Expert Systems with Applications 38 (2011) 3765–3776

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Dynamic Ridge Polynomial Neural Network: Forecasting the univariate non-stationary and stationary trading signals

Rozaida Ghazali [a,*], Abir Jaafar Hussain [b], Panos Liatsis [c]

[a] Information Technology and Multimedia Faculty, Universiti Tun Hussein Onn Malaysia, Malaysia
[b] School of Computing and Mathematical Sciences, Liverpool John Moores University, UK
[c] School of Engineering and Mathematical Sciences, City University, London, UK

ARTICLE INFO

Keywords:
Dynamic Ridge Polynomial Neural Network
Financial signals
Higher order neural network
Time series prediction

ABSTRACT

This paper considers the prediction of noisy time series data, specifically, the prediction of financial signals. A novel Dynamic Ridge Polynomial Neural Network (DRPNN) for financial time series prediction is presented which combines the properties of both higher order and recurrent neural network. In an attempt to overcome the stability and convergence problems in the proposed DRPNN, the stability convergence of DRPNN is derived to ensure that the network posses a unique equilibrium state. In order to provide a more accurate comparative evaluation in terms of profit earning, empirical testing used in this work encompass not only on the more traditional criteria of NMSE, which concerned at how good the forecasts fit their target, but also on financial metrics where the objective is to use the networks predictions to generate profit. Extensive simulations for the prediction of one and five steps ahead of stationary and non-stationary time series were performed. The resulting forecast made by DRPNN shows substantial profits on financial historical signals when compared to various neural networks; the Pi-Sigma Neural Network, the Functional Link Neural Network, the feedforward Ridge Polynomial Neural Network, and the Multilayer Perceptron. Simulation results indicate that DRPNN in most cases demonstrated advantages in capturing chaotic movement in the financial signals with an improvement in the profit return and rapid convergence over other network models.

1. Introduction

Neural networks have been shown to be a promising tool for forecasting financial times series. Numerous research and applications of neural networks in business have proven their advantage in relation to classical methods that do not include artificial intelligence. What makes this particular use of neural networks so attractive to financial analysts and traders is the fact that governments and companies benefit from it to make decisions on investment and trading. However, when the number of inputs to the model and the number of training examples becomes extremely large, the training procedure for ordinary neural network architectures becomes tremendously slow and unduly tedious. To overcome such time-consuming operations, this research work focuses on using various higher order neural networks (HONNs) which have a single layer of learnable weights, therefore reducing the networks' complexity. HONNs contain summing unit and product units that multiply their inputs. These high order terms or product units can increase the information capacity of higher order network in comparison to standard neural networks with summation units only. The utilization of higher order terms allows the neural networks to expand the input space into a higher dimensional space where linear separability is possible.

HONNs have applications in wide range areas of human interests such as pattern recognition (Artyomov & Pecht, 2004), function approximation (Ghosh & Shin, 1992; Shin & Ghosh, 1995), process optimization (Cass & Radl, 1996), system identification (Mirea & Marcu, 2002), image processing (Hussain & Liatsis, 2002), classification (Shin & Ghosh, 1995), time series prediction (Tawfik & Liatsis, 1997), and intelligent control (Patra & Bos, 2000). Nevertheless, literatures on the use of HONNs for financial time series prediction are limited and have not been adequately addressed.

2. The networks

Functional Link Neural Network (FLNN) (Giles & Maxwell, 1987) is a type of HONN which naturally extends the family of theoretical feedforward network structure by introducing nonlinearities in input patterns enhancements. The network, however, suffers from the combinatorial explosion in the number of weights, when the order of the network becomes excessively high.

* Corresponding author. Tel.: +6 07 4538066; fax: +6 07 4532199.
E-mail address: rozaida@uthm.edu.my (R. Ghazali).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.09.037

A simple yet efficient alternative to FLNN is the Pi-Sigma Neural Network (PSNN) which was proposed by Shin and Ghosh (1991). PSNN was introduced to overcome the problem of weight explosion in FLNN. The network has a regular structure and requires a smaller number of free parameters, when compared to other single layer HONNs. However, PSNN is not a universal approximator (Shin & Ghosh, 1995).

A generalization of PSNN is the Ridge Polynomial Neural Network (RPNN) (Shin & Ghosh, 1995). The network has a well regulated structure which is constructed by the addition of PSNNs of varying orders. Contrary to the FLNN, which utilizes multivariate polynomials, thus leading to an explosion in the number of free parameters, RPNN uses univariate polynomials which are easy to handle. RPNN is a universal approximator (Shin & Ghosh, 1995), and the network maintains the fast learning and powerful mapping properties of single layer HONNs and avoids the explosion of weights, as the number of inputs increases. 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 (Shin & Ghosh, 1995):

$$
\begin{aligned}
f(x) &= \sigma \sum_{i=1}^{N} P_i(x) \\
P_i(x) &= \prod_{j=1}^{i} \left(\langle X, W_j \rangle + W_{j0} \right), \quad i = 1, \ldots, N
\end{aligned}
\tag{1}
$$

where '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

Applications in forecasting and signal processing require explicit treatment of dynamics. The behavior of the financial signal itself related to some past inputs on which the present inputs depends. The inherent nonlinearity of financial time series can prevent a single neural network from being able to accurately forecast an extended trading period even if it could forecast changes in the testing data. To overcome the problems associated with neural networks when used for financial time series forecasting; in this work, a new dynamically sized higher order recurrent neural network architecture is proposed. The network will start with small basic structure, which will grow as the learning proceeds until the desired mapping task is carried out with the required degree of accuracy. The network is called the Dynamic Ridge Polynomial Neural Network (DRPNN). DRPNN has the extension architecture and functionality of the ordinary feedforward RPNN.

The structure of the DRPNN is constructed from a number of increasing order of Pi-Sigma units with the addition of a feedback connection from the output layer to the input layer. The feedback connection feeds the activation of the output node to the summing nodes in each Pi-Sigma units, thus allowing each building block of Pi-Sigma unit to see the resulting output of the previous patterns. In contrast to RPNN, the proposed DRPNN, as shown in Fig. 1 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.

This architecture of DRPNN is similar to the Jordan recurrent network (Jordan, 1986). The feedforward part of Jordan network is a restricted case of a non-linear AR model, while the configuration with context units fed by the output layer is a restricted case of non-linear MA model (Beale & Jackson, 1990). 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.

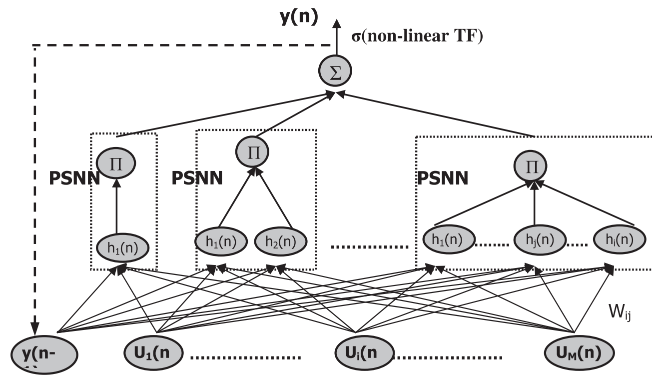

Fig. 1. Dynamic Ridge Polynomial Neural Network of Kth order.

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:

$$
Z_i(n) = \begin{cases} U_i(n) & \text{if } 1 \leqslant i \leqslant M \\ y(n-1) & i = M+1 \end{cases}
\tag{2}
$$

The output of the kth order DRPNN is determined as follows:

$$
\begin{aligned}
y(n) &= \sigma\left(\sum_{i=1}^{k} P_i(n) \right), \\
P_i(n) &= \prod_{j=1}^{i} \left(h_j(n) \right), \\
h_j(n) &= \sum_{i=1}^{M+1} W_{ij} Z_i(n) + W_{jo},
\end{aligned}
\tag{3}
$$

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 Real Time Recurrent Learning algorithm (Williams & Zipser, 1989) 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:

$$
E(n) = \frac{1}{2} \sum e^2(n)
\tag{4}
$$

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

$$
e(n) = d(n) - y(n)
\tag{5}
$$

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:

$$
\Delta W_{kl}(n) = -\eta \left(\frac{\partial E(n)}{\partial W_{kl}} \right)
\tag{6}
$$

where η is the learning rate. The value $\left(\frac{\partial E(n)}{\partial W_{kl}} \right)$ is determined as:

$$\left(\frac{\partial E(n)}{\partial W_{kl}}\right) = e(n)\frac{\partial y(n)}{\partial W_{kl}} \tag{7}$$

$$\frac{\partial y(n)}{\partial W_{kl}} = \frac{\partial y(n)}{\partial P_i(n)}\frac{\partial P_i(n)}{\partial W_{kl}} \tag{8}$$

where

$$\frac{\partial y(n)}{\partial P_i(n)} = f'\left(\sum_{i=1}^{k} P_i(n)\right)\left(\prod_{\substack{j=1 \\ j\neq i}}^{i} h_j(n)\right) \tag{9}$$

and

$$\frac{\partial P_i(n)}{\partial W_{kl}} = \left(W_{ij}\frac{\partial y(n-1)}{W_{kl}}\right) + Z_j(n)\delta_{ik} \tag{10}$$

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

$$D_{ij}(n) = \frac{\partial y(n)}{\partial W_{kl}} \tag{11}$$

The state of a dynamical system is formally defined as a set of quantities that summarizes all the information about the past behavior of the system that is needed to uniquely describe its future behavior (Haykin, 1999). Substituting Eqs. (9) and (10) into (8) results in:

$$D_{ij}(n) = \frac{\partial y(n)}{\partial W_{kl}} = f'\left(\sum_{i=1}^{k} P_i(n)\right) \times \left(\prod_{\substack{j=1 \\ j\neq i}}^{i} h_j(n)\right)(W_{ij}D_{ij}(n-1) + Z_j(n)\delta_{ik}) \tag{12}$$

For simplification, the initial values for $D_{ij}(n-1) = 0$, and $Z_j(n-1) = 0.5$. Then the weights updating rule is:

$$\begin{aligned}\Delta W_{ij}(n) &= \eta e(n) D_{ij}(n) + \alpha \Delta W_{ij}(n-1) \\ W_{ij}(n+1) &= W_{ij}(n) + \Delta W_{ij}(n)\end{aligned} \tag{13}$$

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

DRPNN follows the following steps for updating 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., $\left|\left(\frac{(e(n)-e(n-1))}{e(n-1)}\right)\right| < r$, 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 1–4) 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. Issues of stability in DRPNN

One of the most useful properties of networks with recurrent connection is their ability to model the behavior of arbitrary dynamical system. Hence, the existence of feedback in the proposed DRPNN is expected to improve the performance of a given network. Despite the potential and capability of the DRPNN which comprises the recurrent connection, the problems of complexity and difficulty of training the network exist in the proposed DRPNN, which are:

- The states of the processing elements, denoted by D_{ij} in Eq. (11), affect both the output and the gradient. Therefore, calculating the gradients and updating the weights of a recurrent network is much more difficult.
- The network is more difficult to train than ordinary RPNN. The training algorithm could become unstable; the error between the target and the output of the DRPNN may not be monotonically decreasing, the gradient computation is more complicated, and the convergence time may be long.

In an attempt to overcome the stability and convergence problems in the proposed DRPNN, the convergence of the DRPNN is derived to ensure that the network posses a unique equilibrium state (see Appendix A). Based on the stability theorem for a general network proposed by Atiya (1988) and shown in Eq. (A.1) (in Appendix A), any network that satisfies this theorem exhibits no other behavior except going to a unique equilibrium for a given input: From the given theorem, a unique fixed point is reached regardless of the initial condition. Therefore, the aim is to adjust the weights of the network, such that it allows the unique equilibrium state to move in a way that the output of the network goes as close as possible to the required output. This means that we are looking for a condition in the weight matrix. Derived from the stability convergence shown in Appendix A, the condition for DRPNN to converge is described by:

$$\left(\max\left(\sum_{k=1}^{A}\sum_{L=1}^{k}|W_{L(M+2)}| \cdot \prod_{\substack{S=1 \\ S\neq L}}^{k}\sum_{m=1}^{M+2}|W_{Sm}|\right)\right) < \frac{1}{(\max|f'|)} \tag{14}$$

Therefore, this work guarantees the stability of DRPNN for the equilibrium problem.

5. Financial time series forecasting

Financial time series prediction 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 many traders. Assisted by powerful computer technologies, traders no longer rely on a single technique to provide information about the future of the market. Over the past few years, neural networks have been widely advocated as a new alternative modeling method to more traditional econometric and statistical approaches, claiming increasing success in the fields of economic and financial forecasting. This has resulted in many publications comparing neural networks with traditional forecasting methods (Dunis & Williams, 2002; Yao & Tan, 2000; Yumlu, Gurgen, & Okay, 2005) and many more. This is not surprising since neural networks are capable of describing the dynamics of non-stationary time series due to their non-parametric, adaptive and noise tolerant properties.

A review on existing literature reveals financial studies on a wide variety of subjects such as stock price forecasting (Leung, Chen, & Daouk, 2000), exchange rate forecasting (Chen & Leung, 2005; Yao & Tan, 2000), returns prediction (Dunis & Williams, 2002), predicting government treasury bond (Cheng, Wagner, & Lin, 1996) and forecasting currency volatility (Yumlu et al., 2005). Neural networks are an emerging and challenging computational technology that can offer a new avenue to explore the dynamics of a variety of financial applications. They can make contributions to the maximization of returns, while reducing costs, and limiting risks.

6. Prediction of financial signals using Dynamic Ridge Polynomial Neural Network

In this work, 10 noisy financial time series signals are considered as shown in Table 1. All the signals were obtained from a historical database provided by Datastream® forepart from the IBM common stock closing price time series, which was taken from the Time Series Data Library (Hyndman, 1980). The networks are tested for the prediction of one and five steps ahead predictions of financial time series in which two methods are utilized; in the first method the data are passed directly to the neural network as non-stationary signals while in the second method the financial data are transformed into stationary signals.

Most financial data is non-stationary in nature, meaning that the statistical properties (e.g. mean and variance) of the data change over time. These changes are caused as a result of various business and economic cycles. For non-stationary signals, all the univariate data are presented to the networks directly without any transformation. The data are scaled between the upper and lower bounds of the transfer function. On the other hand, to have a stationary version of the signals, we did some series of transformations on the non-stationary signal. Therefore, we systematically investigate a method of pre-processing the original signals in order to reduce the influence of their trends. The idea of transforming the signal into the stationary version is due to the characteristics of the financial data which exhibit high volatility, complexity, and noise. Pre-processing and proper sampling of input data can give a significant impact on the forecasting performance. 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 difference in percentage of price (RDP) (Thomason, 1999). The calculations for the transformation of input and output variables are presented in Table 2. Subsequent to transformation, all the input and output variables in Table 2 were scaled between the upper and lower bounds of the transfer function in order to avoid computational problems and to meet algorithm requirements.

7. Training of the networks

The performance of the new proposed DRPNN is benchmarked against the performance of MLP, FLNN, PSNN, and RPNN. MLP, FLNN and PSNN which were trained using the incremental back-propagation learning algorithm (Haykin, 1999). Early stopping was utilized and each signal was divided into three data sets which are the training, validation and the out-of-sample data which represent 50%, 25%, and 25% of the entire data set, respectively. For FLNN and PSNN, the higher order terms were empirically selected between 2 and 5. The MLP were trained with hidden units varies from 3 to 8.

An early stopping method was not employed for the training of the RPNN and DRPNN. This is because every time a higher order PSNN unit is added to the networks, the monitored mean squared error will slightly increase before it gradually decreases. If an early stopping criteria was used, the training will usually stop after a PSNN unit is added, at the same time that new added PSNN is about to be trained. This will result in truncated and incomplete learning. Therefore, the signals were segregated into two partitions; the training and the out-of-sample data with a distribution of 75% and 25%, respectively. The RPNN were trained with a constructive learning algorithm (Shin & Ghosh, 1995). DRPNN follows the training steps discussed previously.

For DRPNN and RPNN, the networks were trained up to 5th order network architecture. In the case of DRPNN, the training of network is halted when the network learning become unstable and divert from the stability convergence. This condition is checked every time before adding a higher order Pi-Sigma unit to the network. In other words, when DRPNN does not satisfy the stability condition, as shown in Eq. (14), training is terminated. All network models used in this work were trained at a maximum epoch of 3000 with the experimentally chosen parameters as shown in Table 3.

The prediction performance of all networks was evaluated using three financial metrics (Dunis & Williams, 2002), where the objective was to use the networks predictions to make money, and three statistical metrics (Cao & Tay, 2003; Hussain & Liatsis, 2002) which provide accurate tracking of the signals, as shown in Table 4. In order to measure profits generated from the networks predictions, a simple trading strategy is used. If the network predicts a positive change for the next five day RDP, a 'buy' signal is sent, otherwise

Table 1
Time series data used.

No.	Time series data	Total
1	IBM common stock closing price (IBM) 17/05/1961–02/11/1962	360
2	Standard & Poor 500 stock index futures (CMESP) 01/01/1988–11/07/1995	1963
3	The United States 10-year government bond (CBT-10) 01/06/1989–11/12/1996	1965
4	The United States 30-year government bond (CBT-30) 01/10/1990–24/04/1998	1975
5	UK pound to EURO exchange rate (UK/EU) 03/01/2000–04/11/2005	1525
6	UK pound to US dollar exchange rate (UK/US) 03/01/2000–04/11/2005	1525
7	US dollar to EURO exchange rate (US/EU) 03/01/2000–04/11/2005	1525
8	Japanese Yen to EURO exchange rate (JP/EU) 03/01/2000–04/11/2005	1525
9	The Japanese Yen to US dollar exchange rate (JP/US) 03/01/2000–04/11/2005	1525
10	The Japanese Yen to UK pound exchange rate (JP/UK) 03/01/2000–04/11/2005	1525

Table 2
Calculations for transformation of input and output variables.

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

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

Table 3
Parameters used in all networks.

Neural networks	Learning rate (n)	dec_n	Threshold (r)	dec_r
MLP FLNN PSNN	0.1 or 0.05	–	–	–
RPNN DRPNN	[0.05,0.5]	0.8	[0.00001,0.7]	[0.05,0.2]

Table 4
Performance metrics and their calculations.

Annualized return (AR, %)	Normalized mean squared error (NMSE)
$AR = \frac{Profit}{Allprofit} * 100$	
$Profit = \frac{252}{n} * CR,\ CR = \sum_{i=1}^{n} R_i$	$NMSE = \frac{1}{\sigma^2 n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
$R_i = \begin{cases} +\lvert y_i \rvert & \text{if } (y_i)(\hat{y}_i) \geqslant 0, \\ -\lvert y_i \rvert & \text{otherwise} \end{cases}$	$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2$
$Allprofit = \frac{252}{n} * \sum_{i=1}^{n} abs(R_i)$	$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$
Maximum Drawdown (MD)	Signal to noise ratio (SNR)
$\mathrm{MD} = \min\left(\sum_{t=1}^{n} (\mathrm{CR}_t - \max(\mathrm{CR}_1, \ldots, \mathrm{CR}_t))\right)$	$\mathrm{SNR} = 10 * \log_{10}(sigma)$
$\mathrm{CR}_t = \sum_{i=1}^{t} R_i, t = 1, \ldots, n$	$sigma = \frac{m^2 * n}{SSE}$
	$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
$R_i = \begin{cases} +\lvert y_i \rvert & if (y_i)(\hat{y}_i) \geqslant 0, \\ -\lvert y_i \rvert & otherwise \end{cases}$	$m = \max(y_i)$
Annualized Volatility (VOL)	Correct directional change (CDC)
	$\mathrm{CDC} = \frac{1}{n} \sum_{i=1}^{n} d_i$
$\mathrm{VOL} = \sqrt{252} * \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (R_i - \overline{R})^2}$	$d_i = \begin{cases} 1 & \text{if } (y_i - y_{i-1})(\hat{y}_i - \hat{y}_{i-1}) \geqslant 0, \\ 0 & \text{otherwise} \end{cases}$

n is the total number of data patterns.
y and $\hat{y}$ represent the actual and predicted output value, respectively.

a 'sell' signal is sent. The ability of the networks as traders was evaluated by the annualized return (AR), a real trading measurement which used to test the possible monetary gains and to measure the overall profitability in a year, through the use of the 'buy' and 'sell' signals. Maximum Drawdown (MDD) is the minimum of the accumulated losses and is used as a risk assessment measure for various financial prediction models. It measures the downside risk, which is the maximum loss of the model during the sample period. Meanwhile, Annualized Volatility (VOL) is the measure of the changeability in asset returns, which means less volatility is preferable. It describes the variability in a stock price and it is used as an estimate of investment risk and for profit possibilities. The volatility is of great interest for financial analyst and provides useful information when estimating investment risk in real trading. The Normalized Mean Squared Error (NMSE) is used to measure the deviation between the actual and the predicted signals. The smaller the value of NMSE, the closer is the predicted signals to the actual signals. The Signal to Noise Ratio (SNR) provides the relative amount of useful information in a signal; as compared to the noise it carries. Correct Directional Change (CDC) measures the capacity of a model to correctly predict the subsequent actual change of a forecast variable.

7.1. Simulation results

As we are concerned with financial time series prediction, in these extensive experiments, our primary interest is to concentrate on the profitable value contained in the DRPNN predictions against other neural networks models. For this reason, the neural networks structure which provides the highest percentage of AR on out-of-sample data is considered the best model. Tables 5–8 summarize the average results of 20 simulations obtained on out-of-sample data for the prediction of both stationary and non-stationary signal, when used to predict one and five steps ahead.

The results of the Annualized Return (AR) from Tables 5–8 obviously demonstrated that the proposed DRPNN, in most cases, profitably attained the highest profit return compared to other network models. In the pertinent signals where DRPNN made the highest AR, the network successfully outperformed other networks on the average by 1.96–10.19% (Table 5), 0.25–2.70% (Table 6), 0.15–11.23% (Table 7), and 0.02–6.80% (Table 8). The results in Ta-

Table 5
Average results on stationary signals for the prediction of 1-step ahead.

Performance measures	Neural networks	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
AR (%)	MLP	81.108	80.408	81.309	78.531	69.653	76.431	76.649	73.794	74.330	74.169
	FLNN	81.428	80.235	80.371	78.885	69.482	76.627	76.135	73.514	74.129	74.210
	PSNN	80.998	79.634	80.011	78.980	69.494	77.463	76.808	74.354	74.723	73.983
	RPNN	82.350	80.736	81.307	79.433	69.430	78.262	76.791	75.090	74.836	74.243
	DRPNN	79.557	82.698	86.257	83.992	77.164	82.059	82.622	83.707	77.743	80.776
MD	MLP	−2.002	−0.683	−0.400	−0.759	−0.572	−0.869	−1.647	−0.762	−0.849	−0.645
	FLNN	−1.908	−0.683	−0.495	−0.759	−0.579	−0.869	−1.647	−0.753	−1.097	−0.651
	PSNN	−1.908	−0.683	−0.620	−0.759	−0.579	−0.838	−1.675	−0.753	−0.849	−0.648
	RPNN	−1.908	−0.683	−0.400	−0.759	−0.564	−0.785	−1.073	−0.753	−0.849	−0.648
	DRPNN	−1.908	−0.552	−0.342	−0.659	−0.568	−0.677	−0.709	−0.738	−0.797	−0.495
VOL	MLP	18.146	4.314	2.900	3.728	2.802	3.922	4.091	3.704	4.075	3.632
	FLNN	18.111	4.320	2.921	3.718	2.805	3.916	4.108	3.712	4.081	3.631
	PSNN	18.167	4.339	2.929	3.716	2.805	3.891	4.086	3.688	4.064	3.637
	RPNN	17.991	4.304	2.900	3.703	2.806	3.867	4.086	3.666	4.060	3.630
	DRPNN	18.348	4.240	2.781	3.568	2.658	3.748	3.878	3.390	3.972	3.451
NMSE	MLP	0.457	0.429	0.398	0.424	0.451	0.398	0.400	0.439	0.484	0.462
	FLNN	0.457	0.423	0.399	0.422	0.451	0.391	0.404	0.440	0.480	0.456
	PSNN	0.461	0.426	0.401	0.423	0.451	0.396	0.401	0.439	0.482	0.458
	RPNN	0.456	0.425	0.396	0.424	0.456	0.409	0.402	0.435	0.479	0.452
	DRPNN	0.417	0.330	0.301	0.342	0.366	0.318	0.330	0.362	0.399	0.374
CDC	MLP	59.350	65.130	65.440	66.780	63.250	62.480	66.270	64.800	61.950	60.280
	FLNN	57.740	65.030	65.380	67.380	62.930	62.790	66.670	63.200	62.330	60.160
	PSNN	57.920	64.810	65.870	67.020	63.520	63.110	66.480	63.760	61.720	60.630
	RPNN	59.400	65.310	65.720	67.220	63.910	62.430	64.990	65.130	61.850	61.450
	DRPNN	58.990	66.540	69.420	66.420	63.560	64.930	68.440	65.370	62.450	60.810
SNR (dB)	MLP	20.970	29.750	24.530	23.520	24.420	23.570	23.460	26.480	24.720	25.750
	FLNN	20.970	29.810	24.520	23.530	24.420	23.650	23.430	26.470	24.750	25.800
	PSNN	20.930	29.780	24.510	23.520	24.430	23.590	23.460	26.480	24.740	25.780
	RPNN	20.980	29.790	24.550	23.520	24.370	23.460	23.450	26.520	24.760	25.840
	DRPNN	21.370	30.890	25.750	24.450	25.330	24.540	24.310	27.320	25.560	26.670

Table 6
Average results on stationary signals for the prediction of 5-steps ahead.

Performance measures	Neural networks	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
AR (%)	MLP	89.402	85.645	86.103	88.688	86.645	88.134	87.880	87.052	83.551	88.971
	FLNN	90.212	85.900	86.274	89.170	85.643	88.058	87.458	87.336	84.752	88.841
	PSNN	90.104	85.584	86.168	88.730	86.345	87.987	87.536	87.056	83.526	88.860
	RPNN	90.713	85.644	86.596	88.751	86.644	87.145	88.319	87.483	84.837	89.252
	DRPNN	90.708	85.769	87.353	88.097	87.573	87.467	88.828	87.814	86.239	89.497
MD	MLP	−6.763	−2.046	−1.983	−1.307	−1.543	−1.518	−2.645	−1.799	−2.071	−1.983
	FLNN	−3.668	−2.045	−1.879	−1.085	−1.543	−1.518	−2.645	−1.522	−1.587	−1.983
	PSNN	−5.367	−2.045	−1.879	−1.151	−1.543	−1.558	−2.391	−1.833	−1.594	−1.983
	RPNN	−4.314	−2.045	−1.898	−1.262	−1.431	−1.514	−1.589	−1.862	−2.965	−1.488
	DRPNN	−3.617	−2.046	−1.607	−1.163	−1.013	−1.486	−1.452	−1.833	−2.746	−1.355
VOL	MLP	51.233	13.587	9.503	11.244	8.311	12.556	12.383	11.188	12.438	10.870
	FLNN	50.428	13.562	9.442	11.164	8.398	12.558	12.426	11.155	12.295	10.892
	PSNN	50.866	13.599	9.448	11.226	8.336	12.621	12.402	11.183	12.431	10.878
	RPNN	50.854	13.599	9.504	11.227	8.339	12.725	12.273	11.215	12.486	10.856
	DRPNN	50.007	13.617	9.336	11.306	8.307	12.668	12.271	11.093	12.659	10.799
NMSE	MLP	0.334	0.295	0.254	0.215	0.221	0.209	0.238	0.216	0.269	0.208
	FLNN	0.276	0.295	0.251	0.214	0.224	0.207	0.241	0.213	0.257	0.208
	PSNN	0.286	0.295	0.252	0.217	0.223	0.206	0.237	0.213	0.266	0.209
	RPNN	0.270	0.296	0.256	0.214	0.231	0.209	0.251	0.215	0.293	0.208
	DRPNN	0.306	0.291	0.255	0.214	0.223	0.216	0.258	0.216	0.303	0.212
CDC	MLP	64.110	64.810	67.990	65.580	66.010	60.350	66.240	64.690	58.490	59.510
	FLNN	64.180	64.510	67.940	64.800	65.930	59.970	66.220	64.640	58.520	59.040
	PSNN	65.000	64.630	67.520	65.100	66.550	60.050	66.150	64.040	58.730	59.920
	RPNN	63.330	64.800	67.130	65.120	65.170	61.130	64.230	64.240	58.590	59.680
	DRPNN	62.980	65.540	66.870	64.770	65.750	61.250	64.110	64.920	59.370	61.240
SNR (dB)	MLP	21.680	27.390	25.200	25.750	26.660	25.710	23.810	27.840	25.600	26.610
	FLNN	22.480	27.390	25.240	25.770	26.600	25.750	23.740	27.880	25.790	26.590
	PSNN	22.340	27.380	25.230	25.730	26.610	25.770	23.820	27.890	25.650	26.600
	RPNN	22.580	27.370	25.150	25.760	26.460	25.700	23.580	27.850	25.240	26.600
	DRPNN	22.050	27.450	25.180	25.780	26.620	25.560	23.460	27.830	25.090	26.540

Table 7
Average results on non-stationary signals for the prediction of 1-step ahead.

Performance measures	Neural networks	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
AR (%)	MLP	−4.612	−5.504	12.122	2.759	−5.730	−0.702	0.964	−2.158	−3.424	8.031
	FLNN	−9.035	−6.339	9.609	5.967	−1.652	−1.253	−0.374	−0.970	−1.651	4.786
	PSNN	−7.587	−6.239	8.695	5.688	−1.340	−0.073	−0.942	−0.707	−2.966	5.554
	RPNN	0.464	7.873	11.610	5.663	−0.793	1.796	1.051	−0.827	−4.146	7.785
	DRPNN	2.195	6.373	13.985	5.017	−0.296	1.296	0.084	−0.559	−2.370	8.779
MD	MLP	20.650	−18.442	−4.168	−8.701	12.139	12.344	−8.943	10.129	11.467	−6.690
	FLNN	23.248	−18.955	−5.548	−6.578	−9.223	12.592	−10.571	−8.654	10.754	−7.451
	PSNN	22.344	−20.437	−6.066	−6.503	−9.242	13.032	−10.011	−9.351	12.531	−7.036
	RPNN	17.662	−12.469	−4.596	−6.568	−9.186	10.299	−10.059	−8.131	13.636	−6.386
	DRPNN	16.521	−13.971	−4.481	−7.416	−9.166	−9.458	−8.800	−9.177	11.389	−6.564
VOL	MLP	31.634	9.285	6.021	7.842	5.598	8.240	8.912	7.935	8.642	7.405
	FLNN	31.582	9.288	6.030	7.835	5.599	8.240	8.909	7.938	8.645	7.412
	PSNN	31.626	9.281	6.031	7.836	5.601	8.238	8.910	7.935	8.643	7.411
	RPNN	31.673	9.272	6.024	7.835	5.602	8.235	8.904	7.936	8.639	7.406
	DRPNN	31.660	9.282	6.014	7.836	5.602	8.236	8.911	7.933	8.642	7.403
NMSE	MLP	0.266	3.379	0.027	0.070	0.485	0.208	2.004	12.887	0.076	0.385
	FLNN	2.578	0.035	0.023	0.035	0.108	2.228	0.092	0.222	0.052	0.153
	PSNN	3.817	1.383	0.033	0.086	1.690	1.268	0.575	0.295	0.059	0.222
	RPNN	1.349	0.146	0.026	0.047	0.077	0.520	0.173	0.174	0.044	0.139
	DRPNN	12.232	0.074	0.019	0.043	0.083	0.850	0.077	0.215	0.039	0.104
CDC	MLP	50.630	48.890	55.710	54.700	49.720	51.440	47.220	49.890	48.870	57.840
	FLNN	50.060	47.420	53.640	54.490	52.410	50.750	50.040	51.530	49.120	55.780
	PSNN	48.810	47.890	52.630	53.950	52.040	51.600	48.400	51.610	48.840	56.240
	RPNN	51.590	52.480	55.300	54.800	52.070	51.360	49.170	50.770	48.540	57.180
	DRPNN	51.590	51.790	54.640	54.510	51.480	50.540	46.830	51.520	48.940	57.780
SNR (dB)	MLP	25.870	20.820	35.380	31.660	27.700	26.170	21.500	23.140	29.520	32.140
	FLNN	15.670	39.230	36.190	34.830	34.060	15.420	34.280	36.250	31.180	36.250
	PSNN	16.420	30.220	34.620	31.630	29.640	18.790	31.200	35.640	30.580	34.700
	RPNN	23.690	33.200	35.740	34.450	35.720	22.790	32.650	37.250	31.930	36.620
	DRPNN	10.070	35.720	36.910	34.310	35.270	21.150	34.930	36.540	32.460	37.780

Table 8
Average results on non-stationary signals for the prediction of 1-step ahead.

Performance measures	Neural networks	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
AR (%)	MLP	4.631	−4.357	−11.398	0.925	−2.451	12.297	−0.405	−1.416	8.279	4.610
	FLNN	2.292	−3.754	−3.326	−0.215	2.949	7.197	1.876	−1.004	7.588	5.646
	PSNN	2.128	−3.248	−1.888	0.212	1.659	10.633	2.134	−0.732	7.326	5.136
	RPNN	3.381	−3.293	1.881	0.138	2.379	10.746	1.808	1.607	9.177	6.930
	DRPNN	4.626	−3.254	0.547	0.965	4.345	8.845	2.155	1.123	7.325	7.104
MD	MLP	17.022	−23.948	−22.032	−8.245	−8.519	−5.598	−12.916	11.332	−9.123	−9.119
	FLNN	19.488	−20.585	−12.300	−11.928	−6.017	−5.438	−10.518	10.720	11.821	−8.659
	PSNN	18.895	−21.071	−11.146	−10.350	−5.961	−5.524	−11.098	11.145	10.196	−5.593
	RPNN	17.957	−20.332	−8.801	−9.563	−6.472	−5.921	−10.254	10.772	−8.111	−7.746
	DRPNN	19.999	−21.563	−10.107	−10.060	−7.100	−6.237	−10.280	−9.608	−9.558	−8.339
VOL	MLP	31.700	9.294	5.996	7.839	5.610	8.157	8.870	7.915	8.597	7.380
	FLNN	31.720	9.302	6.016	7.836	5.608	8.185	8.865	7.912	8.600	7.379
	PSNN	31.718	9.302	6.015	7.838	5.610	8.166	8.861	7.908	8.600	7.379
	RPNN	31.699	9.301	6.013	7.837	5.609	8.167	8.867	7.907	8.591	7.372
	DRPNN	31.622	9.296	6.013	7.838	5.607	8.174	8.862	7.907	8.600	7.373
NMSE	MLP	1.485	3.480	0.114	0.878	4.304	2.874	5.080	18.641	0.465	1.551
	FLNN	2.069	0.116	0.111	0.626	0.345	4.469	0.331	0.496	0.210	0.480
	PSNN	1.709	0.132	0.494	0.409	1.405	4.527	4.943	8.502	0.576	0.559
	RPNN	1.224	0.101	13.926	0.164	0.255	0.258	9.638	0.487	0.884	0.515
	DRPNN	2.334	0.091	1.358	0.117	0.395	0.236	0.636	0.515	0.155	0.419
CDC	MLP	51.550	52.570	50.340	53.260	53.390	53.620	51.260	50.930	52.090	54.390
	FLNN	49.710	52.580	52.390	53.570	52.720	52.500	50.830	52.310	51.880	55.210
	PSNN	50.460	52.750	52.750	53.690	53.520	53.410	51.830	52.120	52.820	54.220
	RPNN	51.950	52.670	54.060	53.380	53.940	53.070	50.690	52.840	52.920	54.870
	DRPNN	52.640	52.700	54.100	53.750	53.740	52.970	51.790	52.510	52.900	55.220
SNR (dB)	MLP	18.380	19.740	29.220	20.610	18.510	14.500	16.880	19.550	21.690	27.660
	FLNN	16.730	34.240	29.340	22.400	29.020	12.360	28.920	32.630	25.190	31.150
	PSNN	19.330	34.480	26.800	23.930	24.670	20.880	19.680	29.560	20.730	30.510
	RPNN	19.290	34.430	9.510	28.800	30.310	24.770	21.820	32.760	19.230	30.850
	DRPNN	18.050	35.290	21.930	29.460	28.680	25.190	25.810	32.520	26.430	31.740

Table 9
The average epoch and CPU time usage for the prediction of 1-step ahead stationary signals.

The networks	Measures	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
MLP	Epoch	2543	390	1234	2298	415	1712	1017	1118	2837	638
	CPU time	94	100	80	445	139	330	133	89	103	315
FLNN	Epoch	2473	375	154	2608	924	3000	1647	300	2705	2851
	CPU time	57	27	12	484	382	382	367	394	381	373
PSNN	Epoch	1929	433	156	618	237	304	486	1702	2163	738
	CPU time	131	114	35	256	99	215	89	269	159	81
RPNN	Epoch	2897	158	187	287	172	21	27	119	292	142
	CPU time	94	53	66	111	43	9	11	41	158	67
DRPNN	Epoch	480	137	48	47	132	110	73	258	178	96
	CPU time	54	95	38	28	37	74	67	71	67	38

Table 10
The average epoch and CPU time usage for the prediction of 5-steps ahead stationary signals.

The networks	Measures	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
MLP	Epoch	568	907	1395	744	1365	2050	3000	3000	699	1179
	CPU time	29.38	520.98	99.30	286.80	299.72	288.84	190.23	649.61	190.02	311.08
FLNN	Epoch	2519	645	1104	2870	2851	2983	3000	3000	1489	3000
	CPU time	72.47	63.94	282.52	476.55	376.42	374.89	374.19	4.11	151.91	386.09
PSNN	Epoch	651	312	242	244	893	2543	1294	871	1141	1078
	CPU time	31.34	51.75	33.83	110.03	201.72	539.13	261.89	187.66	247.45	142.98
RPNN	Epoch	425	193	86	155	44	245	24	817	8	298
	CPU time	18.75	42.97	7.30	66.62	6.05	12.66	7.13	63.63	2.31	51.84
DRPNN	Epoch	79	255	30	216	57	54	5	130	9	42
	CPU time	17.58	60.06	35.64	22.06	38.97	80.50	2.92	101.08	4.03	21.77

bles 7 and 8 for the prediction of non-stationary signals show that some of the networks produce negative AR. This indicates that the non-stationary signals exhibit a very strong trend and the values of the daily prices contain a high-frequency component and their relative magnitudes are more difficult to be modeled. Hence, to correctly predict the price from day to day point is a difficult task. As a result, when calculating the AR based on the magnitude size of correct directional change, the resulting profit is likely unpromising and unsatisfactory. On the other hand, results for the prediction of stationary signals in Tables 5 and 6 apparently reveal that all networks produce positive AR. The stationary signals exhibit a linear trend after applying the pre-processing technique and demonstrate a huge reduction in the trends and day to day variations. The signals therefore provide the networks with easier training and help the networks to capture the essence of the background movement. The assumption that pre-processing the signal before using them in the networks will lead to better forecasting seems to hold for all the time series used in this research work. Hence, the result presented in Tables 5–8 supports the theoretical concept of non-stationary signal which is much harder to predict than the stationary signals.

By looking at other financial metrics; the Maximum Drawdown (MD) and Volatility (VOL), results in the tables clearly show that the best values were mostly dominant by DRPNN. This suggests that DRPNN have lower maximum loss and less downside risk compared to other networks when predicting financial signals. It is worth pointing here that for the AR and MD, a bigger value is preferable. Meanwhile for VOL, a lower value is desirable. When measuring the NMSE, CDC, and SNR, it can be noticed that DRPNNs broadly outperformed other networks in most of the cases, with lower NMSE, and higher CDC and SNR.

The average number of epochs reached for the prediction of all data signals during the training of the networks is shown in Tables 9–12. 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. Results for the number of epochs demonstrate that the proposed DRPNN reveal to use least number of epochs to converge during the training of most of the signals. In the relevant signals where DRPNN showed the fastest convergence, the network successfully outperformed other networks on the average by 1.15–55.49 times faster (Table 9), 2.87–600 times faster (Table 10), 1.10–375 times faster (Table 11), and 1.83–104 times faster (Table 12). Results from the all tables show that FLNNs and MLP, in most of the signals, appeared to utilize more epochs to complete the training. In terms of the amount of CPU usage, DRPNN in most cases used the least CPU time when used to learn the stationary signals in comparison to other networks. The network outperformed other neural network models by 1.05–17.37 times faster (Table 9) and 1.07–128.15 time faster (Table 10). Most of the longest CPU times for learning the stationary signals were found in FLNNs. Meanwhile, results from Tables 11 and 12 for non-stationary signals reveal that FLNNs broadly used the least CPU time when compared to other neural networks. The networks appeared to outperform other neural networks with a speed of 1.29–322.68 faster (Table 11) and 1.01–77.28 time faster (Table 12). For non-stationary signals, MLP took the longest time to learn in most of the signals.

For purpose of demonstration, Figs. 2 and 3 show the best prediction, histogram of signal error, and the learning curve from the prediction of US/EU and UK/EU using the proposed DRPNN. Plots in Fig. 2 depict the results from the learning and prediction of stationary signal, while plots in Fig. 3 show that of non-stationary signals. Notice that the plots for the best forecast (column 'a') were taken from the first 100 data points from the unseen part of the data. For stationary signal in Fig. 2, the plots for the best forecast show that the original and predicted signals are pretty close to each other. This may indicate that DRPNN is likely capable at mapping the underlying movements in stationary financial markets. Meanwhile, the histograms of the prediction errors on stationary signal signify that all the prediction errors are near to zero and follow

Table 11
The average epoch and CPU time usage for the prediction of 1-step ahead non-stationary signals.

The networks	Measures	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
MLP	Epoch	3000	95	2724	2861	2067	1443	2353	905	3000	2697
	CPU time	161.92	33.20	563.73	869.25	520.59	611.55	728.47	347.19	690.28	713.83
FLNN	Epoch	3000	19	186	89	24	3000	27	20	3000	125
	CPU time	104.27	4.42	29.02	4.17	4.05	395.36	3.45	2.72	387.39	2.75
PSNN	Epoch	2493	94	98	38	31	2759	198	21	2564	562
	CPU time	153.05	5.69	68.13	13.39	4.70	642.53	7.64	4.00	597.66	658.08
RPNN	Epoch	61	2173	494	58	1377	26	249	60	1899	1325
	CPU time	4.61	57.38	153.19	10.92	66.36	8.98	29.39	7.98	671.97	448.80
DRPNN	Epoch	8	2071	1789	46	22	102	947	19	730	2656
	CPU time	4.97	603.89	289.59	17.53	12.02	57.77	193.14	12.80	609.37	887.36

Table 12
The average epoch and CPU time usage for the prediction of 5-steps ahead non-stationary signals.

The networks	Measures	IBM	CMESP	CBT-10	CBT-30	UK/EU	UK/US	US/EU	JP/EU	JP/US	JP/UK
MLP	Epoch	3000	104	2524	2520	1040	945	1554	937	2301	2490
	CPU time	156.64	30.27	591.23	253.89	180.52	245.11	93.42	230.30	674.69	667.44
FLNN	Epoch	3000	19	116	21	42	3000	28	19	2610	2006
	CPU time	95.44	3.58	24.53	4.13	2.86	372.36	2.94	2.98	389.36	389.33
PSNN	Epoch	3000	34	40	479	544	739	241	47	755	1374
	CPU time	133.98	5.77	24.75	9.55	8.16	56.23	15.53	18.81	213.92	337.72
RPNN	Epoch	128	51	72	293	91	140	10	1149	124	1496
	CPU time	5.37	10.45	25.23	37.61	15.17	53.28	3.78	21.61	54.50	356.57
DRPNN	Epoch	70	20	84	35	10	189	16	24	415	176
	CPU time	28.55	13.41	61.38	28.58	6.38	27.86	11.16	15.94	42.42	125.86

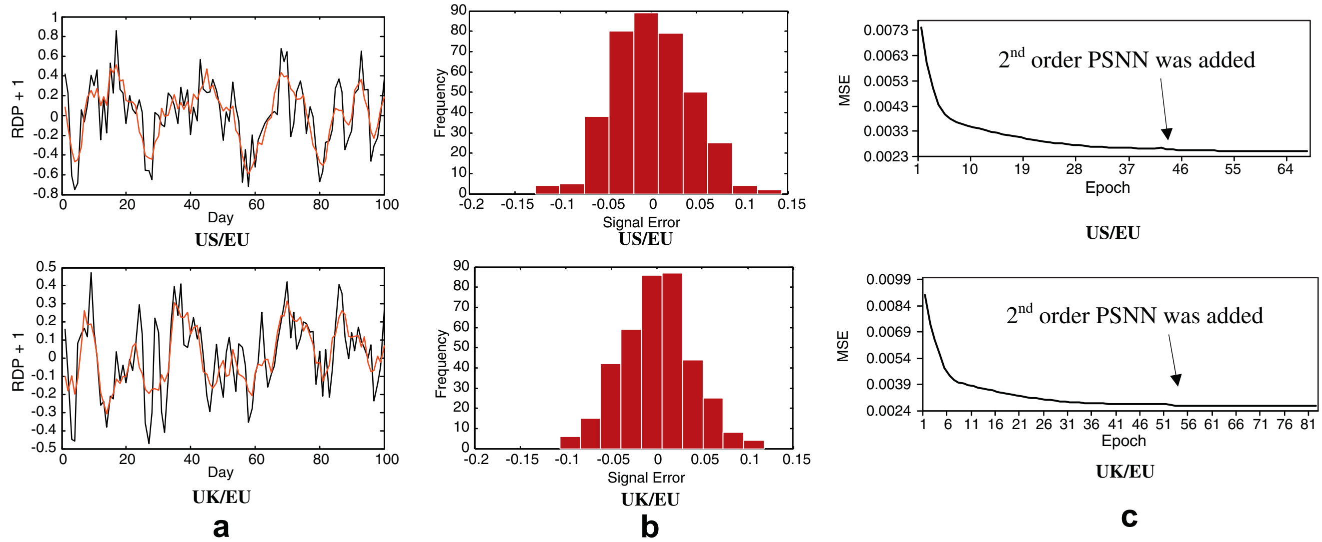

Fig. 2. (a) Forecast made by DRPNN on stationary signals; —— – original signal, —— – predicted signal. (b) Histogram of error signal by DRPNN for the prediction of stationary signals. (c) Learning curve of DRPNN when learning the stationary signals.

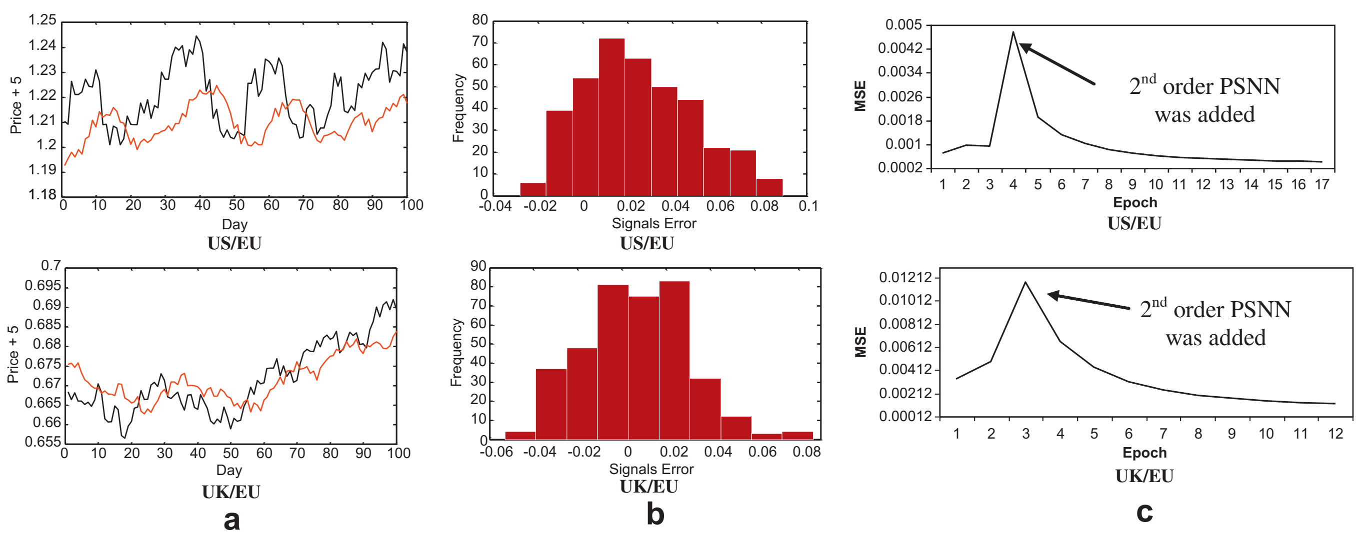

Fig. 3. (a) Forecast made by DRPNN on non-stationary signals; —— – original signal, —— – predicted signal. (b) Histogram of error signal by DRPNN for the prediction of non-stationary signals. (c) Learning curve of DRPNN when learning the non-stationary signals.

closely to the normal distribution. When learning the stationary signals (refer to Fig. 2, column 'c'), DRPNN show a rapid learning, considering the Mean Squared Error (MSE) curves end up at less than 90 epochs.

Meanwhile, by looking at the forecast plot by DRPNN when used to predict the non-stationary signals (refer to Fig. 3, column 'a'), it can be noticed that at some points, the original and predicted signals are a bit distant from each other. This can suggest that the non-stationary time series signals are harder to predict, as compared to the stationary signals. In the case of the prediction errors, histograms in column 'b' demonstrate that most of the signals error are not approaching zero, especially for the prediction of US/EU even though the histograms show a normal distribution. The learning curves plotted in Fig. 3, column 'c', apparently showed the ability of DRPNN to converge extremely fast when learning the non-stationary signals. For both signals, the network exhibits a remarkably stable learning and the MSE continuously decreased every time a Pi-Sigma unit of a higher degree is added to the networks. Notice that each spike in the curve was resulted from the addition of Pi-Sigma unit in the networks.

8. Discussions

Simulation results show that all the neural networks used in this work were potentially profitable, however, in most cases the proposed DRPNN is by far the most beneficial as money-making predictor. The networks 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. 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 DRPNN is guaranteed to exhibit a unique equilibrium state, as the stability convergence was applied during their training to ensure that the network always posses a stable condition. The network can robustly process the underlying dynamics of a non-stationary environment with a vast speed in convergence time. A noteworthy advantage of DRPNN is the fact that there is no requirement to select the order of the networks as in PSNN and FLNN, or the number of hidden units as in MLP.

9. Conclusion and future work

This research work underlines an important contribution of a new developed Dynamic Ridge Polynomial Neural Network; namely its elegant ability to approximate nonlinear financial time series. The network has shown its advantages in forecasting both stationary and non-stationary signals. A considerable profitable value does exist in DRPNN when compared to other networks and the network demonstrated a vast speed in convergence time.

It is anticipated that DRPNN can be used as an alternative tool 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 real world problems. It should be emphasized that DRPNN is not without problem. The main intricacy when using DRPNN is to find the suitable parameters for successively adding a higher degree of Pi-Sigma unit in the networks. With respect to this deficiency, it might be worthwhile to consider how Genetic Algorithm can be used to automatically generating suitable parameters for the network.

Acknowledgment

The work of R. Ghazali is supported by Universiti Tun Hussein Onn Malaysia (UTHM).

Appendix A

A network satisfying

$$\sum_{i=1}\sum_{j=1} w_{ij}^2 < \frac{1}{\max(f')^2} \tag{A.1}$$

exhibits no other behavior except going to unique equilibrium for a given input (Atiya, 1988).

Let $y_1(t+1)$ and $y_2(t+1)$ be two outputs for the DRPNN.

$$y_1(t+1) = f\left(\sum_{k=1}^{A}\prod_{L=1}^{k} h_{1L}(t+1)\right) \tag{A.2}$$

where f is a nonlinear transfer function.

$$y_2(t+1) = f\left(\sum_{k=1}^{A}\prod_{L=1}^{k} h_{2L}(t+1)\right) \tag{A.3}$$

$$\begin{aligned} h_{1L}(t+1) &= \sum_{i=1}^{M} W_{Li}X_i + W_{L(M+1)} + W_{L(M+2)}y_1(t) \\ &= \alpha_L + \beta_L y_1(t) \end{aligned} \tag{A.4}$$

with

$$\alpha_L = \sum_{i=1}^{M} W_{Li}X_i + W_{L(M+1)} \tag{A.5}$$

and

$$\beta_L = W_{L(M+2)} \tag{A.6}$$

while

$$\begin{aligned} h_{2L}(t+1) &= \sum_{i=1}^{M} W_{Li}X_i + W_{L(M+1)} + W_{L(M+2)}y_2(t) \\ &= \alpha_L + \beta_L y_2(t) \end{aligned} \tag{A.7}$$

The aim is to get J approaching '0', which means that the two outputs of a given input are close. Let $J(t+1)$ be:

$$J(t+1) = \|y_1(t+1) - y_2(t+1)\| \tag{A.8}$$

where $\| \|$ is the norm. Based on Mean Value Theorem, which states that for a function $f(x)$ which is continuous on the closed interval $[a, b]$ and differentiable on the open interval (a,b), there exists a value c on the interval (a,b) such that

$$f'(c) = \frac{f(b) - f(a)}{b - a} \tag{A.9}$$

where f' is the derivation of the function. Hence

$$f(b) - f(a) = f'(c)\cdot(b - a) \tag{A.10}$$

and

$$\|f(b) - f(a)\| = \|f'(c)\|\cdot\|b - a\| \tag{A.11}$$

which leads to

$$\|f(b) - f(a)\| \leqslant \max\|f'(c)\|\cdot\|b - a\| \tag{A.12}$$

substituting Eqs. (A.2) and (A.3) into Eq. (A.8), results into

$$J(t+1) = \left\| f\left(\sum_{k=1}^{A}\prod_{L=1}^{k} h_{1L}(t+1)\right) - f\left(\sum_{k=1}^{A}\prod_{L=1}^{k} h_{2L}(t+1)\right)\right\| \tag{A.13}$$

using Mean Value Theorem, leads to

$$\begin{aligned} &\left\| f\left(\sum_{k=1}^{A}\prod_{L=1}^{k} h_{1L}(t+1)\right) - f\left(\sum_{k=1}^{A}\prod_{L=1}^{k} h_{2L}(t+1)\right)\right\| \\ &\quad \leqslant \max|f'|\cdot\left\|\sum_{k=1}^{A}\prod_{L=1}^{k} h_{1L}(t+1) - \sum_{k=1}^{A}\prod_{L=1}^{k} h_{2L}(t+1)\right\| \end{aligned} \tag{A.14}$$

therefore, from Eq. (A.13), Eq. (A.14) becomes

$$J(t+1) \leqslant \max|f'|\cdot\left\|\sum_{k=1}^{A}\prod_{L=1}^{k} h_{1L}(t+1) - \sum_{k=1}^{A}\prod_{L=1}^{k} h_{2L}(t+1)\right\| \tag{A.15}$$

from Eqs. (A.2) and (A.4), let $g(y)$ be

$$g(y) = \sum_{k=1}^{A}\prod_{L=1}^{k}(\alpha_L + \beta_L y) = \sum_{k=1}^{A}\prod_{L=1}^{k} h_L(t+1) \tag{A.16}$$

hence

$$\begin{aligned} &\left\|\sum_{k=1}^{A}\prod_{L=1}^{k} h_{1L}(t+1) - \sum_{k=1}^{A}\prod_{L=1}^{k} h_{2L}(t+1)\right\| \\ &\quad = \|g(y_1(t)) - g(y_2(t))\| \end{aligned} \tag{A.17}$$

using the Mean Value Theorem again, leads to:

$$\|g(y_1(t)) - g(y_2(t))\| \leqslant \max|g'|\cdot\|y_1(t) - y_2(t)\| \tag{A.18}$$

hence, from Eqs. (A.15), (A.17) and (A.18), results into

$$J(t+1) \leqslant (\max|f'|)\cdot(\max|g'|)\cdot\|y_1(t) - y_2(t)\| \tag{A.19}$$

let δ be

$$\delta = (\max|f'|)\cdot(\max|g'|) \tag{A.20}$$

then

$$J(t+1) \leqslant \delta \|y_1(t) - y_2(t)\| \tag{A.21}$$

from Eq. (A.8), Eq. (A.21) becomes

$$J(t+1) \leqslant \delta J(t) \tag{A.22}$$

The aim is to get both $J(t+1)$ and $J(t)$ approaching very close to zero, and for large (t), and for any value of (t). To achieve this, δ has to be very small value, which is less than 1. Hence, from Eq. (A.20), when δ is <1, leads into:

$$(\max|f'|) \cdot (\max|g'|) < 1 \tag{A.23}$$

from Eq. (A.16), $g(y)$ will be

$$g(y) = \sum_{k=1}^{A} \prod_{L=1}^{k} (\alpha_L + \beta_L y) \tag{A.24}$$

let $P(y)$ be

$$P(y) = \prod_{L=1}^{k} (\alpha_L + \beta_L y) \tag{A.25}$$

then

$$\sum_{k=1}^{A} P(y) = \sum_{k=1}^{A} \prod_{L=1}^{k} (\alpha_L + \beta_L y) \tag{A.26}$$

therefore

$$g(y) = \sum_{k=1}^{A} P(y) \tag{A.27}$$

and

$$(g(y))' = \sum_{k=1}^{A} P'(y) \tag{A.28}$$

from Eq. (A.25)

$$\ln(P(y)) = \sum_{L=1}^{k} \ln(\alpha_L + \beta_L y) \tag{A.29}$$

and

$$\frac{P'(y)}{P(y)} = \sum_{L=1}^{k} \frac{\beta_L}{(\alpha_L + \beta_L y)} \tag{A.30}$$

hence

$$P'(y) = P(y) \sum_{L=1}^{k} \frac{\beta_L}{(\alpha_L + \beta_L y)} \tag{A.31}$$

substitute Eq. (A.25) into Eq. (A.31), having

$$P'(y) = \prod_{L=1}^{k} (\alpha_L + \beta_L y) \cdot \sum_{L=1}^{k} \frac{\beta_L}{(\alpha_L + \beta_L y)} \tag{A.32}$$

then

$$P'(y) = \sum_{L=1}^{k} \beta_L \cdot \sum_{L=1}^{k} \frac{1}{(\alpha_L + \beta_L y)} \cdot \prod_{L=1}^{k} (\alpha_L + \beta_L y) \tag{A.33}$$

$$P'(y) = \sum_{L=1}^{k} \beta_L \cdot \sum_{L=1}^{k} \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \tag{A.34}$$

$$P'(y) = \sum_{L=1}^{k} \beta_L \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \tag{A.35}$$

substituting Eq. (A.35) into Eq. (A.28), results into

$$(g(y))' = \sum_{k=1}^{A} \sum_{L=1}^{k} \beta_L \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \tag{A.36}$$

and

$$|(g(y))'| = \left| \sum_{k=1}^{A} \sum_{L=1}^{k} \beta_L \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \right| \tag{A.37}$$

therefore

$$\left| \sum_{k=1}^{A} \sum_{L=1}^{k} \beta_L \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \right| = \left| \sum_{k=1}^{A} \sum_{L=1}^{k} \beta_L \right| \cdot \left| \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \right| \tag{A.38}$$

hence

$$\left| \sum_{k=1}^{A} \sum_{L=1}^{k} \beta_L \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \right| \leqslant \sum_{k=1}^{A} \sum_{L=1}^{k} |\beta_L| \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (|\alpha_S| + |\beta_S y|) \tag{A.39}$$

note that from Eq. (A.5)

$$\alpha_L = \sum_{i=1}^{M} W_{Li} X_i + W_{L(M+1)} \tag{A.40}$$

therefore

$$|\alpha_L| = \sum_{m=1}^{M+1} |W_{Lm}| \tag{A.41}$$

note that from Eqs. (A.6) and (A.39) results

$$\left| \sum_{k=1}^{A} \sum_{L=1}^{k} \beta_L \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \right| \leqslant \sum_{k=1}^{A} \sum_{L=1}^{k} |W_{L(M+2)}| \cdot \prod_{\substack{S=1\\S\neq L}}^{k} \left(\sum_{m=1}^{M+1} |W_{Sm}| + |W_{S(M+2)}| \right) \tag{A.42}$$

hence

$$\left| \sum_{k=1}^{A} \sum_{L=1}^{k} \beta_L \cdot \prod_{\substack{S=1\\S\neq L}}^{k} (\alpha_S + \beta_S y) \right| \leqslant \sum_{k=1}^{A} \sum_{L=1}^{k} |W_{L(M+2)}| \cdot \prod_{\substack{S=1\\S\neq L}}^{k} \sum_{m=1}^{M+2} |W_{Sm}| \tag{A.43}$$

therefore, from Eqs. (A.37) and (A.43) results into

$$|(g(y))'| = \sum_{k=1}^{A} \sum_{L=1}^{k} |W_{L(M+2)}| \cdot \prod_{\substack{S=1\\S\neq L}}^{k} \sum_{m=1}^{M+2} |W_{Sm}| \tag{A.44}$$

substituting Eq. (A.44) into Eq. (A.23), we get

$$(\max|f'|) \cdot \left(\max\left(\sum_{k=1}^{A} \sum_{L=1}^{k} |W_{L(M+2)}| \cdot \prod_{\substack{S=1\\S\neq L}}^{k} \sum_{m=1}^{M+2} |W_{Sm}| \right) \right) < 1 \tag{A.45}$$

Hence, the condition for DRPNN to converge is described by

$$\left(\max\left(\sum_{k=1}^{A} \sum_{L=1}^{k} |W_{L(M+2)}| \cdot \prod_{\substack{S=1\\S\neq L}}^{k} \sum_{m=1}^{M+2} |W_{Sm}| \right) \right) < \frac{1}{(\max|f'|)} \tag{A.46}$$

References

Artyomov, E., & Pecht, O. Y. (2004). Modified high-order neural network for pattern recognition. *Pattern Recognition Letters*.

Atiya, A. F. (1988). Learning on a general network. In *Proceedings of the IEEE conference on neural information processing systems*.

Beale, R., & Jackson, T. (1990). *Neural computing: An introduction*. Bristol: Hilger.

Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks, 14*(6), 1506–1518.

Cass, R., & Radl, B. (1996). Adaptive process optimization using Functional-Link Networks and Evolutionary Algorithm. *Control Engineering Practice, 4*(11), 1579–1584.

Chen, A. S., & Leung, M. T. (2005). Performance evaluation of neural network architectures: The case of predicting foreign exchange correlations. *Journal of Forecasting, 24*, 403–420.

Cheng, W., Wagner, L., & Lin, C. H. (1996). *Forecasting the 30-year US Treasury Bond with a system of neural networks*. USA: Finance & Technology Publishing, Inc..

Dunis, C. L., & Williams, M. (2002). Modeling and trading the UER/USD exchange rate: Do neural network models perform better? *In Derivatives Use, Trading and Regulation, 3*(8), 211–239.

Ghosh, J., & Shin, Y. (1992). Efficient higher-order neural networks for function approximation and classification. *International Journal of Neural Systems, 3*(4), 323–350.

Giles, C. L., & Maxwell, T. (1987). Learning, invariance and generalization in high-order neural networks. *Applied Optics, 26*(23), 4972–4978.

Haykin, S. (1999). Neural networks. *A comprehensive foundation* (2nd ed.). New Jersey: Prentice-Hall, Inc..

Hussain, A. J., & Liatsis, P. (2002). Recurrent pi-sigma networks for DPCM image coding. *Neurocomputing, 55*, 363–382.

Hyndman, R. J. (n.d.). Time Series Data Library. Downloaded from: <http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/>. Original source from: McCleary & Hay, Applied Time Series Analysis for the Social Sciences, 1980, Sage Publications.

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

Leung, M. T., Chen, A. S., & Daouk, H. (2000). Forecasting exchange rates using general regression neural networks. *Computers and Operations Research, 27*, 1093–1110.

Mirea, L., & Marcu, T. (2002). System identification using Functional-Link Neural Networks with dynamic structure. In *15th triennial world congress, Barcelona, Spain*.

Patra, J. C., & Bos, A. V. D. (2000). Modeling of an intelligent pressure sensor using functional link artificial neural networks. *ISA Transactions, 39*, 15–27.

Shin, Y., & Ghosh, J. (1991). The pi-sigma networks: An efficient higher-order neural network for pattern classification and function approximation. In *Proceedings of international joint conference on neural networks, Seattle, Washington* (Vol. 1, pp. 13-018).

Shin, Y., & Ghosh, J. (1995). Ridge polynomial networks. *IEEE Transactions on Neural Networks, 6*(3), 610–622.

Tawfik, H., & Liatsis, P. (1997). Prediction of non-linear time-series using higher-order neural networks. In *Proceeding IWSSIP'97 conference, Poznan, Poland*.

Thomason, M. (1999). The practitioner method and tools. *Journal of Computational Intelligence in Finance, 7*(3), 36–45.

Williams, R. J., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1*, 270–280.

Yao, J., & Tan, C. L. (2000). A case study on neural networks to perform technical forecasting of forex. *Neurocomputing, 34*, 79–98.

Yumlu, S., Gurgen, F. S., & Okay, N. (2005). A comparison of global, recurrent and smoothed-piecewise neural models for Istanbul stock exchange (ISE) prediction. *Pattern Recognition Letters, 26*, 2093–2103.