CS271 Randomness & Computation **Spring 2020**

Lecture 10: February 20

Instructor: Alistair Sinclair

10.1 Unbiased Estimators

Suppose we are given two sets R and A in the plane such that $R \subset A$, and we wish to estimate the area of R, given that of A (or compute the size of R given that of A if R and A are finite point sets). One approach is the Monte-Carlo method: pick an element of A uniformly at random and check to see if this element is in R. Repeat this procedure several times (independently) to estimate the size of R relative to that of A.

Let $p_1, p_2, \ldots, p_t$ be t *i.i.d* elements sampled uniformly from A. Define

$$X_i = \begin{cases} 1 & \text{if } p_i \in R; \\ 0 & \text{otherwise,} \end{cases}$$

and the sample mean

$$X = \frac{1}{t} \sum_{i=1}^{t} X_i.$$

Then $\mathrm{E}[X_i] = \frac{|R|}{|A|}$, so each X_i estimates the value of $\frac{|R|}{|A|}$. We call X_i an *unbiased estimator* of $\frac{|R|}{|A|}$. Also, $\mathrm{E}[X] = \mathrm{E}[X_i] = \frac{|R|}{|A|}$, so X is also unbiased and we can output the value $X|A|$ to get an estimate of the desired quantity $|R|$. Note that the variance of X decreases with the number of trials t: specifically, $\mathrm{Var}[X] = \frac{1}{t}\mathrm{Var}[X_i]$. So, of course, we would expect the quality of our estimate to improve as the sample size t increases.

A well known classical application of this idea is *Buffon's Needle*. Given a number of parallel lines at unit distance apart, what is the probability that a randomly dropped needle of unit length will cross one of the lines? It turns out that the solution to this problem gives an estimate of π.

Suppose the needle lands at an acute angle θ to the lines. Let x be the distance from the center of the needle to the closest line. Then x is uniformly distributed between 0 and $\frac{1}{2}$, and the needle crosses some line exactly when $x \leq \frac{1}{2}\sin\theta$, so for any given θ the probability that the needle crosses a line is $\sin\theta$. Integrating over θ, we arrive at

$$\Pr[\text{needle crosses a line}] = \frac{1}{\pi/2} \int_{\theta=0}^{\frac{\pi}{2}} \sin\theta \, \mathrm{d}\theta = \frac{2}{\pi}.$$

Thus the indicator r.v. of the event that the needle crosses a line is an unbiased estimator of the quantity $\frac{2}{\pi}$.

In any application of unbiased estimators, the key question is how many samples are needed in order to obtain an estimate within a specified accuracy with specified confidence. The following simple theorem quantifies this.

Theorem 10.1 *[Unbiased Estimator Theorem] Let* $\{X_i\}_{i=1}^t$ *be i.i.d with* $\mathrm{E}[X_i] = \mu$ *and* $\mathrm{Var}[X_i] = \sigma^2$*. Let* $X = (1/t)\sum_{i=1}^t X_i$*. Then, if* $t \geq \frac{4}{\epsilon^2}\frac{\sigma^2}{\mu^2}$*, we have*

$$\Pr[|X - \mu| > \epsilon\mu] \leq \frac{1}{4}.$$

In other words, X will be within ratio $(1 \pm \epsilon)$ of the true μ with probability $\frac{3}{4}$, provided we take the average of at least $\frac{4\sigma^2}{\epsilon^2\mu^2}$ trials.

Proof: By linearity of expectation, $\mathrm{E}[X] = \mu$, and by independence of the X_i's, $\mathrm{Var}[X] = \sigma^2/t$. Thus by Chebyshev's inequality,

$$\Pr[|X - \mu| \geq \epsilon\mu] \leq \frac{\mathrm{Var}[X]}{\epsilon^2\mu^2} = \frac{\sigma^2}{t\mu^2\epsilon^2}.$$

This final expression will be $\leq \frac{1}{4}$ provided $t \geq \frac{4}{\epsilon^2}\frac{\sigma^2}{\mu^2}$, as claimed. ■

Note that the required sample size is proportional to $\frac{1}{\epsilon^2}$ (which is the price we pay for increased accuracy) and to $\frac{\sigma^2}{\mu^2}$, the ratio of the variance to the square of the mean (which is the "inherent cost" of our estimator, sometimes called its "critical ratio"). In the important special case where X_i is a binary random variable (such as the area example above), we have $\sigma^2 = \mathrm{Var}[X_i] = \mathrm{E}[X_i^2] - \mathrm{E}[X_i]^2 = \mu - \mu^2$, so $\sigma^2 \leq \mu$ and the required value of t simplifies to $t \geq \frac{4}{\mu\epsilon^2}$. Thus the critical ratio in this case is essentially $\frac{1}{\mu}$. This reflects the intuitive fact that, if the event whose probability we are trying to estimate is very rare (or equivalently, the area of R is much smaller than that of A), then we will require a very large number of samples to estimate it. (We are looking for a "needle in a haystack.")

Why do we stop at bounding the error probability by $\frac{1}{4}$? It turns out that, once we have a $\frac{1}{4}$ bound on this probability, we can apply the so-called "median trick" to efficiently reduce the error probability to any desired $\delta > 0$.

Median trick: Perform t' trials as above, each trial having t experiments, and take the median of the values of X obtained from each trial. Let this median value be X'. Then, as long as $t' > 2\log_{(4/3)}\delta^{-1}$, we get $\Pr[|X' - \mu| > \epsilon\mu] \leq \delta$.

To prove this, note that we can treat each trial as a coin flip, where the result is "heads" if $|X - \mu| \leq \epsilon\mu$ and "tails" otherwise. If X' doesn't lie in the interval $[\mu(1-\epsilon), \mu(1+\epsilon)]$, then at least half of the trials must also lie outside of the interval (i.e., they all correspond to "tails" tosses). Then the median trick bounds this probability as a consequence of the following lemma:

Lemma 10.2 *If* $2s+1$ *coin flips are performed with* $\Pr[\text{Heads}] \geq 3/4$*, then* $\Pr[\leq s \text{ flips are Heads}] \leq (3/4)^s$.

The proof is left as an exercise (either by reading it off from the calculation for error reduction in BPP algorithms in Lecture 1, or as a simple application of Chernoff bounds as discussed in a later lecture).

10.2 Counting satisfying assignments of a DNF formula

Definition 10.3 *A function* $f : \Sigma^* \to \mathbb{N}$ *is in* $\#\mathcal{P}$ *if there exists a polynomial time non-deterministic Turing machine such that on every input* x*, the number of accepting computations is* $f(x)$.

Informally, $\#\mathcal{P}$ is the class of problems that are "counting" versions of problems in $\mathcal{NP}$, i.e., they are problems of the form "compute $f(x)$", where $f(x)$ is the number of accepting paths of an $\mathcal{NP}$ machine, or equivalently, "solutions" (or "witnesses") for a problem in $\mathcal{NP}$.

Definition 10.4 *A function $f \in \#\mathcal{P}$ is said to be* $\#\mathcal{P}$-complete *if every function $g \in \#\mathcal{P}$ is Turing reducible to f.*

Showing that a function is $\#\mathcal{P}$-complete makes a very strong statement about its intractability: if such a function were computable in polynomial time then the consequences would be even more dramatic than $\mathcal{P} = \mathcal{NP}$.

It is easy to see that the counting versions of $\mathcal{NP}$-complete problems are $\#\mathcal{P}$-complete. For example, #SAT (which counts the satisfying assignments of a boolean formula in CNF) can be shown to be $\#\mathcal{P}$-complete by the proof of Cook's theorem, which constructs a boolean formula whose satisfying assignments are in one-to-one correspondence with the accepting computations of an arbitrary $\mathcal{NP}$ machine. However, what makes the class $\#\mathcal{P}$ interesting is that many problems in $\mathcal{P}$ have counting versions that are $\#\mathcal{P}$-complete. Thus, in a sense, "counting is harder than decision."

To give a simple example, we show that #DNF (whose decision problem is trivial) is $\#\mathcal{P}$-complete.

Definition 10.5 *A formula is in Disjunctive Normal Form (DNF) if it is a disjunction of one or more terms, each of which is a conjunction of one or more literals. For example,*

$$\varphi = (x_1 \wedge x_2 \wedge \overline{x}_3) \vee (\overline{x}_2 \wedge x_5) \vee \ldots$$

Finding a satisfying assignment is trivial: we need only satisfy one of the terms, so a satisfying assignment is easily obtained as long as there exists a term in which no two literals are negations of one another. If not, then the formula is not satisfiable.

However, the #DNF problem (computing the number of satisfying assignments to a DNF formula) is much harder. To see this, suppose φ is a formula in Conjunctive Normal Form (as in SAT) with n variables. Note that the formula $\neg\varphi$ can be converted in polynomial time to DNF using de Morgan's laws. Moreover, we clearly have

$$\#\mathrm{SAT}(\varphi) = 2^n - \#\mathrm{DNF}(\neg\varphi).$$

Hence, if we could evaluate #DNF in polynomial time, we would also have a polynomial time algorithm for #SAT. We have thus exhibited a polynomial time Turing reduction from #SAT to #DNF, so #DNF is $\#\mathcal{P}$-complete.

Nonetheless, we do know very good approximation schemes for some $\#\mathcal{P}$-complete problems:

Definition 10.6 *A fully polynomial randomized approximation scheme (FPRAS) for $f : \Sigma^* \to \mathbb{N}$ is an algorithm that, on input (x, ϵ), runs in time polynomial in $|x|$ and $\frac{1}{\epsilon}$, and outputs a value Z (a random variable) such that*

$$\Pr[(1-\epsilon)f(x) \leq Z \leq (1+\epsilon)f(x)] \geq 3/4.$$

Given a FPRAS for a function $f(x)$, using the median trick we can approximate $f(x)$ within $1 \pm \epsilon$ with error probability $\delta > 0$ in time polynomial in $|x|$, $\frac{1}{\epsilon}$ and $\log \frac{1}{\delta}$. A FPRAS is generally accepted as a robust notion of "approximation algorithm" for counting and other similar function evaluation problems. Note that we can only hope for a FPRAS for problems whose decision version is in BPP, since a FPRAS by definition correctly determines whether $f(x) = 0$ with high probability.

The following remarkable theorem, due to Karp and Luby [KL83], shows that the $\#\mathcal{P}$-complete problem #DNF has this very strong type of approximation algorithm:

Theorem 10.7 [KL83] *There exists a FPRAS for #DNF.*

Before describing the Karp/Luby algorithm, let's first discuss a few unsuccessful attempts at finding a FPRAS for #DNF. Let φ be a formula in disjunctive normal form (with n variables and m terms). Let S_i be the set of satisfying assignments for the i^{th} term of φ. Then our goal is to compute the total number of satisfying assignments of φ, which is the size of the union $|\bigcup_{i=1}^{m} S_i|$. The following are two "obvious" approaches to this task:

(1) Use inclusion-exclusion: The inclusion-exclusion formula states that :

$$|\bigcup_{i=1}^{m} S_i| = \sum_{i=1}^{m} |S_i| - \sum_{i \neq j} |S_i \cap S_j| + \sum_{i \neq j \neq k} |S_i \cap S_j \cap S_k| - \dots$$

This summation contains exponentially many terms, so we cannot hope to evaluate it explicitly. We might, however, try to truncate it after a small number of terms and get an approximation. Unfortunately, one can come up with examples to show that any polynomial-length truncation of the sum fails to give a good approximation in general.

(2) Naive Monte Carlo : Another possibility is to apply the Monte Carlo approach described at the beginning of this lecture. Let A be the set of all possible assignments (so $|A| = 2^n$), and let $R = \bigcup_i S_i$ (the set of assignments satisfying φ). Our goal is to estimate $\frac{|R|}{|A|}$. By the Unbiased Estimator Theorem, we would require $\sim \frac{4}{\epsilon^2 \mu}$ trials to get an error of $1 \pm \epsilon$ with probability $> \frac{3}{4}$. Recall that $\mu = \frac{|R|}{|A|}$, so in this case, $\mu = \frac{|\bigcup_i S_i|}{2^n}$; unfortunately this may be exponentially small, so the naive Monte Carlo method may require an exponentially large number of trials.

The reason that the naive Monte Carlo method fails is that the space A is too large with respect to our set of satisfying assignments. The key idea of the Karp/Luby algorithm is to apply the Monte Carlo method, but with respect to a much smaller space A.

Proof: Let φ be a DNF formula with n variables and m terms. Let S_i be the set of satisfying assignments for the i^{th} term. The goal is to compute the size of the union $\bigcup_i S_i$. Let $\mathcal{U} = \{(a, i) : a \text{ satisfies the } i^{th} \text{ term}\}$.

Enumerating the satisfying assignments of φ as $a_1, \dots, a_s$ (where $s = |\bigcup_i S_i|$), we can depict the elements of $\mathcal{U}$ in a chart as follows (marking satisfying pairs with $\times$ and $\otimes$):

	a_1	a_2	a_3	$\dots$	a_s
S_1	$\otimes$		$\otimes$	$\dots$	
S_2	$\times$	$\otimes$		$\dots$	
S_3			$\times$	$\dots$	$\otimes$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
S_m	$\times$			$\dots$	$\times$

For $(a, i) \in \mathcal{U}$, say (a, i) is *special* if $i = \min\{i' : (a, i') \in \mathcal{U}\}$. In the example chart above, these special pairs are marked with $\otimes$. Note that special pairs are always the topmost symbol in their column, so each a_k is in exactly one special pair. Therefore, the number of special pairs is exactly $s = |\bigcup_i S_i|$. Now, we can apply the standard Monte Carlo algorithm with $A = \mathcal{U}$ and $R = \{(a, i) \in \mathcal{U} : (a, i) \text{ is special}\}$:

for $j = 1$ to t **do**
  pick (a, i) uniformly at random from $\mathcal{U}$
  set $X_j = \begin{cases} 1 & \text{if } (a, i) \text{ is special;} \\ 0 & \text{otherwise} \end{cases}$
output $\left(\frac{1}{t} \sum_{j=1}^{t} X_j\right) \cdot |\mathcal{U}|$

Note that $E(X_j) = \frac{\text{number of special pairs}}{|\mathcal{U}|} = \frac{|\bigcup_i S_i|}{|\mathcal{U}|}$, so the algorithm is an unbiased estimator for $|\bigcup_i S_i|$ as desired.

The issues left to cover in the above algorithm are:

The value of t: This is the crucial point. The total number of pairs in $\mathcal{U}$ is at most ms (the size of the above chart), and exactly s of them are special. Hence $\mu = \mathrm{E}[X_j] \geq \frac{s}{ms} = \frac{1}{m}$. Thus by the Unbiased Estimator Theorem, the number of trials required for a $(1 \pm \epsilon)$ approximation is only

$$t = \frac{4}{\epsilon^2 \mu} \leq \frac{4m}{\epsilon^2}.$$

Computing the size of $\mathcal{U}$: Given a term i with q_i literals on distinct variables, the number of satisfying assignments $|S_i|$ is 2^{n-q_i} since q_i variables are "fixed" and the remaining variables can take two values each. Summing over all terms gives $|\mathcal{U}|$.

Picking an element of $\mathcal{U}$ uniformly at random: We pick term i with probability $\frac{|S_i|}{\sum_i |S_i|}$, and then pick a random satisfying assignment for this term. This is done by making the necessary assignments to the variables in the term and assigning (uniformly) random values to the remaining variables.

Complexity: Computing $|\mathcal{U}|$ requires $O(n)$ work for each term, which is $O(nm)$ work overall. Checking to see if a particular pair (a, i) is special requires checking for all $j < i$ if (a, j) is in $\mathcal{U}$. This requires $O(nm)$ work also. Generating a random pair (a, i) takes $O(n + m)$ work once each $|S_i|$ has been computed. So the total running time per trial is $O(nm)$, and since there are $O(m\epsilon^{-2})$ trials, the total running time for the entire algorithm is $O(nm^2/\epsilon^2)$. Thus we have a FPRAS. ■

We conclude with a few additional remarks:

1. With a slightly cleverer implementation, the running time of the above algorithm can be improved to $O(nm\frac{1}{\epsilon^2})$; see [KLM89].
2. The algorithm can be used to compute the size of the union of any collection of finite sets S_i for which the above operations (computing size and picking a random element) can be carried out efficiently.
3. The algorithm easily generalizes to the "probabilistic DNF" problem, in which we are given a DNF formula φ together with a probability p_i for each variable X_i. Our goal is to compute the probability that φ is satisfied, assuming that each variable is set to true with probability p_i, independently of the other variables. The proof is left as an exercise. We shall see an interesting application of probabilistic DNF in the next lecture.

References

[KL83] R.M. Karp and M. Luby, "Monte-carlo algorithms for enumeration and reliability problems," *In Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, 1983, pp. 56–64.

[KLM89] R.M. Karp, M. Luby and N. Madras, "Monte Carlo approximation algorithms for enumeration problems," *Journal of Algorithms* **10** (1989), pp. 429–448.