AUTOMATED STOCK TRADING USING NEURAL NETWORKS

MATTHEW KNOWLES

1. Mathematical Background I: Activation

We first look at the mathematical background behind the neural network. Assume we have k hidden layers, sandwiched by an input layer and an output layer. Since the aim of neural networks is to model how a brain processes information, we often refer to the connections between nodes in each layer as "synapses". Theses synapses are characterised by their weights and biases. Let $W^{(l)}$ be the matrix containing the weights between layer l and $l-1$. If layer l has n nodes, and $l-1$ has m nodes, then we have:

$$W^{(l)} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{bmatrix} \tag{1}$$

Which gives the weights of the synapses between layer l and $l-1$. Combining this with a vector of biases for the same inter-layer, $b^{(l)}$, we are almost able to calculate the values of the nodes in layer l. However before this can be done we need an activation function. In a lot of cases, as was found in [1], the choice of activation function doesn't really matter. The most common choice is the "sigmoid function" defined in equation 2.

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2}$$

Finally, piecing this all together, for the layer l, we get the values of the nodes in the layer using the matrix equation 3.

$$a^{(l)} = \sigma(W^{(1)}a^{(l-1)} + b^{l}) \tag{3}$$

Where $a^{(l-1)}$ is the activation values for the previous layer, and $\sigma()$ is applied to all values in the resulting vector.

2. Mathematical Background II: Backpropogation

The values in the network are initially random. The way the network learns is by an algorithm called "backpropogation".The idea is to define an error function:

$$E(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i' - y_i)^2.$$

The parameter θ incorperates all the weights and biases, while X is a vector of input values. The y_i' term is the approximated value by the network, and y_i is the actual value for the X vector.

Given the number of parameters in this equation, finding minima is not as simple as simply taking a derivative. The method of minimising this function is done using "gradient descent". As discussed in [2], there are 3 varients of this algorithm. For a learning rate α, we have the following variants.

Date: Autumn Term 2021.

- Batch Gradient Descent: $\theta = \theta - \alpha \nabla_\theta E(X, \theta)$
- Mini-batch Gradient Descent (n training examples): $\theta = \theta - \alpha \nabla_\theta E(x^{(i:i+n)}; y^{(i:i+n)}; \theta)$
- Stochastic Gradient Descent (SGD): $\theta = \theta - \alpha \nabla_\theta E(x^{(i)}; y^{(i)}; \theta)$

SGD compares to Mini-batch by performing parameter updates for each training example. It is noted that mini-batch is the most common algorithm used in training neural networks. One key issue is chosing the optimal learning rate α. Too low, and convergence will take too long, too big and convergence won't be achieved.
Finally, for the k^{th} layer, we update the weight between node i to j in layer $k+1$ via:

$$\Delta w_{ij}^k = -\alpha \frac{\partial E(X, \theta)}{\partial w_{ij}^k}. \tag{4}$$

3. Why use Neural Networks in Stock Trading?

Historically, the use of financial ratios, the Efficient Market Hypothesis and The Capital Asset Pricing Model were key methods for forecasting the price of financial assets. There are many financial ratios and characteristics, too many for a human to look at and identify underlying patterns.

This is where Neural Networks come in, they can identify important information that may go undetected by the human analyst [3]. The immense amount of financial data available to the modern investor means NNs can be trained very effectively, and again pick up on relatioships between financial data that would be missed by the human investor.

However, the issue here is that the networks cannot *explain* why they make the decisions they do. This is fine if the model is making money, however, if the model starts to malfunction, it can be hard to debug it and find out why. This is why so much "backtesting" is necessary- the practice of running a model on unseen past data to see how it performs, before giving it real money.

4. How do we use Neural Networks in Stock Trading?

4.1. **Buy-Hold-Sell Signalling.** We use the paper by Kimoto, [4] to examine this method. The idea can be visualised in figure **??**.

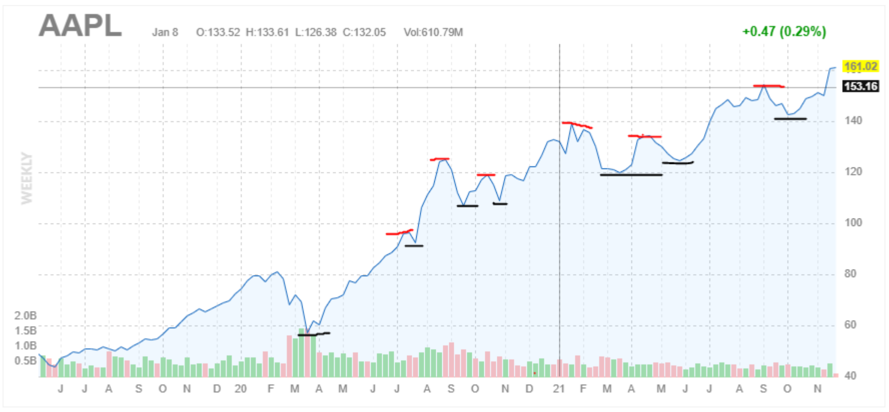

Figure 1. $AAPL Stock with ideal buy (black) and sell (red) indicators

The technique is to train a neural network to predict when it should buy or sell based on past behaviour of the stock. This can lead to continuing profit over a training window. The model in [4] uses a 3-layer network, with only one hidden layer. The input of the network takes a vector curve- showing how the stock has moved in the past, the turnover, domestic interest rate, foreign exchange rate, the New York Dow-Jones average and any other metrics at the investors will.

The authors found this method to be more profitable than simply buying and holding the stock.

4.2. **Value Prediction.** As opposed to the previous method for directly making profit by letting the model itself trade on a stock market, this method is used for trying to predict the price of a stock at some future time, thus allowing an investor to either buy and hold, or go short on the stock, making a profit if the stock price falls.

In one study, [5], the authors compare a neural network model to an ARIMA (Auto-Regregressive Integrated Moving Average) model for price prediction. For the experiment, they ran different network configurations, with a different range of iterations for the training. The authors found that while both gave good forecast results, but the neural network did outperform the ARIMA model overall.
This method is suitable for informing investment decisions, however not so much for autnomously trading.

5. Improvements to Automated Trading

Neural networks have to be trained on a portfolio of assets. But selecting the optimal portfolio to use trading methods on is a problem in itself. In [6], the authors used a total of six heuristic and metaheuristic techniques in finding an optimal portfolio. They found a Sawtooth Genetic Algorithm is the fastest computationally, and therefore provides a more efficient improvement over a standard genetic algorithm.

One of the main ways of improvement in the past is using global search algorithms, focussing on improving the values of the weights in the neural network. Or similarly, imprving the algorithms for training the networks. One big improvement, relevant to this module in particular, is [7]. In this paper, the authors use a genetic algorithm to search for the optimal set of weights in the netowrk. The authors compare three models for determining weights in the network. The genetic algorithm approach they use for reducing the dimensionality of data has a $10 - 11\%$ higher prediction accuracy. Reducing the dimensionality is a great way to make improvements, since market data is very complex and dense, so searching it for key features using conventional methods is not an easy task.

Combining the evolutionary technique for portfolio selection with the neural network method for automatic trading could lead to a very powerful investment stratergy.

6. Conclusion

Modern quantitative finance funds, for obvious reasons, do not disclose the methods they use in trading. However, given the extensive amount of research being done on applying biologically-inspired techniques, such as neural networks and evolutionary algorithms to financial problmes, it is safe to assume some of the methods discussed in this report are being used to make large sums of money on the worlds stock markets. The papers that we have looked at in this report have shown how using neural networks in trading has outperformed other, more traditional statistical techniques. By employing these models, investors can continually make profits that outperform the market, at the expense of the technical knowledge required to employ them.

As more research goes into improving the efficiency and predictive power of these techniques, they will only continue to help investors make bigger and more consistent profits. However, the stock market also allows a great playground for building models that can be used in other fields. This could, for example, be applied to predicting rainfall based on past time-series data, or how populations of animals will change over time.

References

[1] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," *Journal of theoretical and applied information technology*, vol. 47, no. 3, pp. 1264–1268, 2013.

[2] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[3] P. R. Burrell and B. O. Folarin, "The impact of neural networks in finance," *Neural Computing & Applications*, vol. 6, no. 4, pp. 193–200, 1997.

[4] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka, "Stock market prediction system with modular neural networks," in *1990 IJCNN international joint conference on neural networks*, pp. 1–6, IEEE, 1990.

[5] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of arima and artificial neural networks models for stock price prediction," *Journal of Applied Mathematics*, vol. 2014, 2014.

[6] A. Adewumi and A. Moodley, "Comparative results of heuristics for portfolio selection problem," in *2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pp. 1–6, IEEE, 2012.

[7] K.-j. Kim and I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index," *Expert systems with Applications*, vol. 19, no. 2, pp. 125–132, 2000.