CS 388R: Randomized Algorithms Fall 2015

Lecture 2 — 31 August 2014

Prof. Eric Price *Scribe: Sid Kapur, Neil Vyas*

1 Overview

In this lecture we will first introduce some probability machinery: linearity of expectation, the central limit theorem, and the discrete-setting Chernoff bound.

We will then apply these tools to some simple problems: repeated Bernoulli trials and the coupon collecting problem.

Finally, we will discuss how randomized algorithms fit in to the landscape of complexity theory.

2 Food for Thought

1. Suppose you flip an unbiased coin 1000 times. How surprised would you be if we observed
   - 500 heads?
   - 510 heads?
   - 600 heads?
   - 1000 heads?
2. Suppose you have a biased coin that lands heads with an unknown probability p. How many flips will it take to learn p to an error of $\pm\epsilon$ with probability $1 - \delta$?

3 Probability Background

Theorem 1. ***Linearity of expectation.***

If $X_1, \ldots, X_n$ are random variables, then

$$\mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{E}[X_i]$$

Example: Suppose $X_1, \ldots, X_n$ are Bernoulli trials with $p = 0.5$. Let $X = \sum_{i=1}^{n} E_i$. Then

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \frac{n}{2}$$

Definition 2. *The variance of a random variable X, denoted* $\mathbf{Var}[X]$*, is defined as*

$$\mathbf{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$

Observation 3. *Note that if Y_1 and Y_2 are independent random variables with $\mathbb{E}[Y_1] = \mathbb{E}[Y_2] = 0$, then*

$$\begin{aligned}
\mathbf{Var}[Y_1 + Y_2] &= \mathbb{E}\left[(Y_1 + Y_2)^2\right] \\
&= \mathbb{E}\left[Y_1^2 + Y_2^2 + 2Y_1Y_2\right] \\
&= \mathbb{E}\left[Y_1^2\right] + \mathbb{E}\left[Y_2^2\right] + 2\,\mathbb{E}[Y_1]\,\mathbb{E}[Y_2] \qquad \textit{(since } Y_1 \textit{ and } Y_2 \textit{ are independent)} \\
&= \mathbb{E}\left[Y_1^2\right] + \mathbb{E}\left[Y_2^2\right] \\
&= \mathbf{Var}[Y_1] + \mathbf{Var}[Y_2]
\end{aligned}$$

Claim 4. *We claim (without proof) that if Y_1 and Y_2 are independent, then $\mathbf{Var}[Y_1 + Y_2] = \mathbf{Var}[Y_1] + \mathbf{Var}[Y_2]$ (i.e. $\mathbb{E}[Y_1]$ and $\mathbb{E}[Y_2]$ need not be zero).*

Example: Suppose $X_1, \ldots, X_n$ are independent Bernoulli trials with $p = 0.5$. Let $X = \sum_{i=1}^{n} E_i$. Then

$$\begin{aligned}
\mathbf{Var}[X] &= \sum_{i=1}^{n} \mathbf{Var}[X_i] \qquad \text{(by Claim ??)} \\
&= n\left[\frac{1}{2}\left(\frac{1}{2}\right)^2 + \frac{1}{2}\left(\frac{1}{2}\right)^2\right] \\
&= \frac{n}{4}
\end{aligned}$$

If $n = 1000$, then $\mathbf{Var}[X] = 250$ and the standard deviation of $X = \sqrt{\mathbf{Var}[X]} \approx 16$.

Theorem 5. ***The central limit theorem*** *Suppose $X_1, \ldots, X_n$ are independent identically distributed random variables with expected value μ and variance σ^2.*

Then as $n \to \infty$, the sample average $S_n = \frac{X_1 + \cdots + X_n}{n}$ converges to the Gaussian

$$N\left(\mu, \frac{\sigma^2}{n}\right)$$

Some statements of the central limit theorem give bounds on convergence, but unfortunately these bounds are not tight enough to be useful in this example.

By the central limit theorem, as the number of coin tosses goes to infinity, our binomial distribution of coin tosses converges to a Gaussian $N(\mu, \sigma^2)$ where $\mu = \frac{n}{2}$ and $\sigma^2 = \frac{n}{4}$, as we calculated above.

4 Bounding the Sum of Bernoulli Trials

Let's return to our coin-tossing problem.

Can we find an upper bound on the probability that the number of coin tosses exceeds a certain number?

Claim 6. *If Z is drawn from a Gaussian with mean μ and variance σ^2, then*

$$\mathbb{P}[Z \geq \mu + t] \leq e^{-\frac{t^2}{2\sigma^2}} \quad \forall t \geq 0$$
$$\mathbb{P}[Z \leq \mu - t] \leq e^{-\frac{t^2}{2\sigma^2}} \quad \forall t \geq 0$$

(We omit the proof.)

If our distribution were Gaussian (or if we were willing to invoke the central limit theorem), we could use the bound from Claim **??**:

$$\mathbb{P}[X \geq 500 + \sigma t] \leq e^{-\frac{t^2}{2}}$$
$$\Rightarrow \quad \mathbb{P}[X \geq 500 + 16t] \leq e^{-\frac{t^2}{2}}$$
$$\Rightarrow \quad \mathbb{P}[X \geq 600] \leq e^{-18}$$

But unfortunately, we have a binomial distribution, not a Gaussian, and we don't want to use the central limit theorem. What should we do?

Definition 7. ***Chernoff bound (for the discrete setting)*** *Let $X = \sum_{i=1}^{n} X_i$, where the X_i take on values in $[0, 1]$ and are independent. (The X_i can take on any value between 0 and 1. Also, the X_i need not be identical.) Then*

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

If we use the Chernoff bound, we get:

$$\begin{aligned}
\mathbb{P}[X \geq 500 + \sigma t] &\leq e^{-\frac{2t^2\sigma^2}{n}} \\
&= e^{-\frac{2t^2(n/4)}{n}} && (\text{since } \sigma^2 = \frac{n}{4}) \\
&= e^{-\frac{t^2}{2}}
\end{aligned}$$

Note that this is the same bound that we got from the Gaussian! So the Chernoff bound is basically tight in this case.

4.1 Evaluating the Chernoff Bound

There are a few limitations of the Chernoff bound in this situation.

The main limitation is that it does not depend on the variance of the X_i. Suppose each X_i has a very low probability, say,

$$\mathbb{P}[X_i] = \frac{1}{10\sqrt{n}}$$

Then

$$\mathbb{E}[X] = \frac{1}{10\sqrt{n}} \cdot n = \frac{\sqrt{n}}{10}$$

Then the Chernoff bound would give us $1-\delta$ confidence interval of

$$\mathbb{P}[X \geq \mathbb{E}[X] + t] \leq e^{-\frac{2t^2}{n}} = \frac{\delta}{2}$$

$$\Rightarrow \quad t = \sqrt{\frac{n \log(2/\delta)}{2}}$$

So the Chernoff bound gives us $X \in \mathbb{E}[X] \pm \sqrt{\frac{n \log(2/\delta)}{2}}$ with probability $1-\delta$.

If we pick $\delta = 0.10$, then the bound is $X \in \mathbb{E}[X] \pm \sqrt{10n}$. Note that the width of our confidence interval is about 60 times the expected value!

We know that the bound can be made much tighter here because the variance of the X_i is small. The Chernoff bound doesn't take variance into account. Later in this class, we will encounter Bernstein-type inequalities, which do account for the variance.

Also, note that the Chernoff bound can be less than 0 or greater than n, which does not make sense in this problem.

4.2 Second Food for Thought

Suppose you have a biased coin that lands heads with an unknown probability p. How many flips will it take to learn p to an error of $\pm\epsilon$ with probability $1-\delta$?

We need to find n such that

$$\mathbb{P}[X \geq (p+\epsilon)n] \leq \frac{\delta}{2}$$

By the Chernoff bound,

$$\begin{aligned}\mathbb{P}[X \geq (p+\epsilon)n] &\leq e^{-\frac{2(\epsilon n)^2}{n}} \\ &= e^{-2\epsilon^2 n}\end{aligned}$$

So we must pick n such that

$$e^{-2\epsilon^2 n} \leq \frac{\delta}{2}$$

i.e.

$$n \geq \frac{1}{2\epsilon^2} \log\left(\frac{2}{\delta}\right)$$

5 The Coupon Collecting Problem

Suppose there exist n types of coupons, with an infinite supply of each. If you collect coupons uniformly at random, what is the expected number of coupons you must collect to hold at least one of each type?

T_0 must be 1, since the next coupon we receive is the first coupon, so it must be a new type.

T_{n-1} must be n, since for each coupon we receive, there is a $\frac{1}{n}$ probability that it is the coupon that we haven't received yet.

In general, $T_{n-k} = \frac{n}{k}$ because there are k coupon types remaining, so at each step the probability that we receive a new coupon is $\frac{k}{n}$.

So

$$T = \sum T_i = \sum_{i=1}^{n} \frac{n}{k} = nH_k \leq n(\log n + 1)$$

where $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the nth harmonic number.

6 Background on Complexity Theory

Let's begin our whirlwind tour of complexity theory. We'll explore, to varying degrees, **P**, **NP**, $\mathbf{Z_{PP}}$, $\mathbf{R_P}$, $\mathbf{coR_P}$, and **BPP**.

We use as supplemental material section 1.5 from *Randomized Algorithms* by Motwani and Raghavan [**?**]. Note that formally, complexity classes are sets of languages, but we'll be looser and also speak in terms of algorithms inhabiting these classes.

6.1 P: Polynomial-time deterministic algorithms

These can be described by Turing machines and *languages*. A *language* is a set $\ell \subseteq \{0,1\}^n$. Given an input string $x \in \{0,1\}^n$, a Turing machine either accepts the string, rejects the string, or loops. The language of a Turing machine is the set of strings that the Turing machine accepts.

It is possible to phrase many non-decision problems as decision problems. For example, suppose the original problem is to find an integer n that has some property. The corresponding decision problem would be: Given an input x, is x less than or equal to n? In this setting, we can use binary search to find x.

Definition 8. *The class* **P** *consists of all languages ℓ that have a polynomial-time algorithm $\mathcal{A}$ such that for any input x,*

If $x \in \ell$, $\mathcal{A}$ accepts
If $x \notin \ell$, $\mathcal{A}$ rejects

6.2 NP: P with a good "advice" string

"Advice" given to a Turing machine is an additional input string allowed to depend on the length of the input, n, but not the input itself.

Definition 9. *A language l is in* **NP** *if there exists a polynomial-time algorithm $\mathcal{A}$ such that for any input x,*

If $x \in \ell$, there exists an advice string y such that $\mathcal{A}$ accepts the input (x, y)
If $x \notin \ell$, then for all strings y, $\mathcal{A}$ rejects the input (x, y)

For our purposes, the advice string may be arbitrarily long, but the algorithm can only inspect a polynomial-in-the-length-of-the-input amount of it.

Note the resemblance between advice strings and random bits. We will come back to this later.

6.3 $\mathbf{R_P}$ and co$\mathbf{R_P}$: One-sided error

Definition 10. *An algorithm $\mathcal{A}$ is in $\mathbf{R_P}$ if it has polynomial worst-case runtime and, for all inputs x,*

$$\text{If } x \in \ell, \text{ then } \mathbb{P}[\mathcal{A} \text{ accepts } x] \geq \frac{1}{2}$$
$$\text{If } x \notin \ell, \text{ then } \mathbb{P}[\mathcal{A} \text{ accepts } x] = 0$$

Definition 11. *An algorithm $\mathcal{A}$ is in $\mathbf{coR_P}$ if it has polynomial worst-case runtime and, for all inputs x,*

$$\text{If } x \in \ell, \text{ then } \mathbb{P}[\mathcal{A} \text{ accepts } x] = 1$$
$$\text{If } x \notin \ell, \text{ then } \mathbb{P}[\mathcal{A} \text{ accepts } x] < \frac{1}{2}$$

Note that an $\mathbf{R_P}$ algorithm corresponds to a Monte Carlo algorithm that can err only when $x \in \ell$, and, similarly, an algorithm in $\mathbf{coR_P}$ corresponds to a Monte Carlo algorithm that can err only when $x \notin \ell$ [**?**].

6.4 $\mathbf{Z_{PP}}$ Zero-error with expected polynomial time

This complexity class corresponds to Las Vegas-type randomized algorithms. Thus, $\mathbf{Z_{PP}} = \mathbf{R_P} \cap \mathbf{coR_P}$.

6.5 PP and BPP: Two-sided error

Definition 12. *An algorithm $\mathcal{A}$ is in PP (probabilistic polynomial time) if it has polynomial worst-case runtime and, for all inputs x,*

$$\text{If } x \in \ell, \mathbb{P}[\mathcal{A} \text{ accepts } x] \geq \frac{1}{2}$$
$$\text{If } x \notin \ell, \mathbb{P}[\mathcal{A} \text{ accepts } x] \leq \frac{1}{2}$$

Unfortunately, **PP** is not a very useful class. For example, $\mathbf{NP} \subseteq \mathbf{PP}$. To show this we consider SAT, the unrestricted satisfiability problem: Given $f(x_1, \ldots, x_n)$, is it satisfiable?

We propose the following algorithm:

```
function RandomizedSAT(f : {True, False}^n → {True, False}): {True, False}
    for all i ∈ {1 ... n} do
        x_i ← Uniform(True, False)
    end for
    if f(x_1, ..., x_n) = True then
        return True
    else
        b ← Uniform(True, False)
        return b
    end if
end function
```

Suppose that the input is satisfiable; then $\mathbb{P}[\text{RandomizedSAT accepts}] \geq \frac{1}{2} + \frac{1}{2^{n+1}}$. Similarly, if the input is not satisfiable, $\mathbb{P}[\text{RandomizedSAT rejects}] = \frac{1}{2}$. Thus SAT is in **PP**.

Note that since we can reduce every problem in **NP** to SAT, every **NP** is in **PP**. Thus, **PP** isn't a very useful class to reason about.

We introduce **BPP** in the hopes of remedying this situation.

Definition 13. *An algorithm $\mathcal{A}$ is in* **BPP** *(bounded probabilistic time) if, for all inputs x,*

$$\textit{If } x \in \ell,\ \mathbb{P}[\mathcal{A} \textit{ accepts } x] \geq \frac{2}{3}$$
$$\textit{If } x \notin \ell,\ \mathbb{P}[\mathcal{A} \textit{ rejects } x] \leq \frac{2}{3}$$

In the next section, we will show that we can amplify any **BPP** algorithm to arbitrary accuracy.

6.6 Amplification of BPP algorithms

Now, let's get back to that last remark; let's amplify the probability of success of a **BPP** algorithm.

Suppose that we want a **BPP** algorithm with $\mathbb{P}[\text{accept } x \text{ if } x \in \ell] \geq 1 - \delta$ and $\mathbb{P}[\text{reject } x \text{ if } x \notin \ell] \geq 1 - \delta$. We claim that for a **BPP** algorithm, this can be accomplished with only $O(\log \frac{1}{\delta})$ time overhead.

Claim 14. ***Amplification.*** *Given a* **BPP** *algorithm for some problem that accepts correct answers with probability at least $\frac{2}{3}$, and rejects incorrect answers with probability at least $\frac{2}{3}$, there exists a* **BPP** *algorithm that accepts correct answers with probability at least $1 - \delta$ and rejects incorrect answers with probability at least $1 - \delta$, for any $0 \leq \delta < 1$.*

Proof: Consider the algorithm $\mathcal{A}'$ that runs k independent iterations of $\mathcal{A}$, and accepts if the majority of the runs accepted, and rejects if the majority of the runs rejected.

and then employ the Chernoff bound to bound the probability that the majority vote across all iterations is incorrect.

Suppose $x \in \ell$. Then $\mathbb{E}[\text{num. runs that accept } x] \geq \frac{2}{3}k$. So

$$\mathbb{P}\left[\text{num accepts } \leq \frac{2}{3}k - t\right] \leq e^{-\frac{2t^2}{k}} \qquad \text{(by Chernoff Bound)}$$

$$\Rightarrow \quad \mathbb{P}\left[\text{num accepts } \leq \frac{1}{2}k\right] \leq e^{-\frac{k}{18}} \qquad \text{(letting } t = k/6\text{)}$$

$$\Rightarrow \quad k \geq -18 \log \delta \qquad \text{(setting } e^{-\frac{k}{18}} \leq \delta\text{)}$$

So, the probability that the majority of the k runs falsely reject (and thus $\mathcal{A}'$ rejects) when $x \in \ell$ is bounded by δ if we set $k \geq -18 \log \delta$.

The proof that $\mathcal{A}_1$ rejects incorrect answers with probability at least $1 - \delta$ is similar.

So, $\mathcal{A}'$ is in **BPP** with the $1 - \delta$ bounds that we desired, if we set k appropriately.

6.7 $\mathbf{P_{poly}}$

$\mathbf{P_{poly}}$ is the set of all problems that are decidable by deterministic polynomial-time algorithms that can take in an advice string that depends on the size of the input, but not the input itself.

We'll show one relevant theoretical use of $\mathbf{P_{poly}}$ in a later section.

6.8 Containments

It is known that $\mathbf{P} \subseteq \mathbf{Z_{PP}}$, but it is not known whether $\mathbf{Z_{PP}} \subseteq \mathbf{P}$. It should also be apparent that $\mathbf{Z_{PP}} = \mathbf{R_P} \cap \mathbf{coR_P}$, as stated above.

We claim that $\mathbf{P} \subseteq \mathbf{R_P} \subseteq \mathbf{NP}$, and that $\mathbf{R_P} \subseteq \mathbf{BPP}$; here we will only prove that $\mathbf{R_P} \subseteq \mathbf{NP}$.

Claim 15. $\mathbf{R_P} \subseteq \mathbf{NP}$.

Proof:

Suppose $\mathcal{A}$ is a $\mathbf{R_P}$ algorithm for the language ℓ.

To prove that $\ell \in \mathbf{NP}$, it is sufficient to show that $\mathcal{A}$ has polynomial runtime and that:

1. For every $x \in \ell$, there exists an advice string r such that $\mathcal{A}(x, r)$ accepts
2. For every $x \notin \ell$, for all advice strings r, $\mathcal{A}(x, r)$ rejects.

(1) is true since, by the definition of $\mathbf{R_P}$, there is a random string r such that $\mathcal{A}(x, r)$ accepts.

(2) is true since, by the defintion of $\mathbf{R_P}$, there is no random string r such that $\mathcal{A}(x, r)$ accepts.

So $\ell \in \mathbf{NP}$.

It is not known whether the following statements hold:

- $\mathbf{BPP} \subset \mathbf{NP}$?

- $\mathbf{R_P} = \mathbf{Z_{PP}} = \mathbf{coR_P}$?

7 Adleman's Theorem

Theorem 16. ***Adleman's Theorem:*** $\mathbf{BPP} \subseteq \mathbf{P_{poly}}$.

Proof:

Let $\mathcal{A} \in \mathbf{BPP}$.

By Claim **??**, there exists an algorithm $\mathcal{A}' \in \mathbf{BPP}$ with $\delta = \frac{1}{2^{n+1}}$.

Let $\text{Bad}(x)$ be the set of advice strings r such that $\mathcal{A}'(x, r)$ is incorrect. So, over the advice string R,

$$\begin{aligned} P[\exists x \text{ s.t. } R \in \text{Bad}(x)] &= P\left[\bigcup_{x \in \{0,1\}^n} R \in \text{Bad}(x)\right] \\ &\leq \sum_{x \in \{0,1\}^n} P(R \in \text{Bad}(x)) \qquad \text{(union bound)} \\ &\leq 2^n \cdot \frac{1}{2^{n+1}} \\ &= \frac{1}{2} < 1 \end{aligned}$$

Thus, there is a positive probability that $\mathcal{A}'$ succeeds on every x. Thus there exists a random advice string (call it r_n) such that for all inputs x of length n, $\mathcal{A}'(x, r_n)$ gives the correct answer.

So, the algorithm $\mathcal{A}'(x, r_n)$ is a polynomial-time deterministic algorithm that, given the appropriate advice string r_n, gives the correct answer for all inputs x of length n. So $\mathcal{A}' \in \mathbf{P_{poly}}$.

So, in a sense, randomness doesn't help *that* much. Every problem in **BPP** can be solved in polynomial time without randomness (assuming we have an advice string for the $\mathbf{P_{poly}}$ algorithm). However, randomized algorithms often give us a smaller polynomial runtime.

References

[MR] Rajeev Motwani, Prabhakar Raghavan Randomized Algorithms. *Cambridge University Press*, 0-521-47465-5, 1995.