Counting using Markov chains

Scribe : Rahul Gupta

February 14, 2003

1 Introduction

In this lecture, we will look at Markov-chain based Monte Carlo approximation algorithms for enumeration problems. Typically, the exact solutions to these problems will be exponential in the size of the input. Hence we will devise randomized approximation algorithms whose relative error is bounded by a constant.

Consider the following enumeration problems. The algorithms devised will be illustrated on them.

1. **Counting perfect matchings**. Here the input is an n-vertex bipartite graph with $O(n^2)$ edges and we wish to compute the number of distinct matchings in the graph. Clearly, the number of perfect matchings can be as large as $(n/2)!$ for the complete bipartite graph $K_{n/2,n/2}$.

2. **Lattice point counting :** Given an arbitrary convex polytope in n-dimensions, compute $|\{\vec{x} = (x_1, \ldots, x_n) \mid \vec{x} \in \textit{polytope}, \forall i : x_i \in \mathcal{Z}\}|$. If we consider the n-dimensional hypercube, then the number of lattice points inside the hypercube is $O(a^n)$, where a is some constant.

3. **Weighted counting :** Consider n particles placed in a $\sqrt{n} \times \sqrt{n}$ grid. Let $\sigma : \{1, \ldots, n\} \to \{-1, 1\}$ be the 'spin' function. σ_i denotes the spin on particle i. Further, with a spin function σ, we associate an energy function given by:

$$H(\sigma) = \sum_{i,j \ adj.} w_{ij} \sigma_i \sigma_j \tag{1}$$

   where w_{ij} is the weight given to the interaction between neighbors i and j. Also, the probability of a configuration σ is given by

$$P(\sigma) = e^{-\beta H(\sigma)} / \sum_{all \ \chi} e^{-\beta H(\chi)} \tag{2}$$

   The problem is to enumerate the $O(2^n)$ different spin configurations and then compute their corresponding $P(\sigma)$'s.

All these problems are #P-hard.

2 Uniform generation

For examples 1 and 2, we can use a technique called *Uniform Generation* to approximately count the desired quantity. The idea of uniform generation is simple. Assume that we are interested in counting the number (or proportion) of objects that satisfy a certain property p. We are also given a black box that generates sample objects uniformly at random (uar) from the sample space. Let N_p be the number of objects generated that satisfy the property p, and let N be the total samples generated. Then N_p/N is the required approximation to the global proportion.

However, to ensure a meaningful running time, we can generate atmost a polynomial number of objects using the black box. Later we will see how we can bound our relative error while still managing polynomial running time. Let us look at how this approach can be used to solve problem 1.

Assume that we have a black box $PM(G)$ that, given an input graph G, generates a perfect matching uar from amongst the various possible matchings.

```
Procedure CountMatchings(G)
    (V,E) <- G;
    if(|E| <= 1) return |E|;
    let e be an arbitrary edge in E;
    count <- 0; count_e <- 0;
    repeat this loop 'sufficient' number of times
        pm <- PM(G);
        if(pm has edge e) then count_e <- count_e + 1;
        count_e <- count + 1;
    end loop
    return CountMatchings(G \ {e}) * count / (count - count_e);
End Procedure
```

This procedure takes polynomial time as $CountMatchings(.)$ is called $O(n^2)$ times and in each call, the black box is called a polynomial number of times. Note that in general, the observed quantity $count/(count - count\_e)$ (= say p) is exponentially small and will be found to be almost always 0 in practice. In that case, the algorithm can be modified so that multiple edges are deleted instead of just one.

In the following sections, we describe the internals of the 'black box' that makes up the core of our algorithm.

3 Markov chains

Recall that a Markov chain M is a pair (Ω, P). Ω is the finite set of states in the chain and P is a matrix such that P_{ij} is the transition probability from state i to j.

Extending the notion to multiple transitions, P^t_{ij} denotes the probability of a t-step transition from state i to j.

Definition 1 *A Markov chain M is called connected if the underlying directed graph is connected viz. we can move from any state i to any state j.*

Definition 2 *A Markov chain M is called lazy if $\forall i \in \Omega : P_{ii} \geq 1/2$.*

Definition 3 *A Markov chain M is symmetric if $\forall i, j \in \Omega : P_{ij} = P_{ji}$.*

Note that any Markov chain M can be made *lazy* by using a biased coin. If in M, we are at a state i, then we can use a biased coin with heads probability of P_{ii}. We then toss the coin and follow M's move only when tail appears, otherwise we just stall at i. The expected running time increases by factor of 2.

Theorem 1 (Fundamental theorem) *If a Markov chain is connected and lazy, then it has steady state probabilities:*

$$\lim_{t \to \infty} P_{ij}^t = \pi_j \tag{3}$$

This theorem states that for any state j, no matter what the starting state is, we can eventually reach j with the same probability π_j.

Lemma 1 *A connected, symmetric and lazy Markov chain has equal steady state probabilities.*

This is a direct corollary of Theorem 1 (just swap i and j).

3.1 Lattice point counting

We now discuss the uniform generation technique used for Problem 2. We will construct a Markov chain whose state will correspond to certain points inside the polytope. The Markov chain will have uniform steady state probabilities. Whenever the chain is requested to generate a point, we just initiate a sufficiently long run in the chain. Because of uniformity of steady state probabilities, points generated will be uar. [1]

To generate the Markov chain, put a n-dimensional grid structure over the polytope. Each grid point that is inside or on the polytope corresponds to a state in the chain. If i and j are two adjacent grid points inside the polytope, then we define P_{ij} and P_{ii} as

$$\begin{aligned} P_{ij} &= 1/4n \;\; (i, j \textit{ distinct and adjacent}) && (4) \\ P_{ii} &= \sum_{j \;\, adj, j \neq i} P_{ij} && (5) \end{aligned}$$

Note: 1. In general, i can be adjacent to atmost $2n$ other grid points, hence we get $P_{ii} \geq 1/2$, which makes the chain lazy.
2. The grid should be 'fine' enough so as to ensure the connectivity of the

[1] A 'random sampling in the bounding box' approach will not work in this problem because the number of samples that will lie inside the polytope will be exponentially small in the general case.

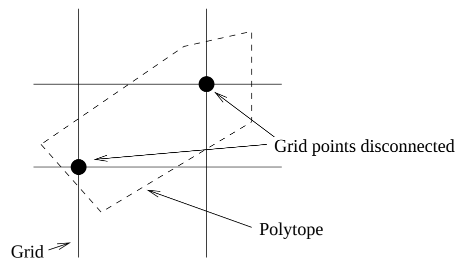

Figure 1: Example of a sparse grid disconnecting the chain

chain. For example, Figure 1 illustrates a sparse grid causing the chain to be disconnected.
3. Since the chain is lazy, symmetric and connected, the steady states probabilities are uniform.

Once the chain is ready, to generate a random point, we simply follow a 'sufficient' number of transitions. An initial point can be chosen by taking the interior lattice point that is closest to some arbitrary vertex of the polytope. The rapid mixing theorem in the following section will show that a polynomial number of moves will suffice.

4 Rapid Mixing

Ideally we would like to sample from the Markov chain state space with sampling probabilities equal to the steady state probabilities. However, if we are willing to tolerate ϵ-relative error, then rapid mixing proves that in $t = O(poly(\epsilon, |\Omega|))$ steps :

$$(1-\epsilon)\pi_j \leq P_{ij}^t \leq (1+\epsilon)\pi_j \quad \forall j \in \Omega \tag{6}$$

However, certain conditions must be met before such a t can exist. For this, consider the following notion of conductance.

Definition 4 *Conductance of a set $S \subseteq \Omega$ is:*

$$\phi(S) = \frac{\sum_{i \in S, j \in \overline{S}} \pi_i P_{ij}}{\sum_{i \in S} \pi_i \sum_{j \in \overline{S}} \pi_j} \tag{7}$$

Hence, for the uniform case, we have:

$$\phi(S) = \frac{|\Omega| \sum_{i \in S, j \in \overline{S}} P_{ij}}{|S||\overline{S}|} \tag{8}$$

Note that $\sum_{i \in S, j \in \overline{S}} P_{ij}$ is the capacity of the cut $(S, \overline{S})$. Another interpretation of $\phi(S)$ can be made in terms of network flows. For this, consider any uniform flow (a flow is uniform if each vertex sends an equal amount of flow for every other vertex.).

Lemma 2 *If there is a uniform flow of value f, then $\forall S \subseteq \Omega : |\Omega| f \leq \phi(S)$. (A maximum of $f|S||\overline{S}|$ flow can go across the cut $(S, \overline{S})$. But the capacity of this cut is $\sum_{i \in S, j \in \overline{S}} P_{ij} = \phi(S)|S||\overline{S}|/|\Omega|$.)*

Let $\phi = \min_{\{S: 0 < \pi(S) < 1\}} \phi(S)$, where $\pi(S) = \sum_{i \in S} \pi_i$. Hence, ϕ is the bottleneck conductance of the Markov chain. Also note that $\phi(S)$ can be seen as the probability of escaping from S to $\overline{S}$ in 1-step, conditioned on starting in S in the steady state, for $\pi(S) \leq 1/2$. Hence ϕ defines the worst case 'escape' probability. Hence, a necessary condition for rapid mixing would be to have a high escape probability from any S. Consider, for example a chain which comprises of two dense large graphs which are connected by just a single edge whose transition probability is also arbitrarily small. In that case, the escape probability from either graph would be quite low and that would hinder rapid mixing.

Theorem 2 (Rapid Mixing) *If a Markov chain is connected and lazy, then*

$$\max_{i,j} \left|\frac{P_{ij} - \pi_j}{\pi_j}\right| \leq (1 - \phi^2/2)^t |\Omega| \tag{9}$$

Note that if $\phi > 0$, then with each increment in t, the relative error falls proportional to $|\Omega|$, which is exponential in the problems under discussion. Moreover, the higher the escape probability (ϕ), the less the number of steps we have to take to bound the relative error.

Also note that the right hand side of Equation 9 is bounded by $e^{-\phi^2 t/2}|\Omega|$. Hence given a relative error bound ϵ, it is enough to have :

$$t \geq \frac{2}{\epsilon \phi^2} + log|\Omega| \tag{10}$$

Hence, even if Ω is exponential, only polynomial number of moves are required. We now compute ϕ and t for a simple Markov chain.

4.1 A simple example

Take an n-dimensinal square grid of side m. Let the grid points correspond to states of the Markov chain and the only transitions are between neighbors (and self). Hence $\Omega = \{1, \ldots, m\}^n$ and for each edge e, $P_e = 1/4n$. Let us calculate the maximum uniform flow possible in this network. Let each vertex a send a flow f to any other vertex b. Each vertex is represented by an n-dimensional vector. The routing strategy for the flow is - in the i^{th} round, an edge e is chosen such that the i^{th} coordinates of the current node and the destination node match. An example path is drawn in Figure 2 for the two dimensional case.

Let $e = (x, y)$, where $x = (x_1, \ldots, x_n)$ and $y = (x_1, \ldots, x_i, x_{i+1}+1, x_{i+2}, \ldots, x_n)$. Then the possible source vertices that will use this edge are the ones whose last $n - i - 1$ coordinates match viz. any vertex of the form $(\_, \ldots, \_, x_{i+2}, \ldots, x_n)$

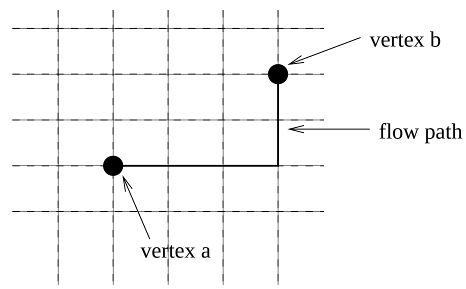

Figure 2: Example of a flow path in a 2-d grid

is a possible source. Number of such sources is m^{i+1}. Similarly, the number of possible sinks is m^{n-i}.

Therefore, the total flow possible through e is fm^{n+1}. However, the capacity of the edge is $P_e = 1/4n$. Hence, the maximum flow possible is $f_{max} = 1/4nm^{n+1}$. So, by Lemma 2, we have :

$$\phi \geq |\Omega| f = 1/4nm \tag{11}$$

Hence, by Equation 10, we have :

$$t \geq O(m^2 n^2/\epsilon) + nlogm \tag{12}$$

5 Non-uniform steady states

We can extend the notion of uniform steady states to unequal steady states easily. Let us say that we wish to change a Markov chain (Ω, P) with uniform steady states to (Ω, Q) with non-uniform ones.

In general we will also provide a weight function $F : \Omega \rightarrow \Re_+$ such that in the new chain, π_i is proportional to $F(i)$. Metropolis gave a rule for converting P to Q that respects this requirement.

$$\begin{aligned} Q_{ij} &= P_{ij} min(1, F(j)/F(i)) \quad i \neq j & (13) \\ Q_{ii} &= 1 - \sum_{j \neq i} Q_{ij} & (14) \end{aligned}$$

In the new chain (Ω, Q), $\pi_i \propto F(i)/\sum_j F(j)$.

Chains with non-uniform steady states can be used to solve problems like the weighted counting problem described in Section 1.