \leqslant \Pr\left[\mathcal{A}_k \middle| \bigcap_{i \in \mathcal{S} \setminus \mathcal{S}_1} \overline{\mathcal{A}}_i\right] = \Pr[\mathcal{A}_k] \leqslant x_k \prod_{(k,j) \in \mathcal{E}} (1 - x_j)$$

We now bound the denominator as below. Let $S_1 = \{i_1, i_2, \ldots, i_r\}$.

$$\begin{aligned}
&\Pr\left[\bigcap_{i\in S_1} \overline{\mathcal{A}}_i \middle| \bigcap_{i\in S\setminus S_1} \overline{\mathcal{A}}_i\right] \\
&= \Pr\left[\overline{\mathcal{A}}_{i_1} \middle| \bigcap_{i\in S\setminus S_1} \overline{\mathcal{A}}_i\right] \Pr\left[\overline{\mathcal{A}}_{i_2} \middle| \overline{\mathcal{A}}_{i_1} \bigcap_{i\in S\setminus S_1} \overline{\mathcal{A}}_i\right] \cdots \Pr\left[\overline{\mathcal{A}}_{i_r} \middle| \overline{\mathcal{A}}_{i_1} \cap \cdots \overline{\mathcal{A}}_{i_{r-1}} \bigcap_{i\in S\setminus S_1} \overline{\mathcal{A}}_i\right] \\
&\geqslant \prod_{(k,j)\in\mathcal{E}} (1 - x_j)
\end{aligned}$$

From the bounds on the numerator and the denominator, we have $\Pr[\mathcal{A}_k | \bigcap_{i\in S} \overline{\mathcal{A}}_i] \leqslant x_k$. Now the statement of the lemma follows easily.

$$\Pr\left[\bigcap_{i=1}^{n} \overline{\mathcal{A}}_i\right] = \Pr[\overline{\mathcal{A}}_1] \Pr[\overline{\mathcal{A}}_2|\overline{\mathcal{A}}_1] \cdots \Pr\left[\overline{\mathcal{A}}_n \middle| \bigcap_{i=1}^{n-1} \overline{\mathcal{A}}_i\right] \geqslant \prod_{i=1}^{n} (1 - x_i) > 0$$

□

The above general version of the Lovász local lemma is often difficult to apply since we need to guess the values of x_i. The following simplified version known as the symmetric Lovász local lemma is often easier to apply.

Corollary 6.4.1 (Symmetric Lovász Local Lemma)**.** *Let $\mathcal{A}_i, i \in [n]$ be any n events in a common underlying probability space with $\mathcal{G} = (\mathcal{V} = [n], \mathcal{E})$ being their dependency graph. Suppose we have $\Pr[\mathcal{A}_i] \leqslant p$ for every $i \in [n]$ and the out degree of every vertex in $\mathcal{G}$ is at most d. If $ep(d+1) \leqslant 1$, then $\Pr\left[\bigcap_{i=1}^{n} \overline{\mathcal{A}}_i\right] > 0$.*

Proof. If $d = 0$, then the statement follows from the product rule of probability. Otherwise, setting $x_i = \frac{1}{d+1}$ for every $i \in [n]$ satisfies the assumption in the Lovász local lemma as can be seen below.

$$x_i \prod_{(i,j)\in\mathcal{E}} (1 - x_j) \geqslant \frac{1}{d+1}\left(1 - \frac{1}{d+1}\right)^d \geqslant \frac{1}{e(d+1)} \geqslant p$$

Now the statement follows from the Lovász local lemma. □

Let us see an application of Lovász local lemma.

Theorem 6.4.1. *If no variable in an exact k-SAT formula appears in more than $\frac{1}{k}\left(\frac{2^k}{e} - 1\right)$ clauses, then there formula is satisfiable.*

Proof. Consider independently assigning each variable to TRUE or FALSE with equal probability. Let E_j be the event that the assignment does not satisfy the j-th clause for $j \in [m]$. Then we have $\Pr[E_j] = 2^{-k} = p$ (say). From the assumption, the maximum degree d in any dependency graph is at most $\frac{2^k}{e} - 1$. Hence we have the following.

$$ep(d+1) \leqslant e2^{-k}\frac{2^k}{e} \leqslant 1$$

Hence, by the symmetric Lovász local lemma, we have $\Pr\left[\bigcap_{j=1}^{m} \overline{E}_j\right] > 0$ and thus there exists a satisfying assignment for the formula. □

Chapter 7

Derandomization Using Conditional Expectation

Suppose we have a randomized algorithm $\mathcal{A}$ for some problem Π which uses k random bits. Let us denote these random bits as $X_1, X_2, \ldots, X_k$. Suppose we have shown that the randomized algorithm is a α factor approximation algorithm. Suppose ALG is the random variable denoting the value of the output of the randomized algorithm. Then, $\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)] \geqslant \alpha OPT$ (for maximization problem). The method of conditional expectation is a general framework for designing a polynomial time deterministic algorithm with value of the output at least $\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)]$. Suppose the randomized algorithm samples $X_1, X_2, \ldots, X_k$ in this sequence. $\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)]$ can be written as follows. Suppose $X_1 \sim Ber(p_1)$.

$$\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)] = p_1\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = 1] + (1 - p_1)\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = 0]$$

Now, from averaging principle, it follows that,

$$\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)] \leqslant \max\{\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = 1], \mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = 0]\}$$

On a high level, the deterministic follows the randomized algorithm exactly, except when the randomized algorithm samples X_1, the deterministic algorithm sets $X_1 = 1$ if $\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = 1] \geqslant \mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = 0]$ and 0 otherwise. In general, say when the randomized algorithm samples X_i and deterministic algorithm have already set $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$, the deterministic algorithm sets $X_i = 1$ if $\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_i = 1] \geqslant \mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_i = 0]$ and 0 otherwise. It follows inductively that, $\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_i = x_i] \geqslant \mathbb{E}[ALG(X_1, X_2, \ldots, X_k)]$. Hence, when the deterministic algorithm terminates, we have $ALG_{\text{deterministic}} = ALG(X_1 = x_1, \ldots, X_k = x_k) \geqslant \mathbb{E}[ALG(X_1, X_2, \ldots, X_k)]$. In particular, if the randomized algorithm is a α factor approximation algorithm, the deterministic algorithm is also a α factor approximation algorithm. This technique is called the method of conditional expectation. Note that, to be able to apply the above technique, one needs to be able to compute $\mathbb{E}[ALG(X_1, X_2, \ldots, X_k)|X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_i = x_i]$ in polynomial time for every $i \in [n]$ which may not be always straightforward. Let us now see a concrete example.

In the E3SAT problem, the input is a set of m 3SAT clauses over n Boolean variables and the goal is to set these variables to Boolean values which satisfies a maximum number of clauses. There is an easy randomized $\frac{7}{8}$ approximation algorithm for the E3SAT problem – simply set each of the n variables to TRUE or FALSE with equal probability independently. Let Z_j be the indicator random variable for the event that j-th clause is satisfied for $j \in [m]$ and $\mathcal{Z} = \sum_{j=1}^{m} Z_j$ denotes the number of satisfied clauses. Then we have $\mathbb{E}[Z_j] = \frac{7}{8}, \mathbb{E}[\mathcal{Z}] = \sum_{j=1}^{m} \mathbb{E}[Z_j] = \frac{7m}{8}$. Since any assignment satisfies at most m clauses, it follows that the above algorithm achieves an approximation ratio of $\frac{7}{8}$.

Now, the above de-randomization technique applied straightforwardly as follows. Suppose the variables are $x_i, i \in [n]$. We set $x_1 = \text{TRUE}$ if $\mathbb{E}[Z|x_1 = \text{TRUE}] \geqslant \mathbb{E}[Z|x_1 = \text{FALSE}]$ and $x_1 = \text{FALSE}$ otherwise. Iteratively, in the i-th iteration, suppose we have set the variables $x_1, \ldots, x_{i-1}$ to $a_1, \ldots, a_{i-1} \in \{\text{TRUE}, \text{FALSE}\}$ respectively. Then we set $x_i = \text{TRUE}$ if $\mathbb{E}[Z|x_1 = a_1, \ldots, x_{i-1} = a_{i-1}, x_i = \text{TRUE}] \geqslant \mathbb{E}[Z|x_1 = a_1, \ldots, x_{i-1} = a_{i-1}, x_i = \text{FALSE}]$ and $x_i = \text{FALSE}$ otherwise. Also it is immediate that $\mathbb{E}[Z|x_1 = a_1, \ldots, x_{i-1} = a_{i-1}, x_i = a_i]$ can be computed in polynomial time for every $a_1, \ldots, a_i \in \{\text{TRUE}, \text{FALSE}\}$ (what is the algorithm?). Hence, it follows that the deterministic algorithm also achieves an approximation ratio of $\frac{7}{8}$.

Chapter 8

Hashing

A hash function $h : \mathcal{U} \longrightarrow \mathcal{M}$ is a function which maps some universe $\mathcal{U}$ to a set $\mathcal{M}$. Two elements $i, j \in \mathcal{U}, i \neq j$ are said to collide under h if $h(i) = h(j)$. Hash tables are often used as a data structure which guarantees various operations like search, insertion, etc. in (either worst case or average case) $\mathcal{O}(1)$ time. Typically, the universe $\mathcal{U}$ is much larger than $\mathcal{M}$ and thus the function $\mathcal{M}$ will not be injective (and hence there will be collisions). However, often the design goal is to minimize the effect of collision as far as possible. Usually, "the description of h" is available to adversaries and thus the adversary will always be able to "create inputs" so that the number of collisions is more which in turn degrades system performance. The idea of universal hashing addresses this issue.

8.1 Universal Hashing

As argued, any fixed hash function can be exploited by an adversary to cook up a "bad input." So the idea is to not fix the hash function h apriori. Instead, fix a set $\mathcal{H}$ of hash functions and select a hash function h from $\mathcal{H}$ uniformly randomly at run time (hence, even the system designer also does not know which hash function will be used at run time). The set $\mathcal{H}$ of hash functions is called a *hash family*. Intuitively speaking, a hash family is called universal if a random function from $\mathcal{H}$ looks like a random function from $\mathcal{U}$ to $\mathcal{M}$. However, it can be proved that, for a hash function h, picked uniformly randomly from $\mathcal{H}$, to behave exactly like a random function, the family $\mathcal{H}$ should be the set of all functions from $\mathcal{U}$ to $\mathcal{M}$ (there are $|\mathcal{M}|^{|\mathcal{U}|}$ possible functions) which is computationally not appealing. So, we ask if it is possible to get "limited random" behavior by much smaller hash family. This brings us to the notion of k-universal hash family (also known as k-way independent hash family). A family $\mathcal{H}$ of hash functions from $\mathcal{U}$ to $\mathcal{M}$ is called k-universal if for any distinct $x_1, x_2, \ldots, x_k \in \mathcal{U}$ and any $y_1, y_2, \ldots, y_k \in \mathcal{M}$, we have $\Pr[h(x_1) = y_1, \ldots, h(x_k) = y_k] = \frac{1}{m^k}$ (what is this probability over?) where m is $|\mathcal{M}|$. It follows that, for a k-universal hash family $\mathcal{H}$, $\Pr[h(x_1) = h(x_2) = \cdots = h(x_k)] = m^{-(k-1)}$ for every $x_1, x_2, \ldots, x_k \in \mathcal{U}$. The 2-universal hash family is particularly interesting due to its application in numerous different context. Before seeing a construction of a 2-universal hash family, let us see some use cases of it.

8.1.1 Application of Universal Hashing: Data Structure

Arguably the most immediate application of hashing is hash tables. We wish to store some set $S \subset \mathcal{U}$ of n elements in a way so that we can perform various operations like search, insertion, deletion, etc. quickly. A popular among such data structures is a hash table. In a typical hash table, we keep an m-length array $\mathcal{A}$ of "buckets". There is a hash function $h : \mathcal{U} \longrightarrow [m]$. To insert an element $x \in \mathcal{U}$, we simply put x in the bucket $\mathcal{A}[h(x)]$. To search for an element $x \in \mathcal{U}$, we search in the bucket $\mathcal{A}[h(x)]$. To delete an element $x \in \mathcal{U}$, we delete x from the bucket $\mathcal{A}[h(x)]$. It immediately follows that the efficiency of the above operations depends on the size of the buckets and how quickly we can find $h(x)$ given x. Typically, $h(x)$ can be computed quickly from x. So the performance depends solely on size of the buckets. Obviously, the size of individual buckets depends on the input set S and the hash function h. Since size of $\mathcal{U}$ is often much larger than the size of S, for any hash function h, one can find a set S so that the operations take (close to) linear time. To avoid this situation, the idea is to not fix the hash function h at design time; instead fix only a hash family $\mathcal{H}$ and pick a hash function h from $\mathcal{H}$ uniformly at random at run time. It turns out that if the hash family $\mathcal{H}$ is 2-universal, then we obtain what we want – insertion, search, and deletion can be performed quickly for any S. Let $S = \{x_1, x_2, \ldots, x_n\} \subseteq \mathcal{U}$. Suppose we wish to search/insert/delete some $y \in \mathcal{U}$. We will show that the expected size of the bucket $\mathcal{A}[h(y)]$ is $\mathcal{O}(1)$ for $n = m$ which immediately implies that all the three operations could be performed in $\mathcal{O}(1)$ expected time.

Let X_i be the indicator random variable for the event that $h(y) = h(x_i)$. Let Z be the random variable denoting the size of the bucket $\mathcal{A}[h(y)]$; that is $Z = \sum_{i=1}^{n} X_i$. Then, we have $\Pr[h(y) = h(x_i)] = m^{-1}$ since $\mathcal{H}$ is a 2-universal hash family. Hence, $\mathbb{E}[X_i] = m^{-1}$ and, by linearity of expectation, we have $\mathbb{E}[Z] = n/m = 1$.

8.1.2 Application of Universal Hashing: Perfect Hashing

The above application shows that, using 2-universal hash family, one can build data structure which guarantees $\mathcal{O}(1)$ time per operation in expectation. Can we improve this guarantee to worst case? Perfect hashing achieves this goal under the assumption that the set $\mathcal{S}$ that we wish to store is static, for example, the set of words in the English dictionary. For simplicity, let us assume $\mathcal{S} = [n]$ and we wish them to store in a table $\mathcal{T}$ of size $m = \lambda n$ for some constant λ (let the requirement decide the value of λ). Suppose we hash $\mathcal{S}$ to $\mathcal{T}$ using a hash function picked uniformly at random from a 2-universal hash family. Let us first bound the number of collisions. For $x, y \in \mathcal{S}, x \neq y$, let $C_{x,y}$ be the indicator random variable for the event that x and y collides. Then we have, $\mathbb{E}[C_{x,y}] = 1/m$. Let the random variable C denotes the number of collisions. Then we have the following.

$$\mathbb{E}[C] = \sum_{x,y \in \mathcal{S}, x \neq y} \mathbb{E}[C_{x,y}] = \binom{n}{2}/m = \mathcal{O}(n)$$

By Markov's inequality, the probability that the number of collisions is more than $\mathcal{O}(n)$ is at most $1/10$. If $m = \Omega(n^2)$, then we would have achieved what we want with few tries (in expectation). But the goal is to achieve $\mathcal{O}(1)$ look up using $\mathcal{O}(n)$ space. The idea is to use two level hash function – have a hash function picked uniformly randomly from a 2-universal hash family and use another hash function again picked uniformly randomly from a 2-universal hash family per bucket to resolve collision. Let, the size of the buckets of $\mathcal{T}$ be $b_1, b_2, \ldots, b_m$ in expectation. Now hash the elements in bucket i (which has b_i elements) to another table $\mathcal{T}_i$ of size $\mathcal{O}(b_i^2)$ and the above calculation shows that a hash function can be chosen using few expected tries which makes no collision in $\mathcal{T}_i$. So, the total space used is $\mathcal{O}(n) + \sum_{i=1}^{m} \mathcal{O}(b_i^2)$. We are done if we prove $\sum_{i=1}^{m} b_i^2 = \mathcal{O}(n)$ which we do next.

Observation 8.1.1. *Suppose* $b_i, i \in [m]$ *are as defined above. Then we have* $\sum_{i=1}^{m} b_i^2 = \mathcal{O}(n)$.

Proof. The total number of collisions in expectation is $\sum_{i=1}^{m} b_i^2$ which we have already shown to be $\mathcal{O}(n)$. □

Perfect hashing is also known as the FKS hashing after the name of the inventors Fredman, Komlós, and Szemerédi [FKS84].

8.1.3 Application of Universal Hashing: Data Streaming

In the streaming model, the data is huge and thus one can never store the entire data in any fast storage device at some place (in these applications, data is often distributed in different locations and stored in low cost slow devices like magnetic tape, etc.). For example, think of a network router which observes the stream of IP packets flowing through it or a very large data base application. In these settings, it is desirable that your algorithm makes only one pass of the data (note that, this restriction is forced in some applications like Internet router). If n denotes the size of the data, then we are given $\text{poly}(\log n)$ amount of data storage capacity. One can intuitively see that most of the interesting functions of the data can not be computed exactly in the restricted data streaming model (needless to say that one can formally prove this statement for individual functions). Hence, one usually tries to compute these functions approximately; the notion of approximation of course varies from application to application.

One of the basic function is to compute frequencies of individual elements. Formally, we are given a stream of elements of length m (think of m as the number of IP packets flowing through some router per day) from a universe $\mathcal{U}$ of size n (think if n as the total number of different IP packets possible). The task is to compute the frequencies of individual elements of $\mathcal{U}$ in the stream. We will now see the count min sketch algorithm for this problem. The high level idea is to incrementally build a small data structure called *sketch* such that, whenever a query comes for an element $a \in \mathcal{U}$, we can answer an estimate $\hat{f}(a)$ of the frequency of $f(a)$ with the guarantee that $\Pr\left[\hat{f}(a) \in [f(a), f(a) + \varepsilon m]\right] \geqslant 1 - \delta$.

We have a two dimensional integer array $\mathcal{A}$ with $r = \lceil \ln(1/\delta) \rceil$ rows and $c = \lceil e/\varepsilon \rceil$ columns. We choose r hash functions $h_i, i \in [r]$ uniformly randomly from a 2-universal hash family which maps $\mathcal{U}$ to $[c]$. The array $\mathcal{A}$ is initialized to 0; that is $\mathcal{A}[i][j] = 0$ for every $i \in [r], j \in [c]$. Whenever an element $a \in \mathcal{U}$ appears in the stream, we increment $\mathcal{A}[i][h_i(a)]$ by 1 for every $i \in [r]$. Whenever a query comes with an element $x \in \mathcal{U}$, we output $\hat{f}(x)$ to be $\min\{\mathcal{A}[i][h_i(x)] : i \in [r]\}$.

Lemma 8.1.1. *For every* $a \in \mathcal{U}$*, we have* $\Pr\left[\hat{f}(a) \in [f(a), f(a) + \varepsilon m]\right] \geqslant 1 - \delta$.

Proof. The lower bound $\hat{f}(a) \geqslant f(a)$ obviously holds with probability 1. For proving the upper bound, we define an indicator random variable $\mathcal{I}_i(b)$ for $b \in \mathcal{U} \setminus \{a\}$ for the event that $h_i(a) = h_i(b)$. Then we have the following.

$$\begin{aligned}
\mathcal{A}[i][h_i(a)] - f(a) &= \sum_{b \in \mathcal{U} \setminus \{a\}} \mathcal{I}_i(b) f(b) \\
\Rightarrow \mathbb{E}[\mathcal{A}[i][h_i(a)]] - f(a) &= \sum_{b \in \mathcal{U} \setminus \{a\}} f(b) \mathbb{E}[\mathcal{I}_i(b)] \\
&= \sum_{b \in \mathcal{U} \setminus \{a\}} f(b) \Pr[h_i(a) = h_i(b)] \\
&= \sum_{b \in \mathcal{U} \setminus \{a\}} f(b) \frac{\varepsilon}{e}
\end{aligned}$$

$$\leqslant \frac{\varepsilon m}{e}$$

Now by Markov's inequality, we have $\Pr[\mathcal{A}[i][h_i(a)] > f(a)+\varepsilon m] \leqslant \frac{1}{e}$. For $\hat{f}(a)$ to be more than $f(a)+\varepsilon m$, it should be the case that $\mathcal{A}[i][h_i(a)] > f(a) + \varepsilon m$ for every $i \in [r]$ which happens with probability at most $e^{-\ln(1/\delta)} = \delta$. □

The streaming model described above is called the "insertion only" model or vanilla model – elements are only inserted and never deleted. One could consider more sophisticated models. One such model is the "turnstile model" where both insertions and deletions are allowed. One can change the above count min sketch to what is called the count median sketch – when an element is deleted, we decrement corresponding entries in each row and instead of outputting minimum of $\{\mathcal{A}[i][h_i(x)] : i \in [r]\}$, we output median of $\{\mathcal{A}[i][h_i(x)] : i \in [r]\}$. One can perform similar analysis to show that the count median sketch algorithm (ε, δ) approximates the frequency of every element in $\mathcal{U}$ with same order of space and (update and query) time complexity.

8.1.4 Construction of 2-universal Hash Family: Using Finite Fields

Let us now see a construction of 2-universal hash family. Let $\mathcal{U}$ be a given universe and $[m]$ be a given co-domain of the hash functions. We choose a prime number p with $|\mathcal{U}| \leqslant p$. Then $\mathcal{U}$ can be associated with a subset of the field $\mathbb{F}_p$ ($\mathbb{F}_p$ is same as $\mathbb{Z}_p$ with addition and multiplication module p). Our hash family is defined by $\mathcal{H} = \{h_{a,b}(x) = (ax + b \pmod p) \pmod m : a, b \in \mathbb{F}_p, a \neq 0\}$. We now show that $\mathcal{H}$ forms a 2-universal hash family.

Lemma 8.1.2. *$\mathcal{H} = \{h_{a,b}(x) = (ax + b \pmod p) \pmod m : a, b \in \mathbb{F}_p, a \neq 0\}$ is a weakly 2-universal hash family.*

Proof. Let $x_1, x_2 \in \mathcal{U}, x_1 \neq x_2$, $y_1, y_2 \in [m], y_1', y_2' \in \mathbb{F}_p$ be arbitrary. We now have the following.

$$\Pr[ax_1 + b \equiv y_1' \pmod p, ax_2 + b \equiv y_2' \pmod p] = \Pr\left[a = \frac{y_1' - y_2'}{x_1 - x_2}, b = \frac{y_1'x_2 - y_2'x_1}{x_2 - x_1}\right]$$
$$= \begin{cases} \frac{1}{p(p-1)} & \text{if } y_1' \neq y_2' \\ 0 & \text{otherwise} \end{cases}$$

We now have the following.

$$\Pr[h_{a,b}(x_1) = y_1, h_{a,b}(x_2) = y_2] = \sum_{\substack{y_1' \in \mathbb{F}_p, y_1' \equiv y_1 \pmod m \\ y_2' \in \mathbb{F}_p, y_2' \equiv y_2 \pmod m}} \Pr[ax_1 + b \equiv y_1' \pmod p, ax_2 + b \equiv y_2' \pmod p]$$
$$= \sum_{\substack{y_1' \in \mathbb{F}_p, y_1' \equiv y_1 \pmod m \\ y_2' \in \mathbb{F}_p, y_2' \equiv y_2 \pmod m \\ y_1' \neq y_2'}} \Pr[ax_1 + b \equiv y_1' \pmod p, ax_2 + b \equiv y_2' \pmod p]$$
$$= \sum_{\substack{y_1' \in \mathbb{F}_p, y_1' \equiv y_1 \pmod m \\ y_2' \in \mathbb{F}_p, y_2' \equiv y_2 \pmod m \\ y_1' \neq y_2'}} \frac{1}{p(p-1)}$$
$$= \left\lceil \frac{p}{m} \right\rceil \left\lceil \frac{p}{m} - 1 \right\rceil \frac{1}{p(p-1)}$$

$$\leqslant \frac{p(p-1)}{m^2}\frac{1}{p(p-1)}$$
$$= \frac{1}{m^2}$$

The proof of the inequality $\left\lceil \frac{p}{m} \right\rceil \left\lceil \frac{p}{m} - 1 \right\rceil \leqslant \frac{p(p-1)}{m^2}$ is left as an exercise. □

The weaker version of 2-universality holds nevertheless (that is why our definition of 2-universality is sometimes called strong 2-universality) – for any different $x_1, x_2 \in \mathcal{U}$, we must have $\Pr[h(x_1) = h(x_2)] \leqslant 1/m$. Now one can check that our applications of 2-universal hash family still work with this weak 2-universal hash family.

8.1.5 Construction of k-universal Hash Family

The construction of 2-universal hash family can be straightforwardly generalized to k-universal hash family. Assuming same setting as 2-universal hash family, the k-universal hash family is defined by $\mathcal{H} = \left\{h_{a_0,a_1,\ldots,a_{k-1}}(x) = \left(a_0 + a_1 x + \cdots + a_{k-1}x^{k-1} \pmod p\right) \pmod m : a_0, a_1, \ldots, a_{k-1} \in \mathbb{F}_p\right\}$. The proof of k-universality is also along the same line. One needs elementary linear algebra (like Vandermonde matrix is non-singular) in the proof.

8.2 Cuckoo Hashing

The guarantees provided by perfect hashing is very good except it assumes that the set to be stored is static. One could hope to achieve similar guarantee for dynamic set. Cuckoo hashing is one such data structure which provides such kind of guarantees for dynamic sets. The search time for Cuckoo hashing is $\mathcal{O}(1)$ in the worst case and the expected insertion time is $\mathcal{O}(1)$. The idea of Cuckoo hashing is as follows.

It uses two hash functions h_1 and h_2 chosen uniformly random from a k-universal family of hash functions for some appropriate large k. It stores the data in an one dimensional table $\mathcal{T}$. It maintains the invariant that, for every element $x \in \mathcal{S}$ ($\mathcal{S}$ is the set of elements stored in the hash table), x is stored in either $\mathcal{T}[h_1(x)]$ or $\mathcal{T}[h_2(x)]$. So, to search for any element $x \in \mathcal{U}$, we only need to look at $\mathcal{T}[h_1(x)]$ and $\mathcal{T}[h_2(x)]$ and thus search takes $\mathcal{O}(1)$ time in the worst case. The insertion operation works as follows. To insert an element $x = x_1 \in \mathcal{U}$, if $\mathcal{T}[h_1(x_1)]$ or $\mathcal{T}[h_2(x_1)]$ is empty, we store x in $\mathcal{T}[h_1(x_1)]$ or $\mathcal{T}[h_2(x_1)]$. Otherwise, suppose x_2 is currently stored in $\mathcal{T}[h_1(x)]$. We store x_1 in $\mathcal{T}[h_1(x_1)]$. If $\mathcal{T}[h_2(x_2)]$ is empty, we store x_2 in $\mathcal{T}[h_2(x_2)]$. Otherwise, let $x_3 = \mathcal{T}[h_2(x_2)]$. We store x_2 in $\mathcal{T}[h_2(x_2)]$ and now we once again have another new value x_3 to insert in the table. The process continues until an empty cell has been found or it has run for some large number N iterations where m is the size of $\mathcal{T}$. If we could not find any empty cell for N iterations, then rehash – pick once again two hash functions, initialize the table to empty, and insert all the elements currently present in the table. The running time analysis is little involved. We refer interested readers to [PR04, Pag06].

8.3 Bloom Filter

Bloom filter is a highly space efficient data structure which have small probability of false positive – the data structure can sometime wrongly output that a given element $x \in \mathcal{U}$ belongs to the stored set $\mathcal{S}$ even if $x \notin \mathcal{S}$. However, the probability of such an event can be made arbitrarily small. The data structure is as follows.

We have a Boolean one dimensional array $\mathcal{T}$ of size m (the value of m will be decided later). The table $\mathcal{T}$ is initialized to 0. We choose k hash functions $h_1, \ldots, h_k$ uniformly randomly from a k-universal hash family $\mathcal{H}$ (again the value of k will be decided later) which maps $\mathcal{U}$ to $[m]$. To store an element $x \in \mathcal{U}$, we set $\mathcal{T}[h_i(x)] = 1$ for every $i \in [k]$. When a query for an element x is made, the data structure returns YES if $\mathcal{T}[h_i(x)] = 1$ for every $i \in [k]$ and returns NO otherwise. Obviously, for elements which are indeed inserted, the data structure can never make any mistake. Hence one only need to analyze the probability that an element $x \in \mathcal{U}$ which is never inserted and the data structure answers YES when queried for the element x. Let us postpone the analysis until we cover martingale.

Chapter 9

Sparsification Techniques

We now see various randomized sparsification techniques. On a very high level, sparsification techniques are tools that reduce the volume of data in a way so that we can still compute our desired functions of the data approximately.

9.1 Dimensionality Reduction: Johnson-Lindenstrauss Lemma

Our first such tool is Johnson-Lindenstrauss lemma which is a dimensionality reduction technique. The set up is as follows. Suppose we have n data points $x_1, x_2, \ldots, x_n \in \mathbb{R}^m$ in m dimensional Euclidean space and we wish to compute some function on these data points which depends on the pairwise distance of these points. Example of one such useful functions is classification. In typical computer science applications, data often resides in high dimension (m is high). Johnson-Lindenstrauss lemma (J-L lemma for short) states that there exists a linear transform $\mathcal{T} : \mathbb{R}^m \longrightarrow \mathbb{R}^d$ such that the images of $x_1, x_2, \ldots, x_n \in \mathbb{R}^m$ under $\mathcal{T}$ respects pairwise distances and d is around $\mathcal{O}(\log n)$. The linear transformation $\mathcal{T}$ in the J-L lemma is called the J-L transformation.

Lemma 9.1.1 (Johnson-Lindenstrauss Lemma)**.** *Let $x_1, x_2, \ldots, x_n \in \mathbb{R}^m$ any n points and $\varepsilon \in (0,1)$ be any. Then for $d = \mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$, there exists a linear transform $\mathcal{T} : \mathbb{R}^m \longrightarrow \mathbb{R}^d$ such that*

$$\|\mathcal{T}(x_i) - \mathcal{T}(x_j)\|_2 \in [(1-\varepsilon)\|x_i - x_j\|_2, (1+\varepsilon)\|x_i - x_j\|_2] \text{ for every } i, j \in [n]$$

$$\|\mathcal{T}(x_i)\|_2 \in [(1-\varepsilon)\|x_i\|_2, (1+\varepsilon)\|x_i\|_2] \text{ for every } i \in [n]$$

Moreover, a $\mathcal{T}$ can be found in polynomial time which satisfies the above inequalities with probability at least $1 - \frac{2}{n}$.

In the proof of Lemma 9.1.1 needs the following fact from elementary probability.

Claim 9.1.1. *Let $X_1, \ldots, X_n$ be n independent Normal random variables where $X_i \sim \mathcal{N}(0, \sigma_i^2)$. Then $X_1 + X_2 + \cdots + X_n \sim \mathcal{N}(0, \sum_{i=1}^n \sigma_i^2)$.*

Proof of Lemma 9.1.1: Using Claim 9.1.1, the rest of the proof is simple application of Chernoff bounds. Let $v \in \mathbb{R}^m$ be any vector of unit length. Let us first prove that there is a linear transformation $\mathcal{T}$ which

preserves the length of v with high probability. Let $r_i \in \mathbb{R}^m$ for $i \in [d]$ where every coordinate of r_i is sampled from $\mathcal{N}(0,1)$. The linear transformation $\mathcal{T}$ is defined as follows – for any $w \in \mathbb{R}^m$, the i-th component of $\mathcal{T}(w)$ is defined to be $\frac{r_i^t w}{\sqrt{d}}$. The length of $\mathcal{T}(w)$ can be written as follows.

$$d\|\mathcal{T}(w)\|_2^2 = \sum_{i=1}^{d} (r_i^t w)^2$$

And thus the expected squared length of $\mathcal{T}(v)$ is as follows.

$$d\mathbb{E}[\|\mathcal{T}(v)\|_2^2] = \sum_{i=1}^{d} \mathbb{E}[(r_i^t v)^2] = \sum_{i=1}^{d}\sum_{j=1}^{m} v_j^2 = d$$

Applying Chernoff bound on the random variable $Y = d\|\mathcal{T}(v)\|_2^2$. But since we have proved Chernoff bounds for sum of independent Bernolli random variables, let us follow the proof technique of the Chernoff bound. Let us define $X_i = r_i^t w$ for $i \in [d]$. Then each X_i follows $\mathcal{N}(0,1)$. Let us first prove the upper bound.

$$\begin{aligned}
\Pr\left[Y > (1+\varepsilon)^2 d\right] &= \Pr\left[e^{tY} > e^{t(1+\varepsilon)^2 d}\right] && \left[\text{for every } 0 < t < \frac{1}{2}\right] \\
&\leqslant e^{-t(1+\varepsilon)^2 d}\mathbb{E}\left[e^{tY}\right] \\
&= e^{-t(1+\varepsilon)^2 d}\mathbb{E}\left[e^{t\sum_{i=1}^{d} X_i^2}\right] \\
&= e^{-t(1+\varepsilon)^2 d}\prod_{i=1}^{d}\mathbb{E}\left[e^{tX_i^2}\right] \\
&= e^{-t(1+\varepsilon)^2 d}\prod_{i=1}^{d}\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{ty^2} e^{-\frac{y^2}{2}}\,dy && [X_i \sim \mathcal{N}(0,1)] \\
&= e^{-t(1+\varepsilon)^2 d}\prod_{i=1}^{d}\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-y^2\left(\frac{1}{2}-t\right)}\,dy \\
&= e^{-t(1+\varepsilon)^2 d}\prod_{i=1}^{d}\frac{1}{\sqrt{2\pi}\sqrt{1-2t}}\int_{-\infty}^{\infty} e^{-\frac{z^2}{2}}\,dz && [z = y\sqrt{1-2t}] \\
&= e^{-t(1+\varepsilon)^2 d}\prod_{i=1}^{d}\frac{1}{\sqrt{1-2t}} \\
&= e^{-\frac{d}{2}\left(2t(1+\varepsilon)^2 + \ln(1-2t)\right)} \\
&\leqslant e^{-\frac{d}{2}\left((1+\varepsilon)^2 - 1 - 2\ln(1+\varepsilon)\right)} && \left[t = \frac{1-\frac{1}{(1+\varepsilon)^2}}{2}\right] \\
&\leqslant e^{-\frac{d}{2}\left(\varepsilon^2 + 2\varepsilon - 2\varepsilon + \frac{\varepsilon^2}{2}\right)} && [\ln(1+x) \leqslant x - x^2/4] \\
&= e^{-\frac{3d\varepsilon^2}{4}} \\
&\leqslant \frac{1}{n^3} && \left[d = \frac{12\ln n}{3\varepsilon^2}\right]
\end{aligned}$$

Hence we have the following.

$$\Pr\left[||\mathcal{T}(v)||_2 > (1+\varepsilon)\right] \leqslant \frac{1}{n^3}$$

The technique can be analogously followed to derive a similar lower bound and thus by union bound we have the following for $d = \mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ for any $w \in \mathbb{R}^m$.

$$\Pr\left[||\mathcal{T}(v)||_2 \in [(1-\varepsilon),(1+\varepsilon)]\right] \geqslant 1 - \frac{2}{n^3}$$

We now apply the above result for $v = \frac{x_i}{||x_i||_2}$ for $i \in [n]$ and $v = \frac{x_i - x_j}{||x_i - x_j||_2}$ and by union bound, we have the result. □

9.1.1 Remarks on Johnson Lindenstrauss Lemma

It turns out that the guarantee on the dimension provided by the Johnson Lindenstrauss lemma is optimal because of the following result by Larsen and Nelson [LN17].

Theorem 9.1.1. *For $m, n \geqslant 2$ and $\frac{1}{(\min\{n,d\})^{0.499}} < \varepsilon < 1$, there exists a set $\mathcal{X}$ of n points in $\mathbb{R}^m$ such that there exists an embedding $f : \mathcal{X} \longrightarrow \mathbb{R}^d$ satisfying*

$$||\mathcal{T}(x_i) - \mathcal{T}(x_j)||_2 \in [(1-\varepsilon)\,||x_i - x_j||_2, (1+\varepsilon)\,||x_i - x_j||_2] \text{ for every } x_i, x_j \in \mathcal{X}$$

$$||\mathcal{T}(x_i)||_2 \in [(1-\varepsilon)\,||x_i||_2, (1+\varepsilon)\,||x_i||_2] \text{ for every } x_i \in \mathcal{X}$$

Then we have $d = \Omega\left(\frac{\log n}{\varepsilon^2}\right)$.

One can also try to extend Johnson-Lindenstrauss lemma to other distance metrics. It turns out that the Johnson-Lindenstrauss lemma strongly depends on the properties of the ℓ_2 norm. For example, it is known that, for ℓ_1 norm, any map which provides a guarantee like J-L must have $d = \Omega\left(n^{1/(1+\varepsilon)^2}\right)$ [BC05].

9.2 Sub-Gaussian Random Variables and Chernoff Bounds

We have seen that Chernoff type bounds hold for sum of Bernolli random variables and sum of Gaussian random variable. We can ask for what other kind random variables we can have Chernoff type bounds. It turns out that if a random variable is sub-Gaussian, then Chernoff type bounds hold. On a high level, Chernoff bound holds for those random variables which show Gausian type probability distribution. To formalize the notion of "Gaussian type", we need the concept of moment generating function.

Let $\mathcal{Z}$ be a random variable. Then the moment generating function $\mathcal{M} : \mathbb{R} \longrightarrow \mathbb{R}$ of $\mathcal{Z}$ is defined as $\mathcal{M}(s) = \mathbb{E}\left[\exp(sZ)\right]$. Let us compute the moment generating function of a standard Gaussian random variable $\mathcal{Z}$.

$$\mathcal{M}(s) = \mathbb{E}\left[\exp(sZ)\right] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{sz} e^{-z^2/2} dz = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-\frac{(z-s)^2}{2}} e^{\frac{s^2}{2}} dz = e^{\frac{s^2}{2}}$$

The sub-Gaussian random variables are defined as follows.

Definition 9.2.1 (Sub-Gaussian Random Variable)**.** *A random variable $\mathcal{X}$ is called sub-Gaussian with parameter σ^2 and denoted by $\mathcal{X} \sim subG(\sigma^2)$ if its moment generating function $\mathcal{M}$ exists and it satisfies $\mathcal{M}(s) \leqslant e^{\frac{\sigma^2 s^2}{2}}$ for every $s \in \mathbb{R}$.*

Let us see some examples of sub-Gausian random variables. Obviously a Gaussian random variable is sub-Gausian with parameter 1.

Example 9.2.1 (Rademacher random variable)**.** *A Rademacher random variable $\mathcal{X}$ takes values* 1 *and* -1 *with equal probability. The following shows that a Rademacher random variable is sub-Gaussian random variable with parameter* 1.

$$\mathbb{E}[e^{s\mathcal{X}}] = \frac{1}{2}\left(e^{s} + e^{-s}\right) = \sum_{i=0}^{\infty} \frac{s^{2i}}{(2i)!} \leqslant 1 + \sum_{i=1}^{\infty} \frac{s^{2i}}{2^i i!} \leqslant e^{s^2/2}$$

□

Before proceeding further, let us prove an easy but useful fact about Rademacher random variable. We say two random variables X and Y have the same distribution, denoted by $X \overset{D}{=} Y$, if $\Pr[X \in B] = \Pr[Y \in B]$ for every Borel set B. A random variable X is called symmetric if $\Pr[X \in B] = \Pr[X \in -B]$ for every Borel set B where $-B = \{-x : x \in B\}$.

Observation 9.2.1. *Let* X *be a symmetric random variable,* Y *a Rademacher random variable, and* X *and* Y *are independent. Then, we have* $X \overset{D}{=} YX$.

Proof. Let B be any Borel set. Then we have

$$\Pr[YX \in B] = \Pr[Y = 1]\Pr[X \in B] + \Pr[Y = -1]\Pr[X \in -B] = \Pr[X \in B].$$

□

Example 9.2.2 (Bounded random variables)**.** *Let $\mathcal{X}$ be a bounded random variable with upper and lower bounds being* a *and* b *respectively. Then following shows that $\mathcal{X} - \mathbb{E}[\mathcal{X}]$ is a sub-Gausian random variable with parameter $(b-a)^2$. Let $\mathcal{X}'$ be another random variable having same distribution as of $\mathcal{X}$. Also observe that, for a Rademacher random variable $\mathcal{Y}$, the distributions of $\mathcal{X} - \mathcal{X}'$ and $\mathcal{Y}(\mathcal{X} - \mathcal{X}')$ are the same.*

$$\begin{aligned}
\mathbb{E}_{\mathcal{X}}[e^{s(\mathcal{X} - \mathbb{E}_{\mathcal{X}'}[\mathcal{X}'])}] &\leqslant \mathbb{E}_{\mathcal{X},\mathcal{X}'}[e^{s(\mathcal{X}-\mathcal{X}')}] && [\textit{Jensen's inequality}] \\
&= \mathbb{E}_{\mathcal{X},\mathcal{X}',\mathcal{Y}}\left[e^{s\mathcal{Y}(\mathcal{X}-\mathcal{X}')}\right] && [\mathcal{Y} \textit{ is a Rademacher random variable}] \\
&= \mathbb{E}_{\mathcal{X},\mathcal{X}'}\left[\mathbb{E}_{\mathcal{Y}}\left[e^{s\mathcal{Y}(\mathcal{X}-\mathcal{X}')}\right]\right] \\
&\leqslant \mathbb{E}_{\mathcal{X},\mathcal{X}'}\left[\exp\left\{\frac{s^2(\mathcal{X}-\mathcal{X}')^2}{2}\right\}\right] && [\textit{Example 9.2.1}] \\
&\leqslant \mathbb{E}_{\mathcal{X},\mathcal{X}'}\left[\exp\left\{\frac{s^2(b-a)^2}{2}\right\}\right] \\
&= \exp\left\{\frac{s^2(b-a)^2}{2}\right\}
\end{aligned}$$

□

Let us now prove the main result of this section – Chernoff type bounds hold for sub-Gausian random variable and in particular to bounded random variables.

Proposition 9.2.1. *Let $\mathcal{X} \sim subG(\sigma^2)$. Then, we have $\mathbb{E}[\mathcal{X}] = 0$ and $Var(\mathcal{X}) = \mathbb{E}[\mathcal{X}^2] \leqslant \sigma^2$.*

Proof. Since $X \sim \text{subG}(\sigma^2)$, we have $\mathbb{E}[\exp(sX)] \leqslant e^{\frac{\sigma^2 s^2}{2}}$. And thus from Lebesgue's dominated convergence theorem[1], we have the following for every $s \in \mathbb{R}$.

$$\sum_{i=0}^{\infty} \frac{s^i}{i!}\mathbb{E}[X^i] = \mathbb{E}[\exp(sX)] \leqslant e^{\frac{\sigma^2 s^2}{2}} = \sum_{i=0}^{\infty} \frac{(\sigma s)^{2i}}{i!}$$

And thus we have

$$\sum_{i=1}^{\infty} \frac{s^i}{i!}\mathbb{E}[X^i] \leqslant \sum_{i=1}^{\infty} \frac{(\sigma s)^{2i}}{i!}$$

Dividing both sides of the above inequality by s and then putting $s = 0$, we have $\mathbb{E}[X] \leqslant 0$. Replacing s with $-s$, dividing both sides of the above inequality by s, and putting $s = 0$, we get $\mathbb{E}[X] \geqslant 0$. Hence, we have

$$\sum_{i=2}^{\infty} \frac{s^i}{i!}\mathbb{E}[X^i] \leqslant \sum_{i=1}^{\infty} \frac{(\sigma s)^{2i}}{i!}$$

Dividing both sides of the above inequality by s^2 and then putting $s = 0$, we have $\mathbb{E}[X^2] \leqslant \sigma^2$. Since we already have $\mathbb{E}[X] = 0$, we now have $\text{Var}(X) \leqslant \sigma^2$. □

Just as sum of Gaussian random variables is another Gaussian random variable, the same holds for sub-Gaussian random variables.

Proposition 9.2.2. *Let $X_i \sim subG\left(\sigma_i^2\right)$ for $i \in [n]$. Then $\sum_{i=1}^n X_i \sim subG\left(\sum_{i=1}^n \sigma_i^2\right)$.*

Proof. It is enough to prove the result for $n = 2$. We have the following.

$$\mathbb{E}[\exp(sX_1 + sX_2)] = \mathbb{E}[\exp(sX_1)]\,\mathbb{E}[\exp(sX_2)] \leqslant e^{\frac{(\sigma_1^2+\sigma_2^2)s^2}{2}}$$

□

Following exact same proof of the Chernoff bounds (for Gaussian and Bernolli random variables), one could prove the following.

Theorem 9.2.1. *[Hoeffding Bound] If $X \sim subG(\sigma^2)$, then for any $t \geqslant 0$, we have the following.*

$$\Pr[X \geqslant t] \leqslant \exp\left\{-\frac{t^2}{2\sigma^2}\right\}$$

Hence, if X_i be random variables with expectation μ_i and $X_i - \mu_i \sim subG(\sigma_i^2)$ for $i \in [n]$, then we have the following for any $t \geqslant 0$.

$$\Pr\left[\sum_{i=1}^{n}(X_i - \mu_i) \geqslant t\right] \leqslant \exp\left\{-\frac{t^2}{2\sum_{i=1}^n \sigma_i^2}\right\}$$

Proof. For any positive real number s, we have the following.

$$\Pr[X \geqslant t] = \Pr[e^{sX} \geqslant e^{st}] \leqslant \exp\{-st\}\mathbb{E}[e^{sX}] \leqslant \exp\left\{\frac{\sigma^2 s^2}{2} - st\right\} \leqslant \exp\left\{-\frac{t^2}{2\sigma^2}\right\}$$

□

[1]The version used here is the following: Let $X_i, i \in \mathbb{N}$ be a sequence of random variables in some underlying probability space $(\Omega, \mathcal{F}, \mathcal{P})$ having finite expectation. Suppose there exists another random variable Y such that $X_i(\omega) \leqslant Y(\omega)$ for every $\omega \in \Omega$ for every $i \in \mathbb{N}$ and the expectation of Y is finite. Then $\mathbb{E}[\lim_{i\to\infty} X_i] = \lim_{i\to\infty} \mathbb{E}[X_i]$.

9.3 Probabilistic Tree Embedding

Any undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ induces a metric $d_{\mathcal{G}}$ on $\mathcal{V}$: for $u, v \in \mathcal{V}$, $d_{\mathcal{G}}(u, v)$ is the distance between u and v in $\mathcal{G}$. This metric is called a *graph metric*. A simpler sub-class of graph metric is *tree metric* where the graph $\mathcal{G}$ involved is actually a tree $\mathcal{T}$. A tree metric $d_{\mathcal{T}}$ is said to approximate a graph metric $d_{\mathcal{G}}$ within a factor α if $\mathcal{V}[\mathcal{G}] \subseteq \mathcal{V}[\mathcal{T}]$ and the following holds.

$$d_{\mathcal{G}}(u, v) \leqslant d_{\mathcal{T}}(u, v) \leqslant \alpha d_{\mathcal{G}}(u, v), \forall u, v \in \mathcal{V}[\mathcal{G}]$$

Given a graph $\mathcal{G}$ can we find a tree $\mathcal{T}$ so that $d_{\mathcal{T}}$ provides a good approximation for $d_{\mathcal{G}}$? If $\mathcal{G}$ is a cycle on n vertices, then it is known that every tree $\mathcal{T}$ which approximates $\mathcal{G}$ have an approximation ratio of at least $\frac{n}{3} - 1$ [RR98]. To tackle the above lower bound, Bartal introduced the notion of *probabilistic tree embedding* [Bar96]. A probabilistic tree embedding is a collection $\mathcal{T}$ of trees $\{T_i : i \in [k]\}$ along with a probability distribution p on $\mathcal{T}$. A probabilistic tree embedding $(\{T_i : i \in [k]\}, p)$ is said to approximate a graph metric $d_{\mathcal{G}}$ within a factor α if $\mathcal{V}[\mathcal{G}] \subseteq \mathcal{V}[T_i]$ for every $i \in [k]$ and the following holds.

$$d_{\mathcal{G}}(u, v) \leqslant d_T(u, v), \forall u, v \in \mathcal{V}[\mathcal{G}], T \in \mathcal{T}$$

$$\mathbb{E}_{T \sim \mathcal{T}}[d_T(u, v)] \leqslant \alpha d_{\mathcal{G}}(u, v), \forall u, v \in \mathcal{V}[\mathcal{G}]$$

Bartal showed that, for every graph $\mathcal{G}$, there exists a probabilistic tree embedding which approximates $d_{\mathcal{G}}$ within a factor of $\mathcal{O}(\log n)$ [Bar96] which we see next.

Given a graph $\mathcal{G}$, we construct its corresponding tree $\mathcal{T}$ by what is known as a *hierarchical cut decomposition*. Let Δ be the smallest power of 2 greater than the diameter of the graph $\mathcal{G}$. A hierarchical cut decomposition is a rooted tree with height $\lg \Delta$ (that is $\lg \Delta + 1$ levels); the level of the leaves is 0 and the level of the root is $\lg \Delta$. Every node in the tree $\mathcal{T}$ corresponds to some subset $\mathcal{S}$ of $\mathcal{V}[\mathcal{G}]$. The root node corresponds to $\mathcal{V}[\mathcal{G}]$ and leaf nodes correspond to singleton sets. The children of any internal node $\mathcal{S}$ is a partition of $\mathcal{S}$ with the property that the diameter of the sub-graph induced on each children node is at most half the diameter of the sub-graph induced on $\mathcal{S}$. The weight of the edges between level i and level $i+1$ is 2^i. Finally, we identify the leaf nodes of the tree by the vertices they contain. We are done with our description of the hierarchical cut decomposition if we specify the children of every node. But before that, let us make the following easy observation.

Observation 9.3.1. *Let $\mathcal{T}$ be hierarchical cut decomposition of a graph $\mathcal{G}$. Then, every vertex $u \in \mathcal{V}[\mathcal{G}]$ corresponds to one leaf in $\mathcal{T}$ and, for every $u, v \in \mathcal{V}[\mathcal{G}]$, we have $d_{\mathcal{T}}(u, v) \geqslant d_{\mathcal{G}}(u, v)$. Further, if the least common ancestor of u and v is at level i, then we have $d_{\mathcal{T}}(u, v) \leqslant 2^{i+2}$.*

Let us now describe children nodes of any internal node $\mathcal{S}$. We pick $r_0 \in [1/2, 1)$ uniformly at random and a permutation π on $\mathcal{V}[\mathcal{G}]$ uniformly at random from the set of all permutations. Define $r_i = 2^i r_0$ for $i \in [\lg \Delta]$. We observe that r_i is distributed uniformly in $[2^{i-1}, 2^i)$. Suppose the level of $\mathcal{S}$ is i. We define the first children of $\mathcal{S}$ to be the set of vertices in $B(\pi(1), r_{i-1}) \cap \mathcal{S}$ and delete the $B(\pi(1), r_{i-1})$ from $\mathcal{S}$. Iteratively, for $i = 2, 3, \ldots, n$, if $B(\pi(i), r_{i-1}) \cap \mathcal{S}$ is non-empty, define $B(\pi(i), r_{i-1}) \cap \mathcal{S}$ to be another child of $\mathcal{S}$ and remove $B(\pi(i), r_{i-1}) \cap \mathcal{S}$ from $\mathcal{S}$.[2] We now prove the main theorem of this section.

[2] We observe that this randomized construction of the hierarchical cut tree is equivalent to defining a family of trees along with a probability distribution on it.

Theorem 9.3.1. *For any graph $\mathcal{G}$, let $\mathcal{T}$ be a hierarchical cut tree output by the above randomized algorithm. Then for every $u, v \in \mathcal{V}[\mathcal{G}]$, we have the following.*

$$d_{\mathcal{G}}(u,v) \leqslant d_{\mathcal{T}}(u,v), \mathbb{E}[d_{\mathcal{T}}(u,v)] \leqslant \mathcal{O}(\log n) d_{\mathcal{G}}(u,v)$$

Proof. We already have $d_{\mathcal{G}}(u,v) \leqslant d_{\mathcal{T}}(u,v)$ due to Observation 9.3.1. So, we only need to prove the other inequality. Let $u, v \in \mathcal{V}[\mathcal{G}]$ be any two vertices. Then, we also have $d_{\mathcal{T}}(u,v) \leqslant 2^{i+3}$ if the least common ancestor of u and v is at level $i+1$; u and v belong to different nodes at level i. This can only happen if there is a vertex w so that exactly one of u and v belongs to the ball centered at w. We say that w *settles* the pair u and v at level i if w is the first vertex in the random permutation π such that at least one of u and v belongs to $B(w, r_i)$. We say that w *cuts* the pair u and v at level i if exactly one of u and v belongs to $B(w, r_i)$ (note that w need not be the first vertex in the permutation π in case of cutting unlike settling). Let $X_{i,w}$ be the event that w cuts the pair u and v at level i and $S_{i,w}$ be the event that w settles the pair u and v at level i. Then, we have the following.

$$\begin{aligned}
d_{\mathcal{T}}(u,v) &\leqslant \max_{i=0,1,\ldots,\lg\Delta-1} \mathbb{I}(\exists w \in \mathcal{V}[\mathcal{G}] : X_{i,w} \wedge S_{i,w}) 2^{i+3} \\
&\leqslant \sum_{w \in \mathcal{V}[\mathcal{G}]} \sum_{i=0}^{\lg\Delta-1} \mathbb{I}(X_{i,w} \wedge S_{i,w}) 2^{i+3} \\
\Rightarrow \mathbb{E}[d_{\mathcal{T}}(u,v)] &\leqslant \sum_{w \in \mathcal{V}[\mathcal{G}]} \sum_{i=0}^{\lg\Delta-1} \Pr[X_{i,w} \wedge S_{i,w}] 2^{i+3} \\
&= \sum_{w \in \mathcal{V}[\mathcal{G}]} \sum_{i=0}^{\lg\Delta-1} \Pr[X_{i,w}] \Pr[S_{i,w} | X_{i,w}] 2^{i+3} \\
&\leqslant \sum_{w \in \mathcal{V}[\mathcal{G}]} \left(\max_{i=0,\ldots,\lg\Delta-1} \Pr[S_{i,w} | X_{i,w}] \right) \left(\sum_{i=0}^{\lg\Delta-1} \Pr[X_{i,w}] 2^{i+3} \right) \\
&\leqslant 16 d_{\mathcal{G}}(u,v) \left(\sum_{w \in \mathcal{V}[\mathcal{G}]} \max_{i=0,\ldots,\lg\Delta-1} \Pr[S_{i,w} | X_{i,w}] \right) && \text{[From Claim 9.3.1]} \\
&= \mathcal{O}(\log n) d_{\mathcal{G}}(u,v) && \text{[From Claim 9.3.2]}
\end{aligned}$$

We now prove Claims 9.3.1 and 9.3.2.

Claim 9.3.1. $\sum_{i=0}^{\lg\Delta-1} \Pr[X_{i,w}] 2^{i+3} \leqslant 16 d_{\mathcal{G}}(u,v)$.

Proof. Without loss of generality, we may assume that $d_{\mathcal{G}}(u,w) \leqslant d_{\mathcal{G}}(v,w)$. Then, the probability that w cuts $\{u,v\}$ at the i-th level is the probability that $u \in B(w, r_i)$ and $v \notin B(w, r_i)$ which is same as $r_i \in [d_{\mathcal{G}}(u,w), d_{\mathcal{G}}(v,w))$. Thus, we have the following.

$$\Pr[X_{i,w}] = \frac{|[2^{i-1}, 2^i) \cap [d_{\mathcal{G}}(u,w), d_{\mathcal{G}}(v,w))|}{|[2^{i-1}, 2^i)|} = \frac{|[2^{i-1}, 2^i) \cap [d_{\mathcal{G}}(u,w), d_{\mathcal{G}}(v,w))|}{2^{i-1}}$$

Summing over all i, we have the following.

$$\sum_{i=0}^{\lg\Delta-1} \Pr[X_{i,w}] 2^{i+3} = 16 \sum_{i=0}^{\lg\Delta-1} |[2^{i-1}, 2^i) \cap [d_{\mathcal{G}}(u,w), d_{\mathcal{G}}(v,w))| = 16(d_{\mathcal{G}}(u,w) - d_{\mathcal{G}}(v,w)) \leqslant 16 d_{\mathcal{G}}(u,v)$$

□

Claim 9.3.2. $\sum_{w \in \mathcal{V}[\mathcal{G}]} \left(\max_{i=0,\ldots,\lg \Delta - 1} \Pr[S_{i,w} | X_{i,w}]\right) = \mathcal{O}(\log n)$.

Proof. We observe that, if $X_{i,w}$ has happened, then at least one of u and v belong to $B(w, r_i)$. Now for $S_{i,w}$ to happen, every vertex $z \in \mathcal{V}[\mathcal{G}]$ that is closer to u and v than w (that is $\min\{d_{\mathcal{G}}(z,u), d_{\mathcal{G}}(z,v)\} \leqslant \min\{d_{\mathcal{G}}(w,u), d_{\mathcal{G}}(w,v)\}$) must appear after w in the random permutation π. Hence, among all vertices, if w is the j-th closest to u and v, then $\Pr[S_{i,w} | X_{i,w}] \leqslant 1/j$. Since j is independent of i, we have $\sum_{w \in \mathcal{V}[\mathcal{G}]} \left(\max_{i=0,\ldots,\lg \Delta - 1} \Pr[S_{i,w} | X_{i,w}]\right) \leqslant \sum_{j=1}^{n} 1/j \leqslant \ln n$. □

□

The trees we get in the underlying probability space in Theorem 9.3.1 has some new vertices other than $\mathcal{V}[\mathcal{G}]$. Often, in applications of tree embedding, we require to have the trees in the underlying probability space to be exactly on the set $\mathcal{V}[\mathcal{G}]$ of vertices. We show next that one can construct a tree embedding on $\mathcal{V}[\mathcal{G}]$ itself from a hierarchical cut tree obtained in Theorem 9.3.1.

Lemma 9.3.1. *For any hierarchical cut tree $\mathcal{T}$ defined on $\mathcal{V}'$, we can construct another tree $\mathcal{T}'$ on the set $\mathcal{V}$ of leaves of $\mathcal{T}$ such that, for every $u, v \in \mathcal{V}$, we have $d_{\mathcal{T}}(u,v) \leqslant d_{\mathcal{T}'}(u,v) \leqslant 4 d_{\mathcal{T}}(u,v)$.*

Proof. Initialize $\mathcal{T}'$ to be $\mathcal{T}$. Let $v \in \mathcal{V}$ be any vertex so that its parent w in $\mathcal{T}'$ is not a leaf node in $\mathcal{T}$; such a vertex always exists unless $\mathcal{T}'$ is a tree on $\mathcal{V}$. We contract the edge $\{v, w\}$ in $\mathcal{T}'$ and identify the new vertex in $\mathcal{T}'$ as v. We repeat this process until the vertex set of $\mathcal{T}'$ is $\mathcal{V}$. Finally we multiply the weight of every edge in $\mathcal{T}'$ by 4. We now claim that $\mathcal{T}'$ satisfies the guarantee required in the statement.

Let $u, v \in \mathcal{V}$ be any two vertices. Since contracting edges can only decrease the distance between u and v, we have $d_{\mathcal{T}'}(u,v) \leqslant 4 d_{\mathcal{T}}(u,v)$. Suppose the lowest common ancestor of u and v is at level i. Then, we have $d_{\mathcal{T}}(u,v) = 2\sum_{j=0}^{i} 2^j = 2^{i+2} - 4$. A key observation is that, the contraction process contracts at most one child of any internal node of $\mathcal{T}$. Hence, we have $d_{\mathcal{T}'}(u,v) \geqslant 4 2^i \geqslant 2^{i+2} - 4 = d_{\mathcal{T}}(u,v)$. □

Hence, the main Theorem in this section can be further strengthened as the following.

Theorem 9.3.2. *For any graph $\mathcal{G}$, there is a polynomial time randomized algorithm to construct a tree $\mathcal{T}$ on $\mathcal{V}[\mathcal{G}]$ such that, for every $u, v \in \mathcal{V}[\mathcal{G}]$, we have the following.*

$$d_{\mathcal{G}}(u,v) \leqslant d_{\mathcal{T}}(u,v), \mathbb{E}[d_{\mathcal{T}}(u,v)] \leqslant \mathcal{O}(\log n) d_{\mathcal{G}}(u,v)$$

9.3.1 Application: Buy-at-Bulk Network Design

Let us now see an application of probabilistic tree embedding. We consider the *buy-at-bulk network design* problem. In this problem, the input is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each edge $e \in \mathcal{E}$ has a length $\ell_e \in \mathbb{R}_+$. There are k source-destination pairs (s_i, t_i) with demand d_i for $i \in [k]$. We need to send d_i units of flow from s_i to t_i. Every edge has infinite capacity. However, if we wish to send c_e units of flow through the edge $e \in \mathcal{E}$, then we need to pay $f(c_e)\ell_e$ rupees. The function f is sub-additive: $f(0) = 0$ and $f(x+y) \leqslant f(x) + f(y)$ for every $x, y \in \mathbb{R}_+$. A solution is a set of paths $p_i, i \in [k]$ where p_i is a path from s_i to t_i. The cost of a solution is $\sum_{e \in \mathcal{E}} f(c_e)\ell_e$ where c_e is the total amount of flow passing through the edge e. The task is to find a minimum cost solution.

We observe that the problem is trivial for trees because the solution is unique. We now design a $\mathcal{O}(\log n)$ factor randomized approximation algorithm for this problem. Using Theorem 9.3.2, we compute a tree $\mathcal{T}$. Let $p_{i,\mathcal{T}}$ be the unique paths from s_i to t_i in $\mathcal{T}$ for $i \in [k]$. For $u, v \in \mathcal{V}[\mathcal{G}]$, let $p_{u,v}$ be a shortest path between

u and v. We construct a path $p_{i,\mathcal{G}}$ between s_i and t_i in $\mathcal{G}$ by concatenating $p_{u,v}$ for every edge $\{u,v\} \in p_{i,\mathcal{T}}$. We now prove the approximation factor of the algorithm.

The analysis has three steps. We need to map solutions of graph and tree back and forth. Let's first discuss how we do that. Let $p_{\{x,y\},\mathcal{T}}$ and $p_{\{x,y\},\mathcal{G}}$ denote a shortest path between $x \in \mathcal{V}[\mathcal{G}]$ and $y \in \mathcal{V}[\mathcal{G}]$ respectively in $\mathcal{T}$ and $\mathcal{G}$. A solution can be viewed as buying capacities to every edge. Let $c_{e,\mathcal{G}}, e \in \mathcal{E}[\mathcal{G}]$ be a solution of $\mathcal{G}$. Then the corresponding solution $c_{e,\mathcal{T}}, e \in \mathcal{E}[\mathcal{T}]$ is obtained as follows – for every edge $e \in \mathcal{E}[\mathcal{T}]$, we define $c_{e,\mathcal{T}} = \sum_{\{x,y\} \in \mathcal{E}[\mathcal{G}]: e \in p_{\{x,y\},\mathcal{T}}} c_{\{x,y\},\mathcal{G}}$. Similarly, given a solution $c_{e,\mathcal{T}}, e \in \mathcal{E}[\mathcal{T}]$ of $\mathcal{T}$, the corresponding solution in $\mathcal{G}$ is obtained as follows – for every edge $e \in \mathcal{E}[\mathcal{G}]$, we define $c_{e,\mathcal{G}} = \sum_{\{x,y\} \in \mathcal{E}[\mathcal{T}]: e \in p_{\{x,y\},\mathcal{G}}} c_{\{x,y\},\mathcal{T}}$.

Step 1: The optimal solution of tree when mapped to the graph does not increase the cost Let ALG be the cost of the solution that the algorithm outputs. The following shows that ALG is at most $\sum_{e \in \mathcal{E}[\mathcal{T}]} \ell_{e,\mathcal{T}} f(c_{e,\mathcal{T}})$.

$$\begin{aligned}
\text{ALG} &= \sum_{e \in \mathcal{E}[\mathcal{G}]} \ell_{e,\mathcal{G}} f(c_{e,\mathcal{G}}) \\
&= \sum_{e \in \mathcal{E}[\mathcal{G}]} \ell_{e,\mathcal{G}} f\left(\sum_{\{x,y\} \in \mathcal{E}[\mathcal{T}]: e \in p_{x,y}} c_{\{x,y\},\mathcal{T}} \right) \\
&\leqslant \sum_{e \in \mathcal{E}[\mathcal{G}]} \ell_{e,\mathcal{G}} \sum_{\{x,y\} \in \mathcal{E}[\mathcal{T}]: e \in p_{x,y}} f\left(c_{\{x,y\},\mathcal{T}}\right) \\
&= \sum_{\{x,y\} \in \mathcal{E}[\mathcal{T}]} f\left(c_{\{x,y\},\mathcal{T}}\right) \sum_{e \in p_{x,y}} \ell_{e,\mathcal{G}} \\
&= \sum_{\{x,y\} \in \mathcal{E}[\mathcal{T}]} f\left(c_{\{x,y\},\mathcal{T}}\right) d_{\mathcal{G}}(x,y) \\
&\leqslant \sum_{\{x,y\} \in \mathcal{E}[\mathcal{T}]} f\left(c_{\{x,y\},\mathcal{T}}\right) \ell_{\{x,y\},\mathcal{T}}
\end{aligned}$$

Hence, step 1 shows that it is enough to prove that the optimal solution in the tree has "low cost."

Step 2: The cost of optimal solution of tree is at most the cost of the optimal solution of the graph mapped to tree Obvious.

Hence, steps 1 and 2 show that it is enough to prove that the optimal solution of the graph when mapped to the tree has "low cost" which step 3 proves.

Step 3: The expected cost of the optimal solution of the graph mapped to tree is at most $O(\log n)$ times the cost of optimal solution of the graph Let $c^*_{e,\mathcal{G}}$ be any optimal solution of $\mathcal{G}$; that is $\text{OPT} = \sum_{e \in \mathcal{E}[\mathcal{G}]} f(c^*_{e,\mathcal{G}}) \ell_{e,\mathcal{G}}$. Then we bound the expected cost of the optimal solution of $\mathcal{G}$ when mapped to $\mathcal{T}$ as follows.

$$\begin{aligned}
\mathbb{E}\left[\sum_{e \in \mathcal{E}[\mathcal{T}]} f\left(\sum_{\{x,y\} \in \mathcal{E}[\mathcal{G}]: e \in p_{\{x,y\},\mathcal{T}}} c^*_{\{x,y\},\mathcal{G}} \right) \ell_{e,\mathcal{T}} \right] &\leqslant \mathbb{E}\left[\sum_{e \in \mathcal{E}[\mathcal{T}]} \sum_{\{x,y\} \in \mathcal{E}[\mathcal{G}]: e \in p_{\{x,y\},\mathcal{T}}} f\left(c^*_{\{x,y\},\mathcal{G}}\right) \ell_{e,\mathcal{T}} \right] \\
&= \mathbb{E}\left[\sum_{\{x,y\} \in \mathcal{E}[\mathcal{G}]} f\left(c^*_{\{x,y\},\mathcal{G}}\right) \sum_{e \in p_{\{x,y\},\mathcal{T}}} \ell_{e,\mathcal{T}} \right]
\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}\left[\sum_{\{x,y\}\in\mathcal{E}[\mathcal{G}]} f\left(c^*_{\{x,y\},\mathcal{G}}\right) d_{\mathcal{T}}(x,y)\right] \\
&= \sum_{\{x,y\}\in\mathcal{E}[\mathcal{G}]} f\left(c^*_{\{x,y\},\mathcal{G}}\right) \mathbb{E}\left[d_{\mathcal{T}}(x,y)\right] \\
&\leqslant \mathcal{O}(\log n) \sum_{\{x,y\}\in\mathcal{E}[\mathcal{G}]} f\left(c^*_{\{x,y\},\mathcal{G}}\right) d_{\mathcal{G}}(x,y) \\
&= \mathcal{O}(\log n)\mathrm{OPT}
\end{aligned}
$$

From steps 1, 2, and 3, we have $\mathbb{E}[\mathrm{ALG}] \leqslant \mathcal{O}(\log n)\mathrm{OPT}$. □

Chapter 10

Martingales

We have seen that the basic Chernoff bound can be extended to a class of random variables known as sub-Gaussian random variables. That is, if we have k independent random variables $X_i, i \in [k]$ where X_i is sub-Gaussian with parameter σ_i for $i \in [k]$, then Chernoff type bounds hold for $\sum_{i=1}^{k} X_i$. We now generalize Chernoff type bounds to other direction – do we need to have complete independence to have Chernoff type concentration? It turns out that Chernoff type concentration holds for $\sum_{i=1}^{k} X_i$ if X_i $i \in [k]$ forms a martingale[1]. While arguing about sum of k random variables, we generally assume that k is fixed (not random). We will also see what can we say if k itself is random.

10.1 Definition

Intuitively, martingale models fair games in the following sense. Suppose we play a gambling game some n number of times. Our total profit after playing i games is given by a random variable Z_i for $i \in \{0, 1, \ldots, n\}$. Hence, by definition, we have $Z_0 = 0$. Suppose X_i is the outcome of the i-th game. Intuitively speaking, a game may be termed fair if, in expectation, we neither gain nor lose. That is, $\mathbb{E}[Z_1|X_0] = Z_0, \mathbb{E}[Z_2|X_0, X_1] = Z_1, \ldots, \mathbb{E}[Z_n|X_0, X_1, \ldots, X_{n-1}] = Z_{n-1}$; hence after playing i rounds of the game, expected profit after playing one more round is the same as the current profit. When such a thing happens, we say that $Z_i, i \in \{0, 1, \ldots, n\}$ forms a martingale with respect to $X_i, i \in \{0, 1, \ldots, n\}$.

Definition 10.1.1 (Martingale)**.** *A sequence $Z_i, i \in \mathbb{N}$ of random variables is called a martingale with respect to another sequence $X_i, i \in \mathbb{N}$ of random variables if the following holds for every $n \in \mathbb{N}$.*

(i) Z_n *is a function of* $X_i, 0 \leqslant i \leqslant n$

(ii) $\mathbb{E}[|Z_n|] < \infty$

(iii) $\mathbb{E}[Z_{n+1}|X_0, X_1, \ldots, X_n] = Z_n$

A canonical example of martingale is a gambler who plays a sequence of fair games. Let $X_i, i \in \mathbb{N}$, be his profit from the i-th game (X_i is negative in case of loss). If each game is fair, then we have $\mathbb{E}[X_i] = 0$ for every $i \in \mathbb{N}$. The random variable Z_i denotes the gambler's total profit after playing i games. That is, we

[1] Needless to say that generalizing Chernoff type bounds is not the only motivation for defining and studying martingale. Martingale naturally arises in many real worlds applications and gambling games.

have $Z_i = X_0 + X_1 + \cdots + X_i$. If each $X_i, i \in \mathbb{N}$, is bounded, then $Z_i, i \in \mathbb{N}$, is also bounded. We now have the following.

$$\mathbb{E}[Z_{i+1}|X_0, X_1, \ldots, X_i] = \mathbb{E}[X_1 + X_2 + \cdots + X_i + X_{i+1}|X_0, X_1, \ldots, X_i] = Z_i + \mathbb{E}[X_{i+1}] = Z_i$$

Hence $\{Z_i\}_{i\in\mathbb{N}}$ forms a martingale with respect to $\{X_i\}_{i\in\mathbb{N}}$.

10.2 Doob Martingale

One of the most useful martingales is Doob martingale. Let $X_0, X_1, \ldots, X_n$ be a sequence of random variables and Y be a function of $X_0, X_1, \ldots, X_n$. Suppose we have $\mathbb{E}|Y| < \infty$. An important example of the above setting is the following. The random variables $X_0, X_1, \ldots, X_n$ corresponds to the pairs of vertices denoting whether there exists an edge between them or not and Y is the size of minimum vertex cover or the size of maximum clique, etc. We can naturally define a martingale in this setting as follows. For $i \in \{0, 1, \ldots, n\}$, we define a random variable $Z_i = \mathbb{E}[Y|X_0, X_1, \ldots, X_i]$. Obviously, Z_i is a function of $X_0, X_1, \ldots, X_i$ for every $i \in \{0, 1, \ldots, n\}$. Also we have $\mathbb{E}[|Z_i|] < \infty$ for every $i \in \{0, 1, \ldots, n\}$ since $\mathbb{E}[|Z_n|] < \infty$. We now prove that $Z_i, i \in \{0, 1, \ldots, n\}$ satisfy the third property.

$$\begin{aligned}
\mathbb{E}[Z_{i+1}|X_0, X_1, \ldots, X_i] &= \mathbb{E}[\mathbb{E}[Y|X_0, X_1, \ldots, X_i, X_{i+1}]|X_0, X_1, \ldots, X_i] \\
&= \mathbb{E}[Y|X_0, X_1, \ldots, X_i] && [\mathbb{E}[\mathbb{E}[A|B, C]|B] = \mathbb{E}[A|B]] \\
&= Z_i
\end{aligned}$$

The Doob martingale corresponding to the setting where $X_0, X_1, \ldots, X_n$ corresponds to the pairs of vertices denoting whether there exists an edge between them or not is called the *edge exposure martingale*. Similarly, one can think of a setting where $X_0, X_1, \ldots, X_n$ corresponds to the vertices where X_i is the random variable denoting the neighbors of the vertex i in $\{0, 1, \ldots, i\}$. Then the corresponding Doob martingale is called the *vertex exposure martingale*.

10.3 Stopping Time

In our canonical example of a gambler, if the gambler decides to play n rounds of a fair play, then one can show that $\mathbb{E}[Z_n] = \mathbb{E}[Z_0]$ as

$$\mathbb{E}[Z_n] = \mathbb{E}[\mathbb{E}[Z_n|X_0, X_1, \ldots, X_{n-1}]] = \mathbb{E}[Z_{n-1}] = \cdots = \mathbb{E}[Z_0].$$

However, gambler can "manipulate" such a fair sequence of games and guarantee profit – he gambles $\$2^i$ in the i-th game and leaves whenever he wins first. Observe that, when he wins first, he compensates all his previous losses and makes a positive profit![2] The famous *martingale stopping time theorem* proves that, intuitively speaking, the above strategy is the only unfair strategy. *A stopping time $\mathcal{T}$ is a random variable which takes only non-negative integer numbers and the event $\mathcal{T} = n$ depends only on the random variables $Z_i, i \in [n]$ where $Z_i, i \in \mathbb{N}$ is the martingale under consideration.* We now state the martingale stopping time theorem without proving it.

[2]Observe that this strategy ensures positive profit even for unfair games!

Theorem 10.3.1 (Martingale Stopping Time Theorem). *Let $Z_i, i \in \mathbb{N}$ be a martingale with respect to $X_i, i \in \mathbb{N}$ and $\mathcal{T}$ be a stopping time for $X_i, i \in \mathbb{N}$. Then we have $\mathbb{E}[Z_{\mathcal{T}}] = \mathbb{E}[Z_0]$ if any one of the following holds.*

(i) there is a constant c such that $|Z_i| < c$ for every $i \in \mathbb{N}$.

(ii) $\mathcal{T}$ is bounded.

(iii) $\mathbb{E}[\mathcal{T}] < \infty$ and there is a constant d such that $\mathbb{E}\left[|Z_{i+1} - Z_i| | X_0, X_1, \ldots, X_i\right] < d$ for every $i \in \mathbb{N}$.

Let us see some applications of the martingale stopping time theorem. Suppose, a gambler starts playing with $\$\ell_1$ and he continues to play until he either wins $\$\ell_2$ or loses all his money. Then, what is the probability that he loses all his money? To find this probability, consider the stopping time $\mathcal{T}$ of the canonical gambler martingale as the first time the total profit of the gambler hits either ℓ_2 or $-\ell_1$. Then, by martingale stopping time theorem, we have $\mathbb{E}[Z_{\mathcal{T}}] = \mathbb{E}[Z_0] = 0$. On the other hand, we have $\mathbb{E}[Z_{\mathcal{T}}] = (1-q)\ell_2 - q\ell_1$. Equating these two, we have $q = \frac{\ell_2}{\ell_1+\ell_2}$.

10.4 Wald's Equation

An important corollary of the martingale stopping time theorem is the Wald's equation.

Theorem 10.4.1 (Wald's Equation). *Let $\{\mathcal{X}_i\}_{i\in\mathbb{N}}$ be sequence of non-negative[3], independent, and identically distributed random variables. If $\mathcal{T}$ is a stopping time for the sequence (i.e. the event $\mathcal{T} = n$ depends only on the random variables $\mathcal{X}_i, i \in [n] \cup \{0\}$) and $\mathbb{E}|\mathcal{T}| < \infty$ and $\mathbb{E}|\mathcal{X}_0| < \infty$, then we have the following.*

$$\mathbb{E}\left[\sum_{i=0}^{\mathcal{T}} \mathcal{X}_i\right] = \mathbb{E}[\mathcal{T}]\mathbb{E}[\mathcal{X}_0]$$

Proof. For $n \in \mathbb{N}$, let us define $\mathcal{Z}_n = \sum_{i=0}^{n} (\mathcal{X}_i - \mathbb{E}[\mathcal{X}_0])$. Then, it follows that $\{\mathcal{Z}_i\}_{i\in\mathbb{N}}$ is a martingale with respect to $\{\mathcal{X}_i\}_{i\in\mathbb{N}}$. Also, we have $\mathbb{E}[\mathcal{Z}_0] = 0$.

We have $\mathbb{E}[\mathcal{T}] < \infty$ and we also have the following.

$$\mathbb{E}\left[|\mathcal{Z}_{i+1} - \mathcal{Z}_i| | \mathcal{X}_0, \ldots, \mathcal{X}_i\right] = \mathbb{E}\left[|\mathcal{X}_{i+1} - \mathbb{E}[\mathcal{X}_0]|\right] \leqslant 2\mathbb{E}[\mathcal{X}_0] < \infty$$

Hence, we can apply the martingale stopping time theorem by which we have the following.

$$\mathbb{E}\left[\mathcal{Z}_{\mathcal{T}}\right] = \mathbb{E}[\mathcal{Z}_0] = 0$$

On the other hand, we have the following from which the statement follows.

$$\begin{aligned}
\mathbb{E}\left[\mathcal{Z}_{\mathcal{T}}\right] &= \mathbb{E}\left[\sum_{i=0}^{\mathcal{T}} (\mathcal{X}_i - \mathbb{E}[\mathcal{X}_0])\right] \\
&= \mathbb{E}\left[\left(\sum_{i=0}^{\mathcal{T}} \mathcal{X}_i\right) - \mathcal{T}\mathbb{E}[\mathcal{X}_0]\right] \\
&= \mathbb{E}\left[\left(\sum_{i=0}^{\mathcal{T}} \mathcal{X}_i\right)\right] - \mathbb{E}\left[\mathcal{T}\right]\mathbb{E}[\mathcal{X}_0]
\end{aligned}$$

□

[3]The non-negativity assumption is a caveat of our proof. The result holds without the non-negativity assumption also.

Let us see an immediate use case of Wald's equation. In Las Vegas type randomized algorithm, we often run some randomized procedure which succeeds with probability p say repeatedly until the randomized procedure suceeds. If X_i is the running time of the i-th run of the algorithm and X_is are distributed independently and identically, then, by Wald's equation, the running time of our Las Vegas type randomized algorithm is as follows.

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] = \mathbb{E}[N]\,\mathbb{E}[X_1] = \frac{\mathbb{E}[X_1]}{p}$$

Next, we prove that Chernoff type concentration bounds hold for martingales too.

10.5 Tail Bounds for Martingales: Azuma-Hoeffding Inequality

Theorem 10.5.1 (Azuma-Hoeffding Inequality)**.** *Let $\{X_i\}_{i\in\mathbb{N}}$ be a martingale such that $|X_k - X_{k-1}| \leqslant c_k$ for every $k \in \mathbb{N}$. Then for every $t \in \mathbb{N}$ and $\lambda > 0$, we have the following.*

$$\Pr[|X_t - X_0| \geqslant \lambda] \leqslant 2\exp\left\{-\frac{\lambda^2}{2\sum_{i=1}^{t} c_i^2}\right\}$$

Proof. The proof follows the same template as the proof of Chernoff bound. Hence, we first have to derive an upper bound for $\mathbb{E}[\exp\{\alpha(X_t - X_0)\}]$ for $\alpha > 0$. We define the following for $i \in \mathbb{N}$,

$$Y_i = X_i - X_{i-1}$$

Hence, we have $|Y_i| < c_i$ for every $i \in \mathbb{N}$. We now have the following.

$$\begin{aligned}
\mathbb{E}[Y_i|X_0,\ldots,X_{i-1}] &= \mathbb{E}[X_i - X_{i-1}|X_0,\ldots,X_{i-1}] \\
&= \mathbb{E}[X_i|X_0,\ldots,X_{i-1}] - X_{i-1} \\
&= 0 && [\{X_i\}_{i\in\mathbb{N}} \text{ is a martingale}]
\end{aligned}$$

To bound $\mathbb{E}\left[e^{\alpha Y_i}|X_0,\ldots,X_{i-1}\right]$, we have the following.

$$\begin{aligned}
Y_i &= -c_i\frac{1-\frac{Y_i}{c_i}}{2} + c_i\frac{1+\frac{Y_i}{c_i}}{2} \\
\Rightarrow e^{\alpha Y_i} &\leqslant \frac{1-\frac{Y_i}{c_i}}{2}e^{-\alpha c_i} + \frac{1+\frac{Y_i}{c_i}}{2}e^{\alpha c_i} && [e^{\alpha x} \text{ is convex}] \\
&= \frac{e^{\alpha c_i} + e^{-\alpha c_i}}{2} + \frac{Y_i}{2c_i}\left(e^{\alpha c_i} - e^{-\alpha c_i}\right) \\
\Rightarrow \mathbb{E}\left[e^{\alpha Y_i}|X_0,\ldots,X_{i-1}\right] &\leqslant \frac{e^{\alpha c_i} + e^{-\alpha c_i}}{2} \\
&\leqslant e^{\frac{(\alpha c_i)^2}{2}}
\end{aligned}$$

We now bound $\mathbb{E}\left[e^{\alpha(X_t - X_0)}\right]$.

$$\begin{aligned}
\mathbb{E}\left[e^{\alpha(X_t - X_0)}\right] &= \mathbb{E}\left[\prod_{i=1}^{t} e^{\alpha Y_i}\right] \\
&= \mathbb{E}\left[e^{\alpha Y_t}\prod_{i=1}^{t-1} e^{\alpha Y_i}\right]
\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}\left[\mathbb{E}\left[e^{\alpha Y_t}\prod_{i=1}^{t-1}e^{\alpha Y_i}\middle| X_0,\ldots,X_{t-1}\right]\right]\\
&= \mathbb{E}\left[\mathbb{E}\left[e^{\alpha Y_t}\middle| X_0,\ldots,X_{t-1}\right]\prod_{i=1}^{t-1}e^{\alpha Y_i}\right]\\
&\leqslant \mathbb{E}\left[e^{\frac{(\alpha c_t)^2}{2}}\prod_{i=1}^{t-1}e^{\alpha Y_i}\right]\\
&= e^{\frac{(\alpha c_t)^2}{2}}\mathbb{E}\left[\prod_{i=1}^{t-1}e^{\alpha Y_i}\right]\\
&\leqslant e^{\alpha^2\frac{\sum_{i=1}^{t}c_i^2}{2}}
\end{aligned}
$$

Now, using Markov, we have the following.

$$
\begin{aligned}
\Pr\left[X_t - X_0 \geqslant \lambda\right] &= \Pr\left[e^{\alpha(X_t-X_0)} \geqslant e^{\alpha\lambda}\right]\\
&\leqslant \frac{\mathbb{E}\left[e^{\alpha(X_t-X_0)}\right]}{e^{\alpha\lambda}}\\
&\leqslant e^{\alpha^2\frac{\sum_{i=1}^{t}c_i^2}{2}-\alpha\lambda}\\
&\leqslant e^{-\frac{\lambda^2}{2\sum_{i=1}^{t}c_i^2}} \qquad \left[\alpha = \frac{\lambda}{2\sum_{i=1}^{t}c_i^2}\right]
\end{aligned}
$$

□

10.6 Applications of Azuma's Inequality: Concentration for Lipschitz Functions

Definition 10.6.1 (Lipschitz Functions)**.** *A function* $f:\mathbb{R}^n \longrightarrow \mathbb{R}$ *is called* $(c_i)_{i\in[n]}$*-Lipschitz for* $c_i \in \mathbb{R}_{\geqslant 0}, i \in [n]$ *if for every* $i \in [n]$ *and every* $x_1,\ldots,x_n,x_i' \in \mathbb{R}$*, we have the following.*

$$|f(x_1,\ldots,x_i,\ldots,x_n) - f(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_n)| \leqslant c_i$$

We can associate a martingale with respect to any Lipschitz function as follows.

Lemma 10.6.1. *Let* $f:\mathbb{R}^n \longrightarrow \mathbb{R}$ *be a bounded* $(c_i)_{i\in[n]}$*-Lipschitz function and* $X_1,X_2,\ldots,X_n$ *are independent. Let* $Z_1,\ldots,Z_n$ *be the corresponding Doob martingale. Then we have* $|Z_i - Z_{i-1}| \leqslant c_i$*.*

Proof. We will prove the result only for the case when the domain of f is a finite set.[4] Recall the definition of Z_i.

$$
\begin{aligned}
Z_i &= \mathbb{E}[f|X_1,\ldots,X_i]\\
&= \sum_{(a_{i+1},\ldots,a_n)\in \text{support}(X_{i+1},\ldots,X_n)} f(X_1,\ldots,X_i,a_{i+1},\ldots,a_n)\Pr[X_j = a_j \forall i+1 \leqslant j \leqslant n | X_1,\ldots,X_i]\\
&= \sum_{(a_{i+1},\ldots,a_n)\in \text{support}(X_{i+1},\ldots,X_n)} f(X_1,\ldots,X_i,a_{i+1},\ldots,a_n)\prod_{j=i+1}^{n}\Pr[X_j = a_j]
\end{aligned}
$$

[4] The result is true without this assumption.

We now bound $|Z_i - Z_{i-1}|$ as follows.

$$\begin{aligned}
|Z_i - Z_{i-1}| &= \sum_{(a_{i+1},\ldots,a_n)} \left| f(X_1,\ldots,X_i,a_{i+1},\ldots,a_n) - \sum_{a_i} f(X_1,\ldots,X_{i-1},a_i,\ldots,a_n)\Pr[X_i = a_i] \right| \\
&\quad \Pr[X_{i+1} = a_{i+1},\ldots,X_n = a_n] \\
&\leqslant \sum_{(a_{i+1},\ldots,a_n)} c_i \Pr[X_{i+1} = a_{i+1},\ldots,X_n = a_n] \\
&= c_i
\end{aligned}$$

□

Hence, we get the following corollary from the Azuma's inequality which is popularly known as McDiarmid's inequality.

Corollary 10.6.1 (McDiarmid's Inequality)**.** *If $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a bounded $(c_i)_{i\in[n]}$-Lipschitz function and $X_1, X_2, \ldots, X_n$ are independent, then we have the following.*

$$\Pr[|f - \mathbb{E}[f]| \geqslant \lambda] \leqslant 2\exp\left\{-\frac{\lambda^2}{2\sum_{i=1}^n c_i^2}\right\}$$

10.7 Applications of Concentration Bound for Lipschitz Functions: Balls and Bins

Suppose we throw m balls uniformly into n bins and we are interested in maximum load. Let $X_i, i \in [m]$ be a random variable denoting the bin where the i-th ball lands and $f(X_1,\ldots,X_m)$ be the maximum load. We observe that f is Lipschitz with $c_i = 1$ for every $i \in [m]$. Hence, using McDiarmid's inequality, we have the following.

$$\Pr\left[|f - \mathbb{E}[f]| \geqslant \varepsilon\sqrt{m}\right] \leqslant 2\exp\left\{-\varepsilon^2/2\right\}$$

10.8 Applications of Concentration Bound for Lipschitz Functions: Graph Chromatic Number

Let G be a random graph in $\mathcal{G}_{n,p}$, and $\chi(G)$ is the minimum number of colors needed to properly color the vertices of G. Considering the vertex exposure martingale, we observe that $\chi(G)$ is Lipschitz with $c_i = 1$ for every i. Hence, using McDiarmid's inequality, we have the following.

$$\Pr\left[|\chi(G) - \mathbb{E}[\chi(G)]| \geqslant \varepsilon\sqrt{n}\right] \leqslant 2\exp\left\{-\varepsilon^2/2\right\}$$

Chapter 11

Statistical Learning Theory

11.1 Basic Learning Framework

This chapter gives a brief introduction to the vast area of statistical learning theory. The framework is different from the classical algorithmic framework of worst-case analysis. So, let us begin with formally defining the basic learning framework. An algorithm is often interchangeably called a *learner* in this setting.

1. **Learner's Input:** The learner *knows* the domain set $\mathcal{X}$ and the label set $\mathcal{Y}$, which we will assume to be binary, that is, either $\{0, 1\}$ or $\{+1, -1\}$. Each point in $\mathcal{X}$ has a label which is unknown to the learner. The learner is also given m points from $\mathcal{X}$ along with its label $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$; this set is often called the training set. The learner also knows that the training set is generated by sampling each point from $\mathcal{X}$ using a fixed but unknown distribution $\mathcal{D}$ independently of everything else and then applying a fixed but unknown labeling function $f : \mathcal{X} \longrightarrow \mathcal{Y}$.

2. **Learner's Output:** The learner outputs a function $h : \mathcal{X} \longrightarrow \mathcal{Y}$, also called *prediction rule* or *hypothesis* or *classifier* or *model*.

3. **Performance Measure of the Learner:** The error of a prediction rule h is

$$L_{\mathcal{D},f}(h) = \Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)]$$

   $L_{\mathcal{D},f}(h)$ is also called *generalization error* or *risk* or true error of h.

Clearly, if the labeling function f is allowed to be arbitrary, no learner have any hope of correctly predicting the label of an unseen (not in training set S) point from $\mathcal{X}$. Hence, we also assume that the learner knows a class $\mathcal{H}$ of prediction rules such that there exists a prediction rule $h^* \in \mathcal{H}$ such that $L_{\mathcal{D},f}(h^*) = 0$. This is called the *realizability assumption*. The set $\mathcal{H}$ is called *hypothesis class* or *inductive bias*. Since the learner has access to only S, a natural hypothesis for the learner to output is one in $\mathcal{H}$ which makes least error in S. Such a learner is called empirical risk minimizer (ERM).

$$\text{ERM}_{\mathcal{H}}(S) \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h), \text{ where } L_S(h) = \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}.$$

With this basic framework, we are now ready to prove our first result in learning theory. Let the hypothesis class is finite.

Theorem 11.1.1. *Let $\mathcal{H}$ be a finite hypothesis class, $\delta \in (0,1)$, $\varepsilon > 0$, and m be any integer that satisfies*

$$m \geqslant \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon}.$$

Then, the loss of every ERM hypothesis h_S is at most ε with probability at least $1-\delta$ (over the randomness of S) for every labeling function f and every distribution $\mathcal{D}$, under realizability assumption.

Proof. Let $\mathcal{H}_B$ be the set of "bad" hypotheses, that is

$$\mathcal{H}_B = \{h \in \mathcal{H} : L_{\mathcal{D},f}(h) > \varepsilon\}.$$

It is enough to show that the probability (over the randomness of training set) that any ERM hypothesis belongs to $\mathcal{H}_B$ is at most δ. Let $\mathcal{M}$ be the training sets, restricted to $\mathcal{X}$, where some bad hypothesis works perfect. That is,

$$\mathcal{M} = \{S|_x : \exists h \in \mathcal{H}_B, L_S(h) = 0\}$$

Note that, ERM algorithm outputs a bad hypothesis only if the training set S belongs to $\mathcal{M}$. This is so because of the realizability assumption: there exists a hypothesis $h^* \in \mathcal{H}$ such that $L_S(h^*) = 0$ for every training set S. Let $h \in \mathcal{H}_B$ be any bad hypothesis. Then we have

$$\Pr[\{x \in \mathcal{X} : h(x) = h^*(x)\}] = 1 - L_{\mathcal{D},f}(h) \leqslant 1 - \varepsilon,$$

and thus

$$\Pr[S : L_{\mathcal{D},f}(h) = 0] \leqslant (1-\varepsilon)^m \leqslant e^{-\varepsilon m}.$$

Now using union bound, we have

$$\Pr[S : \exists h \in \mathcal{H}_B, L_{\mathcal{D},f}(h) = 0] \leqslant |\mathcal{H}_B| e^{-\varepsilon m} \leqslant |\mathcal{H}| e^{-\varepsilon m} \leqslant \delta$$

for $m \geqslant \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon}$. □

Theorem 11.1.1 says that, for any sufficiently large training set, the $\text{ERM}_{\mathcal{H}}$ hypothesis, over a finite hypothesis class $\mathcal{H}$, is *probably* (with confidence at least $1-\delta$) *approximately* (up to an error of ε) correct. This called formally called the model of Probably Approximately Correct (PAC) learning.

Definition 11.1.1 (PAC Learnability)**.** *A hypothesis class $\mathcal{H}$ is called PAC learnable if there exists a function $m_{\mathcal{H}} : (0,1)^2 \longrightarrow \mathbb{N}$ and a learning algorithm such that: for every $\varepsilon, \delta \in (0,1)$, distribution $\mathcal{D}$ over $\mathcal{X}$, and labeling function $f : \mathcal{X} \longrightarrow \{0,1\}$, if the realizability assumption holds with respect to $\mathcal{H}$, $\mathcal{D}$, f, then when running the learning algorithm on $m \geqslant m_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that $L_{\mathcal{D},f}(h) \leqslant \varepsilon$ with probability at least $1-\delta$.*

The function $m_{\mathcal{H}}$ in Definition 11.1.1 is called *sample complexity* of learning $\mathcal{H}$. Hence, Theorem 11.1.1 shows that every finite hypothesis class is PAC learnable with sample complexity $\left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon} \right\rceil$.

11.2 A More General Learning Model

We first do away with our realizability assumption. The mativation for this is that our view of the world is almost always incomplete. In many real life scenarios, medical domain for example, we are rarely able to predict some medical condition with 100% accuracy from any set of physiological parameters. Hence,

assuming that there exists a deterministic function which can assign a label from a feature vector is often unrealistic. Instead, we assume our distribution $\mathcal{D}$ to be over $\mathcal{X} \times \mathcal{Y}$. The true error or true risk of a prediction rule $h : \mathcal{X} \longrightarrow \mathcal{Y}$ is

$$L_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y] = \mathcal{D}(\{(x, y) \in \mathcal{X} \times \mathcal{Y} : h(x) \neq y\}).$$

However, the learner does not know $\mathcal{D}$; it has access to a trainig sample $\mathcal{S}$. Our goal is to find a prediction rule which has the minimum true eror. For binary label, that is $\mathcal{Y} = \{0, 1\}$, and a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$, the prediction rule achieving the minimum true error is

$$f_{\mathcal{D}}(x) = \begin{cases} 1 & \text{if } \Pr[y = 1|x] \geqslant 1/2 \\ 0 & \text{otherwise.} \end{cases}$$

This is called Bayes optimal predictor. Unfortunately, since the learner does not know $\mathcal{D}$, it cannot compute $f_{\mathcal{D}}$. Furthermore, it is known that, without any assumption on $\mathcal{D}$, no algorithm can compute a predictor which is as good as $f_{\mathcal{D}}$: this is the famous *No Free Lunch Theorem*. Hence, we modify our goal to finding the best possible predictor in a given hypothesis class.

Definition 11.2.1 (Agnostic PAC Learnability)**.** *A hypothesis class $\mathcal{H}$ is called agnostic PAC learnable if there exists a function $m_{\mathcal{H}} : (0, 1)^2 \longrightarrow \mathbb{N}$ and a learning algorithm such that: for every $\varepsilon, \delta \in (0, 1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$, when running the learning algorithm on $m \geqslant m_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$, the algorithm returns a hypothesis h such that $L_{\mathcal{D}}(h) \leqslant \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') + \varepsilon$ with probability at least $1 - \delta$.*

We also use a *generalized loss function*. Given any set $\mathcal{H}$ (which plays the role of our hypothesis class) and any domain $\mathcal{Z}$ (plays the role of $\mathcal{X} \times \mathcal{Y}$), a generalized loss function is any function $\ell : \mathcal{H} \times \mathcal{Z} \longrightarrow \mathbb{R}_{\geqslant 0}$. With respect to a loss function ℓ, the true error of a prediction rule is

$$L_{\mathcal{D}}(h) = \mathbb{E}_{z\sim\mathcal{D}}[\ell(h, z)].$$

The loss function that we have been working so far is the $0 - 1$ loss function which is commonly used for classification tasks:

$$\ell_{0-1}(h, (x, y)) = \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{otherwise.} \end{cases}$$

The squared loss function is suitable for regression problems.

$$\ell_{sq}(h, (x, y)) = (h(x) - y)^2$$

Definition 11.2.2 (Agnostic PAC Learnability with General Loss Function)**.** *A hypothesis class $\mathcal{H}$ is called agnostic PAC learnable with respect to a set $\mathcal{Z}$ and a loss function $\ell : \mathcal{H} \times \mathcal{Z} \longrightarrow \mathbb{R}_{\geqslant 0}$ if there exists a function $m_{\mathcal{H}} : (0, 1)^2 \longrightarrow \mathbb{N}$ and a learning algorithm such that: for every $\varepsilon, \delta \in (0, 1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$, when running the learning algorithm on $m \geqslant m_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$, the algorithm returns a hypothesis h such that $L_{\mathcal{D}}(h) \leqslant \min_{h'\in\mathcal{H}} L_{\mathcal{D}}(h') + \varepsilon$ with probability at least $1 - \delta$.*

Can we show a result equivalent to Theorem 11.1.1 for agnostic PAC learnability with general loss functions? The answer is yes! For that, we introduce a toll called *uniform convergence*.

We observe that the ERM algorithm still exist in our updated model: given a training set $\mathcal{S}$, the algorithm evaluates each hypothesis h in the hypothesis class $\mathcal{H}$, and outputs one which has the minimum empirical

error.[1] This approach works if, for every hypothesis $h \in \mathcal{H}$, empirical error is a good proxy for true error. This idea is formalized in ε-representative samples.

Definition 11.2.3 (ε-representative sample)**.** *A training set S is called ε-representative with respect to a domain Z, hypothesis class $\mathcal{H}$, loss functino ℓ, and distribution $\mathcal{D}$, if*

$$\forall h \in \mathcal{H}, |L_{\mathcal{D}}(h) - L_S(h)| \leqslant \varepsilon.$$

For ease of exposition, we omit saying "with respect to a domain Z, hypothesis class $\mathcal{H}$, loss functino ℓ, and distribution $\mathcal{D}$" every time we say ε-representative samples and it should be understood implicit.

Observation 11.2.1. *If S is an $\frac{\varepsilon}{2}$-representative sample, then*

$$L_{\mathcal{D}}(h_{ERM}) \leqslant \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \varepsilon.$$

Proof. For every $h \in \mathcal{H}$, we have

$$L_{\mathcal{D}}(h_{ERM}) \leqslant L_S(h_{ERM}) + \frac{\varepsilon}{2} \leqslant L_S(h) + \frac{\varepsilon}{2} \leqslant L_{\mathcal{D}}(h) + \varepsilon.$$

□

Hence, to ensure that ERM is an agnostic PAC learner, it is enough to ensure that the probability that the training set S is an $\frac{\varepsilon}{2}$-representative sample is at least $1 - \delta$. This is called uniform convergence.

Definition 11.2.4 (Uniform Convergence)**.** *We say that a hypothesis class has the uniform convergence property with respect to a domain Z and a loss function ℓ if there exists a function $m^{UC}_{\mathcal{H}} : (0,1)^2 \longrightarrow \mathbb{N}$ such that, for every $\varepsilon, \delta \in (0,1)$ and every probability distribution $\mathcal{D}$ over Z, a sample S of size at least $m^{UC}_{\mathcal{H}}(\varepsilon, \delta)$ drawn i.i.d. using $\mathcal{D}$, then S is an ε-representative sample with probability at least $1 - \delta$.*

Hence, we have the following immediate corollary from Observation 11.2.1.

Corollary 11.2.1. *If a hypothesis class $\mathcal{H}$ has the uniform convergence property, then $ERM_{\mathcal{H}}$ is an agnostic PAC learner for the class with sample complexity $m_{\mathcal{H}}(\varepsilon, \delta) = m^{UC}_{\mathcal{H}}(\frac{\varepsilon}{2}, \delta)$.*

We now show that finite hypothesis classes are agnostic PAC learnable.

Theorem 11.2.1. *Let $\mathcal{H}$ be a finite hypothesis class, Z a domain, and $\ell : \mathcal{H} \times Z \longrightarrow [0,1]$ a loss function. Then $\mathcal{H}$ has uniform convergence property with sample complexity*

$$m^{UC}_{\mathcal{H}}(\varepsilon, \delta) \leqslant \left\lceil \frac{\ln(2|\mathcal{H}|/\delta)}{2\varepsilon^2} \right\rceil.$$

Hence, ERM is an agnostic PAC learner with sample complexity

$$m_{\mathcal{H}}(\varepsilon, \delta) \leqslant \left\lceil \frac{2\ln(2|\mathcal{H}|/\delta)}{\varepsilon^2} \right\rceil.$$

Proof. Fix a prediction rule $h \in \mathcal{H}$. Let $S = (z_1, \ldots, z_m)$ be m i.i.d. samples from $\mathcal{D}$. We define a random variable $Z_i = \ell(h, z_i)$ for $i \in [m]$. Then $L_S(h) = \frac{1}{m}\sum_{i=1}^{m} Z_i$ and $Ł_{\mathcal{D}}(h) = \mathbb{E}[\frac{1}{m}\sum_{i=1}^{m} Z_i]$. Using Theorem 9.2.1, we have the following.

$$\Pr[|L_S(h) - L\mathcal{D}(h)| > \varepsilon] = \Pr[|\sum_{i=1}^{m} Z_i - \mathbb{E}[\sum_{i=1}^{m} Z_i]| > m\varepsilon] \leqslant 2\exp(-2m\varepsilon^2).$$

[1] Let us not worry about running time now.

Now using union bound, we have for $m = \frac{\ln(2|\mathcal{H}|/\delta)}{2\varepsilon^2}$.

$$\Pr[\exists h' \in \mathcal{H} : |L_S(h') - L_{\mathcal{D}}(h')| > \varepsilon] \leqslant 2|\mathcal{H}| \exp(-2m\varepsilon^2) \leqslant \delta.$$

□

11.3 No Free Lunch Theorem

We now prove that there is no universal learner in the sense that it can achieve arbitrarily small error with arbitrarily high confidence without bechmarking any hypothesis class.

Theorem 11.3.1 (No Free Lunch)**.** *Let $\mathcal{A}$ be any learner for the binary classification task with $0-1$ loss over a domain $\mathcal{X}$ and the size m of the training set is at most $\frac{|\mathcal{X}|}{2}$. Then there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:*

(i) There exists a function $f : \mathcal{X} \longrightarrow \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$ and

(ii) With probability at least $\frac{1}{7}$ over the randomness of S, we have $L_{\mathcal{D}}(\mathcal{A}(S)) \geqslant \frac{1}{8}$.

Proof. Let $\mathcal{C}$ be any subset of $\mathcal{X}$ of cardinality $2m$. The idea is to ensure that any learner which sees only half of $\mathcal{C}$ has no clue about the labels of unseen points. There are $T = 2^{2m}$ functions from $\mathcal{C}$ to $\{0,1\}$. Let they be $f_1, \ldots, f_T$. For each $i \in [T]$, we define a distribution $\mathcal{D}_i$ on $\mathcal{C} \times \{0,1\}$ as

$$\mathcal{D}_i(x,y) = \begin{cases} \frac{1}{|\mathcal{C}|} & \text{if } y = f_i(x) \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, we have $L_{\mathcal{D}_i}(f_i) = 0$. We will show that, for every learner $\mathcal{A}$ that receives a training set of size m from $\mathcal{C} \times \{0,1\}$ and outputs a prediction rule $\mathcal{A}(S) : \mathcal{C} \longrightarrow \{0,1\}$ satisfies

$$\max_{i \in [T]} \mathbb{E}_S[L_{\mathcal{D}_i}(\mathcal{A}(S))] \geqslant \frac{1}{4}$$

which proves the result. There are $k = (2m)^m$ possible sequences of points from $\mathcal{C}$; let they be $S_1, \ldots, S_k$. If $S_j = (x_1, \ldots, x_m)$, then by S_j^i we denote $((x_1, f_i(x_1)), \ldots, (x_m, f_i(x_m)))$ for $i \in [T]$. Then, we have

$$\mathbb{E}_S[L_{\mathcal{D}_i}(\mathcal{A}(S))] = \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i))$$

We now have

$$\begin{aligned} \max_{i \in [T]} \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) &\geqslant \frac{1}{T} \sum_{i=1}^{T} \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \\ &= \frac{1}{k} \sum_{j=1}^{k} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \\ &\geqslant \min_{j \in [k]} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \end{aligned}$$

For some fixed training set $j \in [k]$, let $v_1, \ldots, v_p \in \mathcal{X}$ be the points that do not appear in the training set S_j. We have $p \geqslant m$. For every hypothesis $h : C \longrightarrow \{0, 1\}$ and every $i \in [T]$, we have

$$\begin{aligned} L_{\mathcal{D}_i}(h) &= \frac{1}{2m} \sum_{x \in C} \mathbb{I}[h(x) \neq f_i(x)] \\ &\geqslant \frac{1}{2m} \sum_{r=1}^{p} \mathbb{I}[h(v_r) \neq f_i(v_r)] \\ &\geqslant \frac{1}{2p} \sum_{r=1}^{p} \mathbb{I}[h(v_r) \neq f_i(v_r)] \end{aligned}$$

Hence,

$$\begin{aligned} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) &\geqslant \frac{1}{T} \sum_{i=1}^{T} \frac{1}{2p} \sum_{r=1}^{p} \mathbb{I}[h(v_r) \neq f_i(v_r)] \\ &= \frac{1}{2p} \sum_{r=1}^{p} \frac{1}{T} \sum_{i=1}^{T} \mathbb{I}[h(v_r) \neq f_i(v_r)] \\ &\geqslant \frac{1}{2} \min_{r \in [p]} \frac{1}{T} \sum_{i=1}^{T} \mathbb{I}[h(v_r) \neq f_i(v_r)] \\ &= \frac{1}{2} \min_{r \in [p]} \frac{1}{T} \frac{T}{2} \\ &= \frac{1}{4}. \end{aligned}$$

We thus have

$$\max_{i \in [T]} \mathbb{E}_S[L_{\mathcal{D}_i}(\mathcal{A}(S))] = \max_{i \in [T]} \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \geqslant \min_{j \in [k]} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \geqslant \frac{1}{4}.$$

□

Theorem 11.3.1 in particular shows that the set of all functions from an infinite domain $\mathcal{X}$ to $\{0, 1\}$ (which is equivalent to having no prior knowledge) as an hypothesis class is not PAC learnable. This justifies our assumption of prior knowledge that we know a hypothesis class $\mathcal{H}$ which. for every distribution, contains a prediction rule that performs well on $\mathcal{Z}$.

11.4 VC Dimension

We have seen that finite hypothesis classes are PAC as well as agnostic PAC learnable. On the other hand, we have also seen that, for any infinite domain $\mathcal{X}$, the hypothesis class of all prediction rules from $\mathcal{X}$ to $\{0, 1\}$ is not PAC learnable. Can we have an infinite hypothesis class which is PAC learnable? Is there any crisp characterization of a hypothesis class that determines its PAC learnability?

We begin our discussion with an example of a infinite class which is PAC learnable.

Lemma 11.4.1. *Let $\mathcal{H} = \{h_a : a \in \mathbb{R}\}$ where $h_a(x) = \mathbb{I}\{x < a\}$ be the set of threshold functions over the real line. Then $\mathcal{H}$ is PAC learnable (that is, under the realizability assumption), using ERM, with sample complexity $m_{\mathcal{H}} \leqslant \lceil \frac{\ln(2/\delta)}{\varepsilon} \rceil$.*

Proof. Let $a^* \in \mathbb{R}$ be such that $L_{\mathcal{D}}(h_{a^*}) = 0$, $\mathcal{D}_x$ the marginal distribution of $\mathcal{D}$ over $\mathcal{X}$, and a_l and a_r be such that

$$\Pr_{x \sim \mathcal{D}_x}[x \in (a_l, a^*)] = \Pr_{x \sim \mathcal{D}_x}[x \in (a^*, a_r)] = \varepsilon.$$

Given a training set S of size m, let

$$b_l = \max\{x \in \mathcal{X} : (x, 1) \in S\} \text{ and } b_r = \min\{x \in \mathcal{X} : (x, 0) \in S\}.$$

Let $b \in \mathbb{R}$ be the threshold corresponding to the ERM hypothesis with respect to S. Then we have $b \in [b_l, b_r]$. It is enough to show $[b_l, b_r] \subseteq [a_l, a_r]$. We observe that $b_l < a_l$ if and only if all the training points not fall in the interval $[a_l, a^*]$. Hence, we have

$$\Pr[b_l < a_l] \leqslant (1 - \varepsilon)^m \leqslant \exp\{-\varepsilon m\}.$$

Similarly, we have

$$\Pr[b_r > a_r] \leqslant \exp\{-\varepsilon m\}.$$

Now using union bound, we have, for $m = \frac{\ln(2/\delta)}{\varepsilon}$,

$$\Pr[[b_l, b_r] \subseteq [a_l, a_r]] = 1 - \Pr[b_l < a_l \text{ or } b_r > a_r] \geqslant 1 - 2\exp\{-\varepsilon m\} \geqslant 1 - \delta.$$

□

Hence, although the hypothesis class being finite is a sufficient condition for PAC learnability, it is not necessary. It turns out that there is another property of a hypothesis class, called VC dimension, which is the correct characterization of PAC learnability. To build the theory of VC dimension, let us start at no-free-lunch theorem and ask how the adversary in the proof of no-free-lunch theorem was able to make any learning algorithm fail. The adversary first chooses a finite set $C \subset \mathcal{X}$ and constructed a family of distributions all concentrated on C. Each distribution was defined using a function from C to $\{0, 1\}$. Since the supports of all these distributions are subsets of C, as far as PAC learnability of any hypothesis class $\mathcal{H}$ is concerned, all that matters is how $\mathcal{H}$ behaves on C. This is called restriction of $\mathcal{H}$ to C.

Definition 11.4.1 (Restriction of $\mathcal{H}$ to C)**.** *Let $\mathcal{H}$ be any hypothesis class and $C = \{c_1, \ldots, c_m\} \subseteq \mathcal{X}$ any subset of domain $\mathcal{X}$. The restriction of $\mathcal{H}$ to C is the set of functions of $\mathcal{H}$ restricted to C which is*

$$\mathcal{H}_C = \{(h(c_1), \ldots, h(c_m)) : h \in \mathcal{H}\}^2.$$

If the restriction of $\mathcal{H}$ to C is the set of all functions from C to $\{0, 1\}$, then we say that $\mathcal{H}$ *shatters* C. For example, let us take $\mathcal{H} = \{h_a : a \in \mathbb{R}\}$ where $h_a(x) = \mathbb{I}\{x < a\}$ to be the set of all threshold functions over $\mathbb{R}$. We can see that $\mathcal{H}$ shatters every singleton set but no subset of $\mathbb{R}$ of size at least 2. We see that, if a set C is shattered by the hypothesis class $\mathcal{H}$, then the adversary "fool" any learner by designing distributions over C as demonstrated in the proof of No-Free-Lunch theorem.

Observation 11.4.1. *Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\{0, 1\}$ and m the size of training set. If there exists a set $C \subseteq \mathcal{X}$ of size $2m$ that is shattered by $\mathcal{H}$, then, for every learning algorithm $\mathcal{A}$, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$ and a predictor $h : \mathcal{X} \longrightarrow \{0, 1\}$ such that $L_{\mathcal{D}}(h) = 0$ but $L_{\mathcal{D}}(\mathcal{A}(S)) \geqslant \frac{1}{8}$ with probability at least $\frac{1}{7}$.*

[2]Any function from C to $\{0, 1\}$ can be equivalently denoted as a bit vector of length $|C|$.

Proof. Ditto as No-Free-Lunch theorem. □

Observation 11.4.1 says that, if $\mathcal{H}$ shatters any set of size $2m$, then we cannot learn $\mathcal{H}$ with m samples. This leads to the definition of VC dimension.

Definition 11.4.2 (VC Dimension)**.** *The VC dimension of a hypothesis class $\mathcal{H}$ is the maximum size of any subset C of $\mathcal{X}$ that $\mathcal{H}$ shatters. If $\mathcal{H}$ shatters arbitrarily large subsets of $\mathcal{X}$, then the VC dimension is defined to be infinity.*

Again, from the proof of No-Free-Lunch theorem, we see that every hypothesis class of infinite VC dimension is not PAC learnable. Let us see some hypothesis classes and their VC dimension.

- ▷ Let the domain be $\mathcal{X} = \mathbb{R}$, $\mathcal{H} = \{h_{(a,b)} : -\infty < a < b < \infty\}$ the set of intervals where $h_{(a,b)}(x) = \mathbb{I}\{x \in (a, b)\}$. We see that $\mathcal{H}$ shatters every subsets of $\mathbb{R}$ of size at most 2 but no subset of $\mathbb{R}$ of size more than 2. Hence, VCdim($\mathcal{H}$)= 2.
- ▷ Let the domain be $\mathcal{X} = \mathbb{R}^2$, $\mathcal{H} = \{h_{a_1,a_2,b_1,b_2} : -\infty < a_1 < a_2 < \infty, -\infty < b_1 < b_2 < \infty\}$ the set of all axis-parallel rectangles where $h_{a_1,a_2,b_1,b_2}(x_1, x_2) = \mathbb{I}\{a_1 \leqslant x_1 \leqslant a_2 \text{ and } b_1 \leqslant x_2 \leqslant b_2\}$. We see that $\mathcal{H}$ shatters the vertices of every non-degenarate rectangles in $\mathbb{R}^2$ but no subset of $\mathbb{R}^2$ of size at least 5. Hence, VCdim($\mathcal{H}$)= 4.
- ▷ Let the domain be $\mathcal{X} = \mathbb{R}$, and $\mathcal{H} = \{\lceil 0.5 \sin(\theta x) : \theta \in \mathbb{R}\rceil\}$. It can be shown that VCdim($\mathcal{H}$)= ∞.

The following result is known as the fundamental theorem of statistical learning theory.

Theorem 11.4.1 (Fundamental Theorem of Statistical Learning Theory)**.** *Let $\mathcal{H}$ be a hypothesis class from any domain $\mathcal{X}$ to $\{0, 1\}$ of VC dimension $d(< \infty)$ and $0 - - - 1$ loss function is used. Then there exists constants C_1 and C_2 such that:*

1. *$\mathcal{H}$ satisfies the uniform convergence property with sample complexity*

$$C_1 \frac{d + \ln(1/\delta)}{\varepsilon^2} \leqslant m^{UC}_{\mathcal{H}}(\varepsilon, \delta) \leqslant C_2 \frac{d + \ln(1/\delta)}{\varepsilon^2}$$

2. *$\mathcal{H}$ is agnostic PAC learnable with sample complexity*

$$C_1 \frac{d + \ln(1/\delta)}{\varepsilon^2} \leqslant m^{UC}_{\mathcal{H}}(\varepsilon, \delta) \leqslant C_2 \frac{d + \ln(1/\delta)}{\varepsilon^2}$$

3. *$\mathcal{H}$ is PAC learnable with sample complexity*

$$C_1 \frac{d + \ln(1/\delta)}{\varepsilon^2} \leqslant m^{UC}_{\mathcal{H}}(\varepsilon, \delta) \leqslant C_2 \frac{d + \ln(1/\delta)}{\varepsilon^2}$$

Acknowledgement

I thank Prof. Jaikumar Radhakrishnan for insightful discussion and his notes on path coupling. I also thank my colleagues and students for their suggestions.

Bibliography

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. 37-th Conference on Foundations of Computer Science (FOCS)*, pages 184–193. IEEE, 1996.

[BC05] Bo Brinkman and Moses Charikar. On the impossibility of dimension reduction in l_1. *J. ACM*, 52(5):766–788, 2005.

[FKS84] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, June 1984.

[LN17] Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638. IEEE, 2017.

[MH96] C.J.H. McDiarmid and R.B. Hayward. Large deviations for quicksort. *J. Algorithms*, 21(3):476–507, November 1996.

[Pag06] Rasmus Pagh. Cuckoo hashing for undergraduates. *IT University of Copenhagen*, 6, 2006.

[PR04] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.

[RR98] Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.