13 Monte Carlo Methods

Monte Carlo is an umbrella term referring to a set of numerical techniques for solving one or both of the following problems:

1. Approximating expected values that cannot be compute in closed-form
2. Sampling from distributions for which is a simple sampling algorithm is not available.

Recall that mathematical expectation of a function $\phi(\mathbf{x})$ of a continuous variable $\mathbf{x}$ with respect to a distribution $p(\mathbf{x})$ is defined as:

$$E_{p(\mathbf{x})}[\phi(\mathbf{x})] \equiv \int p(\mathbf{x})\phi(\mathbf{x})d\mathbf{x} \tag{13.1}$$

Monte Carlo methods approximate this integral by drawing N independent samples from $p(\mathbf{x})$

$$\mathbf{x}_i \sim p(\mathbf{x}) \tag{13.2}$$

and then approximating the integral by the weighted average:

$$E_{p(\mathbf{x})}[\phi(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x}_i) \tag{13.3}$$

Estimator properties. This estimate is unbiased:

$$E_{p(\mathbf{x}_{1:N})}\left[\frac{1}{N}\sum_i \phi(\mathbf{x}_i)\right] = \frac{1}{N}\sum_i E_{p(\mathbf{x}_i)}[\phi(\mathbf{x}_i)] = \frac{1}{N}NE_{p(\mathbf{x})}[\phi(\mathbf{x})] = E_{p(\mathbf{x})}[\phi(\mathbf{x})] \tag{13.4}$$

Furthermore, the variance of this estimate is inversely proportional to the number of samples:

$$\text{var}_{p(\mathbf{x}_{1:N})}\left[\frac{1}{N}\sum_i \phi(\mathbf{x}_i)\right] = \frac{1}{N^2}\sum_i \text{var}_{p(\mathbf{x}_{1:N})}[\phi(\mathbf{x}_i)] = \frac{1}{N^2}N\text{var}_{p(\mathbf{x}_i)}[\phi(x_i)] = \frac{1}{N}\text{var}_{p(\mathbf{x})}[\phi(\mathbf{x})] \tag{13.5}$$

Hence, the more samples we get, the better our estimate will be. The weak law of large numbers tells us that, in the limit the estimator will converge to the true value.

Dealing with unnormalized distributions. We often wish to compute the expected value of a distribution for which evaluating the normalization constant is difficult. For example, the posterior distribution over parameters $\mathbf{w}$ given data D is:

$$p(\mathbf{w} \mid D) = \frac{p(D \mid \mathbf{w})\, p(\mathbf{w})}{p(D)} \tag{13.6}$$

The posterior mean and covariance ($\bar{\mathbf{w}} = E[\mathbf{w}]$ and $E[(\mathbf{w} - \bar{\mathbf{w}})(\mathbf{w} - \bar{\mathbf{w}})^T]$) can be useful to understand this posterior, i.e., what we believe the parameter values are "on average," and how much

uncertainty there is in the parameters. The numerator of $p(\mathbf{w} \mid D)$ is typically easy to compute, but $p(D)$ entails an integral which is often intractable, and thus must be handled numerically.

More generally, we can pose the problem as estimating the expected value with respect to a distribution $p(\mathbf{x})$ that can only be evaluated up to a multiplicate constant Z, e.g., P^*, where

$$p(\mathbf{x}) \equiv \frac{1}{Z} P^*(\mathbf{x}) \,, \qquad Z = \int P^*(\mathbf{x}) d\mathbf{x} \tag{13.7}$$

Monte Carlo methods will allow us to handle distributions of this form.

13.1 Sampling Gaussians

We begin with algorithms for sampling from a Gaussian distribution. For the simple 1-dimensional case, $x \sim \mathcal{N}(0, 1)$, there is well-known algorithm called the Box-Muller Method that is based on an approach called rejection sampling. It is implemented in MATLAB in the command `randn`.

For a general 1D Gaussian, $x \sim \mathcal{N}(\mu, \sigma^2)$, we sample a variable $z \sim \mathcal{N}(0, 1)$, and then set $x = \sigma z + \mu$. You should be able to show that x has the desired mean and variance. For the multi-dimensional case, $\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$, each element is independent and Gaussian: $x_i \sim \mathcal{N}(0, 1)$ and so each element can be sampled with `randn`.

To sample from a Gaussian with general mean vector $\boldsymbol{\mu}$ and variance matrix $\boldsymbol{\Sigma}$ we first sample $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, and then set $\mathbf{x} = \mathbf{L}\mathbf{z} + \boldsymbol{\mu}$, where $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$. We can compute $\mathbf{L}$ from $\boldsymbol{\Sigma}$ by the Cholesky Factorization of $\boldsymbol{\Sigma}$, which must be positive definite. Then we have

$$E[\mathbf{x}] = E[\mathbf{L}\mathbf{z} + \boldsymbol{\mu}] = \mathbf{L}E[\mathbf{z}] + \boldsymbol{\mu} = \boldsymbol{\mu} \,, \tag{13.8}$$

and

$$E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = E[\mathbf{L}\mathbf{z}(\mathbf{L}\mathbf{z})^T] = \mathbf{L}E[\mathbf{z}\mathbf{z}^T]\mathbf{L}^T = \mathbf{L}\mathbf{L}^T = \boldsymbol{\Sigma} \,. \tag{13.9}$$

13.2 Sampling Categorical Distributions

Another common distribution we may want to sample from is a categorical distribution over a discrete set of possible states. Suppose we have a categorical distribution over K mutually exclusive states with probabilites P_1, P_2, ..., P_K, where of course $P_j \geq 0$ and $\sum_{j=1}^{K} P_j = 1$. As is common in many samplers, we first draw a random rample from a uniform distribution, for which good algorithms exist. We then transform the sample using the inverse cumulative distribution function of the categorical distribution to obtain a fair sample. Here is a description of the algorithm using (Matlab) pseudocode:


```
// Construct a CDF from vector of probabilities
CDF = cumsum(prob);
// extract a sample from uniform distribution on [0,1]
z = rand(1, 1);
// use first index j for which CDF(j) > z
m = find(CDF > z);
return m(1)
```

13.3 Importance Sampling

In most situations it is difficult to sample from the desired distribution $p(\mathbf{x})$, but we can sample from a similar distribution $q(\mathbf{x})$. Importance sampling is a technique that allows one to approximate expectation with respect to $p(\mathbf{x})$ by sampling from $q(\mathbf{x})$. The only requirement on q is that it have the same support as p, i.e., q is nonzero everywhere that p is nonzero.

Importance sampling is based on the following equality:

$$E_{q(\mathbf{x})}\left[\frac{p(\mathbf{x})}{q(\mathbf{x})}\phi(\mathbf{x})\right] = \int \frac{p(\mathbf{x})}{q(\mathbf{x})}\phi(\mathbf{x})q(\mathbf{x})d\mathbf{x} \tag{13.10}$$

$$= \int \phi(\mathbf{x})p(\mathbf{x})dx \tag{13.11}$$

$$= E_{p(\mathbf{x})}\left[\phi(\mathbf{x})\right] \tag{13.12}$$

In other words, we can compute the desired expectation by sampling values $\mathbf{x}_i$ from $q(\mathbf{x})$, and then computing

$$E_q\left[\frac{p(\mathbf{x})}{q(\mathbf{x})}\phi(\mathbf{x})\right] \approx \frac{1}{N}\sum_i \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)}\phi(\mathbf{x}_i) \tag{13.13}$$

It often happens that p and/or q are known only up to multiplicative constants. That is,

$$p(\mathbf{x}) \equiv \frac{1}{Z_p}P^*(\mathbf{x}) \quad \text{and} \quad q(\mathbf{x}) \equiv \frac{1}{Z_q}Q^*(\mathbf{x}) \tag{13.14}$$

where P^* and Q^* are easy to evaluate but the constants Z_p and Z_q are not.

Then we have:

$$E_{p(\mathbf{x})}[\phi(\mathbf{x})] = \int \frac{\frac{1}{Z_p}P^*(\mathbf{x})}{\frac{1}{Z_q}Q^*(\mathbf{x})}\phi(\mathbf{x})q(\mathbf{x})dx = \frac{Z_q}{Z_p}E_{q(\mathbf{x})}\left[\frac{P^*(\mathbf{x})}{Q^*(\mathbf{x})}\phi(\mathbf{x})\right] \tag{13.15}$$

and so it remains to approximate $\frac{Z_q}{Z_p}$. If we substitute $\phi(\mathbf{x}) = 1$, the above formula states that

$$\frac{Z_q}{Z_p}E_{q(x)}\left[\frac{P^*(x)}{Q^*(x)}\right] = 1 \tag{13.16}$$

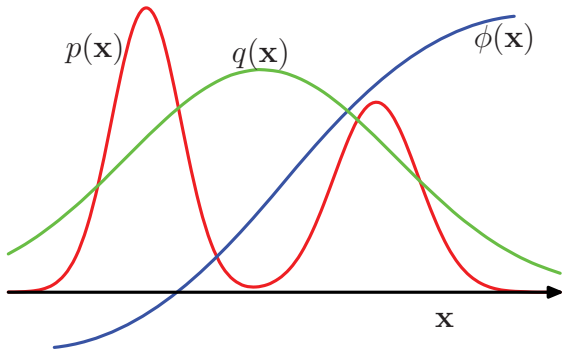

Figure 13.1: Importance sampling can be used to sample complicated distributions like this bimodal $p(\mathbf{x})$. Here we draw samples from the simpler unimodal distribution $q(\mathbf{x})$. This will produce many samples with a low weight since $q(\mathbf{x})$ is large where $p(\mathbf{x})$ is near zero. But $q(\mathbf{x})$ has ample probability mass around the modes of $p(\mathbf{x})$, so it is a reasonable choice. When $q(\mathbf{x})$ had very little probability mass around modes of $p(\mathbf{x})$, then estimates obtained from importance sampling will have high variance. (Figure from *Pattern Recognition and Machine Learning* by Chris Bishop.)

and so $\frac{Z_p}{Z_q} = E_{q(\mathbf{x})}[\frac{P^*(\mathbf{x})}{Q^*(\mathbf{x})}]$. Thus we have:

$$E_{p(\mathbf{x})}[\phi(\mathbf{x})] = \frac{E_{q(\mathbf{x})}\left[\frac{P^*(\mathbf{x})}{Q^*(\mathbf{x})}\phi(\mathbf{x})\right]}{E_{q(\mathbf{x})}\left[\frac{P^*(\mathbf{x})}{Q^*(\mathbf{x})}\right]} \tag{13.17}$$

Hence, the importance sampling algorithm is:

1. Sample N values $\mathbf{x}_i \sim q(\mathbf{x}_i)$
2. Compute
$$w_i = \frac{P^*(\mathbf{x}_i)}{Q^*(\mathbf{x}_i)} \tag{13.18}$$
3. Estimate the expected value
$$E[\phi(\mathbf{x})] \approx \frac{\sum_i w_i \phi(\mathbf{x}_i)}{\sum_i w_i} \tag{13.19}$$

The importance sampling algorithm will only work well when $q(\mathbf{x})$ is sufficiently similar to the function $p(\mathbf{x})|\phi(\mathbf{x})|$. Put more concretely, the variance of the estimator grows as the dissimilarity between $q(\mathbf{x})$ and $p(\mathbf{x})|\phi(\mathbf{x})|$ grows (and is minimized when they are equal). An alternative is to use the MCMC algorithm to draw samples directly from $p(\mathbf{x})$, as described below.

13.4 Markov Chain Monte Carlo (MCMC)

MCMC is a general algorithm for sampling from any distribution, such as the posterior distribution over model parameters $\mathbf{w}$. It is an iterative algorithm that, given a sample $\mathbf{x}_t \sim p(\mathbf{x})$, modifies that sample to produce a new sample $\mathbf{x}_{t+1} \sim p(\mathbf{x})$. This modification is done using a proposal

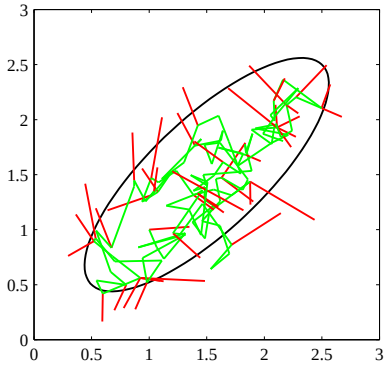

Figure 13.2: MCMC applied to a 2D Gaussian with a proposal distribution consisting of a circular Gaussian centered on the previous sample. Green lines are accepted proposals. Red lines are rejected. (Figure from *Pattern Recognition and Machine Learning* by Chris Bishop.)

distribution $q(\mathbf{x}'|\mathbf{x})$, that, given a $\mathbf{x}$, randomly selects a "mutation" to $\mathbf{x}$. The proposal distribution may be almost anything, and it is up to the user of the algorithm to choose this distribution. A common choice is a Gaussian centered at $\mathbf{x}$: $q(\mathbf{x}'|\mathbf{x}) = \mathcal{N}(\mathbf{x}'|\mathbf{x}, \sigma^2\mathbf{I})$. The entire algorithm is

select initial point $\mathbf{x}_1$
$t \leftarrow 1$
loop
  Sample $\mathbf{x}' \sim q(\mathbf{x}'|\mathbf{x}_t)$
  $\alpha \leftarrow \frac{P^*(\mathbf{x}')}{P^*(\mathbf{x}_t)} \frac{q(\mathbf{x}_t|\mathbf{x}')}{q(\mathbf{x}'|\mathbf{x}_t)}$
  Sample $u \sim \text{Uniform}[0, 1]$
  if $u \leq \alpha$ **then**
    $\mathbf{x}_{t+1} \leftarrow \mathbf{x}'$
  else
    $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t$
  end if
  $t \leftarrow t + 1$
end loop

Amazingly, it can be shown that, if $\mathbf{x}_1$ is a sample from $p(\mathbf{x})$, then every subsequent $\mathbf{x}_t$ is also a sample from $p(\mathbf{x})$, if they are considered in isolation. The samples are correlated to each other via the Markov Chain, but the marginal distribution of any individual sample is $p(\mathbf{x})$.

So far we assumed that $\mathbf{x}_1$ is a sample from the target distribution, but obtaining this first sample is difficult. Instead, we must perform a process called **burn-in**: we initialize with any $\mathbf{x}_1$, and then discard the first T samples obtained by the algorithm. If we pick T large enough we are guaranteed that remaining samples are fair samples from the target. There is no exact method for determining a sufficient T however, and so heuristics are used.

