Forecasting Market Volatility Using Artificial Neural Networks

Artur Sak

Abstract

In recent years, stock market forecasting has been extensively studied in applied finance literature as well as in industry. The theory behind this research dictates that there are repeating patterns in the historical prices of stocks that can be identified and used to forecast future price fluctuations. Accurate prediction of future trends can allow investors, and other market actors, to adjust their behavior in the current period and potentially change expected future outcomes. In this paper, I explore the application of Artificial Neural Networks (ANNs) for forecasting financial market volatility, and test the robustness of the model vis-à-vis prediction.

Introduction

At a very basic theoretical level, investors attempt to assemble portfolios which maximize expected return for a given level of risk. Such is the basis, for the mathematical framework for assembling portfolio assets, which was first introduced by Economist Harry Markowitz in 1952. [1] While, maximizing expected return and minimizing the risk of a portfolio may be at the core of each investor's decision, the actual construction of his portfolio depends on his subjective risk tolerance. As such, changes in market volatility evoke varying responses from market participants. Some perceive these changes as profit opportunities, and attempt to capitalize; while others may see them as threats, and begin to divest to mitigate the risk of incurring losses.[2]

Volatility, with respect to financial instruments, can be divided into two categories, market and firm-specific volatility. The former reflects the level of investor uncertainty, which may be attributed to various internal and/or external macroeconomic factors. The latter, pertains to individual stocks and reflects perceived growth prospects for a company or sector *(Chadhuri & Ghosh 2015)*[1]. There have been a multitude of econometric models developed to measure these volatilities; specifically, the Single-Index Model attempts to capture the contribution of each to the volatility of a market index. Many of these models, however, impose a linearity onto the relationship, and therefore add a level of complexity to analysis when dealing time series data.

It is for this reason, the purpose of this paper is to explore a framework for forecasting volatility that does not presuppose any linearity in the input-output relationship and furthermore, allows for interaction and feedback between the input data. The organization is as follows: Section 2 discusses the Artificial Neural Network architecture used in this experiment, Section 3 outlines the learning algorithm, input data set, and pre-processing, and Section 4 shows the prediction results and reports the robustness of the model in predicting actual stock price fluctuations.

[1] This framework has since been dubbed, Modern Portfolio Theory (MPT)

[2] For a detailed discussion on investor decision-making see: *"Portfolio Selection" (Markowitz 1952)*[3].

Artificial Neural Network Architecture

Many time series, particularly financial time series, contain an unknown auto-regressive structure, where future realizations depend on past information. Formally, we can say that stock prices are a martingale (i.e. $E[P_t|\mathcal{F}_{t-1}] = P_{t-1}$ for all t; where $\mathcal{F}_t$ is an information set of a stochastic process up until time t).

In financial economics, this relationship has been captured by the predominant theory on asset prices, the Efficient-market Hypothesis (EMH). Developed by economist Eugene Fama in 1970, the EMH states that asset prices fully reflect all available information *(Fama 1970)*[2]. Following the EMH to it's logical end, means that it is impossible to make economic profits by predicting the market based on an information set $\mathcal{F}_t$.[3]

Using the EMH as a foundation; training a neural network on sliding windows of past prices can be used to extract an estimation of the auto-regressive structure hidden in realizations of the time series and then used to predict future price movements *(Giacomini 2003)[4]*. [4]

In this experiment, the network was organized into a feed-forward architecture called a Multi-Layer Perceptron (MLP), with an input layer consisting of 5 nodes, two "hidden" layers, consisting of 100 and 50 nodes respectively, and an output layer consisting of one node. In a MLP network, every node in a layer shares a weighted connection (edge) with every node in the next layer.

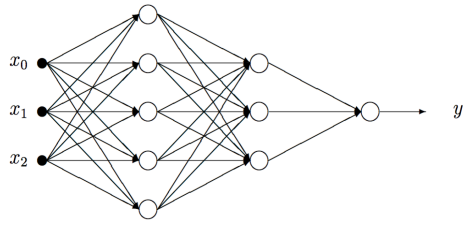

Figure 1: A Simple (2,5,3,1) MLP

[3]A discussion on the extent to which the EMH holds true is beyond the scope of this paper, however there is a significant amount of literature that can be explored on the topic.

[4]In a way, this can be seen as an extension of auto-regressive time series modeling.

Error Backpropagation

The Chicago Board Options Exchange (CBOE) Volatility Index (^VIX) is a popular measure of the implied volatility of S&P 500 options. Theoretically, it is calculated as a weighted blend of prices for options on the S&P 500 index, and while it may not be entirely accurate, data from this index can be introduced to the experimental framework to forecast market volatility.

Let $\{P_t\}_{t=1}^{N}$ represent the time series of VIX prices and $P_{t+1} = \mathbf{\Phi}_{MLP}(\cdot)$ represent the (5-100-50-1) MLP, where P_{t+1} is the predicted price in period $t+1$ that is given as output. Therefore the vector,

$$\boldsymbol{P}_t = (P_t, P_{t-1}, ..., P_{t-4}) \tag{1}$$

represents a sliding window of five days of closing prices. To increase the input data density and avoid the *curse of dimensionality* [5] the price window is normalized according to the formula,

$$a_t = \frac{P_{t-i} - P_{t-i,min}}{P_{t-i,max} - P_{t-i,min}}(h_i - l_i) + l_i \quad (i \in [0, 4]) \tag{2}$$

where P_{t-i} represents the closing price at $t-i$, $a_{t-i,max} = \max(\boldsymbol{P}_{t-i})$, $P_{t-i,min} = \min(\boldsymbol{P}_{t-i})$ and h_i and l_i are the upper and lower bounds of the normalizing interval, in this case $(-1.0, 1.0)$, respectively[6]. Therefore, denote the normalized vector as $\boldsymbol{a}_t$.

Each hidden layer node, as well as the output node, contain an input value given as,

$$n_i = \sum_j w_{ij} a_j = \boldsymbol{w}^T \boldsymbol{a} \tag{3}$$

where $\boldsymbol{w}$ is a vector of incoming connection weights (initially random) between the current and preceding layer nodes and $\boldsymbol{a}$ is a vector of preceding layer node activation values. The activation values in each layer (besides the input layer) are calculated based on a logistical (or "sigmoid") function,

$$a_i = \frac{1}{1 + e^{-n_i}} \tag{4}$$

[5] "The aim of linear scaling is to independently normalize each feature component to the specified choice. It ensures the larger value input attributes do not overwhelm smaller value inputs, which helps to reduce prediction errors" *(Masoud 2013)[5].*

[6] $a_t > 0$ represents buy signal $a_t < 0$ represents a sell signal

where, n_i is the input value from equation (3).

The training algorithm thus works as follows. First, the network is fed a window of normalized prices ($\boldsymbol{\Phi}_{MLP}(\boldsymbol{a}_t)$), which sets the input layer activation values to the normalized prices, and moves the values forward by updating each proceeding layer (hence the "feed-forward architecture"). Once the output layer is reached an error is calculated based on the difference between the actual price (d_i) at time $t+1$ and the predicted activation value (a_i) multiplied by the derivative of the logistic function.

$$e_i = \begin{cases} (d_i - a_i) \cdot a_i(1 - a_i) & \text{for output layer} \\ \sum_j w_{ji} e_j = \boldsymbol{w}^T \boldsymbol{e} \cdot a_i(1 - a_i) & \text{for hidden layers} \end{cases} \tag{5}$$

Next, the errors are "propagated" backward through the network and the connections accumulate a delta (change) of their weights calculated by,

$$\Delta w_{ij} = \eta e_i a_j \tag{6}$$

where, η is some learning rate between 0 and 1 (in this case set at 0.001).

The network is trained on each five day window of prices over a 20 day period (batch updating) after which, the aggregate Δw's are added to the connection weights (in matrix $\boldsymbol{w}$). Additionally, to prevent the network from getting stuck in local minima, the values in $\Delta \boldsymbol{w}$ are not reset to 0, but instead multiplied by a scalar $\mu \in (0.0, 1.0)$ (for maintaining "momentum"). This entire process is repeated for several iterations (epochs), until the network reaches one of two settling criteria. Either the change in error from one iteration to the next falls below a threshold, in this case chosen to be ≤ 0.001, or the network reaches a maximum limit of 100 epochs.

Finally, the network is tested on the full set of closing prices (over a year), by imposing sliding windows and updating forward, then "un-normalizing" the output price by taking a_t^{-1}(inverting equation 2) and calculating an error to see if the network converges to the correct price.

Results and Conclusions

The results from the experiment are presented below.

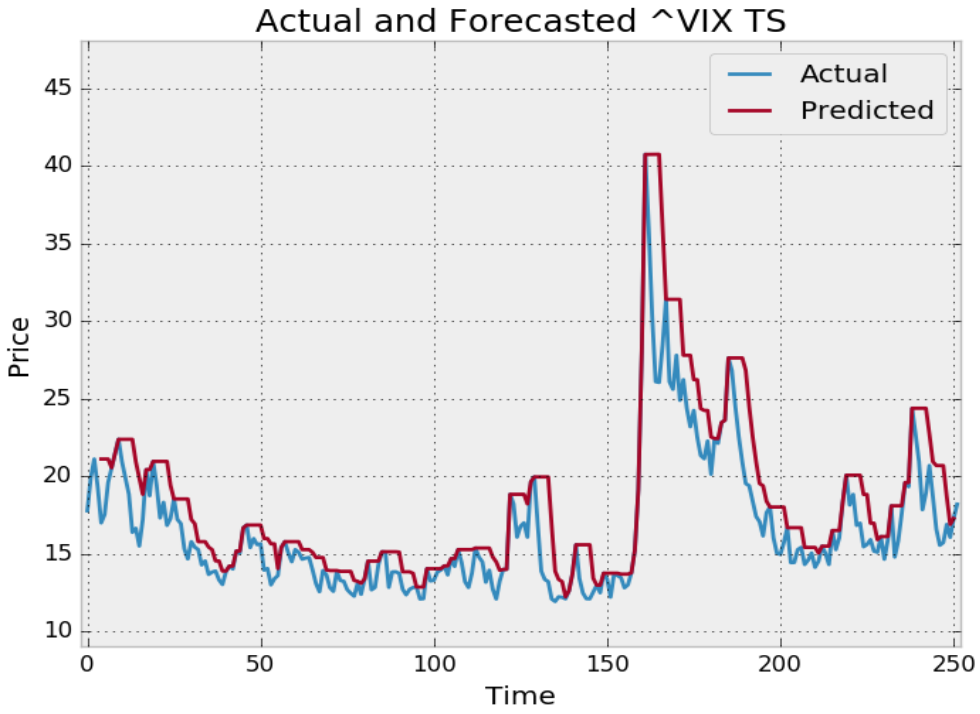

Figure 2: Graph of VIX and MLP predictions

The variables I chose for the experiment were,

- Inputs: normalized price window from $t-4...t$
- Output: Close price at $t+1$
- Percentage of Training data: $\approx 16\%$
- Percentage of Testing data: $\approx 84\%$
- Number of Observations: 252

Table 1

	Max	Mean	Min
Actual	40.740002	0.0	11.95
Predicted	40.7377796667	0.0	12.249955471

Data was gathered on the output values produced by the network as it was tested on sliding windows of ˆVIX prices between January 01, 2015 and January 01, 2016. The convergence values were "un-normalized" and several measures of robustness were calculated,

$$SSR = \sum_{i=0}^{N} (P_{t+1,i} - \hat{P}_{t+1,i})^2 \tag{7}$$

Where $P_{t+1,i}$ and $\hat{P}_{t+1,i}$ are the actual and predicted prices respectively.

$$SST = \sum_{i=0}^{N} (\hat{P}_{t+1,i} - \bar{P}_{t+1,i})^2 \tag{8}$$

Where $\hat{P}_{t+1,i}$ and $\bar{P}_{t+1,i}$ are the predicted and average of the predicted values respectively.

$$R^2 = 1 - \frac{SSR}{SST} \tag{9}$$

And,

$$\rho = \sqrt{R^2} \tag{10}$$

Where ρ measures the correlation of the two series.

$$F = \frac{R^2/k}{(1 - R^2)/(n - k - 1)} \tag{11}$$

Where k represents degrees of freedom.

The results of these calculations for the data produced by the MLP are shown in Table 2 below.

Table 2

SSR	3280.11236596
SST	90517.5551668
R^2	0.963762693768
ρ	0.981714160929

This means that calculating an *F-statistic* with $k = 5$ degrees of freedom and 252 observations is ≈ 1306.1719 which is significant at the $\alpha = 5\%$. At first glance it may appear that the predicted measure is a very good fit, however the test statistics may be inaccurate due to auto-correlation. A more robust treatment of the forecasted data is needed, not only adjusting for auto-correlation but also estimating the cross-correlation between the two series.

The purpose of this paper was to explore a non-linear framework for forecasting volatility in the stock market using artificial neural networks and test the robustness thereof. The network seemed to correctly categorize ˆVIX fluctuations between January 01, 2015 and January 01, 2016 after training on a small set of recent data. However, a more rigorous treatment of the test results and/or an alternate configuration to the parameters of the MLP may be necessary to gauge the full extent of the architecture's forecasting accuracy, and must therefore be reserved for future study [7].

[7]The full source code used to generate the results of this experiment can be found here: `https://github.com/artur-sak13/Neural_Networks/tree/master/Stock_Prediction`

Bibliography

[1] Dattachaudhuri, T., & Ghosh, I. (2015). Forecasting Volatility in Indian Stock Market using Artificial Neural Network with Multiple Inputs and Outputs. *International Journal of Computer Applications IJCA, 120*(8), 7-15. doi:10.5120/21245-4034

[2] Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance, 25*(2), 383. doi:10.2307/2325486

[3] Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance, 7*(1), 77. doi:10.2307/2975974

[4] Giacomini, E. (2003). Neural Networks in Quantitative Finance. 0-66.

[5] Masoud, N. (2014). Predicting Direction of Stock Prices Index Movement Using Artificial Neural Networks: The Case of Libyan Financial Market. *British Journal of Economics, Management & Trade BJEMT, 4*(4), 597-619. doi:10.9734/bjemt/2014/5519