Market Prediction using Neural Networks

Assignment Report

January 4th, 2016

Roland Bogosi

Table of Contents

1 Introduction

The purpose of this classroom assignment is to use neural networks in order to predict stock market or currency fluctuations. There are many ways to tackle this time-series prediction problem, each of them having their own pros and cons:

Index Prediction – Try to teach the individual values of a series. This way the neural network will receive an input which in some form indicates the day we would like to evaluate it/predict for and as the output of the network, we get the prediction.

Indicator Prediction – Transform the dataset into indicators used during stock analysis, and teach the neural network the pattern of these. This way, the neural network will not be able to predict the index itself, but it should be able to predict the trend within the indicator that is used to make the actual decision if a stock should be purchased or not.

In both cases, as described in [1], one of the main challenges was to decide what inputs and outputs should the neural network be trained for. My initial thought was to feed the date as input, and get the predicted index as the output. However, upon experimentation I learned that this model would not work, and after further researching, the winner method seemed to be having one or more input variables which are the index values of the previous days.

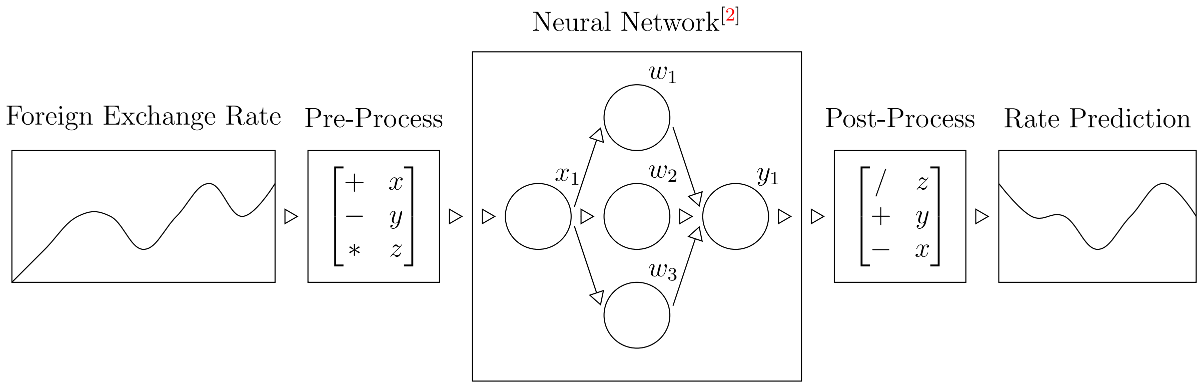
Neural Network[2]
Foreign Exchange Rate
Pre-Process
$\begin{bmatrix} + & x \\ - & y \\ * & z \end{bmatrix}$
w_1
x_1
w_2
y_1
w_3
Post-Process
$\begin{bmatrix} / & z \\ + & y \\ - & x \end{bmatrix}$
Rate Prediction

Figure 1: Basic Architecture of the System

For the purposes of this classroom assignment, I decided focusing on using currency rates, since stock data has way too many outside factors influencing it. I would have had to analyze these outside factors for each stock individually and encode it as an input to the neural network in some form, in order to create an accurate model. For example, the stock of a car manufacturing company might go up when they hold a press release and announce a model, and might suddenly go down if the new model gets a bad review from a reputable source or an issue arises with it, possibly needing call-backs.

2 Network Structure

After examining existing papers in the field of stocks and neural networks, I came upon paper [2] which summarizes a multitude of previous experiments done in this topic, including the neural networks and training sets used with the varying degrees of success.

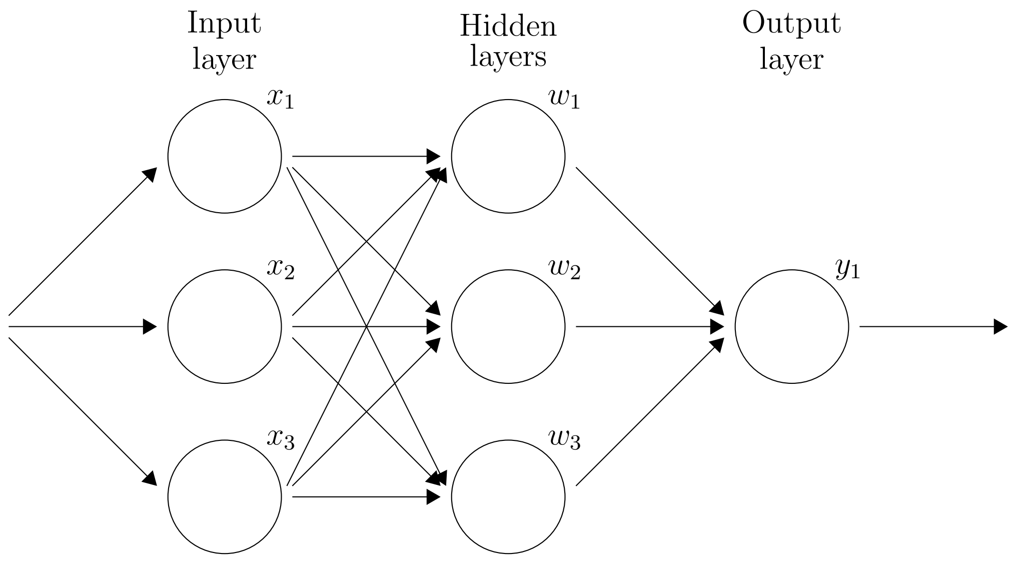

Figure 2: Structure of the Neural Network

The neural network shown in figure 2 is a multilayer perceptron network, whose purpose generally is to map a set of inputs onto a set of outputs, as such its application is popular for solving time-series prediction problems. It has a variable number of inputs, x_1 through x_n, which are the indices of the stock or currency for those days. The output of the network, y_1, is the predicted value for the *next* day.

The number of hidden layers in the network varies based on how many training samples (days) we would like to train the network on, and also on how many days we would like to predict beyond the training samples. While the general rule is to have $x_n \cdot 2$ hidden layers, I evaluated this during several trial and error runs and it did not seem to work well with this application. Even with the momentum learning parameter (m) set to a high value, $0.9 < m < 1$, the neural network was not able to accurately fit the data, it only generated an approximate oscillation.

The neural network is trained with backpropagation, which is a supervised learning method. The network weights are randomly initialized to values between the range of [0.25..0.35], which was determined to yield the best results after a few trial and error runs. The activation function used with the trainer is sigmoid: $f(x) = \frac{1}{e^{-\alpha \cdot x}+1}$

After some experimentation, I found the number of hidden layers in order to accurately fit the data as a result of $1,000$ iterations of the backpropagation teacher algorithm, to be an approximation of the size of training samples. As such, the network configuration will vary for each experiment individually.

3 Usage Intricacies

3.1 Error Calculation

As shown later on, I conducted a multitude of experiments in order to try and find a configuration with the lowest learning and prediction error for each common training size and prediction requirement. However, in order to quantify the error, I had to look up the methods available for determining the accuracy of trend estimation and found a comparison between the common methods in paper [3].

MAPE (*Mean Absolute Percentage Error*) ended up as being the winning method, since it has been used in many other papers, such as [4], which allows me to have a baseline comparison for my results. *MAPE* is defined as:

$$\epsilon = (\frac{1}{n}\sum_{i=1}^{n}\frac{|x_i - x'_i|}{x_i}) \cdot 100$$

where n is the number of predictions, x_i is the actual value, and x'_i is the predicted value.

3.2 Parameter Optimization

In order to find the best possible parameters for any given dataset and any requested prediction size, I set up a script to find the most optimal parameters for the given dataset by training and evaluating neural networks with varying parameters using the `MarketPrediction.NeuralNetwork.TrainAndEval()` API. A snippet of this script can be observed in listing 1.

```
double smallestError = double.MaxValue;

for (int inputLayers = 1; inputLayers < min(10, dataSetSize); inputLayers++) {
    for (int hiddenLayers = 1; hiddenLayers < dataSetSize; hiddenLayers++) {
        // ...additional parameters as needed...

        double error = TrainAndEval(inputLayers, hiddenLayers, ...);

        if (error < smallestError) {
            // save current parameters as best-performing...
            smallestError = error;
        }
    }
}
```

Listing 1: Exhaustive Search for Highest Accuracy

The complexity of finding the most optimal parameters for any given input is:

$$\mathcal{O}(\prod_{i=1}^{P_n}\sum_{j=1}^{P_{i,m}} P_{i,j})$$

where P_n is the number of parameters being optimized for, $P_{i,m}$ is the number of possible options within that parameter, and $P_{i,j}$ is the number of neural networks that need to be trained in order to determine the error accurately. This number is usually 1, however for parameters where there are random factors, this number can be higher, and then the error can be the average of all runs with that parameter.

Since this optimization problem is composed of a multitude of independent loops, it can easily be parallelized by asynchronously running one or more of the loops. As such, it becomes a fast and easy method to find the best configuration for any data set. The error value the search is optimizing for is the *MAPE* value, as described in section 3.1.

4 Experiments

4.1 7-day Prediction for any 30-day EUR/USD Data

In this experiment, I focused my efforts on tuning a neural network to predict as accurately as possible a full week's exchange rates in advance for the currency pair EUR/USD.

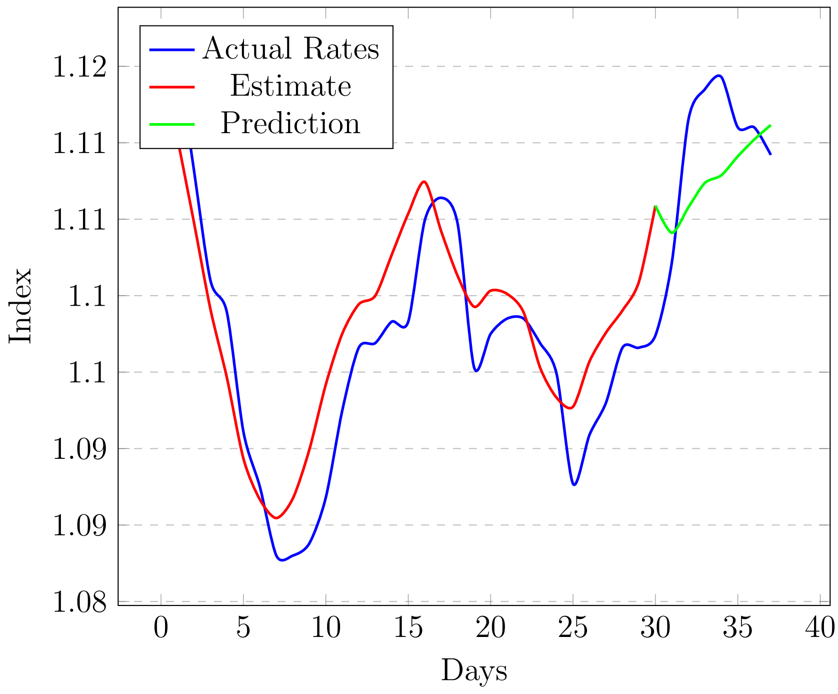

Figure 3: EUR/USD Rates for August 2015 using Neural Networks

The network shown in figure 3 was trained for 1,000 iterations, with a *learning rate* of 0.05 and a *momentum learning coefficient* of 0. It has 3 inputs and 8 hidden layers.

Its mean absolute percentage error for the training set is 0.4825%, while the prediction error for the next 7 days is 0.2790%.

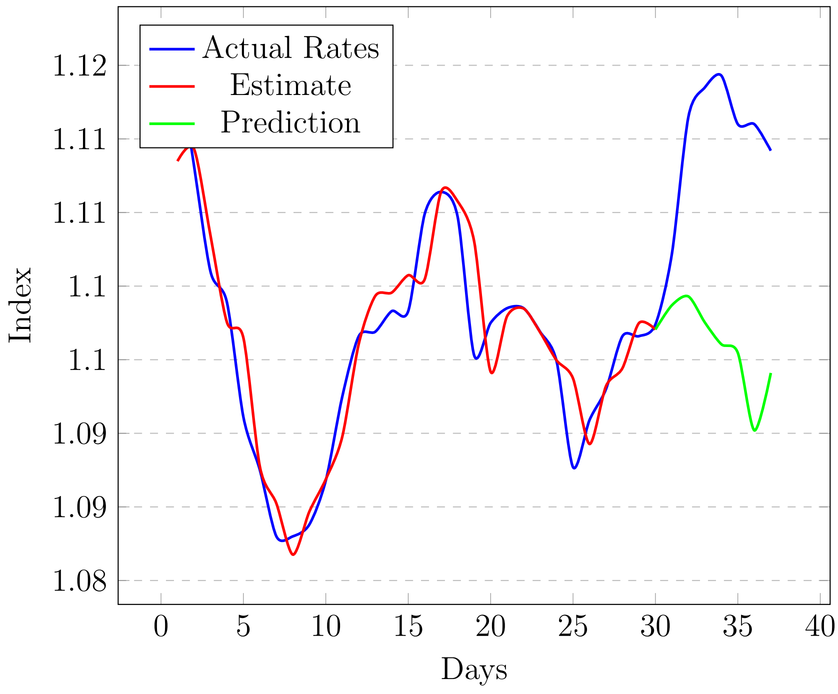

Figure 4: EUR/USD Rates for August 2015 using Genetic Algorithms

As a comparison, in figure 4, the same dataset is taught with a genetic algorithm. The algorithm ran for 10,000 iterations, while having a population size of 100 and using a selection method which selects the chromosomes with the highest fitness value. The configuration had 10 variable inputs and 3 constants, as explained in table 1.

$	**Value**	**Explanation**	**Type**
a..j	1.0875	Moving values from set.	Variable
k	1.0830	Minimum of the training set.	Constant
l	1.0977	Arithmetic average of the training set.	Constant
m	1.1143	Maximum of the training set.	Constant

Table 1: Genetic Algorithm Inputs: Variables and Constants

The final fittest chromosome has a mean absolute percentage error of 0.1821% against the training set, and a prediction error of 0.1415% for the next 7 days.

The output of the genetic algorithm, meaning the fittest chromosome, is the following equation:

$$\frac{a \cdot d + b \cdot d \cdot g \cdot g - b \cdot k + d \cdot f \cdot g \cdot g - f \cdot k}{d}$$

which can be simplified to:

$$a + (b + f) \cdot (g^2 - \frac{k}{d})$$

4.2 Prediction Accuracy Across Various Sliding Windows

In this experiment, I set out to measure the accuracy of predictions in regards to the number of days trained and requested for prediction.

The foreign exchange data form 2013 to 2015 for the currency pairs EUR/USD, EUR/GBP and EUR/RON were split into 180-day chunks, resulting in a total of 5 subsets per pair. The resulting 15 subsets were processed as described in section 3.2 in order to determine the best parameters when training with a training set of size 14, 30, 90 and 180. Errors were calculated, as described in section 3.1, for prediction windows of 1, 5, 7, 14 and 30 days.

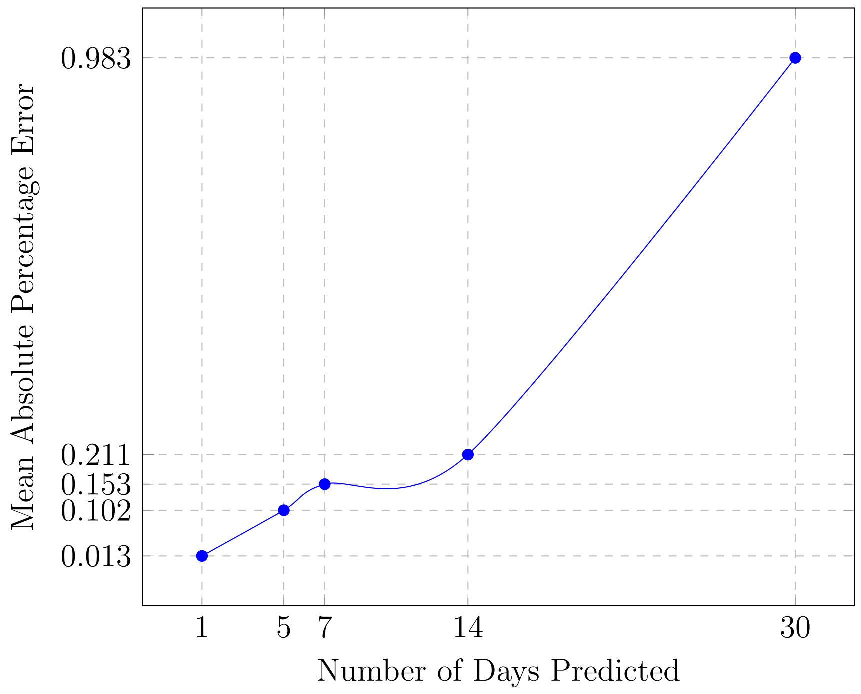

Figure 5: Average Error Increase per Days Predicted

Days	**Predictions**	**Inputs**	**Hiddens**	**Learn Rate**	**Momentum**	**MAPE**
14	1	3	5	0.05	0.05	0.0133
14	5	3	8	0.5	0.5	0.1654
14	7	3	5	0.05	0.05	0.2804
14	14	3	5	0.05	0.05	0.3006
30	1	3	6	0.05	0.05	0.0128
30	5	3	2	1	0.988	0.1475
30	7	3	8	0.05	0	0.2790
30	14	3	9	0.05	0.988	0.4808
90	1	3	4	1	0.988	0.0175
90	5	3	4	1	0.988	0.1022
90	7	3	4	1	0.988	0.1530
90	14	4	6	1	0.988	0.2106
180	1	4	6	0.05	0.988	0.0540
180	5	3	7	0.05	0.988	0.4458
180	7	3	7	0.05	0.988	0.6074
180	14	3	7	0.05	0.988	0.7265
180	30	5	3	0.05	0.988	0.9826

Table 2: Best-Performing Neural Configurations Across Various Data Sets

4.3 Prediction Accuracy for Moving Averages

Since the actual rates, as fed into the neural network previously, contains extensive oscillations, I decided to try and use a moving average as the input, and try to predict that.

In figure 6, 90 days of EUR/USD data is displayed, with a 30-day *SMA* (*Simple Moving Average*) overlapped on top of it. As it can be observed, the moving average line resembles the trend of the actual rates, with the oscillations taken out of it.

The configuration of the neural network used to train this dataset had 3 inputs and 12 hidden layers, closely resembling the configuration of previous experiments. However, it should be noted that the *learning momentum* parameter of the back-propagation teacher, while set as close as possible to maximum in previous experiments, this time it was set to 0.0, as oscillations are not part of this dataset.

The mean absolute percentage error was 0.0663% for the training set and 0.0985% for the 30 days of predictions. This is a significantly lower error value than what is presented in table 2, however those are actual index predictions, not moving average estimations.

This is not a useless feat, however. Most technical indicators, such as *RSI* and *MACD* do use moving averages such as *SMA* or *EMA* as their basis, so accurate estimation of the moving average lines can be an important factor in larger decision-maker systems.

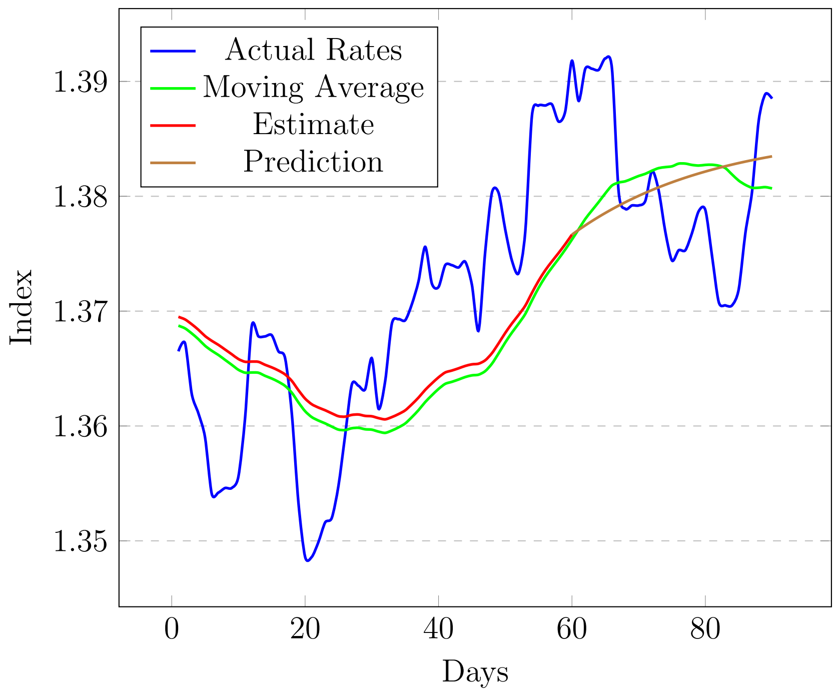

Figure 6: SMA of EUR/USD Rates for January 2014 using Neural Networks

4.4 Technical Indicators via Genetic Algorithms

4.4.1 Relative Strength Index

RSI[5] (*Relative Strength Index*) is a technical momentum indicator, which indicates overselling and overbuying by comparing the recent gains and losses averages. In order to calculate it, two 15 or 30-day moving averages are required to track gains and losses:

$$RSI = 100 - \frac{100}{1 + \frac{EMA_{gains}}{EMA_{losses}}}$$

If the RSI approaches 30%, the asset is being oversold and therefore its price is approaching its peak. On the other hand, if the RSI approaches 70%, the asset is overvalued, and therefore its price is bound to drop.

For the results shown in figure 7, the mean absolute percentage error was 5.6955% for the training set and 13.4531% for the 30 days of predictions. The parameters and input variables were set-up the same as described in section 4.1. The equation of the final fittest chromosome was:

$$\frac{5 \cdot d^2}{a^3} + a + \frac{1}{b}$$

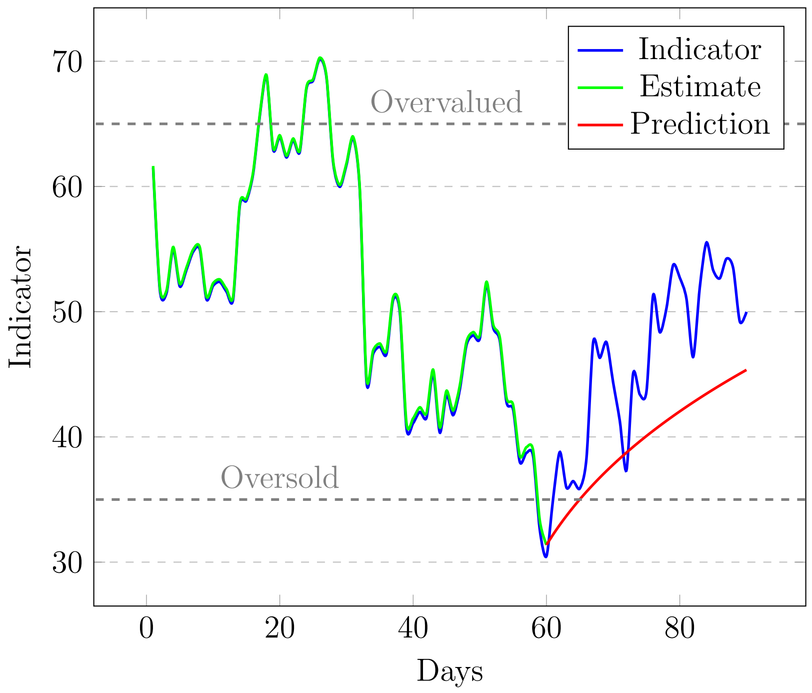

Figure 7: RSI of XBT/USD Rates for February 2015 using Genetic Algorithms

4.4.2 Moving Average Convergence/Divergence Oscillator

MACD[6] (*Moving Average Convergence/Divergence*) is another industry-standard technical indicator. This indicator can be computer by subtracting the shorter moving average line (e.g. 12-day *EMA*) from the longer moving average line (e.g. 26-day *EMA*) thus revealing the trend following tendencies:

$$MACD = EMA_{short} - EMA_{long}$$

The *MACD* line is fluctuates around the zero line, however the same information is conveyed in related indicators[7], such as *PPO* (*Price Percentage Oscillator*) and *DPO* (*Detreneded Price Oscillator*) in a percentage form. For the purposes of this experiment, only *MACD* will be demonstrated, since *PPO* is just scaled to a different range, and *DPO* is just delayed for detrending purposes.

For the results shown in figure 8, the mean absolute percentage error was 8.1986% for the training set and 6.2930% for the 30 days of predictions. The parameters and input variables were set-up the same as in the previous section. The fittest chromosome was:

$$7a(3a - b) + (3a + d)^2 + a - b^2 - 2d$$

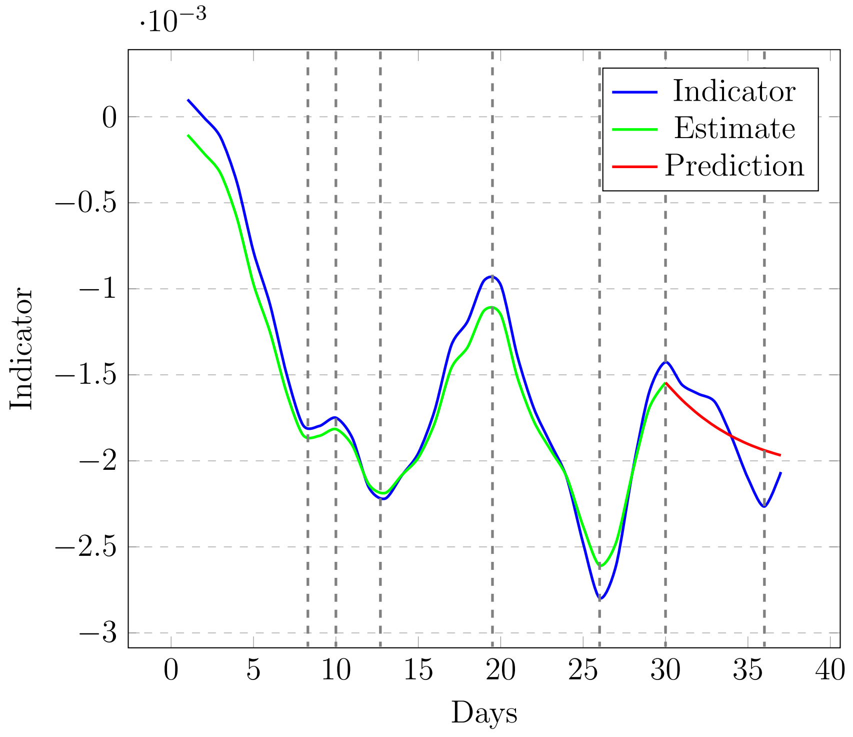

Figure 8: MACD of EUR/GBP Rates for December 2013 using Genetic Algorithms

5 Comparison of Local Results against Literature

After having definitive results for most data sets, see table 2, I decided to compare my *MAPE* values with those published in relevant research papers.

In paper [4], various S&P 500 stock indices are evaluated with various methods. The best result obtained in that paper has a mean absolute percentage error of 0.6747 when predicting 1 day into the future, as seen in table 2, using *ABFO* (*Adaptive Bacterial Foraging Optimization*). The local mean absolute percentage error obtained in this assignment when predicting 1 day into the future ranges between 0.0133 to 0.0540, depending on the size of the training set.

In paper [8], IBM, Dell, British Airlines and Ryanair Airlines stocks are being evaluated using several methods, such as stock market forecasting through *HMM*[9] (*Hidden Markov Model*), fusion of *HMM and ANN*[10] (*Artificial Neural Network*), combination of *HMM and fuzzy logic*[11], and last but not least, the proposed method within the paper, *CGFS* (*Clustering-Genetic Fuzzy System*). The training set used within consists of 287 data points spreading between 2003 and 2004. The lowest mean absolute percentage error value is 0.5340 when forecasting a span of 88 days of Dell stocks. The local results presented in table 2 do not go as far as 88 days, however the mean absolute percentage error for 30 days is 0.9826 when trained on a data set consisting of 180 data points. Re-running the algorithm presented in section 3.2 for a training data set of 287 days of EUR/GBP data and predicting a span of 88 days results in a mean absolute percentage error of 0.7206.

6 Future Work

As noted in section 1, one of the main challenges of prediction is the fact that many outside factors may disrupt the natural oscillation of a stock or foreign exchange rate. Further avenues that can be explored are, similar to what paper [12] experimented with, the correlation of news events with the data points.

For example, the news headlines can be fetched from reputable sources for a given stock's parent company or a currency's central bank (e.g. *ECB* for *EUR*). The news snippets can then be individually analyzed using entity-oriented sentiment analysis in order to determine whether the observed body has had a positive or negative impact described in the news article.

7 Bibliography

[1] F. W. Op't Landt *et al.*, "Stock price prediction using neural networks," 1997. ⇒ 3

[2] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10389–10397, 2011. ⇒ 4

[3] J. S. Armstrong and F. Collopy, "Error measures for generalizing about forecasting methods: Empirical comparisons," *International journal of forecasting*, vol. 8, no. 1, pp. 69–80, 1992. ⇒ 5

[4] R. Majhi, G. Panda, B. Majhi, and G. Sahoo, "Efficient prediction of stock market indices using adaptive bacterial foraging optimization (abfo) and bfo based techniques," *Expert Systems with Applications*, vol. 36, no. 6, pp. 10097–10104, 2009. ⇒ 5, 13

[5] J. Wilder, *New Concepts in Technical Trading Systems.* Trend Research, 1978. ⇒ 11

[6] G. Appel, *The Moving Average Convergence-divergence Trading Method: Advanced Version.* Scientific Investment Systems, 1985. ⇒ 12

[7] G. Appel, *Technical Analysis: Power Tools for Active Investors.* Prentice Hall, 2011. ⇒ 12

[8] E. Hadavandi, H. Shavandi, and A. Ghanbari, "Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting," *Knowledge-Based Systems*, vol. 23, no. 8, pp. 800–808, 2010. ⇒ 13

[9] M. R. Hassan and B. Nath, "Stock market forecasting using hidden markov model: a new approach," in *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pp. 192–196, IEEE, 2005. ⇒ 13

[10] M. R. Hassan, B. Nath, and M. Kirley, "A fusion model of hmm, ann and ga for stock market forecasting," *Expert Systems with Applications*, vol. 33, no. 1, pp. 171–180, 2007. ⇒ 13

[11] M. R. Hassan, "A combination of hidden markov model and fuzzy model for stock market forecasting," *Neurocomputing*, vol. 72, no. 16, pp. 3439–3446, 2009. ⇒ 13

[12] Y. Zhai, A. Hsu, and S. K. Halgamuge, "Combining news and technical indicators in daily stock price trends prediction," in *Advances in Neural Networks–ISNN 2007*, pp. 1087–1096, Springer, 2007. ⇒ 13