COMP 480/580 — Probabilistic Algorithms and Data Structure Oct 18, 2022

Lecture 16 - Basic Sampling 2

Lecturer: Anshumali Shrivastava *Scribe By: Tianling(Frank) Feng*

1 Review: Sampling Techniques

In the previous lecture, we introduced basic ideas and formulations for sampling. We answered the question: How do we sample from a distribution with CDF $F(x) = Pr(X \leq x)$ with access to only a uniform distribution $U \sim [0, 1]$? We covered inversion sampling, importance sampling and rejection sampling. In this lecture, we will explore some other sampling methods as well as the difficulty of sampling in a higher dimension.

2 Transformation Based Sampling Method

2.1 Sampling with Discrete Probability

If the distribution is discrete instead of continuous, then our goal becomes to sample x from $(o_1, o_1, \dots, o_n)$ proportional to probability $(p_1, p_1, \dots, p_n)$. To solve this problem, we will first create a number line from 0 to 1. On this number line will be nodes starting from P_1 and $P_1 + P_2$ until the last node $P_1 + P_2 + \cdots + P_n$, which is equal to 1. Then, we will simply pick a uniform random number between 0 and 1, and sample based on this number (i.e. choose the corresponding outcome). In this case, it will take $O(n)$ to create the data structure and $O(\log n)$ to query the number line.

Algorithm:
1. Sample $u \sim U[0, 1]$
2. Return o_j if $\sum_{i=1}^{j-1} p_i \leq u \leq \sum_{i=1}^{j} p_i$

The problem of this approach is that it will have a hard time dealing with exponential size of data. For instance, if we want to shuffle 52 cards, this will have 52! possible combinations. It would be very slow if we want to sample one combination from these combinations.

2.2 Box-Muller Transformation

An alternative to Inverse Transformation is the Box-Muller Transform. This is a well-known sampling method for generating pairs of independent Gaussian variables with zero mean and unit variance. This method is important because sampling from a Gaussian distribution can be difficult. Since there is no closed form formula for the CDF of Gaussian distribution, we can't apply inverse transformation to sample from it. The Box-Muller Transformation can solve this problem with the following algorithm.

Algorithm:
1. Sample $u_1, u_2 \sim U[0, 1]$
2. Let $R = \sqrt{-2\log(u_1)}, \theta = 2\pi u_2$
3. Return $x_1 = R\cos\theta, x_2 = R\sin\theta$

2.3 Samples from a Composition Form

So far, we are only considering sample from one distribution. But what about sampling from a combination form of two distributions. Consider the following example: we can sample from $f(x)$ and we can sample from $g(x)$, is it possible for us to sample from $(1-\alpha)f(x)+\alpha * g(x)$ given $x \leq \alpha \leq 1$? The answer is yes. And the algorithm is quite simple. We just create a hash function and randomly generate 0 or 1. If the hash function returns 0, we will sample from $f(x)$. If the hash function returns 1, we will sample from $g(x)$.

3 The difficulty of Sampling in a Higher Dimension

3.1 Use Monto Carlo Method to estimate area of circle in 2-dimension

Let's first consider an example in 2-dimensional space. Let's say we have a 1×1 square and a circle inscribed into this square. We want to find out the area of circle respective to square. One way to do so is just repetitively throw darts at the shape and see how many darts (out of all throwing darts) land in the circle. In a 2-dimensional space, the answer would be $\pi/4$.

3.2 The Difficulty that Arise with Higher Dimension

As the dimension goes up, we are no longer sampling from a circle but rather from a high-dimensional sphere in a high-dimensional cube. It turns out that as the dimensionality $d \longrightarrow \infty$ the fraction of the sphere volume to the cube volume $\frac{V(\text{ sphere })}{V(\text{ cube})} \longrightarrow 0$. This result suggests that we need to sample from an exponential amount of population before getting something interesting in the higher dimension. In the end, the odds of sampling anything useful would be close to 0. In general, our everyday intuition that applies very well to low-dimensional spaces does not generalize to high-dimensional spaces. Our algorithms tend to behave in strange, counterintuitive, and unpleasant ways. Methods which are optimal in low dimensions must be replaced by methods for high-dimensional versions of the problem.

3.3 A Naive Approach to Solve this problem: Monte Carlo Method

Monte Carlo offers a solution for sampling in higher dimensions. Let's use a to represent the region of circle and A to represent the region of square. We generate N samples $x_1, x_2, \ldots, x_N$ uniformly from area A. Assume I_i is the indicator variable where $I_i = 1$ if x_i is a sample from the region of interest (a) and $I_i = 0$ if x_i is a sample outside of the region of interest (a). It is clear that $E\left(I_i\right) = \frac{a}{A} = \mu = \left(\frac{1}{\text{factor}}\right)^d$, and for high dimensions this value decreases exponentially in dimension.

Lemma: Let $I_1, I_2, \ldots, I_N$ be *iid* indicator variables and $\mu = E\left(I_i\right)$, then one can show (using the Chernoff bound) that $\Pr\left(\sum_{i=1}^{N} I_i - \mu \geq \epsilon\mu\right) \leq \delta$ if $N \geq \frac{1}{\mu} * \frac{1}{\epsilon^2}$.

In other words, we require a number of samples that is exponential in the number of dimensions to have a given failure probability. As the dimension increases, the required N would quickly blow up. Therefore, this naive approach will not work But if we can sample from a relatively small region surrounding the area of interest a, sampling would become much easier.

4 Markov Chains: a Method to Solve this Problem

4.1 DNT Counting Problem:

A disjunctive normal form (DNF) is a boolean expression that is structured as an OR of clauses $C_1 \vee C_2 \vee \ldots \vee C_m$, where each clause C_i is an AND of literals $X_1 \wedge X_2 \wedge X_3 \wedge \ldots \wedge X_d$. For example, consider the following DNF.

$$(x_5 \wedge \overline{x_4} \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_4) \vee (\overline{x_{18}} \wedge x_{48}) \vee \ldots \vee (x_{72} \wedge x_{45})$$

It is clear that if any one of the clauses in DNF is satisfied, the whole DNF is satisfied. Therefore, we can easily find an assignment of the literals that satisfies the DNF - we simply have to find an assignment to satisfy one of the clauses. We will refer to an assignment that satisfies the DNF as a satisfiability assignment. Our problem is that we want to count how many satisfying assignments there are. Since we know the total number of possible assignments is 2^n - each of the n literals can be either true or false, this problem is hard.

4.2 A Naive Approach:

A naive approach would simply be to generate s random assignments and see how many assignments out of s are satisfying. Then we can use this fraction to compute the estimated counts of satisfying assignments. However, the problem is that the possibility of hitting a satisfying assignment is low. Note that the expected value of the indicator variable is the number of satisfiability assignments over $2^n : \mathbb{E}\left(I_i\right) = \frac{N(s)}{2^n}$. Therefore, the number of samples needed is $\frac{1}{\mathbf{E}(I_i)} = \frac{2^n}{N(s)}$. This is not desirable because $N(s)$ is usually small in comparison with 2^n. And we will require a huge amount of samples to get something interesting.

4.3 A Better Approach:

Firstly, we will look at the first clause and satisfies this clause. After satisfying this clause, we would then randomly assign the rest of clauses. This would make the entire assignment satisfying since we only need one true clause. Let's call this assignment SC_1. And we would repeat this approach for all the clauses. And we would form a duplicate union of all SC_i and sample from this set.

Put it into mathmatical equation, we define a universal set: $U = \{(i, a); a \in SC_i\}$ where SC_i includes the satisfiability assignments that satisfy clause i in DNF. Note that $N(s) = \bigcup_i SC_i$ and does not include repeated terms (common terms in SC_i s). However U has the repetitions, therefore $|U| = \sum_i |SC_i| \leq |N(s)|$.

In this case, we shrink down the pool from 2^n to a much smaller size. And it would be easier to sample in this circumstance. The detailed algorithm will be discussed next lecture.