MONTE PYTHON - OPEN SOURCE MONTE CARLO DEMOS IN PYTHON: REJECTION SAMPLING

JULIA MILTON

1. Introduction. Monte Carlo simulations are a class of useful tools to predict the outcome of problems where random variables are present, or that are difficult to solve through other approaches. Many papers have been written that explain and expand the mathematical concepts behind Monte Carlo strategies, however, there are fewer accessible implementations and visual explanations of these methods. Much of the existing code online for methods such as importance sampling and Markov Chain Monte Carlo is also produced in MATLAB, which is a powerful language for analysis and visualization, but is proprietary software that users must purchase a license to use.

The motivation behind this project was to make the information that is necessary to understand and implement some foundational Monte Carlo techniques more accessible to people who do not have extensive backgrounds in mathematics and/or statistics, but who may benefit from using Monte Carlo methods in their work. While rigorous math and notation is essential for developing these techniques and explaining them in academic papers, the intent of each of the demos is to build intuition and provide visualizations of the techniques. Therefore, the mathematical formulations of the each of the methods is provided only briefly, with the bulk of the content of each demo consisting of description of the methods and code that the user can change key parameters in and run to see the results. The simulations are also written in Python - an open-source language - to make the information more accessible.

The overall objective of this project was to develop open source, visual demos for some of the key concepts in Monte Carlo simulations. Included in these demos currently are: rejection sampling, importance sampling, and Markov Chain Monte Carlo. All of the code for these demos is written in Python and is available on GitHub at: https://github.com/juliamilton/Monte-Carlo-Demos

2. Rejection Sampling. Rejection sampling is used to generate samples from a distribution where we know and can express the density as a function (call it the "target distribution", or $p(x)$) but do not have a way to draw random samples from that distribution. Why would that occur? Many distributions with commonly known densities (such as the normal, exponential, and gamma distributions) have well-documented methods to sample from them and there are frequently built-in functions in many programming languages to do this. However, these built-in functions rely on methods such as the inverse-transform method and the Box-Mueller method to generate samples. For complicated distributions, however, we may not be able to use the inverse-transform or Box-Mueller methods, or they may not be very efficient. Rejection sampling can be used in this case.

3. Basic idea of the method. We "cover" the target distribution $p(x)$ with another "proposal" distribution, $f(x)$, that we know how to sample from. (See Fig. 1.) We then sample from $f(x)$ and reject the point if it is above $p(x)$. Although we can use a uniform distribution or any other distribution that *completely covers* $p(x)$, it can be more efficient to choose our proposal distribution $f(x)$ wisely. For this, we

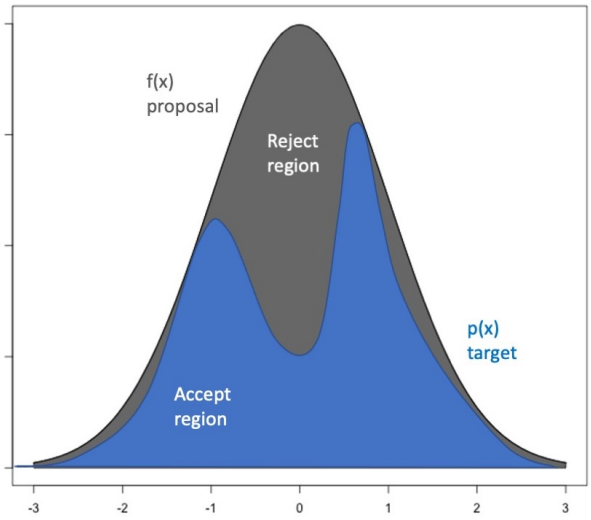

FIG. 1. *Covering the target distribution, $p(x)$, with the proposal distribution, $f(x)$, in rejection sampling.*

can multiply the proposal distribution by a scaling factor, k, so that it just covers $p(x)$. The scaling factor k is calculated as:

$$k = max(\frac{p(x)}{f(x)})$$

To determine if we will accept or reject the point, we first calculate a probability ratio that compares the probability that the point came from our target distribution to the probability that it came from the proposal distribution:

$$\alpha = \frac{p(x)}{k * f(x)}$$

Next, we draw a point between 0 and 1 randomly, $u \sim U(0, 1)$. If $u < \alpha$, we accept the point. Otherwise, we reject it. We repeat this step until we have run all n samples, or until we reach some pre-determined variance threshold and decide to stop.

This is the basic algorithm that is run in the provided code. **The actual rejection sampling algorithm is shown in lines 100-134 of the code.** (Much of the rest of the code is for visualization and plotting.)

4. Uses and drawbacks:. Rejection sampling can be used when we know the target distribution, $p(x)$, but cannot directly sample from it (*i.e. when we know the closed form of* $p(x)$). Drawbacks of this method include the fact that it may require many samples in order to produce an approximation with acceptable accuracy - this is the problem we are seeking to mitigate somewhat by choosing a proposal distribution that is fit well to the target distribution. This drawback is even more pronounced when the target distribution has more dimensions, however, due to what is known as the "curse of dimensionality", because the target grows even more sparse compared to the total proposal that must be sampled.

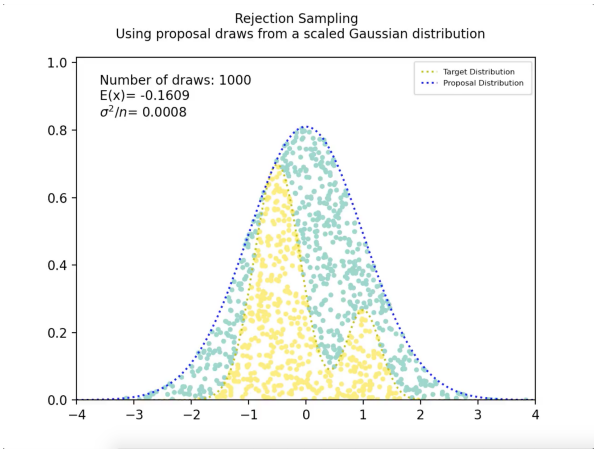

FIG. 2. *Screen shown at the end of the rejection sampling run*

5. The code and visualizations. In this demo, we use rejection sampling to evaluate some parameter of interest about our distribution, the mean of x (or expectation, E[X]). The default target distribution $p(x)$ that is included in this demo is given by:

$$p(x) = \frac{0.2}{0.3\sqrt{2\pi}} * e^{(-\frac{1}{2}*(\frac{x-1}{0.3})^2)} + \frac{0.7}{0.4\sqrt{2\pi}} * e^{(-\frac{1}{2}*(\frac{x+0.5}{0.4})^2)}$$

Which is a Gaussian mixture model which we are just using for illustration purposes, since E[X] can be calculated analytically: $E[X] = -0.15$. We will compare this value to the approximation of E[X] from the Monte Carlo draws:

$$\bar{E}[X] = \frac{1}{n}\sum_{i=1}^{n} x_i$$

In this demo, the user can change a number of different inputs, including the target distribution, the proposal distribution, and the number of runs to do. **The user defined parameters can be changed in lines 179-200.**

Once the user has specified their inputs, the code then runs and plots in real time as samples are chosen from $f(x)$ and accepted or rejected based on the criteria laid out in the "basic idea of the method" section. E[X] and a normalized variance measure, σ^2/n are also calculated in real time. Fig. 2 shows the end result of a demo run that stops at 1000 draws. E[X] and σ^2/n are displayed for comparison with other methods.

REFERENCES

[1] J. K. KRUSCHKE, *Doing Bayesian data analysis : a tutorial with R and BUGS*, Academic Press, Burlington, MA, 2011, http://www.amazon.com/Doing-Bayesian-Data-Analysis-Tutorial/dp/0123814855.

[2] A. B. OWEN, *Monte Carlo theory, methods and examples*, 2013.

[2] [1]