CS 388R: Randomized Algorithms Fall 2017

Lecture 2 — Sep 5, 2017

Prof. Eric Price *Scribe: V. Orestis Papadigenopoulos and Patrick Rall*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In the last lecture we got a first taste of randomized algorithms, from definition to conception. We demonstrated the main reasons why randomization is useful and presented different types of such algorithms, as well as two examples, QuickSort and Karger' s Min-Cut.

In this lecture we make a brief introduction to several fundamental concepts of probabilistic analysis as well as some basic and extremely useful concentration inequalities. Moreover, we discuss the basic randomized complexity classes, as well as some basic relations among them and among other classes of the complexity zoo [**?**].

2 Probability Basics and Fundamental Inequalities

In this section, we present some basic concepts on the probabilistic analysis, as well as some concentration inequalities that are going to be useful in the reasoning of algorithms during this course.

2.1 Motivation

Say we flip an unbiased coin 1000 times. How many heads? Sure, probably 500. But then is it more likely to *not* get exactly 500? So around 500 I guess? How close to 500? If I get 600 is that a surprise?

The number of heads X is given by the binomial distribution:

$$X \sim B(1000, 1/2) \qquad \mathbb{P}[X = x] = \binom{1000}{x} 2^{-1000}$$

But that gives me no intuition! Let's explore some tools to estimate the probabilities.

2.2 Central Limit Theorem

Let $X_1 ... X_n$ be independent, identically distributed (i.i.d.) random variables. Define $X = \sum_{i=1}^{n} X_i$. Then as $n \to +\infty$, we have $X \sim N(n\,\mathbb{E}(X_i), n\mathrm{Var}(X_i))$. In other words X approaches the normal distribution with n times the mean and n times the variance. This implies that the average of n

i.i.d. random variables approaches $X \sim N(\mathbb{E}(X_i), \frac{\mathrm{Var}(X_i)}{n})$, a fact that explains our intuition that by averaging samples from the same distribution, we become more and more confident about its actual mean value.

Let's apply this to our coins. By definition of the variance, we have $\mathrm{Var}(X_i) = \mathbb{E}((X_i - \mathbb{E}(X_i))^2) = 1/4$. Therefore, for $n = 1000$ we have $\sigma = \sqrt{\mathrm{Var}\sum_i X_i} = \sqrt{\sum_i \mathrm{Var}X_i} = \sqrt{n/4} = \sqrt{250} \approx 16$.

We can expect to be within 2σ (95% chance), so we probably get 470 to 530 heads. 600 is a big surprise!

Unfortunately, even though the Central Limit Theorem is a result of great importance in probability theory, it does not appear to be particularly useful in theoretical computer science and randomized analysis where we care about quantitative results and finite number of elements. In the following, we present several tail inequalities that appear to be extremely useful in our context.

2.3 Chebyshev's inequality

The first and simpler concentration bound we present is Markov's inequality. Even though this trivial bound appears to be very loose in many applications, it is useful for proving other, more powerful inequalities.

Markov's inequality: Let X be a non-negative random variable. It is the case that: $\mathbb{E}(X) \geq t\,\mathbb{P}[y \geq t]$, therefore:

$$\mathbb{P}[y \geq t] \leq \frac{\mathbb{E}(X)}{t}$$

Now, we can use Markov's inequality to prove a way more useful bound:

Chebyshev's inequality: Let X a random variable of variance σ^2 that can now take negative values. It is the case that:

$$\mathbb{P}\left[|X - \mathbb{E}(X)| > t\sigma\right] \leq \frac{1}{t^2}$$

Proof. We would like to bound the probability that X away from its expected value more than t times its standard deviation , that is: $\mathbb{P}[|X - \mathbb{E}(X)| > t\sigma]$. Since $|X - \mathbb{E}(X)|$ is a non-negative random variable, we can show using Markov's that: $\mathbb{P}[|X - \mathbb{E}(X)| > t\sigma] = \mathbb{P}[(X - \mathbb{E}(X))^2 > t^2\sigma^2] \leq \frac{\mathbb{E}((X-\mathbb{E}(X))^2)}{t^2\sigma^2} = \frac{1}{t^2}$ □

Back to our coin problem, using Chebyshev's inequality for n coins such that $\mathbb{E}(X) = np$ and $\mathrm{Var}(X) = \frac{n}{4}$ we have:

$$\mathbb{P}\left[|X - \mathbb{E}(X)| > t\sigma\right] \leq \frac{1}{t^2}$$

In that case, for $n = 1000$ coins, the probability of $|X - 500| > 2\sigma$ is less than $\frac{1}{4}$. In other words, we expect with probability greater than $3/4$, the number of heads to be between $[468, 516]$.

2.4 Chernoff's inequalities

Chebyshev's inequality works well for a constant chance: $1/n$. But we really want that Gaussian e^{-x^2} fall off! Can we do better?

Let $X = \sum_{i=1}^{n} X_i$, where X_i are independent and $x_i \in [0,1]$ for all i. In this case, it holds that:

$$\mathbb{P}\left[X \geq \mathbb{E}(X) + t\right] \leq e^{-\frac{2t^2}{n}}$$
$$\mathbb{P}\left[X \leq \mathbb{E}(X) + t\right] \leq e^{-\frac{2t^2}{n}}$$

Moreover, the multiplicative versions of the above inequalities can be proved useful:

$$\mathbb{P}\left[X \geq (1+\epsilon)\,\mathbb{E}(X)\right] \leq e^{-\frac{\epsilon^2}{2+\epsilon}\mu}$$
$$\mathbb{P}\left[X \leq (1-\epsilon)\,\mathbb{E}(X)\right] \leq e^{-\frac{\epsilon^2}{2}\mu}$$

Back to our coin experiment, we would like to bound using Chernoff the following probability:

$$\mathbb{P}\left[X - 500 \geq 2\sigma\right] \leq e^{-2}$$

Say we toss 10^6 coins with probability of heads 10^{-3}. If $X = \sum_{i=1}^{n} X_i$, we have $\mathbb{E}(X) = 1000$. Then:

$$\mathbb{P}[\text{negative number of heads}] = \mathbb{P}[X \leq \mathbb{E}(X) - 1000] \leq e^{-2} \approx 13\%$$

So Chernoff's inequality is the most useful when $E(X_i) \approx 1/2$, since it gives the best bound so far. At a first glance, it seems weird the fact that on the last case study where the probability of heads is really small the bound stays the same. The reason why this happens is that by definition, Chernoff bounds make no use of the variance of the distribution of each individual sample but only the fact that the value of every sample lays within $[0,1]$.

2.5 Application: Amplification

Suppose we are given an algorithm to estimate some value V that outputs x_i such that $x_i = V$ with probability $\frac{2}{3}$. How can we learn V with $1-\delta$ probability?

Suppose we repeat the algorithm k times and let $x_1, \cdots, x_k$ the outputs of our trials. If we order them in a non-decreasing way and take the median then the probability that the median is not the correct answer implies that more than half of the k trials gave the wrong answer. Therefore, if we set $y_i = \mathbf{1}[x_i = V]$, the event that the i-th sample returned the correct answer, then we know for sure that $y_i \in [0,1]$ (actually we know the stronger $y_i \in \{0,1\}$). Moreover, we know that $\mathbb{E}[y_i] = \mathbb{P}[y_i] \geq \frac{2}{3}$. Therefore, using the additive Chernoff bound, we can see that:

$$\mathbb{P}\left[\sum_{i=1}^{k} y_i \leq \frac{k}{2}\right] \leq \mathbb{P}\left[\sum_{i=1}^{k} y_i \leq \mathbb{E}(\sum_{i=1}^{k} y_i) - \frac{k}{6}\right] \leq e^{-\frac{2(k/6)^2}{k}} = e^{-\frac{k}{18}}$$

Therefore, for $k \geq 18\log(\frac{1}{\delta})$ we learn V with probability $1-\delta$.

2.6 Application 2: Estimating a biased coin

Say we have a coin with unknown bias p. Let's estimate p using $\hat{p} = \frac{1}{n}\sum_i X_i$. How large n do we need to achieve $|\hat{p} - p| \leq \epsilon$ with probability $1 - \delta$?

We are wrong, i.e. $|\hat{p} - p| \geq \epsilon$, iff $|\sum X_i - np| \geq n\epsilon$. Using $\sigma = \sqrt{1/\epsilon^2}$, the Chernoff bound tells us our failure probability is less than $2e^{-2(\epsilon n)^2/n} = \delta$. (The factor of 2 comes from the fact that we have to deal with both inequalities above.) Thus we need $n \geq \frac{1}{2\epsilon^2}\log\frac{2}{\delta}$.

What about multiplicative error? Suppose now that we would like to find a $\hat{p}$, such that $\hat{p} \in [1-\epsilon, 1+\epsilon]p$. Using the multiplicative version of the Chernoff bound to bound $\mathbb{P}[|\hat{p} - p| > \epsilon p]$, we get a bound of the form $\mathcal{O}(\frac{1}{p}\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$. Note that the sample complexity of this bound includes a factor of $\frac{1}{p}$, compared to that of the additive estimation. As a result, for small enough p, the sample complexity may increase dramatically! Intuitively, this makes complete sense given the fact that for really small p, the multiplicative error, even for relatively large ϵ depends also on p, a fact that translates into a small additive error.

More information about concentration inequalities and their applications can be found in [**?**, **?**].

3 Randomized Complexity Classes

Yes-no questions can be expressed as a *language* L: the set of all inputs x for which the answer is yes. A complexity class is a set of languages, usually defined by the amount of resources given to decide membership: $x \in L$ or $x \notin L$.

If an algorithm A claims that $x \in L$ we say "A accepts x", or vice-versa "A rejects x". An algorithm *decides* a language L if it *accepts* iff $x \in L$, and *rejects* iff $x \notin L$. Here are some relevant classes:

- **P** "Deterministic Polynomial Time". We have $L \in \mathbf{P}$ iff there exists a polynomial-time algorithm that decides L.
- **NP** "Nondeterministic Polynomial-Time". We have $L \in \mathbf{NP}$ iff for every input $x \in L$ there exists some advice string y such that a polynomial-time algorithm can accept x if $x \in L$ given the advice y.

Randomized algorithms follow a similar advice string structure, except that the advice string is random. Think of it as a random seed. In **NP** there *exists* an advice string that can help you decide yes-instances. For randomized algorithms we would ideally like *most* advice strings to work.

ZPP "Zero-error probabilistic Polynomial-Time". $L \in \mathbf{ZPP}$ iff there exists an algorithm that decides L, and the *expected value* of its running time is polynomial. Note that this class describes the so-called *Las Vegas* algorithms.

RP "Randomized polynomial time". This class has tolerance for one-sided error, that is, $L \in \mathbf{RP}$ iff there exists a polynomial-time algorithm A such that:
If $x \in L$ then A accepts with probability $\geq 1/2$.
If $x \notin L$ then A rejects.

In Lecture 1 we gave a randomized algorithm for MIN-CUT, which shows that MIN-CUT $\in$ **RP**. In this case we can define $G \in$ MIN-CUT if G is a graph with min-cut $\leq c$, where c is some constant. The algorithm we gave computed an upper bound c' for the input's min-cut, so if we say $c' \leq c$ accepts and $c' > c$ rejects, then the algorithm sometimes rejects yes-inputs but never accepts no-inputs.

PP "Probabilistic Polynomial". $L \in$ **PP** iff there exists a polynomial-time algorithm A s.t.:
If $x \in L$ then A accepts with probability $\geq 1/2$.
If $x \notin L$ then A rejects with probability $\geq 1/2$.

This class is huge. More specifically: **NP** $\subseteq$ **PP**. (*Proof.* Guess a random advice string and check it. If it works, accept. If it fails, toss a coin to accept or reject. The tiny bias introduced by possibly guessing the advice string satisfies both inequalities. QED.) **PP** algorithms are useless in the real world because the success probability cannot be amplified with a reasonable number of repetitions.

BPP "Bounded Probabilistic Polynomial". $L \in$ **BPP** iff $\exists$ a poly-time algorithm A s.t.:
If $x \in L$ then A accepts with probability $\geq 2/3$.
If $x \notin L$ then A rejects with probability $\geq 1/3$.

Actually the numbers 2/3 and 1/3 are arbitrary provided they are not equal. If there is a constant gap between the accept and reject probabilities, then we can amplify the probability as much as we like, just by repeating the algorithm and taking the majority-vote. **BPP** is probably the best complexity class to reflect the power of randomized algorithms in the real world.

What do we know?

In general, we know very little about how complexity classes relate. In this case, we do know that:

$$\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{NP} \subseteq \mathbf{PP}$$

What else can we say about **BPP** other than that **BPP** $\subseteq$ **NP**? The class **P/poly** is similar to **P** but algorithms are provided a polynomial-size advice string that cannot depend on the instance x, but only on the language and the size of the instance n.

Adelman's theorem: BPP $\subseteq$ P/poly.

Proof. Say we have some $L \in$ **BPP**. Take the **BPP** algorithm and amplify until the failure probability is $\leq 1/2^{n+1}$. We will need $\sim \log(2^{n+1}) \sim n+1$ repetitions, so the algorithm still runs in polynomial time.

Say we use y as our advice string, i.e. random seed. For a fixed input x, less than $1/2^{n+1}$ of all possible y cause the algorithm to be incorrect. There exist 2^n possible inputs x of length n. Thus less than $2^n/2^{n+1} = 1/2$ possible y cause *some* x to fail. Thus there *exist* advice strings for which the algorithm always succeeds! Thus $L \in$ **P/poly**. □

A good source for further information on structural (randomized) complexity theory is the book of Arora and Barak: *Computational Complexity: A Modern Approach* [**?**]

References

[CZ] Complexity Zoo `https://complexityzoo.uwaterloo.ca/Complexity_Zoo`

[MU05] Michael Mitzenmacher and Eli Upfal *Probability and Computing: Randomized Algorithms and Probabilistic Analysis* Cambridge University Press, New York, 2005.

[MR95] Rajeev Motwani and Prabhakar Raghavan *Randomized Algorithms* Cambridge University Press, New York, 1995.

[AB09] Sanjeev Arora and Boaz Barak *Computational Complexity: A Modern Approach* Cambridge University Press, New York, 2009.