Comparing Unsupervised Layers in Neural Networks for Financial Time Series Prediction

Asmaa Mahdi, Tillman Weyde
School of Mathematics, Computer Science and Engineering
City, University of London, UK
Email: {asmaa.mahdi, t.e.weyde}@city.ac.uk

Dhiya Al-Jumeily
School of Computer Science
Liverpool John Moores University, UK
d.aljumeily@ljmu.ac.uk

Abstract—**In this study, we propose and compare neural network models that use unsupervised layers for the prediction of financial time series. We compare the novel FL-RBM and FL-SMIA-RMB models that integrate a Restricted Boltzmann Machine (RBM) and the self-organizing layer of the Self-organized Multi-Layer Network using the Immune Algorithm (SMIA) with the FL-SMIA network and a standard MLP. We aim to investigate the performance of unsupervised learning in comparison to purely supervised and other mixed models. The FL-RBM model combines the products of raw input features (the Functional Link, FL), with the Restricted Boltzmann Machine RBM as a self-organizing first hidden layer, while the FL-SMIA model uses the Immune Algorithm on the first layer. The FL-SMIA-RBM model, combines both self-organizing layers with a back-propagation network.**

The results show that the FL-SMIA model outperforms the FL-RBM, the FL-SMIA-RBM and the MLP as measured by Annualized Return (AR) in one-day-ahead prediction on exchange rates time series. In terms of volatility, the FL-SMIA and MLP perform similarly.

I. Introduction

Financial markets are an important aspect in modern economies, which attracts much interest from both commercial and academic communities. Therefore, a wide range of investigations is ongoing in the financial domain. Predicting financial time series has an effect on the trading decisions of many companies and organizations, which are searching for new technologies that will support them in becoming more profitable and competitive [1], [2], [3]. Financial time series are highly non-linear and complex [4], as many risk factors, affect prices and exchange rates [5].

Artificial neural networks have gained significant interest in financial research [6], [7] but they suffer from some problems, such as over-fitting [8], [9]. Therefore, we propose novel neural network models for financial prediction: the FL-RBM model and the FL-SMIA-RBM model.

To reduce over-fitting, many methods and approaches have been introduced, such as the *Functional Link Neural Network* (FLNN) [10], [11]. The FLNN model uses inputs and their products and removes the hidden layer from the standard Multilayer Perceptron (MLP) architecture, in order to enable the network to perform non-linearly separable classification tasks and to reduce the model complexity compared to an MLP. Due to an exponential increase in the number of inputs units the FLNN architecture can suffer from the combinatorial explosion. Therefore, only products of two or three inputs are recommended for neural networks [12],[13].

The Self-organized Multilayer neural network using the Immune Algorithm (SMIA) [14], is another approach which has successfully applied to improve the MLP models performance which is based on the immune learning algorithm inspired by biological immune systems.

In [15], the integration of the FLNN model and SMIA in the *Functional Link Self-organized Multilayer neural network using the Immune Algorithm* (FL-SMIA) has been proposed as a novel method for financial time series prediction and to improve the prediction ability for the multilayer networks.

In this paper, firstly we propose a model that combines higher-order (inputs and their products) with the Restricted Boltzmann Machine (FL-RBM). Secondly, we proposethe FL-SMIA-RBM model which combines the FL-SMIA model and the RBM network as a novel method for financial time series prediction.

The rest of this research paper is organized as follows: Section 2 discusses related work in the literature. The Functional Link and the Restricted Boltzmann Machine model (FL-RBM) are discussed in Section 3. In section 4, the combination of the FL-SMIA model with the RBM network (FL-SMIA-RBM) is detailed. The experimental design is presented in Section 5. The results are presented and discussed in Section 6. Conclusions and future work are provided in Section 7.

II. Related Work

In this section, a literature review with a focus on the Immune Algorithm and the Functional Link Self-organizing Multilayer Network using the Immune Algorithm (FL-SMIA) will be presented.

A. The Immune Algorithm

The concept of the immune algorithm was initially discussed in [16], [17]. The self-organization inspired by the immune system appeared in [18], where the researchers used one layer networks combined with contiguity constrained method for clustering analysis.

The immune algorithm is based on the principle of the natural immune system. It is based on the relationship between its components that consist of antigens and cell. The basic concept of the Immune Algorithm is a set of B cells, which each respond to a set of antigens, thus clustering patterns in the training data [19]. When a cell is matched with an antigen then this antigen stimulates the cell to duplicate itself and a mutated cell to produce a diverse set of antibodies in order to remove and fight the intruder attacking the body [19]. Thus, the immune algorithm allows a system to let its components change and learn patterns.

The self-organization inspired by the immune system appeared in [20], where the researchers used one layer networks combined with a contiguity constrained method by [18] for clustering analysis. Later, The the SMIA model has been applied for financial prediction [14] and is extended with product term inputs in [15].

B. The Functional Link Self-organizing Multilayer Network using the Immune Algorithm (FL-SMIA)

The Functional Link-Self-organized Multilayer network using the Immune Algorithm combines aspects of Functional Link Neural Network with the Self-organizing Multilayer network using the Immune Algorithm in the structure and the learning algorithm.

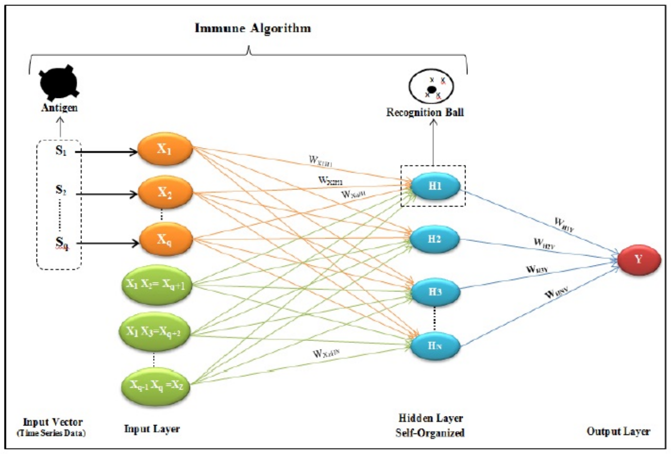

Fig. 1. The FL-SMIA model (Functional Link Self-organized Multilayer using the Immune Algorithm).

The architecture of the FL-SMIA network consists of the input layer, which comprises a number of input units $X_1, X_2, \ldots, X_Z$, the self-organizing hidden layer with units $H_1, H_2, \ldots, H_N$, and the output layer consisting of one output unit as shown in Fig. 1. Here Z, and N refers to the number of units in each layer. This research focuses on adding second order terms to the input units. In our example below the network has five input features $X_1, \ldots, X_5$. Adding the second order term to the inputs results in 10 additional inputs $(X_1X_2, X_1X_3, X_1X_4, \ldots, X_4X_5)$, leading to fifteen input units in total, five with input features and ten with products of inputs features as represented in Fig. 1. The FL-SMIA network uses a hidden layer which operates like in [14] and [21].

The output of the hidden units is determined using the Euclidean distance between the input units (X_i) and the connection weights between the input units and the hidden units (W_{Hij}). The advantage of using the Euclidean distance is to make the network capable of exploiting local information of the input data. The output of a hidden unit H_j is calculated as:

$$H_j = f_{hts}\left(\sqrt{\sum_{i=1}^{Z}(W_{Hij} - X_i)^2 + B_j}\right) \quad (1)$$

where W_{Hij} represents the weight of the connection from the i^{th} input unit to the j^{th} hidden unit, and f_{hts} is the hyperbolic tangent sigmoid function. The number of hidden units is determined from the data by learning with the Immune Algorithm as described in the next section.

The outputs of the hidden units are aggregated in a standard layer with the network output given by:

$$Y = f_{ls}\left(\sum_{j=1}^{N} W_{Hjy}.H_j + B_y\right) \quad (2)$$

where W_{Hjy} represent the strength of the connection weights between the j^{th} hidden unit and the output unit, B_y is the bias of the output unit Y, and f_{ls} is the logistic sigmoid function.

The FL-SMIA as described above has two weight matrices, the first between the input layer and the hidden layer, the second between the hidden layer and the output layer. The second weight matrix is trained using the standard back-propagation algorithm [22] with regularisation to penalise large weights [23] in batch mode. In our case with a single output neuron the weight change is calculated as:

$$\Delta W_{Hjk} = -\eta_b \frac{\partial J}{\partial W_{Hj}} - \lambda W_{Hjk} \quad (3)$$

where W_{Hjk} is the weight of the connection from hidden units Hj to the output unit, $\eta_b \in [0, 1]$ is the learning rate, and J the mean squared error on the training set. The second term on the right-hand side effects the regularisation, which is controlled by the parameter λ. The bias is adapted in the same way but without regularisation.

Before the second weight matrix is trained, the first set of weights and the structure of the hidden layer are trained using the Immune Algorithm [21] as indicated in Fig. 1. In the Immune Algorithm a hidden unit corresponds to a recognition ball (RB) in the immune system. Each hidden unit represents one or more input vectors with the weights of the connections from the input layer to the hidden unit. The hidden unit Hj is represented by (P_j, W_{Hj}), where P_j is the number of input vectors associated with Hj and W_{Hj} is the vector of weights from the input layer to Hj.

We start with one hidden unit $(N = 1)$ and the first hidden unit is created with $P_1 = 1$ and $W_{H1} = X_1$. The Immune Algorithm then performs the following steps to create and update the hidden units until all inputs of the network have found their corresponding hidden unit.

1) For $m = 1, \ldots, M$ perform the following:
   a) For $j = 1, \ldots, N$, calculate the Euclidean distance between the m-th input and the weight vector of the j^{th} hidden unit:

$$dist_{mj} = \sqrt{\sum_{i=1}^{Z}(x_{mi} - w_{Hji})^2} \quad (4)$$

Where x_{mi} is the i^{th} element of input vector x_m, and w_{Hji} is the i^{th} component of vector w_{Hj}, i.e. the weight of the connection from input m to hidden unit j.

b) Determine the closest unit c, i.e. the unit with the shortest distance to x_m:

$$dist_{mc} = min_j(dist_{mj}) \tag{5}$$

c) If the shortest distance $dist_{mc}$ is below the stimulation level Sq (where Sq is selected between 0 and 1), then the input has found its corresponding hidden unit. In this case the weight vector w_{Hc} of the hidden unit closest to x_m will be updated as

$$w_{Hc_{new}} = w_{Hc} + \eta * dist_{mc} \tag{6}$$

where $\eta_i \in (0, 1)$ is the learning rate for the Immune Algorithm, w_{Hc} is the weight vector of the hidden unit closest to x_m. P_c will be incremented by 1.
Otherwise, the shortest distance $dist_{mc}$ is greater than the stimulation level Sq. This means that no matching hidden unit was found for the input and we create a new hidden unit (P_N, W_N) with $P_N = 1$ and $W_{HN} = X_m$. Then we update $N = N + 1$.

2) Repeat from step 1 as long as new hidden units have been created.

III. The Functional Link and Restricted Boltzmann Machine model (FL-RBM)

The Restricted Boltzmann Machine is an unsupervised learning model that has been introduced in [24]. RBMs have been used in several applications including classification[25], feature learning [26], and dimensionality reduction [27]. It has been used as pre-training to render deep learning architectures more effective [28].

The RBM is a network of two layers, the first layer includes a number of inputs or visible units (v) as binary inputs, and the second layer with a number of hidden units (h). The visible neurons are connected to the hidden neurons in a stochastic way and without connections between each of visible neurons in the input layer or hidden neurons in the hidden layer. In other words, in an RBM network there are no direct connections between units in the same layer, so which is why it is called restricted [27]. Every unit has a binary state of $0 or 1$, and the input and hidden units are random variables (v, h). The visible units are combined with biases a_i and the hidden units are combined with biases b_j. The RBM learns to extract features from the data by reconstructing the inputs. The RBM's energy E is given by the following:

$$E(v,h) = -\sum_{i}^{N}\sum_{j}^{L} W_{ij}v_i h_j - \sum_{i}^{N} a_i v_i - \sum_{j}^{L} b_j h_j, \tag{7}$$

where v_i is the binary state of visible unit i, h_j is the binary state of hidden unit j, a_i and b_j are their biases, W_{ij} is the weight between the visible unit i and hidden unit j, N refers to the number of visible units, and L is the number of hidden units. The probability of the state of the network is defined as

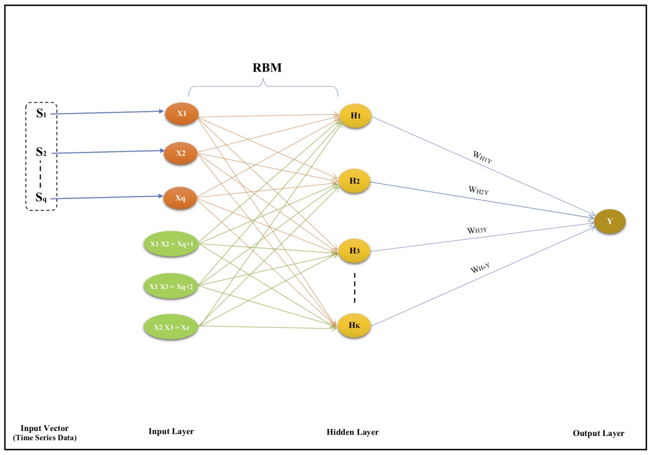

Fig. 2. The proposed network FL-RBM

$$P(v,h) = \frac{1}{Z}e^{-E(v,h)}, \tag{8}$$

where Z is the partition function for normalization. The probability of a hidden unit state given a visible vector is given as:

$$p(h_j = 1|v) = (b_j + \sum_i v_i W_{ij}). \tag{9}$$

The derivative of the log probability of a training vector v with respect to a weight is:

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{10}$$

The $\langle \cdot \rangle$ indicates the expected value under the distribution in the index. For the model case we estimate the expectation with the a single reconstruction cycle and get a simple update rule [29]:

$$\Delta w_{ij} = \lambda\left((v_i h_j)_{data} - (v_i h_j)_{model}\right) \tag{11}$$

Where λ is a learning rate. The update for the biases is analogous.

The outputs of the hidden units of the RBM are aggregated as a standard hidden layer H_j with the network output given by:

$$Y = f_{sig}\left(\sum_{j=1}^{N} W_{jy}.H_j + B_y\right) \tag{12}$$

where W_{jy} represent the connection weights between the j^{th} hidden unit and the output unit, B_y is the bias of the output unit Y, and f_{sig} is the sigmoid activation function.

IV. The FL-SMIA with Restricted Boltzmann Machine (FL-SMIA-RBM)

In this section, we propose a novel model based on learning using a Functional Link-Self-organized network of the Immune Algorithm (FL-SMIA) and the Restricted Boltzmann Machines (FL-SMIA-RBM).

In general, the model proposed (FL-SMIA-RBM) is a multilayer neural network using unsupervised and supervised learning for improving the time series forecasting. The structure of the network (FL-SMIA-RBM) includes the following layers:

1. The input layer consists of the 15 input units (5 inputs and their products as a higher order).
2. The first hidden layer (SMIA) is the self-organized layer using the immune learning algorithm (unsupervised learning). The number of hidden units in this layer is a combination of the units that generated by the SMIA algorithm and extra 5 hidden units, these hidden units are a various number depending on the data-set trained. SMIA hidden units are represented as visible units to the next hidden layer.
3. The second hidden layer (RBM) is the Restricted Boltzmann Machines (unsupervised learning). the number of hidden units is 10.
4. The output layer with only one unit output is trained with the Back Propagation Algorithm (supervised learning).

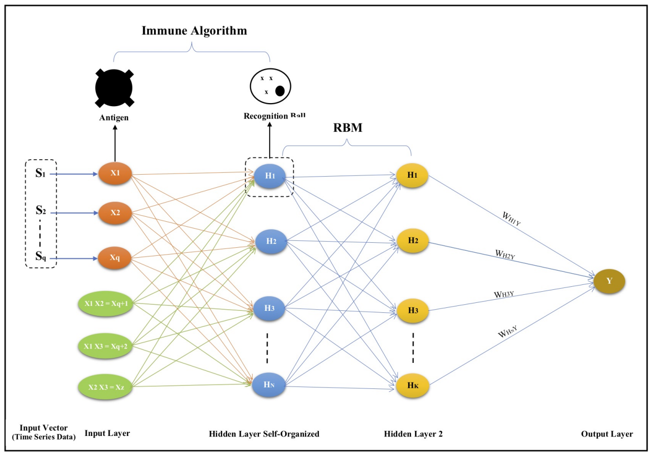

Fig. 3. The proposed network FL-SMIA-RBM

V. Experimental Design

We used two financial time series (exchange rate prices) to evaluate the FL-SMIA-RBM and the FL-RBM model. We use time series data available from the Federal Reserve, Board of Governors. [1]. This is daily time series covering the period from 1/7/2002 to 11/11/2008, giving 1605 trading days. The first time series is the US dollar to UK pound exchange rate (US/UK), the second time series is the US dollar to EURO exchange rate (US/EU).

TABLE I
Financial time series datasets.

No	Time series dataset	Acronym
1	US dollar to UK pound exchange rate	US/UK
2	US dollar to EURO exchange rate	US/EU

A. Pre-Processing of Data

The relative difference in the percentage of the price (RDP) has been used in this work in order to reduce the non-stationarity of the financial data-sets as they are known to be highly noisy [30]. This transformation makes the distribution of the data more symmetrical and closer to a normal distribution. To the input and output data were transformed as in [9], [14]. The inputs are EMA15, the difference between a 15-day exponential moving average and original signal, and four RDP values based on five-day periods (RDP-5, RDP-10, RDP-15, and RDP-20).

In this research our forecast horizon is 1 day. Therefore the output variable presented as a relative difference of price in percent for the next day (RDP+1) . As explained in [9], the RDP+1 and RDP+5 values are calculated as relative difference in percent of 3-day exponential moving average for one day ahead. The calculations for all the indicators are given in Table II.

TABLE II
INPUT AND OUTPUT VARIABLES according to [9].

Indicator	Calculation
Input variables	
EMA15	$P(i) - EMA_{15}(i)$
RDP-5	$(P(i) - P(i-5))/P(i-5) * 100$
RDP-10	$(P(i) - P(i-10))/P(i-10) * 100$
RDP-15	$(P(i) - P(i-15))/P(i-15) * 100$
RDP-20	$(P(i) - P(i-20))/P(i-20) * 100$
Output variable	
RDP+h	$(P(i+h) - P(i))/P(i) * 100$ where $P(i) = EMA_3(i)$

The EMA$_n(i)$ is the n-day exponential moving average of the i^{th} day, $p(i)$ is the signal value (refer to (y_i)) of the i^{th} day, and h is the prediction horizon of 1 day or 5 days.

For the purpose of reducing the range difference in the data and to avoid the computational problems, the time series data which have been used in this research are scaled. All input and output variables were scaled in order to produce a new data range which is more suitable to the network transfer function.

The RDP series have been scaled in this work by using the standard minimum and maximum normalization method, as follows:

$$N_x = (Max_2 - Min_2) * \left(\frac{x - Min_1}{Max_1 - Min_1} \right) + Min_2 \quad (13)$$

Where N_x is the normalized value, Min_1 and Max_1 refer to the minimum and maximum values of the original series, Min_2 and Max_2 are the desired minimum and maximum of the new scaled series and x is the original value of the time series.

The prediction results of the FL-RBM model and the FL-SMIA-RBM model has been compared to the performance of threee neural networks models which are the Multilayer Perceptron (MLP), Regularised Multilayer Perceptron (R-MLP), and Functional Link-Self-organized network using the Immune Algorithm (FL-SMIA) model.

In this research, the data have been applied for training, validation and test sets comprising 50%, 25%, and 25%

[1]http://economagic.com/ecb.htm/fedstl.htm

respectively of the data. Early stopping has been used for all networks.

The combinations of the parameters (momentum, learning rate, and decay rate) have been explored in a grid search for all networks learning that used in this research. The best values found are in the following ranges: for the momentum 0.3 to 0.5, for the learning rate values are 0.04 to 0.4, and for the decay rate parameter which has been used in the R-MLP, R-SMIA, FL-SMIA, FL-RBM and FL-SMIA-RBM networks optimal values are 0.0001 to 0.001.

B. Financial Evaluation Metrics

To evaluate the performance of the networks that have been used in this research, we focus on Annualised Return (AR) and Annualized Volatility (AV) measuring the profitability and investment risk of the strategy over a year. [14].

1) Annualized Return (AR): The Annualised Return (AR) measure estimates the effectiveness of a model for automatic trading. It measures the total profitability in a year of a strategy using *buy* and *sell* signals generated by the models [31].

The Annualised Return (AR) is represented relative to the maximal return by buying and selling using perfect prediction, and is calculated as follows [31]:

a) Calculate the returns (R_i):

$$R_i = \begin{cases} +|y_i| & \text{if} y_i y_i^* \geq 0 \\ -|y_i| & \text{otherwise,} \end{cases} \tag{14}$$

where y_i is the target output value and y_i^* represent the predicted output value.

b) Find the sum of profits:

$$CR = \sum_{i=1}^{n} (R_i), \tag{15}$$

where n is the total number of data samples.

c) Calculate the profit and all profit, i.e. the maximal possible profit:

$$Profit = \left(\frac{252}{n}\right) * CR, \tag{16}$$

$$AllProfit = \left(\frac{252}{n}\right) * \sum_{i=1}^{n} abs(R_i), \tag{17}$$

where n is the total number of the data sample, and 252 is taken as the number of trading days per year.

d) Calculate the Annualized Return (AR), which is expressed as percentage of the actual profit relative to the maximal profit per year:

$$AR = \left(\frac{Profit}{AllProfit}\right) * 100 \tag{18}$$

2) Annualized Volatility (AV): In order to evaluate the investment risk and profits possibility, the Annualized Volatility (AV) has been used in this work to provide information related to the variability of the prices. The volatility measure is used to measure the variance of returns over a period of time. The small value of volatility is preferable for financial predictions. Furthermore, in real trading, the volatility measure provides significant information for investment risk which makes it useful for financial analysis.

To calculate the variance of returns, firstly should the daily returns R_i be calculated and use it to calculate the average of the returns R^*. Secondly, calculate the variance of the returns which equal to the average of square of the difference between the returns and the average of the returns. Then the standard deviations (the square root of the variance of the returns) must be calculated to produce the daily volatility.

$$V_d = \sqrt{\left(\frac{1}{n-1}\right) * \sum_{i=1}^{n} (R_i - R^*)^2} \tag{19}$$

where V_d is the daily volatility, n represents the total number of the data sample, R_i illustrate the returns for each time period, and R^* is the average of the returns.

After the daily volatility is calculated, then it will be easy to get the annualized volatility over the year by calculated it as follows:-

$$AV = \sqrt{252} * \sqrt{\left(\frac{1}{n-1}\right) * \sum_{i=1}^{n} (R_i - R^*)^2} \tag{20}$$

Where 252 is the number of trading days in a year. We have also used a variety of statistical metrics to further evaluate the performance of the models [31], which we are not reporting here.

VI. Results and Discussion

The networks have been tested on all data-sets from Table I using the metrics described above.

TABLE III
THE BEST US/UK RESULTS FOR ONE DAY AHEAD PREDICTION IN PERCENT.

Network	Hidden units	AR	AV
MLP	8	71.4873	4.3927
R-MLP	8	72.1075	**4.3763**
FL-SMIA	40	**76.2248**	4.6260
FL-RBM	10	55.8498	4.7507
FL-SMIA-RBM	40	59.4492	4.6794

As shown in Table III and Table IV. The AR results for one day ahead prediction proved that the FL-SMIA network outperformed all the networks. The FL-SMIA network predicts higher annualized return than the all other networks for both of the data-sets. The AR results for the FL-RBM and FL-SMIA-RBM models as worse than the other models.

TABLE IV
THE BEST US/EU RESULTS FOR ONE DAY AHEAD PREDICTION IN PERCENT.

Network	Hidden units	AR	AV
MLP	8	71.5395	4.6284
R-MLP	8	73.2384	4.5828
FL-SMIA	40	**76.6475**	**4.4865**
FL-RBM	10	61.3265	4.8785
FL-SMIA-RBM	40	60.6847	4.6537

Regarding the investment risk, the Annualized Volatility (AV) results proved that the FL-SMIA model reduced the investment risk by produced the lowest AV value than all other models when using the US/EU data, While for the US/UK data, the R-MLP model has the lowest AV value than all other of all networks used in this research.

VII. Conclusion

In this research, we proposed the FL-RBM model and the FL-SMIA-RBM model. For FL-RBM model, we combine inputs and their product with and RBM network. For the FL-SMIA-RBM model, we combine the inputs and their product with a self-organizing hidden layer and the RBM network, using the Immune Algorithm as in SMIA networks. All networks have been evaluated using two financial data-sets (exchange rates) for prediction one day ahead. The FL-SMIA outperformed all other networks for AR results. While in terms of volatility, the FL-SMIA and MLP perform similarly.

Overall the use of the unsupervised Immune learning Algorithm improved performance on financial time series prediction compared to multilayer networks and the RBM networks.

Acknowledgment

Asmaa Mahdi would like to thank the Republic of Iraq, Ministry of Higher Education and Scientific Research for supporting her Ph.D study.

References

[1] D. K. Bebarta, A. K. Rout, B. Biswal, and P. Dash, "Forecasting and classification of indian stocks using different polynomial functional link artificial neural networks," in *2012 Annual IEEE India Conference (INDICON)*. IEEE, 2012, pp. 178–182.

[2] K. Li, "Forecasting of government's financial educational fund by using neural networks model," in *Genetic and Evolutionary Computing, 2008. WGEC'08. Second International Conference on*. IEEE, 2008, pp. 120–123.

[3] K. Zou and R. Dong, "A novel approach for time series analysis based rbf neural network," in *Information Technology and Applications (IFITA), 2010 International Forum on*, vol. 3. IEEE, 2010, pp. 139–142.

[4] Y. Li and F. Ying, "Multivariate time series analysis in corporate decision-making application," in *Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on*, vol. 2. IEEE, 2011, pp. 374–376.

[5] C. Chatfield, "The analysis of time series: theory and practice," *Chapman Hall*, 1975.

[6] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 389–10 397, 2011.

[7] G. Zhang and M. Y. Hu, "Neural network forecasting of the british pound/us dollar exchange rate," *Omega*, vol. 26, no. 4, pp. 495–506, 1998.

[8] S. Lawrence and C. L. Giles, "Overfitting and neural networks: conjugate gradient and backpropagation," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 1. IEEE, 2000, pp. 114–119.

[9] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, 2001.

[10] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order neural networks," *Applied optics*, vol. 26, no. 23, pp. 4972–4978, 1987.

[11] Y. Pao, "Adaptive pattern recognition and neural networks," 1989.

[12] T. Kaita, S. Tomita, and J. Yamanaka, "On a higher-order neural network for distortion invariant pattern recognition," *Pattern Recognition Letters*, vol. 23, no. 8, pp. 977–984, 2002.

[13] G. Thimm, "Optimization of high order perceptrons," 1997.

[14] A. Mahdi, A. J. Hussain, P. Lisbo, and D. Al-Jumeily, "The application of the neural network model inspired by the immune system in financial time series prediction," in *Developments in eSystems Engineering (DESE), 2009 Second International Conference on*. IEEE, 2009, pp. 370–376.

[15] A. Mahdi, T. Weyde, and D. Al-Jumeily, "The fl-smia network: A novel architecture for time series prediction," in *2017 10th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2017, pp. 31–36.

[16] L. N. De Castro and J. Timmis, *Artificial immune systems: a new computational intelligence approach*. Springer Science & Business Media, 2002.

[17] J. Timmis, "Artificial immune systems: a novel data analysis technique inspired by the immune network theory," Ph.D. dissertation, Department of Computer Science, 2000.

[18] M. Y. Kiang, "Extending the kohonen self-organizing map networks for clustering analysis," *Computational Statistics & Data Analysis*, vol. 38, no. 2, pp. 161–180, 2001.

[19] X. Shen, X. Z. Gao, and R. Bie, "Artificial immune networks: Models and applications," *International Journal of Computational Intelligence Systems*, vol. 1, no. 2, pp. 168–176, 2008.

[20] R. Widyanto, Y. T. Megawati, and K. Hirota, "Clustering analysis using a self-organized network inspired by immune algorithm," in *Proceeding of the IASTED International Conference on Artificial and Computational Intelligence, ACTA Press, Tokyo*, 2002, pp. 197–202.

[21] M. R. Widyanto, H. Nobuhara, K. Kawamoto, K. Hirota, and B. Kusumoputro, "Improving recognition and generalization capability of back-propagation nn using a self-organized network inspired by immune algorithm (sonia)," *Applied Soft Computing*, vol. 6, no. 1, pp. 72–84, 2005.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1986, vol. 1.

[23] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[24] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," Colorado Univ at Boulder Dept of Computer Science, Tech. Rep., 1986.

[25] H. Larochelle and Y. Bengio, "Classification using discriminative restricted boltzmann machines," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 536–543.

[26] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[27] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[28] N. Le Roux and Y. Bengio, "Representational power of restricted boltzmann machines and deep belief networks," *Neural computation*, vol. 20, no. 6, pp. 1631–1649, 2008.

[29] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 599–619.

[30] R. Ghazali, "Higher order neural networks for financial time series prediction," Ph.D. dissertation, Liverpool John Moores University, 2007.

[31] C. Dunis and M. Williams, "Modelling and trading the eur/usd exchange rate: Do neural network models perform better?" *Derivatives use, trading and regulation*, vol. 8, no. 3, pp. 211–239, 2002.