CS 761 : Randomized Algorithms , Spring 2018 , Waterloo

Lecture 17: Markov chain Monte Carlo method

We first see the Monte Carlo method and two interesting algorithms.

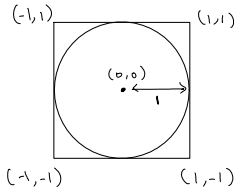
Then we will see the Markov chain Monte Carlo method and the reduction from counting to sampling.

Monte Carlo method

It is a method to estimate values through sampling and simulation.

For example, to estimate the value of π, one can draw a circle of radius one and then repeatedly draw random points from the square and count the fraction of points that fall inside the circle.

The probability that a random point is in the circle is $\frac{\pi}{4}$.



So, if we choose m random points, then the expected number of points in the circle is $\frac{m\pi}{4}$.

Hence, if there are W points fall inside the circle in our experiment, then it is natural to use $\frac{4W}{m}$ as an estimate of the value of π.

How good is this estimation?

Intuitively, it is more accurate if we take more samples, and it is easy to make it precise using the by-now standard Chernoff bound.

<u>Theorem</u> Let $X_1, \ldots, X_m$ be independent, identically distributed indicator random variables, with $\mu = E[X_i]$.

If $m \geq \frac{3\ln(2/\delta)}{\varepsilon^2 \mu}$, then $\Pr\left(\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| \geq \varepsilon\mu\right) \leq \delta$.

<u>Definition</u> A randomized algorithm gives an (ε, δ)-approximation for the value V if the output X of the algorithm satisfies $\Pr(|X - V| \leq \varepsilon V) \geq 1 - \delta$.

So, the above theorem gives an upper bound on the number of samples for an (ε, δ)-approximation.

An important point to notice is that the algorithm is efficient if μ is large.

<u>Definition</u> A <u>fully polynomial randomized approximation scheme</u> (FPRAS) for a problem is a randomized algorithm for which, given an input x and a parameter $0 < \varepsilon < 1$, the algorithm outputs an $(\varepsilon, 1/4)$-approximation to $V(x)$ in time that is polynomial in $1/\varepsilon$ and the size of the input x.

Remark: It is easy to obtain an (ε, δ)-approximation by using an $(\varepsilon, 1/4)$-approximation $O(\ln \frac{1}{\delta})$ times. Checking this is left as an exercise. So, to obtain an (ε, δ)-approximation, we just work with the above definition and do not worry about δ.

DNF counting

This is an example where straightforward application of the Monte Carlo method won't work, but a cleverer application would work.

Usually we consider a CNF (conjunctive normal form) formula which is a conjunction (AND) of the clauses where each clause is a disjunction (OR) of variables, e.g. $(\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4)$.

Here we consider a DNF (disjunctive normal form) formula which is a disjunction (OR) of the clauses where each clause is a conjunction (AND) of variables, e.g. $(x_1 \wedge \bar{x}_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (\bar{x}_1 \wedge x_3 \wedge x_4)$.

We are interested in counting the number of satisfying assignments of a DNF formula.

By the DeMorgan's laws, a DNF formula is just the negation of a CNF formula with the same number of clauses and variables, and so counting DNF solutions is as hard as counting CNF solutions, which is #P-complete.

Thus, we are interested in obtaining FPRAS for the DNF counting problem.

A straightforward approach is to pick many random assignments and count the fraction of satisfying assignments among them.

Applying the Monte Carlo theorem, we get an (ε,δ)-approximation if the number of samples $\Omega(\frac{\ln(1/\delta)}{\varepsilon^2 \mu})$.

In estimating π, μ is $\pi/4$ and so the number of samples required is small.

For the DNF counting problem, μ could be exponentially small, and thus the number of samples required is exponential in n (e.g. if # solution $= n^2$, in polynomial number of steps, the answer is zero whp).

To get an efficient estimation, the idea is to construct a sample space that includes all the satisfying assignments, but at the same time the sample space is not too large.

First, note that finding a satisfying assignment of a DNF formula is easy:

Just pick a clause and satisfy it (since we only need to satisfy one clause to satisfy the formula).

Let $F = C_1 \vee C_2 \vee \dots \vee C_m$.

Let SC_i be the set of assignments that satisfy C_i.

Note that if C_i has l_i variables, then $|SC_i| = 2^{n-l_i}$, as there is only one choice for the variables in C_i but arbitrary values for the remaining variables.

Since each satisfying assignment must be contained in SC_j for some j, and so $|\bigcup_{i=1}^{m} SC_i|$ is exactly the number that we want to compute approximately.

Let $U = \{ (i,a) \mid 1 \le i \le m,\ a \in SC_i \}$. So, $|U| = \sum_{i=1}^{m} |SC_i|$.

This is an over-estimate of $|\bigcup_{i=1}^{m} SC_i|$, since one assignment could satisfy more than one clauses.

To avoid the over-counting, we define a set S with size $|\bigcup_{i=1}^{m} SC_i|$ as follows:

$S = \{ (i,a) \mid 1 \le i \le m,\ a \in SC_i$ but $a \notin SC_j$ for $j < i \}$.

Note that, given a member in U, we can check whether it belongs to S efficiently.

Now, the idea is to approximate $|S|$ by estimating $|S|/|U|$ by Monte Carlo simulation.

The point is that $|S|/|U| \ge 1/m$ as each assignment can satisfy at most m clauses.

So, if we use U as the sample space, then by the Monte Carlo theorem, one can get an (ε, δ)-approximation of $|S|/|U|$ using at most $O(m \ln(1/\delta)/\varepsilon^2)$ samples.

Since we can compute $|U|$ efficiently, we can thus get an (ε,δ)-approximation of $|S|$ as well.

So, it remains to generate a uniform sample from U.

This is easy. Just pick i with probability $|SC_i| / \sum_{j=1}^{m} |SC_j|$ and then pick a uniform member of SC_i by setting each variable not in clause i randomly and independently with probability $1/2$ to be true and probability $1/2$ to be false.

Then, $\Pr((i,a) \text{ is chosen}) = \Pr(i \text{ is chosen}) \cdot \Pr(a \text{ is chosen}) = \frac{|SC_i|}{|U|} \cdot \frac{1}{|SC_i|} = \frac{1}{|U|}$.

Putting together we have a FPRAS for the DNF counting problem.

Network reliability

This is an interesting application of the DNF counting result.

The problem is simple: given an undirected graph where each edge will fail with probability p, Compute the probability that it is disconnected.

This "simple" problem turns out to be #P-complete, and thus we can only hope for a FPRAS.

The solution of this problem nicely combine several ideas we have learnt.

Let FAIL(p) be the probability that the graph is disconnected when the failure probability of an edge is p.

Note that a graph is disconnected iff all edges of some cut failed.

Let C be the size of a minimum cut in G.

First we consider an easy case.

Case 1: $p^C \geq n^{-4}$ This implies that $\text{FAIL}(p) \geq n^{-4}$.

Therefore, if we run the Monte Carlo simulation, then we can get an (ε,δ)-approximation of FAIL(p) using at most $O\left(\frac{n^4 \ln(1/\delta)}{\varepsilon^2}\right)$ samples since $\mu = \text{FAIL}(p) \geq n^{-4}$.

The remaining case is the interesting case.

Case 2: $p^C \leq n^{-4}$ The idea is simple and elegant. The proof has two main steps.

① We know that an undirected graph has at most $O(n^{2\alpha})$ cuts with at most αC edges.

Since p^C is small, those cuts with more than $O(n^{2\alpha})$ cuts with at most αC edges are very unlikely to fail, with probability $p^{\alpha C} \leq n^{-4\alpha}$.

Therefore, we can set α to be small (but big enough) and focus only on the cuts with $\leq \alpha C$ edges.

In fact, we can generate all of these small cuts in polynomial time using the random contraction algorithm.

② The key observation is that checking whether some small cuts will fail can be reduced to a generalization of the DNF counting problem, where each variable is set to true with prob p.

Now we go into some details for each step.

Enumerating small cuts: We leave it as an exercise that it can be done in $\tilde{O}(n^{2\alpha} \cdot T_{\text{contraction}})$, where $T_{\text{contraction}}$ denotes the running time of Karger's random contraction algorithm.

Ignoring large cuts: Let $p^C = n^{-(2+k)}$ for some $k > 2$.

Then, $\Pr(\text{some cut with } \geq \alpha C \text{ edges fails}) \leq \int_{\beta \geq \alpha} n^{2\beta} p^{\beta C} d\beta \leq \int_{\beta \geq \alpha} n^{-k\beta} d\beta = O(n^{-k\alpha})$.

By setting $\alpha = 2 + \frac{\ln(\varepsilon / O(1))}{k \ln n}$, then $O(n^{-k\alpha}) = n^{-2k} \cdot \varepsilon \leq \varepsilon \cdot \text{FAIL}(p)$

Therefore, ignoring all large cuts, we still have a $(1-\varepsilon)$-approximation of FAIL(p).

Furthermore, since $\alpha = O(1)$, the above enumeration can be done in polynomial time.

Reduction to DNF counting. The reduction is actually straightforward.

Create a variable for each edge e.

For each small cut, create a clause $(x_{e_1} \wedge x_{e_2} \wedge \cdots \wedge x_{e_\ell})$ if the cut has ℓ edges.

Set x_e to be true if edge e fails.

Then the formula is true if and only if all the edges in some cut fail.

Therefore, this is just the DNF counting problem when each variable is true with probability p.

We may ask the details in the homework.

So, using a FPRAS for DNF-counting (with prob p) can obtain a FPRAS for network reliability.

Open question: Is there a FPRAS for computing the probability that s and t is disconnected in the resulting graph where each edge fails with probability p?

Approximate Sampling and approximate counting

We have seen examples of how to use sampling to do approximate counting.

We will show that approximate counting can be reduced to approximate sampling.

The reduction is quite general, but we will use the example of counting independent sets of a graph to illustrate the technique.

Definition Let w be the random output of a sampling algorithm for a finite sample space Ω.

The sampling algorithm generates an ε-uniform sample of Ω, if for any element z in Ω, $\frac{1}{2} \sum_{z \in \Omega} \left| \Pr(w=z) - \frac{1}{|\Omega|} \right| \leq \varepsilon$, that is, the total variation distance between the output distribution and the uniform distribution is at most ε.

A sampling algorithm is a fully polynomial almost uniform sampler (FPAUS) for a problem if, given an instance x and a parameter $\varepsilon > 0$, it generates an ε-uniform sample of $\Omega(x)$ and runs in time that is polynomial in $\ln(\frac{1}{\varepsilon})$ and the size of the instance x.

Our goal is to show that, given a FPAUS for sampling independent sets, we can construct a FPRAS for counting the number of independent sets.

Let $e_1, e_2, \ldots, e_m$ be the edges of G. Let E_i be the first i edges and $G_i = (V, E_i)$.

Let $\Omega(G_i)$ be the set of independent sets in G_i.

Then, $|\Omega(G)| = \frac{|\Omega(G_m)|}{|\Omega(G_{m-1})|} \cdot \frac{|\Omega(G_{m-1})|}{|\Omega(G_{m-2})|} \cdot \ldots \cdot \frac{|\Omega(G_1)|}{|\Omega(G_0)|} \cdot |\Omega(G_0)|$.

Since G_0 has no edges, every subset is an independent set, and so $|\Omega(G_0)| = 2^n$.

Let $r_i = |\Omega(G_i)| / |\Omega(G_{i-1})|$ for $1 \le i \le m$. Then $|\Omega(G)| = 2^n \prod_{i=1}^{m} r_i$.

To approximate $|\Omega(G)|$, we will compute approximation $\tilde{r}_i$ for r_i, and output $2^n \prod_{i=1}^{m} \tilde{r}_i$ as the approximation to $|\Omega(G)|$.

To bound the error, we need to bound the ratio $R = \prod_{i=1}^{m} \frac{\tilde{r}_i}{r_i}$. The following lemma is left as exercise.

Lemma Suppose $\tilde{r}_i$ is an $(\varepsilon/2m, \delta/m)$-approximation to r_i for all $1 \le i \le m$.

Then $\Pr(|R-1| \le \varepsilon) \ge 1-\delta$, which implies an (ε, δ)-approximation to $|\Omega(G)|$.

To obtain a $(\varepsilon/2m, \delta/m)$-approximation for r_i, we estimate it by a Monte Carlo algorithm that uses the FPAUS for Sampling independent sets.

The idea is to (approximately) sample independent sets in G_{i-1} and compute the fraction of these sets that are also independent in G_i.

Let $e_i = (u, v)$. The only independent sets in G_{i-1} that are not independent in G_i are those that contain both u and v.

The important point is that $r_i \ge \frac{1}{2}$.

To see this, map each independent set I in $\Omega(G_{i-1}) - \Omega(G_i)$ to the independent set $I - v$ in G_i.

Note that each independent set in $\Omega(G_i)$ is mapped by at most one independent set in $\Omega(G_{i-1}) - \Omega(G_i)$.

This implies that $|\Omega(G_i)| \ge |\Omega(G_{i-1}) - \Omega(G_i)|$, and thus $r_i = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1})|} = \frac{|\Omega(G_i)|}{|\Omega(G_{i-1}) - \Omega(G_i)| + |\Omega(G_i)|} \ge \frac{1}{2}$.

Therefore, if we have a uniform sampler for independent sets in G_{i-1}, then we can get a good approximation $\tilde{r}_i$ to r_i using only a few samples by using the Monte Carlo theorem.

The difference is that we only have a FPAUS for sampling independent sets, but not surprisingly, using a good enough sampler will give us a good enough approximation to r_i.

There are two errors to bound.

- Since we are only using a FPAUS, the expected value may not be equal to r_i.

  But using an $(\varepsilon/6m)$-uniform sampler, we can show that $|E[\tilde{r}_i] - r_i| \leq \varepsilon/6m$.

- By taking only some samples, we can only get an approximation to $E[\tilde{r}_i]$.

  Since r_i is big (i.e. $\geq \frac{1}{2}$), $E[\tilde{r}_i]$ is big and thus the Monte Carlo theorem would work to show that $\tilde{r}_i$ is close to $E[\tilde{r}_i]$ by taking only $O(\frac{m^2}{\varepsilon^2} \ln(\frac{1}{\delta}))$ samples.

Combining these, we can show that $\tilde{r}_i$ is a good enough approximation to r_i.

Thm Given a FPAUS for sampling independent sets, one can construct a FPRAS for counting independent sets.

The reduction works for many "self-reducible" problems, e.g. counting number of graph colorings.

Markov chain Monte Carlo method

If there is an approximate sampler, then we can do approximate counting.

But the hard part is to construct an approximate sampler.

Here is where Markov chain comes into the picture.

The basic idea is to construct a finite, irreducible, aperiodic Markov chain whose set of states is the sample space (e.g. each node corresponds to an independent set) and whose stationary distribution is the required distribution (e.g. uniform distribution).

We know that it will converge to the stationary distribution. So, after a long enough time, the distribution will be close to the required distribution and we get an approximate sample.

For this approach to be efficient, we need to show that the convergence rate is fast and each step can be implemented efficiently.

This is usually the hard part, and we will come back to this next time.

For now we just consider how to construct a Markov chain with the required stationary distribution.

First, we state a useful lemma.

Lemma Given a finite, irreducible, aperiodic Markov chain with transition matrix P.

If $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ satisfies $\sum_{i=1}^{n} \pi_i = 1$ and $\pi_i P_{i,j} = \pi_j P_{j,i}$ for all $i \neq j$,

then $\vec{\pi}$ is the stationary distribution of P.

Proof For any j, $\pi_j = \sum_{i=1}^{n} \pi_i P_{i,j} = \sum_{i=1}^{n} \pi_j P_{j,i} = \pi_j \sum_{i=1}^{n} P_{j,i} = \pi_j$, and so $\vec{\pi}$ is the unique stationary distribution. □

The condition $\pi_i P_{i,j} = \pi_j P_{j,i}$ is called time reversible.

If the underlying graph of the Markov chain is an undirected graph, then we know that the stationary probability of a vertex is proportional to its degree.

Therefore, if we want the uniform distribution to be the stationary distribution, we just need to make sure that the graph is regular. If not regular, we just add some self-loops to make it regular.

Lemma For a finite state space Ω and neighborhood structure $\{N(x) \mid x \in \Omega\}$, let $N = \max_{x \in \Omega} |N(x)|$ and $M \geq N$. Consider the following Markov chain where

$$P_{x,y} = \begin{cases} 1/M & \text{if } x \neq y \text{ and } y \in N(x) \\ 0 & \text{if } x \neq y \text{ and } y \notin N(x) \\ 1 - |N(x)|/M & \text{if } x = y. \end{cases}$$

If the chain is irreducible and aperiodic, then the stationary distribution is the uniform distribution.

Proof Since $P_{i,j} = P_{j,i}$ for all $i \neq j$, if $\pi_i = \pi_j \ \forall i \neq j$, then $\pi_i P_{i,j} = \pi_j P_{j,i}$. □

For the independent set problem, one can apply this lemma to prove that the following Markov chain has the uniform distribution as its stationary distribution.

① Start from an arbitrary independent set X_0 (e.g. the empty set)

② To compute X_{i+1}

- ⓐ Choose v uniformly at random from G.
- ⓑ If $v \in X_i$, then $X_{i+1} = X_i - v$.
- ⓒ If $v \notin X_i$ and $X_i + v$ is an independent set, then $X_{i+1} = X_i + v$.

This chain is irreducible because each state can reach every other state by first deleting vertices and then adding vertices.

It is aperiodic because it has a self loop as long as the graph has at least one edge.

It follows from the above lemma that the stationary distribution is the uniform distribution.

The Metropolis algorithm

This is a general construction of a Markov chain with any stationary distribution.

Lemma Under the same condition as in the previous lemma, for any distribution $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$,

$$P_{x,y} = \begin{cases} (1/M)\min\{1, \alpha_y/\alpha_x\} & \text{if } x \neq y \text{ and } y \in N(x) \\ 0 & \text{if } x \neq y \text{ and } y \notin N(x) \\ 1 - \sum_{y \neq x} P_{x,y} & \text{if } x = y. \end{cases}$$

Then $\vec{\alpha}$ is the stationary distribution of this Markov chain.

Proof Assume $\alpha_x \leq \alpha_y$. Then $P_{x,y} = \frac{1}{M}$ and $P_{y,x} = \frac{1}{M}\frac{\alpha_x}{\alpha_y}$. This is time reversible. □

For example, suppose we want a Markov chain for independent sets with stationary probability proportional to $\lambda^{|I|}$ where λ is a constant and $|I|$ is the size of the independent set.

By the above lemma, we just need to modify the Markov chain as follows:

- if $v \in X_i$, set $X_{i+1} = X_i - v$ with probability $\min\{1, 1/\lambda\}$.
- if $v \notin X_i$ and $X_i + v$ is an independent set, set $X_{i+1} = X_i + v$ with probability $\min\{1, \lambda\}$.

Then, it is easy to verify that the stationary probability is proportional to $\lambda^{|I|}$.

To summarize, it is relatively easy to construct a Markov chain with our desired stationary distribution. The hard part is usually to analyze its mixing time.

We will study the technique of "coupling" to establish fast mixing time.

References Most of the materials in this lecture is from chapter 10 of Mitzenmacher and Upfal.

The network reliability example is taken from the lecture notes of Eric Vigoda.