Pricing Options with an Artificial Neural Network: A Reinforcement Learning Approach

Haakon A. Trønnes

Department of Economics

Norwegian University of Science and Technology

June 2018

Forord

Denne masteroppgaven er en del av min master i finansiell økonomi ved NTNU. Jeg vil takke min veileder, Joakim Blix Prestmo for god veiledning og interessante diskusjoner. Jeg vil også takke mamma og pappa for deres støtte og innspill.

Abstract

I develop and present a non-parametric and empirical method for pricing derivative securities. The method involves estimating an artificial neural network. The estimation is based on a time series of the underlying asset price and relies on the no-arbitrage argument. I focus on the pricing of European call options. To assess the feasibility of the method I first apply it on a simulated data set, satisfying the assumptions of the Black-Scholes model. The results show that the method is able to accurately estimate both the option price and its derivatives, based on a two-year sample of the price of the underlying asset. Further, I apply the method on the S&P500 index, with data from 2014 to 2016. I compare the out-of-sample performance of my method to two rival models, the Black-Scholes model and a traditional non-parametric method. The models are evaluated on both pricing and delta-hedging. My method outperforms the Black-Scholes model in this analysis, but is mostly outperformed by the other non-parametric method.

Oppsummering

Jeg utvikler og presenterer en ikke-parametrisk empirisk metode for prising av finansielle derivater. Metoden involverer estimeringen av et kunstig nevralt nettverk basert på en tidsserie av det underliggende aktivum og arbitrasje-argumentet. Jeg fokuserer på prising av europeiske kjøpsopsjoner. Jeg undersøker om metoden er levedyktig ved å anvende den på et simulert datasett, som oppfyller forutsetningene til Black-Scholes-modellen. Resultatene viser at metoden kan predikere både opsjonsprisen og dens deriverte med høy nøyaktighet, basert på to år med daglige observasjoner av prisen til det underliggende aktivum. Videre anvender jeg metoden på S&P500 indeksen, med data fra 2014 til og med 2016. Jeg sammenlikner metodens egenskaper med to konkurrerende modeller, Black-Scholes-modellen og en tradisjonell ikke-parametrisk metode. I analysen gir min metode bedre resultater både i prising og delta-hedging av opsjoner enn Black-Scholes-modellen. Sammenliknet med den andre ikke-parametrisk metoden, oppnår min metode svakere resultater i de fleste tilfeller.

Contents

1 Introduction

Financial derivatives, particularly options, are of interest both because they are popular financial products that trade in high volumes, and also because of their application in pricing other assets, e.g. corporate debt, and risk forecasting and analysis, e.g. the VIX-index. Therefore, a new approach to pricing them, with some clear advantages over established methods, is of interest to researchers and practitioners alike.

In this Master's thesis I develop and present a novel method for pricing financial derivatives. Though the method can be applied to price a wide range of derivative securities, I have chosen to focus mainly on European call options.

A European call option is characterized by the underlying asset on which it is written, the time until its expiration date, T, and the strike price, K. It gives its holder the right, but not the obligation, to purchase the underlying asset for the strike price, on and only on, the expiration date. The value of the European call option is therefore quite easy to determine on its expiration date. If the price of the underlying asset, S, exceeds the strike price, the option is exercised for a profit of $S - K$. If the price of the underlying asset does not exceed the strike price, it is not beneficial to exercise the option, so the value is 0. The value of a European call option on its expiration day is therefore $max\{S - K, 0\}$. A much more difficult problem, and the subject of this paper, is determining its value before expiration.

Black and Scholes (1973) showed that if the price of an asset follows a particular stochastic process, geometric Brownian motion, a European option on it can be priced exactly, using a no-arbitrage argument. This is a good approximation, but real world stock prices are not generated by geometric Brownian motion. Several empirical studies show that the Black-Scholes prices of options deviate in systematic ways from real world observed prices [(MacBeth and Merville 1979), (MacBeth and Merville 1980), (Rubinstein 1985), (Black and Scholes 1973)].

Since the publication of the Black-Scholes model, several other parametric models have been proposed for different underlying processes. Merton (1976) proposed a model where the price of the underlying asset is generated by the sum of a diffusion process, geometric Brownian motion, and a (discontinuous) jump process. Cox, Ross, and Rubinstein (1979) propose a discrete binomial model, where the price of the underlying asset moves up or down by a factor at a fixed frequency. Heston (1993) finds a closed form solution for the option price on an asset following geometric Brownian motion with stochastic volatility.

By using a non-parametric model, the issue of identifying and modeling the stochastic process of the asset can be completely sidestepped. Hutchinson, Lo, and Poggio (1994) used several non-parametric models, an artificial neural network (ANN) among them, to price options. These models were estimated by minimizing the mean squared error of the model predictions compared to market prices. Meaning that instead of using a no-arbitrage argument like earlier works, they assume that historical market prices were correct. There are downsides to this approach. First, to estimate a good model a large amount of data is required. This makes the method only useful for assets/markets where there is a history of an active/liquid option market. Second, if the historical market prices have biases, the model is likely to learn them as well.

This paper attempts to solve issues associated with both the parametric no-arbitrage models, like Black and Scholes (1973), and the non-parametric models, like Hutchinson, Lo, and Poggio (1994), by developing a non-parametric model estimated using a no-arbitrage argument. This is implemented using an artificial neural network, trained only on the historic price of the underlying asset, not the historical prices of the associated options. This way, the model will not be relying on a (mis)specified stochastic process, or limited historical options price data. For reasons that will become clear later, I will call this model the reinforcement ANN. The question I attempt to illuminate in this text is, can the reinforcement ANN improve upon established methods for pricing options? If so, under what circumstances is it most useful?

The remaining sections of this paper are organized as follows; in section 2, I build the theoretic foundation for the reinforcement ANN and discuss established models. Section 3 presents the reinforcement ANN and how it is estimated. In section 4, I test the model in a controlled environment using simulated data. Next, in section 5, I apply the model on real market data and compare its performance to established models. Summary and discussion are in section 6.

2 Theory and Previous Literature

In this section I lay the theoretical foundation for my new model, which will be introduced in section 3. This is includes the basic financial reasoning, the hedging or replicating portfolio, and the regression model of choice, artificial neural networks.

I also review previous literature and present the models I will use for comparison in section 5.

2.1 Hedging and the No-Arbitrage Argument

There are many reasons for buying or selling options. An investor may buy a put option on her portfolio as insurance against declines, a speculator may buy a call option on a stock as a bet on the stock increasing in value, or sell both put and call options as a bet against volatility. These are all examples of market actors buying or selling options as part of an investment strategy. They all hope to make money from capital gains or other investment income (e.g. dividends, interest, or coupon payments).

The goal of the market maker is fundamentally different. A market maker buys at the bid price, and sells at the ask price. It is in this spread that they find their profit. As they are not investing, market makers attempt to neutralize their exposure to market risk. This is done through hedging.

Hedging means offsetting the risk of one position with another. For example, if a market maker sold a call option on a stock, their position will fall in value as stock increases in value. This risk can be offset by buying shares in the stock. That way, if the value of the stock should rise (fall) the loss (gain) in the call option position will be offset by the gain (loss) in the stock position. This raises the question, how many shares of stock should the market maker buy to best offset the risk exposure on the call option?

Let $C(S,K,T)$ denote the value of the option, then the change in value of the option for a small change in the stock price is $\frac{\partial C(S,K,T)}{\partial S}$. $\frac{\partial C}{\partial S}$ is often denoted Δ[1]. Holding Δ shares of stock would perfectly offset changes in the value of the option caused by small movements in the stock price. This is called delta-hedging. The stock portfolio can be viewed as a first-order Taylor series approximation of the option with respect to the asset price. To see this, let $T(S)$ be the first order Taylor series expansion of $C(S,K,T)$ around the point (a,b,c). Then

$$T(S) = C(a,b,c) + (S-a)\Delta \tag{1}$$

The change in value of the option from time t to $t+h$ is

$$C(S_{t+h}, K, T-h) - C(S_t, K, T) \tag{2}$$

where S_t denotes price of the underlying asset at time t. The first order Taylor approximation of (2) is

$$T(S_{t+h}) - T(S_t) = (S_{t+h} - S_t)\Delta \tag{3}$$

As will be clear later, this forms the basis for the ANN model introduced in section 3. However, there are some shortcomings to this simplified approach.

Firstly, as time goes by, the price of the underlying asset is not the only thing that is changing. The time until expiration is also decreasing, and this is not taken into account in this simplified approach. Secondly, if the price of the underlying asset is driven by a Brownian motion, a very small change in time, even infinitesimal (dt), does not imply that the move in asset price

[1]The partial derivatives of derivative securities, e.g. options, have a Greek symbol associated with it. Therefore, they are often called the "Greeks" collectively.

is very small. If driven by a Brownian motion, or something like it, the asset price will be so volatile, that the price can and will move finite amounts over infinitesimally small time intervals (the quadratic variation is nonzero). For instance, let $Z(t)$ be a standard Brownian motion, then

$$Z(t+h) - Z(t) \sim N(0, h) \tag{4}$$

Equation (4) implies that the standard deviation of a change in the process $Z(t)$ over an interval h is $\sqrt{h}$. As the time interval h decreases, the standard deviation $\sqrt{h}$ decreases more slowly, meaning that as $\lim_{h\to 0} h = dt$ the standard deviation will not be infinitesimally small. This is a problem as the Taylor series approximation will only be accurate for very small changes, if the function being approximated is non-linear. Therefore, even over infinitesimally small time intervals, a first order Taylor series approximation with respect to asset price is not sufficient.

This leads us to the Black-Scholes model.

2.2 Parametric Models

Black and Scholes (1973) make the assumptions that the price of the underlying asset follows geometric Brownian motion (5),that trading is continuous, and that assets can be bought or sold (short) at the same price, without transaction costs.

$$dS = \alpha S dt + \sigma S dZ, \tag{5}$$

where α is the continuous expected return, σ is the volatility of the return, and $Z(t)$ is a standard Brownian motion.

Let us consider the portfolio of the market maker under these conditions. The value of the portfolio is given by:

$$V = \Delta S - C \tag{6}$$

Where the arguments of the functions are suppressed for simplicity. Using Itô's formula:

$$dV = \Delta dS - (C_S dS + C_t dt + \frac{1}{2}C_{SS}(dS)^2) \tag{7}$$

Partial derivatives are denoted by subscript, e.g. $\frac{\partial C}{\partial S} = C_S$. The time until expiration decreases at the same rate that time passes , therefore $C_t = C_T \frac{dT}{dt} = -C_T$. Substituting for $\Delta = C_S$ and $(dS)^2 = (\alpha S dt + \sigma S dZ)^2 = \sigma^2 S^2 dt$:

$$dV = -C_t dt - \frac{1}{2}\sigma^2 S^2 C_{SS} dt \tag{8}$$

dV in (8) does not depend on dS or any stochastic process, it is therefore a risk-free position and should yield the risk-free rate r. If this were not the case, it would present an arbitrage opportunity. An arbitrage is often defined as an investment that requires no capital, has a positive probability of profit, and zero probability of loss (Shreve 2004). If the return on the portfolio, dV, was higher (lower) than the risk-free rate, an arbitrage could be made by borrowing (lending) at the risk-free rate and buying (selling) the portfolio. Unlike the simpler approximation, (8) accounts for time decay by including a time term, and larger movements in stock price by including a second order term with respect to asset price.

$$dV = -C_t dt - \frac{1}{2}\sigma^2 S^2 C_{SS} dt = rV dt \tag{9}$$

substituting for (6) and dividing by dt:

$$C_t + \frac{1}{2}\sigma^2 S^2 C_{SS} = r(C - C_s S) \tag{10}$$

Equation (10) is the Black-Scholes equation. The Black-Scholes equation can be written on the following form, by a change of variable (Black and Scholes 1973):

$$u_t = c^2 u_{xx} \tag{11}$$

Equation (11) is often called the heat equation (Kreyszig 2010)[2]. That is because it can be used to model how heat spreads throughout an object, where $u(t, x)$ is the heat at the point x at time t. The heat equation can be solved analytically, but this requires some additional conditions. First, the initial distribution of heat in the object, called the initial condition. In this case, that is really the terminal condition, and gives the value of the option at expiration, eq. (12). Second, we need to to know the dynamics of the temperature at the ends of the object. This is analogous to the value of the option at the extremes of underlying asset price. The price of the underlying asset cannot be smaller than 0, but can grow infinitely large. The boundary conditions are given by eq. (13).

Initial condition:

$$C(S, K, 0) = max\{S - K, 0\} \tag{12}$$

Boundary conditions:

$$C(0, K, T) = 0 \tag{13a}$$

$$\lim_{S \to \infty} C(S, K, T) \to S \tag{13b}$$

When the underlying asset price is 0 it cannot change according to (5), and so the call option is also worthless. In the case where the underlying asset is a stock, it means that the company is bankrupt. Eq. (13a) follows.

As the price of the underlying asset goes to infinity, the probability that the option will be exercised goes to 1, and the price of exercising, K, becomes

[2]It is also called the diffusion equation, hence why Black-Scholes is called a diffusion model.

negligible. Therefore, the price of the option will go towards the price of the asset. Eq. (13b) follows. Solving the Black-Scholes equation (10) subject to the conditions (12) and (13) gives the famous Black-Scholes formula (14).

$$C(S, K, T, r, \sigma) = SN(d_1) + Ke^{-rT}N(d_2), \tag{14}$$

where $N()$ is the cumulative probability density function of the standard normal distribution and

$$d_1 = \frac{\ln(\frac{S}{Ke^{-rT}}) + \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}},$$
$$d_2 = d_1 - \sigma\sqrt{T}$$

2.3 Neural Networks

An artificial neural network is a non-linear regression method loosely based on its biological counterpart. They are often visualized as in Figure 1. Continuing the biological analogy, the circles represent "neurons" and the lines represent "synapses", the connections between neurons. The brain learns by strengthening or weakening these connections. In the artificial neural network these connections are represented by weights, which can be increased or decreased in order for the output of the network to approach the desired value. Depending on the strength of the input signals, the neurons are activated to varying degrees, which in turn will affect the activation of the neurons in the following layer. In the artificial neural networks this is represented by applying some non-linear function to the weighted inputs of the neuron. The so-called "activation function" is often a sigmoid-shape, like the logistic or hyperbolic tangent functions (Fig. 2).

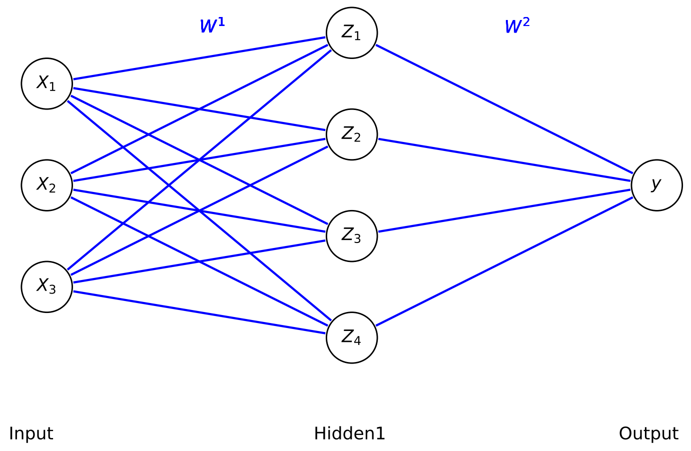


Figure 1: Artificial Neural Network Architecture, $Z = f(XW^1)$ and $y = ZW^2$

Linear regression models are more familiar to many, and can also be visualized as an artificial neural network (see Fig.3).

The parameters (weights) of the artificial neural network are chosen to optimize the value of some objective function. Often there are some values y, that one wants the outputs of the neural network $\hat{y}$ to match. This is called *supervised learning* because it requires supervising the learning process of the ANN by showing it the values y it should map the inputs to. A popular objective function in this case is the mean square error (MSE), $\frac{1}{n}\sum(y-\hat{y})^2$.

In some cases the ideal output values are unknown. In such cases, another way of evaluating the neural networks output can be used as the objective function. For example, if you are training an artificial neural network to play a video game, you don't necessarily know the ideal "move" in each situation. But better moves will improve the game score more than bad moves. In this case the network could learn by maximizing the game score. This approach is called *reinforcement learning*.

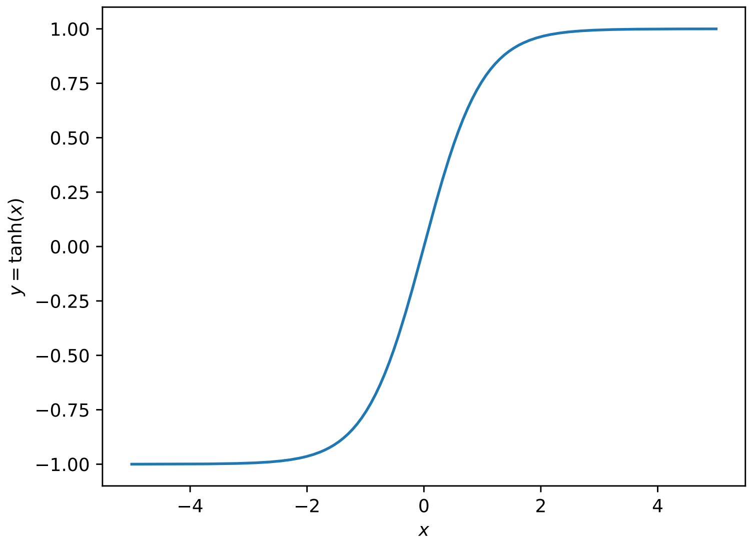


Figure 2: The Hyperbolic Tangent Function

Artificial neural networks are well suited for estimating option price functions because they excel at approximating non-linear functions. Even a neural network with a single hidden layer can approximate a continuous function arbitrarily closely given enough neurons (Hornik, Stinchcombe, and White 1989). This is also true of the derivatives of the function (Hornik, Stinchcombe, and White 1990). Which is particularly useful for option pricing given the importance of the derivatives (the so-called ”Greeks”) to hedging strategies.

2.4 Earlier Option Pricing Neural Networks

Pricing options using artificial neural networks is not a novel endeavor. Hutchinson, Lo, and Poggio (1994) is the seminal paper on the subject, and an article using a similar approach was published even earlier (Malliaris and Salchenberger 1993). Some even use other regression models like ordinary

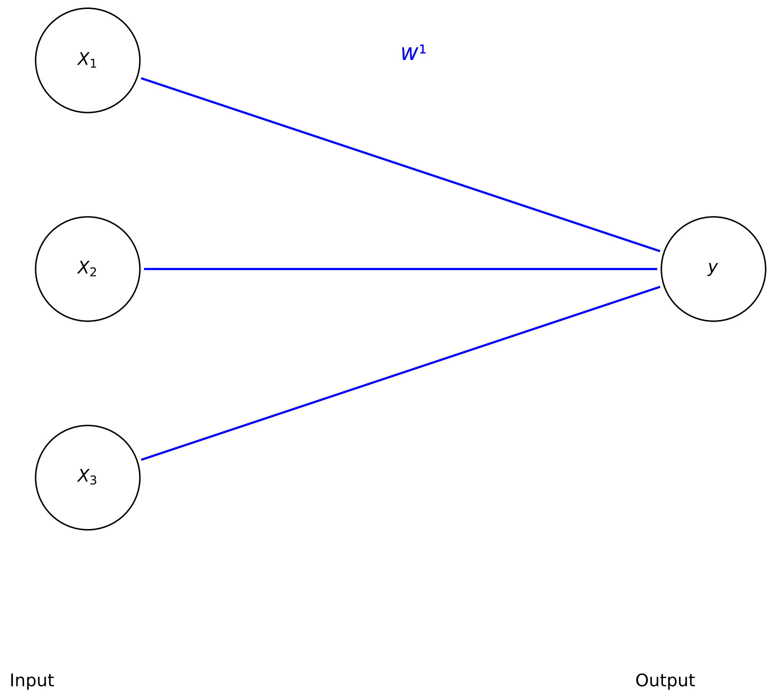


Figure 3: Linear Regression Architecture, $y = XW^1$

least squares (Hutchinson, Lo, and Poggio 1994) or support vector machines (Liang et al. 2009). Many of the non-parametric models perform well, often outperforming parametric methods. Park, Kim, and Lee (2014) provide an overview of this literature.

The established non-parametric models all have one thing in common, they are fitted using supervised learning. As many studies have shown, this can work very well, but there are some disadvantages to this approach. First, to estimate a good model, a large amount of data is required. This makes the method only useful for assets/markets where there is a history of an active/liquid option market. Second, if the historical market prices have biases, the model is likely to learn them as well.

In section 5, I will compare the performance of the Black-Scholes model and an ANN trained using supervised learning, with my new method, an ANN trained using reinforcement learning.

3 The Model

In this section I present the reinforcement ANN.

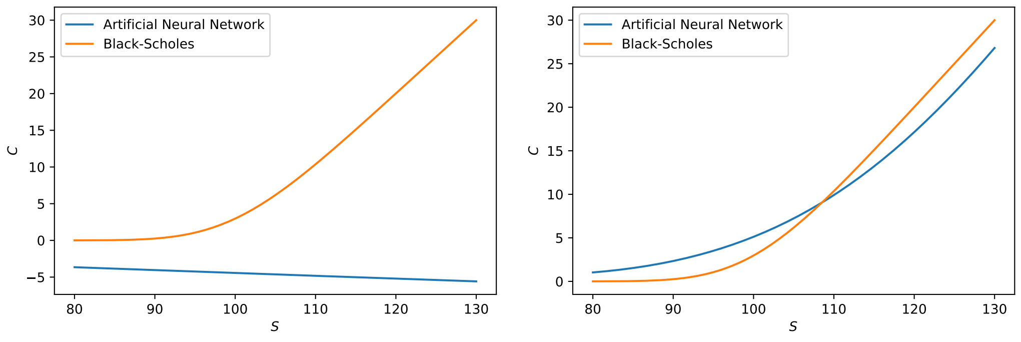
3.1 Architecture and Functional Form

An artificial neural network is capable of learning complex non-linear functions, but as the functions become more complex more data is required to learn. Therefore, if there is an opportunity to reduce the dimensionality of the function without losing information it should be seized.

The goal of the model is estimating an option pricing function $C(S, K, T)$. Merton (1973) showed under very weak assumptions that C is homogeneous of degree one in S and K. Meaning that $C(\frac{S}{K}, 1, T) = \frac{C(S,K,T)}{K}$. $\frac{S}{K}$ is a measure of the options moneyness, and using this as an input feature instead of S and K is often called the "homogeneity hint" (Bennell and Sutcliffe 2004). The neural network will therefore be using $\frac{S}{K}$ and T as input features and will output $\frac{\hat{C}}{K}$[3]. $\hat{C}$ can, of course, be easily recovered by multiplying the output by K.

Merton (1973) also conjectures that the two arguments interest rate, r, and exercise price, K, can be replaced with a single argument, the present value of exercise price, $PV(K) = e^{-rT}K$. This is certainly the case with the Black-Scholes formula (14). In this case, the interest r does not affect the stochastic process generating the underlying asset price. Using this, the interest rate can be set to $r = 0$, and ignored during model training. When the model is used later the present value of the exercise price , $e^{-rT}K$, is used instead of K.

[3]Let a parameter with a hat, $\hat{}$, denote the estimate of that parameter.



(a) Randomly initialized parameters (b) After 15000 iterations

Figure 4: Price for call option with $K = 100$ and $T = 90$ as calculated by ANN and BS during training, see Fig. 6 for finished training

3.2 Training

The main contribution of this text is the method of training the model. The non-parametric models presented in previous works are all taught using supervised learning. The models are shown the correct option price corresponding to the inputs, over and over, until they learn the relationship [(Hutchinson, Lo, and Poggio 1994), (Anders, Korn, and Schmitt 1998), (Galindo-Flores 2000), (Amilon 2003), (Bennell and Sutcliffe 2004), (Hamid and Habib 2005), (Park, Kim, and Lee 2014), (Montesdeoca and Niranjan 2016)]. My model, on the other hand, is not given information about the correct option price. It must teach itself, using only fundamental properties of derivative securities. Namely, the relationship between the price of the derivative security and the underlying asset. As you will see in this section, the training process is closely related to Black and Scholes (1973) derivation of their formula. In their derivation, Black and Scholes solve the Black-Scholes equation subject to some additional conditions. In the reinforcement ANN method, an objective function, analogous to the Black-Scholes equation, is minimized subject to the same initial condition as used by Black and Scholes.

The objective function needs to evaluate the performance of the model's pricing, without knowing the correct option prices. To do this I will view the model from a delta-hedging market maker's perspective. By buying $\Delta = \frac{\partial C}{\partial S}$ of the underlying asset the market makers' positions should be hedged. If this is done at time t, the change in value of the portfolio at time $t+h$ is $\Delta(S_{t+h} - S_t) - (C_{t+h} - C_t)$. The parameters of the ANN are therefore chosen to minimize the objective function (15), where $\hat{C}_t$, $\hat{C}_{t+h}$, and $\hat{\Delta}$ are all calculated by the artificial neural network. In other words the neural network is trained to minimize the squared tracking error of the hedging portfolio, (or the volatility of the market maker's portfolio). At first the reinforcement ANN's estimates $\hat{C}_t$, $\hat{C}_{t+h}$, and $\hat{\Delta}$ will be completely incorrect, and determined by the random parameter initialization, but they will improve gradually during training (see Fig. 4).

$$[\hat{\Delta}(S_{t+h} - S_t) - (\hat{C}_{t+h} - \hat{C}_t)]^2, \tag{15}$$

where $\hat{\Delta} = \frac{\partial \hat{C}}{\partial S}$. By itself, the loss function (15) "incentivizes" the model to predict $\hat{C}$ as a constant. That way the first term of (15) will be zero because $\hat{\Delta} = \frac{\partial \hat{C}}{\partial S} = 0$ and the second term will be zero because $\hat{C}_{t+h} = \hat{C}_t = constant$. This can be solved by imposing an initial condition, as discussed in section 2.2. At its expiration date the option is worth $C(S, K, 0) = \max(S - K, 0)$. This condition is enforced on the model by bypassing the ANN, and returning $\max(S - K, 0)$ if $T = 0$. The condition does not directly affect the predicted option prices at $T > 0$, but does so indirectly through the loss function.

$$\text{The model returns} \begin{cases} \hat{C} \text{ as predicted by the ANN,} & T > 0 \\ \max(S - K, 0), & T = 0 \end{cases} \tag{16}$$

Before the training procedures are initiated, the parameters of the model are randomly initialized. The training procedures for the model work as follows: a day from the sample is picked randomly, along with a random strike price K and time to maturity T. This can be thought of as the characteristics of an invented theoretical option. The model then estimates the value of the specified option on that day, $\hat{C}_t$, and on the folowing trading day, $\hat{C}_{t+h}$, along with the delta of the model on the first day, $\hat{\Delta}$. This is sufficient information to estimate the change in value of the hedging portfolio, and option position. This is done on a batch of $10,000$ options at a time. The objective function (15) is calculated for all the theoretical options, and the mean is taken. The derivatives of the loss function is calculated, and the parameters are updated to minimize the loss function using backpropagation (Rumelhart, Hinton, and Williams 1986). This procedure is repeated until the loss function is no longer decreasing. In this case, that takes around $500,000$ iterations. This means that during training, the value of roughly 5 trillion theoretical options are calculated at two separate points in time.

3.3 Applications

The focus in this text is on European call option pricing with a neural network, but the training procedure is not limited to that special case. The same training procedure could be applied with other types of non-linear regression models, e.g. linear regression with higher order terms or a support vector machine. ANNs are just a good choice because of their scalability. Using backpropagation, a neural network with hundreds of thousands of parameters can be estimated fairly quickly and easily on a home computer.

More interestingly, the reinforcement ANN method is not limited to European call options. By changing the initial condition (12), the method can be used to price any contingent claim that is not path dependent, and has a fixed expiration or maturity date. Examples of this include European puts, digital options (all-or-nothing), gap options, a derivative that pays the square of the underlying asset price on exercise, and so on. Section 4.3 provides an example of this, by pricing a cash-or-nothing put option.

3.4 Limitations and Assumptions

Assumptions

The basis for the reinforcement ANN and the Black-Scholes model are the same, the no-arbitrage argument. Therefore the necessary assumptions for both models are also similar. Following Merton (1973), I will go through the assumptions of the Black-Scholes model and discuss the differences and commonalities.

1. ***Frictionless markets.*** **There are no transaction costs, buying and selling is done at the same price. This also applies to the credit market, meaning equal lending and borrowing rate. Trading occurs continuously.**

   The first part applies to the reinforcement ANN as well as the Black-Scholes model. Continuous trading only occurs in the Black-Scholes model. I will return to the implications of this below.

2. ***Stock price dynamics.*** **The price of the underlying asset is generated by geometric Brownian motion.**

   Relaxing this assumption is one of the main benefits of the reinforcement ANN. The equivalent assumption for the reinforcement ANN, is that the generating process in the training sample must be representative of the generating process going forward. This means that if there is a regime shift, e.g. significantly increased volatility, after the training, the model will need to be re-estimated. This assumption may be relaxed further by including parameter(s) describing the changing aspects of the price dynamics, e.g. volatility.

3. ***Investor preferences and expectations.*** **No assumption on investor preferences are required by the Black-Scholes model, the market actors only need to agree on σ in the geometric Brownian motion (eq. 5).**

   This is not strictly the case for the reinforcement ANN. The Black-Scholes model is independent of investor preferences because a delta-hedged position is completely risk free. This is possible because the price generating process of the underlying asset is continuous (assumption 2) and the portfolio can be rebalanced continuously (assumption 1). This is not the case for the reinforcement ANN. A delta hedged market maker assumes some risk between opportunities to rebalance their portfolio. This is similar to the jump-diffusion model proposed by Merton (1976). The market maker assumes risk, but Merton (1976) argues that this is non-systemic risk, and can be diversified away. Thus Merton avoids the issue of investor preferences. In the reinforcement ANN, investor preferences are reflected in the loss function.

Why is the loss function squared? Does this choice assume that larger deviations are more important to market makers? Using the square of the errors, effectively give greater weight to larger deviations. In this case the reasoning behind the choice is more practical than theoretical. As opposed to other choices, like absolute deviation, the derivatives of the function are continuous. This is particularly important, given the techniques used for parameter estimation. The square errors are also quick and simple to calculate. Perhaps for these very reasons, squared errors are almost considered the default option for model estimation.

First Order Approximation

The basis for the loss function (15) is that the change in the hedging portfolio $\hat{\Delta}(S_{t+h} - S_t)$ should be as close as possible to the change in the estimated option value $\hat{C}_{t+h} - \hat{C}_t$ over a given time interval. As discussed in section 2.1, this is a simplification. The hedging portfolio will only change in value when the underlying asset price changes, while the option price also changes as a result of time passing. Another problem is that the option price is not a linear function of underlying asset price. The proportional change in option value depends on the direction and magnitude of the price change in the underlying asset. Not accounting for these effects introduces a bias in the model. In the Black-Scholes model, these issues are accounted for by including a time term, and a second order term for the underlying asset price. Could this be done for the reinforcement ANN as well? The loss function could be:

$$\left[\left(\hat{\Delta}(S_{t+h} - S_t) + \frac{1}{2}\hat{\Gamma}(S_{t+h} - S_t)^2 + \hat{\Theta}h\right) - (\hat{C}_{t+h} - \hat{C}_t)\right]^2, \qquad (17)$$

where $\hat{\Gamma} = \frac{\partial^2 \hat{C}}{\partial S^2}$ and $\hat{\Theta} = \frac{\partial \hat{C}}{\partial T}$.

This is not as simple to implement as it might seem. The problem is that by adding two more unknowns, Γ and Θ, without introducing more information, the system becomes underdetermined. There are fewer equations than unknowns. The neural network is unable to work out what accounts for the change in option value over a time interval. Is it time-decay or a change in the underlying asset price? The Black-Scholes model solves this issue by imposing the two boundary conditions, (13). However, that is not straight forward to do in an empirical model, such as the reinforcement ANN. The underlying asset price will not usually reach its boundaries in the training sample, so how can these conditions be imposed? Finding answers to these questions could likely greatly improve the performance of the reinforcement ANN. Sadly, they will not be resolved in this text.

4 Testing on Synthetic Data

In this section I apply the model on simulated data. This provides a good proof-of-concept before I apply it on real market data. I attempt to showcase the flexibility of the method by pricing both a standard European call option, and a more exotic cash-or-nothing put option.

The models are implemented in Python, using a number of scientific libraries; including TensorFlow, numpy, pandas, and matplotlib[4]. A simplified version of the code for the reinforcement ANN is included in Appendix A.

4.1 Simulating Black-Scholes Data

Before applying the model on real market data, it is useful to test it in an environment where the correct option prices are known. This can be accomplished by training the model on data simulated according to the assumptions made by the Black-Scholes model. This way, the Black-Scholes formula will give the correct price.

The Black-Scholes model assumes that the underlying asset price follows geometric Brownian motion as in (18).

$$dS(t) = \alpha S(t)dt + \sigma S(t)dZ(t) \tag{18}$$

where α is the continuous expected return, σ is the volatility of the return, and $Z(t)$ is a standard Brownian motion.

By taking the integral on both sides we get:

$$S(t) = S(0) + \alpha \int_0^t S(u)du + \sigma \int_0^t S(u)dZ(u) \tag{19}$$

[4]All figures in this text were generated by matplotlib.

where $\int_0^t S(u)du$ is a Lebesgue integral and $\int_0^t S(u)dZ(u)$ is an Itô integral. A solution is

$$S(t) = S(0)e^{(\alpha - \frac{1}{2}\sigma)t + \sigma Z(t)} \tag{20}$$

This can be easily verified by applying Itô's formula. Using $f(t,x) = S(0)e^{(\alpha - \frac{1}{2}\sigma)t + \sigma x}$ we have

$$dS(t) = df(t, Z(t)) = f_t(t, Z(t))dt + f_x(t, Z(t))dZ(t) + \frac{1}{2}f_{xx}(t, Z(t))dt$$

$$dS(t) = (\alpha - \frac{1}{2}\sigma^2)S(t)dt + \sigma S(t)dZ(t) + \frac{1}{2}\sigma^2 S(t)dt$$

$$dS(t) = \alpha S(t)dt + \sigma S(t)dZ(t)$$

which is the same geometric Brownian motion as in (18).

This process can therefore be simulated at the discrete points $t_0, ..., t_N$ by (21).

$$S_{t_n} = S_{t_0}e^{(\alpha - \frac{1}{2}\sigma^2)(t_n - t_0) + \sigma \sum_{i=1}^{n}(\sqrt{(t_i - t_{i-1})}Z_i)}, \tag{21}$$

where $Z_1, ..., Z_N \overset{iid}{\sim} N(0,1)$.

Using this method, I simulate two years of daily prices, assuming 250 trading days per year (Fig. 5). I use $S_0 = 100$, $\alpha = 0.05$, and $\sigma = 0.15$ as parameters for the simulation.

The model is then trained on the data as described in section 3.2.

4.2 Results

Figure 6 and 7 show the call option price and delta, respectively, 90 days before expiration as calculated by the neural network and Black-Scholes model.

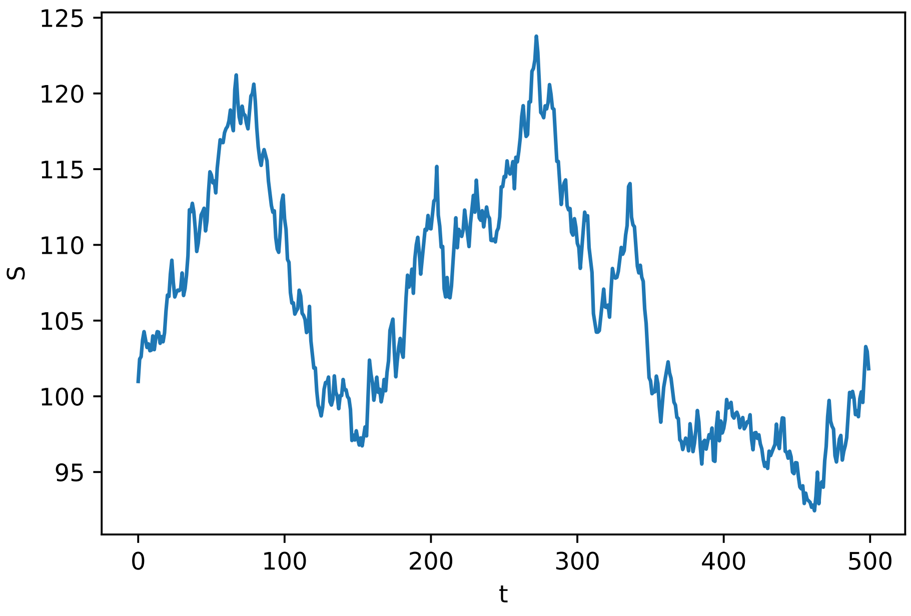


Figure 5: Asset price over 500 trading days

Figure 9a and 9b show the same, 10 days before expiration. As can be seen in the figures, the reinforcement ANN was able to teach itself the value of the options to a high degree of accuracy. Figure 8 show that even the "Greeks" Γ and Θ, the second order derivative with respect to asset price and the derivative with respect to time respectively, are also estimated accurately.

As one would expect, the results are not affected by reasonable choices of α. Different choices of σ also affects the results as on would expect, in line with the Black-Scholes model.

The results are good, but not perfect. As can be seen in figure 7, the reinforcement ANN estimates higher Δ out-of-the-money, and lower Δ in-the-money than is correct. This leads to a slight overvaluation of the option close to the money, as is evident in figure 6. This is not a result of imperfect approximation by the ANN, but bias associated with not accounting for time-

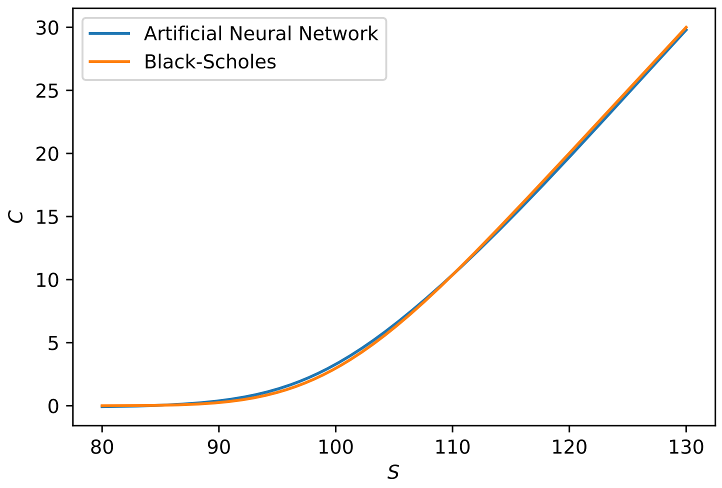


Figure 6: Plot of ANN and BS values for call option with $K = 100$ and $T = 90$

decay and higher order changes in underlying asset price. In this particular setting with Black-Scholes data, I can actually account for these effects in the loss function. Because I know the Black-Scholes formula to be correct in this setting, I can use the Black-Scholes estimates of Θ and Γ in the modified loss function:

$$\left[\left(\hat{\Delta}(S_{t+h} - S_t) + \frac{1}{2}\hat{\Gamma}_{BS}(S_{t+h} - S_t)^2 + \hat{\Theta}_{BS}h\right) - (\hat{C}_{t+h} - \hat{C}_t)\right]^2 \tag{22}$$

When using this loss function, the bias disappears from the model. Figure 10 shows the price and delta plots of the reinforcement ANN estimated with modified loss function. Of course, in other contexts, using the Black-Scholes estimates of Θ and Γ is inappropriate. Relying on Black-Scholes estimates for estimating the reinforcement ANN also defeats the purpose of the model.

Even with the bias, this test of the model in a controlled environment provides a great proof-of-concept. In section 5 the model is applied on real market data of the S&P 500 index.

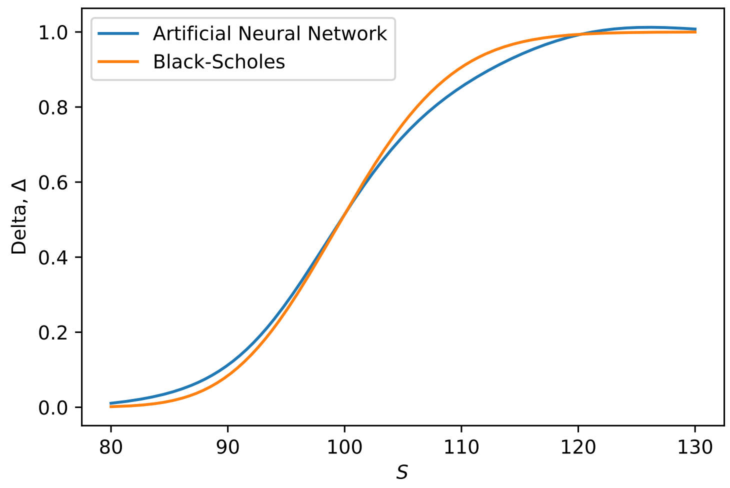


Figure 7: Plot of ANN and BS delta values for call option with $K = 100$ and $T = 90$

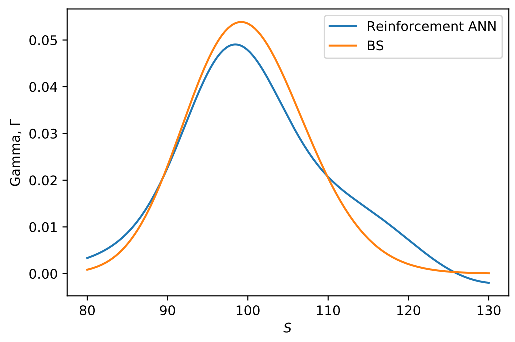
4.3 Pricing a Cash-or-Nothing Put Option

In this short aside, I apply the reinforcement ANN method on a cash-or-nothing put option, using the same synthetic data as above. The experiment highlights the flexibility of the reinforcement learning method. By changing one line in the code, the initial condition, an entirely different type of derivative security can be priced effectively.

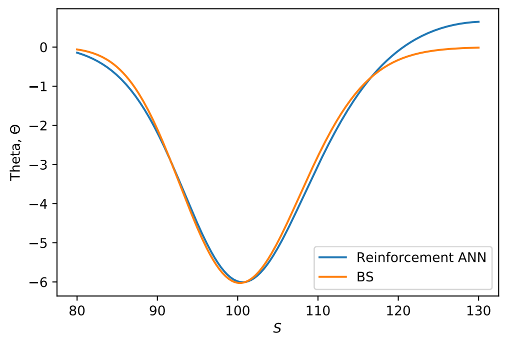
A cash-or-nothing put option is a financial derivative that pays \$1 if the underlying asset price S is lower than the strike price K. That is if $S < K$. Solving the Black-Scholes equation for this security shows that its value is given by (23) (McDonald, Cassano, and Fahlenbrach 2006).

$$e^{-rT} N(-d_2) \tag{23}$$

Figure 11 shows the results. The possible applications of the reinforcement ANN is far from limited to European call options.

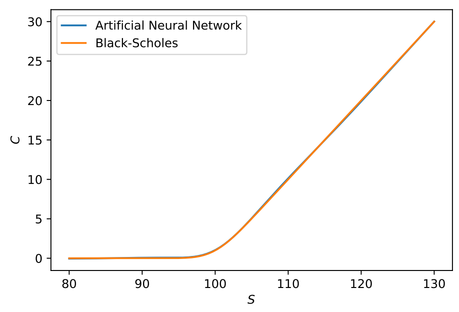


(a) $\frac{\partial^2 C}{\partial S^2} = C_{SS} = \Gamma$

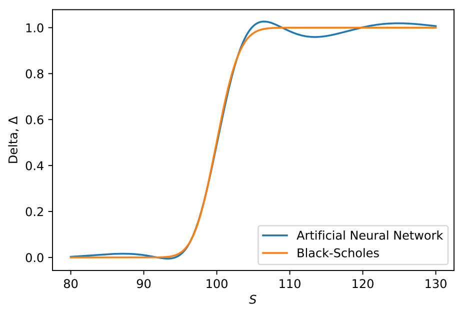


(b) $\frac{\partial C}{\partial t} = C_t = \Theta$

Figure 8: Derivatives of option price with $K = 100$ and $T = 90$ as calculated by ANN and BS

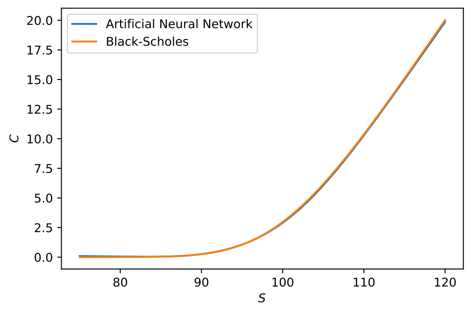


(a) Option price as a function of S

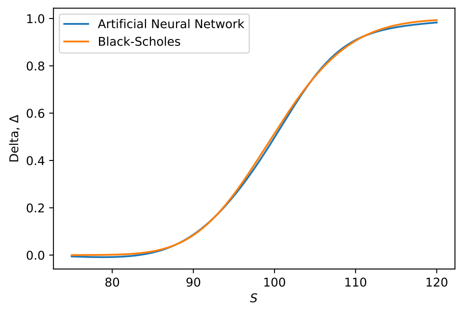


(b) Option delta as a function of S

Figure 9: Price and delta for call option with $K = 100$ and $T = 10$ as calculated by ANN and BS

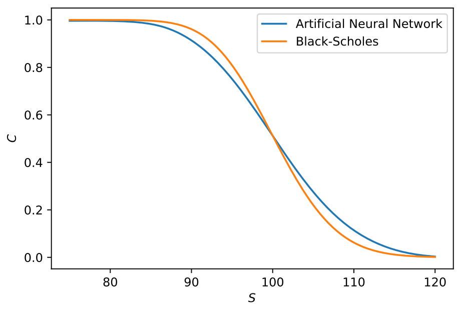


(a) Option price as a function of S

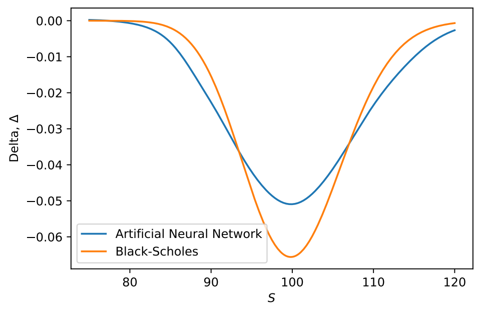


(b) Option delta as a function of S

Figure 10: Price and delta for call option with $K = 100$ and $T = 90$ as calculated by ANN with modified loss function, and BS



(a) Option price as a function of S



(b) Option delta as a function of S

Figure 11: Price and delta for cash-or-nothing put option with $K = 100$ and $T = 60$ as calculated by ANN and BS

5 Applied on the S&P500 Index

In this section, I apply the model on real market data, and compare its out-of-sample performance to two rival models. The rival models are the Black-Scholes model and an artificial neural network trained using supervised learning.

5.1 The Data

I have chosen to use call options on the Standard and Poor's 500 (S&P500) index traded on the Chicago Board Options Exchange (CBOE). There are a few reasons for this. First, the S&P500 is the most widely used stock market index, and so options on it is thickly traded. Second, the options are European style (Chicago Board Options Exchange 2018). Third, the index does not pay dividends, simplifying the analysis.

I use three years of daily data, from 2014 up to and including 2016. There are a total of $456,936$ observations on call options with non-zero daily trading volume in this period. The dataset was purchased from www.discountoptiondata.com, but originates from the Options Price Reporting Authority (Discount Option-Data 2018). I use the 3-month treasury bill rate (DTB3) as interest rate. A time series with daily observations is obtained from the Federal Reserve Economic Data (FRED 2018).

One challenge with this dateset is that it does not track when the last option trade was made. This means that $LastPrice$, the closing price of the option, may be from a different point in the day than the recorded closing price of the index, $UnderlyingPrice$. If the index price moved significantly since the latest transaction of a particular option was traded, the recorded option price may no longer be current. To combat this I drop all observations

DataDate	Expiration	AskPrice	BidPrice	LastPrice	StrikePrice	Volume	UnderlyingPrice
2015-06-05	2015-06-26	4.70	4.30	4.40	2140.0	2279	2093.29
2015-07-01	2015-07-31	13.60	12.90	13.00	2115.0	233	2076.74
2016-08-12	2018-12-21	206.70	199.30	204.00	2175.0	600	2183.97
2015-03-03	2017-12-15	1.60	1.10	1.15	3500.0	1	2107.78
2015-02-12	2015-03-27	1.35	0.85	1.05	2215.0	10	2088.48

Table 1: Selected columns from www.discountoptiondata.com dataset sample

of options where less than 100 contracts where traded, this is 67% of the original data set. I also drop all observations where the closing price is larger than the ask price or smaller than the bid price, this is 50% of the original data set.

There are relatively few observations that have more than 120 days until expiration, are very deep in-the-money ($\frac{S}{K} > 1.2$), or very far out-of-the-money ($\frac{S}{K} < 0.8$). The lack of data makes it difficult for the supervised ANN to learn the price of such contracts. The reinforcement ANN and Black-Scholes models are not affected by the lack of data, but I exclude observations outside this range in an attempt to not disadvantage the supervised ANN. This excludes 12% of the remaining data. In total, $74,919$ observations remain for the three years. Table 2 shows a sample of the data set after processing.

The dataset is split into three parts. Two years for training, two months for validation, and 10 months for testing. The split is illustrated in figure 12. The training data is used to estimate the model parameters. The validation data is used for model selection. Finally, model performance is evaluated on the test data.

Date	S	K	T	C	S_-	T_-
2016-02-03	1911.08	2015.0	44.0	8.05	1915.64	43.0
2015-10-23	2075.70	2085.0	14.0	13.47	2071.17	11.0
2015-11-03	2109.53	2150.0	38.0	10.80	2099.67	37.0
2014-04-28	1869.43	1950.0	19.0	0.40	1878.33	18.0
2015-09-04	1927.97	2130.0	42.0	1.07	1968.96	38.0

Table 2: Selected columns from processed dataset sample

5.2 Model Selection

There are a number of consequential choices to be made when using artificial neural networks. Number of layers, number of neurons in each layer, batch size, learning rate, choice of activation function, input variables and so on. These are often called hyperparameters. In a sense, the validation data is used to estimate optimal hyperparameters. There are an infinite number of permutations for the hyperparameters, so an exhaustive search is clearly impossible.

Both the supervised and the reinforcement learning models will have the same input and output variables. $\frac{S}{Ke^{-rT}}$ and T are the inputs, and C is output. The reinforcement ANN will be trained as described in section 3.2, using the daily closing price of the S&P 500 index. The supervised ANN is estimated using the observed market option prices, minimizing the observed MSE (25).

The search space can be limited by rules of thumb and existing literature as guides. To further narrow down the selection, I have used a combination of intuition guided experiments and stochastic grid search.

Using these methods, I find that a neural network with two hidden layers

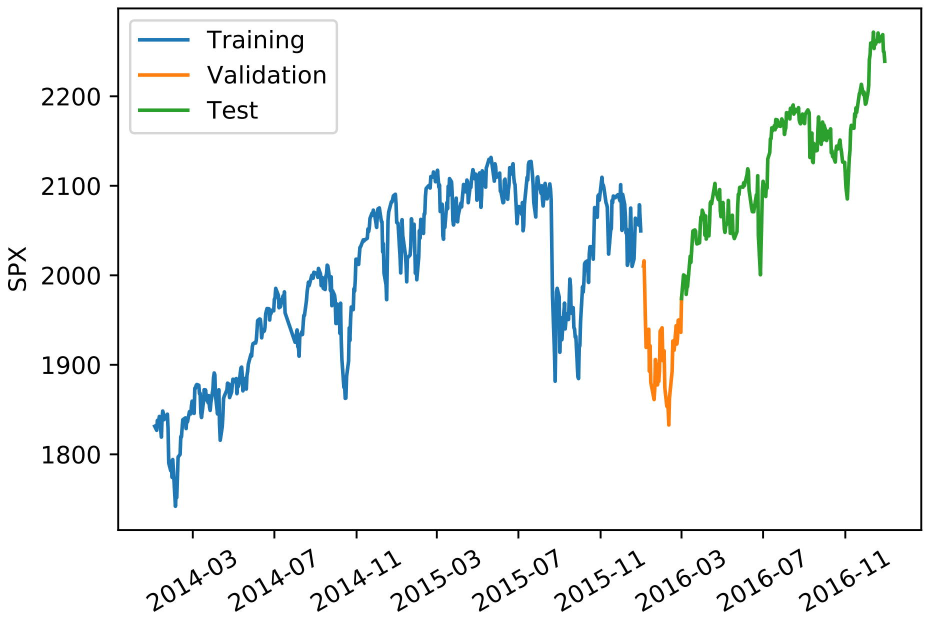


Figure 12: SPX Index and Split

of 50 neurons each is close to ideal for the supervised learning ANN. This means that there are a total of $2,752$ parameters to be estimated in the model.

For the reinforcement learning ANN, three hidden layers, also with 50 neurons each, performs best on the validation set. This equates to a total of $5,302$ estimated parameters.

Why is the ideal size of the reinforcement network greater than that of the supervised network? This disparity highlights one of the main benefits of the reinforcement learning approach. Generally, a neural networks ability to accurately approximate a complex function increases with its size, but as the size and number of parameters increase, the amount of data needed to fit the parameters also increase. Too many parameters in relation to the amount of data available will result in overfitting. The limited amount of market data limits the size of supervised network, while the reinforcement ANN can be fitted using an infinite number of 'theoretical' option contracts. Of course, the 'theoretical' option contracts are not completely independent of each other. Similar 'theoretical' options will provide similar returns over the same time intervals. So the reinforcement ANN is also limited by the amount of data, but much less so than the supervised ANN. This effect would be much more pronounced in other more thinly traded markets. The amount of option market data on the Norwegian stock market index OBX, for example, will be orders of magnitude less than on the S&P 500 (Oslo Børs 2018).

5.3 Performance Measures

There are two main approaches for evaluating the performance of an option pricing model. If the the true value of the option is known, the simplest approach is to use some function of the difference between the models predicted price and the true option value. Arguably, the most popular example of this is the mean squared error (MSE) or the squareroot of the MSE (RMSE). For example Bennell and Sutcliffe (2004), Amilon (2003), Park, Kim, and Lee (2014), and Galindo-Flores (2000) all use this measure. Other examples of this type of measure include the squared correlation coefficient (R^2) (used by Hutchinson, Lo, and Poggio (1994) and Bennell and Sutcliffe (2004)), and mean absolute deviation (MAD) (used by Bennell and Sutcliffe (2004)).

The second approach does not require knowing the correct option price. The measure is based on some function of the difference between the evolution of the option price and a replicating (hedging) portfolio based on the model. Either the observed option price, or the model estimate of the option price can be used in comparison to the replicating portfolio. Hutchinson, Lo, and Poggio (1994) uses the observed option price, but Amilon (2003) argues that this cannot be good practice if there is doubt that the observed price is the true value. Either way, this method requires a time series of the underlying asset price. The objective function of the reinforcement ANN is such a function, the MSE.

I will use two different measures from each approach, the mean square error (MSE) and the mean absolute deviation (MAD).

The mean absolute deviation between the observed price C and the model predicted price $\hat{C}$.

$$\text{Observed MAD} = \frac{1}{n}\sum_{i=1}^{n} |\hat{C} - C| \tag{24}$$

The mean square error between the observed price C and the model predicted price $\hat{C}$.

$$\text{Observed MSE} = \frac{1}{n}\sum_{i=1}^{n} (\hat{C} - C)^2 \tag{25}$$

The mean absolute deviation between the price change in the option $\hat{C}_{t+h} - \hat{C}_t)$ and the hedging portfolio $\hat{\Delta}(S_{t+h} - S_t)$.

$$\text{Hedging MAD} = \frac{1}{n}\sum_{i=1}^{n} |\hat{\Delta}(S_{t+h} - S_t) - (\hat{C}_{t+h} - \hat{C}_t)| \tag{26}$$

The mean square error between the price change in the option $\hat{C}_{t+h} - \hat{C}_t)$ and the hedging portfolio $\hat{\Delta}(S_{t+h} - S_t)$.

$$\text{Hedging MSE} = \frac{1}{n}\sum_{i=1}^{n} \left[\hat{\Delta}(S_{t+h} - S_t) - (\hat{C}_{t+h} - \hat{C}_t)\right]^2 \tag{27}$$

5.4 Experimental Design

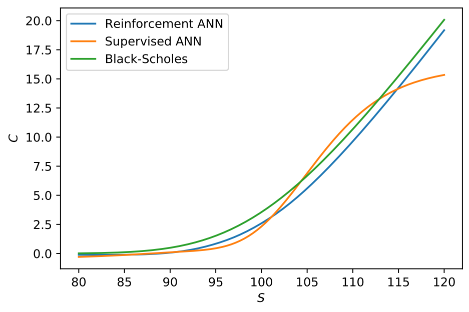
First the competing models are fitted on the training set. Because the model parameters are randomly initialized, otherwise identical models will be slightly different after training. To mitigate this, each model is trained ten times with different initial parameters. The one that performs best on the validation sample is chosen for further analysis. Bennell and Sutcliffe (2004) use a similar process. The choice depends on which performance measure is used, I use hedging MAD. Table 3 shows the performance of the ten training runs on the validation sample.

Model	Hedging MSE	Hedging MAD	Observed MSE	Observed MAD
Reinforcement ANN 0	2.22238	0.738734	20.9957	2.93025
Reinforcement ANN 1	2.10409	0.725239	23.7122	3.30179
Reinforcement ANN 2	2.14614	0.749549	20.355	2.69666
Reinforcement ANN 3	2.26206	0.724356	25.6384	3.0741
Reinforcement ANN 4	2.52527	0.79524	35.1993	3.97151
Reinforcement ANN 5	2.36396	0.738884	25.0172	3.02967
Reinforcement ANN 6	2.2757	0.747626	19.1331	2.67247
Reinforcement ANN 7	2.13811	0.725381	28.0196	3.31719
Reinforcement ANN 8	2.33193	0.744567	40.6807	5.52289
Reinforcement ANN 9	2.16678	0.754914	18.7411	2.76621
Black-Scholes	2.70696	0.856054	21.3	3.28285

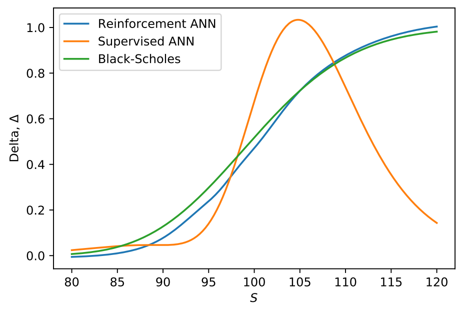
Table 3: Performance of the reinforcement ANN on the validation set over ten training runs

5.5 Results

In Table 4 we see that the reinforcement ANN outperforms the Black-Scholes model on every single performance metric. The supervised ANN outperforms both the reinforcement ANN and the Black-Scholes model on every metric except Hedging MSE. The supervised ANN have the smallest hedging error on average, but some of the errors are relatively large. Examining figure 13 and 14 can give an idea why. The three models make relatively close



(a) Option price as a function of S



(b) Option delta as a function of S

Figure 13: Price and delta for call option with $K = 100$ and $T = 90$ as calculated by supervised ANN, reinforcement ANN, and BS

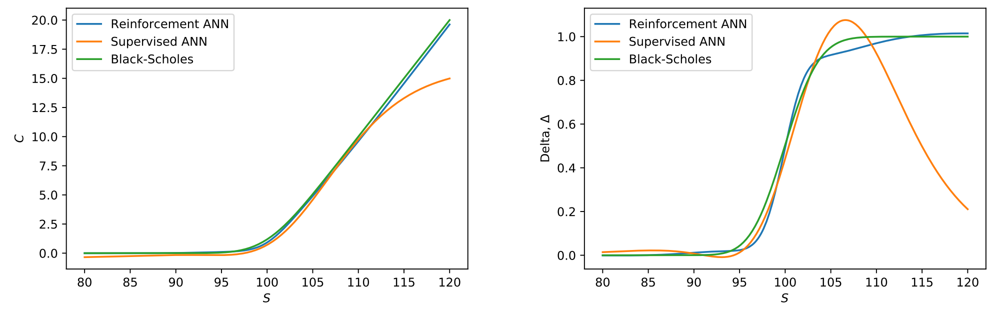
predictions, particularly for the short maturity options, until the options are well in-the-money.

Figure 15 show that there is less observations with long maturities and moneyness far from unity. This can explain why the supervised ANN, that depends on large amounts of data to learn, performs worse on these options. The Black-Scholes model and the reinforcement ANN on the other hand, are not reliant on this data and perform more consistently.

Model	Hedging MSE	Hedging MAD	Observed MSE	Observed MAD
Reinforcement ANN (mine)	2.65657	0.858184	54.9678	6.25654
Supervised ANN	3.57705	0.775563	19.3856	3.12608
Black-Scholes	3.40099	1.139	220.997	12.0978

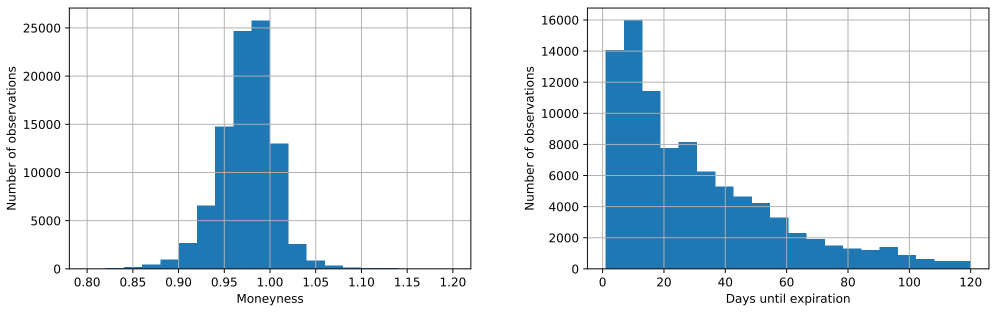
Table 4: Performance of the three models on the test set

Splitting the test sample by month reveals the same pattern, see table 5. The reinforcement ANN outperform the Black-Scholes model on every metric every month. And almost every month, the supervised ANN outperform the



(a) Option price as a function of S (b) Option delta as a function of S

Figure 14: Price and delta for call option with $K = 100$ and $T = 10$ as calculated by supervised ANN, reinforcement ANN, and BS



(a) Moneyness (b) Days until expiration

Figure 15: Histograms of 'Moneyness' and 'Days until expiration' for the S & P 500 index options

Month	Model	Hedging MAE	Hedging MSE	Observed MAE	Observed MSE
Mar-2016	Black-Scholes	121.904182	1.942933	8.764583	0.895147
	Reinforcement ANN	27.639412	1.890050	4.380633	0.700358
	Supervised ANN	22.454878	3.099291	3.481279	0.636125
Apr-2016	Black-Scholes	167.847626	2.001734	10.659591	0.976048
	Reinforcement ANN	43.667641	1.760359	5.662245	0.805555
	Supervised ANN	12.191474	2.481467	2.442907	0.704860
May-2016	Black-Scholes	149.477371	2.548478	10.241510	0.946158
	Reinforcement ANN	40.115330	1.832577	5.529256	0.731027
	Supervised ANN	10.822154	1.727955	2.462537	0.596977
Jun-2016	Black-Scholes	118.036026	6.013115	8.305344	1.180875
	Reinforcement ANN	33.083168	5.082230	4.633369	1.051576
	Supervised ANN	29.696371	6.373221	3.936042	1.092728
Jul-2016	Black-Scholes	248.456375	2.800205	13.038374	1.168717
	Reinforcement ANN	72.138412	2.093725	7.328163	0.870965
	Supervised ANN	18.358173	2.950025	2.738436	0.726481
Aug-2016	Black-Scholes	285.189514	3.333716	14.298230	1.261132
	Reinforcement ANN	96.343925	2.549990	8.693088	0.904770
	Supervised ANN	17.648865	2.923735	3.313208	0.671951
Sep-2016	Black-Scholes	218.458923	4.170218	12.102951	1.196445
	Reinforcement ANN	63.544788	3.300921	6.744524	0.941828
	Supervised ANN	14.441139	3.846450	2.824541	0.884574
Oct-2016	Black-Scholes	187.673462	2.733971	11.673301	1.077606
	Reinforcement ANN	54.719086	2.083249	6.588595	0.752417
	Supervised ANN	21.655910	2.049526	2.528810	0.541491
Nov-2016	Black-Scholes	184.832001	2.933357	10.840181	1.093040
	Reinforcement ANN	55.737530	2.874570	6.307992	0.891128
	Supervised ANN	25.222687	3.783088	3.445950	0.805178
Dec-2016	Black-Scholes	247.691193	2.592708	12.789544	1.073321
	Reinforcement ANN	68.852280	2.297923	7.114644	0.881539
	Supervised ANN	22.338226	7.192957	2.924213	0.871336

Table 5: Performance of the three models on the test set by month

reinforcement ANN on all metrics except Hedging MSE.

Table 6 divides the dataset by moneyness. The same pattern holds for out-of-the-money options as well as for at-the-money options, but the Black-Scholes model outperforms both ANN based models on every metric for in-

Moneyness	Model	Hedging MAE	Hedging MSE	Observed MAE	Observed MSE
OTM, $\frac{S}{K} < 0.97$	Black-Scholes	200.156403	0.588415	11.307669	0.551204
	Reinforcement ANN	55.318222	0.260011	6.267632	0.345770
	Supervised ANN	7.251135	0.378938	2.225297	0.317774
ATM, $0.97 < \frac{S}{K} < 1.03$	Black-Scholes	191.135056	4.934059	11.412901	1.457267
	Reinforcement ANN	55.504204	4.096605	6.312860	1.168480
	Supervised ANN	19.572832	4.410431	3.189572	0.953541
ITM, $1.03 < \frac{S}{K}$	Black-Scholes	39.767418	0.334403	4.537481	0.380637
	Reinforcement ANN	39.650070	3.135508	4.968401	0.911614
	Supervised ANN	181.914169	26.621176	10.268289	2.227875

Table 6: Performance of the three models on the test set by moneyness

the-money options. The supervised ANN performs by far the worst on this category.

6 Summary and Discussion

In this Master's thesis I develop and present a new method for pricing financial derivatives. The economic reasoning behind the method is very similar to that of Black and Scholes (1973) seminal paper on option pricing. The biggest difference is that Black and Scholes solve the problem analytically, while I do it numerically with the help of a non-linear regression model (an artificial neural network). The main benefit of this approach is that it does not rely on a parametric specification of the stochastic process that generates the price of the underlying asset, like Black and Scholes assumes geometric Brownian motion. Instead I use a time series of the realized price of the underlying asset. Though the method can be applied to price a wide range of derivative securities, I have chosen to focus mainly on European call options.

The results in section 5.5 show that, for the most part, the supervised ANN model outperforms the reinforcement learning approach by most metrics. Still, I would not discard the reinforcement ANN for option pricing just yet (though I may be biased). The data set used in section 5 is in many ways the best case scenario for the data hungry supervised ANN. Almost any other underlying asset will have a less active options market than the S&P500 index. This will dramatically reduce the efficacy of a supervised ANN, but would not affect the performance of the reinforcement ANN as long as there is sufficient data on the underlying asset. As shown in section 4.3, the reinforcement ANN is also flexible in the kind of derivative securities it prices. The fact that it drastically outperforms the Black-Scholes model shows that it would be a good choice in any situation when the data from derivatives market is not substantial enough for the supervised learning approach. This could be the case for other underlying assets or other less traded derivative securities.

As discussed in section 3.4, there is also much room for improvement in the reinforcement ANN. Possible improvements include expanding from one dimensional first order approximation to multidimensional higher order approximation of the hedging portfolio. This requires solving some challenges, but is likely possible. Another possible improvement is to include more input variables. The most obvious choice is a measure of volatility. This could be based on a historic moving average, a GARCH-model estimate, or even implied volatility from options market data. More than one of these options could be used concurrently. For instance Amilon (2003) uses both 30-day and 10-day trailing volatility as input variables, theoretically enabling the neural network to learn the significance of both medium and short term volatility trends. Montesdeoca and Niranjan (2016) experiment with other input variables, such as trading volume.

Another possible use for the reinforcement ANN is as part of an ensemble model. An ensemble model aggregates the predictions of a set of models. The aggregate prediction is often better than that of the best single model in the set (Géron 2017). The aggregation can be done in multiple ways. A naive way would be to take an average of the various model's predictions. The set of models could also be used as inputs for an aggregating learning algorithm, that can learn the relative strengths and weaknesses of the input models, and weight them accordingly. Anders, Korn, and Schmitt (1998) found that including the Black-Scholes estimated option price as an input variable in a supervised ANN improved its predictions. As the reinforcement ANN outperformed the Black-Scholes model in almost all situations in section 5.5, it is likely that including it could improve performance even further.

References

Amilon, Henrik (2003). "A neural network versus Black–Scholes: a comparison of pricing and hedging performances". In: *Journal of Forecasting* 22.4, pp. 317–335.

Anders, Ulrich, Olaf Korn, and Christian Schmitt (1998). "Improving the pricing of options: A neural network approach". In: *Journal of forecasting* 17.5-6, pp. 369–388.

Bennell, Julia and Charles Sutcliffe (2004). "Black–Scholes versus artificial neural networks in pricing FTSE 100 options". In: *Intelligent Systems in Accounting, Finance and Management* 12.4, pp. 243–260.

Black, Fischer and Myron Scholes (1973). "The pricing of options and corporate liabilities". In: *Journal of political economy* 81.3, pp. 637–654.

Chicago Board Options Exchange, Cboe (2018). *S&P 500® Index Options - SPX.* URL: `http://www.cboe.com/products/stock-index-options-spx-rut-msci-ftse/s-p-500-index-options` (visited on 04/13/2018).

Cox, John C, Stephen A Ross, and Mark Rubinstein (1979). "Option pricing: A simplified approach". In: *Journal of financial Economics* 7.3, pp. 229–263.

Discount OptionData (2018). *Frequently Asked Questions.* URL: `https://www.discountoptiondata.com/home/faq` (visited on 04/13/2018).

FRED, Federal Reserve Economic Data (2018). *3-Month Treasury Bill: Secondary Market Rate.* URL: `https://fred.stlouisfed.org/series/DTB3` (visited on 04/13/2018).

Galindo-Flores, J (2000). "A framework for comparative analysis of statistical and machine learning methods: an application to the Black–Scholes option pricing model". In: *Computational Finance 1999*, pp. 635–660.

Géron, Aurélien (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.". Chap. 7, pp. 181–202.

Hamid, Shaikh A and Abraham Habib (2005). "Can neural networks learn the Black-Scholes model? A simplified approach". In:

Heston, Steven L (1993). "A closed-form solution for options with stochastic volatility with applications to bond and currency options". In: *The review of financial studies* 6.2, pp. 327–343.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5, pp. 359–366.

— (1990). "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks". In: *Neural networks* 3.5, pp. 551–560.

Hutchinson, James M, Andrew W Lo, and Tomaso Poggio (1994). "A non-parametric approach to pricing and hedging derivative securities via learning networks". In: *The Journal of Finance* 49.3, pp. 851–889.

Kreyszig, Erwin (2010). *Advanced engineering mathematics.* John Wiley & Sons. Chap. 12, pp. 540–605.

Liang, Xun et al. (2009). "Improving option price forecasts with neural networks and support vector regressions". In: *Neurocomputing* 72.13-15, pp. 3055–3065.

MacBeth, James D and Larry J Merville (1979). "An Empirical Examination of the Black-Scholes Call Option Pricing Model". In: *The journal of finance* 34.5, pp. 1173–1186.

— (1980). "Tests of the Black-Scholes and Cox Call Option Valuation Models". In: *The Journal of Finance* 35.2, pp. 285–301.

Malliaris, Mary and Linda Salchenberger (1993). “A neural network model for estimating option prices”. In: *Applied Intelligence* 3.3, pp. 193–206.

McDonald, Robert Lynch, Mark Cassano, and Rüdiger Fahlenbrach (2006). *Derivatives markets.* Vol. 2. Addison-Wesley Boston. Chap. 28, pp. 697–726.

Merton, Robert C (1973). “Theory of rational option pricing”. In: *The Bell Journal of economics and management science*, pp. 141–183.

— (1976). “Option pricing when underlying stock returns are discontinuous”. In: *Journal of financial economics* 3.1-2, pp. 125–144.

Montesdeoca, Luis and Mahesan Niranjan (2016). “Extending the feature set of a data-driven artificial neural network model of pricing financial options”. In: *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on.* IEEE, pp. 1–6.

Oslo Børs (2018). *OBX Total Return Index Derivatives.* URL: `https://www.oslobors.no/markedsaktivitet/#/derivativeUnd/OBX.OSE` (visited on 05/31/2018).

Park, Hyejin, Namhyoung Kim, and Jaewook Lee (2014). “Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over KOSPI 200 Index options”. In: *Expert Systems with Applications* 41.11, pp. 5227–5237.

Rubinstein, Mark (1985). “Nonparametric tests of alternative option pricing models using all reported trades and quotes on the 30 most active CBOE option classes from August 23, 1976 through August 31, 1978”. In: *The Journal of Finance* 40.2, pp. 455–480.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). “Learning representations by back-propagating errors”. In: *nature* 323.6088, p. 533.

Shreve, Steven E (2004). “Stochastic calculus for finance II: Continuous-time models”. In: vol. 11. Springer Science & Business Media. Chap. 4, p. 188.

A Simplified Example Code

Importing required libraries.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import tensorflow as tf

directory = 'C:/Users/Haakon'
```

Defining the neural network.

```
#network for synthetic data

activation = tf.tanh #activation function
hidden_layer = [50,50,50] #number neurons in each hidden layer
n_outputs = 1
learning_rate = .001

tf.reset_default_graph()

with tf.name_scope('inputs_processing'):
    X_input = tf.placeholder(tf.float32, shape = (None, 3), name =
    ↪  'X_input') #S, K, T
    X_input_ = tf.placeholder(tf.float32, shape = (None, 2), name =
    ↪  'X_input_') #S_, T_
    r = tf.fill([tf.shape(X_input)[0],1], 0., name = 'r') #interest rate
    ↪  if applicable
    #S, K, and T denote the underlying asset price, strike (exercise)
    ↪  price, time until expiration (exercise date)
    #The training requires the value of S and T at both times t and (t+h),
    ↪  the latter is denoted by a an underscore "_" at the end

    S = tf.slice(X_input, (0,0), (-1,1))
    K = tf.slice(X_input, (0,1), (-1,1))
    T = tf.slice(X_input, (0,2), (-1,1))
    X = tf.concat([S/(K*tf.exp(-r*T)), T], 1)#input matrix for ANN
    S_ = tf.slice(X_input_, (0,0), (-1,1))
    T_ = tf.slice(X_input_, (0,1), (-1,1))
    X_ = tf.concat([S_/(K*tf.exp(-r*T_)), T_], 1)#input matrix for ANN_

with tf.name_scope('ann'):

    #defines the nerual network architecture, inputs are S/K and T, output
     ↪  is C/K
    def ann(x, hidden_layer, n_outputs, activation, reuse = False):
        Z = tf.layers.dense(x, hidden_layer[0], activation = activation,
         ↪  name = 'hidden1', reuse = reuse)
        for i in range(1, len(hidden_layer)):
            Z = tf.layers.dense(Z, hidden_layer[i], activation =
             ↪  activation, name = 'hidden' + str(i+1), reuse = reuse)
        return tf.layers.dense(Z, n_outputs, name = 'out', reuse = reuse)

    out = ann(X, hidden_layer, n_outputs, activation) #out is ANN estimate
     ↪  of C/K
    out = tf.where(tf.greater(T, 1e-3), out, tf.maximum(S/K - 1, 0)) #if
     ↪  T<0.001 (basically if T==0), then max(S/K-1,0) is returned
     ↪  instead of ANN estimate
    out = K*out # multiply (C/K) by K to obtain C

    #derivatives of option price is computed
    delta = tf.gradients(out, S)[0]
    theta = tf.gradients(out, T)[0]
    gamma = tf.gradients(delta, S)[0]

    #same as above, but for option price at (t+h)
    out_ = ann(X_, hidden_layer, n_outputs, activation, reuse = True)
    out_ = K*tf.where(tf.greater(T_, 1e-3), out_, tf.maximum(S_/K - 1, 0))

with tf.name_scope('loss'):
    hedging_mse = tf.losses.mean_squared_error(labels = delta*(S_-S),
     ↪  predictions = (out_-out)) #this is the loss (objective) function,

with tf.name_scope('training'):
    optimizer = tf.train.AdamOptimizer(learning_rate) #ADAM optimization
     ↪  is used
    training_op = optimizer.minimize(hedging_mse)
```

```
with tf.name_scope('init_and_saver'):
    init = tf.global_variables_initializer()
    saver = tf.train.Saver()
```

Code for simulating geometric Brownian motion, and feeding data to the artificial neural network.

```
def stock_sim_path(S, alpha, delta, sigma, T, N, n):

    """Simulates geometric Brownian motion."""
    h = T/n
    mean = (alpha - delta - .5*sigma**2)*h
    vol = sigma * h**.5
    return S*np.exp((mean + vol*np.random.randn(n,N)).cumsum(axis = 0))

def get_batch2(stock_path,n, moneyness_range = (.5,2)):
    """Constructs theoretical options based on the time series
     ↪  stock_path"""

    picks = np.random.randint(0, len(stock_path)-1, n)
    T = np.random.randint(1, 150, (n,1))
    S = stock_path[picks]
    S_ = stock_path[picks+1]
    K = np.random.uniform(*moneyness_range, (n,1))*S
    X = np.hstack([S, K, T/250])
    X_ = np.hstack([S_, (T-1)/250])
    return X, X_
```

Training procedure.

```
#model training
n_epochs = 500 #number of training epochs
n_batches = 1000 #number of batches per epoch
batch_size = 10000 #number of theoretical options in each batch
T = 2 #years of training data
days = int(250*T)
```

```
stock_path = stock_sim_path(100, .05, 0, .15, T, 1, days) #simulate stock
 ↪  path
stock_path_test = stock_sim_path(100, .05, 0, .15, T, 1, days) #simulate
 ↪  stock path for cross-validation

#plot stock paths
plt.plot(stock_path, label = 'Training')
plt.plot(stock_path_test, label = 'Test')
plt.legend()
plt.show()

X_test, X_test_ = get_batch2(stock_path_test, batch_size) #get test-set

with tf.Session() as sess: #start tensorflow session
    init.run() #initialize variables
    for epoch in range(n_epochs):
        for batch in range(n_batches):
            X_train, X_train_ = get_batch2(stock_path, batch_size) #get
             ↪  batch of theoretical options
            sess.run([training_op], feed_dict = {X_input: X_train,
             ↪  X_input_: X_train_}) #training operation
        epoch_loss = hedging_mse.eval({X_input: X_test, X_input_:
         ↪  X_test_})
        print('Epoch:', epoch, 'Loss:', epoch_loss, 'BS Loss:',
         ↪  bs_hedging_mse.eval({X_input: X_test, X_input_: X_test_}))

    save_path = saver.save(sess, directory + '/ann_save.ckpt') #save model
     ↪  parameters
```