A comparison between ANNs and traditional stock market forecasting techniques

Alexandros Antoniou

Centre for Computational Finance and Economic Agents

CSEE

University of Essex

Submitted in partial satisfaction of the requirements for the
Degree of Master of Science
in
Computational Finance

Supervisor Dr Maria Kyropoulou
Second Supervisor Dr John O'Hara

August 2020

Acknowledgements

I would like to thank my supervisor, Dr Maria Kyropoulou, for helping me with this thesis. I would also like to thank my friends and family for keeping me sane during this lock-down and overall worldwide panic. It has not been a fun time.

Abstract

TODO

Keywords: Deep Learning, Artificial Neural Networks, Stock Market, Time Series prediction, Neural Networks

Table of Contents

1 Introduction **1**

2 Overview of Neural Networks **2**

2.1 Neurons 2

2.2 Activation Functions 3

2.2.1 Rectified Linear Unit 3

2.2.2 Sigmoid 4

2.2.3 Hyperbolic tangent 4

2.3 Backpropagation 5

2.4 ARIMA 6

2.5 Testing some math 6

2.6 Testing citations 6

3 Autoregressive models **7**

4 Methodology **8**

4.1 Time-Series Forecasting 8

4.2 ARIMA 8

4.3 Model Description 8

4.4 Evaluation 10

5 Results **12**

5.1 Dataset 12

5.2 Evaluation Metrics 12

5.3 Results 12

6 Conclusions **13**

1 Introduction

The ability to predict changes in stock prices is extremely important to the financial world as it influences trading strategies and reduces risks in the market. Forecasting has long been a problem for the business and technology communities and has seen little advances until quite recently, with the advent of neural networks and deep learning.

Before artificial neural networks, the finance world used other methods to model the time series that arose from the continuous updating of stock prices. Models like the autoregressive integrated moving average model (ARIMA) and the generalised autoregressive conditional heteroscedasticity model (GARCH) are key econometric methods for forecasting time series and are still widely used in finance.

The focus of this project is to provide a comparison between the traditional methods for time series forecasting mainly the ARIMA model, and simple implementations of artificial neural networks, in the context of financial time series prediction.

Forecasting stock prices with moving averages belongs to the technical analysis category of financial analysis. Kirkpatrick and Dahlquist define *technical analysis* as the study of prices in freely traded markets with the intent of making profitable trading or investment decisions[8], hence the models will only use daily prices for the selected stocks, ignoring company financial data.

2 Overview of Neural Networks

In this section, we provide a brief history of neural networks in the context of finance and time series prediction, as well as a more detailed description of the architecture used in the paper.

Neural networks are systems largely belonging to the study of Machine Learning, typically associated with solving computational problems that other models and algorithms struggle to. Loosely based on biological neural networks, like the brain, artificial neural networks are collections of neuron-like nodes and links that connect them. Their resemblance to the brain in terms of architecture is part of what allowed neural networks to grow in computing power and popularity over the past decades, from their emergence in the 1940s[9][11].

2.1 Neurons

Neurons are the artificial equivalent of a biological neuron and are the fundamental components of a neural network. Neurons perform three tasks, receive input from other neurons, apply an activation function to the input and output a value to other neurons. Neurons in the network are connected and each of these connections carries a signal and has certain weight attached to it, which affects the signal carried[16][14]. Graphically, a neuron could be represented in figure 2.1.

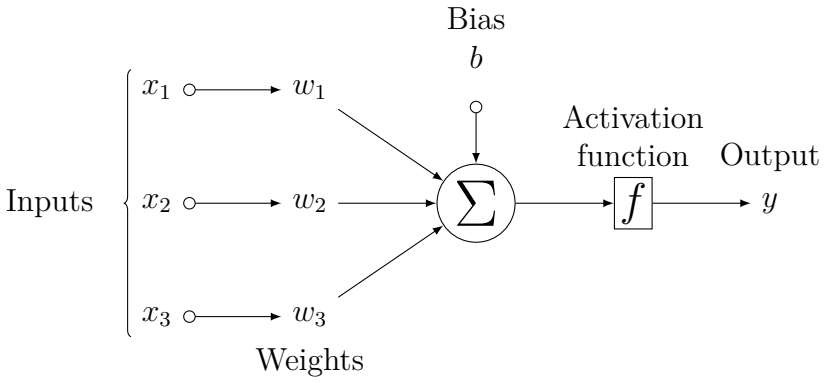

Figure 2.1: Graphical representation of a Neuron

2.2 Activation Functions

Wilson (2009) defines the activation function of a neural network as the function that governs the output behaviour of a neuron, given a set of input values. There are several types of activation functions, the simplest being the step function.

In mathematical terms, the step function, or the unit step function, is defined as

$$H(x) := \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases} \tag{2.1}$$

where $H(x)$ is the Heaviside step function[1], and at value 0, an output of $H(0) = 1$ is selected and passed down to the next layer of neurons.

2.2.1 Rectified Linear Unit

The Rectified Linear Unit (ReLU) is an activation function mathematically defined as

$$f(x) = [x]^+ = \max(0, x) \tag{2.2}$$

where x is the neuron's input value. Plotted on cartesian axes, the graph shows a linear relationship for $f(x)$ and x where $x > 0$. Generating a plot of ReLU further explains its activation range.

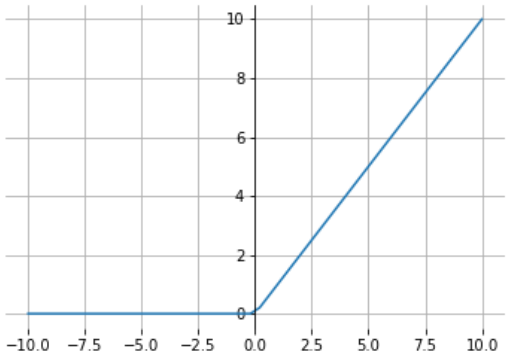

Figure 2.2: Plot of the Rectified Linear Unit function

ReLU was first introduced in the 2000 and 2001 Hahnloser papers[7]. ReLU is currently the most widely used activation function[12] and has found great success in training deep neural networks primarily in the fields of computer vision[5] and speech recognition[10].

2.2.2 Sigmoid

The Sigmoid function is a logistic activation function, mathematically defined as

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{2.3}$$

where

L is the maximum point of the curve,

k is the steepness of the curve,

x_0 is the value of the midpoint of the curve.

Sigmoid functions are non-linear, differentiable, defined for all real input values and have an output greater than zero for all inputs, in contrast to the ReLU activation function whose output is greater than zero only with positive inputs. A well known example of a sigmoidal curve is the cumulative distribution function of the normal distribution.

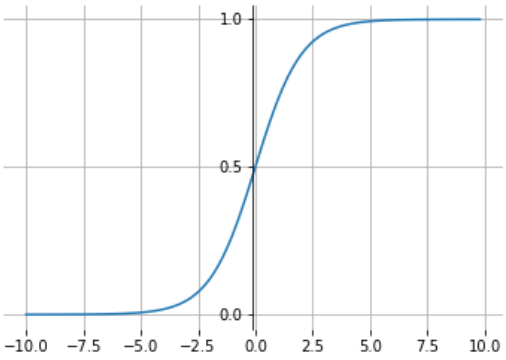

Figure 2.3: Plot of the sigmoid function

2.2.3 Hyperbolic tangent

The hyperbolic tangent is a specialised case of the sigmoid function, mathematically defined as

$$f(x) = \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.4}$$

Unlike the standard sigmoid function, tanh can output negative values for negative inputs giving it double the range that the standard sigmoid has.

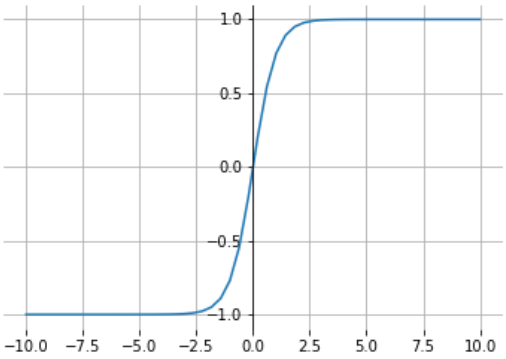

Figure 2.4: Plot of the hyperbolic tangent function

2.3 Backpropagation

The backpropagation algorithm is a key feature of neural networks as it allowed for fast and efficient training of multi-layered networks by distributing error values back through the layers of the network, adjusting the weights in the connections between neurons respectively[15]. In more detail, backpropagation functions compute the gradient $\nabla_x f(\mathbf{x}, \mathbf{y})$ numerically in a simple and inexpensive procedure[13]. What computing the gradient really means is calculating the derivative of the loss function in the model, with the loss function being a selected error function.

The chain rule is a way to calculate the derivate of functions by decomposing a function to other functions whose derivative is known. Let x be a real number and f and g be functions that map from $\mathbb{R}$ to $\mathbb{R}$. By generalising the chain rule of calculus

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{2.5}$$

beyond the scalar case into vector notation, such that

$$\nabla_x z = (\frac{\delta y}{\delta x})^T \nabla_y z \tag{2.6}$$

where $\frac{\delta y}{\delta x}$ is a Jacobian matrix of g. The backpropagation algorithm is essentially a fast computation of this Jacobian gradient for each connection and node in the network[6].

2.4 ARIMA

The prices of stocks can be modelled as non-linear time series, which have been at the centre of attention in the finance world since the 1970s with George Box and Gwilym Jenkins popularised their Box-Jenkins method for finding the best-fit of a time series model[4].

2.5 Testing some math

Here are two equations:

And here is some text with some nice inline math, (x, y) wow γ so cool ρ.

2.6 Testing citations

This is Fama[3] and this is Goodfellow. This is another GAN citation.

3 Autoregressive models

4 Methodology

The following section provides details in the construction of the model for predicting stock prices, as well as a description of the ARIMA model to which we compare the network.

4.1 Time-Series Forecasting

Financial data are discrete in time and as such can be modelled as time-series with calculated means and standard deviations.

* time series analysis

* exploratory analysis

4.2 ARIMA

4.3 Model Description

The network is a relatively simple network by most accounts, comprised of a single hidden layer. The simplicity of the model only goes to show the power of neural networks in fitting and forecasting time-series. The model consists of three layers, an input layer, a Long Short-Term Memory (LSTM) layer and the output layer, as shown in 4.1.

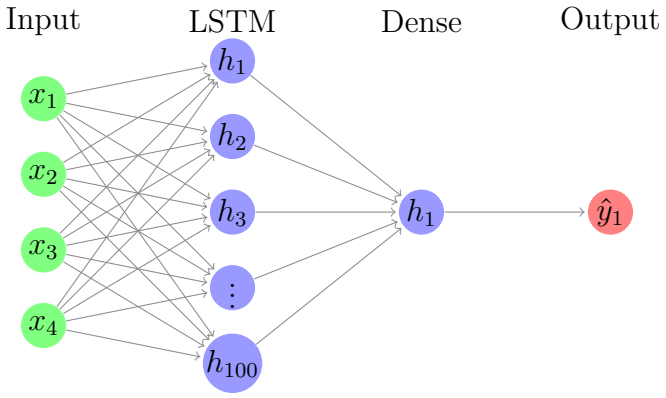

Figure 4.1: Model Architecture

The input layer receives a 2-dimensional array of historical stock values, including the previous day's opening, highest, lowest and closing stock prices. The network then allows the

neurons to compete amongst each other and determing an appropriate output. Output is in the shape of a 1-dimensional array containing four values, the future day's predicted stock values for open, high, low and close. The LSTM layer contains 100 nodes and is trained for 100 epochs. The Rectifier Linear Unit was used as its activation function and the Mean Absolute Error was used as its loss function.

Nodes	Epochs	Optimizer	Learning Rate	Activation	Loss
100	100	Adam	0.001	ReLU	MAE

Table 4.1: Description of the hidden LSTM layer

In this project we made use of the Keras API to implement the network. Keras is written in Python and operates ontop of TensorFlow, which is an extremely popular and widely used Deep Learning library[2]. A simple sequential model using LSTM cells built using Keras would look like this:

```
1  model = Sequential()
2  model.add(LSTM(100, activation="relu", input_shape=(n_steps, n_features)))
3  model.add(Dense(n_features))
4  opt = Adam(learning_rate=0.001)
5  model.compile(optimizer=opt, loss="mae", metrics=["mse"])
```

With as little as 5 lines of code, we have a working Long Short-Term Memory model ready for training. The LSTM cell easily remembers the long term dependencies in the data and outputs a 1-dimensional array containing future values. Figure 4.2 contains a more detailed breakdown of the layers used in the network.

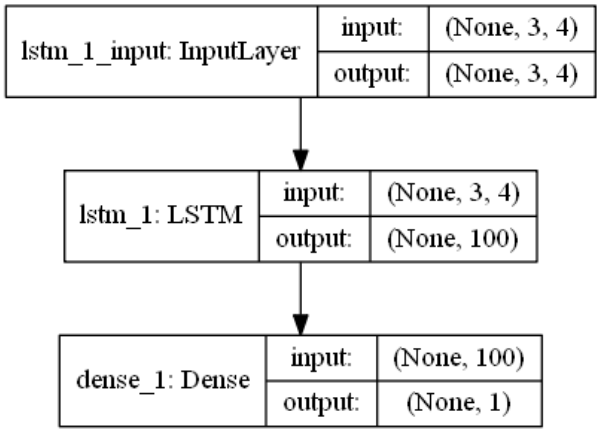

Figure 4.2: Description of layers

A Dense layer is a fully-connected feedforward layer. Feedforward layers are the simplest type of architecture employed in machine learning, which is essentially a collection of neurons that pass signals in one direction only, the layer in front of them. These layers do not form cycles in their connections which is fundamentally different from Recurrent Neural Networks like the LSTM. The purpose of the Dense layer in the model is to reduce the dimensionality of the data from the 100-dimensional space created in the LSTM down to a 1-dimensional array of values, which is then lead to the output layer. The Dense layer performs no operations on the data, it's activation function is linear.

4.4 Evaluation

To perform an evaluation of the model, we feed into it the testing set gathered from the dataset. The model uses the Mean Absolute Error metric to guide the minimisation of its loss function during training. Mathematically, the Mean Absolute Error is defined as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{4.1}$$

and is the sum of the absolute differences of predicted data points to actual data points divided by the number of data points in the set. MAE is also used in the evaluation of the compiled and trained model.

Evaluation of the model also includes a comparison of Mean Squared Error values. MSE is mathematically defined as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{4.2}$$

and is the mean of the squared differences between the predicted values and the actual data points. MAE was chosen for the training of the model because of its linear relationship between penalties and errors, in that it treats a predicted to actual difference of 1 in a way proportionally to how it treats a predicted to actual difference of 5. MSE treats larger differences between predictions and actual data non-linearly.

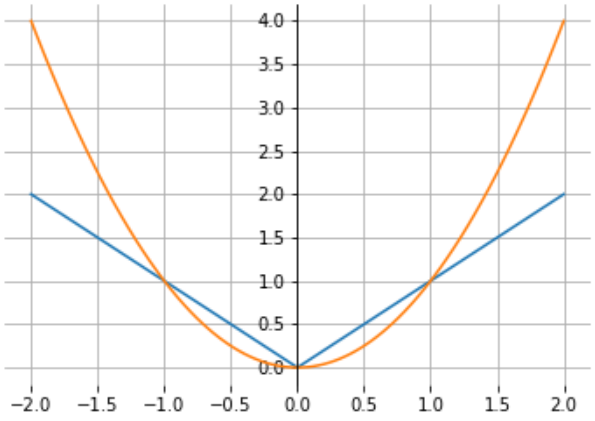

Figure 4.3: Comparison of MAE (blue) and MSE (orange). Note how MAE scales linearly with higher error values while MSE scales quadratically, treating higher errors more harshly.

5 Results

5.1 Dataset

We retrieve data on tickers available at `https://www.tiingo.com/`. The stock exchanges targeted in this project are the New York Stock Exchange (NYSE) and the National Association of Securities Dealers Automated Quotations (NASDAQ). The data comprise of companies from the technology and the financial services industries.

The data consist of daily values for the period starting January 1, 2016 and ending December 31, 2019. The dataset includes daily information for the `close`, `open`, `high` and `low` prices of each trading day. Below is a table listing all stocks considered for the project:

Stock Name	**Symbol**	**Stock Exchange**
Apple Inc.	AAPL	NASDAQ
Amazon Inc.	AMZN	NASDAQ
American Express Company	AXP	NYSE
Boeing Co	BA	NYSE
Bank of America Corp	BAC	NYSE
Citigroup Inc	C	NYSE
Ford Motor Company	F	NYSE
Facebook Inc.	FB	NASDAQ
General Electric Company	GE	NYSE
Alphabet Inc. Class C	GOOG	NASDAQ
Goldman Sachs Group Inc.	GS	NYSE
JPMorgan Chase & Co.	JPM	NYSE
Morgan Stanley	MS	NYSE
Microsoft Corporation	MSFT	NASDAQ
Wells Fargo & Co.	WFC	NYSE

Table 5.1: Stocks included in the study

5.2 Evaluation Metrics

5.3 Results

6 Conclusions

...

References

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematican Functions with Formulas, Graphs and Mathematical Tables*. National Bureau of Standards, 1972 (cit. on p. 3).

[2] F. Chollet *et al.* (2015). 'Keras,' [Online]. Available: `https://github.com/fchollet/keras` (cit. on p. 9).

[3] E. Fama, 'Efficient capital markets: A review of theory and empirical work,' *Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970 (cit. on p. 6).

[4] G. J. George Box, 'Time series analysis, Forecasting and control,' 1970 (cit. on p. 6).

[5] X. Glorot, A. Bordes and Y. Bengio, 'Deep sparse rectifier neural networks,' *Artificial Intelligence and Statistics (AISTATS*, vol. 2011, 2011 (cit. on p. 4).

[6] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org` (cit. on p. 6).

[7] R. H. R. Hahnloser and H. S. Seung, 'Permitted and forbidden sets in symmetric threshold-linear networks,' *NIPS*, 2001 (cit. on p. 4).

[8] C. D. Kirkpatrick and J. R. Dahlquist, *Technical Analysis: The Complete Resource for Financial Market Technicians*. 2006, ISBN: 978-0-13-153113-0 (cit. on p. 1).

[9] S. Kleene, 'Representation of events in nerve nets and finite automata,' *Annals of Mathematics Studies*, pp. 3–41, 1956 (cit. on p. 2).

[10] A. L. Maas, A. Y. Hannun and A. Y. Ng, 'Rectifier nonlinearities improve neural network acoustic models,' *Stanford Press*, 2014 (cit. on p. 4).

[11] W. McCulloch and W. Pitts, 'A logical calculus of ideas immanent in nervous activity,' *A Logical Calculus of Ideas Immanent in Nervous Activity*, vol. 5, no. 4, 1943. DOI: `10.1007/BF02478259` (cit. on p. 2).

[12] R. Prajit and B. Zoph, 'Searching for activation functions,' 2017 (cit. on p. 4).

[13] D. Rumelhart, G. Hinton and R. Williams, 'Learning representations by back-propagating errors,' *Nature*, vol. 323, pp. 533–536, 1986. DOI: `10.1038/323533a0` (cit. on p. 5).

[14] S. Russel and P. Norvig, *Artificial Intelligence, A Modern Approach*, Third. Pearson, 2010 (cit. on p. 2).

[15] P. J. Werbos, *Beyond Regression: New Tools for Predictions and Analysis in the Behavioural Sciences*. Harvard University Press, 1975 (cit. on p. 5).

[16] W. H. Wilson, *The Machine Learning Dictionary*. Saint Elizabeth University, 2009 (cit. on pp. 2, 3).

List of Tables

4.1 Description of the hidden LSTM layer 9

5.1 Stocks included in the study 12

List of Figures

2.1 Graphical representation of a Neuron 2
2.2 Plot of the Rectified Linear Unit function 3
2.3 Plot of the sigmoid function 4
2.4 Plot of the hyperbolic tangent function 5

4.1 Model Architecture 8
4.2 Description of layers 10
4.3 Comparison of MAE (blue) and MSE (orange). Note how MAE scales linearly with higher error values while MSE scales quadratically, treating higher errors more harshly. 11