Computational Call Pricing

Zella Baig

HT23

1 Introduction

Within mathematical finance, there are a variety of methods used to price financial assets in order to extract utility from trades. One of the key hurdles within pricing these assets has been in mathematically modelling the highly nonlinear and stochastic behaviour of markets causing erratic changes in value. Driven by recent breakthroughs in computing, it has become viable to try and implement machine-learning based tools such as neural networks to price these assets. The goal with such approaches is to be able to then pick out any behaviours which are too difficult to analytically describe, an approach which has enjoyed success in many fields.

We look at the Black-Scholes-Merton (BSM) model, and use this model to train a neural network via artificial trades generated via a binomial pricing model. We then compare the performance of this model to a neural network trained on real data via extracting real-world trades taken from the S&P 500 [25] (a stock index for 500 popular US-based private corporations), with the aim of examining why one or the other might perform better at accurately predicted asset prices.

1.1 Preliminaries

Before we delve into the bulk of our discussion let us introduce some financial terminology [4].

1. A *call* is a financial option allowing the owner the right, but not obligation, to buy the asset at a specified price at a given time. In particular, we focus on *European calls*, which may only be exercised at their maturity time T, but we also briefly discuss *American calls* which may be exercised any time up to their maturity time.

2. The *strike price* K is the fixed price at which an asset may be exercised. In the context of calls, this is the price at which the option gives you the right to purchase the underlying asset.

3. The *risk-free-rate* r is the rate of return of an asset which has zero risk. In practice this may refer to long-term government bond rates [15].

4. Lastly, the *volatility* σ of an asset is a measure of the deviation in return for said asset.

With these definitions, we look now towards our pricing models.

2 Black-Scholes-Merton Model

In order to derive the expression for the European call price within this model we follow a similar procedure to that in [24]. We first assume the existence of an underlying asset S (e.g. a stock) which undergoes geometric Brownian motion, and a risk-free[1] asset B (such as a government bond). We then know that we have the equations

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

$$dB_t = rB_t dt \tag{2}$$

for a Wiener process W_t, adopting the notation that $X_t := X(t)$. We pause now to derive *Ito's Lemma*, an important result required to proceed. We follow the procedure outlined in [16].

Lemma 1. *Ito's Lemma: The differential of a continuous, time-dependent stochastic process $f(S,t)$ is given by*[2]

$$df(S,t) = \left(\frac{\partial f(S,t)}{\partial S}\mu(S,t) + \frac{\partial f(S,t)}{\partial t} + \frac{1}{2}\frac{\partial^2 f(S,t)}{\partial S^2}\sigma^2(S,t) \right) dt + \frac{\partial f(S,t)}{\partial S}\sigma(S,t)\,dW, \tag{3}$$

where S follows a geometric Brownian process

$$dS = \mu(S,t)\,dt + \sigma(S,t)dW \tag{4}$$

associated with drift μ, volatility σ, and Wiener process W.

Proof. Before proceeding let us immediately we drop arguments for $\mu(S,t), \sigma(S,t)$ for clarity. We have that

$$df(S,t) = \frac{\partial f}{\partial S}dS + \frac{\partial f}{\partial t}dt. \tag{5}$$

The Wiener process W by definition satisfies

$$dW \sim N(0, dt), \tag{6}$$

$$Cov(dW(t), dW(\tau)) = 0 \tag{7}$$

[1] Here "risk-free" refers to the idea that there is virtually no risk of non-payment of the value owed, as in a government bond.

[2] Note that we have dropped the subscript t for clarity, but both S and W are time dependent.

for $t \neq \tau$. Noting that in Ito calculus we have $dW^2 = dt$ [11], we write

$$\Delta S = \mu \Delta t + \sigma \Delta W \tag{8}$$
$$= \mu \Delta t + \sigma \epsilon \sqrt{\Delta t} \tag{9}$$

for $\epsilon \sim N(0,1)$. Then squaring and dropping higher order terms we have

$$\Delta S^2 = \sigma^2 \epsilon^2 \Delta t. \tag{10}$$

Further, given ϵ is a standard normal distribution we have the variance $\mathbb{V}[\epsilon]$ defined by

$$\mathbb{V}[\epsilon] := \mathbb{E}\left[\epsilon^2\right] - \mathbb{E}[\epsilon]^2 = 1 \quad \Rightarrow \quad \mathbb{E}\left[\epsilon^2\right] = 1, \tag{11}$$

where the simplification in (11) follows from the expectation of ϵ being 0. This then implies that

$$\mathbb{E}\left[\Delta t \epsilon^2\right] = \Delta t \tag{12}$$

following a simple multiplication. To calculate the variance $\mathbb{V}$ of $\epsilon^2 \Delta t$ note that

$$\mathbb{V}\left[\epsilon^2 \Delta t\right] = \mathbb{E}\left[\left(\epsilon^2 \Delta t - \mathbb{E}\left[\epsilon^2 \Delta t\right]\right)^2\right] \tag{13}$$
$$= \mathbb{E}\left[\left(\epsilon^2 \Delta t - \Delta t\right)^2\right] \tag{14}$$
$$= \Delta t^2 \mathbb{E}\left[\left(\epsilon^2 - 1\right)\right] \tag{15}$$
$$= \Delta t^2 \mathbb{E}\left[\epsilon^4 - 2\epsilon^2 + 1\right]. \tag{16}$$

Noting that

$$\mathbb{E}\left[\epsilon^4\right] = Cov(\epsilon^2, \epsilon^2) + \mathbb{E}[\epsilon^2]^2 = 2, \tag{17}$$

we write (16) as

$$\mathbb{V}\left[\epsilon^2 \Delta t\right] = \Delta t^2 (2 - 2 \cdot 1 + 1) = \Delta t^2. \tag{18}$$

Therefore as $\Delta t \to 0$ we have that $\epsilon^2 \Delta t \to \mathbb{E}\left[\epsilon^2 \Delta t\right] := \Delta t$. Applying this result in (10) we have that

$$\lim_{\Delta S, \Delta t \to 0} \Delta S^2 \to \sigma^2 dt, \tag{19}$$

which we insert into

$$\Delta f = \frac{\partial f}{\partial S} \Delta S + \frac{\partial f}{\partial t} \Delta t + \frac{1}{2} \frac{\partial^2 f}{\partial S^2} \Delta S^2 + \frac{\partial^2 S}{\partial S \partial t} \Delta S \Delta t + \frac{1}{2} \frac{\partial^2 f}{\partial t^2} \Delta t^2 \ldots \tag{20}$$

to give (after ignoring higher order terms)

$$df = \frac{\partial f}{\partial S} dS + \frac{\partial f}{\partial t} dt + \frac{1}{2} \frac{\partial^2 f}{\partial S^2} \sigma^2 dt. \tag{21}$$

Substituting our expression for the Ito process (4) into (21), we finally arrive at

$$df = \left(\frac{\partial f}{\partial S}\mu + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2\right)dt + \frac{\partial f}{\partial S}\sigma dW. \tag{22}$$

□

Applying Ito's Lemma to $\ln(S)$ and $\ln(B)$ from (1)-(2), we arrive at the stochastic differential equations (reintroducing subscripts)

$$d\ln(S_t) = \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dW_t, \tag{23}$$

$$d\ln(B_t) = r dt. \tag{24}$$

Following [4], we then integrate from t to T to have the solutions

$$S(t;T) = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)(T-t) + \sigma\int_t^T dt\, W_t\right), \tag{25}$$

$$B(t;T) = \exp\left(r(T-t)\right). \tag{26}$$

One of the requirements for the BSM model is performance under a measure accounting for no arbitrage, which necessitates a risk-neutral measure $\mathbb{Q}$ [24] such that the discounted stock price dependant on B is *martingale*, a variable such that the conditional expectation of the next value in a series is the present value regardless of previous information held [26].

To this end write

$$\frac{dS_t}{S_t} = r dt + \sigma dW_t^{\mathbb{Q}} \tag{27}$$

where

$$W_t^{\mathbb{Q}} = W_t + \frac{\mu - r}{\sigma}t, \tag{28}$$

and consider the *discounted price*

$$\hat{S}_t := \frac{S_t}{B_t}, \tag{29}$$

where we claim that $\hat{S}_t$ will be a martingale.

Applying Ito's Lemma to $\hat{S}_t$ and substituting in for S_t and B_t, we have that

$$d\hat{S}_t = \frac{\partial \hat{S}_t}{\partial B_t}dB_t + \frac{\partial \hat{S}_t}{\partial S_t}dS_t = \sigma \hat{S}_t dW_t^{\mathbb{Q}}, \tag{30}$$

which when integrating from 0 to t and with $\int_0^t dW_t^{\mathbb{Q}} = W_t^{\mathbb{Q}}$ has the solution

$$\hat{S}_t = \hat{S}_0 \exp\left(-\frac{\sigma^2}{2}t + \sigma W_t^{\mathbb{Q}}\right). \tag{31}$$

Now consider for $p < t$ we have (with $\mathbb{F}$ referring to all information we know up to time p),

$$\mathbb{E}^{\mathbb{Q}}\left[\hat{S}_t|\mathbb{F}_p\right] = \hat{S}_0 \exp\left(-\frac{\sigma^2}{2}t\right)\mathbb{E}^{\mathbb{Q}}\left[\exp\left(\sigma W_t^{\mathbb{Q}}\right)|\mathbb{F}_p\right] \tag{32}$$

$$= \hat{S}_0 \exp\left(-\frac{\sigma^2}{2}t + \sigma W_p^{\mathbb{Q}}\right)\mathbb{E}^{\mathbb{Q}}\left[\exp\left(\sigma\left(W_t^{\mathbb{Q}} - W_p^{\mathbb{Q}}\right)\right)|\mathbb{F}_p\right] \tag{33}$$

$$= \hat{S}_0 \exp\left(-\frac{\sigma^2}{2}t + \sigma W_p^{\mathbb{Q}}\right)\mathbb{E}^{\mathbb{Q}}\left[\exp\left(\sigma\left(W_{t-p}^{\mathbb{Q}}\right)\right)|\mathbb{F}_0\right] \tag{34}$$

$$= \hat{S}_0 \exp\left(-\frac{\sigma^2}{2}t + \sigma W_p^{\mathbb{Q}}\right)\exp\left(\frac{\sigma^2}{2}(t-p)\right) \tag{35}$$

$$= \hat{S}_0 \exp\left(-\frac{\sigma^2}{2}p + \sigma W_p^{\mathbb{Q}}\right) \tag{36}$$

$$= \hat{S}_p, \tag{37}$$

where we have added 0 with the term in green in (33), used the definition of $W^{\mathbb{Q}}$ in (34), and used the moment-generating function of a normal distribution in (35). Thus, we have shown $\hat{S}$ to be $\mathbb{Q}$-martingale.

Pricing now a call C_E^{BSM} with time to maturity τ we have, with F being the log-normal cumulative distribution function,

$$C_E^{BSM} = \exp(-r\tau)\mathbb{E}^{\mathbb{Q}}\left[(S_T - K)^+|\mathbb{F}_t\right] \tag{38}$$

$$= \exp(-r\tau)\int_K^\infty dF(S_T)\,(S_T - K) \tag{39}$$

$$= \exp(-r\tau)\int_K^\infty dF(S_T)\,S_T - \exp(-r\tau)\int_K^\infty dF(S_T)\,K. \tag{40}$$

With ϕ the cumulative distribution function of the normal distribution, the first integral in (40) is simply

$$\int_K^\infty dF(S_T)\,S_T = \mathbb{E}^{\mathbb{Q}}[S_T|S_T > K] = S_t e^{r\tau}\phi(d_1), \tag{41}$$

where

$$d_1 := \frac{\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\tau\right)}{\sigma\sqrt{\tau}}. \tag{42}$$

Note that in (42) we have used the standard result for the conditional expectation of a log-normal distribution, which can be derived as done in e.g. [6]. The crux of the argument relies upon the fact that S_T follows a log-normal distribution with mean $\ln(S_t) + \left(r - \sigma^2/2\right)\tau$ and variance $\sigma^2\tau$, from where the result directly follows.

Proceeding, the second integral in (40) may be written as

$$\int_K^\infty dF(S_T) = 1 - F(K) \tag{43}$$

$$= \phi(d_2), \tag{44}$$

with

$$d_2 := 1 - d_1 \tag{45}$$

$$= \frac{ln\left(\frac{S_t}{K}\right) + \left(r - \frac{\sigma^2}{2}\tau\right)}{\sigma\sqrt{\tau}}. \tag{46}$$

With the notation in [4], we then have that in the BSM model a European call option is priced at

$$C_E^{BSM}(t,T,S,K) := S_t\phi(d_1) - Ke^{-r(T-t)}\phi(d_2). \tag{47}$$

We look next at examining pricing via a binomial model. This has several advantages, namely that it is computationally simple to implement for e.g. American options as it allows calculation of prices during steps intermediate to the current time and expiry time. We seek to implement this model primarily to draw comparisons against the BSM model, and ultimately to compare with pricing via a neural network.

3 Binomial Pricing Model

The basic principle by the binomial (BM) model is outlined e.g. in [12], in which we utilise a tree with two different branches at each node: we claim that at a given time-step the price S may either go up to uS with probability p, or go down to dS with probability $1-p$. If we assume intermediate time-steps Δt and risk-free rate r, we wish to find the probability p such that we may construct the previous price

$$C_{E,0} := \exp(-r\Delta t)\left(pC_{up} + (1-p)C_{down}\right) \tag{48}$$

where C_{up} and C_{down} are the prices of the option at the next time-step assuming the price of the underlying asset moves up or down. By successively combining these single binary splits we are thus able to construct a tree to price a given option.

We note that (27) has the solution

$$S_t = S_0 \exp\left((r - \frac{\sigma^2}{2})t + \sigma W_t^{\mathbb{Q}}\right), \tag{49}$$

and so we note that in a discrete setting

$$\frac{S_{t+\Delta t}}{S_t} = \exp\left(\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\left(W^{\mathbb{Q}}_{t+\Delta t} - W^{\mathbb{Q}}_t\right)\right) \tag{50}$$

which is distributed log-normal$\left((r-\frac{\sigma^2}{2})\Delta t, \sigma^2\Delta t\right)$. Then, equating our conditions on the mean and variation of the stock price S to the mean and probability of this log-normal distribution, which give the equations

$$pu + (1-p)\,d = \exp(r\Delta t), \tag{51}$$

$$pu^2 + (1-p)\,d^2 = \exp\left(\left(2r+\sigma^2\right)\Delta t\right). \tag{52}$$

Using the conditions in the Cox-Ross-Rubinstein model that $ud = 1$ [8], we may then solve for p, u, d to get

$$p := \frac{\exp(r\Delta t) - d}{u - d}, \tag{53}$$

$$u := \exp\left(\sigma\sqrt{\Delta t}\right), \tag{54}$$

$$d := \exp\left(-\sigma\sqrt{\Delta t}\right). \tag{55}$$

We can then use our parameters p, u, d alongside (48) to derive the price of the call at a given node in a tree:

$$C_E^{BM} := \exp(-rN\Delta t)\sum_{i=0}^{N}\binom{N}{i}p^i(1-p)^{N-i}\max\left\{S_0u^id^{N-i} - K, 0\right\}. \tag{56}$$

We can now demonstrate that (56) converges to the expected formula in the BSM model. We follow a similar procedure as in [22].

Define α to be the smallest integer such that $S_0u^\alpha d^{N-\alpha} > K$, or in other words we have

$$\alpha := \left\lceil \frac{\ln\left(\frac{K}{Sd^N}\right)}{\ln\left(\frac{u}{d}\right)} \right\rceil. \tag{57}$$

Noting then that for all $i < \alpha$ the term inside the maximum in (56) is 0, and for all $i > \alpha$ it is $S_0u^\alpha d^{N-\alpha} - K$, we write

$$C_E^{BM} = \exp(-rN\Delta t)\sum_{i=\alpha}^{N}\binom{N}{i}p^i(1-p)^{N-i}\left(S_0u^id^{N-i} - K\right). \tag{58}$$

Expanding (58) out we have

$$C_E^{BM} = S_0\sum_{i=\alpha}^{N}\binom{N}{i}p^i(1-p)^{N-i}e^{-rN\Delta t}u^id^{N-i} - e^{-rN\Delta t}K\sum_{i=\alpha}^{N}\binom{N}{i}p^i(1-p)^{N-i}. \tag{59}$$

We now claim that we may write the right hand side of(59) as

$$S\Phi[\alpha, N; \hat{p}] - Ke^{-rN\Delta t}\Phi[\alpha, N; p], \tag{60}$$

where $\Phi[n_1, n_2; p]$ is the discrete binomial cumulative distribution function (that is to say, the probability for a successful binomial trial to occur within $n_1, n_1 + 1 \dots n_2$ trials with probability p), and

$$\hat{p} := \frac{up}{\exp(r\Delta t)}. \tag{61}$$

Proof. The second sum in (59) is $\Phi[\alpha, N, p]$ by definition. Considering the first sum in (59), we write $\exp(r\Delta t) := \beta$ for simplicity, such that $\hat{p} = up/\beta$. Now note that

$$\hat{p}^i(1-\hat{p})^{N-i} = \left(\frac{up}{\beta}\right)^i \left(1 - \frac{up}{\beta}\right)^{N-i} \tag{62}$$

$$= \frac{p^i u^N}{\beta^N} \left(\frac{\beta - up}{u}\right)^{N-i} \tag{63}$$

$$= \frac{p^i u^i}{\beta^N} (d(1-p))^{N-i} \tag{64}$$

$$= \frac{p^i u^i d^{N-i}}{\beta^N} ((1-p))^{N-i} \tag{65}$$

where in going from (63) to (64) we have used (51). Substituting this result into the first summation in (59), and reinserting the full expression for β we have thus shown that

$$C_E^{BM} = S\Phi[\alpha, N; \hat{p}] - Ke^{-rN\Delta t}\Phi[\alpha, N; p]. \tag{66}$$

As we take the limit of $N \to \infty$ and $\Delta t \to 0$ we can thus arrive at our expression (47), with details of these extra steps (which involve taking limits of the discrete binomial cumulative distribution) may be found in e.g. [8]. We can also examine this convergence computationally, as shown in Figure 1 which demonstrates a clear inverse relationship between total least-squares loss and depth of binomial tree. □

Having now developed the binomial pricing model, we move towards developing an understanding of neural networks as to be able to apply them to pricing calls via the BSM model.

4 Neural Networks

In order to build up to what a neural network is we must first define an artificial neuron. An artificial neuron (sometimes known as a unit) takes as input a vector $\vec{x}$, bias $\vec{b}$,

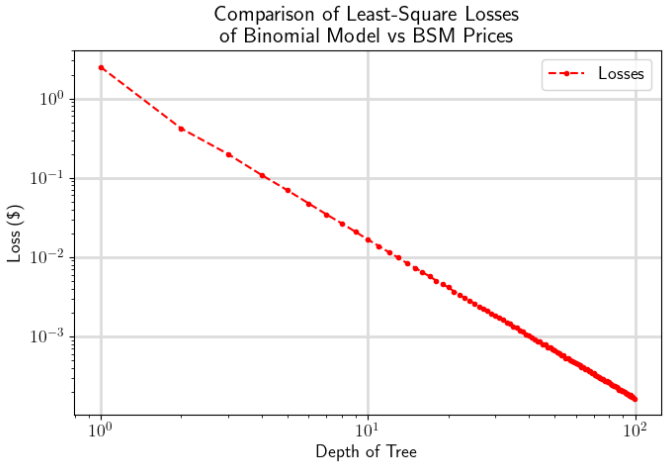


Figure 1: Log-log plot of least-squares loss of the binomial pricing model against the BSM model, for tree depths ranging from $1-100$.

weights $\vec{w}$, and activation function f, and outputs the value of $f\left(\vec{b}+\vec{x}\cdot\vec{w}\right)$ [2]. The name for this object comes from the rough relation between a biological neural which may be "activated". Thus, a *neural network* may be thought of as multiple layers of neurons, each neuron taking as input the outputs from the neurons from the layer before it.

We introduce some notation to expand upon this point. Denote by a^i_j the j^{th} neuron in the i^{th} layer of the neural network. Thus, in theory we may have many layers of neurons giving rise to a *Deep Neural Network* (DNN). We present a pictorial example of DNN in Figure 2; note the *hidden layers* which correspond to the neurons not visible as either inputs or outputs from the neural network.

In simple terms, neural networks work by seeking to optimise the weights $\vec{W}^i$ for a given layer, usually via seeking to minimise some loss function $L(\vec{W}^i)$. Standard techniques build off the principles of gradient-descent, where we have for a general vector $\vec{x}$

$$\vec{x}_{t+1}=\vec{x}_t-\eta\vec{g}_t, \tag{67}$$

with $\vec{g}_t:=\nabla_{\vec{x}_t}L\left(\vec{x}\right)$ being the gradient of the loss function, and η represents a hyperparameter known as the *learning rate*, i.e. the "stepsizes" between successive iterations of weights. In practice gradient-descent presents issues when the amount of functions to be differentiated is high (given the complexity of calculation) and so modern methods often utilise *stochastic* gradient-descent based algorithms. In these methods we replace $\vec{g}_t$ with $\vec{G}_t$, a randomly chosen vector with the condition that $\mathbb{E}\left[\vec{G}_t\right]=\vec{g}_t$. Realistically, when

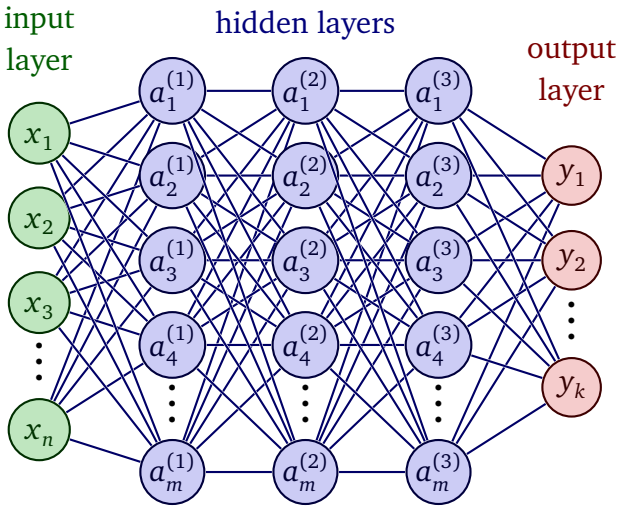


Figure 2: An example of a DNN with n inputs, k outputs, and three hidden layers with m neurons per hidden layer.

we may write $L(\vec{x}) = \sum_i l_i$, we then choose $\vec{G}_t = \nabla_{\vec{x}_t} l_i$. Stochastic gradient-descent then reduces the computational complexity of each update step potentially by orders of magnitude, at the cost of potentially erratic convergence [5].

We take a moment to discuss the optimisation algorithm we employ within our neural network, Adaptive Moment Estimation (commonly referred to as Adam).

4.1 Adam

Adam seeks to implement the benefits of two recently successful algorithms, AdaGrad and RMSProp [10], both of which utilise adaptive stepsizes. To illustrate why using adaptive stepsizes might be desirable, consider η as an analogue for "resolution"; where the gradient is changing rapidly we seek to capture these rapid shifts, and where the gradient is flat we wish not to expend computational power on relatively pointless calculations.

With all operations done element-wise and with $(\alpha)^t$ denoting raising α to the power t, we define the scheme as [17]

$$\textbf{Adam:} \qquad \vec{x}_{t+1} = \vec{x}_t - \frac{\eta_t}{\sqrt{\vec{B}_{t+1}}}\vec{M}_{t+1}, \qquad \begin{cases} \vec{m}_{t+1} = \beta_1 \vec{m}_t + (1-\beta_1)\vec{g}_t, \\ \vec{b}_{t+1} = \beta_2 \vec{b}_t + (1-\beta_2)\vec{g}_t^{\,2}, \\ \vec{M}_{t+1} = \dfrac{\vec{m}_{t+1}}{1-(\beta_1)^t}, \\ \vec{B}_{t+1} = \dfrac{\vec{b}_{t+1}}{1-(\beta_2)^t}, \end{cases} \tag{68}$$

with $\vec{m}_0 = \vec{b}_0 = 0$, η_t being the stepsize (which is sometimes set to $\eta_t := \eta/\sqrt{t}$), and $\beta_1, \beta_2 \in [0,1)$;he naming of the scheme "Adaptive Moment Estimation" arises from the utilisation of both the first and second moments of the gradient $\vec{g}_t$. The key facet of the method is within the ratio $\Delta_{t+1} := \vec{M}_{t+1}/\sqrt{\vec{B}_{t+1}}$ which may be thought of as a very loose analogue of the "average" gradient over the "deviation" of the gradient. When the variance in the gradient is high Δ_{t+1} shrinks (leading to finer resolution in step direction), and vice-versa when the variance in gradient is large. While it is known that Adam does not always converge to an optimal solution, proposed alternative methods such as AMSGrad do not necessarily give better results in practical contexts [13], and so Adam continues to be widely used.

We also discuss in brief the activation function we use in our neurons.

4.2 Rectified Linear Activation Function

An issue inherent to many activation functions such as the *sigmoid*, where $f(x) = (1+e^{-x})^{-1}$, is *saturation*. Saturation refers to the phenomenon in which the gradients of the activation functions are very flat, which means that for a small perturbation of input, the output of the neuron does not change much - potentially trapping the neuron in some sub-optimal parameter region. To account for this we utilise the *rectified linear activation function*, commonly referred to as ReLU, which is defined as $f(x) = \max(0,x)$ [2]. A visual comparison between ReLU and sigmoid activation functions is shown in Figure 3. Note how the gradient for the sigmoid is approximately zero for all points except around a small region near the origin, whereas with the ReLU we avoid saturation on half the real line, often leading to better performance experimentally than other activation functions.

With these tools in place, we now turn towards implementing a call option pricing model via neural networks.

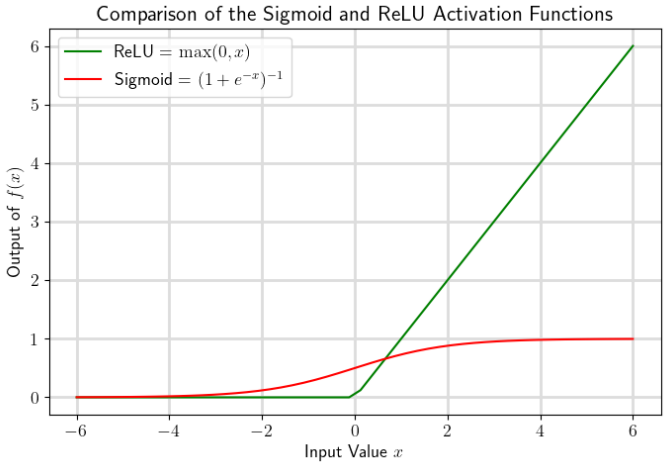


Figure 3: Comparison of the sigmoid and ReLU activation functions for $x \in [-6, 6]$.

5 Implementation

5.1 Dataset

We consider data from the S&P 500, for non-dividend paying American calls (which we know from e.g. [3] are priced the same as European calls). The dataset was scraped from Yahoo Finance [28] over three days, with information on implied volatility coming from AlphaQuery [1], and has information on 13,669 options gathered over this period. We present some information on the dataset in Table 1.

Parameter:	C (\$)	S (\$)	K (\$)	τ (Y)	σ
Range:	$0-541$	$3-1475$	$1-1710$	$0.004-3.034$	$0.163-0.666$
Mean:	28	194	218	0.472	0.378
Variance:	2200	25249	40280	0.332	0.011

Table 1: Parameters gathered from 13,669 trades taken from the S&P 500 over three days in March 2023.

We split the dataset into a 81% : 9% : 10% training-to-validation-to-testing split such that we aim to prevent over-fitting of our model (via comparing our models' performance to the validation set whilst still training), and also leave an unseen dataset to ultimately compare our model against once training has been completed. Furthermore, we use a feature of the BSM model wherein we can divide (47) by K to attain what we dub the "normalised" call price. This step is useful as we then reduce the number of "features"

(or variables) we seek to fit via our neural network as we can then seek to fit S_t/K rather than both S_t and K; this in turns reduces the computational complexity of our problem.

5.2 Models

Before proceeding, let us introduce some notation to aid us in defining the models we use.

- C_{scrape}: The set of call prices for the scraped option data.
- D_{scrape}: The set of tuples of parameters $(S_t, K, \tau, \sigma, r)$ for the scraped option data.
- C_{binom}: The set of predicted prices generated when each tuple in D_{scrape} is fed into the binomial model ran for N steps, as outlined in (56).

We attempt to generate two different neural networks:

1. The "stock" model, which is trained using data from D_{scrape} and C_{scrape}, and outputs predicted prices when given some option parameters D.
2. The "binomial" model, which is trained using data from D_{scrape} and C_{binom}, and outputs predicted prices when given some option parameters D.

For both these models, we utilise a least-squares loss function

$$L(\tilde{C}) := \frac{1}{2m}\sum_{i=1}^{m}(\tilde{C}_i - C_{scrape,i})^2 \tag{69}$$

whilst training the models, with $\tilde{C}_i$ a single prediction on a call within the entire array $\tilde{C}$ of call predictions, for a given model. Note that we discuss performance in both cases against C_{scrape} as we wish to compare how either method of training a model fares against real-life trades.

Immediately an issue is raised wherein we must make a choice to truncate the sum in (56) to some arbitrary N. We choose a depth of tree of 50, which we justify via the consideration of time of generation for the price, which is shown in Figure 4; a depth of 50 leads to a time of generation less than 100ms which we set as our benchmark.

We train using a neural network four layers deep with 100 neurons in each layer, and train for 100 iterations with early stopping enabled to prevent over-fitting if the validation score plateaus; this ends up being a sensible decision as justified by Figure 5. Here we see stopping for the binomial model before 20 iterations and for the stock model before 35 iterations. Note that in both cases the validation curves, that is to

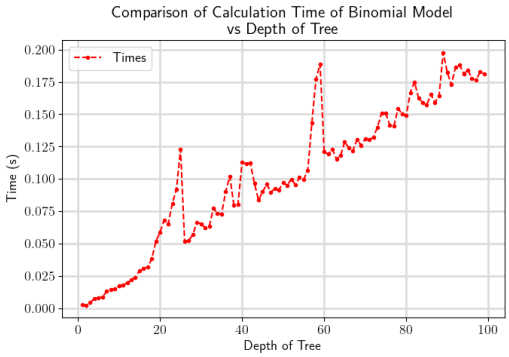


Figure 4: Comparison of time to generate call price using the binomial model for trees ranging between 1−100 nodes deep.

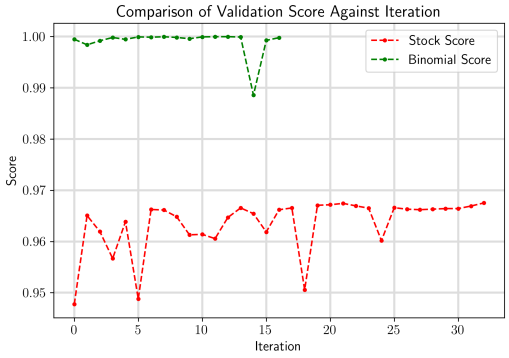


Figure 5: Comparison of validation scores against training iteration for the stock and binomial models.

say the R^2 values for the models' predictions to the validation dataset, plateau near 1. Further, note that the binomial model seems to give somewhat better validation scores.

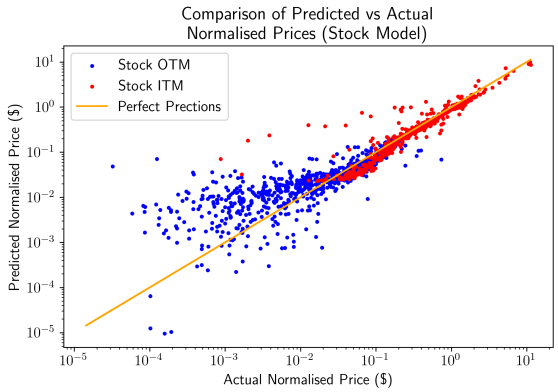
This is as we expect since the validation score is a representation of how well the neural network captures the underlying information within the model, and the binomial model (which itself is an abstraction of the BSM model) is a simplified representation of how we expect real-world calls to behave. It is thus natural that there will be behaviours (such as erratically priced calls) which the BSM model is not able to capture. Note at this point that we see an R^2 for the binomial and stock models of 0.986 and 0.988 respectively on the training data, showing good fitting.

6 Results

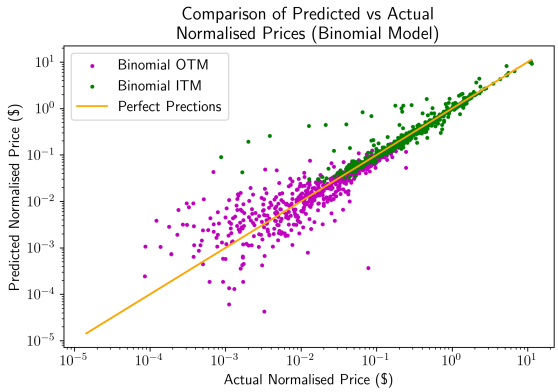
We present our results in Figure 6. We note that the stock model tends to under-price *in the money*[3] (ITM) calls and over-price *out-of-the-money* (OTM) calls (with the over-pricing behaviour significantly worse as actual price drops), whereas the binomial model has a much more even dispersion of pricing for both cases. Nevertheless, we actually see that the binomial model performs slightly worse than the stock model: we see an R^2 of 0.952 and 0.953 on predictions versus testing data for the two models respectively. However, this result is promising as it demonstrates both models perform reasonably well on European call data; this is demonstrated further in Figure 6(c) which shows good adherence to the perfect prediction line for both models.

We also note that in general, both models seem to predict ITM calls worse than they

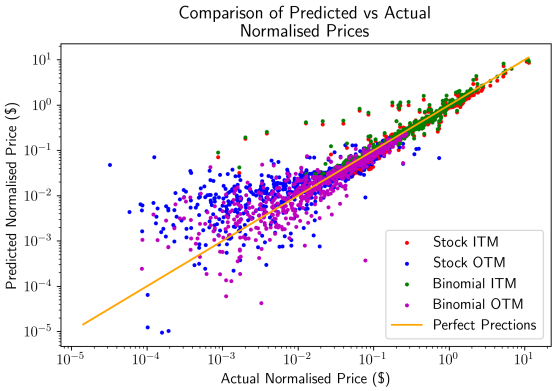
[3]Where $S > K$.



(a) Prediction comparison for stock model.



(b) Prediction comparison for binomial model.



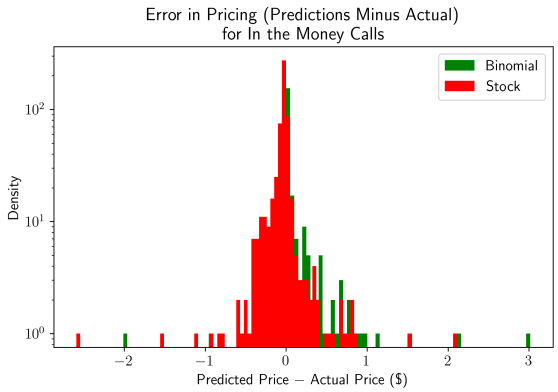
(c) Prediction comparison for both models.

Figure 6: Comparison of predicted and actual prices via neural networks trained.

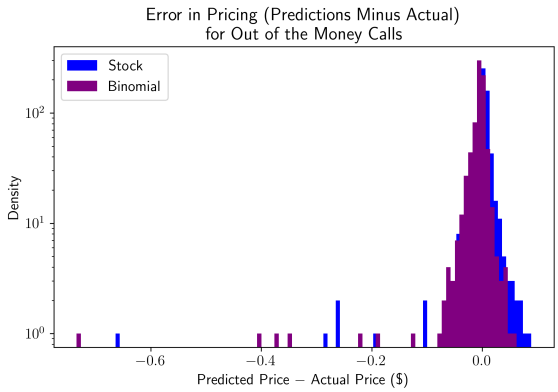
do OTM calls, which can be shown in Figures 6(a) and 6(b) by both the ITM predictions lying below the perfect prediction line. Given we are using a log-log scale this erroneous prediction might instead demonstrate difficulties in pricing very highly valued assets, which lines up with the discussion above on under-pricing ITM calls.

We can also examine the errors in more detail as shown in Figure 7. Figure 7(a) demonstrates well how both models tend to under-price ITM calls slightly, and Figure 7(b) demonstrates how the stock model over-prices OTM calls more than the binomial model. However, we can also see from this figure that the binomial model also presents more skewed results in pricing, which is perhaps better illustrated in Figure 7(c).

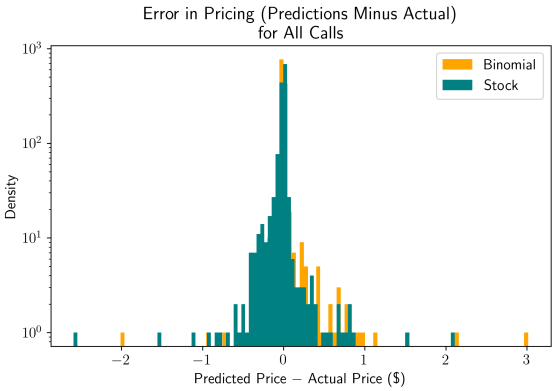
Examining the errors for all calls in more detail, consider Table 2 where we present some parameters for both the stock and binomial models, and where we use kurtosis



(a) Price prediction comparison for ITM calls.



(b) Price prediction comparison for OTM calls.



(c) Price prediction comparison for all calls.

Figure 7: Comparison of pricing errors for the binomial and stock neural networks.

defined as the *excess kurtosis* [20]

$$K(x) := \mathbb{E}\left[\left(\frac{x-\mu}{\sigma}\right)^4\right] - 3 \tag{70}$$

as the measure of tailed-ness of our errors.

Parameter:	Mean ($)	Variance ($\$^2$)	Skew	Kurtosis
Stock Model:	−0.0193	0.0245	−1.5678	96.9448
Binomial Model:	0.0020	0.0258	6.7271	143.5688

Table 2: Comparison of pricing errors in the stock and binomial neural network models via various statistical parameters.

Note in particular that the stock model outperforms the binomial model in every metric (significantly so in the skewness and kurtosis). This is as we expect: the stock

model is more likely to pick up inherent features to call pricing that the BSM model (and thus the binomial model) simply does not account for. An example of this could be seen in its treatment of price fluctuations; the BSM model assumes Gaussian jumps which empirically we know to not be a true representation of price variation (hence the proposal of models such as Merton's Jump Diffusion which seek to incorporate this behaviour [19]).

Nevertheless, given the means of the errors on both models are very similar (and that these values are also relatively low given that the mean and variance for the normalised call price on the testing dataset are 0.2678\$ and 0.5351$\$^2$ respectively), we see that both models are reasonably effective at pricing European call options but that the stock model trained on actual data slightly outperforms.

In theory, we could extend this examination to consider further types of options (such as Americans which may be executed at time $t \leq T$) but we expect that regardless of the model used a model trained on stock data should outperform any other model utilising some set of assumptions. This is, of course, provided we adequately train our models which necessitates further exploration into how much data is required to adequately train our models to some desired degree.

We expect this inherent advantage in models trained on stock data as we expect such a model to gather "insight" into how prices fluctuate without any erroneous simplifying assumptions which would render price predictions incorrect.

Nevertheless, we have shown promising results for neural network-based pricing for options given that we have trained on only a few thousand options. Given recent advances in computational power and optimisation algorithms, amelioration of these results provided more data and potentially better optimised networks is very likely.

7 Conclusions

We have examined the pricing of European call options via computational methods via neural networks. We have first developed an approach towards pricing these calls via an analytic expression and then found a computational approximation of this model via the binomial pricing model for options. Further, we have demonstrated that a neural network trained on data gathered via assuming that a binomial pricing model gives accurate results but slightly under-performs a similar neural network trained instead on actual asset trades.

Furthermore we have also demonstrated some flaws inherent to the BSM model

wherein it cannot accurately account for pricing of calls due to factors such as the assumption of constant volatility and log-normal distributions within stock prices, which we propose that a neural network trained on actual data is able to slightly better infer, and so better price European calls.

We have also demonstrated that both of these models can clearly be extended towards other use-cases such as in examining American options, or in the case of the neural network trained on actual data, other exotic options as well.

In further examination of computational pricing models we propose that data be taken over a wider time-frame (given the current limitation of data taken over three days) and from a wider array of sources rather than from a single index. Given the wide applicability of neural networks an extension in this manner should be able to provide reasonable results in terms of pricing options, though further exploration is necessary in this direction to make concrete claims.

8 References

[1] *AlphaQuery - Get deeper insights into more stocks in less time.* URL: https://www.alphaquery.com/ (visited on 03/14/2023).

[2] Phil Blunsom and Matthew Fellows. *Machine Learning - Michaelmas Term 2022*. Lectures. URL: https://www.cs.ox.ac.uk/teaching/materials22-23/ml/lectures/ (visited on 11/19/2022).

[3] Stephen Blyth. *An Introduction to Quantitative Finance*. Google-Books-ID: SXbcAAAAQBAJ. OUP Oxford, Nov. 2013. 193 pp. ISBN: 978-0-19-966659-1.

[4] Álvaro Cartea. *B8.3: Mathematical Modelling of Financial Derivatives - Hilary Term 2023*. Lecture Slides. (Visited on 03/06/2023).

[5] Don M. Chance. *A Synthesis of Binomial Option Pricing Models for Lognormally Distributed Assets*. Rochester, NY, Dec. 3, 2015. URL: https://papers.ssrn.com/abstract=2698699 (visited on 03/13/2023).

[6] Alfred Chong. *ASRM 410 Investments and Financial Markets - Fall 2018*. Chapter 4 Black-Scholes Option Pricing Model. (Visited on 03/14/2023).

[7] Rama Cont and Peter Tankov. *Financial modelling with jump processes*. Chapman & Hall/CRC financial mathematics series. Boca Raton, Fla: Chapman & Hall/CRC, 2004. 535 pp. ISBN: 978-1-58488-413-2.

[8] John C. Cox, Stephen A. Ross, and Mark Rubinstein. "Option pricing: A simplified approach". In: *Journal of Financial Economics* 7.3 (Sept. 1, 1979), pp. 229–263. ISSN: 0304-405X. DOI: 10.1016/0304-405X(79)90015-1. URL: https://www.sciencedirect.com/science/article/pii/0304405X79900151 (visited on 03/14/2023).

[9] Robert Culkin and Sanjiv R. Das. "Machine learning in finance: the case of deep learning for option pricing". In: *Journal of Investment Management* 15.4 (2017), pp. 92–100.

[10] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. *A Simple Convergence Proof of Adam and Adagrad*. Number: arXiv:2003.02395. Oct. 17, 2022. arXiv: 2003.02395[cs,stat]. URL: http://arxiv.org/abs/2003.02395 (visited on 03/13/2023).

[11] Bruce Dieffenbach. *Financial Economics, Economics 802 - Fall 2013*. Ito's Formula. URL: https://www.albany.edu/~bd445/Economics_802_Financial_Economics_Slides_Fall_2013/Ito_Formula.pdf (visited on 03/14/2023).

[12] Christopher Foot. *Financial Physics - Trinity Term 2013*. The binomial tree model: a simple example of pricing financial derivatives. URL: https://users.physics.ox.ac.uk/~Foot/Phynance/Binomial2013.pdf (visited on 03/13/2023).

[13] Sylvain Gugger. *Experiments with Adam*. original-date: 2018-07-02T22:03:11Z. Feb. 28, 2023. URL: https://github.com/sgugger/Adam-experiments (visited on 03/13/2023).

[14] N. Gugole. "Merton Jump-Diffusion Model Versus The Black and Scholes Approach For the Log-Returns and Volatility Smile Fitting". In: *International Journal of Pure and Apllied Mathematics* 109.3 (Oct. 1, 2016). ISSN: 1311-8080, 1314-3395. DOI: 10.12732/ijpam.v109i3.19. URL: http://www.ijpam.eu/contents/2016-109-3/19/ (visited on 03/12/2023).

[15] Adam Hayes. *What Is the Risk-Free Rate of Return, and Does It Really Exist?* Investopedia. URL: https://www.investopedia.com/terms/r/risk-freerate.asp (visited on 03/14/2023).

[16] John Hull. *Options, futures, and other derivatives*. Ninth edition, global edition. Harlow, England Munich: Pearson, 2018. 891 pp. ISBN: 978-1-292-21289-0.

[17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Number: arXiv:1412.6980. Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980[cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 03/13/2023).

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386 (visited on 03/13/2023).

[19] Yuh-Dauh Lyuu. *Principles of Financial Computing - Spring Semester 2015*. Merton's Jump-Diffusion Model. URL: https://www.csie.ntu.edu.tw/~lyuu/finance1/2015/20150513.pdf (visited on 03/12/2023).

[20] *NIST: 1.3.5.11. Measures of Skewness and Kurtosis*. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm (visited on 03/14/2023).

[21] Gergely (Greg) Orosi. *A Simple Derivation of Risk-Neutral Probability in the Binomial Option Pricing Model*. Rochester, NY, Apr. 24, 2014. DOI: 10.2139/ssrn.2428763. URL: https://papers.ssrn.com/abstract=2428763 (visited on 03/13/2023).

[22] Alessandro Penati and George Pennacchi. *Finance 400 - Spring 2009*. The Cox-Ross-Rubinstein Option Pricing Model. URL: http://home.cerge-ei.cz/petrz/fm/f400n10.pdf (visited on 03/14/2023).

[23] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. *On the Convergence of Adam and Beyond*. Number: arXiv:1904.09237. Apr. 19, 2019. arXiv: 1904.09237[cs, math,stat]. URL: http://arxiv.org/abs/1904.09237 (visited on 03/13/2023).

[24] Fabrice Douglas Rouah. *Mathematical Finance Notes*. Four Derivations of the Black-Scholes Formula. URL: http://www.frouah.com/pages/finmath.html (visited on 03/09/2023).

[25] *S&P 500*. S&P Dow Jones Indices. URL: https://www.spglobal.com/spdji/en/indices/equity/sp-500/ (visited on 03/14/2023).

[26] Scott Sheffield. *18.600 Probability and Random Variables - Spring 2021*. Martingales, risk neutral probability, and Black-Scholes option pricing. URL: https://math.mit.edu/~sheffield/2019600/martingalenotes.pdf (visited on 03/12/2023).

[27] Paul Wilmott, Sam Howison, and Jeff Dewynne. *The Mathematics of Financial Derivatives: A Student Introduction*. Cambridge: Cambridge University Press, 1995. ISBN: 978-0-521-49789-3. DOI: 10.1017/CBO9780511812545. URL: https://www.cambridge.org/core/books/mathematics-of-financial-derivatives/7121345D07C5BCE4FBEC91A8A7E6F267 (visited on 03/08/2023).

[28] *Yahoo Finance – stock market live, quotes, business & finance news*. URL: https://uk.finance.yahoo.com/ (visited on 03/14/2023).