Utilizing Artificial Neural Networks and Genetic Algorithms to Build an Algo-Trading Model for Intra-Day Foreign Exchange Speculation.

Cain Evans[1], Konstantinos Pappas[1], Fatos Xhafa[b]

[a] *Faculty of Technology, Engineering and the Environment School of Computing, Telecommunications and Networks Birmingham City University, UK*
[b] *Dept de Lienguatges I Sistemes Informatics Universitat Politcnica de Catalunya*

Abstract

The Foreign Exchange Market is the biggest and one of the most liquid markets in the world. This market has always been one of the most challenging markets as far as short term prediction is concerned. Due to the chaotic, noisy, and non-stationary nature of the data, the majority of the research has been focused on daily, weekly, or even monthly prediction. The literature review revealed that there is a gap for intra-day market prediction. Identifying this gap, this paper introduces a prediction and decision making model based on Artificial Neural Networks (ANN) and Genetic Algorithms. The dataset utilized for this research comprises of 70 weeks past currency rates of the 3 most traded currency pairs: GBP\USD, EUR\GBP, EUR\USD. The initial statistical tests confirmed with a significance of more than 95% that the daily FOREX currency rates time series are not randomly distributed. Another important result is that the proposed model achieved 72.5% prediction accuracy. Furthermore, implementing the optimal trading strategy, this model produced 23.3% Annualized Net Return.

Keywords:
Foreign Exchange, Artificial Neural Networks, Genetic Algorithms, Trading Strategies, Technical analysis.

1. Introduction

According to the Bank of International Settlements (www.bis.com), FOREX is a fast growing market that at the moment is estimated at $3.98 trillion. The time series of different currency rates are described as chaotic, extremely noisy, and non-stationary [14]. Almost every research paper that presents a prediction model starts with a reference to the Efficient Market Hypothesis (EMH). According to [11], a random walk-efficient market is big enough to be manipulated by an existing large number of profit-maximizers who are competing with each other trying to predict future market values. This statement implies that the price of an asset reflects all the information that can be obtained at that time and that due to conflicting incentives of the participants the price will reach its equilibrium despite temporal disturbances due to market noise. While the theory seems to be correct for the time it was published, with the passing of time and due to future developments, especially in the area of communication and its impact on a globalized market, things seems to be different.

Today the world is highly interconnected and it takes a fraction of a second to transmit financial news around the globe. The reaction by the markets to market news tends to have an impact on the financial time series. The sequence of those reactions creates patterns that some practitioners believe that, as history repeats itself, so the market reaction does [23]. Going further, [2] provided evidences that there are cases where markets appear to be inefficient. Furthermore, [30] developed a hybrid prediction model based on an ANN and a Genetic Algorithm (GA) which gives evidence that financial time series are not entirely random.

Future market forecasting techniques can be classified into two major categories: fundamental analysis, and technical analysis. Fundamental analysis is based on macro-economic data such as Purchasing Power Parity (PPP), Gross Domestic Product (GDP), Balance of Payments (BOP), Purchasing Manager Index (PMI), Central Bank outcomes, etc. It is obvious that this kind of analysis has a more long term prediction spectrum and it is not the case for this paper. On the other hand, technical analysis focuses on past data and potential repeated patterns within those data. The major point here is that the history tends to repeat itself. As opposed to fundamental analysis, technical analysis makes short term predictions such as weekly, daily, or even hourly predictions.

There is a conglomeration of available tools suitable for technical analysis such as ANN, GA, Genetic Programming (GP), Econometrics tools, technical indicators, etc. [16] introduces a forecasting optimization model that is based on a genetic algorithm that automatically generates trading rules based on technical indexes such as Moving Average (MA), and Relative Strength Index (RSI). [36] utilizes wavelet analysis to feed an ANN that predicts FTSE 100 index time series. [5] developed an Evolutionary ANN (EANN) that makes future predictions based on macro-economic data. [32] proposes a Support Vector Machine (SVM) regression approximation model.

The main purpose of this paper is to confirm the hypothesis that intra-day FX market prediction is possible. To achieve this goal, four basic objectives should be satisfied. The first objective is to define the trading strategies and

to develop the algorithms that Implements those strategies. The second goal is to process the dataset and to reject the EMH hypothesis. The third goal is to develop the forecasting model based on ANN and GA and the last objective is to test the developed model and evaluate the performance of the introduced trading strategies.

The remainder of the paper has the following structure: Section 2 presents the literature review, while Section 3 introduces the FOREX market and also describes the trading strategies that will be implemented by the developing model. Section 4 describes the prediction and the decision making models respectively. A comprehensive analysis of the results is presented in Section 5 followed by the conclusions.

2. Relative Work

There is a number of papers in the literature that propose different methodologies and techniques for trading prediction in FOREX markets. [10] experimented with the predictability of ANN on weekly FX data, and concluded that, among other issues, one of the most critical issues to encounter when introducing such models is the structure of the data. [20] compared different ANN models feeding them with technical indicators based on past FOREX data and concluded that Scaled Conjugate Gradient based model achieved closer prediction compared to the other algorithms. [41] presented a high frequency foreign exchange trading strategy model based on GA. This model utilizes different technical indicators such as MA, Moving Average Convergence Divergence (MACD), Slow Stochastic, RSI, Momentum Oscillator, Price Oscillator, Larry Williams, Bollinger Bands, etc. Several test showed an annualised return rate of 3.7 %. [39] presented a dual model for FX time series prediction based on Generalized Linear Autoregressive and ANN model. The experiments showed that the dual model outperformed the single model approach based on econometrics techniques.

3. The FOREX Market and the Trading Algorithm

The modern foreign exchange market started to take shape after the abandonment of Bretton Woods system of monetary management. Some of the unique features of this particular market include: the huge daily trading volume, the geographical dispersion, and the continuous operation during the weekdays. Imagine the FX market as a concentric system with different circles around the central point. At the epicentre of the system, the interbank market is made up of the largest commercial banks and security dealers. The next circle, the second tier, comprises of other smaller participants such as commercial companies, hedge funds, pension funds, foreign exchange traders, etc. The larger the distance from the epicentre, the wider the bid-ask spread.

According to [1] , foreign exchange markets accommodate three types of players: high frequency traders, long term investors, and corporations. This

paper proposes a trading model for high frequency traders who speculate on small intraday price fluctuations. A trading system consists of three major parts: rules for entering and exiting trades, risk control, and money management [7] . Money management refers to the actual size of the trade to be initiated [37] . This paper will concentrate the efforts on return rates, and as such, it will not refer to money management. Just for informative purposes, the lowest limit for joining the FX market is $100000. With a leverage of 100:1, an individual player can initiate the trade starting with $1000. Of course, a high frequency trading technique is structured on the basis of making profits from small price fluctuations, and as such, a substantial amount of money should be initialized. The second important part of a trading model, risk management or hedging, refers to the method of covering potential losses by trading assets that are negatively correlated to the current position. Because of the short timeframe of the high frequency trading, this method is not included to the suggested strategies.

The rules to enter and exit, or strategies, require adequate knowledge of the markets. The definition of a strategy includes the introduction of the entry time, the trading duration and exit conditions, the trading, assets, as well as other parameters such as for instance whether more than one asset will be traded simultaneously, etc. High volume is one of the criteria to decide when to start trading and this happens when two markets overlap. The highest volume is usually observed during the switching between European and American market. The reason that this parameter is so important is because high volume means narrower bid-ask spread. Therefore, in the proposed trading strategy the trading session starts at 12:36 GMT and is being terminated after six hours.

The next section introduces three trading strategies that are implemented by the proposed model. The first is the simplest one. There is a base currency and a quote currency. Based on the outcome of the prediction model that will be introduced later on, the model produces long or short signal. The second strategy involves two currency pairs. Here the decision about the wining trade is based on the best return. The third strategy includes two currency pairs that will be both traded simultaneously. The following predicate logic sentences present the conditions of those strategies. For the sake of brevity, the neutral position has been omitted.

First FX Trading Strategy S1

$$base(curr1) \wedge up(curr2) \Rightarrow trade(long(curr2))$$

$$base(curr1) \wedge down(curr2) \Rightarrow trade(short(curr2))$$

Second FX Trading Strategy S2

$$base(curr1) \wedge down(curr2) \wedge down(curr3) \wedge (Greater(curr2, curr3)) \Rightarrow trade(short(curr3))$$

$$base(curr1) \wedge down(curr2) \wedge down(curr3) \wedge (Greater(curr3, curr2)) \Rightarrow trade(short(curr2))$$

$$base(curr1) \wedge up(curr2) \wedge down(curr3) \wedge (Greater(abs(curr2), abs(curr3))) \Rightarrow trade(long(curr2))$$

$$base(curr1) \wedge up(curr2) \wedge down(curr3) \wedge (Greater(abs(curr3), abs(curr2))) \Rightarrow trade(short(curr3))$$

$$base(curr1) \wedge down(curr2) \wedge up(curr3) \wedge (Greater(abs(curr2), abs(curr3))) \Rightarrow trade(short(curr2))$$

$$base(curr1) \wedge down(curr2) \wedge up(curr3) \wedge (Greater(abs(curr3), abs(curr2))) \Rightarrow trade(long(curr3))$$

$$base(curr1) \wedge up(curr2) \wedge up(curr3) \wedge (Greater(curr2, curr3)) \Rightarrow trade(long(curr2))$$

$$base(curr1) \wedge up(curr2) \wedge up(curr3) \wedge (Greater(curr3, curr2)) \Rightarrow trade(long(curr3))$$

Third FX Trading Strategy S3

$$base(curr1) \wedge down(curr2) \wedge down(curr3) \Rightarrow trade(short(curr2), short(curr3))$$

$$base(curr1) \wedge down(curr2) \wedge up(curr3) \Rightarrow trade(short(curr2), long(curr3))$$

$$base(curr1) \wedge up(curr2) \wedge down(curr3) \Rightarrow trade(long(curr2), short(curr3))$$

$$base(curr1) \wedge up(curr2) \wedge up(curr3) \Rightarrow trade(long(curr2), long(curr3))$$

Having introduced the strategies, now it is time to define the algorithm that executes those strategies. An algorithm is a sequence of executable commands that has a beginning, a body and an end. In addition, an algorithm is fed with some kind of inputs and produces some outputs. In this particular case the algorithm is given the 16 hours previous currency prices in 40 minutes intervals, and the algorithm makes a trading decision. It is important to mention here that the data will be analysed and processed before being ported to the model. The next three figures present the activity diagram of each strategy.

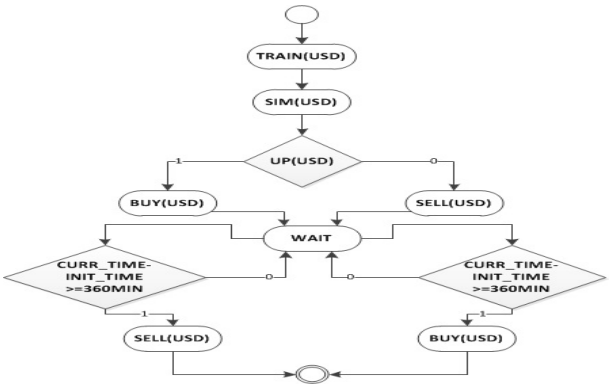


Figure 1: S1 - Activity Diagram

The first two nodes of the activity diagram in figure 1 represent the forecasting part of the algorithm. Given the forecasting results, and depending on whether the prediction shows the currency going up or down, the algorithm sends the appropriate signal. The trading time has been set to six hours. After the pass of this time, the algorithm sends a trading signal to terminate the session and to record possible gains or losses.

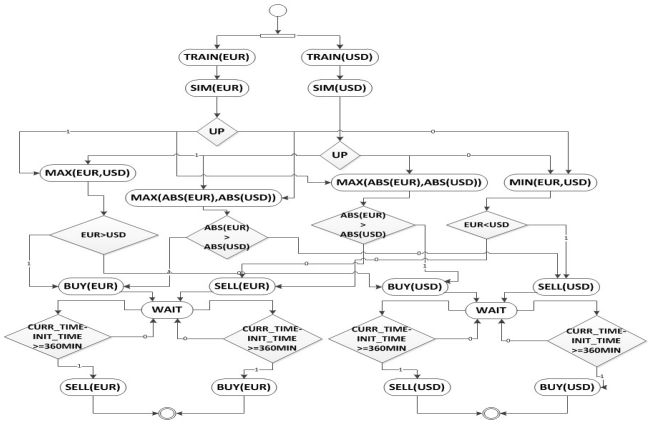


Figure 2: S2 - Activity Diagram

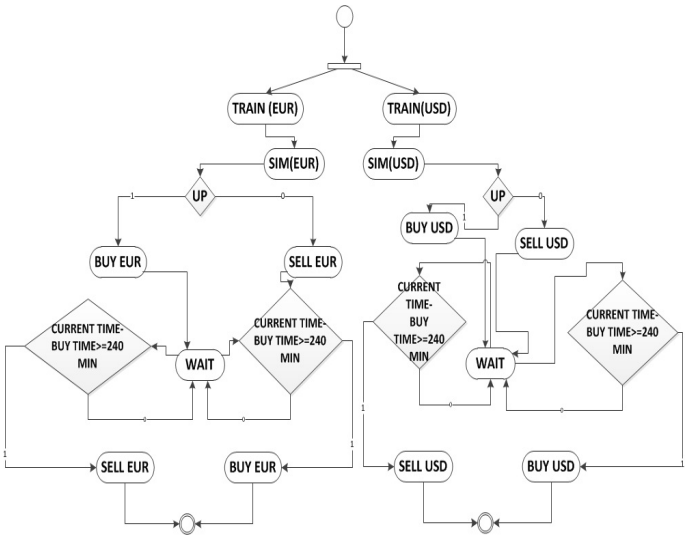


Figure 3: S3 - Activity Diagram

As the activity diagram in Figure 2 depicts, both the first and the second algorithm share the same basic principles. The difference between the first and the second algorithm is in the fact that while the first strategy initiate only one currency pair during the whole session, in the case of the second algorithm two currencies are initiated and the algorithm decides to trade one of the two currencies based on the best performance. On the other hand, the third algorithm that is depicted in figure 3 is somewhere between the previous two algorithms. Here two currencies are initiated and both currencies are traded simultaneously. This is a kind of modest strategy that tries to minimize the risk.

The next picture presents the overall trading algorithm that gives the opportunity to the investor to choose between a single strategy or to leave the system to choose the optimal trading strategy. Therefore, the algorithm accepts as input the base currency and one of the three trading strategies if the case is to run one of the strategies. This is defined by choosing the path A of the trading algorithm. If path B is activated, then there is no need for defining the strategy as the algorithm compares the performance of the three strategies and initiate the most profitable one.

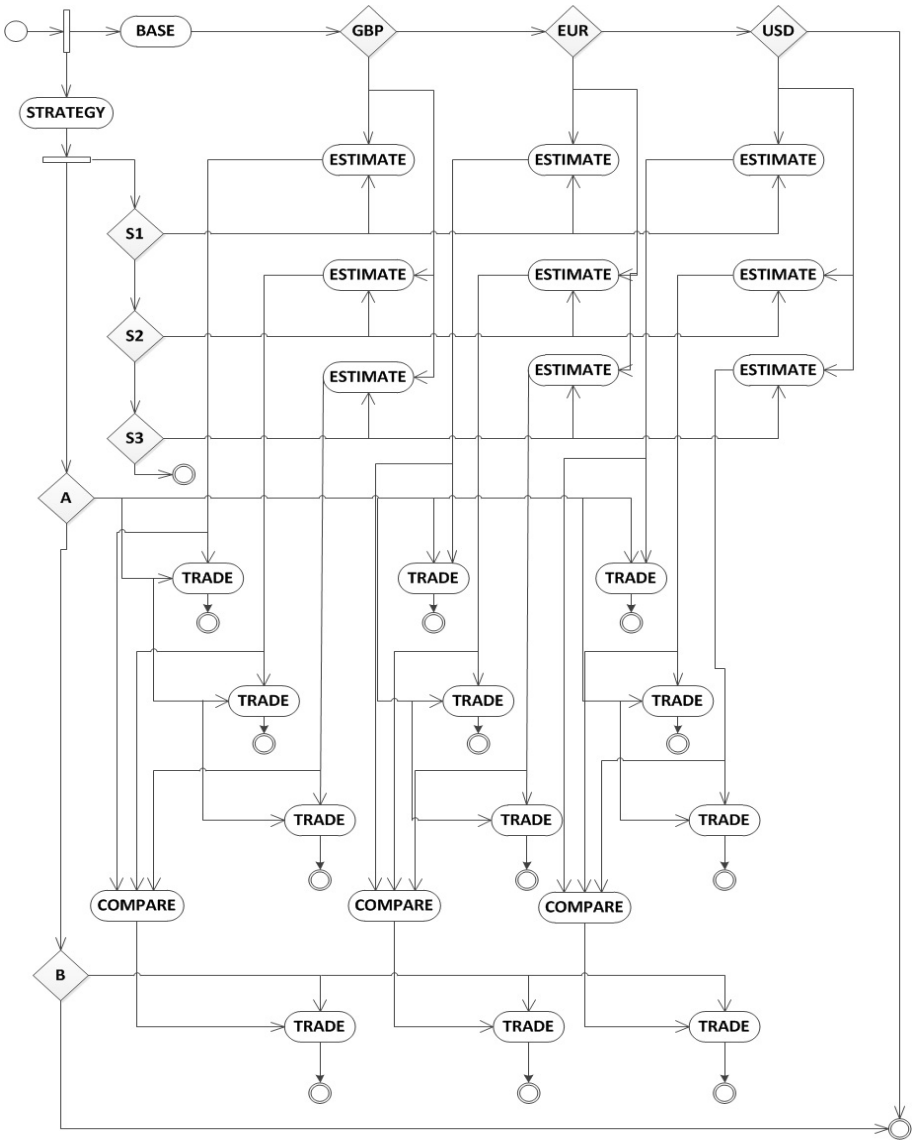


Figure 4: Activity Diagram of the Trading Algorithm

An important property of this algorithm is the scalability which allows additional strategies to be incorporated into the model. Furthermore, the modularity of the proposed model means that the system is able to trade other financial instruments such as securities and more complex products including futures and options.

4. The Forecasting Model

Commonly used methodologies for making currency rate predictions include econometrics, technical indicators, and AI techniques such as ANN and GA. Additionally, those solutions are not mutually exclusive. The literature has revealed that there are cases where the combination of two or more techniques offers a better result. [40] introduce an online learning algorithm to accelerate the learning process of the neural network. The results revealed that the algorithm outperformed other algorithms such as batch learning and Levenberg-Marquard algorithm. [38] use MA5, MA10, MA20, MA60, and MA120 to feed a Feed Forward Neural Network that forecasts several currency rates. The weekly based results are evaluated as positive.

Selecting the proper tool depend on the many factors, such as the nature of data and the adopted methodology. Econometrics tools seem to have an adequate performance when data exhibit linear dependency as opposed when data are non-linearly correlated. On the other hand, neural networks provide generalization and mathematical function approximation to reveal or associate relations between input and target data in the case of supervised learning [10].

4.1. Data Analysis

FX intraday rates time series are described as noisy, chaotic, displaying nonlinear relation, and exhibiting non-stationary behaviour. It is obvious that it is difficult to provide those data for prediction without firstly having some king of transformation. One of the first issues to deal with is the frequency of the sampling. High sampling frequency means additional useless and sometimes disorienting information. On the other hand, lower frequency means that not all the essential information is included. According to [29] it is sufficient to sample the markets at intervals between 5 and 60 minutes depending upon the currency pair. There are different methodologies for defining the appropriate frequency with the most frequent the analysis of the autocorrelation. The noise of the sample is another issue that should be addressed. The literature review revealed that technical indicators e.g. MA is one of the preferred solutions. The last question to be answered before the forecasting model is developed concerns the level of data predictability. This means that the time series should be exposed to statistical tests to confirm or to reject random behaviour. Wald-Wolfowitz test and Kolmogorov-Smirnov are examples of statistical tests that are utilized for this purpose.

This paper examines and experiments with the predictability of three major currency pairs: GBP \USD, EUR \GBP, EUR \USD. These data correspond to 70 week spot rates tick observation from 1/10/2010 to 28/2/2012. The selected sampling frequency is 40 minutes and the noise has been mitigated by taking the average of the 40 minutes time intervals. The following graph shows the behaviour of GBP \USD exchange rate during the sampling period.

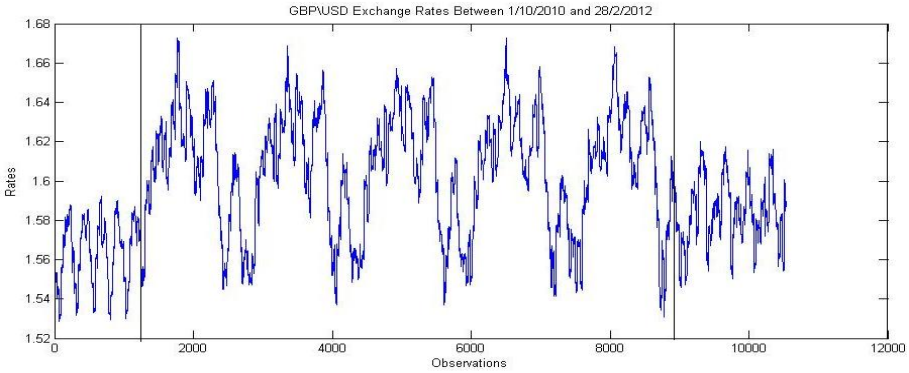


Figure 5: Activity Diagram of the Trading Algorithm

The middle compartment of the graph displays the observations of 2011. As the variance is much greater than the data of 2010 and 2012, it is clear that the volatility of this year is greater than observations coming from the previous or the next year. It is important here to mention that there are two rates in the forex market: the ask or buy, and the bid or sell rate.

To keep the model simple, it has been decided that the model will be fed only with ask data. This choice does not necessarily means that essential information is being lost as the examination of both bid and ask data on each currency pair has showed that bid and ask values are highly correlated. the following table shows the correlation coeficient and the confidence value of bid and ask time series of each currency pair.

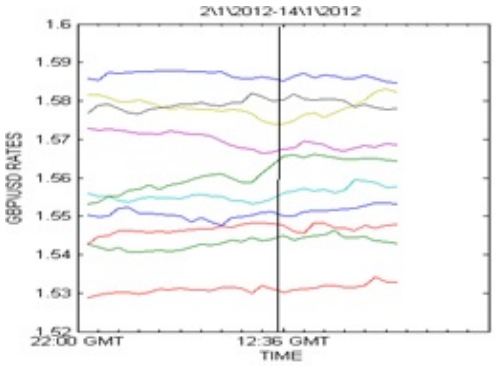
Currency Pair	**Correlation**	**P-value**
GBP\USD	1	0
EUR\GBP	1	0
EUR\USD	1	0

Table 1: Correlation Between Bid and Ask Rates

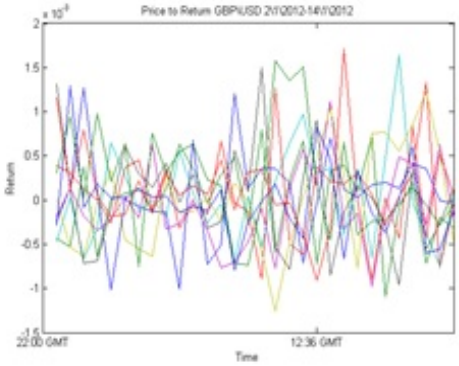
Data representation is the next step of the process. According to [37] it is important not to let the ANN have visibility of the market prices, or currency rates this case. If raw prices are being provided as input, then two identical patterns that defer by a constant will be treated as two different patterns and make very difficult the generalization process. To overcome this issue, the datasets are being transformed to return rates by utilizing the function

$$r = \ln(\frac{P_t}{P_{t-1}})$$

where ln is the natural logarithm, Pt is the price at time t, and Pt-1 is the price at time t-1. The following graphs show the data before and after the transformation.



(a)



(b)

Figure 6: The Data Before and After the Transformation

While this transformation provide better generalization, on the other hand rises new issues such as convergence (generalization and convergence will be covered in more details when referring to neural networks). One frequent way of dealing with this issue is by introducing some kind of technical indicators to smooth the data. Prediction systems that intend to make short term prediction expose relatively adequate positive performance when fed with technical indicators. [26] utilizes Exponential Moving Average (EMA) and Bollinger Bands (BB) to build a day-trading system based on NN that attempts to indicate the optimal enter and exit strategy. Furthermore, [35] feeds the proposed model with a basket of technical and fundamental data both processed with technical indicators such as MA, MACD, and RSI. The experiments showed that the particular system achieved a 70% success in predicting the right direction of the

market.

Moving Average, in one or another form, is a technical indicator found in almost every paper describing financial time series forecasting models. The MA technique performs pretty well when the market follows a trend. However, this indicator performs rather poorly when the index changes direction. To tackle this issue, this paper proposes an alternate version that is derived by the following formula:

$$y_i = \frac{\sum_{j=1}^{i} x_i}{i}$$

This version is named Incremental Window Moving Average (IWMA) and the intention is that the indicator sleeks the data points gradually so that the system can react better to the market turning points. The next figure shows how the data of the previous graph appear after the proposed transformation.

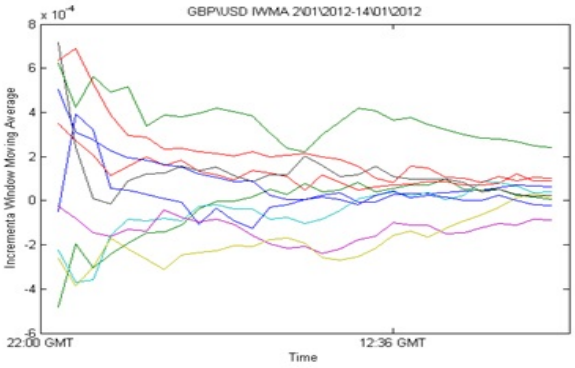


Figure 7: The Data after the IWMA Transformation

After detrending and normalization, there is a last step before providing the data to the ANN. Testing for randomness is a very important issue because it is pointless to try to forecast future values of a time series with random values. There are different methodologies and techniques to test for randomness. This paper adopts two tests: WaldWolfowitz and Kolmogorov-Smirnov tests. To make the process more complete, the results of the descriptive statistics, plus the aforementioned tests are compared with the results coming from a normally distributed random time series of the same length. The upper diagrams of the following figure show the distribution of the data before detrending and normalization, while the diagrams bellow them comprise of the histogram of the corresponding time series. The table that follows figure 9 shows the results of the tests and the descriptive statistics elements.

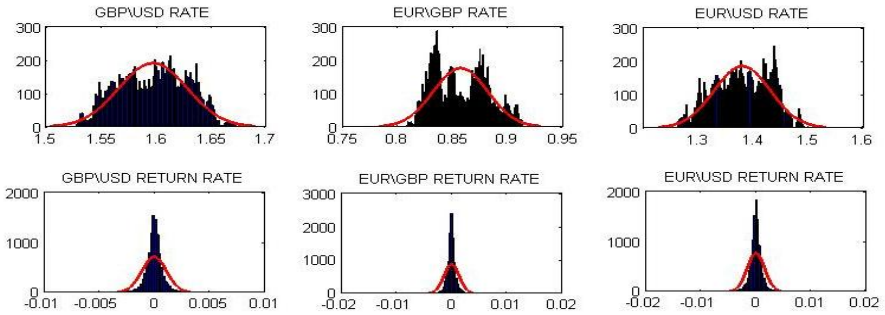


Figure 8: Data Distribution

	GBP\USD	**EUR\GBP**	**EUR\USD**
Observations	8700	8700	8700
Mean	-0.000026	0.000032	0.000038
Median	-0.000028	-0.00003	0.000028
Std. div.	0.000305	0.00029	0.0004
Skewness	-0.2151	-0.5848	0.0798
Kurtosis	13.6738	15.519	11.9587
Minimum	-0.003	-0.0041	-0.0034
Maximum	0.0026	0.0023	0.0045
Kolmogorov H	1	1	1
Kolmogorov P	0	0	0
Kolmogorov ST	0.499	0.4991	0.4988
Wald-W Test H	1	1	1
Wald-W Test P	0	0	0

Table 2: Time Seeries Descriptive Statistics

Both tests indicate that the examined time-series diverges from normal distribution as in all the cases the hypothesis that the data are normally distributes is rejected. Furthermore, the Kolmogorov statistic shows that the data do diverge significantly from normal distribution as the three time series resulted in a Kolmogorov statistic value of 0.49 while in the case of the random time series the value was under 0.01. Kurtosis also helps in making assumptions about the normality of the data. As the real data show a kurtosis far above the value of 3, this is another indication of the normality rejection.

There is a last issue to solve before presenting the neural networks. Data consistency is very important. If the prediction model is fed with irrelevant data, then the results is going to be poor. To tackle with the market evolution, a good practice is to keep the input data consistent. One way to achieve this

is to periodically replace past data with more recent data. In other words, imagine the sample test as a long queue where the data are being placed in reverse chronological order and where for each new trading day, the queue is being pushed by a portion of a trading day data so that the oldest trading day is discarded from the front of the queue, while the most recent data are placed in the back end.

4.2. Neural Networks

Neural networks are described as the processing methodology that maps the input values to the target values. Non-linear modelling, generalization, and universal approximation are some of the advantages of the neural networks [6] . These tools are classified according to the learning techniques in three main categories: supervised learning, reinforcement learning, and unsupervised learning. Feed Forward Multi-Layered Perceptron belongs to the first category. The network has a layered structure where each neurons synapses are connected only to the output of the previous layer and the output of the same neuron is connected only to units of the following layer. There are several important factors to consider when developing a neural network including: the input and output vectors, the activation function, the training function, and the structure of the network.

4.2.1. The Dataset

As mentioned earlier, supervised learning requires some input and the corresponding target data. Given the input data, the network is responsible for being able to produce outputs similar or approximately similar to the target data. The whole data set is usually separated into three parts, the training set (70%), the validation set (15%) and the testing set (15%). The validation set indicates when the network has been trained. This happens when the validation error is becoming greater than the training error. This process guaranties that the networks is not being over-fitted. On the other hand, the testing set measures the forecasting performance of the network. The training and the validation set is shuffled to avoid time dependent learning. The literature review revealed that the testing or the validation set should be approximately from one forth to one eight of the training set [27]. Additionally, [21] suggests that a balanced split is 70-15-15 for training, validation, and testing set.

Back to the proposed model, the dataset comprises of the exchange rates of the three major currency pairs (GBP \USD, EUR \GBP, EUR \USD) the period from 1-10-2010 to 28-2-2012. The previous section described the analysis and the preparation of the data. As such, the input dataset contains vectors of 20 elements of detrended and normalised daily currency rates that correspond to 14.6 hours of daily trading between 22:00 GMT and 12:36 GMT. The target dataset comprises of single point value that corresponds to the detrended and normalized return rate experienced 6 hours (at 18:36 GMT) after the last value of the input dataset. This means that the prediction horizon is 6 hours. The dataset is separated into four parts. From the first 300 time series, 70% of the

data is the training set, 15% the validation set, and 15% as the in-sample testing set. The rest 40 time series serve as the out-of sample testing set.

4.2.2. Activation Function

Activation function is a way that the output of an individual neuron is scaled to the desired value range. There are different activation functions such as the linear activation function, the logistic function, the hyperbolic tangent function, etc. The hyperbolic tangent function takes values from the range [-∞,∞] and squash them to the range [-1,1]. This function is given by the following formula:

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^{xg} - e^{-xg}}{e^{xg} + e^{-xg}} = \frac{e^{2xg} - 1}{e^{2xg} + 1}$$

where g is the gain. Small values of g make the slope very smooth while large values make the function to behave like the step function. The following figure shows the plot of the hyperbolic tangent function where x ∈ [-5,5] and g: 0.15,0.4,0.6,1,50.

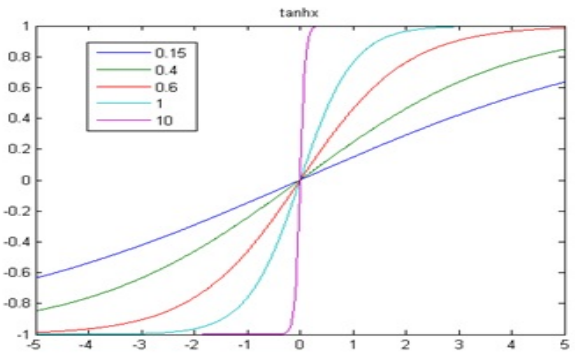


Figure 9: Hyperbolic Tangent Function

With the default value of g set to 1, it is very important that the input data would not be very small neither would be very large as these extreme values do not contribute in nonlinear approximation.

4.2.3. The Back-Propagation Algorithm

The back-propagation method is a technique for adjusting network weights in order to minimise the cost or the energy function. Back-propagation method has two parts: the error calculation, and the learning of the network. There are several training algorithms that follow the general principals of the back-propagation methodology. Levenberg-Marquardt (LM) algorithm is a standard technique used to solve nonlinear least square minimization problems. LM was designed to approach second-order training speed without having to compute

the Hessian matrix which represents the second derivative of the energy function matrix.

The LM curve-fitting method is actually a combination of the gradient descent and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the direction of the greatest reduction of the least squares objective. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic, and finding the minimum of the quadratic. The LM method acts more like a gradient-descent method when the parameters are far from their optimal value and acts more like the Gauss-Newton method when the parameters are close to their optimal value [13]. Since the training algorithm is the most critical part of the neural network, it is very important to spent some time to present the basic principals of the logic behind this algorithm.

Let first introduce the cost function c so that

$$c(\vec{w}) = \frac{1}{2}\sum_{i=1}^{p} e_i^2$$

where e is the error of the network, p is the number of patterns, and w represents the weights.

Let also assume that the networks has only one neuron in the output layer and that the network is being trained in batch mode, meaning that the weights are altered once all the patterns are introduced to the network. This means in each step i=1,2,n=p the networks is realizing the error coming as a result of the input vector I, and then, after n observations, the weight are being updated based on the minimum of the half of the squared Euclidian norm of the error vector.

Let also introduce the function of the updated error as follows:

$$e'(i, \vec{w}) = e(i) + \left[\frac{\partial(e_i)}{\partial \vec{w}}\right](\vec{w} - \vec{w}(n))$$

Setting

$$\vec{e}(n) = [e(1), e(2), ..., e(n)]$$

Then,

$$\vec{e}'(n, \vec{w}) = \vec{e}(n) + \vec{J}(n)(\vec{w} - \vec{w}(n))$$

where the Jacobian matric J is the transposed matrix of the first derivative of the errors with respect to weights.

As mentioned in the previous paragraph, after n observations, the weights of the network are updated based on the function:

$$\vec{w}(n+1) = \min\left\{\frac{1}{2}\|e'(n,\vec{w})\|^2\right\}$$

This equality implies that the new value of weight vector is based on the w vector that minimizes the half of the square Euclidian norm of the error vector which is equal to:

$$\frac{1}{2}\|e'(n,\vec{w})\|^2 = \frac{1}{2}\|\vec{e}(n)\|^2 + \vec{e}^T\vec{J}(n) + \frac{1}{2}(\vec{w}-\vec{w}(n))^T\vec{J}^T(n)\vec{J}(n)(\vec{w}-\vec{w}(n))$$

Next follows the differentiation of the equation with respect to w and setting the result to zero.

$$\vec{J}(n)\vec{e}(n) + \vec{J}^T(n)\vec{J}(n)(\vec{w}-\vec{w}(n)) = 0 \Rightarrow \vec{J}^T(n)\vec{J}(n)(\vec{w}-\vec{w}(n)) = -\vec{J}(n)\vec{e}(n)$$

Assuming that the matrix

$$\vec{J}^T(n)\vec{J}(n)$$

is non-singular, multiply both parts of the equation with the inverse of this matrix.

$$(\vec{w}-\vec{w}(n)) = -(\vec{J}^T(n)\vec{J}(n))^{-1}\vec{J}^T(n)\vec{e}(n) \Rightarrow \vec{w}(n+1) = \vec{w}(n) - (\vec{J}^T(n)\vec{J}(n))^{-1}\vec{J}^T(n)\vec{e}(n)$$

The last equation is called the Gauss-Newton method in its pure form.

As mentioned earlier, to inverse

$$\vec{J}^T(n)\vec{J}(n)$$

which is an approximation of the Hessian matrix,

$$\vec{J}^T(n)\vec{J}(n)$$

should be non-singular, that is

$$rank(\vec{J}^T(n)\vec{J}(n)) = n$$

To avoid rank deficiency, let introduce the positive constant λ and the identity matrix I(n) so that :

$$rank(\vec{J}^T(n)\vec{J}(n) + \lambda I(n)) = n$$

Then, the modified Gauss-Newton method will be:

$$\vec{w}(n+1) = \vec{w}(n) - (\vec{J}^T(n)\vec{J}(n) + \lambda\vec{I}(n))^{-1}\vec{J}^T(n)\vec{e}(n)$$

Based on this method, [22] introduced an algorithm that increases or decreases the constant λ in such a manner that the algorithm interpolates between the Gauss-Newton and the gradient descent method. There are two key points in this algorithm. The first is that the computation complexity of the calculation of the table of the second derivatives of the error term is being avoided as the algorithm requires just an approximation of the Hessian matrix. That is to say that having the function:

$$\nabla^2 c(n) = \vec{J}^T(n)\vec{J}(n) + \sum_{i=1}^{n} e_i(n)\nabla^2 e_i(n) \Rightarrow \vec{H} \approx \vec{J}^T(n)\vec{J}(n)$$

where the residuals

$$e_i(n)$$

and the substance

$$\nabla^2 e_i(n)$$

are omitted due to the very small value.

The second point is that the term λ leverages the algorithm from the process of defining the value of the learning rate and the momentum. In the case of λ there are some predefined rules of the value that this constant takes. A reduction in the error after the update of the weights reduces the constant by a predefined factor. The opposite increases the λ by the same factor. In addition, if λ is equal to zero, then the algorithm behaves like Newtons method, while for large values of λ, the algorithm behave like the gradient descent. One disadvantage of having high values of λ is that the Hessian matrix is not taken into account. Identifying this weakness, [25] altered the Gauss-Newton modified form so that

$$\vec{w}(n+1) = \vec{w}(n) - \left(\vec{H} + \lambda\vec{H}\right)^{-1}\vec{J}^T\vec{e}(n)$$

Since the Hessian matrix is proportional to the curvature of the cost function, the previous equality implies that a large step in the direction when that curvature is low, and a small step in the direction with high curvature [28].

4.2.4. ANN Topology

The choice of number of layers and the neurons inside each layer is very crucial factor to be considered when designing neural networks. The number of units in the input and the output layer is dictated by the solution itself. The proposed model has 20 input nodes and 1 output node. While the definition of the input and output layer is quite straightforward, things are very different when trying to define the number of the hidden layers and the units for each such layer.

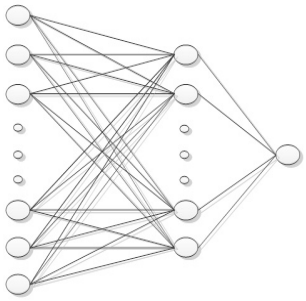


Figure 10: ANN Topology

According to [18] one or two hidden layers is enough to approximate any smooth bounded function. Meanwhile, introducing more hidden layer to the network it make the training more difficult as large networks are hard to train properly and fall victims to over-fitting quite frequently. There is no magic formula for selecting the optimum number of hidden neurons . There are three basic approaches. The first is the so called brute force search that implies that a range of different topologies are created and tested and the choice is based on some criteria such as the networks error etc.

A second solution is the so called method of thumb where there are some basic rules saying for instance that the number of hidden layers are somewhere between the number of input and the number of outputs, or that the hidden neurons are equal to the 75% of the number of neurons in the input layer[3]. It is obvious that this is an approximation of the solution that needs more refinement such as combining it with the third option. The third option is based on the genetic algorithms and their ability to find the global minima by searching at the same time in many directions as opposed to the gradient decent. The next section presents in more details the structure of the genetic algorithms and their applicability to find the optimal neural network topology.

4.3. Genetic Algorithms

Genetic Algorithms (GA), developed by [17], is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals or chromosomes to evolve under predefined selection rules to a state that maximizes the fitness [15]. A typical GA has the structure depicted by the following flowchart.

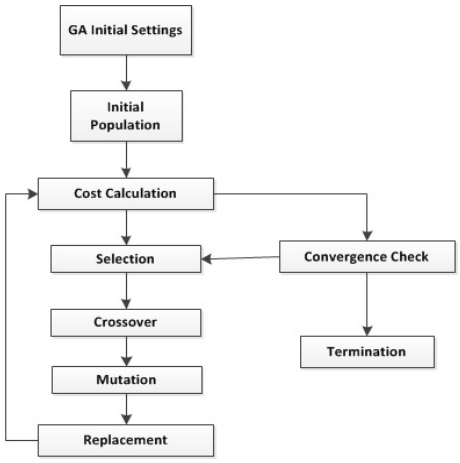


Figure 11: Genetic Algorithm Structure

The algorithm mimics the natural selection in an iterative process that where a population of potential solutions is evolving so that the optimal solution is found. Once the iteration has run the next generation is produced by selecting and processing the most-fit individuals based on their fitness. Each member of the population represents knowledge by a number of genes that form the chromosome. The next section presents the definition of the population as well as the fitness function.

4.3.1. Chromosome, Genes, and Fitness Function

To give some mathematical, let

$$n$$

be the number of the population where each individual represents a chromosome for

$$i = 1...n$$

and let

$$m$$

be the number of genes in each chromosome for

$$j = 1...m$$

where

$$x_i = \{z_1 ... z_m\}$$

Additionally, consider

$$z_j \in N = \{a...b\}$$

Each

$$x_i \in \bigcup_{j=1}^{m} z_j$$

represents a candidate solution to the problem

$$\min f(x_i); x \in \bigcup_{j=1}^{m} z_j \forall z_j \in N = \{a, ..., b\}$$

where the function

$$f$$

is defined over the range of natural numbers

$$N = \{a, ..., b\}$$

Therefore, the task of the genetic algorithm is to minimize the

$$f$$

looking at the space

$$N = \{a, ..., b\}$$

The

$$f$$

is the fitness function and the outcome of the function is the fitness value of the input argument that represents each individual member of the population.

An important factor is the representation of each individual or chromosome. The encoding of the chromosome is dictated by the solution. As far as the topology optimization problem is concerned, the chromosome can be encoded in integer or in binary (or bit-string) format. Adopting one rule of thumb about the number of hidden neurons in an ANN, in this specific solution and based on the mathematical notations presented previously, a=0, b=15 and m=2 as this is the maximum number of hidden layers of the developing ANN. Therefore, a potential structure of the chromosome could be formatted by concatenating two genes each consists of an integer which represents the number of neurons in a hidden layer. This solution makes easier the execution of the fitness function which is presented in the next pseudo-code chunk.

Fitness_Function(chromosome)
hidden_layer_1 **:=** chromosome(0)
hidden_layer_1 **:=** chromosome(1)
input_layer **:=** 20
output_layer **:=** 1
ann **:= CreateAnn**(hidden_layer_1, hidden_layer_2, input_layer, output_laye)

ann := **TRAIN**(ann)
prediction := **SIMULATE**(ann)
results := **EVALUATE**(prediction)
wrong_predictions := **GET_WRONG_PREDICTIONS**(results)
mae := **mae**(results)
aof := a*mae +b*wrong_predictions
return aof
end

Meanwhile, the best solution for presenting the population to the fitness function, the integer encoding causes difficulties in the process of crossover and mutation. It is clear that a better solution would be for the population to be encoded in binary format. With b=15 this means that each gene has (2 ^4=16) 4 bits and with m=2, the chromosome comprises of 8 bits. Therefore, the population is encoded to binary format and is transformed to integer format inside the fitness function.

Having defined the structure of the chromosome, the next step is to define the initial population. According to [24], despite the required processing power, a large initial population may provide a faster convergence. Of course this relies on the size of the search area. Here the initial population is set to 30 individuals and the generation number is set to 40.

Another important factor is the way the initial population is produced. A fully randomly generated initial population may get trapped to local minima while an ad-hoc method maybe too biased and direct the solution to a specific area. However a combination of those solutions seems to be a reasonable compromise.

4.3.2. Selection

Selection can be best described as the process that provides the fitness of each individual number, decides which individual survives for the next generation. There are several algorithms that execute this process. One such example is the roulette-wheel selection [19].

Provided the initial population

$$n$$

the algorithm calculates the cumulative fitness

$$F = \sum_{i=1}^{n} f(x_i)$$

for each individual

$$x_i$$

Then each chromosome is assigned the selection probability

$$f(x_i)/F$$

A portion of the individuals with the highest probability passes directly to the next generation without any changes as an elitism policy which in this solution is set to 1 individual. The rest of the selected population is applied a stochastic process to identify the population that is going to be applied the next stage which is the crossover process.

4.3.3. Crossover

Crossover is the process of choosing a portion of the population with fitness value probably better than the average, and creating pairs that exchange information [4]. The basic idea here is that the parents with a high score may produce children with even higher score. It is the same like a couple where both man and woman are tall and where the obvious answer is that their child is going to be at least as tall as their parents.

Crossover is the central feature of chromosomes. This is the reason why in most of the implementations the crossover parameter is quite high, saying between 50 and 80% of the population. Another important parameter as far as the crossover process is concerned is the crossover point. Depending on the length of the chromosome as well as the nature of the solution, the chromosome can be split on one or more points. The point though that the chromosome is being split can be a predefined value or can be produced by a stochastic process. This solution adopts the two point crossover operator and the crossover rate is set to 80%.

4.3.4. Mutation

The previous process was based on past information. The new individual combines information from both parents. In the nature, apart from the genes that organism has inherited from their parents, there is also something else that affects the process of the evolution and this is the radiation coming from the Sun or from the environment. It is obvious that this is a complete stochastic process.

Mutation directs a portion of the population to an area close to the existing one but never visited before. The hope is that solutions around an individual with high fitness value may prove to have a better performance. The idea is that if a solution is near to a global minima or global maxima, then a small step may set the solution a step closer to the final solution. This process is stochastic and as such a mutation rate should be defined which in this case is set to 20%.

4.3.5. Termination Criteria

There are two basic termination criteria namely the number of generations and the number of stall generations. In this experiment, the number of generations is set to 40 while the number of stall generations has been set to 15. Therefore, the algorithm cannot create more than 40 generations and if the optimal fitness value does not change for 15 consecutive generations, then the algorithm terminates even if the generation number is less than 40.

4.4. Performance Metrics

Choosing the appropriate performance metrics is crucial for both the development of the forecasting model, and for evaluating different trading strategies as well. There are two basic categories of performance metrics: the traditional performance metrics based on statistics outcomes such as Mean Absolute Error (MAE), Mean Square Error (MSE), Theils Inequality Coefficient (Theils U), and those based on direct measurement of the forecasting model objectives such as Annualised Return, Sharpe Ratio, Cumulative Investment Return, etc. [6] assess the efficiency of the learning and discard bad trained nets by utilizing the MSE while [34] developed a genetic algorithm that evaluate net topology using RMSE as a cost function . Similarly, [12] introduced an algorithm that evaluates ANN performance by adopting a technique that is based on the sum of squared error (SSE). On the other hand, according to [8], traditional standard statistical measures are often inappropriate to assess financial performance. Also, in some cases, different trading strategies cannot be compared with these standard measures for the simple reason that they are not based on forecasting the same nature of output.

When developing a prediction model the first and more important metric is the right direction of the market. Another important issue is the prediction error. Meanwhile, the classic optimization problem focuses on single objective approach. When more than one objective ought to be optimized, then the problem is called multi-objective optimization. This approach requires the introduction of a single aggregate objective function (AOF). There are different techniques for creating an AOF. One intuitive approach to creating such function is the weighted linear sum of the objectives. Therefore, the adopted approach includes the construction of an AOF that consists of the weighted sum of the missing trades and the mean absolute error. The following formula depicts the aforementioned function.

$$aof(x_1, x_2) = b_1 x_1 + b_2 x_2$$

The previous function acts as the objective function of the developed GA that optimizes the topology of the forecasting model. However, evaluation of the model as far as the trading strategy is concerned dictates the adoption of those metrics that express the performance of the system in terms of the profitability.

Mean Absolute Error

$$mae(r) = \frac{\sum_{i=1}^{n} |r_i - \overline{r}|}{n}$$

Annualized return

$$r^A = 252\frac{1}{n}\sum_{i=1}^{n} r_i$$

Sharpe Ratio

$$sr = \frac{\frac{\sum_{i=1}^{n} r_i}{n} - \frac{r_{fix}}{252}}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(r-\bar{r})^2}}$$

Correlation Coefficient

$$corr = \frac{\sum_{i=1}^{n}\left((x_i-\overline{x})(y_i-\overline{y})\right)}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2\sum_{i=1}^{n}(y_i-\overline{y})^2}}$$

Annualized return is the return an investment provides over a period of time, expressed as a time-weighted annual percentage. Sharpe Ratio shows how appealing an investment is. The last row of the table presents the correlation coefficient that provide information about how correlated the predicted and the actual values are.

5. Results and Discussion

The performance of the financial forecasting model depends primarily upon three general factors: the appropriate data processing and presentation, the optimal trading strategies, and the structure of the forecasting model. The first two components have already been analysed. Therefore, once the network topology is being defined, then the model is ready to run the trading strategies.

5.1. Optimal Network Topology

Before presenting the results taken after executing the GA, it is important to define optimal topology. One topology is optimal when satisfies three basic features namely convergence, generalization, and stability. An ANN is meant to converge when all the input patterns have been assimilated. The perfect convergence is when the minimum error of each pattern is almost equal. Generalization means that the network exhibits a good performance when introduced out-of sample data. Finally, stability means that even when the network is being fed with slightly different dataset, the performance remains satisfactory.

For instance, the network has been trained with GBP \USD rates, and then it simulates EUR \GBP rates.

As the GA is a stochastic process by its nature, the algorithm has to be executed several times. Therefore, the algorithm has been executed 5 times for each currency pairs. The following figure depicts the execution of the developed genetic algorithm and particularly the optimization of the network for GBP \USD currency pair.

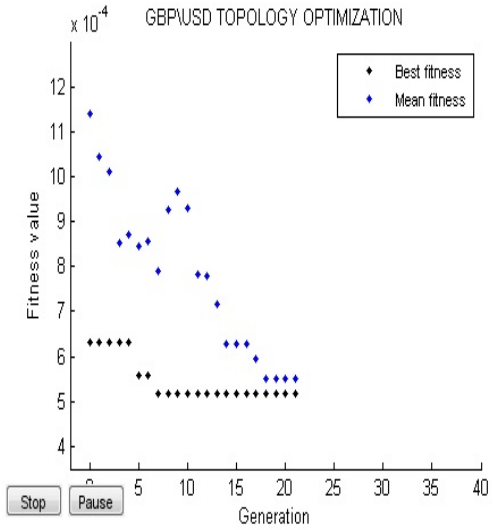


Figure 12: Topology Optimization

The results taken after the execution of the developed GA revealed that topology 20-9-8-1 was the winning topology in four out of the five executions when applied to EUR\GBP. In the case of EUR\USD pair the same topology won three times. On the other hand, GBP\USD optimization indicated topology 20-1-8-1 as the winning topology for three times while the for the other two times the winning topology was 20-9-8-1. The following table shows the results after the fifth execution of the algorithm.

GBP\USD TOPOLOGY	GBP\USD FITNESS	EUR\GBP TOPOLOGY	EUR\GBP FITNESS	EUR\USD TOPOLOGY	EUR\USD FITNESS
10011000	0.0004	10011000	0.0003	00111100	0.0005
00011000	0.0005	10000100	0.0003	10000100	0.0007
01010011	0.0007	10011000	0.0003	00110100	0.0008
01100001	0.0007	10011000	0.0003	01100001	0.0008
00011110	0.0008	10011000	0.0003	00110100	0.0008
10010001	0.0009	00111100	0.0006	01101100	0.001
10110010	0.0009	00111100	0.0006	00100101	0.001
00110000	0.0009	01010011	0.0006	00100000	0.0011
01101001	0.0009	00110100	0.0008	00110101	0.0011
01110001	0.0009	00100000	0.0009	00010000	0.0012
00110010	0.001	00100001	0.0009	00000001	0.0012
01010101	0.001	10010011	0.001	00010100	0.0012
01000001	0.001	00110000	0.001	00010100	0.0012
10001010	0.0011	10111101	0.001	00000001	0.0012
00001011	0.0011	00011000	0.001	00100100	0.0013
11010000	0.0011	10011010	0.0011	00100100	0.0013
00001101	0.0011	00000001	0.0012	01100101	0.0013
10000001	0.0011	00010000	0.0012	00100100	0.0013
00011101	0.0012	01010000	0.0013	01101101	0.0013
11000000	0.0012	10001000	0.0013	01110010	0.0014
00001111	0.0012	01110000	0.0013	11110101	0.0015
11000000	0.0012	10011101	0.0014	10000101	0.0015
11000100	0.0013	00010001	0.0014	11111110	0.0017
11000100	0.0013	10010000	0.0015	00110001	0.0017
10001000	0.0014	10010000	0.0015	00001100	0.0017
01100000	0.0015	10101000	0.0016	00001010	0.0018
00011010	0.0015	00010011	0.0016	00000100	0.0021
00000100	0.0015	10011100	0.0018	10011100	0.0021
01001011	0.0016	00011001	0.0019	10010100	0.0022
00011001	0.0018	00011010	0.002	01100000	0.0025

Table 3: Winning Topologies

Notes: In the topology column, every four bits of the binary number represents the number of the neurons in the corresponding hidden layer.

Therefore, provided the results from the execution of the algorithm, the topologies under investigation are the topologies 20-9-8-1 and 20-1-8-1. Table 5 presents the fitness value and the correlation of each of the testing topologies applied both in in-sample and out-of sample data.

Currency Pair	**20-9-8-1**		**20-1-8-1**	
	Fitness	**Correlation**	**Fitness**	**Correlation**
In-Sample Test				
GBP\USD	0.00038	0.8212	0.0034	0.7348
EUR\GBP	0.00026	0.7302	0.00054	0.5413
EUR\USD	0.00039	0.7647	0.00077	0.2776
Out-Of-Sample Test				
GBP\USD	0.00029	0.8479	0.0029	0.8351
EUR\GBP	0.0005	0.8307	0.00047	0.8444
EUR\USD	0.00026	0.88	0.00027	0.8786

Table 4: Fitnes Function Results

The testing of both topologies with respect to the datasets of the other currency pairs both in in-sample and out-of-sample datasets revealed that the topology 20-9-8-1 is the most stable and efficient, and also outperforms in the case of the out-of-sample dataset apart from EUR\GBP currency pair as in this case the error is higher than when applied in-sample dataset.

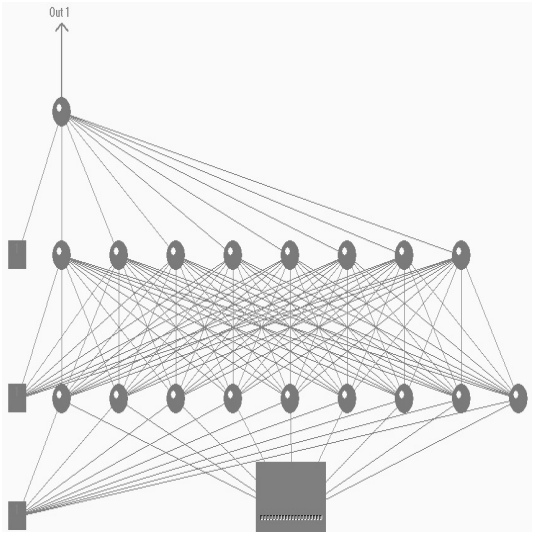


Figure 13: Winning Topology

The previous image depicts the structure of the aforementioned topology. The graph below shows the performance of this particular topology during the training testing and validation stage using in-sample data corresponding to EUR\GBP currency rates.

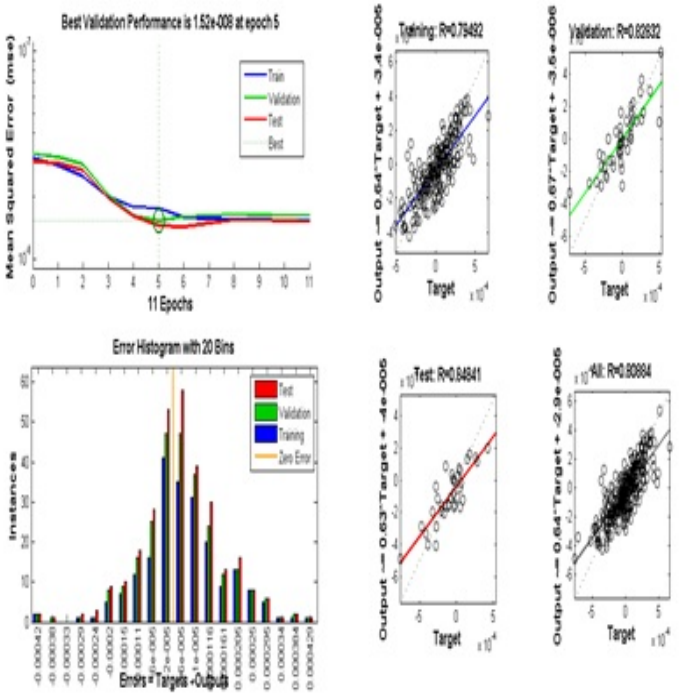


Figure 14: The Performance of the Optimal Topology

Therefore, as the only topology to have satisfied all the predefined conditions, the topology 20-9-8-1 is the optimal topology for the given forecasting model.

5.2. Trading Results

As commented earlier, while mean square error is an acceptable measure for performance, in practice, the ultimate of any testing strategy is to confirm that the forecasting system is to producing positive figures. There are several metrics available for this purpose. The following table contains the average results of the strategies corresponding to the three base currencies.

Strategy	Winning Rate	Hit-Miss Rate	Annualized Gross Return	Sharp Ratio
GBP as the Base Currency				
S1	72.5%	2.6	26.9%	0.7
S2	72.5%	2.6	24%24%	0.62
S3	72.5%	2.6	26.9%	0.7
EUR as the Base Currency				
S1	71,25%	2.48	27.7%	0.6
S2	75%	3	32.33%	0.7
S3	71.25%	2.48	27.7%	0.6
USD as the Base Currency				
s1	71.25%	2.48	27.9%	0.54
s2	70%	2.33	28.9%	0.56
s3	71.25%	2.48	27.9%	0.54

Table 5: Trading Results

The table reveals that the proposed model produces a quite promising profit with an average annualised gross profit at 27.8%. Meanwhile, an important issue that has not been mentioned so far is the trading cost. While there is no direct transaction fee, the bid-ask spread is a kind of indirect charge. The testing dataset showed that the average bid-ask spread during the proposed trading timespan is 3 pips, and the average profit is 36 pips. As the spread affect affects both trading directions, the average trading cost is 16%. This gives an annualized net profit at 23.3% which is in many cases better than that of the co-operate earnings although the risk in FOREX trading is substantially higher.

Another reason that makes the FOREX trading more appealing this period is the very low level of interest rates which has a great influence in fix income products such as government bonds. This is one of the reasons that makes the investment is more risky products more appealing.

A rather reasonable question at this point is what the results of the proposed model would be if conducting Monde Carlo simulation. The next table show exactly those results in comparison with the real data.

	GBP\USD	**EUR\GBP**	**EUR\USD**
Proposed Model	72.50%	72.50%	70%
Random Walk	42.50%	49%	51.60%

Table 6: Monde Carlo Simulation

As the previous table shows, the proposed model has a very good performance in comparison with the random walk model. The following graph shows the performance of the real data on the left graph, and the random time series on the right graph. This experiment concerns the GBP\USD rates.

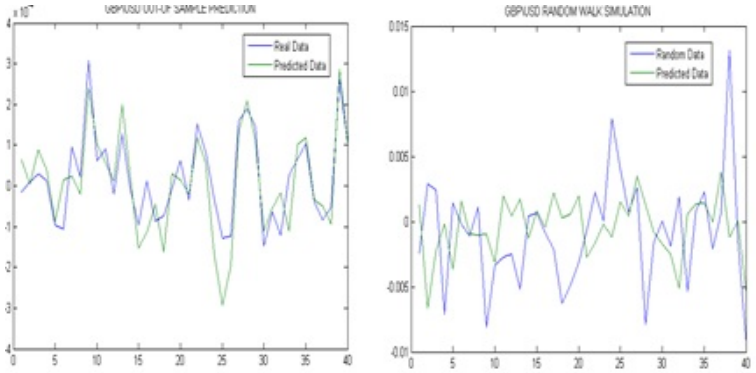


Figure 15: Monde Carlo Simulation

Furthermore, the model appears to have also a good performance in comparison with other proposed models. [9] conducted a comparison between technical

analysis, econometrics, and an ANN Regression techniques as FX forecasting models. The MACD model gave an annualised return at about 4.54% and a winning trades at 30.85% while the ARMA model gave 12.65% and 50.24% respectively. The NNR model outperforms both models by producing an annualized return rate of 19.02% and 48.14 winning rate.

[33] compared the performance of Feed-forward Neural Networks (FNN) and Probabilistic Neural Networks (PNN) models in classifying trade signals into three classes: buy, hold, and sell. Both models produced a correct classification rate of approximately 20% with the PNN model to have shown a slightly better performance. Finally, [31] present an approach to autonomous agent design that utilizes genetic algorithm and genetic programming in order to identify optimal trading strategies. The competing agents recorded an average sharp ratio between 0.33 and 0.85.

6. Conclusions and Future Work

The main purpose of this paper was to introduce a prediction and decision making model that produces profitable intra-day FOREX transactions. The paper has shown that, despite the highly noisy nature of the tick FOREX data, proper analysis and pre-processing can identify repeatable patterns that provide a reliable source for developing a forecasting model. The developed forecasting model is based on Feed Forward Neural Networks with Back-Propagation architecture. Furthermore, a GA was developed to search for the optimal network topology. Several experiments identified the topology 20-9-8-1 as the one that provides the network with better approximation and generalization. Additionally, the proposed trading strategies were able to produce a promising annualized net profit of 23.3% which makes FOREX algorithmic trading an appealing choice.

Back-propagation is a well implemented technique that has dominated the prediction models over the previous many years. Since the introduction by Vapnik in 1995, many argue that Support Vector Machines (SVM) are more robust and tend to provide more accurate models. Considering the market prediction problem as a classification problem where the question is whether the market goes up or down, an interesting question is what the performance of SVM would be in comparison with back-propagation model both operating in a highly noisy data environment such as the FX market.

7. References

[1] Altridge, I., 2010. High Frequency Trading. John Wiley & Sons, NY.

[2] Azzof, M., 1994. Neural Network Time Series Forecasting of Financial Markets. John Wiley &.

[3] Baily, D., Thompson, D., 1990. Developing neural network application. AI Experts, 33–44.

[4] Barolli, L., Spaho, E., Oda, T., Barolli, A., Xhafa, F., Takizawa, M. (Eds.), 2011. Performance Evaluation for Different Settings of Crossover and Mutation Rates Considering the Number of Covered Users: A Case Study. ACM, NY.

[5] Butler, M., Daniyal, A. (Eds.), 2009. Multi-objective Optimization with an Evolutionary Artificial Neural Network for Financial Forecasting. ACM, Mondreal Quebec Canada.

[6] Castiglione, F., 2001. Forecasting price increments using an artificial neural network. Advances in Complex Systems 4, 45–56.

[7] Chante, S., 1997. Beyond Technical Analysis: How to Develop and Implement a Winning Trading System. Wiley, NY.

[8] Dunis, C., Jalilov, J., February 2001. Neural network regression and alternative forecasting techniques for predicting financial variables. online.
URL `www.livjm.ac.uk`

[9] Dunis, C., Laws, L., Nai, P., 2003. Applied Quantitative Methods for Trading and Investment. Wiley, NY.

[10] Enam, A. (Ed.), 2008. Optima Artificial Neural Network Topology for Foreign Exchange Forecasting. ACM, NY.

[11] Fama, E., 1970. Efficient capital markets: A review of theory and empirical work. Journal of Finance 25, 383–417.

[12] Gao, P., Chen, C., Qin, S., 2010. An optimization method of hidden nodes for neural network. In: Proc. Second Int Education Technology and Computer Science (ETCS) Workshop. Vol. 2. pp. 53–56.

[13] Gavin, H., 2011. The levenberg-marquardt method for nonlinear least squares curve fitting problems. online.
URL `www.duke.edu`

[14] Giles, C., Lawrence, S., Tsoi, S., 2001. Noisy time series prediction using a recurrent ann. Journal of Machine Learning 44.

[15] Haupt, R., Haupt, S., 2004. Practical Genetic Algorithm. Wiley, New Jersey.

[16] Hirabayashi, A., Arahna, C., Iba, H., 2009. Optimization of the trading rules in forex using genetic algorithms. online.
URL www.dollar.biz.uiovwa.edu

[17] Holland, J., 1975. Adaptation of Natural and Artificial Systems. University of Michigan Press, Michigan.

[18] Kaastra, I., Boyd, M., 1995. Designing a neural network for forecasting financial and economic time series. online.
URL www.seas.harvard.edu

[19] Kalyvas, E., 2001. Using neural networks and genetic algorithms to predict stock market return. online.
URL www.citeseerx.ist.psu.edu

[20] Kamruzzaman, J., 2004. Ann-based forecasting of foreign currency exchange rates. ACM, Cambera, Australia.

[21] Kaufman, J., 1998. Trading Systems and Methods. Wiley, NY.

[22] Levenberg, K., 1944. A method for the solution of certain problems in least squares. Quarterly of Applied Mathematics 2, 164–168.

[23] Makridakis, S., Wheekwright, C., Hyndman, R., 1998. Forecasting: Method and Applications. John Wiley & Sons, NY.

[24] Man, K., Tang, K., Kwong, S., 1999. Genetic algorithms: Concept and design. Springer-Verlag, Heidelberg.

[25] Marquardt, D., 1962. An algorithm for least squares estimation of nonlinear parameters. SIAM Journal of Applied Mathematics 11, 431–441.

[26] Martinez, L. (Ed.), 2009. From an Artificial Neural Network to a Stock Market Day Trading System: A Case Study on BM&F Bovespera. IEEE, Atlanda, Gorgia, USA.

[27] Pardo, R., 1992. Ddesign, Testing, and Optimization of Trading Systems. Wiley, NY.

[28] Ranganathan, A., 2004. The levenberg-marquardt algorithm. online.
URL www.citeseerx.ist.psu.edu

[29] Refenes, P., 1995. Neural Networks in the Capital Markets. Wiley.

[30] Scabar, A. (Ed.), 2002. Financial Trading and the Efficient Market Hypothesis. Vol. 4. ACM, Darlinghurst, Australia.

[31] Subramanian, H., Ramamoorthy, S., Stone, P., Kuipers, L. (Eds.), 2006. Designing Safe, Profitable Automated Stock Trading Agents Using Ecvolutioanary Algorithms. ACM, NY.

[32] Tay, H., Cao, L., 2001. Application of svm in financial time series. online.
URL www.zenithlib.googlecode.com

[33] Tilakarante, C., Morris, S., Mammadov, M., Hurst, C. (Eds.), 2008. Predicting Stock Market Index Trading Signals Using Neural Networks. Springer-Verlag, Berlin.

[34] Trujillo, L. (Ed.), July 2011. How Many Neurons? A Genetic Programming Answer. ACM, Dublin, Ireland.

[35] Tsang, P., 2006. Design and implementation of ann5 for hong kong stock price forcasting. online.
URL www.sciencedirect.com

[36] Tulson, D., Tulson, S., 2007. Intelligent financial systems. online.
URL www.if5.com

[37] Vastone, B., Finnie, G., 2009. An empirical methodology for developing stockmarket trading systems using artificial neural networks. Expert Systems and Appliactions 35.

[38] Yao, J., Tan, C., 2000. A case study on using neural networks to perform technical forecasting of forex. online.
URL http://citeseerx.ist.psu.edu

[39] Yu, L., Wang, S., Lai, K., 2004. A novel nonlinear ensemble forecasting model incorporating glar & ann for foreign exchange rates. Computers and Operations Research 32.

[40] Yu, L., Wang, S., Lai, K., 2007. An online learning algorithm with adaptive forgetting factors for forward neural networks in financial time series forecasting. online.
URL http://citeseerx.ist.psu.edu

[41] Zhang, H., Ren, R., 2010. High frequency foreign exchange trading strategies based on genetic algorithms. In: Proc. Second Int Networks Security Wireless Communications and Trusted Computing (NSWCTC) Conf. Vol. 2. pp. 426–429.