COMPSCI 532: Design and Analysis of Algorithms	September 25, 2015
Lecture 9	
Lecturer: Debmalya Panigrahi	*Scribe: Allen Xiao*

1 Overview

In this lecture, we introduce a sampling scheme for counting known as the Monte Carlo method.

2 Monte Carlo Sampling

Suppose we have a universe U and some subset $S \subseteq U$. We want to estimate $|S|$, provided we have a uniform sampling procedure for U. An equivalent notion (since we generally know $|U|$) is to estimate $|S|/|U|$. A fairly straightforward algorithm is to take N uniform samples of U, and then estimate $|S|/|U|$ using the *sample mean* of the indicator random variables X_i:

$$X_i = \begin{cases} 1 & \text{if sample } i \text{ is in } S \\ 0 & \text{otherwise} \end{cases}$$

The algorithm outputs the sample mean, $\tilde{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$.

This counting algorithm is often called the *Monte Carlo* method, after a class of algorithms we will describe more next lecture. In expectation, $\tilde{X}$ predicts $|S|/|U|$:

$$\mathbb{E}[\tilde{X}] = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} X_i\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[X_i] = \frac{N}{N}\frac{|S|}{|U|} = \frac{|S|}{|U|}$$

Since $\mathbb{E}[\tilde{X}] = |S|/|U|$, we call $\tilde{X}$ an *unbiased estimator* for $|S|/|U|$. This is independent of the number of samples N, but the value of N greatly influences the *variance*.

Example 1. Consider the following unbiased estimator, which corresponds to the case when $N = 1$: Test a single sample X_1 and estimate $\tilde{X} = 1$ if $X_1 \in S$ ($\implies |S| = |U|$). Similarly, $\tilde{X} = 0$ if $X_1 \notin S$ ($\implies |S| = 0$).

The mean of this estimator is still $|S|/|U|$, by the argument from before. Unless $S = U$ or $S = \emptyset$, this estimator is never really "correct". Somehow, despite the fact this estimator is unbiased, it is always inaccurate. We first need to establish a definition of a good estimator, a stronger notion than that of an unbiased estimator.

Definition 1. *An estimator is* ***probabilistically approximately correct*** *(PAC) if it is almost always nearly correct. Formally, an estimator $\tilde{X}$ for parameter X is an (ε, δ)-estimator if, for $\varepsilon > 0$ and $\delta \in [0,1]$:*

$$\Pr\left(\frac{|\tilde{X} - X|}{X} > \varepsilon\right) < \delta$$

"The estimator is within an ε factor of the true value with probability at least $(1-\delta)$."

Returning to the sampling example, we would like sufficient N to state that:

$$\Pr\left(\left|\tilde{X} - \frac{|S|}{|U|}\right| > \varepsilon\frac{|S|}{|U|}\right) < \delta$$

We can substitute the definition of $\tilde{X}$ to have a sum of independent 0-1 random variables, then apply the Chernoff bound. Recall that each X_i had mean $\mu = \mathbb{E}[X] = |S|/|U|$.

$$\begin{aligned}\Pr\left(\left|\tilde{X} - \frac{|S|}{|U|}\right| > \varepsilon\frac{|S|}{|U|}\right) &= \Pr\left(\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mu\right| > \varepsilon\mu\right) \\ &= \Pr\left(\left|\sum_{i=1}^{N} X_i - \mu N\right| > \varepsilon\mu N\right) \\ &\leq \exp\left(-\varepsilon^2\mu N/3\right)\end{aligned}$$

Now to solve for N:

$$\exp\left(-\varepsilon^2\mu N/3\right) < \delta$$

$$N > \frac{3\log(1/\delta)}{\varepsilon^2\mu}$$

Replacing μ gives us:

$$N > \frac{3|U|\log(1/\delta)}{\varepsilon^2|S|}$$

Practically, we may not know μ or $|S|$, but a rough approximation can be used to choose N. The expression agrees with our intuition: the tighter the error (ε) or lower the failure probability (δ), the more samples we need. It also seems to be harder (require more N) to estimate when μ is small ($|S| \ll |U|$), since it will take more samples to distinguish S from the empty set. As we will see, this is one of challenges in applying Monte Carlo sampling to certain counting problems.

2.1 Importance Sampling

When $1/\mu = |U|/|S|$ is super-polynomial, the Monte Carlo estimator is no longer polynomial time. We will show an example working around this problem using *importance sampling*. The result for this example (DNF counting) is due to Karp, Luby, and Madras [KLM89].

Example 2. A *disjunctive normal form* (DNF) formula in boolean logic is a disjunction (OR) of conjunctive clauses (AND) of literals.

$$(A \wedge B \wedge C) \vee (\neg A \wedge C \wedge \neg D) \vee (B \wedge \neg C)$$

In an assignment, each of A, B, C, D are assigned TRUE or FALSE. A DNF formula is easy to check for satisfiability – if any single clause is satisfiable, the formula is. Because each clause is a conjunction of literals, a clause is satisfiable exactly when it does not contain a contradiction $(x \wedge \neg x)$. *DNF counting* instead asks the question: "How many truth assignments satisfy a given DNF formula?"

Let $\phi = (C_1 \vee \cdots \vee C_m)$ be a DNF formula on n boolean variables $(x_1, \ldots, x_n)$. Let the assignments to variables be $a_i \in \{0,1\}^n$. The problem is to estimate the number of satisfying assignments $|S|$, out of the universe U of possible assignments $\{0,1\}^n$.

It is easy to count the number of satisfying assignments for a clause C_j. First, fix the truth values of all variables in C_j. The unfixed variables each give a multiplicity of 2 to the count. If C_j has t_j fixed variables,

	C_1	C_2	C_3	C_4	$\cdots$
a_1	$\otimes$		$\times$		
a_2		$\otimes$		$\times$	
a_3	$\otimes$	$\times$	$\times$	$\times$	
a_4					
a_5			$\otimes$		
$\cdots$					

Figure 1: A truth table between assignments a_i and clauses C_j. One $\times$ indicates that a_i satisfies C_j. The $\otimes$ at (C_j, a_i) indicates that C_j is the first clause (with respect to j) satisfied by a_i. U' is set of $\times$, while S' is the set of $\otimes$.

then the number of satisfying assignments is $2^{(n-t_j)}$. However, $\sum_j C_j$ will overcount count assignments if any a_i satisfies more than one clause, and will not give us $|S|$.

Recall the Monte Carlo method from the previous section. The number of samples required for failure probability δ was roughly:

$$N \geq \frac{1}{\mu} \cdot \frac{1}{\varepsilon^2} \cdot \log(1/\delta)$$

In DNF counting, the first term may be exponential. $|U| = 2^n$, and $|S|$ may be as small as one assignment for a satisfiable DNF. Even if $|S|$ is polynomial in n and m, the N required for Monte Carlo will be exponential in the number of variables.

Instead of trying to sample from U, we perform the Monte Carlo method on a smaller subset which we know contains S. Let U' be the set of clause-assignment pairs ($C_j \in \phi, a_i \in \{0,1\}^n$):

$$U' = \{(C_j, a_i) \mid C_j(a_i) = 1\}$$

Our original query (S), when modified for U', is the number of unique variables appearing in tuples of U'. We can express this by marking a single tuple of U' for every x. Let $S' \subseteq U'$ be:

$$S' = \{(C_j, a_i) \mid j \text{ is the smallest for which } C_j(a_i) = 1\}$$

Since this is a restriction to one satisfying assignment-clause tuple per satisfying assignment, $|S'| = |S|$.

We now describe a procedure for estimating the size of S' by sampling U', using Monte Carlo sampling. Our estimator will estimate $|S'|/|U'|$, which is generally not equal to $|S|/|U|$. We call such an estimator a *biased estimator*, since it is systematically differs from the true parameter $|S|/|U|$. Since we know the bias (a factor of $|U'|/|U|$), this still serves as an estimator for $|S|/|U|$.

1. To take a uniform random sample from U', we first (non-uniformly) sample a clause C_j. Recall from our discussion earlier that it is possible to compute the number of satisfying assignments N_j for any clause C_j. Formally if t_j is the number of literals in C_j:

$$\begin{aligned} N_j &= |\{(C, a_i) \in U' \mid C = C_j\}| \\ &= \begin{cases} 2^{(n-t_j)} & \text{for no literal } x, \text{ both } x, \bar{x} \in C_j \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

   We sample C_j with probability:

$$\frac{N_j}{\sum_{k=1}^{m} N_k}$$

In total, this takes $O(mn)$ time to precompute all N_j.

2. Second, we choose one satisfying assignment a_i from those satisfying C_j uniformly at random. This can be done in $O(n)$ time by fixing the variables in C_j, then randomly assigning the unfixed variables of a_i. The probability that any single $(C_j, a_i) \in U'$ was chosen is:

$$\begin{aligned}
\Pr((C_j, a_i) \text{ sampled}) &= \Pr(C_j \text{ sampled}) \Pr(a_i \text{ sampled} \mid C_j \text{ sampled}) \\
&= \frac{N_j}{\sum_{k=1}^{m} N_k} \cdot \frac{1}{N_j} \\
&= \frac{1}{\sum_{k=1}^{m} N_k} \\
&= \frac{1}{|U'|}
\end{aligned}$$

   Uniform over U', as we desired.

3. Finally, we check if $(C_j, a_i) \in S'$. We can manually check satisfiability of a_i for all clauses C_k in ascending order of k. If some C_k with $k < j$ is satisfied by a_i, then our sample was not in S'. If $k = j$, our sample was in S'. This check takes $O(mn)$ time.

4. Repeat until we have N samples.

Remember, the number of samples needed by the Monte Carlo method was:

$$N \geq \frac{|U'|}{|S'|} \cdot \frac{1}{\varepsilon^2} \cdot \log(1/\delta)$$

Here, $|U'| = O(m|S|)$, since there are at most m tuples for any satisfying assignment a_i. This gives us:

$$N \geq m \cdot \frac{1}{\varepsilon^2} \cdot \log(1/\delta)$$

This is linear in the number of clauses, and altogether the sampling procedure is polytime.

3 Summary

We presented an unbiased estimator (the Monte Carlo method) for the size of a subset S in a universe U, then used importance sampling to estimate DNF counting in much fewer samples.

From the Chernoff bound for the Monte Carlo estimator:

$$N > \frac{c \log(1/\delta)}{\varepsilon^2 \mu}$$

There is *logarithmic* dependence on $1/\delta$ versus polynomial dependence on $1/\varepsilon$. Lowering δ requires much, much less of an increase in N. In designing these estimators, it is easier to drastically improve "how often estimates are close" than "how close estimates will often be". You may have seen an example of this before: news reports of surveys and polls will often report the margin of error (ε), but omit the confidence ($1 - \delta$) since δ can be driven down exponentially with only a small increase in N.

References

[KLM89] Richard M Karp, Michael Luby, and Neal Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of algorithms*, 10(3):429–448, 1989.