COMP 480/580 — Probabilistic Algorithms and Data Structure March 26, 2019

Lecture 18

Lecturer: Anshumali Shrivastava *Scribe By: Lu Zeng and Yi Gao*

1 Review: Disjunctive Normal Forms(DNF) Counting

1.1 DNF and CNF(Conjunctive Normal Forms)

A logical statement is a disjunctive normal form if it is a disjunction (sequence of ORs) consisting of one or more disjuncts, each of which is a conjunction (AND) of one or more literals. More plainly, a DNF is a OR of AND clauses. The conjunctive normal form is a conjunction of disjunctions of literals (AND of OR clauses).

1.2 Problem Statement

Suppose we have n boolean variables x_1, x_2, ..., x_n, and we want at least one of a set of clauses such as $(x_5 \wedge x_3 \wedge \neg x_2) \vee (\neg x_6 \wedge x_3 \wedge \neg x_1) \vee (x_2 \wedge x_1)$ to be true. If only one solution for assignment of these variables is required, it is easy to find an assignment that makes the DNF true. However, as discussed in the previous lecture, it is difficult to determine the number of assignments that make the DNF true. In the previous lecture, we presented a naive solution to this problem and pointed out several problems. Here, we provide a better solution.

1.3 Naive solution

Recall our naive Monte Carlo solution:

```
1    c = 0
2    for 1 to M
3        generate random assignment
4            If DNF satisfied then c++
5    return (c/M * 2^n)
```

However, this solution requires too many samples to draw the expectation. In particular, if there are few assignments that satisfy the DNF, we require a very large number of samples to solve the problem. To improve our solution, we will consider a bigger set that includes the correct set of assignments.

To start, write the DNF as a set of clauses $C_1, C_2, ...C_n$.

$$F = C_1 \vee C_2 \vee C_3 \vee ...C_n$$

Let SC_i be the set of all assignments that satisfies C_i. We want to approximate $N(S)$. Note that $N(S) = \bigcup_i \mathrm{SC}_i$. To do this, we will break down SC_i as follows

$$\mathrm{SC}_i = (i, e_1), (i, e_2), (i, e_3)...(i, e_k)$$

Where $e_1, e_2, ...$ are assignments that satisfy C_i. We want to find the biggest set

$$U = (i, a)|a \in \mathrm{SC}_i$$

and shrink it and get the target set. It is easy to count $|\mathrm{SC}_i|$ as if C_i has l_i variables, then $2nl_i$ assignments satisfy C_i. As

$$|\mathrm{SC}_i| = 2nl_i|U| = t_i = 1|\mathrm{SC}_i|$$

We will sample element (i, a) from U uniformly at random and determine if $(i, a) \in N(S)$. It is easy to check this since we only need to know whether or not a satisfies C_j for some $j < i$. We claim that

$$\frac{N(S)}{|U|} \geq \frac{1}{t}$$

Each assignment can satisfy at most t different clauses. If we do Chernoff bound analysis, we find that

$$\frac{\ln(\frac{2}{\delta})}{\epsilon^2 \mu}$$

samples will be sufficient. Assuming we can sample from U, the whole process will only take polynomial time! To pick up samples at random from U, we pick i with probability $|\mathrm{SC}_i|$ and we randomly pick an element a of $|\mathrm{SC}_i|$. Then,

$$Pr((i, a)\text{is chosen}) = Pr(i\text{is chosen}) \times Pr(a\text{is chosen } |i\text{is chosen})$$

$$Pr((i, a)\text{is chosen}) = \frac{|\mathrm{SC}_i|}{|U|} \times \frac{1}{|\mathrm{SC}_i|} = \frac{1}{|U|}$$

Therefore, we can sample uniformly at random from U.

2 Markov Chains

There are many situations where we wish to sample from a given distribution, but it is not immediately clear how to do so. For example, we are given N cards and want to generate a random shuffling. Or the sample space of possible outcomes may be exponentially large and we cannot compute it. We approach this problem by introducing Markov Chain Monte Carlo (MCMC) sampling approach.

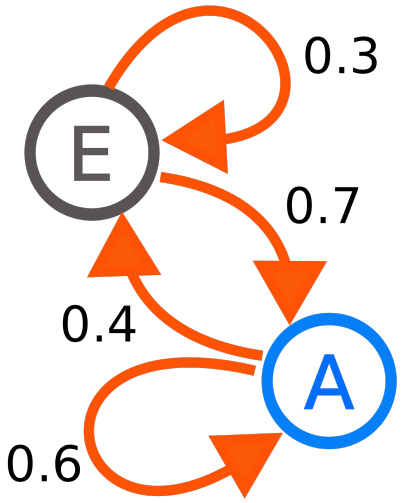
2.1 Notation

We will use Ω to denote the state space and w to denote a weight function for items in the state space. We want to sample $x \in \Omega$ with probability $\frac{w(x)}{Z}$, where Z is the sum of all the weights ($Z = \sum_{x \in \Omega} w(x)$). We will assume that the state space is too large to compute Z directly. Many practical examples have prohibitively large state spaces. For instance, consider the example of shuffling cards. If we have N cards and want to generate a random sequence, the state space is all possible sequences of cards. The cardinality of the state space $|\Omega| = N!$. For a standard 52 card deck, $|\Omega| \approx 8 \times 10^{67}$.[1] Obviously, we need an efficient way to deal with such large-scale state spaces situations.

[1] To put this number in perspective, consider that there are approximately 10^{57} atoms of hydrogen in the sun.

2.2 Markov Chains

A Markov chain is a stochastic process consisting of an indexed sequence of random variables $[x_0, x_1, x_2, ..., x_t, ...]$, $x_i \in \Omega$.



Let x_t be the value (or state) of x at time t. Then the conditional probability of x_{t+1} is

$$Pr(X_{t+1} = y | X_t = x_t, X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}...) = Pr(X_{t+1} = y | X_t = x_t) = P(x, y)$$

Therefore, the Markov Chain is a memoryless system because the next state of the system depends only on the current state. A stochastic process satisfies the Markov property if one can make predictions for the future of the process based solely on its present state.