Neural Networks as an Option Pricing Method

Brendan Dowling, Senior Sophister

Since the bruising losses of the financial crisis over a decade ago, investors have sought out novel and complex ways to "beat the market", aiming to maximize returns and mitigate risk. Many have turned to computer-driven "systematic" investment strategies, which are freed from the shackles of human bias and slow reaction. In this paper, Brendan Dowling suggests the vast potential of neural networks to provide accurate derivative pricing. Derivative pricing has long been viewed as one of the most challenging tasks within financial mathematics, but this paper finds that artificial neural networks provide an excellent approach to this challenge. Traditionally, variations of the Black and Scholes model have been used, but these models rely on the user's ability to accurately model the stochastic process of the stock's price and are thus systematically flawed when faulty assumptions are made or the process is misspecified. Dowling finds that the non-linear relationship between an option's strike, time-to-expiry, and the underlying spot and price of a call option implied by the Black-Scholes model are all well captured by the artificial neural network and that human error is avoided. The paper then makes the case for the role of machine learning more broadly in derivative pricing within finance. Dowling's superb use of financial economic theory, mathematics, statistical modelling and knowledge of machine learning has earned "Neural Networks as an Option Pricing Method" the title of "Best Applied Essay" of the Student Economic Review XXXIV.

I.Introduction

The accurate pricing of derivatives is one of the most complex and challenging tasks in financial mathematics. In 1973, Black and Scholes published their breakthrough formula for pricing European options under a certain set of assumptions (Black, 1973). Since then, several variations of the Black-Scholes model have been devised and employed by investment banks and proprietary trading firms. These variations can potentially come in the form of incorporating jumps in stock prices (e.g. due to earnings releases) or modifying the assumption on the distribution of instantaneous stock returns (classically assumed to be log-normally distributed) (Wyse, 2019).

In all cases, however, the pricing formula relies on the parametric form of the underlying asset's price dynamics. Should one mis-specify or make faulty assumptions regarding the stochastic process for the underlying's price, one will yield a model which systematically mispric-es the contracts of interest. Such models are called parametric.

As more complex derivatives emerge and with the tremendous progress in computational power over the past few decades, there has been a marked increase in interest in using machine learning – data-driv-en approaches – for pricing such securities. Such models are called "non-parametric models" since they do not require any assumptions on the parametric form of the underlying's price dynamics.

These models have some key advantages over their parametric counterparts. First and foremost, they don't need to make any restric-tive parametric assumptions and don't suffer from mis-specification er-ror – this can, in theory, make them more accurate in markets for highly complex securities with non-closed form pricing formulas or non-stan-dard underlying dynamics. Second, they are more flexible, being able to change how they price a contract in the presence of market structure changes. Lastly, they're relatively trivial to implement and can be readily applied to highly differing securities.

The major drawback to such nonparametric methods is that they are extremely data-intensive – they require many months, if not years, of historic data in order to be well-trained. They are also computationally intensive and often require a significant time investment for hyperpa-rameter tuning. Furthermore, naive models which are trained off historic

prices will at best replicate the market's pricing model, which may itself be inaccurate. As such, significant work into determining the "correct" prices may need to undertaken before training. In addition, if the underlying asset's price dynamics are well understood and a closed form expression for the contract's price exists, then a parametric method will always outperform any nonparametric method. Nevertheless, there are certainly situations where such models can be more efficient or beneficial. In this paper, we focus on artificial neural networks – specifically multi-layer perceptrons – but there are several other non-parametric/data-driven modelling techniques which could similarly be employed.

II. Background

Pricing European Call Options

One of the most important findings in financial mathematics is the Black-Scholes equation. This partial differential equation describes the evolution of an option's price, *V,* over time (Black, 1973):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

Applying the corresponding terminal and boundary conditions, we can yield the value of a European call on a non-dividend-paying stock:

$$C(S_t, t) = S_t\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2) = S_t\Phi(d_1) - PV(K)\Phi(d_2)$$
$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$$
$$d_2 = d_1 - \sigma\sqrt{T-t}$$

where Φ is the CDF of a standard normal, S_t is the spot price of the underlying asset, K is the strike price, *r* is the risk-free rate, σ is the volatility of returns of the underlying asset, and T-t is the time to maturity (in years).

The objective of our neural networks will be to recover this

pricing formula from *just* the data. If they are successful, then this will then suggest that neural networks can be used for closely approximating similar pricing formulae which mightn't have an analytic solution but are similarly of high non-linearity (Hutchinson et al., 1994: 3).

Artificial Neural Networks

Artificial neural networks (ANN) are prediction models based on the trends that have occurred in the past. They are inspired by biological neural networks like those found in our brains. They consist of a collection of connected "artificial neurons". These take inputs and produce an output, depending on the node's activation function. The connections/edges allow neurons to transmit signals (their outputs) to other neurons in other "layers" of the network. Typically, the network is composed of the input layer, several "hidden" layers, and the output layer. The nodes and edges have weights (and each layer has an added bias constant) which are adjusted during the learning process to minimize some loss function.

Multilayer perceptrons (MLP) are fully connected ANNs. That is, each neuron is connected to every neuron in the previous and subsequent layer. There are no "loops" in the network, meaning that the value of a neuron does not feed backwards into a neuron of a layer preceding it.

When training MLPs, the back-propagation algorithm is typically used. This uses stochastic gradient descent to recursively optimize the weights and biases on each neuron over random subsets of the data for each iteration. The objective of this algorithm is to find the values for the weights and biases which minimize some specified objective function. For regression, the standard is to minimize the root-mean-square-error (RMSE).

Typically, when training a model, we normalize the input variables to be between 0 and 1 or -1 and 1 (depending on the activation function). This prevents certain input variables from "dominating" the network initially and has been shown to drastically reduce the amount of time needed to adequately train the network.

One important property of neural networks with non-linear activation function is the universal approximation result. Cybenko (1988) and Hornik (1989) demonstrate that an MLP can represent to arbitrary precision almost any linear and nonlinear function with bounded inputs

and outputs (Hutchinson, et al., 1994: 3). As such, it is generally sufficient to only have one hidden layer in the network. When increasing the number of neurons in the hidden layer, there is a (non-linear) tradeoff then faced between model accuracy and computational cost (and risk of over-fitting).

III. Data Set

Simulated Stock Prices

For our first model, we simulate two stock prices independently according to the Black-Scholes assumption of geometric Brownian motion:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t)$$

We take the number of trading days per year to be 252 and then draw 504 pseudorandom variates Z_t where

$$Z_1^{(j)}, \dots, Z_{504}^{(j)} \sim N\left(\frac{\mu}{252}, \frac{\sigma^2}{252}\right)$$

Then, via Ito's lemma, we have

$$S_j(t) = S(0)\, exp\left(\sum_{i=1}^{t} Z_i^{(j)}\right),\ t \geq 0$$

For both simulated stocks, we use a drift parameter of 10% per annum, a volatility parameter of 50% annualized, and a starting price of 2500. The result is two simulated 2-year price sequences, S_1 and S_2.

Historic S&P500 Index Prices

For our second model, we will use the historic price levels of the S&P500 market index. We take two years of historic data between January 1, 2017 and January 1, 2019. The resulting data frame has 502 observations. We find that the average annualized return volatility was 13%.

Synthesizing Option Chains

Unfortunately, historic option data is proprietary and expensive. So, we generate our own option chains around the price levels of both the simulated stocks and the real index.

We employ a rolling-window technique whereby for each observation in the three data sets, we create options at strikes between the current price minus 150 and the current price plus 150, in intervals of 10. For each strike, we then generate options with time to expiries between 1 and 50. We then use the Black-Scholes pricing formula to price each option. For the two simulated stocks, we just use their volatility parameter of 0.5 for σ. For the index, we use the average annualized return volatility of 0.13 for σ. For both, we take the risk-free rate, *r*, to be constant at 1%. This greatly simplifies the learning process since we don't need to model this as an extra variable. If the models can capture the relationship between the Black-Scholes price and the option's strike, time-to-expiry, and the current spot, then learning networks can readily capture the effects of the risk-free rate on the option's price [1].

For the first model, options on S_1 will be used for training and a random sample of options on S_2 will be used for testing. For the second model, we use an 80-20 training-test split of the synthesized options on the index. This is because we don't have another independent price series for which we can say the price dynamics are exactly equivalent.

There are several downsides to our methodology. First, while it is true that expected future volatility is constant for the simulated stocks, it is false for the index. Indeed, the stock market can enter periods of extreme turbulence during which expected future volatility until the option's maturity will vary wildly. Furthermore, expected future volatility is not constant as the time horizon increases. That is, options from the same day with the same strike but with differing time to expiries will generally not have the same implied volatilities. Summarizing these issues, in the real world the "correct" volatility parameter for the Black-Scholes price will be neither time-homogeneous nor independent of time to expiry.

In addition, though less important, this approach means that our neural networks will be trying to emulate the Black-Scholes model. This equivalently relates to our models not being able to capture the implied

1 See Hutchinson, Lo and Poggio (1994) for more details or note that a call's rho is a function of all variables.

volatility "smile" we often see from market prices. However, as mentioned earlier, if the models are successful, we can comfortably conclude that they will generalize to being able to price derivatives which do not quite follow the Black-Scholes formula but are likewise non-linear in parameters. However, it would have been far more desirable to have access to many months – or even years – of historical option data so we could both analyze the deviations from the Black-Scholes model implied by the market, and to confirm the notion that these models can be more general.

IV. Methodology

Independent Variables

The key input variables for the models will be the option's strike, spot price and time to expiry. Since, in all three data sets, we generate the option prices with constant volatility and risk-free rate parameters, we don't use these as independent variables. We again reiterate that including the historic risk-free rate in pricing the contracts at each stage of our rolling-window procedure would be largely superfluous in testing how well these machine learning techniques can reconstruct the Black-Scholes formula. However, if we had lots of actual historic *option* prices, we most certainly would have included both variables (using implied volatility as our volatility parameter [2]) as they are critical in accurately pricing the options.

The Neural Network

We use a standard network type – the multilayer perceptron (MLP) with the 3 input nodes, a hidden layer comprising of twelve[3] artificial neurons, and then a single output neuron. Adding additional hidden layers is likely superfluous, again noting the universal approximation result. Since this paper is concerned with the viability of these models as a pricing method, rather than building a functional predictive tool, we did not extensively tune the hyperparameters. The two models were trained using the Tensorflow2 library in R with 10 epochs.

2A discussion of how to get an accurate measure for σ independently is beyond the scope of this paper.

3 Chosen arbitrarily.

Our MLP has the following functional form:

$$\Upsilon(\vec{z}) = \sigma\left(\sum_{j=1}^{12}\left[w_2(j)\cdot\sigma\left(\sum_{i=1}^{3}[w_1(i,j)\cdot\vec{z}(i)]+b_1(j)\right)\right]+b_2\right)$$

where

- σ : Logistic transfer function, $\frac{1}{1+e^{-x}}$.
- $w_1(i,j)$: Weight between unit i of input layer and unit j of hidden layer.
- $w_2(j)$: Weight between unit j of hidden layer and the output neuron.
- $\vec{z}(i)$: Parameter i of input vector $\vec{z}$.
- $b_1(j)$: Bias applied to unit j of hidden layer.
- b_2 : Bias applied to output neuron.

We note here that the data (independent and dependent variables) was min-max normalized according to the following formula:

$$X_{ij}^{*} = \frac{X_{ij} - min(X_{.j})}{max(X_{.j}) - min(X_{.j})}$$

This was done so as to bound the inputs (and output) to be between 0 and 1 (since we are using logistic transfer function) to help with training. We convert the model predicted values into "real" prediction values by inverting the above process. It is these inverted values that we use in assessing model performance.

V. Results

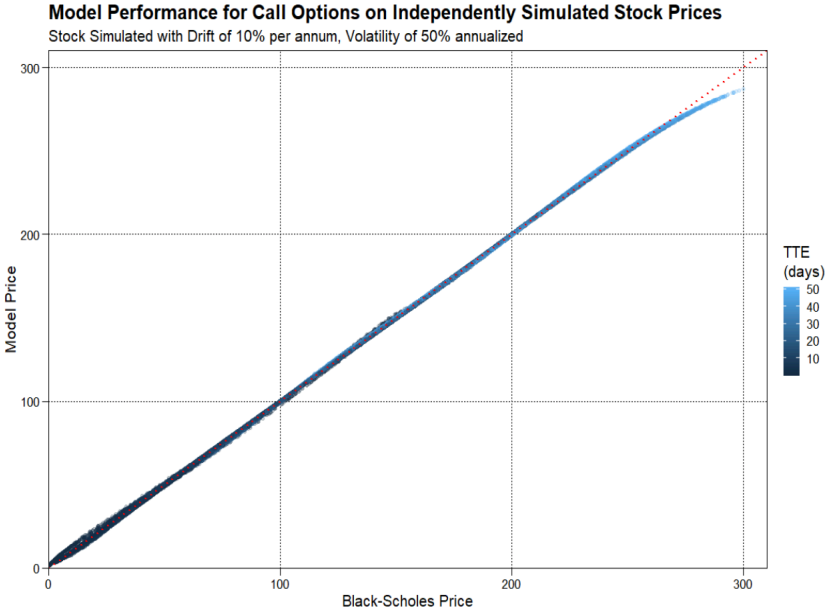
Simulated Stock Performance

The first neural network performed exceptionally well and predicted prices which were largely indistinguishable from the Black-Scholes prices. The following table gives the two key accuracy measures:

R-Squared	99.98
RMSE	0.81

Again, note that this neural network was tested on a data set of

options for a different stock to the one on which it was trained. Furthermore, the spot prices of the underlying stock for the test options were completely independent of the spot prices of the underlying for the training ones. This would suggest that our neural network can be flexibly applied to pricing options on other securities which have the same underlying price dynamics. We show here a plot of the Black-Scholes price of the synthesized options versus what the model predicted the price to be.



We see a near perfect fit/replication except for high priced options, for which our model *undervalues* them relative to the Black-Scholes model. The above graph shows that these tend to be options with high remaining time to expiries (a similar plot colored by "moneyness" showed that these also tend to be those which are deep ITM).

The most likely explanation for the shortcomings of the model at this extremity is that there were insufficient options of this variety in the training set (a byproduct of the option generation procedure). Hence at a macro level, the neural network was not able to perfectly `recon-

struct' the Black-Scholes formula. However, in neighborhoods for which there was a lot of data, the model did perform exceptionally well. This indicates that at the micro level, the neural network could near perfectly "reconstruct" the formula.

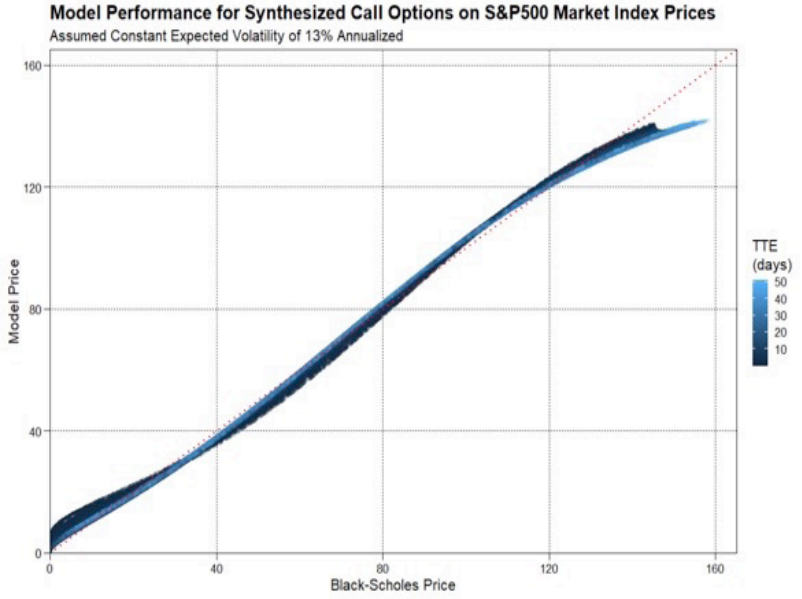
Historic Index Performance

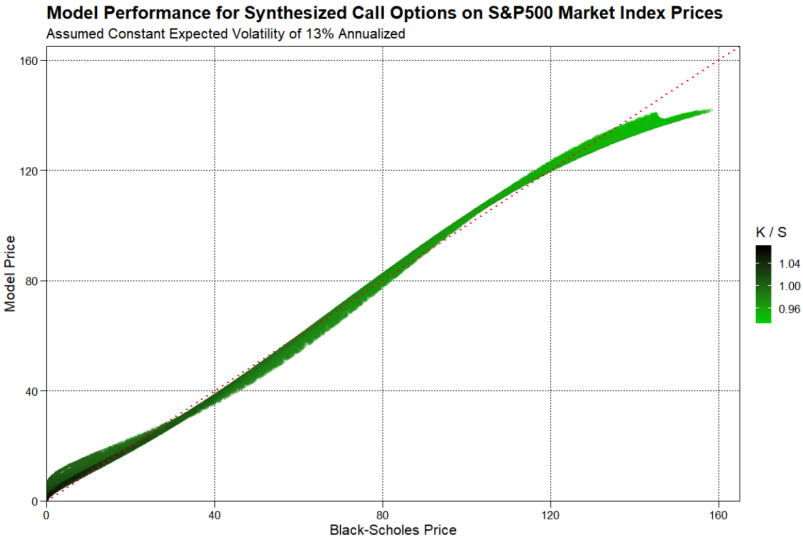
We now look at the second neural network's performance on the historic index derived data set. Again, due to not having a second independent price series with identical price dynamics, the test data set here was a subset of the synthesized option data. Importantly, the test data was not used in any stages of the network's training, so these performance measures are still independent. The following table gives the two key accuracy measures:

R-Squared	99.638
RMSE	2.54

Again, we see excellent fit with almost all variation in the option prices being explained by the model. The performance – particularly in terms of the RMSE – is lower than the first model, however. Before drawing any conclusions, we first look at a similar model performance plot.

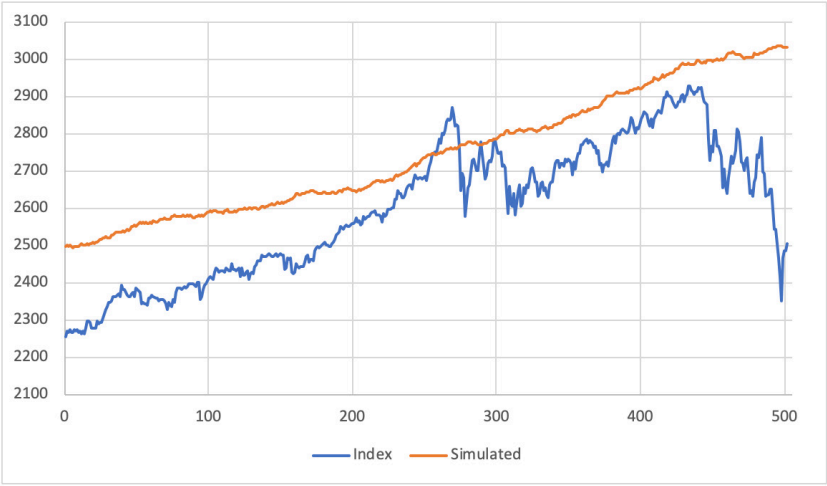
This plot is less straightforward. Firstly, we again see the broad underpricing of high-priced options. We also see the overpricing of cheap options with low time to expiries, followed by the underpricing of moderately priced low time to expiry options. To glean some potential insight, we'll plot the same figure but color the points based on their "moneyness".


Model Performance for Synthesized Call Options on S&P500 Market Index Prices
Assumed Constant Expected Volatility of 13% Annualized
Model Price
Black-Scholes Price
0
40
80
120
160
TTE
(days)
50
40
30
20
10


Model Performance for Synthesized Call Options on S&P500 Market Index Prices
Assumed Constant Expected Volatility of 13% Annualized
Model Price
Black-Scholes Price
0
40
80
120
160
K / S
1.04
1.00
0.96

On its own, this doesn't really tell us much beyond the obvious: the most expensive options are the ones which are furthest ITM. However, we do not to nearly the same degree pick up on the systematic mispricing of options with certain strike/spot ratios like we do in Graph 2 with certain time to expiries. We can infer, thus, that the neural network is not as ideally incorporating the time to expiry variable in its predictions as would be required to perfectly "reconstruct" the Black-Scholes formula.

This deviation could be the result of the more volatile nature of the S&P500 stock returns compared to that of the simulated ones. The below plot shows the spots of the underlying prices for the simulated training stock and the historic S&P500 prices.



We see that the returns are far less smooth with there being many more large drops compared to the simulated prices. This perhaps could have affected the option generation process.

More likely, however, is that the pattern is the result of under-training or from un-tuned hyperparameters. Had we spent time tuning the model's hyperparameters or given the network more time to learn, then we might perhaps achieve a better "global" optimization whereby these inconsistencies in the pricing would be smoothed out.

VI. Conclusion

The major takeaway from this paper is that artificial neural networks can adeptly capture the non-linear relationships between an option's strike, time-to-expiry, & the underlying's spot and the price of a call option implied by the Black-Scholes model. There are countless other "learning" methods which have similar universal approximation results like that of multi-layer perceptrons. Hence, it is reasonable to assume that these other methods will perform similarly or perhaps even better. Some such examples are epsilon-insensitive support vector regression, extreme gradient boosting machines, and projection pursuit regression (Hutchinson et al., 1994: 3). We could also introduce some Bayesian learning techniques to augment many of these methods (Pires, 2005).

The performance of our first network demonstrated to us that this method is flexible and applicable to options on different securities which have similar price dynamics. The performance of our second network indicated that our networks might not necessarily be perfect in capturing the relationship between the time to expiry and the option's price implied by the Black-Scholes model, though this could have been due to the way we synthesized the options.

While the accuracy of the predicted prices was the focus of the paper, we note that this alone is not sufficient to ensure the practical relevance of our approach. The ability to hedge an option position is equally important. Specifically, delta-hedging strategies require an accurate approximation of the partial derivative of the underlying pricing formula with respect to spot. This might be rectified by imposing some smoothness constraint via regularization. MLPs have analytic derivatives (Hermann & Narr, 1997) meaning that we can a *posteriori* always compute it, hence such regularization would be largely superfluous for this paper.

An immediate extension to this paper would be to use real historic option prices on a variety of assets for training and testing. A comparison of the aforementioned machine learning techniques' performances would also be quite interesting. Adding regularization terms to the models would be necessary for such a comparison (Hutchinson et al., 1994: 3).

VII. References

1. Black, F. (1973). The Pricing of Options and Corporate Liabilities. Journal of Political Economy, 81(3), pp. 637-654.
2. Herrmann, R. & Narr, A. (1997). Neural Networks and the Valuation of Derivatives - Some Insights into the Implied Pricing Mechanism of German Stock Index Options. University of Karlsruhe.
3. Hutchinson, J. M., Lo, A. W. & Poggio, T. (1994). A Nonparametric Approach to Pricing and Hedging Derivative Secruities. The Journal of Finance, 49(3), pp. 851-889.
4. Pires, M. (2005). American Option Pricing Using Computational Intelligence Methods, s.l.: University of the Witwatersrand.
5. Wyse, J. (2019). Notes on Stochastic Models in Space and Time I. 1 ed. Dublin: Trinity College Dublin.