CS7140: Advanced Machine Learning **Spring 2020**

Lecture 4: Monte Carlo Methods

Lecturer: Rose Yu *Scribes: Shreegowri Mattighatta Bhojanna*

In practice, probability distributions are very complicated that it is difficult to draw random samples and analytically compute their expectations using exact methods. Hence, we use algorithms that yield approximate solutions to the inference problem. In this chapter, we discuss sampling methods, which is one of the main families of approximate algorithms.

4.1 General problems addressed by Monte Carlo Methods

MC methods are computational techniques that make use of random numbers and aims to solve following problems.
Problem 1: Generate samples from $\{\mathbf{x^{(r)}}\}_{\mathbf{r=1}}^{\mathbf{R}}$ from a given probability distribution $\mathbf{P(x)}$
Problem 2: Estimate expectations of functions under the distribution

$$\mathbf{\Phi} = \mathbb{E}_{\mathbf{x\sim P(x)}}[\boldsymbol{\phi}(\mathbf{x})] \equiv \int \mathbf{P(x)}\boldsymbol{\phi}(\mathbf{x})\mathbf{dx} \tag{4.1}$$

We can solve the second problem by using the random samples $\{\mathbf{x^{(r)}}\}_{\mathbf{r=1}}^{\mathbf{R}}$ to give the estimator,

$$\hat{\mathbf{\Phi}} \equiv \frac{\mathbf{1}}{\boldsymbol{R}} \sum_{\boldsymbol{r}} \boldsymbol{\phi}(\mathbf{x^{(r)}}) \tag{4.2}$$

$$\mathbb{E}(\hat{\mathbf{\Phi}}) = \mathbf{\Phi} \tag{4.3}$$

As number of samples R increases, the variance of $\hat{\Phi}$ will decrease as σ^2/R, where σ^2 is the variance of Φ. So, regardless of the dimensionality of X, variance converges at the the rate of $1/R$. Also, the estimator is unbiased and consistent.

Why is sampling from P(x) hard?

- Assuming we can evaluate P(x) within a multiplicative constant, i.e., we can evaluate P*(x) such that P(x) = P*(x)/Z. We do not know the normalizing constant and computing Z is intractable as we need to visit every point in the space which requires d^N number of evaluations of P*(x).

$$Z = \int d^N \mathbf{x} \mathbf{P}^*(\mathbf{x}) \tag{4.4}$$

- Even when we know Z, we need to evaluate P(x) everywhere in order to obtain correct samples, which are usually from space where P(x) is big.

- We could consider computing expectation by drawing samples uniformly from the state space $\{\mathbf{x^{(r)}}\}_{\mathbf{r=1}}^{\mathbf{R}}$

and evaluating P*(x). Estimate Φ by

$$\Phi = \int d^N \mathbf{x} \phi(\mathbf{x}) \mathbf{P}(\mathbf{x})$$

$$\hat{\Phi} = \sum_{r=1}^{R} \phi(\mathbf{x^{(r)}}) \frac{\mathbf{P^*(x^{(r)})}}{\mathbf{Z_R}}$$

But high dimensional distribution is often concentrated in a small region of the state space. Hence, we need to have sufficiently large number of samples to get good estimate of Φ and these large number would be horribly huge in high dimensional space. So, if the distribution P(x) is not actually uniform, then uniform sampling also not useful in high dimensional problems.

Hence drawing samples from high dimensional distribution is difficult even if it is easy to evaluate.

4.2 Importance Sampling

Importance sampling is a general method used for estimating expectation of a function $\phi(\mathbf{x})$ and it is not a method for generating samples from P(x).
Assume that we can evaluate one-dimensional density P(x) within a multiplicative constant; thus we can evaluate P*(x) such that P(x)=P*(x)/Z. And we can generate samples from simpler sampler density Q(x) which we can evaluate within a multiplicative constant Z_Q.
When we generate samples from Q, samples where Q(x) is greater than P(x) will be over-represented and

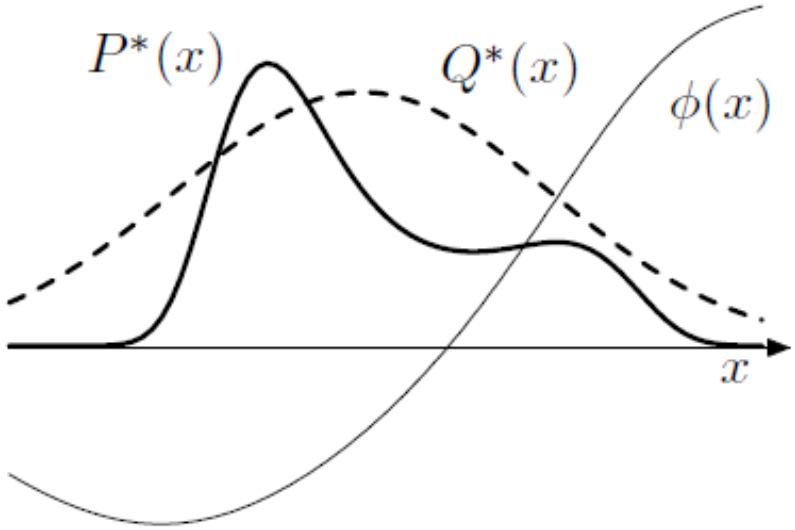

Figure 4.1: Functions involved in importance sampling

points where Q(x) is less than P(x) will be under-represented in the estimator $\hat{\Phi}$. We introduce weights to take into account that we are sampling from wrong distribution:

$$w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})} \tag{4.5}$$

and these weights are used to adjust the 'importance' of each point in estimator, thus:

$$\hat{\Phi} \equiv \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r} \tag{4.6}$$

But with importance sampling it is hard to estimate the reliability of our estimator $\hat{\Phi}$ as the variance of the estimator is unknown beforehand. If Q(x) is small in the region where $\mid \phi$(x)P*(x) $\mid$ is large, even after generating many samples (none of them have fallen in that region) the estimate of Φ would be drastically wrong and there would be no indication that the true variance of $\hat{\Phi}$ is large.

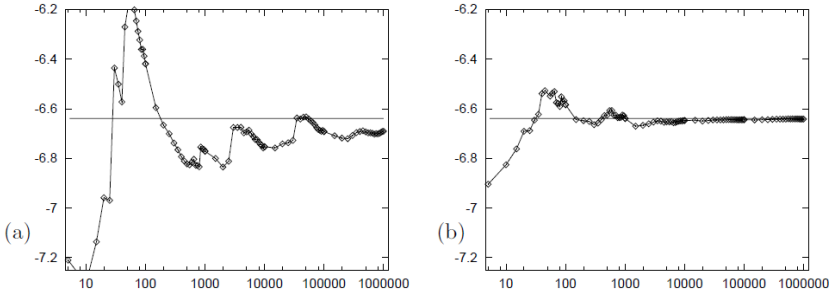

Figure 4.2: Importance sampling using (a) Gaussian (b) Cauchy sampler density

Example in figure 3.2 illustrates that importance sampling converges well using heavy tail sampler densities.

Problems with importance sampling in high-dimensions:

1. We need to obtain samples that lie in the typical set of P (all states in typical set have log probability close to the entropy), which may take a long time unless Q is good approximation of P.

2. Even if we obtain samples from typical set, the weights associated with those samples are likely to vary by large factors.

Thus an importance sampling estimate for a high-dimensional problem will very likely be dominated by a few samples with huge weights.

4.3 Rejection Sampling

We assume again complicated density $P(x)$ and simpler proposal density $Q(x)$, from which we can generate samples. We also assume we know constant c such that

$$cQ^*(x) > P^*(x), \forall x \tag{4.7}$$

We compute the area of the target region by sampling from larger region with a known area and recording fraction of samples that falls within target region.

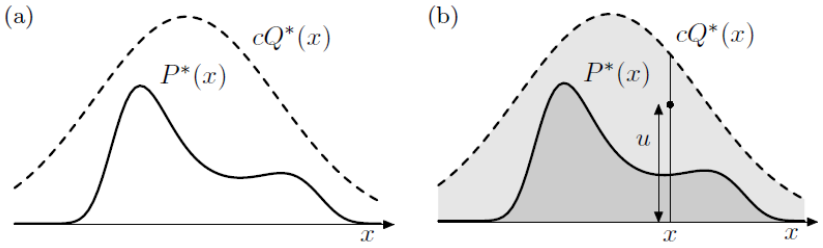

Figure 4.3: Rejection sampling (a) Functions involved (b) Samples generation process

Rejection Sampling Algorithm

1. Draw sample x from proposal density $Q(x)$
2. Evaluate $cQ^*(x)$ and generate random variable u from uniform distribution [0, $cQ*(x)$]
3. Evaluate $P^*(x)$ and reject x if $u > P^*(x)$; accept x otherwise and discard u

If Q is not a good approximation of P, for cQ to exceed P everywhere, c will have to be large which results in higher number of rejections. In high-dimensional problems, this force c to be so huge that acceptance becomes very rare. For example consider N-dimensional Gaussian distributions; proposal density Q with S.D σ_Q and target density with S.D σ_P. So far $cQ > P$ we need to set

$$c = \frac{(2\pi\sigma_Q^2)^{N/2}}{(2\pi\sigma_P^2)^{N/2}} = exp(Nln\frac{\sigma_Q}{\sigma_P}) \tag{4.8}$$

With N=1,000 and $\frac{\sigma_Q}{\sigma_P}$=1.01, $c \simeq 20,000$. So, the acceptance rate $1/c$ is exponentially small in N.
Hence, rejection sampling is not practical for sampling from high-dimensional distributions.

4.4 Metropolis-Hastings Methods

IS and RS works well only when Q(x) is similar to P(x). But for large and complex problems it is difficult to create such Q.
MHS algorithm makes use of Q which depends on the current state $x^{(t)}$. The proposal density $Q(x^{'}; x)$ can be any fixed density from which we can draw samples, it is not necessary to be similar to P(x).

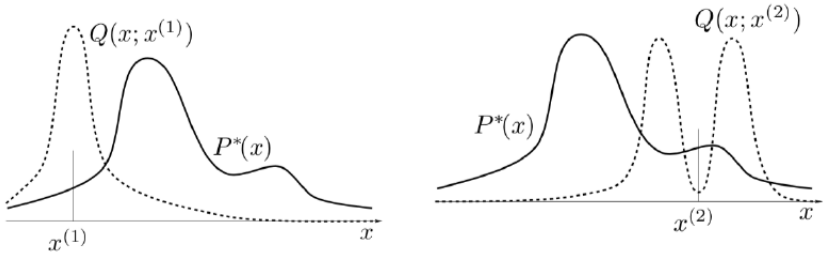

Figure 4.4: MHS in 1-D, here shape of $Q(x^{'}; x)$ changes as x changes

Metropolis-Hastings Algorithm

1. Draw a tentative new state $x^{'}$ from $Q(x^{'}; x^{(t)})$
2. Evaluate $a = \frac{P^*(x^{'})}{P^*(x^{(t)})} \frac{Q(x^{(t)};x^{'})}{Q(x^{'};x^{(t)})}$
   - If $a \geq 1$, accept new state, set $x^{(t+1)} = x^{'}$
   - Otherwise, reject new states, set $x^{(t+1)} = x^{(t)}$

If we consider symmetric Q, then MHS simply compares the value of P at two points.

4.4.1 Convergence of the Metropolis-Hastings method

MHS method is an example of Markov Chain Monte Carlo (MCMC) method. Which involves a Markov process in which a sequence of states $x^{(t)}$ is generated, each sample $x^{(t)}$ probability distribution depends on previous state $x^{(t-1)}$. Markov chain may have to run for considerable time to generate independent samples in P. So, it is difficult to assess convergence of MCMC and to quantify how long to wait to sample independent samples from P.

Many MHS implementations employ Q with a small length scale of ϵ relative to large length scale L of the probable region as shown in figure 3.5. For high-dimensional problems, large step from typical state may end up in less probable state, which are unlikely to be accepted. So, if ϵ is large rate of progress will be slow. On the other hand, when ϵ is random walk takes a long time to get anywhere.

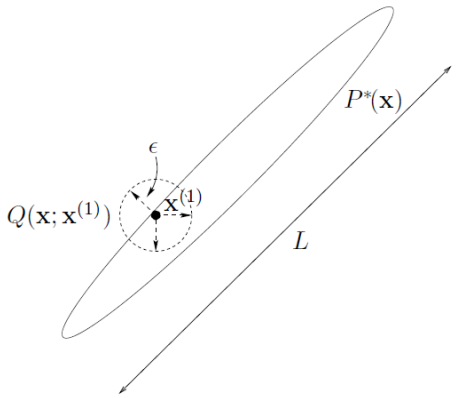

Figure 4.5: MHS method in 2-D

Rule of Thumb 4.1 *lower bound on number of iterations of a Metropolis method. If the largest length scale of the space of probable states is L, a Metropolis method whose proposal distribution generates a random walk with step size ϵ must be run for at least*

$$T \simeq (L/\epsilon)^2 \tag{4.9}$$

iterations to obtain an independent sample. This time is increased by a factor $1/f$ if only fraction f of the steps are accepted on average.

[DM03]

When we use simple spherical proposal distributions, we will need at least $T \simeq (\sigma^m ax/\sigma^m in)$ iterations to obtain an independent sample, where $\sigma^m ax$ and $\sigma^m in$ are the longest and shortest length scales of target distribution.

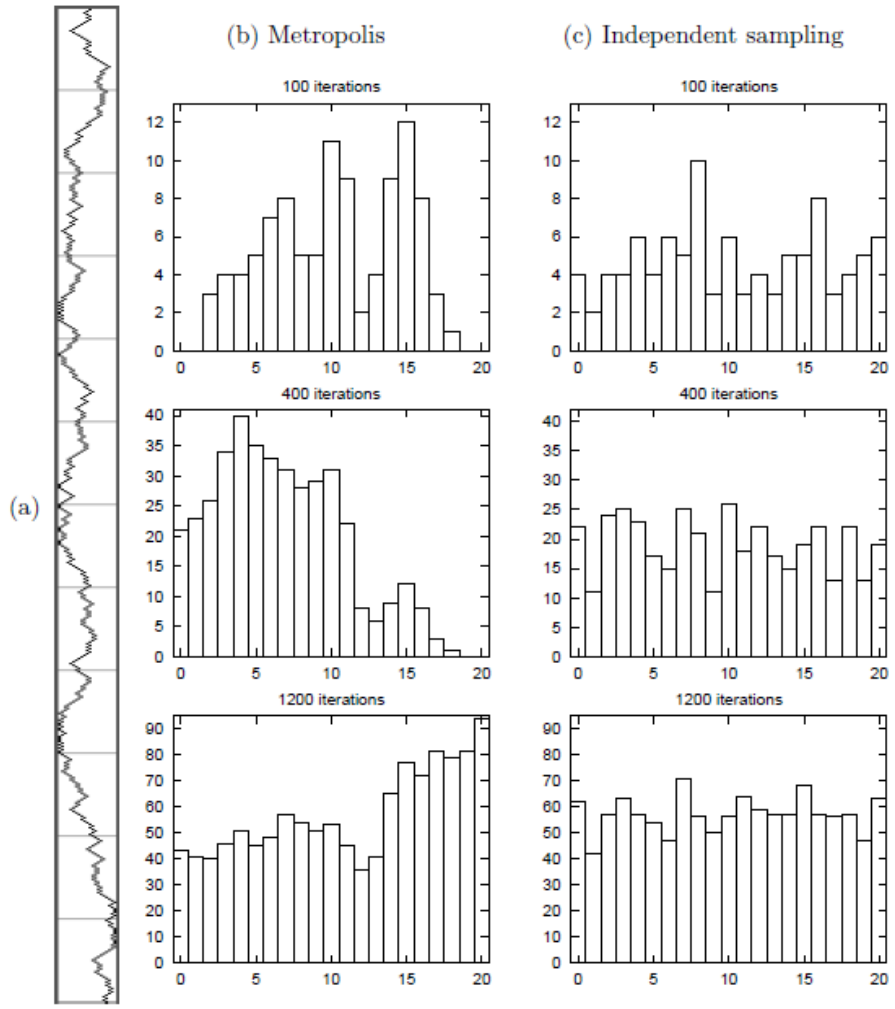

Figure 4.6: MHS method for toy problem. (a) State seq for t=1,..,600. Horizontal = states 0 to 20, Vertical = time 1 to 600 (b) MHS (c) Resulting histograms when successive points drawn independently from P

Figure 3.6 illustrates random walk behaviour when samples are drawn from

$$P(x) = \begin{cases} 1/21 & x \in (0, 1, ..., 20) \\ 0 & otherwise \end{cases} \tag{4.10}$$

The proposal distribution is

$$Q(x^{'}; x) = \begin{cases} 1/2 & x^{'} = x \pm 1 \\ 0 & otherwise \end{cases} \tag{4.11}$$

Independent samples are generated only by simulating for about four hundred iterations per independent sample. However, HMS does not converge well even after 1,200 iterations. It is important to reduce this random walk behaviour.

4.5 Gibbs Sampling

GS also known as 'heat bath method' or 'Glauber dynamics' used for sampling from at least 2-D distributions. Since sampling from joint distribution P(x) is hard, we sample from conditional distribution $P(x_i \mid \{x_j\}_{j \neq i})$, which is tractable.

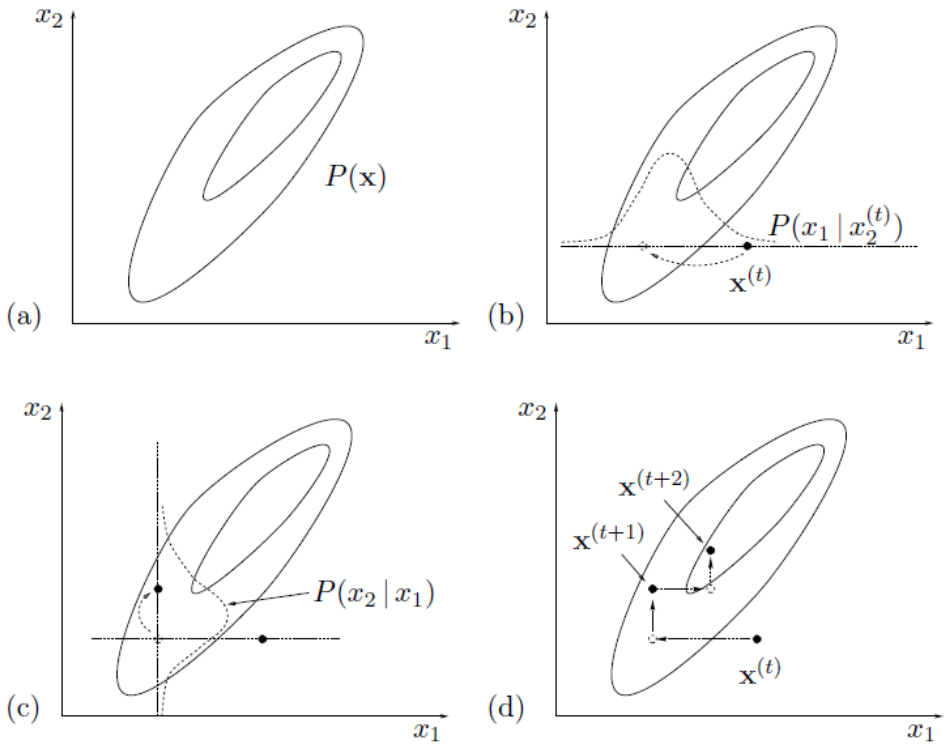

Figure 4.7: Gibbs Sampling (b) and (c) Sampling one parameter at a time (d) After couple of iterations

Starting from current state $x^{(t)}$, x_1 is sampled from conditional density $P(x_1 \mid x_2^{(t)})$. A sample x_2 is drawn from $P(x_2 \mid x_1)$, which finally gives us new state $x^{(t+1)}$.

$$\begin{aligned} x_1^{(t+1)} &\sim P(x_1 \mid x_2^{(t)}, x_3^{(t)}, \ldots, x_K^{(t)}) \\ x_2^{(t+1)} &\sim P(x_2 \mid x_1^{(t+1)}, x_3^{(t)}, \ldots, x_K^{(t)}) \\ x_3^{(t+1)} &\sim P(x_3 \mid x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_K^{(t)}), \text{ etc.} \end{aligned}$$

Figure 4.8: Single iteration of a system with K variables

Like MHS, GS also suffers from slow random walk behaviour unless we have separable P(x).

4.6 Practicalities

Even though MCMC methods are able to sample from right distribution,

- They require a very long time to converge. We can compute simple lower bound on convergence rate but this is difficult to do and upper bound are of no practical use. It is not easy to decide on how much computation to spend on MCMC to find good solution.
- Detecting convergence in running simulation is also a difficult task.
- Using methods such as Hamiltonian Monte Carlo method, Over-relaxation, and Simulated Annealing, we can speed up convergence time and time between independent samples.
- For GS in big problems, it may be efficient to sample groups of variables jointly using several proposal distributions.
- Variance of estimator $\hat{\Phi}$ depends only on the number of independent samples R. We can decide on number of independent samples by deciding on how much precision we need in our estimate as we can estimate Φ to a precision of $\sigma/\sqrt{R}$.
- The three MCMC strategies to make use of limited computer resources are shown in figure 3.9. MCMC simulations including initial period (where we adjust parameters of the simulation), burn-in period(during which we hope simulation converges to the desired distribution) and sampling period (where the samples are recorded).
    1. Make one long run, obtaining all R samples from it. Which has the best chance of convergence.
    2. Make a few medium-length runs with different initial conditions, obtaining some samples from each. This is popular strategy as it meets in the mid between 1 and 3.
    3. Make R short runs, each starting from a different random initial condition, with the only state that is recorded being the final state of each simulation. So, correlation between the samples are smaller.

Figure 4.9: MCMC Strategies

4.7 Histories

- Stanislaw Ulam and John von Neumann developed Importance Sampling and Rejection Sampling along with many other MC methods.
- Nick Metropolis along with Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller proposed Metropolis algorithm in 1953.
- Stuart Geman and Donald Geman proposed Gibbs Sampling.
- MCMC was named in Top 10 most influential algorithms of 20^{th} century by CiSE Journal in the year 2000.

References

[DM03] David J.C. MacKay, "Information Theory, Inference, and Learning Algorithms", *Cambridge University press*, 2003, pp. 357–382.

[KF09] D. Koller and N. Friedman, "Probabilistic Graphical Models: Principles and Techniques", *MIT press*, 2009, pp. 487–523.