CSE 521: Design and Analysis of Algorithms I **Fall 2025**

Lecture 3: Concentration Bounds and Hashing

Lecturer: Shayan Oveis Gharan *10/01/2025* *Scribe:*

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

In the previous lecture, we learnt about some of the concentration bounds commonly used in probability theory. We are going to learn some more in this lecture.

3.1 Law of Large Numbers

The Law of Large Numbers (LLN) is a theorem which states that the average of the results obtained from a large number of independent trials of an experiment tends towards the expected value. Central limit theorems state that for an infintie sequence of random independent variables $X_1, X_2, \ldots$ with mean μ and a bounded variance σ^2,

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right) \to \mathcal{N}(0, \sigma^2). \tag{3.1}$$

as n goes to infinity. In this course, we are interested in quantitative forms of this convergence. We will study this in the form of *strong concentration bounds*, a.k.a., Chernoff bounds.

Recall that Chebyshev's inequality implies that for any random variable X,

$$\mathbb{P}\left[|X - \mathbb{E}\left[X\right]| \geq k\sigma\right] \leq \frac{1}{k^2} \tag{3.2}$$

Strong concentration bounds imply that if X is an average of independent random variables with standard deviation σ, and satisfy certain other properties, then

$$\mathbb{P}\left[|X - \mathbb{E}X| \geq k\sigma\right] \leq e^{-\Omega(k^2)}$$

In other words, they give exponentially improved bounds compared to Chebyshev's inequality. Note that to get this strong bound we want X to be an average of mutually independent random variables; so unlike Chebyshev's inequality pairwise independent is not enough.

3.2 Hoeffding's Inequality

In this part we discuss Hoeffding's inequality. We will see more concentration bounds in the assignments.

Let $X_1, X_2, \ldots X_n$ be n independent random variables such that for all i, $X_i \in [a_i, b_i]$, and let $X = \frac{X_1 + \cdots + X_n}{n}$. Then,

$$\mathbb{P}\left[|X - \mathbb{E}\left[X\right]| \geq \epsilon\right] \leq 2\exp\left(\frac{-2n^2\epsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{3.3}$$

Let us give another interpretation of this inequality. First, observe that

$$\mathrm{Var}[X] = \frac{1}{n^2}\sum_i \mathrm{Var}[X_i] \leq \frac{\sum_i (b_i - a_i)^2}{n^2}.$$

The last inequality uses that $X_i \in [a_i, b_i]$ for all i. So, we can rewrite the above as

$$\mathbb{P}\left[|X - \mathbb{E}X| \geq \epsilon\right] \leq 2\exp(-2\epsilon^2/\operatorname{Var} X)$$

Next, we discuss several applications of Hoeffding's inequality.

3.2.1 Application 1: Polling

Let us continue the polling example that we discussed in the last lecture. Consider a set of n Bernoulli random variables $X_1, X_2, \ldots X_n$ where for all i, $X_i = 1$ w.p. p and $X_i = 0$ w.p. $1-p$. By Hoeffding's inequality,

$$\mathbb{P}\left[\left|\frac{\sum X_i}{n} - p\right| \geq \epsilon\right] \leq 2\exp(\frac{-2n^2\epsilon^2}{n}) = 2\exp(-2n\epsilon^2) \tag{3.4}$$

where we used that $a_i = 0, b_i = 1$.

So, if we want to estimate the probability p within an additive error ϵ with probability $1-\delta$ it is enough to let

$$n = \frac{\ln(2/\delta)}{2\epsilon^2}.$$

To give you a point of comparison, recall that in the last lecture we showed that using Chebyshev inequality, to estimate p with additive error of ϵ with probability at $1-\delta$ we need about $\frac{1}{\delta\epsilon^2}$. So, for example, if we want $1 - 2^{-100}$ probability of success Hoeffding inequality implies we only need $100/\epsilon^2$ many samples, whereas Chebyshev's inequality says we want $2^{100}/\epsilon^2$ many samples. You can see that Hoeffding's inequality implies a significantly smaller number of samples.

Let us give a second example: suppose we have $n = 100$ samples; we want to see for what value of ϵ we can have 99% probability of success? It follows that we get $\epsilon = \frac{1}{6}$. Now, suppose we increase the number of samples to $n = 10000$. How much can we decrease ϵ to get the same 99% probability of success? Observe that we can only let $\epsilon \approx \frac{1}{60}$. Thus, we can decrease ϵ only proportional to square-root of the number of samples.

Upshot: The failure probability decreases exponentially with respect to the number of samples whereas the confidence interval ϵ only decreases proportional to the square-root of the number of samples.

3.2.2 Application 2: Random walk

A random walk is a random process that describes a path consisting of a succession of random steps on some mathematical space. Let us consider the one-dimensional random walk model, in which a person starts at the position 0 (origin) of the integer number line and moves to his right (+1) or to his left (-1) at each step with equal probability. Fig. 3.1 shows the model.

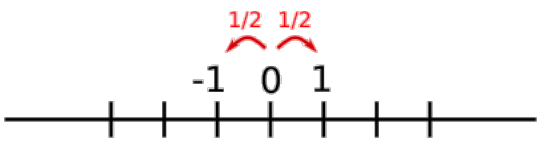

Figure 3.1: One-dimensional random walk model

To define the walk formally, let X_i be a random variable denoting whether the person moves to his right or left in the i^{th} step, so that $X_i = +1$ w.p. $\frac{1}{2}$ and $X_i = -1$ w.p. $\frac{1}{2}$. Thus $X = \sum_{i=1}^{n} X_i$ denotes the final position of the person after n steps. We want to estimate X.

First, observe that $\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = 0$. Hence according to Hoeffding's inequality,

$$\mathbb{P}\left[\left|\frac{X}{n} - 0\right| \geq \epsilon\right] \leq 2e^{\frac{-2n^2\epsilon^2}{4n}} \approx 2e^{\frac{-n\epsilon^2}{2}} \tag{3.5}$$

So,

$$\mathbb{P}[X \geq n\epsilon] \leq 2e^{\frac{-n\epsilon^2}{2}} \tag{3.6}$$

So with high probability, we can say that the person is at most at a distance of $\sqrt{n}$ away from the origin after n steps.

In the next lecture, we prove that with high probability the person is at distance at least $\Omega(\sqrt{n})$ of the origin.

3.2.3 Application 3: Discrepancy theory

In this part we discuss an application of Hoeffding's inequality in discrepancy theory. This is an important area of mathematics and it has many application is several areas of computer science including computational complexity and approximation algorithms.

Given a matrix $A \in \{0,1\}^{n\times n}$, we want to find a vector $x \in \{-1,+1\}^n$ such that $\|Ax\|_\infty \leq$ a small value. Note that for a vector $x \in \mathbb{R}^n$,

$$\|x\|_\infty = \max_i |x_i|.$$

In other words, we would like to color the columns of A with $+1$ or -1 such that the sum is as close to $\mathbf{0}$ as possible in the ℓ_∞ norm.

We show that if we choose x uniformly at random, then with high probability $\|Ax\|_\infty \leq O(\sqrt{n \log n})$.

Theorem 3.1. *Suppose we choose each coordinate of x uniformly at random in $\{+1,-1\}$. Then,*

$$\mathbb{P}\left[\|Ax\|_\infty \leq \sqrt{4n \ln n}\right] \geq 1 - 2/n.$$

Let $a_1, a_2, \ldots, a_n$ be the rows of A, i.e.,

$$A = \begin{pmatrix} \ldots a_1 \ldots \\ \ldots a_2 \ldots \\ \cdot \\ \cdot \\ \ldots a_n \ldots \end{pmatrix} \tag{3.7}$$

Then, we can write,

$$Ax = \begin{pmatrix} < a_1, x > \\ < a_2, x > \\ \cdot \\ \cdot \\ < a_n, x > \end{pmatrix} \tag{3.8}$$

So, to upper bound $\|Ax\|_\infty$, it is enough to show that with high probability for i, $\langle a_i, x\rangle \leq \sqrt{4n \log n}$.

Fix some $1 \leq i \leq n$. First, we show that with high probability $\langle a_i, x\rangle \leq \sqrt{4n \log n}$. First, observe that

$$\mathbb{E}[\langle a_i, x\rangle] = \sum_j a_{i,j} \mathbb{E}[x_j] = 0.$$

Note that in the expression $\sum a_{i,j}x_j$ any j for which $a_{i,j} = 0$, the value of x_j is irrelevant. Let $\|a_i\|_1 = \sum_j |a_{i,j}|$ be the number of nonzero entries of row i. Then, $\sum_j a_{i,j}x_j$ is distributed exactly the same as a random walk process on a line (that we discussed in the last section) of length $\|a_i\|_1$.

So, by (3.6), we have

$$\mathbb{P}\left[|\langle a_i, x\rangle| \geq \epsilon\|a_i\|_1\right] \leq 2e^{-\|a_i\|_1\epsilon^2/2} \tag{3.9}$$

We just replaced n with $\|a_i\|_1$ in the bound in (3.6).

So, for $\epsilon = \sqrt{4\ln n/\|a_i\|_1}$, we have

$$\mathbb{P}\left[|\langle a_i, x\rangle| \geq \sqrt{4\ln n\|a_i\|_1}\right] \leq 2e^{-2\ln n} = \frac{2}{n^2}. \tag{3.10}$$

Now, we use the union bound.

Union Bound Suppose we have a m (possibly intersecting) probability events $E_1, E_2, \ldots, E_m$. Then,

$$\mathbb{P}\left[\cup E_i\right] \leq \sum \mathbb{P}\left[E_i\right].$$

The proof of this simply follows from the following set-theoretic statement: For any family of sets $S_1, \ldots, S_m$,

$$|S_1 \cup S_2 \cup \cdots \cup S_m| \leq |S_1| + \ldots |S_m|.$$

Union bound is used a lot in conjunction with strong concentration bound. The reason is that strong concentration bounds prove a very sharp and small probability of failure so that even if we have many possibilities for failure still we can say none of them occur with high probability.

In the above, we showed that for any row i, with probability at most $2/n^2$,

$$|\langle a_i, x\rangle| \geq \sqrt{4\|a_i\|_1 \ln n}.$$

So, by union bound, with probability at least $1 - 2/n$, for all i,

$$|\langle a_i, x\rangle| \leq \sqrt{4\|a_i\|_1 \ln n} \leq \sqrt{4n \ln n}.$$

In other words, $\|Ax\|_\infty \leq \sqrt{4n\ln n}$ with probability at least $1 - 2/n$ as desired.

3.3 Introduction to Hashing

Now, we start talking about applications of randomization and probability in real-world problems. In the next couple of lectures we talk about Hashing.

Hashing is a technique of mapping the input data (images, vectors etc.) of arbitrary size to a finite set of hash values using a suitable hash function. Usually a special data structure called hash table is created to store the hash values, which makes data search, insertion and deletion faster. Suppose we have a set of large images (say each of size 1 MB) and we want to store them.

Since each image has $1000,000$ bits, we assume that we have a universe of numbers $U = \{1, 2, 3, \ldots, 2^{1000000}\}$ of all possible images. Say we want to store our images in a table of size N. Ideally we want $N \ll |U|$. So, we need a function $h : U \to \{1, 2, \ldots N\}$. Usually h is called a hash function. The question that we want to study is how to choose h.

The choice of the hash function may depend on the nature of the input data and their distribution. An ideal hash function should have the property that the probability that two or more input samples getting mapped to the same hash value is low, that is it should be almost injective. If two or more samples map to the same hash value, we store them in a linked list whose address is stored at the location of the hash value in the hash table. So in order to avoid collision, we create a hash table such that each entry in the table is a linked list. So, ideally we want the length of the largest list to be as small as possible to minimize the time to query a given image.

At first, one might suggest a function that maps each image $h(X_i) = i \bmod N$. But, for such a function if all of our images have the same reminder modulo N, then they all map to the same location of the hash-table and the hashing is useless. In general, for a fixed hash function, we cannot expect to prove any worst case guarantee. So, instead we choose our hash function h from a family of functions $\mathcal{H}$ and we show that a random function chosen from $\mathcal{H}$ has a small number of collisions.

So, the question is how should we choose $\mathcal{H}$. Ideally, we want to choose $\mathcal{H}$ such that a random function maps each image to a uniformly and independently chosen location of the hash-table. Let us formalize this. We say a family $\mathcal{H}$ is 1-wise independent if for any image $X_1 \in U$ and any number $a_1 \in [N]$,

$$\mathbb{P}_{h\sim\mathcal{H}}\left[h(X_1) = a_1\right] = \frac{1}{N}$$

Here the probability is over a uniformly random function h chosen from $\mathcal{H}$.

We say $\mathcal{H}$ is 2-wise independent if for any pair of images X_1, X_2 and any pair of numbers $a_1, a_2 \in [N]$,

$$\mathbb{P}_{h\sim\mathcal{H}}\left[h(X_1) = a_1, h(X_2) = a_2\right] = \frac{1}{N^2}$$

The ideal case is if for any sequence of numbers $a_1, a_2, \ldots, a_{|U|}$ (with repetition) chosen from $[N]$,

$$\mathbb{P}\left[h(1) = a_1, \ldots, h(|U|) = a_{|U|}\right] = \frac{1}{N^{|U|}}$$

We shall study hashing and universal hash functions in more detail in the next lecture.