Monte Carlo Functional Approximation with Neural Networks

Joshua Shing Jun Le

[a]*Nanyang Technological University,*
School of Physical & Mathematical Sciences

Abstract

Monte Carlo (MC) simulation has a wide ranging applications in areas such as finance, project management, engineering and physics. It is known for its forecasting and computation capability in analytic. This independent study seeks to delve deeper into the rigorous mathematical properties MC simulation has on statistical computation, and how MC samples can be used to approximate functions using neural networks.

Keywords: Monte Carlo Simulation, Statistical Computation, Functional Approximation

1. Introduction

MC simulation has been largely used in the natural sciences and the operations sector for its capability of mapping possible event outcomes through random variable generation. Its uses are wide ranging and has brought about many practical uses in decision making processes. Above and beyond its random variable generative property, MC simulation can be used for statistical computation in approximating high computational costs functions such as the Bayesian inferences, MC integration and density approximation.

In today's era of high computation where neural networks (NN) can efficiently compute big data sets, a NN can be trained to approximate a function using MC samples as inputs. It does so by minimising the cost function between true and training data in its forward propagation steps.

This paper aims to provide a thorough explanation of how MC random variable generation works, its uses in statistical computation and how NN can be used to approximate a function generated by MC samples.

1.1. Reference Work

In this independent study, the primary work of reference is Robert et al. (1999). *Monte Carlo Statistical Methods 2nd edition* builds from Bayesian inferences in statistics and outlines the distribution types, random variable generation, MC integration, variance reduction and other random sample generation

algorithms in its chapters. It provides concise explanation of the uses of MC and gives sufficient exercises to mathematically prove its properties.

Another paper published by Nguwi et al. (2022) provides the intuition of MC functional approximation via NN. It mentions how deep learning implementation of MC sampling can provide numerical solutions to PDEs. In the interest of this study, we will use this methodology for statistical methods.

2. Monte Carlo Statistical Methods

2.1. Random Variable Generation

MC simulation is the algorithm of generating repeated random samples from a distribution. The *uniform pseudo-random number generator* forms the basis of sampling from the uniform distribution $U_i \sim U_{[0,1]}$. It generates an initial value u_0 and with a transformation D produces a sequence of samples $(u_1, ..., u_n)$ through $u_i = D(u_{i-1})$.

We can then use sampled u_i as probability measure for the *inverse transform* of F to generate a random sample $X \sim F$. For a non-decreasing function F on $\mathbb{R}$ defined by $F^-(u) = inf\{x : F(x) \geq u\}$, it is sufficient to generate $U \sim U_{[0,1]}$ and make the transformation $x = F^-(u)$.

2.1.1. Accept-Reject Method

Nonetheless not every distribution can be directly simulated by an inverse transform. The accept-reject method utilises a proposal distribution g to simulate draws from f. Suppose that

$$\int_a^b f(x)\,dx = 1$$

and f is bounded by a function m. We can simulate $Y \sim U_{[a,b]}$ and $U|Y = y \sim U_{[0,m]}$ and accept (Y, U) if the constraint $0 < u < f(y)$ is satisfied. Let x be accepted y value. The accepted samples drawn are of the correct distribution because

$$P(X \leq x) = P(Y \leq x | U < f(Y)) = \frac{\int_a^x \int_0^{f(y)} dudy}{\int_a^b \int_0^{f(y)} dudy} = \int_a^x f(y)dy$$

Replacing $m = Mg(x)$ where $M \geq 1$, we get the general *accept-reject method* with $P(accept) = \frac{1}{M}$.

Algorithm 1 Accept-Reject Method

$X \sim g$, $U \sim U_{[0,1]}$
if $U \leq f(X)/Mg(X)$ **do**
    Accept $Y = X$
else
    return **if**

2.2. Monte Carlo Integration

MC can be applied to statistical computation such as expectation computation given by

$$\mathbb{E}_f[h(X)] = \int_{\chi} h(x)f(x)dx$$

Using a sample $(X_1, ..., X_m)$ generated from f by *inverse transform* or *accept-reject method*, we can compute the expectation through

$$\mathbb{E}_f[h(X)] \simeq \frac{1}{m}\sum_{i=1}^{m} h(x_i) = \hat{h_m}$$

The MC estimator $\hat{h_m}$ converges to $\mathbb{E}_f[h(X)]$ by Law of Large Number for large sample size.

2.2.1. Importance Sampling

In the interest of flexibly sample from other arbitrary distribution g under the constraint $supp(f) \subset supp(g)$, we can use the identity

$$\mathbb{E}_f[h(X)] = \int_{\chi} h(x)\frac{f(x)}{g(x)}g(x)dx$$

whereby applying MC integration yields

$$\mathbb{E}_f[h(X)] \simeq \frac{1}{m}\sum_{i=1}^{m} \frac{f(x_i)}{g(x_i)}h(x_i) = \hat{h_m}$$

2.3. Controlling Monte Carlo Variance

The speed of MC estimator convergence $\hat{h_m}$ can be assessed through the variance

$$var(\hat{h_m}) = \frac{1}{m}\int_{\chi} (h(x) - \mathbb{E}_f[h(X)])^2 f(x)dx$$

There are several techniques to reduce MC variance below.

2.3.1. Finite Variance Estimator

For arbitrary g, there exists an optimal choice of g that minimises the MC estimator variance from *importance sampling*.

$$var\left[\frac{h(X)f(X)}{g(X)}\right] = \mathbb{E}_g\left[\frac{h^2(X)f^2(X)}{g^2(X)}\right] - \left(\mathbb{E}_g\left[\frac{h(X)f(X)}{g(X)}\right]\right)^2$$

By Jensen's inequality,

$$\mathbb{E}_g\left[\frac{h^2(X)f^2(X)}{g^2(X)}\right] \geq \left(\mathbb{E}_g\left[\frac{|h(X)|f(X)}{g(X)}\right]\right)^2 = \left(\int |h(x)|f(x)dx\right)^2$$

which yields the optimal choice of g

$$g^*(x) = \frac{|h(x)|f(x)}{\int_{\chi} |h(z)|f(z)dz}$$

2.3.2. Rao-Blackwellisation

If X can be simulated from a joint distribution $f(x, y)$ satisfying

$$\int f(x, y)dy = f(x)$$

where $\hat{h_m}(x) = \mathbb{E}_f[h(X)]$, then the estimator $\hat{h_m}(y) = \mathbb{E}_f[\hat{h_m}(x)|Y]$ can be used on the variance lower bound

$$var(\hat{h_m}(y)) \leq var(\hat{h_m}(x))$$

2.3.3. Riemann-Approximation

The MC integral can be computed using Riemann sums with varying degree of variances for different rules (generally derived from error bound of each rules). For $(X_0, ..., X_m)$ sampled from f with an ordered statistic $X_0 \leq ... \leq X_m$, considering the left end-point

$$\sum_{i=0}^{m-1} h(X_i)f(X_i)(X_{i+1} - X_i) \simeq \int_0^1 h(x)f(x)dx$$

and trapezoidal rule

$$\sum_{i=0}^{m-1} \frac{f(X_i)h(X_i) + f(X_{i+1})h(X_{i+1})}{2}(X_{i+1} - X_i) \simeq \int_0^1 h(x)f(x)dx$$

Therefore the $var(\theta_{trap}) \leq var(\theta_{left})$ will affect speed of convergence of MC.

2.4. Bayesian Inference

In Bayesian statistics, the computation of posterior distribution and normalising constants can be computationally expensive. MC methods are useful in approximating such functions. The Bayesian inference is given by

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\int f(x|\theta)\pi(\theta)d\theta} = \frac{f(x|\theta)\pi(\theta)}{c}$$

In approximating normalising constant for two models, $\pi_1(\theta) = \frac{\tilde{\pi_1}(\theta)}{c_1}$ and $\pi_2(\theta) = \frac{\tilde{\pi_2}(\theta)}{c_2}$, the ratio $\phi = \frac{c_1}{c_2}$ can be computed using the MC estimator

$$\phi \simeq \frac{1}{n}\sum_{i=1}^{n} \frac{\tilde{\pi_1}(\theta_i)}{\tilde{\pi_2}(\theta_i)}, \text{ where } \theta_1, ..., \theta_n \sim \pi_2$$

3. Neural Network: Monte Carlo Functional Approximation

3.1. n-Layers

Neural network can be used to approximate a function generated by MC samples by forward propagation pass through *n-layers*. Each pass uses gradient

descent to minimise the cost function; mean-sqaured error between true and sampled data sets. Given a variable of interests x^* to approximate where $X \sim F$ and H_i is MC sample, we aim to minimise the cost function

$$L = argmin \frac{1}{n} \sum_{i=1}^{n} (H_i - x^*)$$

The error rate generated from the forward propagation is then fed back through the NN in a process called backward propagation to adjust the weights and biases of the nodes. Having this process iterated *(epoch counts)* will yield better accurate weights and biases used for prediction.

In the interest of this study, we want to approximate the PDF of an arbitrary density f through adjusting weights w.

Algorithm 2 MC Functional Approximation by NN

$X \sim F$
Input: The learning rate α and epoch P
Output: $v(\cdot, \cdot; w) \in \mathbb{R}^{n \times 2} \leftarrow$ vector of predicted and true values
$(H_i)_{1 \leq i \leq n} \leftarrow$ monte carlo samples from distribution f
    for $j \leftarrow 1, ..., P$ **do**
        $L \leftarrow \frac{1}{n} \sum_{i=1}^{n} (H_i - x^*)$
        $w \leftarrow w - \alpha \nabla_w x^*$
    end for

A sample of code on Python 3.9 environment can be found in github for an approximation of mean and variance on a Gaussian distribution from MC samples.

3.2. Activation Function

The ideal activation function of choice is the Leaky Rectified-Linear Unit *(Leaky ReLU)* $\sigma_{Leaky}(x) = max(0.01x, x)$ on the layer function $L(x) = \sigma(wx + b)$. Other available activation functions are *ReLU*, *tanh* and *sigmoid* with respective formulae $\sigma_{ReLU}(x) = max(0, x)$, $\sigma_{tanh}(x) = tanh(x)$ and $\sigma_{sig}(x) = \frac{1}{1+e^{-x}}$. The *ReLU* functions are more efficient than *tanh* and *sigmoid* as only a certain number of nodes are activated at the same time which minimise computational costs Sharma et al. (2017). The accuracy of the activation function can be evaluated by the variance of the generated samples.

From the above experiment of MC functional approximation, we can compute the mean of a Gaussian distribution. For a sample size $n = 1500$ and epoch count $P = 1000$ whereby $X_i \sim N(4, 1)$, we want to see the simple computation of the MC integral by minimising following cost function

$$L = argmin \frac{1}{1500} \sum_{i=1}^{1500} (X_i - 4)$$

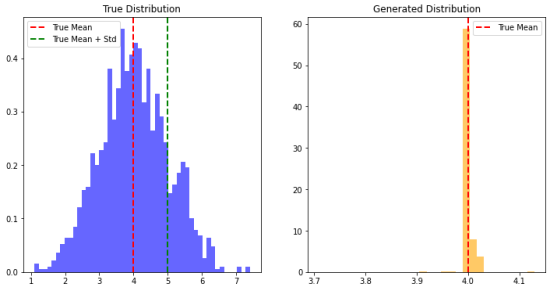

Figure 1: Leaky ReLU Activation Function for true expectation of 4

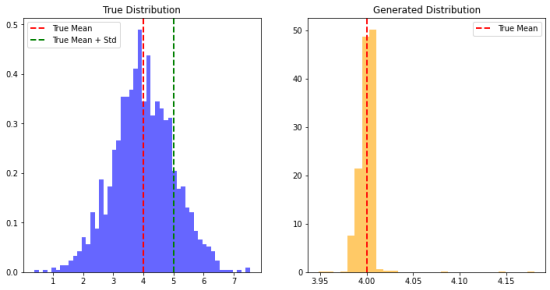

Figure 2: ReLU Activation Function for true expectation of 4

Experimentally, the variance of the *ReLU* and *Leaky ReLU* has minor differences. This can be explained by *ReLU* function deactivating nodes when inputs are negative; $x < 0$. This will retard the performance of the neural network Xu et al. (2020). Therefore when experimenting on NN, it is important to achieve both efficient computational and sufficient nodes activated.

4. Project Extension

4.1. Monte Carlo Marginalisation

MC marginalisation is a technique for calculating a marginal density. It can also be considered as multivariate-rank reduction technique from its simulation from a joint density $(X_i, Y_i) \sim f_{xy}(x, y)$ to give a resultant marginal distribution $f(x^*)$.

$$\mathbb{E}_f[h(x,y)] \simeq \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \frac{f_{xy}(x^*, y_i)\omega(x_i)}{f_{xy}(x_i, y_i)} = \int\int \frac{f_{xy}(x^*, y)\omega(x)}{f_{xy}(x, y)} f_{xy}(x, y)dxdy = f_x(x^*)$$

where $h(x, y) = \frac{f_{xy}(x^*,y)\omega(x)}{f_{xy}(x,y)}$. MC functional approximation can be used to compute complicated integrals induced by an arbitrary $\omega(x)$.

Proof.

$$\mathbb{E}_f[h(x,y)] \simeq \int\int \frac{f_{xy}(x^*,y)\omega(x)}{f_{xy}(x,y)} f_{xy}(x,y)dxdy = \int\int f_{xy}(x^*,y)\omega(x)dxdy$$

$$= \int c_1 g(y)dy = \int c_2 g(y)dy = \int f_{xy}(x^*,y)dy = f_x(x^*)$$

The computation of the integral (highlighted above) for some complicated function $\omega(x)$ can utilise the MC functional approximation methods. The errors of the cost function are minimised through backward propagation process with NN hyper-parameters tuning available. This is the further extension of this study.

Ideally the computation of the integral would be to divisively eliminate $\omega(x)$ depending on the choice of $f_{xy}(x,y)$. Refer to github for a sample code to simulate the marginal density $X^* \sim f_x$ from $X|Y = y \sim Gamma(y,1)$ and $Y \sim Exp(1)$ using the optimal choice for $\omega(x)$, $X \sim Exp(1)$. In this case f_x is the standard Gaussian distribution.

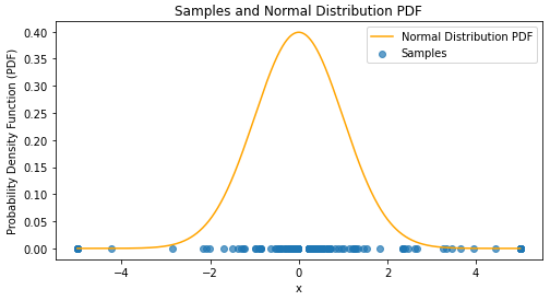

Figure 3: Samples generated from MC functional approximation for $N(0,1)$

5. Conclusion

In conclusion MC simulation can be used for statistical computation such as MC integration, Bayesian inference and distribution approximation. The MC samples can be used for functional approximation with error minimisation through neural networks. As MC methods cover wide variety of domain as a powerful statistical tool, this study should be further continued to broaden the scope of analytic.

References

Nguwi, J.Y., Penent, G., Privault, N., 2022. A deep branching solver for fully nonlinear partial differential equations. arXiv preprint arXiv:2203.03234 .

Robert, C.P., Casella, G., Casella, G., 1999. Monte Carlo statistical methods. volume 2. Springer.

Sharma, S., Sharma, S., Athaiya, A., 2017. Activation functions in neural networks. Towards Data Sci 6, 310–316.

Xu, J., Li, Z., Du, B., Zhang, M., Liu, J., 2020. Reluplex made more practical: Leaky relu, in: 2020 IEEE Symposium on Computers and communications (ISCC), IEEE. pp. 1–7.