International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)
ISSN 2150-7988 Vol.1 (2009), pp. 110–117
http://www.mirlabs.org/ijcisim

Forecasting the UK/EU and JP/UK trading signals using Polynomial Neural Networks

R. Ghazali ‡*, N. Mohd Nawi *, M. Z. Mohd. Salikon *
* *Faculty of Information Technology and Multimedia, Universiti Tun Hussein Onn Malaysia.*
‡ *Corresponding author. E-mail: rozaida@uthm.edu.my*

Abstract

This research investigates the use of Ridge Polynomial Neural Network (RPNN) as non-linear prediction model to forecast the future trends of financial time series. The network was used for the prediction of one step ahead and five steps ahead of two exchange rate signals; the British Pound to Euro and the Japanese Yen to British Pound. In order to deal with a dynamic behavior which exists in time series signals, the functionality and architecture of the ordinary feedforward RPNN were extended to a novel recurrent neural network architecture called Dynamic Ridge Polynomial Neural Network (DRPNN). Simulation results indicate that the proposed DRPNN offers significant advantages over feedforward RPNN and Multilayer Perceptron including such increment in profit return, reduction in network complexity, faster learning, and smaller prediction error.

1. Introduction

Most research on time-series prediction has traditionally concentrated on linear methods, which are computationally inexpensive and mathematically convenient. Unfortunately, most systems that are of interest are non-linear. An important class of non-linear systems appears in financial forecasting, which typically include the prediction of exchange rates and share prices. Forecasting an exchange rate is undoubtedly very challenging and important task in the international monetary markets. The foreign exchange market is the largest and most liquid of the financial market with an estimated $1 trillion traded everyday [1]. Foreign exchange rates are among the most important economic indices in the international monetary markets. The trading of currencies has grown enormously due to the general trend of globalization, the increase of the import and export of commodities all over the world, and an increased interest in international investments [2]. The ability to simulate exchange rate prediction quickly and accurately is of crucial important in the trading market operations.

The traditional methods for exchange rate forecasting are based around statistical approaches. This includes Moving Average, Autoregressive model, Autoregressive Moving Average model, linear regression and exponential smoothing. None of these methods are completely satisfactory due to the nonlinear nature of most of the financial time series. With the growth of cheap computing power, there has been in recent years an increased interest in non-linear models, chief amongst these models are neural networks [3].

The applications of neural network for financial time series prediction have shown better performance in comparison to statistical methods because of its nonlinear nature and learning capability. However, not all of these researches can be used in real commercial applications. This is normally because the size of the neural networks can be potentially so large, therefore preventing the problem solution from being commercialized in the real world applications.

In this paper, a new recurrent network with ridge polynomial structure is introduced. The proposed network combines the properties of both higher-order and recurrent neural networks and it is called Dynamic Ridge Polynomial Neural Network (DRPNN). The structure of DRPNN is similar to the feedforward Ridge Polynomial Neural Network [4] with the addition of feedback connections. The network was used to predict the one step ahead and five steps ahead of the daily exchange rates between the British Pound and the Euro (UK/EU) and the Japanese Yen to British Pound (JP/UK).

2. Ridge Polynomial Neural Network

The use of Higher Order Neural Network (HONN) [5] is circumvented by the fact that the higher the order of the network, the more complex the network becomes and learning is significantly slower. A simple yet efficient alternative to HONN is the Pi-Sigma Neural Network (PSNN) [6]. A PSNN is a feedforward network with a single "hidden" layer of summing units and a product unit in the output layer. The motivation was to develop a systematic method for maintaining the fast learning property and powerful mapping capability of single layer HONN whilst avoiding the combinatorial explosion in the number of free parameters when the input dimension is increased. In contrast to HONN, the number of free parameters in PSNN increases linearly to the order of the network. Figure 1 shows a PSNN, whose output is determined according to the following equations:

$$Y = \sigma\Big(\prod_{j=1}^{K}\sum_{k=1}^{N}\big(W_{kj}X_k + W_{jo}\big)\Big) \qquad (1)$$

where W_{kj} are adjustable weights, W_{j0} are the biases of the summing units, X_k is the input vector, K is the number of summing units (alternatively, the order of the network), N is number of input nodes, and 'σ' is a nonlinear transfer function.

For each increase in order, only one extra summing unit is required. The product units give the networks higher-order capabilities without suffering from the exponential increase in weights, which is a major problem in a single layer HONN. Shin and Ghosh [6] argued that PSNN not only requires less memory (weights and nodes), but typically needs at least two orders of magnitude less number of computations as compared to the MLP for similar performance level, and over a broad class of problems. The presence of only one layer of adaptive weights results in fast learning, however the network is not a universal approximator.

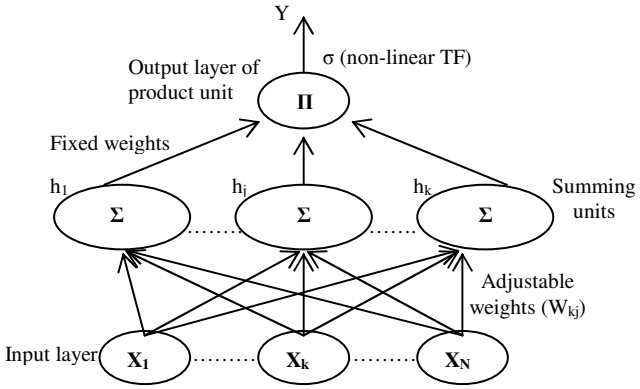

Fig. 1. Pi-Sigma Neural Network of *K*-th order

A generalization of PSNN is the Ridge Polynomial Neural Network (RPNN). RPNN, as shown in Figure 2, is a feedforward network which is constructed by the addition of PSNNs of varying orders until the desired mapping task is carried out with the required degree of accuracy. The network provides a natural mechanism for incrementally growing the networks until it is of appropriate size, and the network decides which higher order terms are necessary for the task at hand. Similar to PSNN, RPNN has only a single layer of adaptive weights; therefore the network preserves all the advantages of PSNN.

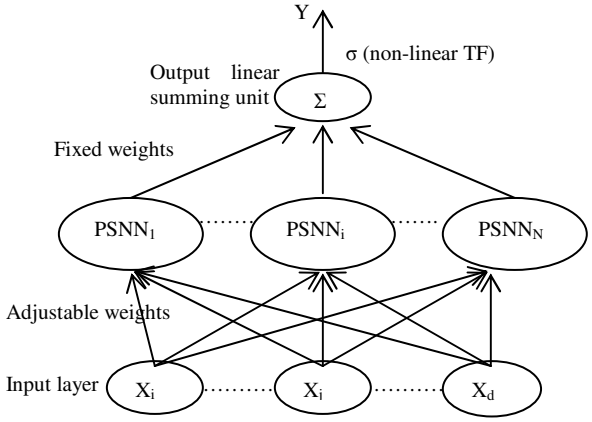

Fig. 2. Ridge Polynomial Neural Network of *N*-th order

Any multivariate polynomial can be represented in the form of a ridge polynomial and realized by RPNN whose output is determined according to the following equations [4]:

$$f(x) = \sigma\sum_{i=1}^{N} P_i(x)$$
$$P_i(x) = \prod_{j=1}^{i}\Big(\big\langle X, W_j\big\rangle + W_{j0}\Big), i = 1,\ldots..,N. \qquad (2)$$

where N is the number of PSNN blocks used, 'σ' denotes a suitable nonlinear transfer function, typically the sigmoid transfer function, W_{j0} are the biases of the summing units in the corresponding PSNN units, N is the number of PSNN units used (or alternatively, the order of the RPNN), and $\langle X,W\rangle$ is the inner product of weights matrix W, and input vector X.

3. Dynamic Ridge Polynomial Neural Network

In order to model dynamical properties of financial time series, it is essential to utilize a system that is capable of storing internal states and implementing complex dynamics. Since the behavior of the time

series itself related to some past inputs on which the present inputs depends, the introduction of recurrence feedback in a network will lead to a proper input-output mapping. Motivated by the ability of recurrent dynamic systems in real world applications, in this work, we propose the extension of the functionality and architecture of feedforward RPNN by introducing a feedback from the output layer to the input layer in order to represent a dynamic system for financial time series prediction. As shown in Figure 3, the new recurrent neural network is called Dynamic Ridge Polynomial Neural Network (DRPNN). The structure of DRPNN is constructed from a number of increasing order Pi-Sigma units. The feedback connection feeds the activation of the output node to the summing nodes in each Pi-Sigma units, thus allowing each block of Pi-Sigma unit to see the resulting output of the previous patterns. In contrast to RPNN, the proposed DRPNN is provided with memories which give the network the ability of retaining information to be used later. All the connection weights from the input layer to the first summing layer are learnable, while the rest are fixed to unity.

The rational of placing the recurrent connection from the output layer back to the input layer in the proposed DRPNN is that instead of learning with complex and fully connected recurrent architectures, redundant connections should be eliminated in order to significantly increase the network's generalization capability. This architecture is similar to the Jordan recurrent network [7]. The feedforward part of Jordan network is a restricted case of a non-linear Autoregressive Model (AR), while the configuration with context units fed by the output layer is a restricted case of non-linear Moving Average model (MA) [8]. From this, the proposed DRPNN which has the feedback connection from the output layer to the input layer is seen to have an advantage over feedforward RPNN in much the same way that ARMA models have advantages over the AR.

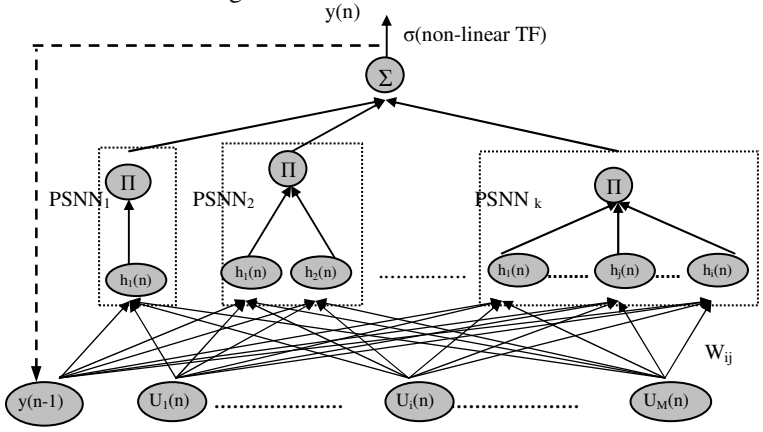

Fig. 3. Dynamic Ridge Polynomial Neural Network of *K*-th order

Suppose that M is the number of external inputs $U(n)$ to the network, and let $y(n-1)$ to be the output of the DRPNN at previous time step. The overall input to the network are the concatenation of $U(n)$ and $y(n-1)$, and is referred to as $Z(n)$ where:

$$Z_i(n) = \begin{cases} U_i(n) & if \;\; 1 \le \;\; i \le \;\; M \\ y(n-1) & i = M+1 \end{cases} \tag{3}$$

The output of the k_{th} order DRPNN is determined as follows:

$$\begin{aligned} y(n) &= \sigma\left(\sum_{i=1}^{k} P_i(n) \right) \\ P_i(n) &= \prod_{j=1}^{i} \left(h_j(n) \right) \quad . \\ h_j(n) &= \sum_{i=1}^{M+1} W_{ij} Z_i(n) + W_{jo} \end{aligned} \tag{4}$$

where k is the number of Pi-Sigma units used, $P_i(n)$ is the output of each PSNN block, $h_j(n)$ is the net sum of the sigma unit in the corresponding PSNN block, W_{jo} is the bias, σ is the sigmoid activation function, and n is the current time step.

DRPNN uses a constructive learning algorithm based on the asynchronous updating rule of the Pi-Sigma unit. The network adds a Pi-Sigma unit of increasing order to its structure when the difference between the current and the previous errors is less than a predefined threshold value. DRPNN follows the same training steps used in feedforward RPNN [4], in addition to the Real Time Recurrent Learning algorithm [9] for updating the weights of the Pi-Sigma unit in the network.

A standard error measure used for training the network is the Sum Squared Error:

$$E(n) = \frac{1}{2} \sum e(n)^2 \tag{5}$$

The error between the target and forecast signal is determined as follows:

$$e(n) = d(n) - y(n) \tag{6}$$

where $d(n)$ is the target output at time n, $y(n)$ is the forecast output at time n.

At every time n, the weights are updated according to:

$$\Delta W_{kl}(n) = \; -\eta \left(\frac{\partial E(n)}{\partial W_{kl}} \right) \tag{7}$$

where η is the learning rate.

The value $\left(\dfrac{\partial E(n)}{\partial W_{kl}} \right)$ is determined as:

$$\left(\frac{\partial E(n)}{\partial W_{kl}}\right) = e(n)\frac{\partial y(n)}{\partial W_{kl}} \tag{8}$$

$$\frac{\partial y(n)}{\partial W_{kl}} = \frac{\partial y(n)}{\partial P_i(n)}\frac{\partial P_i(n)}{\partial W_{kl}} \tag{9}$$

where

$$\frac{\partial y(n)}{\partial P_i(n)} = f'\left(\sum_{i=1}^{k} P_i(n)\right)\left(\prod_{\substack{j=1 \\ j\neq i}}^{i} h_j(n)\right) \tag{10}$$

and

$$\frac{\partial P_i(n)}{\partial W_{kl}} = \left(W_{ij}\frac{\partial y(n-1)}{W_{kl}}\right) + Z_j(n)\delta_{ik} \tag{11}$$

where δ_{ik} is the Krocnoker delta. Assume D as the dynamic system variable (the state of the ij^{th} neuron), where D is:

$$D_{ij}(n) = \frac{\partial y(n)}{\partial W_{kl}} \tag{12}$$

Substituting Equation (10) and (11) into (9) results in:

$$D_{ij}(n) = \frac{\partial y(n)}{\partial W_{kl}} = f'\left(\sum_{i=1}^{k} P_i(n)\right) \times \left(\prod_{\substack{j=1 \\ j\neq i}}^{i} h_j(n)\right)\left(W_{ij}D_{ij}(n-1) + Z_j(n)\delta_{ik}\right) \tag{13}$$

Then the weights updating rule is:

$$\begin{aligned} \Delta W_{ij}(n) &= \eta e(n) D_{ij}(n) + \alpha \Delta W_{ij}(n-1) \\ W_{ij}(n+1) &= W_{ij}(n) + \Delta W_{ij}(n) \end{aligned} \tag{14}$$

where W_{ij} are adjustable weights and ΔW_{ij} are total of weight changes.

DRPNN use the following steps to update its weights:

1. Start with low order DRPNN
2. Carry out the training and update the weights asynchronously after each training pattern.
3. When the observed change in error falls below the predefined threshold r, i.e., $\left|\frac{(e(n)-e(n-1))}{e(n-1)}\right| < r$, a higher order PSNN is added.
4. The threshold, r, for the error gradient together with the learning rate, n, are reduced by a suitable factor *dec_r* and *dec_n*, respectively.
5. The updated network carries out the learning cycle (repeat steps 1 to 4) until the maximum number of epoch is reached.

Notice that every time a higher order PSNN is added, the weights of the previously trained PSNN networks are kept frozen, whilst the weights of the latest added PSNN are trained.

4. Forecasting the exchange rates

Financial time series are among the best application domains for intelligent processing and advanced learning techniques [10]. The prediction of financial time series is an interesting problem to traders and individuals. Researchers and practitioners have been striving for an explanation of the movement of financial time series. To maximize profits from the liquidity market, forecasting techniques have been used by different traders. Assisted by powerful computer technologies, traders no longer rely on a single technique to provide information about the future of the market. Thus, various kinds of forecasting methods have been developed by many researchers and experts [11]. From statistical to artificial intelligence, there are various choices of techniques which can be used to make a forecast. Current research have shown that neural networks are promising tools for forecasting financial times series [12], as they were most implemented in mapping the underlying movement in the financial market

Two daily exchange rate signals are considered in this paper; the UK/EU and JP/UK exchange rates. The signals were obtained from a historical database provided by DataStream® [13], dated from 03/01/2000 until 04/11/2005, giving a total of 1525 data points.

Table 1. Calculations for input and output variables

	Indicator	Calculations
Input variables	EMA15	$p(i) = EMA_{15}(i)$ $EMA_n(i) = \frac{\alpha^0 p_i + \alpha^1 p_{i-1} + \alpha^2 p_{i-2} + \ldots + \alpha^{n-1} p_{i-n+1}}{\alpha^0 + \alpha^1 + \alpha^2 + \ldots + \alpha^{n-1}}$
	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+k	$(p(i+k) - p(i)) / p(i) * 100$ $p(i) = EMA_3(i)$

where $EMA_n(i)$ is the n-day exponential moving average of the i-th day, $p(i)$ is the signal of the i-th day, α is weighting factor, and k is forecast horizon; 1 or 5.

Financial data exhibit high volatility, complex, nonlinear, and noise properties. The Prediction of financial time series is very difficult and a nontrivial problem since it depends on several known and

unknown factors, and frequently data used for the prediction is noisy, uncertain and incomplete. The series are affected by many highly correlated economic, political and even psychological factors. As a result, they need adequate pre-processing before presenting them to the neural network. To smooth out the noise and to reduce the trend, the original raw data was pre-processed into a stationary series by transforming them into measurements of relative different in percentage of price (RDP) [14]. The advantage of this transformation is that the distribution of the transformed data will become more symmetrical and will follow more closely to normal distribution, as illustrated in the histogram plots in Figure 4. The calculations for the transformation of input and output variables are presented in Table 1. The RDP series were subsequently scaled using standard minimum and maximum normalization method which then produces a new bounded dataset.

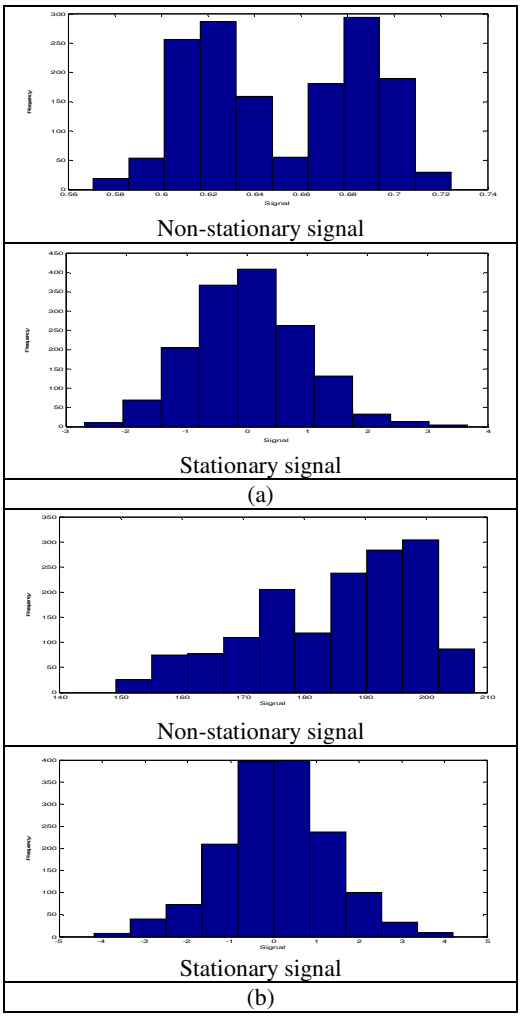

Fig. 4
(a) Histogram of UK/EU signal before and after pre-processing
(b) Histogram of JP/UK signal before and after pre-processing

5. Simulation results

In this simulation, we focus on how the networks generate profits, as this is the main interest in financial time series forecasting. Therefore, during generalization, the network model that endows the highest profit on unseen data is considered the best model. The prediction performance of all networks was evaluated using two financial metrics (refer to Table 2); the Annualized Return (AR) and Maximum Drawdown (MD), where the objective is to use the networks predictions to make money [3], and one statistical metric; the Normalized Mean Squared Error (NMSE) which is used to measure the deviation between the actual and the predicted signals [15].

For all neural networks, an average performance of 20 trials was used and the network parameters were experimentally selected as shown in Table 3. A sigmoid activation function was employed and all networks were trained with a maximum of 3000 epochs. MLP and RPNN were trained with the incremental backpropagation algorithm [16] and constructive learning algorithm [4], respectively. We trained the DRPNN with the learning algorithm as described in section 3. The MLP was trained with one hidden layer, and the hidden nodes were experimentally varied from 3 to 8, whereas for RPNN and DRPNN, the network's order was incrementally grown from 1 to 5.

The simulation results of the proposed DRPNN are benchmarked against the MLP and the ordinary feedforward RPNN. Tables 4 and 5 demonstrate the average results for the AR, MD, and NMSE obtained on out-of-sample data for the prediction of one step ahead and five steps ahead, respectively. Results on the Annualized Return (AR) from both Tables 4 and 5 obviously demonstrate that the proposed DRPNN profitably attained the highest profit return when used to forecast all the exchange rate signals compared to other network models. DRPNN successfully outperformed other networks on the average AR by 6.53% to 7.73% (Table 4) and 0.24% to 0.93% (Table 5). By looking at the Maximum Drawdown (MD), results in Tables 4 and 5 clearly show that the best values were mostly dominant by DRPNN, except for the prediction of one step ahead UK/EU in which feedforward RPNN gave better MD. This suggests that DRPNN have lower maximum loss and less downside risk compared to other networks when predicting the financial signals. It is worth pointing here that for the AR and MD, a bigger value is preferable. When measuring the NMSE, it can be noticed that DRPNN outperformed other networks with lower NMSE when

used to predict the one step ahead RDP. On the other hand, for the prediction of five steps ahead MLP achieved the lowest NMSE on both signals.

Table 2. Performance metrics and their calculations

Metrics	Calculations
AR	$AR = \frac{Profit}{All\ profit} * 100$ $Profit = \frac{252}{n} * CR, \quad CR = \sum_{i=1}^{n} R_i$ $R_i = \begin{cases} +\lvert y_i \rvert & if\ (y_i)(\hat{y}_i) \geq 0, \\ -\lvert y_i \rvert & otherwise \end{cases}$ $All\ profit = \frac{252}{n} * \sum_{i=1}^{n} abs(R_i)$
MD	$MD = min\left(\sum_{t=1}^{n} \left(CR_t - max\left(CR_1, ..., CR_t \right) \right) \right)$ $CR_t = \sum_{i=1}^{t} R_i, t = 1, ..., n$ $R_i = \begin{cases} +\lvert y_i \rvert & if\ (y_i)(\hat{y}_i) \geq 0, \\ -\lvert y_i \rvert & otherwise \end{cases}$
NMSE	$NMSE = \frac{1}{\sigma^2 n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ $\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2$ $\bar{y} = \sum_{i=1}^{n} y_i$

n is the total number of data patterns

y and $\hat{y}$ represent the target and predicted output value, respectively

Table 3. Parameters used in all networks

Neural Networks	Learning Rate (n)	dec_n	Threshold (r)	dec_r
MLP	0.1 or 0.05	-	-	-
RPNN DRPNN	[0.05, 0.5]	0.8	[0.00001, 0.7]	[0.05,0.2]

The average number of epochs reached during the training of the networks for the prediction of one step ahead and five steps ahead are shown in Tables 6 and 7, respectively. In the same tables, the amount of CPU time used to learn all the signals is presented in order to compare the speed of the networks to execute and complete the training. The CPU time was based on a machine with Windows XP 2000, Intel processor (Pentium 4), CPU of 3.00 GHz, and 1 GB of RAM.

Table 4. Average results for the prediction of one step ahead RDP

Performance Measures	Neural Networks	UK/EU	JP/UK
AR(%)	MLP	69.653	74.169
	RPNN	69.430	74.243
	DRPNN	77.164	80.776
MD	MLP	-0.572	-0.645
	RPNN	-0.564	-0.648
	DRPNN	-0.568	-0.495
NMSE	MLP	0.451	0.462
	RPNN	0.456	0.452
	DRPNN	0.366	0.374

Table 5. Average results for the prediction of five steps ahead RDP

Performance Measures	Neural Networks	UK/EU	JP/UK
AR(%)	MLP	86.645	88.971
	RPNN	86.644	89.252
	DRPNN	87.573	89.497
MD	MLP	-1.543	-1.983
	RPNN	-1.431	-1.488
	DRPNN	-1.013	-1.355
NMSE	MLP	0.221	0.208
	RPNN	0.231	0.209
	DRPNN	0.223	0.212

Table 6. Average epoch and CPU time usage for the prediction of one step ahead RDP

The Networks	Measures	UK/EU	JP/UK
MLP	Epoch	415	638
	CPU time	139	315
RPNN	Epoch	172	142
	CPU time	43	67
DRPNN	Epoch	132	96
	CPU time	37	38

Table 7. Average epoch and CPU time usage for the prediction of five steps ahead RDP

The Networks	Measures	UK/EU	JP/UK
MLP	Epoch	1365	1179
	CPU time	299.72	311.08
RPNN	Epoch	44	298
	CPU time	6.05	51.84
DRPNN	Epoch	57	42
	CPU time	38.97	21.77

Results in Table 6 show that the proposed DRPNN reveal to use the least number of epochs to converge during the training and took least CPU time to learn the signals in comparison to other networks. Meanwhile, results from Table 7 demonstrate that DRPNN made the fastest convergence and took least CPU time when learning the JP/UK signals. For the prediction of UK/EU signal, RPNN revealed to use smaller number of epochs and finish the training in shorter CPU time. In both Tables 6 and 7, MLP appeared to utilize more

epochs to complete the training and used longer CPU time in comparison to other networks.

For demonstration purpose, Figure 5 shows the best prediction on out of sample signal using DRPNN. In order to give a closer view, the plots depict just part of the prediction, which are the first 100 data points from the out of sample signal. As it can be noticed, the plots for the original and predicted signals are very close to each other and at some points they are nearly overlapping. This indicates that DRPNN are capable of learning the behavior of chaotic and highly non-linear financial data and they can capture the underlying movements in financial markets.

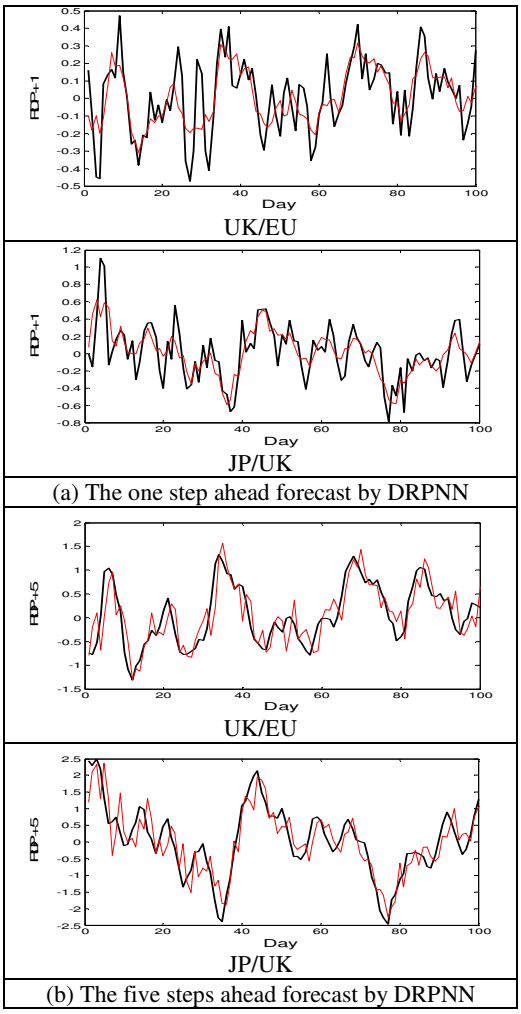

Fig. 5
(a) DRPNN forecast the one step ahead RDP
(b) DRPNN forecast the five steps ahead RDP
original signal, predicted signal

6. Discussions

The use of Dynamic Ridge Polynomial Neural Network in the exchange rate time series prediction showed that the proposed network provides a promising tool to forecasting. The network offers the following advantages:

- It provides better prediction in terms of the profit returns in comparison to other neural network architectures. The prediction attained by the DRPNN for the UK/EU and JP/UK exchange rates is significantly better than the prediction generated by the feedforward RPNN and the MLP.
- In view of the fact that the behavior of the financial signal itself related to some past inputs on which the present inputs depends, it therefore requires explicit treatment of dynamics. The merit of DRPNN, as compared to the feedforward RPNN is its increased inherited nonlinearity which results from the use of recurrent neural network architecture, giving it an advantage when dealing with financial time series forecasting.
- The proposed network demonstrated faster training when used to learn the signals in comparison to other network models.

Simulation results clearly demonstrate that the proposed DRPNN is potentially profitable and beneficial as money-making predictor. The network manifests highly nonlinear dynamical behavior induced by the recurrent feedback, therefore leads to a better input-output mapping and a better forecast. With the recurrent connection, the network outputs depend not only on the initial values of external inputs, but also on the entire history of the system inputs. Therefore, the DRPNN is provided with memory which gives the network the ability of retaining information to be used later. The superior performance of DRPNN is attributed to the natural mechanism for incremental network growth, therefore giving the network a very well regulated structure and smaller network size which led to network robustness. The presence of higher order terms in the network equipped the DRPNN with the ability to forecast the upcoming trends in financial time series signals. The network can robustly process the underlying dynamics of a nonlinear environment with a vast speed in convergence time. A noteworthy advantage of DRPNN and feedforward RPNN is the fact that there is no requirement to select the number of hidden units as in the MLP.

7. Conclusion

This work underlines the predictive capability and an important contribution of a new developed Dynamic Ridge Polynomial Neural Network; namely its elegant ability to approximate nonlinear financial time series. The performance of the network was tested for the prediction of one step ahead and five steps ahead of nonlinear UK/EU and JP/UK exchange rate signals. The extensive simulation results were benchmarked against the performance of the feedforward RPNN and the MLP. The DRPNN has shown its advantages in attaining higher profit return, vast speed of training, and low forecast error when compared to other network models. Hence, it is anticipated that DRPNN can be used as an alternative method for predicting financial variables and thus justified the potential use of this model by practitioners. The superior property hold by DRPNN could promise more powerful applications in many other real world problems.

8. References

[1] Yao, J. and Tan, C. L. (2000). A case study on neural networks to perform technical forecasting of forex. *Neurocomputing 34*, pp. 79-98.

[2] Schwaerzel, R. (1996). *Improving the prediction accuracy of financial time series by using multi-neural network systems and enhanced data preprocessing*. Thesis, Master of Science, The University of Texas at San Antonio.

[3] Dunis, C, L. and Williams, M. (2002) Modeling and trading the UER/USD exchange rate: Do Neural Network models perform better?. *In derivatives Use, Trading and Regulation*, Vol. No. 8, 3, pages 211-239.

[4] Shin, Y. and Ghosh, J. (1995). Ridge Polynomial Networks. *IEEE Transactions on Neural Networks*, Vol.6, No.3, pp.610-622.

[5] Giles, C. L. and Maxwell, T. (1987) Learning, invariance and generalization in high-order neural networks. *In Applied Optics*, vol. 26, no. 23, Optical Society of America, Washington D. C., pp.4972-4978.

[6] Shin, Y. and Ghosh, J. (1991). The Pi-Sigma Networks: An Efficient Higher-order Neural Network for Pattern Classification and Function Approximation. *Proceedings of International Joint Conference on Neural Networks, Vol.1*, pp.13-18.

[7] Jordan, M. I. (1986) Serial order: A parallel distributed processing approach. Institute for Cognitive Science report 8604, Institute for Cognitive Science, University of California, San Diego.

[8] Beale, R. and Jackson, T. (1990). *Neural Computing: An Introduction.* Bristol: Hilger.

[9] Williams, R.J., Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1*. pp. 270-280.

[10] Magdon-Ismail, M., Nicholson, A., Abu-Mostafa, Y.S. (1998). Financial Market: Very Noisy Information Processing. *Proceedings of the IEEE*, Vol. 86, No.11, pp. 2184-2195.

[11] Yao, J. and Tan, C. L. (2000). A case study on neural networks to perform technical forecasting of forex. *Neurocomputing 34*, pp. 79-98.

[12] Hsiao1, K., Li, J-B. and Chen, A-P. (2006). Improving Investing Strategy in Stock Market with Valuation Technology Clustering and Neural Network. *International Journal of Computational Intelligence Research.* Vol.2, No. 1, pp. 26-32

[13] *Datastream International Limited ALL RIGHTS RESERVED/* Datastream database, [machine-readable data] London, Thompson Financial Limited [producer and distributor], retrieved November 14, 2005.

[14] Thomason, M. (1999-a). The practitioner method and tools. *Journal of Computational Intelligence in Finance*, vol. 7, no. 3, pp. 36-45.

[15] Cao, L. J. and Tay, F. E. H. (2003). Support Vector Machine with Adaptive Parameters in Financial time Series Forecasting. *IEEE Transactions on Neural Networks*, Vol. 14, No. 6. pp. 1506-1518.

[16] Haykin, S. (1999) *Neural Networks. A comprehensive foundation*. Second Edition, Prentice-Hall, Inc., New Jersey.