Case 2: Black Scholes Case Study

RSM338

1006581353 - Dawood Khokhar
1006776375 - Marjorie He
1006722101 - Jeffrey Chen
1006769613 - Carson Feng
1005990849 - Sherry You

The Black-Scholes (BS) model is a formula used to determine the value of an European call option. It takes five inputs:

- The stock price
- The strike price
- The risk-free rate of interest
- The dividend yield
- The stock price volatility
- The maturity of the option

The following report discusses the results after building various neural network models to approximate the BS model. The features of each model are the five inputs of the BS model. Our target is the price of a call option. To understand the effect of changing a single parameter, we will hold all other parameters constant (the default values will be Single Neural Network (SNN) with a reLU activation function, adam optimizer, and 100 epochs). Using the base model and our code comparing all 81 possible combinations of parameters, we will determine the best set of parameters for the model that best approximates BS in terms of mean squared error (MSE) and convergence property.

In determining the optimal neural network structure, we observed the changes between using a Single Neural Network, a Deep Neural Network with three layers of 100 neurons, and a Deep Neural Network with six layers of 50 neurons. To compare the effect of changing the neural structure, all three models used reLU activation functions with an adam optimizer and 100 epochs. These changes can be observed by comparing Figure 1, Figure 2, and Figure 3 from the appendix. There is a substantial improvement from Figure 1 to Figure 2 as the MSE decreases and the line smoothes, indicating less variation. From Figure 2 to Figure 3, there is a very slight difference. The Deep Neural Network model with six layers seems slightly worse in the first 40 iterations; however, it improves with a lower MSE afterwards. In theory, adding more layers benefits the model as it allows the model to increase the parameters involved and allows the model to fit complex functions and data.

Next, 3 different activation functions were considered in determining the model function. Firstly, the reLU function was considered. In practice, the reLU function is most commonly used since it is easy to implement and less susceptible to the vanishing gradients issue (when there is little to no change at extreme values) seen by the sigmoid and tanh functions. However, it suffers from the dying units problem, where inputs approaching negative or 0 cannot learn. Tanh and sigmoid activation functions were also considered, although they suffer from the vanishing gradients issue. Through this understanding, we hypothesize that reLU may be the best.

We rotationally evaluated the 3 activation functions with the base function (i.e. 100 epoch, single neural network, and adam) to test this hypothesis. From there, it can be seen that the reLU function is the best both in terms of MSE and convergence. With 100 iterations, the reLU activation function sees an MSE of just under10^{-4} and remains fairly stable, with very low fluctuation per iteration (Figure 4). The Tanh function sees low convergence and a higher MSE

with just under 10^{-3} in 100 iterations and a lot of instability (Figure 5). Finally, the sigmoid function sees a high convergence but a high MSE of just under 10^{-3} (Figure 6). Thus, we extrapolated that the reLU function would be the ideal activation function in terms of both MSE and convergence. The reLU function proved to be the best activation function when testing with all possible features, confirming our hypothesis from the SNN testing of activation functions.

Choosing the right optimizer along with the other parameters can help squeeze out an accurate neural network model to approximate the nonlinear function of BS for financial option derivatives. The purpose of these hyperparameters is to allow the process to converge (shift towards the optimum of the cost function) more quickly while minimizing mse. There are two families of hyperparameters: gradient descent optimizers and adaptive optimizers. Gradient descent algorithms force you to manually tune the learning rate (ex. SDG). On the other hand, the learning rate is automatically adapted in adaptive algorithms (Ex. Adam). Usually, larger learning rates result in rapid changes and require fewer training epochs and vice versa. When the learning rate is too large, the model will converge too quickly to a suboptimal solution, whereas if the learning rate is too small it can cause the process to get stuck. So, if input data is sparse, adaptive learning rate methods usually produce the best result.

When we first tested momentum with the base function, it led to a suboptimal result. Momentum is a gradient descent algorithm that helps accelerate SGD in the relevant directions. It dampens oscillations around points where the surface curves much more steeply in one dimension than in others. However, as the algorithm progresses, it might accelerate too fast and miss the local minima. This is why the mse of 0.009784 is higher than other optimizers.

Root mean squared propagation (RMSprop) is an unpublished adaptive learning method stemming from the need to resolve Adagrad's radically diminishing learning rates. It does not provide the sum of the gradients, but instead an exponentially decaying average. This decaying average is realized through combining the Momentum algorithm and Adagrad algorithm, with a new term. It performed with an average MSE of 0.000279 (worse than Adam but better than momentum).

Adaptive Moment Estimation (Adam) leads to the best results as it combines the best properties of AdaGrad and RMSProp. It usually works best with problems that have large amounts of data or parameters, even if it's noisy, which is true with call option data. Instead of adapting the parameter learning rates based on the mean like RMSProp, Adam also uses the uncentered variance. More specifically, the algorithm calculates the exponential decaying average of past squared gradients like RMSprop, while keeping an exceptionally decaying average of past gradients similar to momentum. This type of memory allows Adam to correct biases and outperform RMSprop towards the end of optimization as gradients become sparser. When Adam was used the lowest mse of 0.000099 was achieved with the base function. Adam should be the best optimizer, which is reflected in the code comparing all possible combinations.

An epoch indicates the number of times the training dataset passes backwards and forwards through the machine learning algorithm. As the number of epochs increases, the

weights are changed in the neural network and the curve goes from underfitting to optimal to overfitting. In the process of choosing the best Epoch number for our model function, we considered 3 different epoch counts: 100, 500, 1000. When epoch equals 100 for all functions, the models demonstrate the most stability between each iteration. While we were able to get a lower validation loss MSE on average as we increased the epoch number for our base model, the fluctuation increased between each iteration compared to models with lower epochs.

When we were deciding which epoch from 100, 500, 1000 to use for our model function, we considered two important factors: Should we use the epoch count that gives the best fitting model or the epoch count that gives a lower MSE on average. Another factor that we considered is run time. As the epoch number gets larger, each neural network takes a longer time to finish running the model due to the larger number of iterations. Especially considering the application of valuing options, the model needed to be quick in order to match market movements. Although 1000 epochs gave the lowest MSE, the model fluctuated too much and took too long to run; at its rate, the market would have changed! Therefore, we chose 100 epochs for our model functions because not only does it take the shortest time to run, but it also gives the best fitting model with the least fluctuation between for each iteration.

To further find the best model for BS, we decided to create a model that compares the MSE for every combination of parameters (81 different models). After comparing the MSE for all 81 models, we found that the model with the lowest MSE of 3.1832e-07 had the following parameters: (‘DNN 6 layers’, ‘relu’, ‘adam’, 1000). This confirms that these are the best set of parameters to use when trying to create a model to approximate BS. However, as mentioned before, we wanted to balance stability, accuracy and speed, therefore after taking these factors into account, we concluded that the best model for BS should be the same model but with only 100 epochs: (‘DNN 6 layers’, ‘relu’, ‘adam’, 100).

Finally, after determining the best model for BS, we decided to try running the model again but with 60% of the data split for training, 20% for validation, and 20% for testing. From the results we can see that the MSE slightly increased to 5.1141e-06. This indicates that once we decreased the percentage of the data used in the training set, there is a larger difference between the estimated values from our model and the actual values from the data set. From this, we can conclude that the model was slightly overfitted as it does not work as well when tested against a new set of data. However, since the increase in MSE between the two models is very small, we can confirm that the SNN with a reLU activation function, adam optimizer, and 100 epochs is a good model to approximate the Black-Scholes Model.

Through our single variable parameter models, we were able to determine that the best set of parameters to model BS was a six layer DNN with reLU activation functions, an adam optimizer, and 100 epochs. In our neural structure analysis, we determined that the deep neural structure with 6 layers was the best out of the three structures as deeper structures are able to learn more complex and nonlinear functions. The reLU function had the lowest MSE out of all the activation functions, and it was also the easiest to implement while being the least susceptible to the vanishing gradients issue. The Adam optimizer was able to combine the best

properties of AdaGrad and RMSProp and achieved the lowest MSE out of the different optimizers. Considering the fast pace of market movements, 100 epochs was chosen to minimize the runtime while producing a model with the least fluctuations between each iteration. Aftering testing the 81 combinations of model parameters, these parameters were further confirmed to be the parameters that produce the best model to approximate Black Scholes.

Appendix

Figure 1: SNN Training Evolution

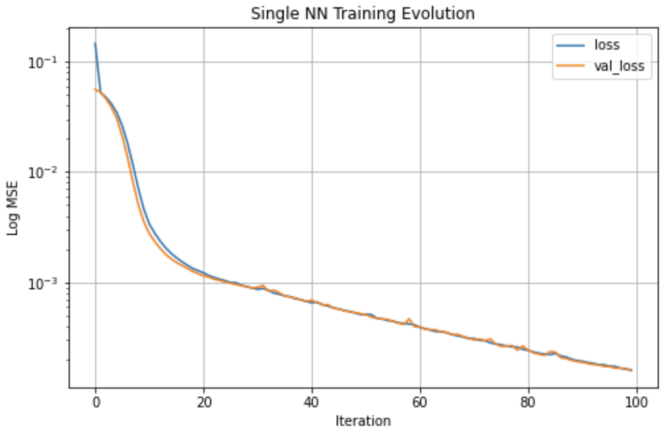

Figure 2: DNN Training Evolution

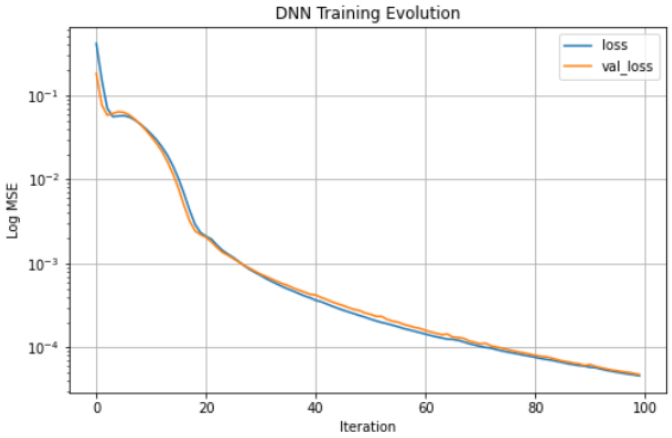

Figure 3: Six Layer DNN Training Evolution

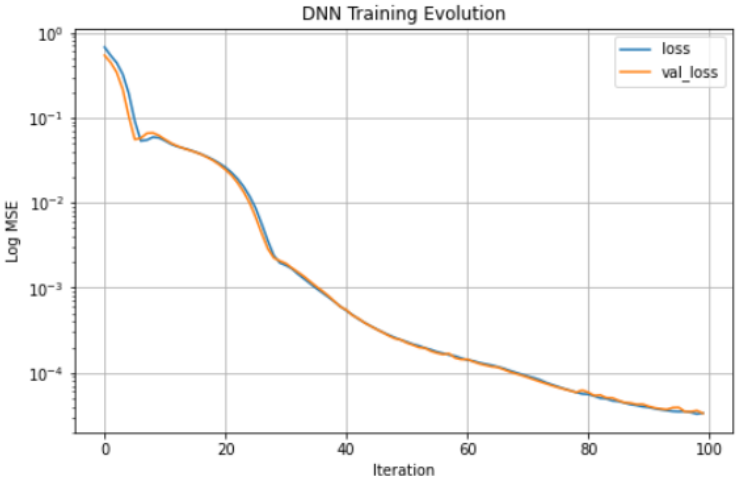

Figure 4: Relu SNN Training Evolution

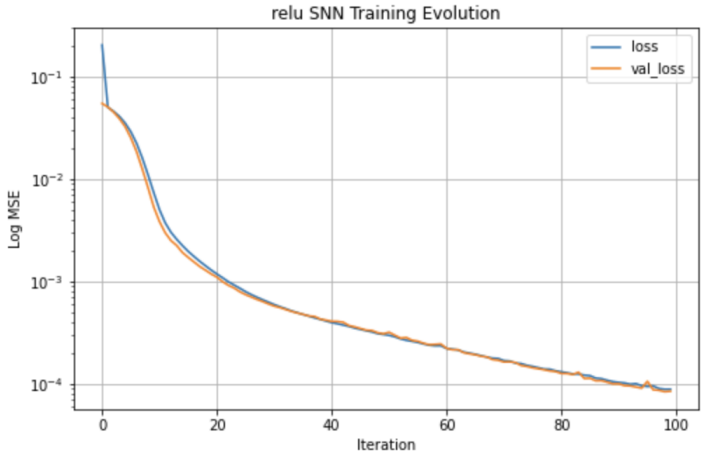

Figure 5: Tanh SNN Training Evolution

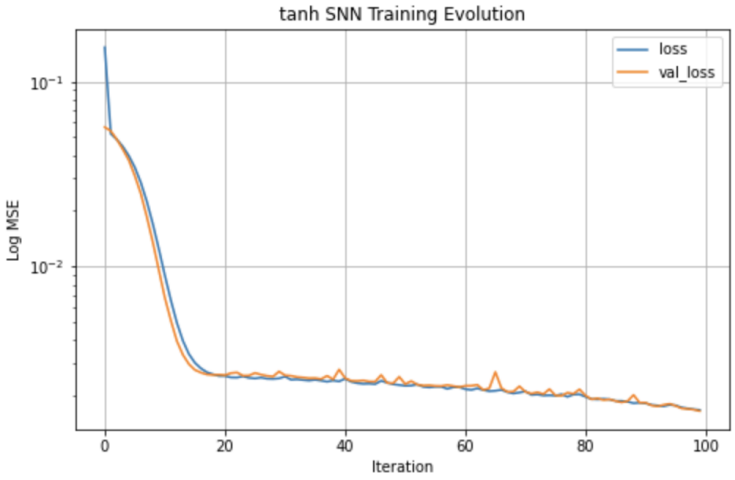

Figure 6: Sigmoid Training Evolution

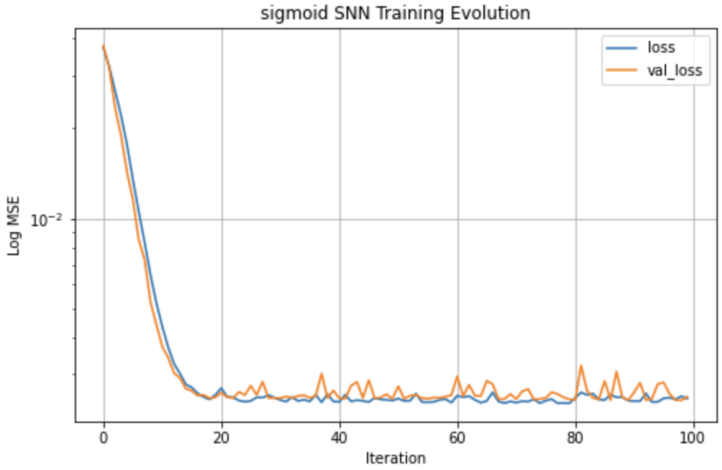

Figure 7: 100 Epochs Training Evolution

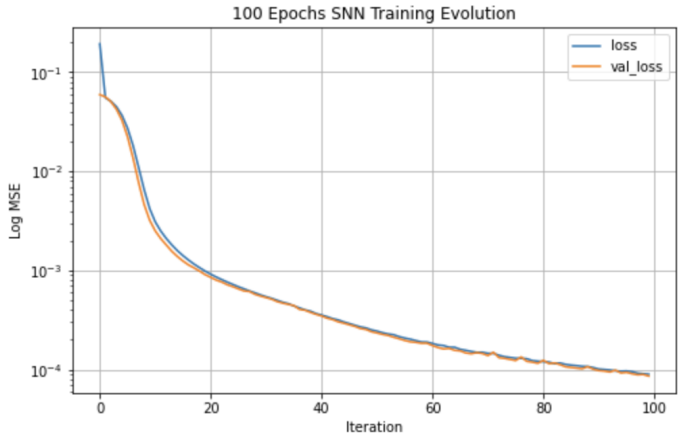

Figure 8: 500 Epochs Training Evolution

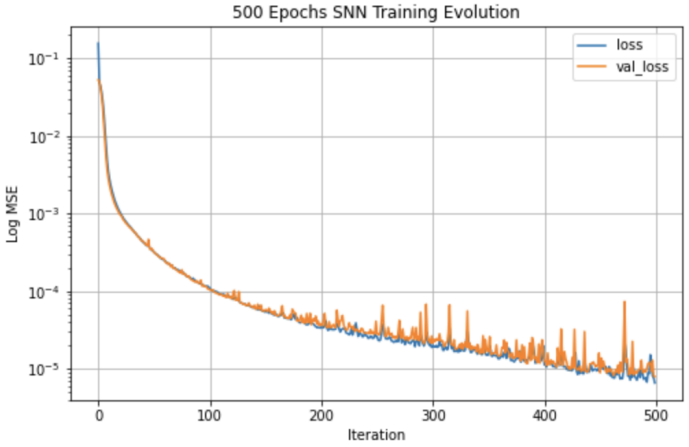

Figure 9: 1000 Epochs Training Evolution

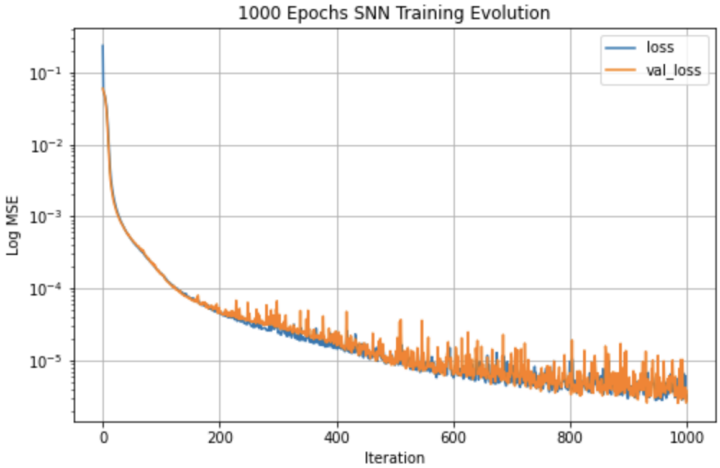

Figure : MSEs of Different Activation Functions

	relu	tanh	sigmoid
MSE	0.000148	0.001889	0.002607

Figure : MSEs of Different Optimizers

	rmsprop	sgd	adam
MSE	0.000279	0.009784	0.000099

Figure : MSEs of Different Epochs

	100	500	1000
MSE	0.000083	0.000005	0.000003