Dissertation notes

Haoyang Zhang

June 4, 2023

Abstract

This is my note on the dissertation work. My results will be continuously updated on this note.

1 Single asset Black-Scholes with Neural Networks

1.1 The problem description

Assume the underlying asset follows the geometric Brownian motion (GBM) with constant drift rate and volatility:

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{1}$$

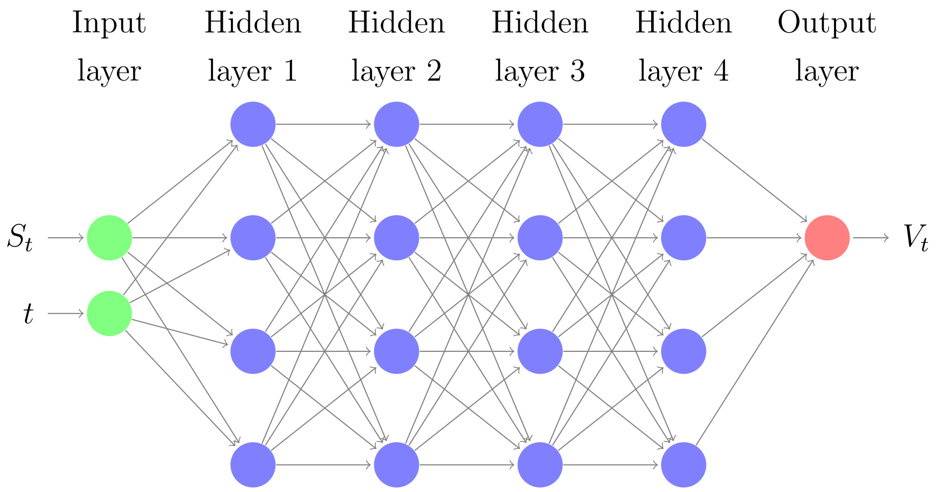
Denote $V(t, S_t)$ as the price of the derivative. From Ito Lemma, V_t obeys the following stochastic process:

$$dV_t = \left(\mu S \frac{\partial V}{\partial S} + \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right) dt + \sigma S \frac{\partial V}{\partial S} dW_t \tag{2}$$

with terminal condition $V(S_T, T) = g(S_T)$ where $g(x)$ is the payoff function.

1.2 The neural network structure

The neural network takes the input of the current underlying price S_t and time t and gives an output of the current price of the derivative $\hat{V}(S_t, t)$.



1.3 The loss function and training process

The Euler-Maruyama scheme of the underlying and derivative price is as follows:

$$S^{n+1} \approx S^n + \mu \Delta t^n + \sigma \Delta W^n \tag{3}$$

$$\begin{aligned} V^{n+1} \approx V^n &+ \left(\mu S^n \frac{\partial V(S^n, t^n)}{\partial S} + \frac{\partial V(S^n, t^n)}{\partial t} + \frac{1}{2}\sigma^2 (S^n)^2 \frac{\partial^2 V(S^n, t^n)}{\partial S^2} \right) \Delta t^n \\ &+ \sigma S^n \frac{\partial V(S^n, t^n)}{\partial S} \Delta W^n \end{aligned} \tag{4}$$

for $n = 0, 1, \ldots, N-1$, where $\Delta t^n := t^{n+1} - t^n = T/N$ and $\Delta W^n \sim \mathcal{N}(0, \Delta t^n)$ is a random variable with mean 0 and standard deviation $\sqrt{\Delta t^n}$. The price of the derivative $V(S_t, t)$ is approximated by the neural network $\hat{V}(S_t, t)$. The partial derivative the of the price $\frac{\partial \hat{V}}{\partial S}, \frac{\partial \hat{V}}{\partial t}, \frac{\partial^2 \hat{V}}{\partial S^2}$ can be evaluated through the Auto-grad through the neural network (more details will be added to later):

$$V^{n+1} \approx V^n + \left(\mu S^n \frac{\partial \hat{V}}{\partial S} + \frac{\partial \hat{V}}{\partial t} + \frac{1}{2}\sigma^2 (S^n)^2 \frac{\partial^2 \hat{V}}{\partial S^2} \right) \Delta t^n + \sigma S^n \frac{\partial \hat{V}}{\partial S} \Delta W^n \tag{5}$$

We want to train our model $\hat{V}(S_t, t)$ such that it can give the right value of $V(S_t, t)$. We construct the loss function as follows:
Along one Monte-Carlo path of S_t, at time step t^n we have the underlying price S^n

1. The current derivative price is approximated by the neural-network $\hat{V}(S^n, t^n)$
2. We generate the normal random variable ΔW^n and the underlying price of the next time step S^{n+1} is evaluated through using equation 3.
3. The derivative price of the next time step V^{n+1} is evaluated through equation 5
4. We calculate the neural network estimate of the derivative price at time step t^{n+1}, which we denote as $\hat{V}(S^{n+1}, t^{n+1})$
5. The loss function is added by $|\hat{V}(S^{n+1}, t^{n+1}) - V^{n+1}|^2$

Also, at the final time step $t^N = T$ we force the terminal condition, and the loss function is added by $|\hat{V}(S^N, T) - g(S^N)|^2$. The loss function is then given by

$$\sum_{m=1}^{M} \sum_{n=0}^{N-1} \left| \hat{V}_m^{n+1} - V_m^{n+1} \right|^2 + \sum_{m=1}^{M} \left| V_m^N - g\left(S_m^N\right) \right|^2 \tag{6}$$

which corresponds to M different realizations of the underlying Brownian motion. The subscript m corresponds to the m-th realisation of the underlying Brownian motion, while the superscript n corresponds to time t^n.

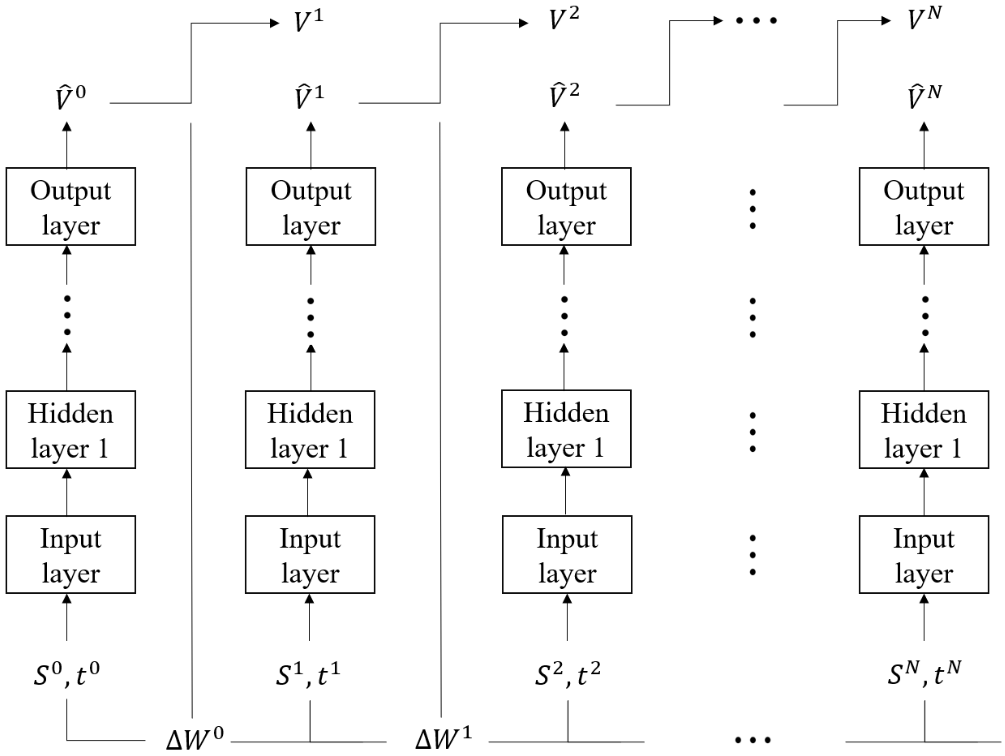


Figure 1: The training process

1.4 Numerical Results

We tested our model on a non-dividend vanilla call option. The payoff function was $g(x) = max(x - K, 0)$ where K was the strike price and $\mu = r$, where r was the risk-free rate. The exact price of the call option $V(S_t, t)$ was given by the Black-Scholes-Merton formula:

$$\begin{aligned} V\left(S_t, t\right) &= N\left(d_+\right) S_t - N\left(d_-\right) K e^{-r(T-t)} \\ d_+ &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\ d_- &= d_+ - \sigma\sqrt{T-t} \end{aligned} \tag{7}$$

We trained our model $V(S_t, t)$ using the described training process. The true values of the parameters were set to $r = 0.05$, $\sigma = 0.4$, and $T = 1$. We chose a batch size of $M = 5$ and the number of intervals in time was $N = 10$. A fully connected forward neural network with 4 hidden layers , each containing 256 neurons, was used and we chose *tanh* as our activation function. Adam [1] was used as our optimisation algorithm with a learning rate of 0.003. The model was trained for 3000 epochs.

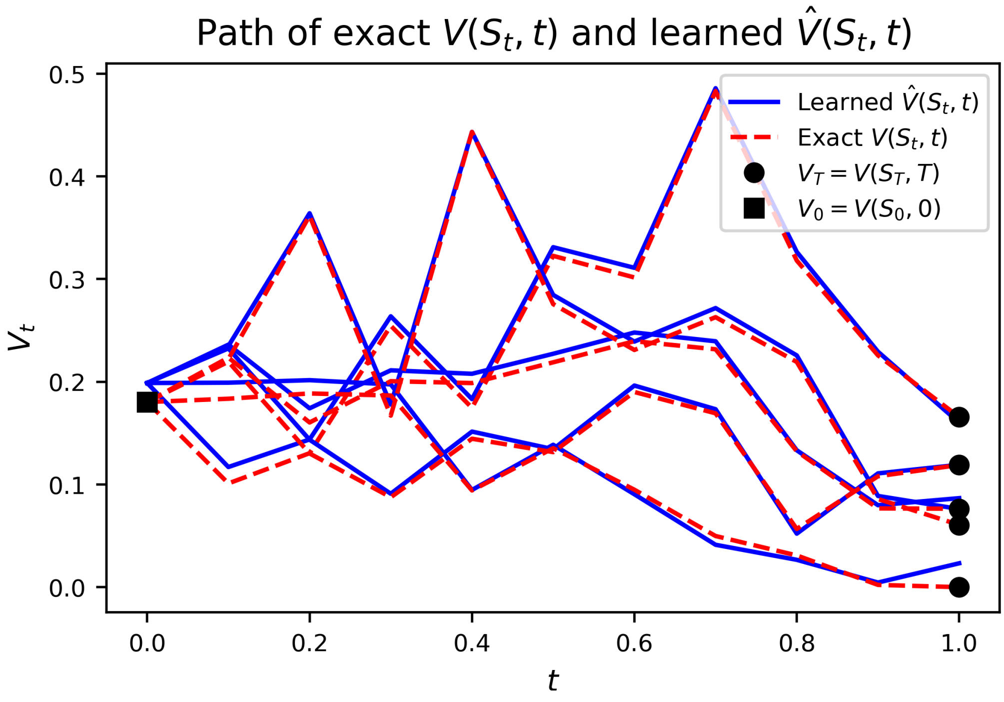


Figure 2: Plot of learned path and the exact path

Figure 2 illustrates the results after training. In the figure, 5 sample paths of the underlying processes S_t were generated with the same S_0, corresponding to 5 processes of the price of the derivative $V(S_t, t)$. The exact path of $V(S_t, t)$ given the path of S_t was determined by the Black-Scholes formula described above. Our neural network output $\hat{V}(S_t, t)$ gives the learned path, which is approximately the same as the exact path, showing that our algorithm is a good approximation of $\hat{V}(S_t, t)$. By learning through more and more Monte Carlo paths, our model $\hat{V}$ would give a good approximation to the derivative price over the whole region. Figure 3 gives the comparison of the derivative price surface between the Black-Scholes model V_{BS} and our trained model $\hat{V}$. As we can see, under our training process with Monte Carlo paths, our output price surface is close to the Black-Scholes prediction, with an absolute error of 0.02 over the region of $t \in [0, 1]$ and $S_t \in [0, 2]$

1.5 Question list for the next meeting

1. Scheme for SDE Simulations: currently using Euler forward: $S^{n+1} = S^n + \mu S^n \Delta t + \sigma S^n \Delta W^n$. Would it be different if we use $S^{n+1} = S^n e^{\mu \Delta t + \sigma \Delta W}$, or another scheme?

2. Greeks approximation This will be tested later.

3. About pathwise MC for Greeks, how to evaluate Rho and Theta?

$$\frac{\partial V}{\partial \theta} = \int \frac{\partial f}{\partial S} \frac{\partial S(T)}{\partial \theta} p_W \, \mathrm{d}W = \mathbb{E}\left[\frac{\partial f}{\partial S} \frac{\partial S(T)}{\partial \theta}\right] \tag{8}$$

$$S(T) = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) T + \sigma W_T\right)$$

$$\frac{\partial S(T)}{\partial t} = -(\mu - \frac{\sigma^2}{2}) S_T?$$

$$t = T - \tau$$

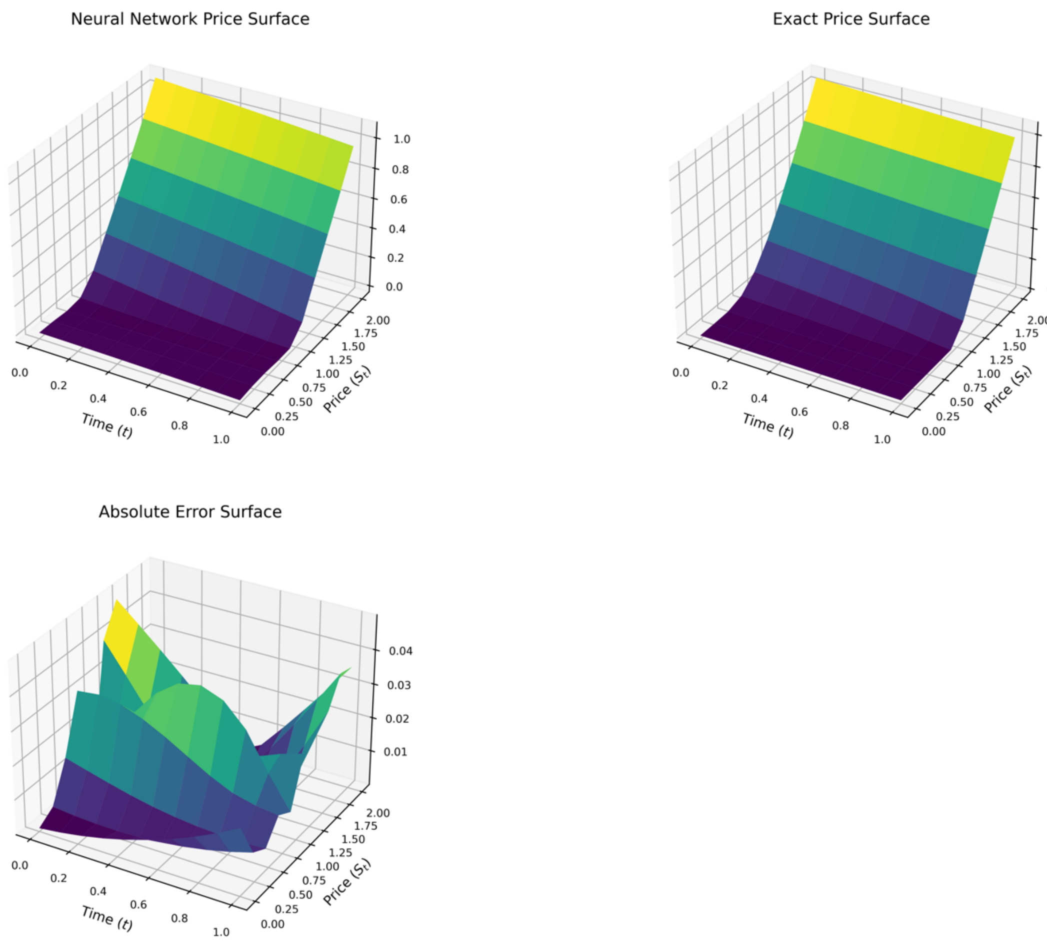


Figure 3: Plot of learned price surface and error

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.