**CS281: Advanced ML** **November 13, 2017**

# Lecture 19: Monte Carlo Basics

*Lecturer: Sasha Rush* *Scribes: Chang Liu, Jiafeng Chen, Alexander Lin*

## 19.1 History

We started with Monte Carlo in the past few lectures. The main method is to use draw samples from a proposal distribution and take sample average to approximate expectations.

$$\begin{aligned}\mathbb{E}_{y\sim p(y|x)}[f(y)] &= \int p(y|x)f(y)dy \\ &\approx \frac{1}{N}\sum_{n=1}^{N} f(\tilde{y}^{(n)})\end{aligned}$$

where $\tilde{y}^{(n)} \sim p(y|x)$. This approach requires the ability to sample $y \sim p(y|x)$.

## 19.2 Univariate Case

Let's start at the beginning: We know $F(x) = p(y \leq x)$ and it is univariate.
Assume we can compute $F^{-1}(u)$, then

$$p(F^u \leq x) = p(u \leq F(x)) = F(x)u$$

where $F$ is the CDF of $u \sim \text{Unif}(0,1)$. However, this works only for univariate case. What this means is that, is we wish to sample $y \sim F$ for some known CDF $F$, we need only to sample $U \sim \text{Unif}$ and transform the sample of uniform random variables through $F^{-1}$ to get a sample of $y$. Formally,

$$U \sim \text{Unif} \implies F^{-1}(U) \sim F$$

(This is known as the probability integral transform).

## 19.3 Normal Samples

We pursue the same strategy as in the univariate case. Given Uniform samples, we wish to apply some transform to obtain a sample of multivariate random variables of some distribution we are interested in. We execute this strategy for Normal random variables (this is known as the Box-Muller transformation). The upshot here is that to sample Normal random variables, we need only to sample Uniform random variables, which is much easier to do.

**Box-Muller** Sample $z_1, z_2 \sim \text{Unif}[-1,1]$. Discard the points outside of the unit circle, so our sample of $\{(z_1, z_2)\}$ is uniform on the unit disk. We would like to transform each $(z_1, z_2)$ into some $(x_1, x_2) \sim \mathcal{N}(0, I)$. We want to find the right transform such that the Jacobian makes the following hold

$$\overbrace{p(x_1, x_2)}^{\text{Normal PDF}} = \overbrace{p(z_1, z_2)}^{\text{Unif disk PDF}} \left|\frac{\partial(z_1, z_2)}{\partial(x_1, x_2)}\right|.$$

We may check that

$$x_1 = z_1 \left( \frac{-2 \log r^2}{r^2} \right)^{1/2}$$
$$x_2 = z_2 \left( \frac{-2 \log r^2}{r^2} \right)^{1/2},$$

where $r^2 = z_1^2 + z_2^2$, is the desired transform.

## 19.4 Rejection Sampling

Assume that we have access to the PDF $p(x)$ or the unnormalized PDF $\tilde{p}(x)$. The idea is to pick a **guide function** (valid PDF) $q(x)$ that is similar to $p$ and easy to compute. We also pick a **scale** $M$. We require that

$$Mq(x) > p(x)$$

for all $x$ and that we have access to $p(x)$ or $\widetilde{p}(x)$. The algorithm is as follows:

1. Sample $x_n \sim q(x)$
2. Draw $u \sim \text{Unif}[0, 1]$
3. If $u < \frac{p(x_n)}{Mq(x_n)}$, then keep $x_n$; otherwise, rerun from 1.

The interpretation is simple. The algorithm "graphs" $p(x)$ and the bounding $Mq(x)$ on a board, then proceeds to throw darts at the board and accepting those darts that hit below $p(x)$. The same algorithm works even if $\tilde{p}$ is unnormalized, since we have the degree of freedom to choose $M$ and thus absorb the normalizing constant.

This method works with $\frac{\tilde{p}(x)}{Z} = p(x)$. But why should it work?
**ANS** This works because for whatever guide function we pick, we can write that guide function to be:

$$\widetilde{M} = ZM$$

### 19.4.1 Proof of Rejection Sampling

$$\begin{aligned}
p(x \le x_0 | x \text{is accepted}) &= \frac{\int_{-\infty}^{x_0} \int_0^1 q(x)\, 1(u \le \frac{\tilde{p}(x)}{Mq(x)}) du dx}{\int_{-\infty}^{\infty} \int_0^1 q(x)\, 1(u \le \frac{\tilde{p}(x)}{Mq(x)}) du dx} \\
&= \frac{\frac{1}{M} \int_{-\infty}^{x_0} \tilde{p}(x) dx}{\frac{1}{M} \int_{-\infty}^{\infty} \tilde{p}(x) dx} \qquad \text{the denominator is probability the acceptance} \\
&= \int_{-\infty}^{x_0} p(x) dx \\
&= p(x \le x_0)
\end{aligned}$$

## 19.5 Examples for Rejection Sampling

1. In Bayesian statistics, we often encounter the following problem. We are given $p(\theta)$, $p(x|\theta)$, and we wish to sample from the posterior $p(\theta|x)$. We can compute the unnormalized posterior $\tilde{p}(\theta|x) = p(x|\theta)p(\theta)$. Set $q(\theta) = p(\theta)$. Choose $M = p(x|\hat{\theta})$, where $\hat{\theta}$ is the maximum likelihood estimator. Then $Mq \geq p$. Thus, rejection sampling says the followng. Sample from the prior $q(\theta) = p(\theta)$, roll a uniform $u$, and keep those $\theta \sim p(\theta)$ with

$$u \leq \frac{p}{Mq} = \frac{p(x|\theta)}{p(x|\hat{\theta})} \leq 1.$$

2. Let $p \sim \mathcal{N}(0, \sigma_p^2 I)$ and $q \sim \mathcal{N}(0, \sigma_q^2 I)$ where $\sigma_q^2 > \sigma_p^2$. Pick

$$M = (\sigma_q/\sigma_p)^D,$$

where $D$ is the dimension of the Multivariate Normal. Note that $M$ becomes very large when $D$ becomes large, and so rejection sampling may be inefficient when $D$ is large. If we imagine $M$ as the metric of which to boost the Gaussians to make random sampling work, due to the known geometry of Gaussian distributions we can imagine as $D$ increase there will be more and more "space" between $p$ and $q$ to fill, thus making Random Sampling quite difficult.

## 19.6 Importance Sampling

We want to approximate the expectation

$$\mathbb{E}_{x\sim p}(f(x)) = \int f(x)p(x)\,dx.$$

So far, we can sample a bunch of points—via, say, rejection sampling—from $p$ and calculate a sample mean to approximate the true expectation. If the structure of $p$ and the structure of $f$ are very different, so Monte Carlo methods so far might be inefficient, since it samples from high density areas in $p$, which may have very low values of $f$, and the Monte Carlo may miss areas with high values of $f$ but low probability of happening.

Consider the integral

$$\int q(x)\frac{p(x)}{q(x)}f(x)\,dx = \mathbb{E}_q\left(f(x)\frac{p(x)}{q(x)}\right) = \mathbb{E}_p(f(x)).$$

We may now apply the same Monte Carlo trick to sample from $q$ and take the sample mean of $f(x)p(x)/q(x)$.

What is the benefit of using $q$? Since we can choose $q$ to be closer to $f$, then more of the sample we choose would be around high values of $f$. Here we don't need to wait for some low-probability tail event in $p$ to happen in order to get reliable estimates of $E_p(f(x))$. Instead, we can directly look at the tail events via $q$ and weight the data appropriately using $p/q$ to still maintain asymptotic convergence.
How exactly do we choose $q$? We want to minimize the variance of $f(x)p(x)/q(x)$ when $x \sim q$, since this allows for faster convergence. Then

$$\mathrm{Var}\left(\frac{f(x)p(x)}{q(x)}\right) = \mathbb{E}\left[\left(\frac{f(x)p(x)}{q(x)}\right)^2\right] - \underbrace{\left(\mathbb{E}\left(\frac{f(x)p(x)}{q(x)}\right)\right)^2}_{\text{constant eventually}}$$

$$\begin{aligned}\mathbb{E}\left(\frac{f(x)p(x)}{q(x)}\right)^2 &\geq \left(\mathbb{E}_q\left(\frac{p(x)|f(x)|}{q(x)}\right)\right)^2 && \text{(Jensen's inequality)}\\ &= \left(\int p(x)|f(x)|\,dx\right)^2\end{aligned}$$

We minimize the lower bound via Jensen's inequality (similar to what we did in variational inference). The optimal $q$ is chosen via

$$q^* = \frac{|f(x)|p(x)}{\int |f(x)|p(x)\,dx}.$$

It may be difficult to normalize $q^*$ in practice, however.

$$\begin{aligned}\mathbb{E}_p[f(x)] &= \int p(x)f(x)dx \\ &\approx \frac{1}{N}\sum_{n=1}^{N} f(\tilde{x}^{(n)}\end{aligned}$$

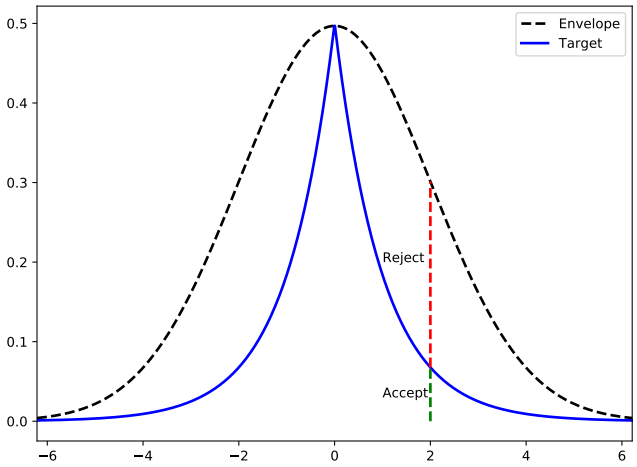
where $\tilde{x}^{(n)} \sim p(x)$

# 1 Exercise: Rejection sampling in python

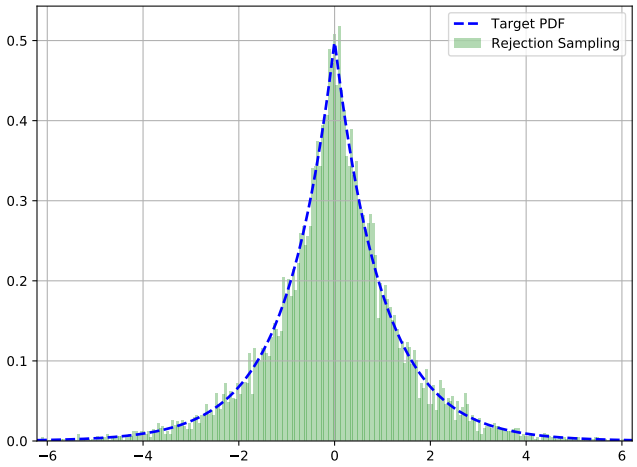
Given this double gamma PDF: $f(x;\alpha) = \frac{1}{2\Gamma(\alpha)}|x|^{\alpha-1}e^{-|x|}$

approximate it using rejection sampling. Use a normal distribution as an envelope. Plot the approximation against the targeted PDF.

```
import random
import scipy as sc
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import pandas as pd
%matplotlib inline

# Target = double gamma distribution
# Envelope (what we will use to approxmiate with) = normal distribution
dg = stats.dgamma(a=1)
norm = stats.norm(loc=0, scale=2)

# Generate samples for PDF
x = np.linspace(min(dg.ppf(0.001), norm.ppf(0.001)),
max(dg.ppf(0.999), norm.ppf(0.999)), 1000)
dg_samples = dg.pdf(x)
norm_samples = norm.pdf(x)

# Find scaling constant for envelope
M = max(dg_samples / norm_samples)

# Plot
df = pd.DataFrame({'Target': dg_samples, 'Envelope': M * norm_samples}, index=x)
ax = df.plot(style=['--', '-'], color=['black', 'blue'],
figsize=(8,6), linewidth=2.0)
ax.plot((2, 2), (0, dg.pdf(2)), 'g--', linewidth=2.0)
ax.plot((2, 2), (dg.pdf(2), M * norm.pdf(2)), 'r--', linewidth=2.0)
ax.text(1.0, 0.20, 'Reject')
ax.text(1.0, 0.03, 'Accept')
```




```
# lests estimate this posterior using rejection sampling
def rejection_sampling():
while True:
# Re-use global parameters from above
x = np.random.normal(0, 2)
envelope = M * norm.pdf(x)
p = np.random.uniform(0, envelope)
if p < dg.pdf(x):
return x

# Generation samples from rejection sampling algorithm
samples = [rejection_sampling() for x in range(10000)]

# Plot Histogram vs. Target PDF
df['Target'].plot(color='blue', style='--', figsize=(8,6), linewidth=2.0)
pd.Series(samples).hist(bins=300, normed=True, color='green',
alpha=0.3, linewidth=0.0)
plt.legend(['Target PDF', 'Rejection Sampling'])
```


Target PDF
Rejection Sampling
0.5
0.4
0.3
0.2
0.1
0.0
−6
−4
−2
0
2
4
6