Supply Chain and Finance, pp. 283-298
P.M. Pardalos, A. Migdalas and G. Baourakis, Editors
World Scientific Publishing, River Edge, NJ, U.S.A., 2004

CHAPTER 17

EXCHANGE RATE FORECASTING THROUGH DISTRIBUTED TIME-LAGGED FEEDFORWARD NEURAL NETWORKS

N.G. Pavlidis

Department of Mathematics,
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras,
GR–26110 Patras, Greece.
E-mail: npav@math.upatras.gr

D.K. Tasoulis

Department of Mathematics, UPAIRC,
University of Patras,
GR–26110 Patras, Greece.
E-mail: dtas@math.upatras.gr

G.S. Androulakis

Computer Technology Institute (CTI),
UPAIRC,
University of Patras,
GR–26110 Patras, Greece.
E-mail: gsa@math.upatras.gr

M.N. Vrahatis

Department of Mathematics, UPAIRC,
University of Patras,
GR–26110 Patras, Greece.
E-mail: vrahatis@math.upatras.gr

Throughout the last decade, the application of Artificial Neural Networks in the areas of financial and economic time series forecasting has been rapidly expanding. The present chapter investigates the ability of Distributed Time Lagged Feedforward Networks (DTLFN), trained through a popular Differential Evolution (DE) algorithm, to forecast the

short–term behavior of the daily exchange rate of the Euro against the US Dollar. Performance is contrasted with that of focused time lagged feedforward networks, as well as with DTLFNs trained through alternative algorithms.

Keywords: Artificial neural networks, differential evolution algorithms, time series prediction.

1. Introduction

A central problem of science is forecasting; how can knowledge of the past behavior of a system be exploited in order to determine its future evolution. As of 1997 the foreign exchange market constitutes the world's largest market with daily transactions surpassing 1 trillion US dollars on busy days. More than 95 percent of this volume is characterized as speculative trading, i.e. transactions performed in order to profit from the short term movement of the exchange rate.

Two schools of thought compete in the field of financial forecasting, fundamentalists and technical analysts. Fundamentalists hold the view that forecasting needs to be based on the identification of a model that approximates the true underlying exchange rate determination dynamics. Thus, models that take into account inflation and interest rate differentials, balance of payments accounts and numerous other economic indicators, are constructed and evaluated. A key limitation of this approach is that most of the involved quantities are known only *a posteriori*. The scope of this approach, therefore, lies more within the realm of justification rather than prediction. In contrast, technical analysts exploit information contained in the history of prices, in market news, as well as in different technical indicators in order to form their expectations about future prices. A central limitation of this approach stems from the well–known Efficient Market Hypothesis (EMH) that states that all information concerning future values of financial assets traded in competitive markets is already incorporated in current prices. The best forecast for the future value is therefore the current value. Yet, as it has been shown in Ref. 4 for market participants to be willing to participate in the market, prices cannot reflect all available information; paradoxically, if that was the case there would be no reason for a market to exist, thus, at least the strong form of the EMH has to be violated. This however does not alter the fact that inferring the future evolution of market prices is a particularly hard problem. Indeed had prices been predictable to a large extent, once more there would be no incentive for market participants to enter the market.

The literature on artificial neural network (ANN) applications in the fields of financial time series has been rapidly expanding throughout the past decade.[11] Several researchers have considered the application of ANNs on the problem of foreign exchange forecasting with promising results.[1,3,8,9,17] The ability of ANNs to outperform different linear models[19] as well as the random walk model, using solely publicly available information, may be taken to imply that a certain structure does in fact exist, and most importantly, that ANNs, if properly designed and trained, can identify and exploit this structure in order to produce accurate forecasts.

The present chapter contributes further in this line of research, by investigating the ability of a particular type of feedforward networks, Distributed Time Lagged Feedforward Networks (DTLFN)[7,18] to forecast the time series of the daily exchange rate of the Euro against the US Dollar (USD). Particular emphasis is attributed to the prediction of the direction of change of the spot exchange rate, since this information is sufficient to render speculative trading profitable. The task at hand is particularly difficult due to the limited number of available observations, the highly competitive nature of the particular market, the noise and finally, the nonstationarity present in the data. A key merit of ANNs is noise tolerance; i.e. the ability to identify patterns in data contaminated with noise. DTLFNs are further characterized by the highly desirable property that they can cope effectively with nonstationarity, unlike static multilayer feedforward networks. The best results obtained were achieved using a novel global optimization technique, called Differential Evolution (DE) algorithm, introduced in Ref. 16 and implemented in the context of ANN training in Ref. 10, in combination with a modified error performance function.

The chapter is organized as follows; Section 2 discusses focused and distributed time–lagged feedforward networks, and also provides a brief introduction to the workings of the DE algorithm. Section 3 presents the empirical results and the Section 4 is devoted to concluding remarks and future work.

2. Artificial Neural Networks

2.1. *Focused Time-Lagged Feedforward Neural Networks*

Artificial Neural Networks (ANNs) are parallel computational models comprised of densely interconnected, simple, adaptive processing units, and characterized by a natural propensity for storing experiential knowledge and rendering it available for use. ANNs resemble the human brain in two

fundamental respects; firstly, knowledge is acquired by the network from its environment through a learning process, and secondly, interneuron connection strengths, known as synaptic weights are employed to store the acquired knowledge.[6,7]

The building block of an ANN is the *artificial neuron*, which consists of a number n of synapses, a weight vector w, an activation function f, and finally output y. Each element of the weight vector w_i corresponds to an input x_i, $i = 1, ..., n$ and the primary operation of an artificial neuron, consists of summing up the products of the inputs and their corresponding weight. This sum, also known as *excitation level*, constitutes the argument of the activation function, which in turn, determines the output of the neuron. It should be noted that a bias term is included by incorporating an additional weight b whose corresponding input is always set to $x_0 = 1$. In general, the operation of a single neuron receiving as input the vector x, is summarized by the following equations:

$$\xi = \sum_{i=0}^{n} w_i x_i, \qquad y = f(\xi).$$

The most frequently encountered types of activation functions are:

(a) the *hard limit* function:

$$f(\xi) = \begin{cases} 1, & \text{if} \quad \xi \geqslant h, \\ 0, & \text{if} \quad \xi < h, \end{cases}$$

(b) the *piecewise linear* function:

$$f(\xi) = \begin{cases} 1, & \text{if} \quad \xi \geqslant 1, \\ 0, & \text{if} \quad \xi \leqslant 0, \\ \xi, & \text{if} \quad 0 < \xi < 1, \end{cases}$$

(c) the *standard (logistic) logsig* function:

$$f(\xi) = \frac{1}{1 + e^{-\xi}},$$

(d) the *hyperbolic tangent* function:

$$f(\xi) = \tanh \frac{\xi}{2} = \frac{1 - e^{-\xi}}{1 + e^{-\xi}}.$$

Combining a set of neurons, extended ANNs with different topologies, can be created. *Feedforward Neural Networks* (FNNs), are ANNs in which neurons are organized in layers and no feedback connections are established.

In FNNs, inputs are assigned to sensory nodes that comprise the input layer, while the output of the network is produced by the neurons that form the output layer. All other neurons are assigned to intermediate, hidden, layers. The output of a neuron in layer j becomes an input of all the neurons that belong to the next layer, $j + 1$, as shown in Figure 1.

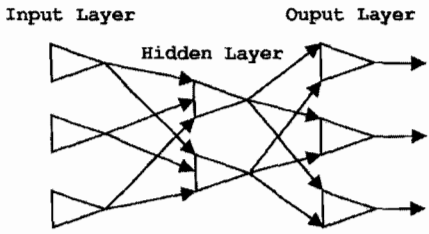

Fig. 1. Feedforward Network

The operation of such networks consists of a series of iterative steps. At the beginning the states of the input layer neurons are assigned to the elements of the input vector, also called pattern vector, and the remaining hidden and output layer neurons are passive. In the next step the neurons of the first hidden layer collect and sum their inputs and compute their output. This procedure is propagated forward through the layers of the FNN until the final network output is computed.

The computational power of FNNs is based on the fact that they can adapt to a specific training set. According to the *universal approximation theorem*,[2] a FNN composed of neurons with nonlinear activation functions and a single hidden layer is sufficient to approximate an arbitrary continuous function. Assuming, as in the case of time series prediction, that the input vector for the network consists of a number of delayed observations of the time series, and the target is the next value, then the *universal myopic mapping theorem*[14,15] states that any shift-invariant map can be approximated arbitrarily well by a structure consisting of a bank of linear filters feeding a static ANN. This type of ANN is known as Focused Time–Lagged Feedforward Neural Networks (FTLFN).

In this context, a training set T of P patterns is defined as:

$$T = \left\{ (x_k, d_k) \left| \begin{array}{l} x_k = (x_t, ..., x_{t-n}) \in R^n \\ d_k = x_{t+1} \in R \end{array} \right. , \quad k = 1, \ldots, P \right\},$$

where x_k represents the kth training pattern, and d_k, the desired response

vector for this pattern. The purpose of training is to assign to the free parameters of the network W, values that will minimize the discrepancy between network output and desired response. The training process starts by presenting all the patterns to the network and computing a total error function E, defined as:

$$E = \sum_{k=1}^{P} E_k,$$

where E_k is the partial network error with respect of the kth training pattern, defined as:

$$E_k = \frac{1}{2}\Big(y_i(W, x_k) - d_{ki}\Big)^2.$$

Each full pass of all the patterns is called a training epoch. If the error performance value drops below the desired accuracy, then it is obvious that the aim of the training algorithm has been fulfilled and the algorithm is terminated. Thus supervised training is a non-trivial minimization problem.

$$\min_{W} E(W)$$

Training a FNN, the information content of the training set is stored in the synaptic weights, forming the long term memory of the network. Once a network is trained the acquired knowledge can be readily exploited. The ability of an ANN to respond correctly to patterns not encountered during training is called generalization. A satisfactory performance on the training set coupled with a poor generalization performance is known as overfitting, or, overtraining. Overfitting implies that the network has adjusted too much on the training set, thereby capturing potential inaccuracies and errors that might exist. Clearly, for the task of time series prediction generalization is of critical importance, since satisfactory performance on the known part of the time series is of little use by itself.

2.2. *Distributed Time–Lagged Feedforward Neural Networks*

The universal myopic theorem previously stated, is limited to maps that are shift invariant. An immediate implication of this limitation is that FTLFN are suitable for modeling stationary processes[7]. As discussed in the following section, using FTLFNs to forecast the future behavior of the exchange rate of the Euro against the USD, frequently results in overfitting and rarely produces a satisfactory performance. A possible alternative in order

to overcome this limitation is to distribute the impact of time throughout the network and not only at the input end. To this end, *Distributed Time Lagged Feedforward Networks* (DTLFN)[18] are considered. The construction of such a network is based on the model of an artificial neuron known as *multiple input neuronal filter*. In this setting the output of each neuron in layer j is inserted in a tapped delay line memory of order m. In effect this line reproduces the m previous values of the output of the neuron and transmits them, as well as the current value, to all the processing units in layer $j+1$. An alternative way to envisage this operation is to imagine that the output of the tapped delay line and the current output are fed to a linear *Finite Impulse Response* filter whose output is in turn transmitted unaltered to neurons in the succeeding layer. The operation of a multiple input neuronal filter is summarized by the following equations:

$$y = f(\sum_{i=1}^{z} \sum_{l=0}^{m} w_{il} x_{i-l} + b).$$

where z stands for the number of inputs of the neuron, and m denotes the order of the tapped delay line.

2.3. *Differential Evolution Training Algorithm*

In a recent work, Storn and Price [16] have presented a novel minimization method, called Differential Evolution (DE), designed to handle non–differentiable, nonlinear and multimodal objective functions. DE exploits a *population* of potential solutions to probe the search space. At each iteration of the algorithm, mutation and crossover are applied in order to obtain more accurate approximations to a solution.

To apply DE to neural network training the approach introduced in Ref. 10 is adopted. Primarily, a number (NP) of N–dimensional weight vectors is specified. Weight vectors are initialized using a random number generator. At each iteration of the algorithm, called *generation*, new weight vectors are generated through the combination of randomly selected members of the existing population. This is the *mutation* operation. The resulting weight vectors are then mixed with a predetermined weight vector, called the *target* weight vector. This stage of the algorithm is called *crossover*. The outcome of the mutation and crossover operation yields the *trial* weight vector. The trial weight vector is accepted if and only if it reduces the value of the error function E. The final operation is known as

selection. Subsequently, a brief outline of the workings of the two DE main operations is presented.

The first DE operator is the *mutation operator.* Specifically, for each weight vector, a new vector v^i_{g+1}, called mutant vector, is generated according to the relation:

$$v^i_{g+1} = w^i_g + \mu(w^{best}_g - w^i_g) + \mu(w^{r1}_g - w^{r2}_g).$$

where w^i_g, $i = 1, ..., NP$ represents the i–th weight vector of the population, g stands for the current generation index, and w^{best}_g denotes the best member of the previous generation. The constant parameter, $\mu > 0$, is a real number, called mutation constant, which controls the amplification of the difference between the two weight vectors. Finally, w^{r1}_g and w^{r2}_g are two randomly selected weight vectors of the current generation, different from w^i_g. To stimulate further the diversity among members of the new population, the *crossover operator* is applied. For each component of $j = 1, ..., n$ of the mutant weight vector a randomly selected real number $r \in [0, 1]$. If $r \leq \rho$, where $\rho > 0$ is the crossover constant, then the j–th component of the trial vector is replaced by the j–th component of the mutant vector. Otherwise the j–th component of the target vector is selected.

3. Empirical Results

The data set used in the present study is provided by the official website of the European Central Bank (ECB), and consists of the daily exchange rate of the Euro against the USD, starting from the introduction of the Euro on January 1st 1999 and extending to October 10th 2001. Observations after October 11th 2001 were excluded, due to the international turmoil which had a major impact on the international foreign exchange rate markets. Clearly, no method can provide reliable forecasts once exogenous factors never previously encountered, exert a major impact on the formulation of market prices. The total number of observations included in present the study was therefore limited to 619.

The time series of daily exchange rates, illustrated in Figure 2, is clearly nonstationary. An approach frequently encountered in the literature, to overcome the problem of nonstationarity is to consider the first differences of the series, or the first differences of the natural logarithms. Both of these approaches transform the original, nonstationary, time series, to a stationary one. If, however, the original series is contaminated with noise, there is a danger that both of these transformations will accentuate the

presence of noise, in other words increase the noise–to–signal ratio, while at the same time eliminate valuable information. Our experience indicates that for the task of one–step–ahead prediction both of these transformations inhibit the training process and effectively impose the use of larger network architectures, while at the same time, there is no indication of superior forecasting ability. Thus, the original time series of daily exchange rates was considered, and for the purpose of training ANNs with nonlinear transfer functions the data was normalized within the range $[-1, 1]$. The data set was divided into a test set, containing the last 30 observations, and a training set containing all previous data points. Input patterns consisted of a number of time lagged values, $x_k = (x_t, ..., x_{t-n})$, whereas the desired response for the network was set to the next day's exchange rate, $d_k = x_{t+1}$.

Numerical experiments were performed using the Neural Network Toolbox version 4.0 for Matlab 6.0, as well as a Neural Network C++ Interface built under the Linux Operating system using the g++ compiler. Using the Neural Network Toolbox version 4.0 provided the opportunity to apply several well–known deterministic training algorithms. More specifically, the following algorithms were considered:

* Standard Back Propagation (BP),[13]
* Back Propagation with Adaptive Stepsize (BPAS),
* Resilient Back Propagation (RPROP),[12]
* Conjugate Gradient Algorithms (CG), and
* Levenberg–Marquardt (LM).[5]

The first three training algorithms, BP, BPAS, and RPROP, exploit gradi-

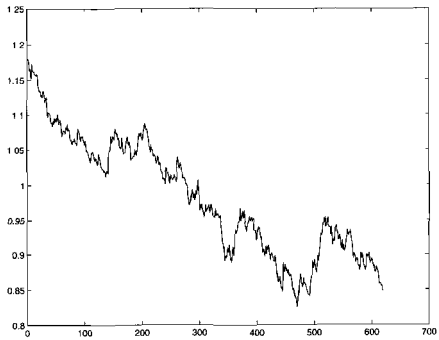

Fig. 2. Time Series of the Daily Euro/USD Exchange Rate

ent information to determine the update of synaptic weights, whereas, CG algorithms and the LM algorithm also incorporate an approximation of the Hessian matrix, thereby exploiting second order information. The DTLFNs trained through the DE algorithm were implemented using a Neural Network C++ Interface built under the Linux Operating system. This interface was selected since it greatly reduced the time required to train ANNs.

All training methods considered were extensively tested with a wide range of parameters. The BP and BPAS algorithms frequently encountered grave difficulties in training. Relative to speed measures RPROP proved to be the fastest. A crucial difference was relative to the reliability of performance were the LM and DE algorithms proved to be the most reliable, in the sense that ANNs with the same topology and activation functions, trained through these algorithms tended to exhibit small variability in performance on the test set. This finding however was contingent on the size of the network; as the number of hidden neurons and layers increased this highly desirable property from the viewpoint of financial forecasting, tended to vanish.

Network topology has been recognized as a critical determinant of network performance. Numerical experiments performed in the context of the present study conform to this finding. Unfortunately, the problem of identifying the "optimal" network topology for a particular task is very hard and currently remains an open research problem. To find a suitable network we proceed according to the following heuristic method. Starting with a network with a single hidden layer and a minimum number of hidden neurons, usually two, we proceed to add neurons and layers as long as performance on both the test and training set is improving. For the purposes of the present study ANNs with a single hidden layer proved to be sufficient. It is worth noting that adding more layers tends to inhibit the generalization ability of the network.

To evaluate the performance of different predictors, several measures have been proposed in the literature.[1,3,8,9,19] The primary focus of the present study was to create a system capable of capturing the direction of change of daily exchange rates, i.e. whether tomorrow's rate will be lower or higher relative to today's rate. To this end a measure called *sign prediction* was applied. Sign prediction measures the percentage of times for which the following inequality holds on the test set:

$$(\widehat{x_{t+1}} - x_t) * (x_{t+1} - x_t) > 0$$

where, $\widehat{x_{t+1}}$ represents the prediction generated by the ANN, x_{t+1} refers to the true value of the exchange rate at period $t+1$ and, finally, x_t stands for the value of the exchange rate at the present period, t. If the above inequality holds, then the ANN has correctly predicted the direction of change of the exchange rate.

As previously mentioned the EMH states that the best possible forecast of tomorrow's rate is today's rate, $\widehat{x_{t+1}} = x_t$. We refer to this forecast as the *naive predictor*, since it requires knowledge of only the last value of the time series. Despite the fact that EMH has been theoretically questioned and different researchers have been capable of outperforming the naive predictor, comparing the accuracy of forecasts with that of the naive predictor remains a benchmark for comparison. To evaluate the performance of different ANNs with respect to the naive predictor a measure called *acrnn* is devised. The acrnn measure captures the percentage of times for which the absolute deviation between the true value and the value predicted by the ANN is smaller than the absolute deviation between the true value and the value predicted by the naive predictor. In other words, acrnn measures the percentage of times for which the following inequality holds on the test set.

$$|x_{t+1} - \widehat{x_{t+1}}| < |x_{t+1} - x_{t+1}^{naive}|$$

where $x_{t+1}^{naive} = x_t$.

Initially, different ANNs were trained on the training set and consequently, their ability to produce accurate predictions for the entire test set was evaluated. Overall, the obtained results were unsatisfactory. Irrespective of the ANN type, topology and, training function applied, the accuracy of the predictions generated by the trained ANNs for the observations that belong to the test set, was not significantly better than the naive predictor. In other words, an acrnn measure consistently above 50% was not achieved by any network. Moreover, with respect to sign prediction, no ANN was capable of consistently predicting the direction of change with accuracy exceeding 55%. Plotting predictions generated by ANNs against the true evolution of the time series it became evident that the predictions produced were very similar to a time–lagged version of the true time series. Alternatively stated, the ANN approximated closely the behavior of the naive predictor. Due to the fact that daily variations were indeed very small compared to the value of the exchange rate this behavior resulted to very small mean squared error on the training set. Performance did not improve as the number of time lags provided as inputs to the network, n, increased. Indeed,

the best results were obtained for values of n higher than 2, and lower than 10. Despite the fact that the task of training different ANNs became easier and performance on the training set was improving, as n was increasing, performance on the test set showed no signs of improvement. This behavior indicated that instead of capturing the underlying dynamics of the system, increasing the free parameters of the network *per se* rendered ANNs prone to overfitting. Similar results were obtained as the number of hidden neurons was increased above 5. To avoid overfitting early stopping was applied. Early stopping implies the division of the data set into a training, a validation and a test set. At each training epoch, the error on the validation set is computed but no information from this measurement is included in the update of synaptic weights. Training is terminated once the error on the validation set increases as training proceeds beyond a critical point. Incorporating early stopping did not produce a significant improvement of performance. This outcome can be justified by the presence of nonstationarity which implies that the selection of an appropriate validation set is not trivial. In effect the patterns selected to comprise the validation set need to bear a structural similarity with those that comprise the test set, a prerequisite that cannot be guaranteed to hold if a set of patterns just before the test set is selected as a validation set for the particular task.

A possible explanation for the evident inability of different ANNs to produce accurate one–step–ahead forecasts of the daily exchange rate of the Euro against the USD is that the market environment in the particular foreign exchange market is rapidly changing. This is a reasonable assumption taking into consideration the intense competition among market participants. If this claim is valid, then the knowledge stored in the synaptic weights of trained ANNs becomes obsolete for the task of prediction, as patterns further in the test set are considered. To evaluate the validity of this claim an alternative approach is considered. More specifically, ANNs are trained on the training set, predictions for a test set consisting of the five patterns that immediately follow the end of the training set are generated and, network performance is evaluated. Subsequently, the first pattern that belongs to the test set is assigned to the training set. A test set consisting of five patterns immediately following the end of the new training set is selected, and the process is repeated until forecasts for the entire test set consisting of 30 observations are generated. To promote the learning process and avoid the phenomenon of generated predictions being a time–lagged version of the true time series a modified error performance function was implemented for DTLFNs trained through the DE algorithm.

This function assigns an error value of zero for the kth pattern as long as the DTLFN accurately predicts the direction of change for the particular pattern, otherwise error performance is computed as in the standard mean squared error performance function.

$$E_k = \begin{cases} 0 & \text{if } (\widehat{x_{t+1}} - x_t) * (x_{t+1} - x_t) > 0 \\ \frac{1}{2}\sum(x_{t+1} - \widehat{x_{t+1}})^2 & \text{otherwise} \end{cases}$$

To generate predictions for the entire test set, the training and evaluation processes were repeated 25 times with the maximum number of training epochs set to 150. The performance a FTLFN and a DTLFN trained though the LM algorithm using a mean squared error performance function, as well as that of a DTLFN trained through the DE algorithm using the modified error performance function, is reported in Table 1. The generalization ability of the first two networks is clearly unsatisfactory. The accuracy of predictions is inferior to that of the naive predictor and average sign prediction is considerably lower than 50%. This is not however the case for the last DTLFN. The predictions generated by the DTLFN that was trained using the DE algorithm and the modified error performance function, clearly outperform the naive predictor, with an average acrnn value of 59.2%. Most importantly average sign prediction assumes a value of 68%, which is substantially above 50% and substantially higher than the sign prediction achieved in Ref. 3. A significant advantage due to the incorporation of the modified error performance function and the DE training algorithm, which is not evident from Table 1, is the fact that network performance on the training set became a much more reliable measure of generalization. In other words, a reduction of the error on the training was most frequently associated with superior performance on the test set. At the same time, the phenomenon of deteriorating performance on the test as training proceeded was very rarely witnessed.

4. Concluding Remarks

The ability of Distributed Time–Lagged Neural Networks (DTLFNs), trained using a Differential Evolution (DE) algorithm and a modified error performance function, to accurately forecast the direction of change of the daily exchange rate of the Euro against the US Dollar has been investigated. To this end only the history of previous values has been exploited. Attention was focused on sign prediction, since it is sufficient to produce a

speculative profit for market participants, but results were also contrasted with the naive predictor. Comparing the out–of–sample performance of the

Table 1. Test Set Performance

Topology: $5 * 5 * 1$						
Iteration	FTLFN		DTLFN LM		DTLFN DE	
	sign	acrnn	sign	acrnn	sign	acrnn
1	100%	40%	40%	0%	80%	60%
2	40%	0%	40%	0%	80%	80%
3	60%	0%	40%	0%	60%	60%
4	20%	20%	60%	0%	80%	80%
5	80%	20%	60%	0%	80%	80%
6	20%	0%	40%	0%	40%	40%
7	60%	20%	40%	0%	80%	80%
8	40%	20%	40%	0%	60%	20%
9	20%	0%	40%	0%	80%	60%
10	60%	20%	80%	0%	60%	60%
11	40%	40%	20%	0%	80%	80%
12	20%	0%	20%	0%	80%	80%
13	80%	20%	60%	20%	80%	80%
14	20%	0%	60%	20%	80%	80%
15	20%	20%	60%	0%	60%	60%
16	80%	20%	40%	0%	80%	80%
17	0%	0%	60%	0%	80%	80%
18	60%	20%	40%	0%	80%	80%
19	80%	40%	100%	20%	100%	100%
20	40%	20%	40%	20%	60%	40%
21	40%	20%	20%	0%	60%	40%
22	40%	0%	60%	0%	40%	20%
23	40%	0%	20%	0%	40%	40%
24	60%	20%	40%	0%	40%	0%
25	20%	0%	20%	0%	40%	0%

DTLFNs trained through the proposed approach with that of DTLFNs and FTLFNs trained through different deterministic algorithms and using the mean squared error performance function, the proposed approach proved to be superior for the particular task. The results from the numerical experiments performed were promising, with correct sign prediction reaching an average of 68% and the average percentage of times for which the predictions were more accurate than the naive predictor being 59.2%. A further advantage of this approach was the fact that network performance on the training set became a more reliable indicator of generalization ability.

Further work will include the application of the present approach to different financial time series as well as the consideration of alternative training methods based on evolutionary and swarm intelligence methods.

References

1. A.S. Andreou, E.F. Georgopoulos and S.D. Likothanassis. Exchange–Rates Forecasting: A Hybrid Algorithm Based on Genetically Optimized Adaptive Neural Networks. *Computational Economics*, in press.
2. G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematical Control Signals Systems*, **2**:303–314 (1989).
3. C.L. Giles, S. Lawrence and A.C. Tsoi. Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. *Machine Learning*, **44**(1/2):161–183 (2001).
4. S.J. Grossman and J. Stiglitz. On the Impossibility of Informationally Efficient Markets. *American Economic Review*, **70**:393–408 (1980).
5. M.T. Hagan and M. Menhaj. Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993 (1994).
6. M.H. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press (1995).
7. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, New York (1999).
8. C.M. Kuan and T. Liu. Forecasting Exchange Rates Using Feedforward and Recurrent Neural Networks. *Journal of Applied Econometrics*, 10:347–364 (1995).
9. M.T. Leung, A.S. Chen and H. Daouk. Forecasting Exchange Rates using General Regression Neural Networks. *Computers & Operations Research*, 27:1093-1110 (2000).
10. V.P. Plagianakos and M.N. Vrahatis. Training Neural Networks with Threshold Activation Functions and Constrained Integer Weights. *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)* (2000).
11. A. Refenes. *Neural Networks in the Capital Markets*. John Wiley and Sons (1995).

12. M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, pp. 586–591 (1993).
13. D.E. Rumelhart, G.E. Hinton and R.J. Williams. Learning Internal Representations by Error Propagation. In D.E. Rumelhart and J.L. McClelland (Eds.) *Parallel distributed processing: Explorations in the microstructure of cognition.* Cambridge, MA, MIT Press, 1:318–362 (1986).
14. I.W. Sandberg and L. Xu. Uniform Approximation of Multidimensional Myopic Maps. *IEEE Transactions on Circuits and Systems*, 44:477–485 (1997).
15. I.W. Sandberg and L. Xu. Uniform Approximation and Gamma Networks. *Neural Networks*, 10:781–784 (1997).
16. R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Contituous Spaces. *Journal of Global Optimization*, 11:341–359 (1997).
17. W. Verkooijen. A Neural Network Approach to Long–Run Exchange Rate Prediction. *Computational Economics*, 9:51–65 (1996).
18. E. Wan. Time Series Prediction using a Connectionist Network with Internal Delay Lines. in *Time Series Prediction: Forecasting the Future and Understanding the Past*, A.S. Weigend and N.A. Gershenfeld. Reading, MA: Addison–Wesley, pp. 195–217 (1993).
19. B. Wu. Model–free Forecasting for Nonlinear Time Series (with Application to Exchange Rates). *Computational Statistics & Data Analysis*, 19:433–459 (1995).