CMPUT 675: Randomized Algorithms **Fall 2005**

Lecture 3: Sept 15

Lecturer: Mohammad R. Salavatipour *Scribe: Zac Friggstad*

3.1 Random Variables and Markov's Inequality

The proofs of all of the results in this section can also be found in [1]. Consider the following experiment: we a set of n coins each of which comes up heads with probability p. The random variable of the number of coins the come up heads is a special kind of random variable kind Binomial random variable.

Definition 3.1 *A* binomial random variable X *with parameters p and n, denoted $B(n,p)$, has the property*

$$\Pr[X = i] = \begin{cases} \binom{n}{i} p^i (1-p)^{n-1} & \text{if } i = 0, 1, 2, \ldots, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 3.2 *If X is a binomial random variable with parameters p and n then $E[X] = np$.*

Proof:

$$\begin{aligned} \mathrm{E}[X] &= \sum_{j=0}^{n} j \binom{n}{j} p^j (1-p)^{n-j} \\ &= \sum_{j=0}^{n} j \frac{n!}{j!(n-j)!} p^j (1-p)^{n-j} \\ &= \sum_{j=1}^{n} \frac{n!}{(j-1)!(n-j)!} p^j (1-p)^{n-j} \\ &= np \sum_{j=1}^{n} \frac{(n-1)!}{(j-1)!((n-1)-(j-1))!} p^{j-1} (1-p)^{(n-1)-(j-1)} \\ &= np \sum_{k=0}^{n-1} \frac{(n-1)!}{k!((n-1)-k)!} p^k (1-p)^{(n-1)-k} \\ &= np \sum_{k=0}^{n-1} \binom{n-1}{k} p^k (1-p)^{(n-1)-k} \\ &= np(p + (1-p))^{n-1} \\ &= np \end{aligned}$$

Where the second last equality follows from the binomial theorem

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

for positive integers n. ■

Now consider the following experiment: we flip a coin comes up heads with probability p until the first time we see heads. The random variable of the number of coin flips before the first heads is a special kind of random variable kind geometric random variable.

Definition 3.3 *A* geometric random variable X *with parameter* p *has the property*

$$\Pr[X = j] = \begin{cases} (1-p)^{(i-1)}p & \text{if } i = 1, 2, \ldots, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 3.4 *For* X *a geometric random variable with parameter* p*,*

$$\mathrm{E}[X] = \sum_{i=1}^{\infty} \Pr[X \geq i].$$

Proof:

$$\begin{aligned} \sum_{i=1}^{\infty} \Pr[X \geq i] &= \sum_{i=1}^{\infty} \sum_{j=i}^{\infty} \Pr[X = j] \\ &= \sum_{j=1}^{\infty} \sum_{i=1}^{j} \Pr[X = j] \\ &= \sum_{j=1}^{\infty} j \cdot \Pr[X = j] \\ &= \mathrm{E}[X]. \end{aligned}$$

The second equality is justified by noticing that each of $Pr[X = j]$ appears exactly j times as a term in the second expression. ■

Lemma 3.5 *For* X *a geometric random variable with parameter* p*,*

$$\mathrm{E}[X] = \frac{1}{p}.$$

Proof:

$$\begin{aligned} Pr[X \geq i] &= \sum_{n=i}^{\infty} (1-p)^{n-1} p \\ &= \sum_{n=i-1}^{\infty} (1-p)^{n} p \\ &= p \sum_{n=0}^{\infty} (1-p)^{n} - p \sum_{n=0}^{i-2} (1-p)^{n} \\ &= p \frac{1}{1-(1-p)} - p \frac{(1-p)^{i-1} - 1}{(1-p) - 1} \\ &= (1-p)^{i-1} \end{aligned} \tag{3.1}$$

by summing both an infinite and a finite geometric series. So

$$\begin{aligned}
E[X] &= \sum_{i=1}^{\infty} \Pr[X \geq i] \\
&= \sum_{i=1}^{\infty} (1-p)^{i-1} \\
&= \sum_{i=0}^{\infty} (1-p)^{i} \\
&= \frac{1}{1-(1-p)} \\
&= \frac{1}{p}
\end{aligned}$$

by summing another infinite geometric series. ■

3.2 Deviation Bounds

Just saying that the expected value of something is X is often not satisfactory. We would like to bound the probability that we are far from the expected value. For this there are a variety of tools. Perhaps the most basic (and so the weakest) one is Markov's inequality.

Theorem 3.6 (Markov's Inequality) *Let X be a random variable that only assumes non-negative values. Then, for all $a > 0$,*

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

Proof: Define I as

$$I = \begin{cases} 1 & \text{if } X \geq a, \\ 0 & \text{otherwise.} \end{cases}$$

Since $X \geq 0$, then

$$I \leq \frac{X}{a}.$$

Also,

$$E[I] = 1 \cdot Pr[I = 1] + 0 \cdot Pr[I = 0] = Pr[X \geq a].$$

So,

$$Pr[X \geq a] = E[I] \leq E\left[\frac{X}{a}\right] = \frac{E[X]}{a}.$$

■

Another form of Markov's Inequality is

$$Pr[X \geq aE[X]] \leq \frac{1}{a}$$

where a and X are as in the above theorem.

3.3 Complexity Classes

Definition 3.7 *A* language $L \subseteq \Sigma^*$ *is a set of finite strings over some fixed (finite) alphabet* Σ *where* Σ^* *is the set of all finite words that can be formed with letters from* Σ*. An algorithm is said to* decide *a language* L *when it accepts a word* x *if and only if* $x \in L$*. If* $x \notin L$ *then the algorithm is said to reject* x*.*

Some of the familiar complexity classes are presented first.

Definition 3.8 *The class* ***P*** *is the set of all languages* L *such that there is a deterministic polynomial time algorithm that decides* L*.*

Definition 3.9 *The class* ***NP*** *consists of all languages* L *that have a witness that can be regognized by a deterministic polynomial time algorithm. Formally, for a language* $L \in$ ***NP*** *we have* $x \in L$ *if and only if there is a* y *such that* $|y| \leq c|x|^d$ *for constants* c, d *and* $P(x, y)$ *where* P *is a polynomial time computable predicate.*

We now discuss randomized complexity classes.

Definition 3.10 *The class* ***RP*** *(Randomized Polynomial time) is the class of all languages* L *that have a polynomial time algorithm* A *such that*

$$x \in L \Rightarrow \Pr[A \text{ accepts } x] \geq \frac{1}{2}$$

$$x \notin L \Rightarrow A \text{ always rejects } x.$$

Such an algorithm A is a Monte Carlo with one-sided error. It is important to note that the $\frac{1}{2}$ acceptance probability is not important since any constant, non-zero acceptance probability can be inflated arbitrarily close to 1 with a constant number of repetitions of the algorithm.

Definition 3.11 *The class* ***co-RP*** *consists of all languages* L *that have a polynomial time algorithm* A *such that*

$$x \in L \Rightarrow A \text{ always accepts } x$$

$$x \notin L \Rightarrow \Pr[A \text{ rejects } x] \geq \frac{1}{2}.$$

Definition 3.12 *The class* ***ZPP*** *(Zero-error Probabilistic Polynomial time) is the class of all languages* L *that have a Las Vegas algorithm running in expected polynomial time.*

Theorem 3.13 ***ZPP*** $=$ ***RP*** $\cap$ ***co-RP***

Proof: If $L \in$ **RP** $\cap$ **co-RP** then there are algorithms A and B such that

$$x \in L \Rightarrow Pr[A \text{ accepts } x] \geq \frac{1}{2} \text{ and } B \text{ always accepts } x$$

$$x \notin L \Rightarrow A \text{ always rejects } x \text{ and } Pr[B \text{ rejects } x] \geq \frac{1}{2}.$$

Consider the following algorithm.

make_ZPP(x)
  do forever
    if $A(x) = accept$ **then return** *accept*
    if $B(x) = reject$ **then return** *reject*.

If the algorithm returns accept then $A(x) = accept$ so $x \in L$. Likewise if the algorithm returns reject then $B(x) = reject$ so $x \notin L$. Since the probability of either of these happining is $\geq \frac{1}{2}$ on a fixed input x, then the expected number of iterations is at most 2. This is the expectation of a geometric random variable where each trial has probability at least $\frac{1}{2}$. Therfore, $\mathbf{RP} \cap \mathbf{co\text{-}RP} \subseteq \mathbf{ZPP}$.

Finally, if $L \in \mathbf{ZPP}$ then there is an algorithm M running in expected polynomial time $p(n)$ on an input of size n that always returns the correct answer. Consider the following algorihm.

make_RP(x)
  run the algorithm M on x for $2p(|x|)$ steps
  if M halted on x **then** return the computed result $M(x)$
  otherwise return *reject*

If $x \notin L$ then this algorithm will always reject. If $x \in L$ then this algorithm will accept if and only if $M(x)$ halts in at most $2p(|x|)$ steps. Let Y be the random variable that is the execution time of $M(x)$. By the Markov inequality, $\Pr[Y \geq 2p(|x|)] \leq \frac{1}{2}$ so the algorithm accepts x with probability at least $\frac{1}{2}$. Since simulating an algorithm can be done with polynomial time overhead then the algorithm runs in polynomial time. Therefore $L \in \mathbf{RP}$ so $\mathbf{ZPP} \subseteq \mathbf{RP}$. $\mathbf{ZPP} \subseteq \mathbf{co\text{-}RP}$ can be shown by similar arguments which proves $\mathbf{ZPP} \subseteq \mathbf{RP} \cap \mathbf{co\text{-}RP}$. Equality follows from both inclusions. ■

Definition 3.14 *The class **BPP** (Bounded error Probabilistic Polynomial time) consists of all languages L that have a polynomial time algorithm A such that*

$$x \in L \Rightarrow \Pr[A \text{ accepts } x] \geq \frac{3}{4}$$

$$x \notin L \Rightarrow \Pr[A \text{ rejects } x] \geq \frac{3}{4}.$$

Some important open questions involving complexity classes are the answers to the predicates $\mathbf{BPP} \subseteq \mathbf{NP}$ and $\mathbf{NP} \subseteq \mathbf{BPP}$. Note that if the latter is true then all problems solved by randomized polynomial time algorithms can be solved by deterministic polynomial time algorithms. It is known that if $\mathbf{NP} \subseteq \mathbf{BPP}$ then $\mathbf{NP} = \mathbf{RP}$.

3.4 Coupon Collector's Problem

The *Coupon Collector's Problem* deals with the problem of collecting all of n types of coupons. The collector picks a random coupon uniformly from all of the n coupons and repeats this process until all types have been collected at least once. We would like to know the expected number of selections before the collector gets all n types coupons.

Formally, let X be the number of trials to collect all coupon types. The question is to determine the value of $\mathrm{E}[X]$. To simplify the problem we break up the selection process and only look at the expected number of picks to get another coupon when i different types of coupons are already collected. For each $i = 0, 1, ..., n-1$

this will be denoted by the random variable X_i. By linearity of expectation we have

$$\mathrm{E}[X] = \sum_{i=0}^{n-1} \mathrm{E}[X_i] = \mathrm{E}[\sum_{i=0}^{n-1} X_i].$$

The probability of drawing a different type of coupon from n coupons when already holding i types of coupons is

$$p_i = \frac{n-i}{n}$$

for one draw. Notice that X_i is a geometric random variable with parameter p_i so

$$\mathrm{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i}.$$

Therefore, the total expected number of trials is

$$\mathrm{E}[X] = \sum_{i=0}^{n-1} \frac{n}{n-i} = n\sum_{k=1}^{n} \frac{1}{k} = nH_n = n \ln n + \Theta(n).$$

The deviation from this expected value is also interesting to determine. The probability that a fixed coupon, say c_i, is not picked in a single trial is $1 - \frac{1}{n}$. Thus the probability that c_i is not picked after r trials is

$$\left(1 - \frac{1}{n}\right)^r \leq e^{-\frac{r}{n}}.$$

If, for some constant k, we pick $kn \ln n$ coupons uniformly at random then the probability that we don't collect coupon c_i is bound from above by

$$e^{-\frac{kn \ln n}{n}} = e^{-k \ln n} = n^{-k}.$$

Recalling that the union bound states that for any countable sequence of events $E_1, E_2, \ldots$,

$$\Pr\left(\bigcup_{i \geq 1} E_i\right) \leq \sum_{i \geq 1} \Pr(E_i)$$

we have the probability that there is some coupon that is not picked in $kn \ln n$ trials is upper bound by $n \cdot n^{-k} = n^{1-k}$. This probability can be made arbitrarily small by selecting larger values of k. We say in this case that "with high probability" (w.h.p.) that the number of trials to collect all coupons is $O(n \ln n)$.

References

1 M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge Univeristy Press, Cambridge, England, 2005.