Du Plooy, Ryno; Venter, Pierre J.

Article

Pricing vanilla options using artificial neural networks: Application to the South African market

Cogent Economics & Finance

Provided in Cooperation with:
Taylor & Francis Group

Suggested Citation: Du Plooy, Ryno; Venter, Pierre J. (2021) : Pricing vanilla options using artificial neural networks: Application to the South African market, Cogent Economics & Finance, ISSN 2332-2039, Taylor & Francis, Abingdon, Vol. 9, Iss. 1, pp. 1-15, https://doi.org/10.1080/23322039.2021.1914285

This Version is available at:
https://hdl.handle.net/10419/270073

Cogent Economics & Finance

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/oaef20

Pricing vanilla options using artificial neural networks: Application to the South African market

Ryno du Plooy & Pierre J. Venter |

To cite this article: Ryno du Plooy & Pierre J. Venter | (2021) Pricing vanilla options using artificial neural networks: Application to the South African market, Cogent Economics & Finance, 9:1, 1914285, DOI: 10.1080/23322039.2021.1914285

To link to this article: https://doi.org/10.1080/23322039.2021.1914285

Published online: 04 May 2021.

du Plooy & Venter, *Cogent Economics & Finance* (2021), 9: 1914285
https://doi.org/10.1080/23322039.2021.1914285

cogent economics & finance

FINANCIAL ECONOMICS | RESEARCH ARTICLE

Pricing vanilla options using artificial neural networks: Application to the South African market

Ryno du Plooy[1] and Pierre J. Venter[1]*

Received: 15 November 2020
Accepted: 02 April 2021

*Corresponding author: Pierre J. Venter, Department of Finance and Investment Management, University of Johannesburg, Auckland Park 2006, South Africa
E-mail: venter.pierre7@gmail.com

Reviewing editor:
David McMillan, University of Stirling, Stirling, United Kingdom

Additional information is available at the end of the article

Abstract: In this paper, a feed-forward artificial neural network (ANN) is used to price Johannesburg Stock Exchange (JSE) Top 40 European call options using a constructed implied volatility surface. The prices generated by the ANN were compared to the prices obtained using the Black-Scholes (BS) model. It was found that the pricing performance of the ANN significantly improves when the number of training samples are increased and that ANNs are able to price European call options in the South African market with a high degree of accuracy.

Subjects: Financial Mathematics; Quantitative Finance; Machine Learning - Design

Keywords: Artificial intelligence; European call options; financial derivatives; implied volatility; Johannesburg Stock Exchange (JSE); machine learning; neural networks

1. Introduction

This paper aims to answer common questions on machine learning applications in quantitative finance asked by researchers and practitioners. Firstly, whether machine learning techniques such as artificial neural networks (ANNs) can be applied to real-world financial derivative pricing problems and secondly, how these techniques compare to traditional frameworks such as the Black-Scholes (BS) model by (Black & Scholes, 1973). These questions are evaluated by gauging the effectiveness of a feed-forward ANN in pricing European call options using a constructed implied volatility surface in a South African market context.

Numerous studies have been done in the past on the pricing performance of machine learning techniques. (Hutchinson et al., 1994) showed that learning networks can be effectively used to price and hedge derivative securities when traditional parametric models fail. (Bennell & Sutcliffe, 2004)

Ryno du Plooy

ABOUT THE AUTHOR

Ryno du Plooy currently works in industry as a model validation analyst and is pursuing a master's degree in Quantitative Finance at the University of Johannesburg. His research interests include quantitative finance and machine learning.
Pierre J. Venter currently works in industry as a quantitative analyst. He is busy with a PhD in Actuarial Science at the University of Pretoria. He is also appointed as a research associate at the Department of Finance and Investment Management, University of Johannesburg. His research interests include quantitative finance and financial econometrics.

PUBLIC INTEREST STATEMENT

The popularity of machine learning has grown exponentially in recent years with numerous applications spanning multiple industries including the financial industry. The focus of this paper is the development of an artificial neural network to price vanilla European call options using option price data from the market. The findings presented in this paper highlight the potential of machine learning applications in derivative pricing and will hopefully help contribute to future research in the field.

established that the performance of ANNs is significantly improved when the price of the underlying asset and the resultant option price is normalised using the strike price. (Culkin & Das, 2017) found that a deep neural network can be trained using artificially generated input data to accurately price European call options. (Liu et al., 2019) concluded that ANNs are able to calculate option prices and implied volatilities efficiently and accurately and have the ability to function as efficient approximation techniques for asset price processes that require time-consuming computations.

The 21st century has seen the emergence of the fourth industrial revolution, the age of artificial intelligence and automation. With reference to previous studies done on machine learning applications in financial derivative pricing problems, ANNs were shown to be powerful data-driven non-parametric alternatives to traditional derivative pricing models. As stated by (Bennell & Sutcliffe, 2004), ANNs are capable of modelling complex non-linear relationships and are not bound by the restrictive assumptions of traditional models such as the BS model. ANNs do not depend on the assumptions of the underlying stochastic process or require an explicit formula.

The BS model will serve as the benchmark for developing the ANN in this paper since the BS model is considered the market standard for pricing vanilla financial derivatives. The aim is to develop an ANN that is able to learn the BS model by modelling the relationship between the inputs to the BS model and the option prices obtained using the closed-form solution of the BS model. Thus, if it can be shown that ANNs are able to approximate the BS model and accurately price European call options using option price data from the market, then the real-world applications of machine learning techniques can be extended to more complex financial derivatives.

This paper consists of the following sections: Section 2 reviews the recent and relevant literature on the use of ANNs in financial derivative pricing problems. Section 3 consists of the research methodology used in this paper, this includes the theory behind ANNs and how they learn, as well as visiting the BS framework. Section 4 comprises of the data and the results of this paper and finally, the findings and concluding remarks are presented in Section 5.

2. Literature review

The application of machine learning techniques in financial derivative pricing and other areas in quantitative finance has been extensively explored in the past. Resources such as Keras and PyTorch that help facilitate the implementation of machine learning techniques in a straightforward manner and have paved the way for a resurgence in machine learning research focused on financial applications. In this section, the relevant literature on the applications of ANNs in financial derivative pricing is reviewed.

(Hutchinson et al., 1994) explored non-parametric methods for pricing and hedging derivative securities since the dynamics of the underlying security can be learned by these methods with minimal assumptions made on the nature of the underlying process. The study compared the pricing performance of four learning networks namely, ordinary least squares (OLS), radial basis function (RBF) networks, multilayer perceptrons (MLPs) and projection pursuit regression (PPR) to the traditional BS model. In the first section of the study, the learning networks were trained using artificial BS option prices and in the second section, the learning networks were trained using daily S&P 500 futures options observed for a 5-year period from January 1987 to December 1991. The study was also the first to propose the use of the homogeneity hint by (Merton, 1973), which entails scaling the underlying spot price with the strike price to reduce the number of inputs to the learning networks. The study concluded that learning networks are able to accurately price and hedge derivative securities and that non-parametric learning networks are useful substitutes for cases when parametric models fail.

(Bennell & Sutcliffe, 2004) compared the performance of the BS model with an ANN or more specifically a MLP, in pricing European call options on the FTSE 100 index. Data on FTSE 100 European call options traded on the London International Financial Futures and Options Exchange (LIFFE) was collected over a period spanning from 1 January 1998 to 31 March 1999. This resulted in 9,556 observations after cleaning the data set. A key feature the study investigated was to determine if the normalisation of the

spot prices of the underlying and European call option prices produced more favourable results when training the ANN. The training set consisted of data that ranged from 1 January 1998 to 31 December 1998 and the testing set comprised of the remaining data. A third of the training set was further divided into a validation set to test the performance of the ANN during training. Additionally, the data were further categorised into two groups namely, "in-the-money" if the ratio of the spot price scaled by the corresponding strike price is strictly greater than one and "out-of-the-money" if otherwise. It was concluded that normalisation significantly improves the pricing performance of ANNs and is a key property that needs to be incorporated when using ANNs for financial derivative pricing problems.

(Culkin & Das, 2017) applied a deep neural network to the pricing of vanilla European call options and compared the performance of the deep neural network to the BS model using a similar approach followed by (Hutchinson et al., 1994). Artificial input data were generated for each of the parameters which resulted in 300,000 call option prices. The input data were partitioned into 240,000 samples for the training set and 60,000 samples for the validation set. The spot prices of the underlying and call option prices were normalised using the homogeneity hint. The input data were fed into a deep neural network consisting of four hidden layers with 100 neurons in each hidden layer. (Culkin & Das, 2017) noted the importance of selecting an appropriate output activation function to ensure that the output of the deep neural network results in a non-negative European call option price. The study found that simplistic deep neural networks can be trained to price European call options accurately.

In a recent paper by (Liu et al., 2019), the performance of ANNs in the pricing of financial derivatives and the calculation of implied volatility was investigated. The ANN was trained on an artificially generated data set where the trained ANN acted as an agent of three different solvers considered in the study. These solvers were the closed-form solution given by the BS model, the COS (Fourier-cosine series expansions) for the Heston stochastic volatility model and Brent's iterative root-finding method for calculating implied volatilities. The ANN consisted of four hidden layers with hyperparameters such as the number of neurons, activation functions and batch sizes being optimised through the use of a random search three-fold cross-validation (CV) test. Model selection was performed using 200 epochs and the mean-squared-error (MSE) as the loss function. One million random samples for the input parameters were generated from both wide and narrow parameter ranges. The randomly generated input samples were then converted into European call option prices using the BS model. The data set consisted of four inputs, namely the spot price scaled by the strike price, time to maturity, the risk-free rate and the volatility parameter. These inputs to the ANN were used to produce a normalised European call option price as output. The results of the study show that ANNs are able to calculate European call option prices and implied volatilities efficiently and accurately, and that the pricing performance is improved when a wider range of parameters are used to train the ANN. It was also found that ANNs have the ability to serve as approximation techniques for asset price processes that require time-consuming computations.

Two shortcomings associated with previous literature have been identified. First, the training and testing sets either consist only of artificial option price data or real-world option price data. This may skew the accuracy of results and fail to capture how well machine learning techniques are able to generalise on unseen data. Second, the accuracy of the results is measured using performance metrics, but no information is provided on what the actual option prices generated by these techniques are. This paper aims to fill this gap in the literature by training an ANN on artificially generated data and then using the trained ANN to price JSE Top 40 European call options using option price data from the South African market. The European call option prices generated by the ANN are then compared to the prices obtained using the closed-form solution of the BS model to get a better insight on what the actual pricing error is.

The methodology of this paper is presented in the next section.

3. Methodology

In this section, the theoretical framework of the BS model for European call options, background to feed-forward ANNs, as well as general concepts that facilitate the learning process of ANNs will be examined.

3.1. Black-Scholes (BS) model

The value of a financial derivative at any time t should depend only on time and the value of the underlying asset at that time (Shreve, 2008). More formally, the value of a European call option at time t that provides a payoff of $\max\{S_T - K, 0\}$ at maturity T is simply equal to the discounted expected payoff under the risk-neutral measure **Q**:

$$c = e^{-r\tau}\mathbf{E}^{\mathbf{Q}}[\max\{S_T - K, 0\}],$$

where:

- $e^{-r\tau}$ is the discount factor,
- r is an arbitrary risk-free rate,
- τ is the year fraction between the valuation date t and the maturity date T,
- S_T is the spot price of the underlying at maturity T,
- K is the strike price.

Under the assumption of the above-mentioned dynamics, a closed-form solution for the value of a European call option on a non-dividend paying stock is given by:

$$c = S_0\Phi(d_1) - Ke^{-r\tau}\Phi(d_2),$$

where:

- $d_1 = \dfrac{\ln\left(\frac{S_0}{K}\right) + \left(r + \frac{1}{2}\sigma_{BS}^2\right)\tau}{\sigma_{BS}\sqrt{\tau}}$,
- $d_2 = \dfrac{\ln\left(\frac{S_0}{K}\right) + \left(r - \frac{1}{2}\sigma_{BS}^2\right)\tau}{\sigma_{BS}\sqrt{\tau}}$,
- $\Phi(\cdot)$ is the cumulative normal distribution function,
- σ_{BS} is the implied volatility parameter,
- S_0 is the spot price of the underlying at the valuation date $t = 0$.

3.2. Artificial neural networks

In the most basic sense, an ANN can be visualised as an artificial representation of the biological brain. ANNs consist of numerous layers of interconnected information processing units called neurons that make up the structure of the ANN. These neurons receive information in the form of outputs from neurons in the preceding layer and process this information using an activation function. The processed information is transferred forward through the network on a neuron-by-neuron basis (Haykin, 1999). (Cybenko, 1989) and (Hornik et al., 1989) proved that a feed-forward ANN with a single hidden or internal layer and a continuous non-linear activation function, such as the sigmoidal function, can approximate any continuous function with precision. A simple diagram of the proposed architecture of the ANN used in this paper can be seen in Figure 1.

cogent economics & finance

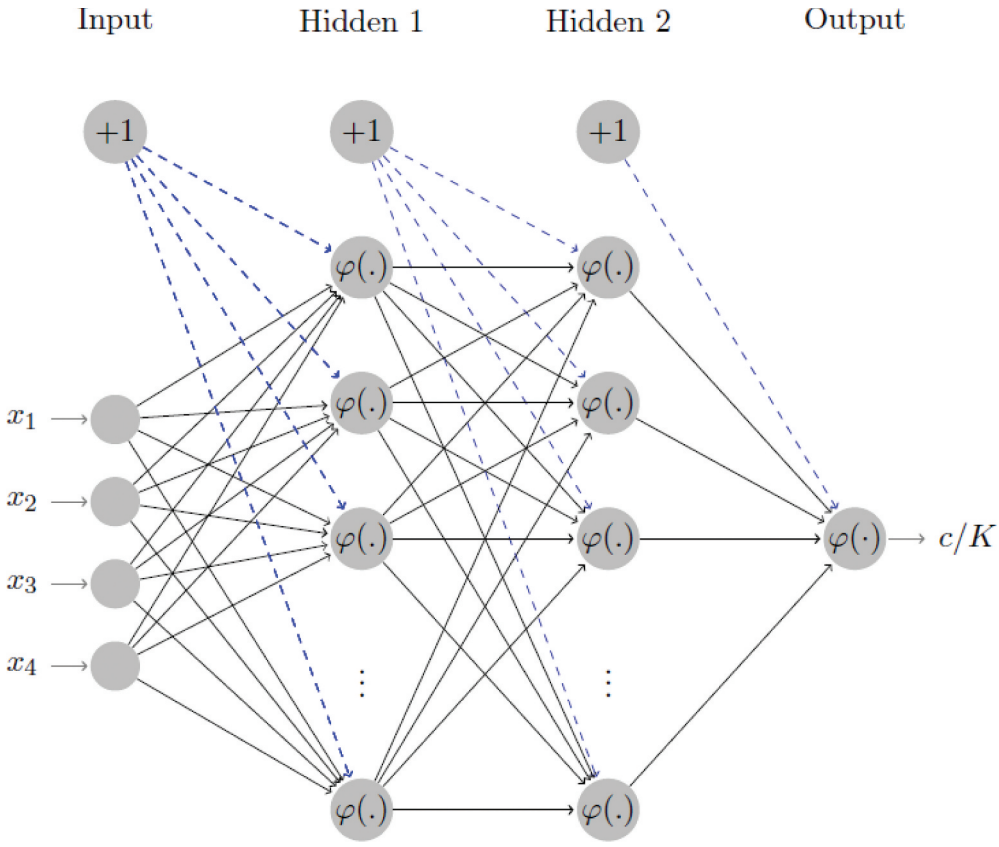


Figure 1. General representation of an Artificial Neural Network (ANN).

where:

- $x_1 = S_0/K$,
- $x_2 = \tau$,
- $x_3 = r$,
- $x_4 = \sigma_{BS}$,
- $\varphi(\cdot)$ is an arbitrary activation function,
- $+1$ is the bias term,
- c/K is the normalised European call option price.

3.2.1. Forward propagation

The flow of information in an ANN as described by (Haykin, 1999) is characterised by a forward pass of the information in a sequential manner throughout the network, which is known as forward propagation. To help ease the burden of notation, let the indices *i* and *j* denote neurons in different layers of the ANN, where neuron *i* is located in the layer to the left of neuron *j*. By making use of this notation, the flow of information is given by:

$$a_j(n) = \varphi_j\left(\sum_{i=0}^{M} w_{ji}(n)a_i(n)\right), i = 0, 1, \ldots, M,$$

where:

- n denotes a single iteration or forward pass of information,
- M is the number of neurons in the previous layer,
- $a_j(n)$ is the output of neuron j,
- $\varphi_j(\cdot)$ is an arbitrary activation function in a certain layer,
- $w_{ji}(n)$ is the synaptic weight connecting the *ith* and *jth* neuron,
- $a_i(n)$ is the output of neurons in the previous layer,
- $w_{j0}(n)$ is connected to a fixed input $a_0 = +1$.

3.2.2. Gradient-based learning

(Rumelhart et al., 1986) derived the formal framework for the learning procedure of ANNs known as the back-propagation algorithm. A feed-forward ANN will calculate the difference between the actual output $y(n)$ and the predicted output $a(n)$ using an arbitrary loss function $J(n)$, which is a function of the iteration n, and subsequently update the synaptic weights in order to minimise the difference between the actual and predicted outputs. To better illustrate the mechanics behind the back-propagation algorithm, consider a single neuron j which is connected to neuron i in the previous layer by $w_{ji}(n)$. The process of updating the synaptic weight $w_{ji}(n)$ under the generalised delta rule for a feed-forward ANN according to (Haykin, 1999), can be formally defined as:

$$\Delta w_{ji}(n) = \eta\delta_j(n)a_i(n), \tag{1}$$

or in an alternative form as:

$$\Delta w_{ji}(n) = -\eta\frac{\partial J(n)}{\partial w_{ji}(n)}, \tag{2}$$

where:

- n denotes a single iteration,
- η is the learning parameter,
- $\Delta w_{ji}(n)$ is the adjustment to the synaptic weight connecting the *ith* and *jth* neuron,
- $\delta_j(n)$ is the local gradient,
- $\frac{\partial J(n)}{\partial w_{ji}(n)}$ is partial derivative of the loss function with respect to the synaptic weight connecting the *ith* and *jth* neuron.

The representation of the local gradient $\delta_j(n)$ can change depending on whether neuron j is located within the output layer or within a hidden layer.

To help with deriving an expression for $\delta_j(n)$ in equation (1), when a neuron j is located within the output layer containing neurons equal to C, it is useful to first define the following:

$$J(n) = \frac{1}{2}\sum_{j\in C}\varepsilon_j^2(n),$$

$$\varepsilon_j(n) = y_j(n) - a_j(n),$$

$$a_j(n) = \varphi_j\big(z_j(n)\big),$$

Through the application of the chain rule of calculus, the local gradient $\delta_j(n)$ in equation 1 when neuron j is located in the output layer can be defined as:

$$\delta_j(n) = -\frac{\partial J(n)}{\partial z_j(n)}$$

$$= -\frac{\partial J(n)}{\partial \varepsilon_j(n)}\frac{\partial \varepsilon_j(n)}{\partial a_j(n)}\frac{\partial a_j(n)}{\partial z_j(n)}$$

$$= \varepsilon_j(n)\varphi_j'\big(z_j(n)\big). \tag{3}$$

For the second case, let neuron j be located within one of the hidden layers, where neuron j is connected to neuron i in the previous layer by $w_{ji}(n)$ and to neuron k in the output layer by $w_{kj}(n)$. It is possible to define a new set of functions to derive an expression for $\delta_j(n)$ when neuron j is located within a hidden layer as:

$$J(n) = \frac{1}{2}\sum_{k\in C}\varepsilon_k^2(n),$$

$$\varepsilon_k(n) = y_k(n) - a_k(n),$$

$$a_k(n) = \varphi_k(z_k(n)),$$

$$z_k(n) = \sum_{j=0}^{\kappa} w_{kj}(n)a_j(n),$$

$$a_j(n) = \varphi_j\big(z_j(n)\big),$$

where the number of neurons in the output layer is equal to C and the number of neurons in the hidden layer preceding the output layer is equal to κ. Using the chain rule, the local gradient $\delta_j(n)$ in equation (1) when neuron j is located within a hidden layer can be defined as:

$$\delta_j(n) = -\frac{\partial J(n)}{\partial z_j(n)}$$

$$= -\frac{\partial J(n)}{\partial \varepsilon_k(n)}\frac{\partial \varepsilon_k(n)}{\partial a_k(n)}\frac{\partial a_k(n)}{\partial z_k(n)}\frac{\partial z_k(n)}{\partial a_j(n)}\frac{\partial a_j(n)}{\partial z_j(n)}$$

$$= \varphi_j'(z_j(n))\sum_k \varepsilon_k(n)\varphi_k'(z_k(n))w_{kj}(n). \tag{4}$$

Using the solution for $\delta_j(n)$ in equation 3 for the expression $\varepsilon_k(n)\varphi_k'(z_k(n))$ and replacing the subscript j with k, equation 4 can be rewritten as:

$$\delta_j(n) = \varphi_j'(z_j(n))\sum_k \delta_k(n)w_{kj}(n). \tag{5}$$

3.2.3. Activation functions

Activation functions are fundamental to how feed-forward ANNs process the information received from the neurons in the preceding layer. An activation function can also be referred to as a "squashing function" since the activation function limits the output of a neuron to a specific range of values. A key property for an activation function as stated by (Haykin, 1999) is that the function must be differentiable. The rectified linear unit (ReLU) and Softplus functions used in this paper are defined as:

(1) ReLU

$$\varphi(z) = \begin{pmatrix} 0 & \text{for} z<0 \\ z & \text{otherwise}, \end{pmatrix}$$

(2) Softplus

$$\varphi(z) = \ln(1+e^z).$$

3.2.4. Performance metrics

Performance metrics are fundamental to the evaluation of how well an ANN performs when comparing the predicted outputs with the actual outputs. The mean-squared error (MSE), root-mean-square error (RMSE) and the coefficient of determination (R^2) are some of the most widely used metrics when evaluating regression-based machine learning problems (Albon, 2018). These metrics are defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2},$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2},$$

where:

- N is the number of observations,
- y_i is the actual value,

- $\hat{y}_i$ is the predicted value,
- $\bar{y}$ is the mean of the actual values.

The data generating process, ANN architecture and results of this paper will be presented in the next section.

4. Results

This section covers the data generation process for the training, validation and testing data set, ANN architecture as well as the results of using a feed-forward ANN to generate a European call option price surface from an implied volatility surface.

4.1. Data

The following section describes the process followed to generate artificial European call option data for training and validating the ANN. The section also includes the process involved with transforming the constructed implied volatility surface into a call option price surface that will be used as the testing set to evaluate the pricing performance of the ANN.

4.1.1. Training and validation data

Since high-quality option price data are scarce in the South African market due to illiquidity, it was necessary to resort to generating artificial training data. The artificial training data were randomly sampled from wide ranges for the input parameters which were transformed into European call option prices using the closed-form solution of the BS model. Some of the parameter ranges were exaggerated to enhance the uniqueness of the artificially generated data, resulting in a substantial number of input parameter combinations to aid the ANN in generalising on unseen data. To better understand the relationship between the number of training samples and the pricing performance of an ANN, two separate training sets were artificially generated where an ANN is trained on each respective set. These two cases are represented as:

- ANN Case 1: ANN trained on 200,000 artificially generated data samples.
- ANN Case 2: ANN trained on 1,000,000 artificially generated data samples.

The same parameter ranges were also used to generate the validation sets consisting of 20% of the number of training samples for each case. The parameter ranges used for generating the two artificial training sets can be seen in Table 1.

Table 1. Artificial input data parameter ranges

Parameter	Range
Underlying spot price (S_0)[1].	[R10,000, R200,000]
Strike price (K)	[60%, 140%] of S_0
Time to maturity[2] (τ)	[7/365, 3]
Risk-free rate (r)	[0%, 30.00%]
Implied volatility (σ_{BS})	[2.00%, 80.00%]

1. Prices are denoted in Rand (R), which is the domestic currency for South African markets.
2. The Actual/365 day-count convention is used in South African markets.

After obtaining call option prices via the closed-form solution of the BS model using the input data generated for both cases, a similar approach to that of (Hutchinson et al., 1994; Bennell & Sutcliffe, 2004; Culkin & Das, 2017; Liu et al., 2019) was used to normalise the features of the data.

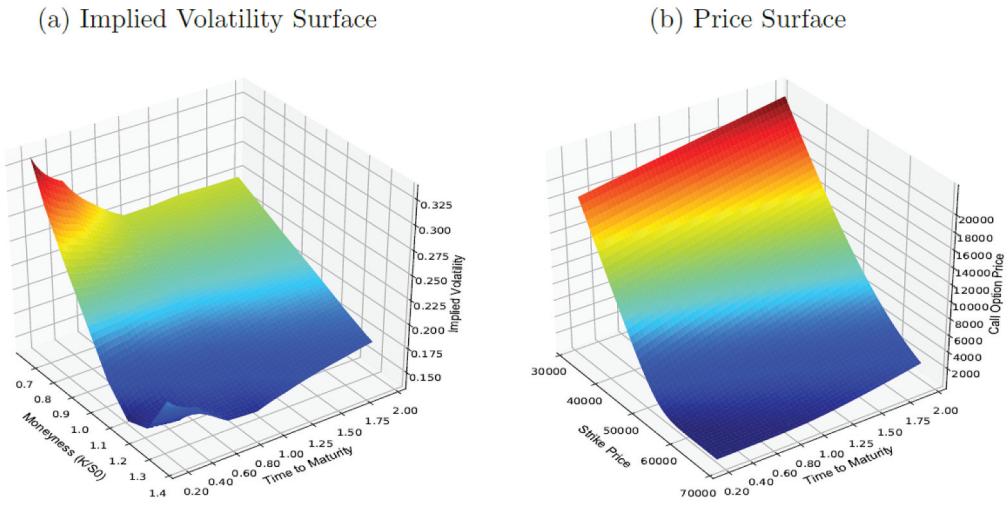
By applying the homogeneity hint by (Merton, 1973), the inputs for the training and validation set are: $x_{Train} = \{S_0/K, \tau, r, \sigma_{BS}\}$ and $x_{Val} = \{S_0/K, \tau, r, \sigma_{BS}\}$ respectively. The homogeneity hint was also applied to the European call option prices obtained from the artificial training and validation data, which resulted in outputs of the form: $y_{Train} = \{c/K\}$ and $y_{Val} = \{c/K\}$.

4.1.2. Testing data

The testing data for the ANN is in the form of an All Share Index (ALSI) implied volatility surface constructed using data obtained from the JSE dated 9 April 2019. A granular implied volatility surface was constructed by assuming linear interpolation in strike, and linear variance in the maturity space. Both of these interpolation techniques are consistent with market best practices. This resulted in a constructed implied volatility surface consisting of 10,000 implied volatility estimates. The price of the underlying index (JSE ALSI Top 40) on the valuation date was R51,564.09 and the risk-free rate was assumed to be equal to the 3-month T-bill rate of 7.01%, which was obtained from the South African Reserve Bank (SARB) on the valuation date. It was also assumed that the continuous dividend yield is equal to zero.

The BS model was used to convert the input parameters into call option prices. The homogeneity hint was once again used to normalise the testing set, which resulted in the following inputs: $x_{Test} = \{S_0/K, \tau, r, \sigma_{BS}\}$ and outputs: $y_{Test} = \{c/K\}$. No additional feature scaling was performed on the other input parameters. The JSE ALSI implied volatility surface and BS price surface are illustrated in Figure 2.

Figure 2. JSE ALSI top 40 call options.



4.2. Artificial neural network architecture

The ANN in this paper was implemented in Python using the Keras Sequential Application Programming Interface (API) based on TensorFlow 2.0, which was initially developed by (Chollet, 2015). Calculations were performed on a Dell Inspiron 3567 i5-7200 U CPU @ 2.50 GHz with 4GB of installed RAM. The following ANN architecture was proposed to price JSE Top 40 European call options, as displayed in Table 2.

To facilitate a dynamically driven process of training the ANN, validation losses were monitored after each epoch to prevent the model from over-fitting. Thus, if the ANN's performance starts degrading after a few epochs, the training process will automatically be terminated. This dynamic

Table 2. Proposed neural network architecture

Parameter	Configuration
Number of hidden layers	2
Neurons in first hidden layer	256
Neurons in second hidden layer	128
Neurons in output layer	1
Hidden layer activation function	ReLU
Output layer activation function	Softplus
Optimizer	Adam
Batch size	64
Epochs	20

training process can further be enhanced by using model checkpoints, which stores the ANN configuration after each epoch. Once training is completed, the optimal ANN configuration is saved, which can then be deployed to price European call options.

A very important feature to consider as highlighted by (Culkin & Das, 2017), is the use of an output activation function which results in a non-negative price. It is common practice to use a linear function in the output layer for financial derivative pricing problems, but this may result in occasional negative prices since the linear function is zero-centred. According to (Hull, 2009), the lower bound of a European call option at time $t = 0$ is given by:

$$c \geq \max\{S_0 - Ke^{-r\tau}, 0\}.$$

This inequality holds due to the optionality embedded in the option. Since the holder of the option has the right but not the obligation to exercise the option. Thus, the value of a European call option cannot be negative. To avoid the violation of this fundamental property, the Softplus function was chosen to be the output layer activation function.

4.3. Numerical results

After extensively training an ANN on each artificially generated training set, the trained ANN from both cases was deployed into production. Given input data consisting of the implied volatility surface and other parameters observed on the valuation date, the ANN from both cases were used to generate the prices of JSE Top 40 European call options. These predicted prices of the form c/K were compared to the actual prices obtained by converting the implied volatility surface into a call option price surface using the BS model. The actual prices generated by the BS model were also scaled to be in the form c/K. The performance on the testing set can be seen in Table 3.

Table 3. Actual versus predicted results (c/K) performance metrics

Case Number	MSE	RMSE	R^2
ANN Case 1	1.84E-06	0.001357	0.999944
ANN Case 2	4.11E-07	0.000641	0.999988

From Table 3, it can be seen that the ANN in both cases produced accurate results. The MSE and RMSE improved in the second case due to the greater number of training samples. The R^2 reported for both cases were close to one. A graphical representation of the performance of the ANN from both cases on the testing set can be seen in Figure 3.

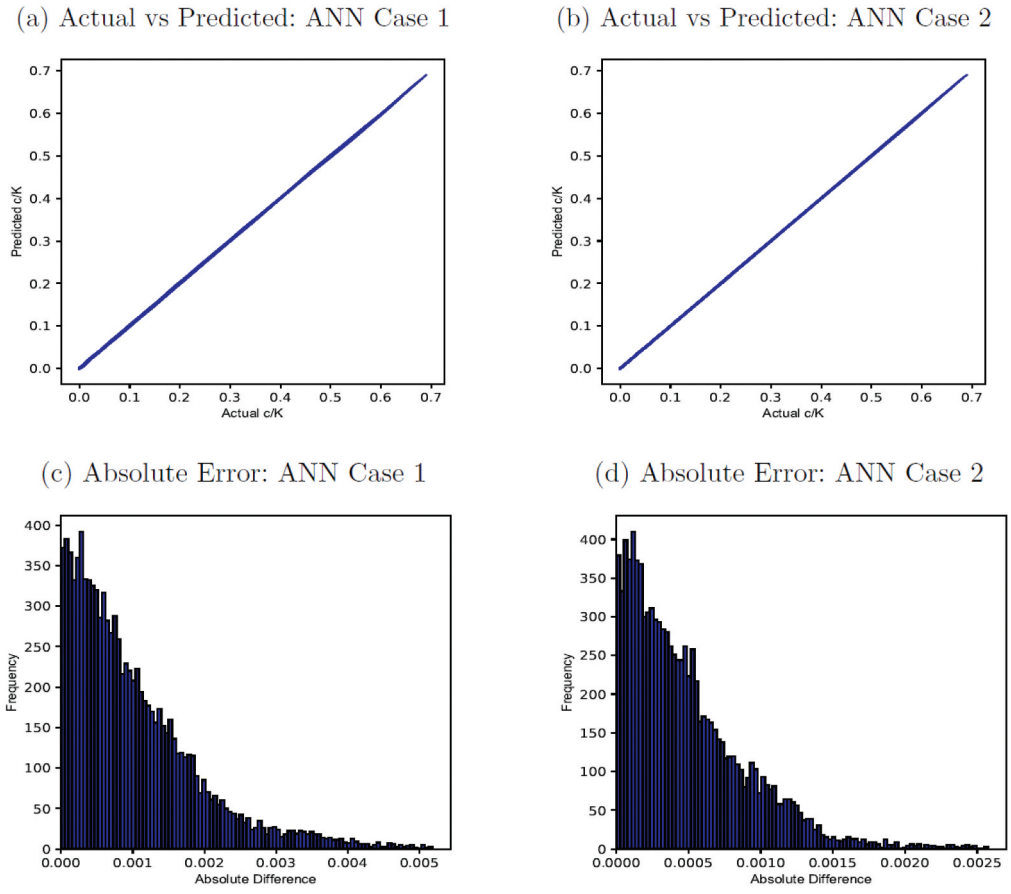


Figure 3. Actual versus predicted results (c/K).

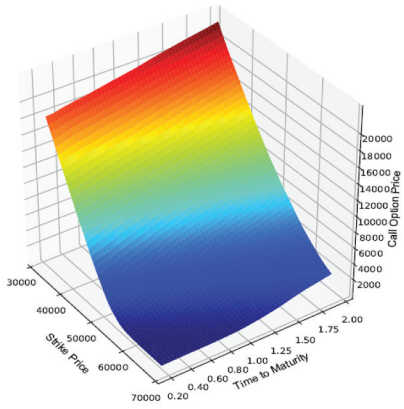
Although the results in Figure 3 are promising, real-world applications demand a non-scaled price. This is achieved by multiplying the generated European call option prices with the corresponding strike price *K*, which results in a price for every point on the implied volatility surface. The price surfaces generated by the ANN from both cases compared to the actual price surface in Figure 2b are illustrated in Figure 4.

By evaluating Figure 4a and 4b, it is evident that the ANN from both cases produced a European call option price surface almost graphically identical to the original price surface. From the absolute errors in Figure 4c and 4d, it can be seen that the magnitude of differences reduced significantly when the number of training samples were increased in the second case. This same observation can be made when comparing the relative errors in Figure 4e and 4f. The size of the relative error for deep "out-of-the-money" short-dated European call options is quite significant compared to the rest of the price surface. This overestimation bias is, however, solely attributable to the Softplus function used in the output layer of the ANN since the function is not zero-centred and the value of deep "out-of-the-money" options are very close to zero. A simple solution to this phenomenon is to increase the number of samples used to train the ANN or to search for an alternative output layer activation function. A more detailed view on the results obtained can be seen in Table 4.

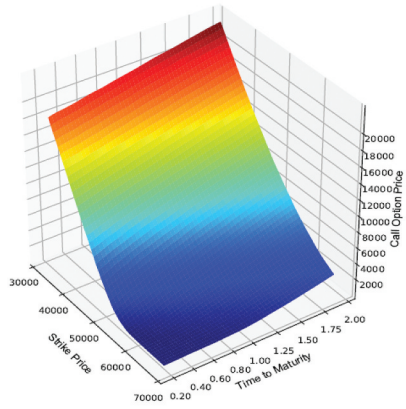
cogent economics & finance

Figure 4. JSE ALSI top 40 call options.

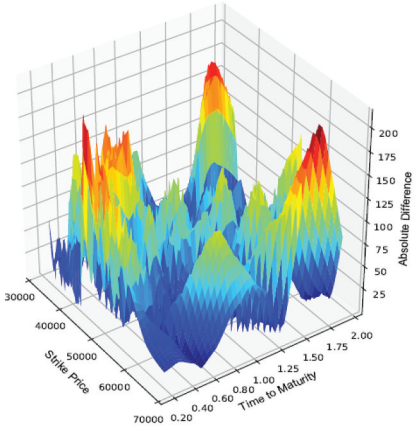
(a) Price Surface: ANN Case 1



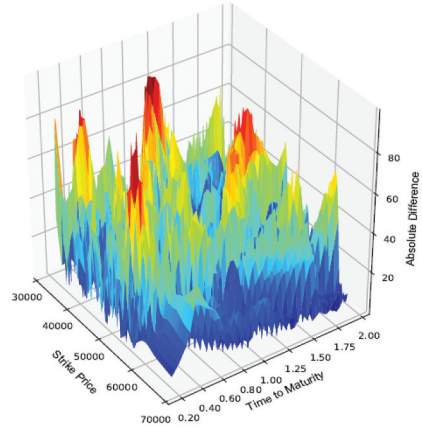
(b) Price Surface: ANN Case 2



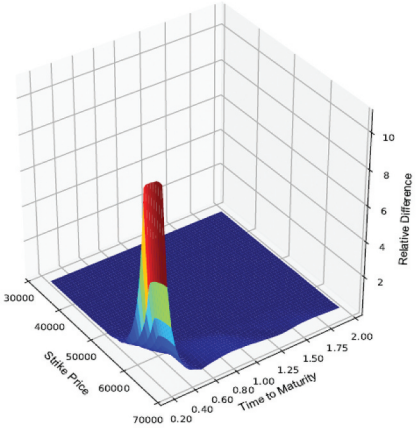
(c) Absolute Error: ANN Case 1



(d) Absolute Error: ANN Case 2



(e) Relative Error: ANN Case 1



(f) Relative Error: ANN Case 2

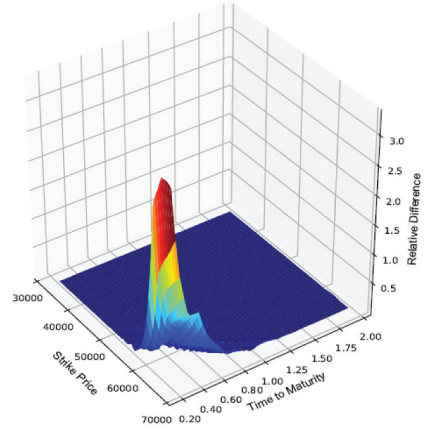


Table 4. ANN versus BS test results

Metric	BS Model	ANN Case 1	ANN Case 2
Min price	R1.32	R15.78	R5.66
Max price	R22,881.99	R22,886.14	R22,887.78
Min absolute error	N/A	R0.01	R0.01
Max absolute error	N/A	R214.72	R98.89

From Table 4, it is evident when considering the range of absolute errors that the pricing performance of the ANN significantly improved in the second case where the number of training samples were increased to 1,000,000 samples. The findings of this paper will be concluded in the next section.

5. Conclusion

The purpose of this paper was to investigate the performance of ANNs when applied to the pricing of European call options in the South African market. The research question can be divided into two parts, firstly, given a feed-forward ANN trained on artificial data, what resultant European call option prices will be generated from an implied volatility surface. Secondly, what degree of error can be expected when comparing the generated prices to prices obtained using a traditional option pricing model such as the BS model.

This paper revisited previous literature on the use of ANNs in modern financial derivative pricing problems such as the work done by (Hutchinson et al., 1994; Bennell & Sutcliffe, 2004; Culkin & Das, 2017; Liu et al., 2019). By making use of an approach that is consistent with previous studies, artificial European call option price data was generated, this resulted in the creation of two training sets. The first training set consisted of 200,000 samples and the second training set consisted of 1,000,000 samples. After training an ANN on each of the training sets, it was found that for both cases, an ANN is able to price European call options with a high degree of accuracy when given an implied volatility surface constructed using option price data from the South African market. It was also found that an increase in the number of training samples resulted in a significant improvement in the pricing performance of an ANN.

Areas for further research include investigating sampling techniques to generate higher quality input data that will result in the need for fewer training samples and thus be computationally less expensive. Furthermore, modern pricing frameworks that incorporate collateral and valuation adjustments should also be considered. Finally, the performance of ANNs applied to exotic options in the South African market should also be investigated.

Acknowledgements
The authors would like to thank the anonymous referees for their insightful feedback that helped improve the quality of this article considerably. A special thanks also to Alexis Levendis for reviewing the article.

Funding
The authors received no direct funding for this research.

Author details
Ryno du Plooy[1]
Pierre J. Venter[1]
E-mail: venter.pierre7@gmail.com
ORCID ID: http://orcid.org/0000-0001-7304-5806
[1] Department of Finance and Investment Management, University of Johannesburg, Auckland Park, 2006, South Africa.

Supplementary materials
Supplemental data for this article can be accessed here.

Citation information
Cite this article as: Pricing vanilla options using artificial neural networks: Application to the South African market, Ryno du Plooy & Pierre J. Venter, *Cogent Economics & Finance* (2021), 9: 1914285.

References

Albon, C. (2018). *Machine learning with python cookbook. practical solutions from preprocessing to deep learning*. O'Reilly Media.

Bennell, J., & Sutcliffe, C. (2004). Black-scholes versus artificial neural networks in pricing FTSE 100 options. *Intelligent Systems in Accounting, Finance and Management*, *12*(4), 243–260. https://doi.org/10.1002/isaf.254

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637–654. https://doi.org/10.1086/260062

Chollet, F. (2015). *Keras*. Available: https://github.com/fchollet/keras.

Culkin, R., & Das, S. R. (2017). Machine learning in finance: The case of deep learning for option pricing. *Journal of Investment Management*, *15*(4), 92–100. https://srdas.github.io/Papers/BlackScholesNN.pdf

Cybenko, G. (1989). Approximation by superstitions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*(4), 303–314. https://doi.org/10.1007/BF02551274

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2*nd* ed.). Prentice Hall.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Hull, J. C. (2009). *Options, futures and other derivatives* (7*th* ed.). Prentice Hall.

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, *49*(3), 851–889. https://doi.org/10.1111/j.1540-6261.1994.tb00081.x

Liu, S., Oosterlee, C. W., & Bohte, S. M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks*, *7*(1), 1–22. https://doi.org/10.3390/risks7010016

Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, *4*(1), 141–183. https://doi.org/10.2307/3003143

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Shreve, S. E. (2008). *Stochastic calculus for finance II: Continuous-time models* (2*nd* ed.). Springer.

***Cogent Economics & Finance* (ISSN: 2332-2039) is published by Cogent OA, part of Taylor & Francis Group.**