DE GRUYTER OPEN

JAISCR, 2018, Vol. 8, No. 2, pp. 121 – 132
10.1515/jaiscr-2018-0009

AN ARMA TYPE PI-SIGMA ARTIFICIAL NEURAL NETWORK FOR NONLINEAR TIME SERIES FORECASTING

Esra Akdeniz[1], Erol Egrioglu[2], Eren Bas[2], Ufuk Yolcu[3]

[1]*Department of Biostatistics, Medical Faculty, Marmara University, Istanbul, Turkey*

[2]*Department of Statistics, Faculty of Arts and Science, Forecast Research Laboratory, Giresun University, Giresun, 28100, Turkey*

[3]*Department of Econometrics, Faculty of Economic and Administrative Sciences, Forecast Research Laboratory, Giresun University, Giresun, 28100, Turkey*

Submitted: 3rd March 2017; accepted: 22th March 2017

Abstract

Real-life time series have complex and non-linear structures. Artificial Neural Networks have been frequently used in the literature to analyze non-linear time series. High order artificial neural networks, in view of other artificial neural network types, are more adaptable to the data because of their expandable model order. In this paper, a new recurrent architecture for Pi-Sigma artificial neural networks is proposed. A learning algorithm based on particle swarm optimization is also used as a tool for the training of the proposed neural network. The proposed new high order artificial neural network is applied to three real life time series data and also a simulation study is performed for Istanbul Stock Exchange data set.

Keywords: High order artificial neural networks, pi-sigma neural network, forecasting, recurrent neural network, Particle Swarm Optimization.

1 Introduction

A great number of methods that have different structure and features are utilized for time series forecasting problem. The used methods can be classified into two types as model based and data based. Artificial neural networks (ANNs) are data based methods with their complicated structure caused by the hidden layers and also they are commonly used in time series forecasting. In ANN approaches, the models undergo change according to data structure by means of altering the number of hidden layer(s) or unit(s) and so they can ensure very high compliance with the data. The most common used ANN type, in time series forecasting literature, is multi-layer perceptron artificial neural network (MLP-ANN) that it uses additive aggregation function in its neurons.

Another widely used ANN type is multiplicative neuron model artificial neural network (MNM-ANN) introduced by Yadav et al. [1]. MNM-ANN is composed of one neuron only and it produces more successful results than MLP-ANN with fewer parameters under favor of multiplicative aggregation function. There are different kinds of MNM-ANN in the literature. While Egrioglu et al. [2] proposed recurrent multiplicative neuron model artificial neural network (RMNM-ANN), Gundogdu et al. [3] presented a new MNM-ANN based on gaussian activation function. Another ANN type is higher order neural networks based on additive and multiplicative aggregation function. Higher order neural networks such as pi sigma artificial neural network (PS-ANN) (Rumelhart DE and Mcclelland, [4]), higher order processing unit neural network (Giles and Maxwell, [5]), product unit neural network (PUNN) (Durbin and Rumelhart, [6]), have been proposed in the literature. Higher order neural networks have much better ability of learning with less weights and biases than MLP's.

PS-ANN proposed by Shin and Ghosh [7] is also a kind of higher order neural network. Ghosh and Shin [7] argued that PS-ANN requires less memory (weights and nodes), and at least two orders of magnitude less number of computations when compared to MLP-ANN for similar performance level, and over a broad class of problems [8]. [9, 10, 11] used PS-ANN for time series forecasting problem. For financial time series prediction an application of ridge polynomial network formed by adding different degrees of Pi–Sigma neural networks has been suggested by Ghazali et al. [9] which is able to find an appropriate input output mapping of various chaotic financial time series data with a good performance in learning speed and generalization capability. [12] realized that hybrid genetic algorithm can search out the global optimum which is faster than genetic algorithm and their proposed hybrid genetic algorithm trained Pi-Sigma network was used to resolve the function optimization problem. On the training of PS-ANN, while Nayak et al. [13] put forward an algorithm in which particle swarm optimization (PSO) and genetic algorithm (GA) are utilized together, Nayak et al. [14] proposed a hybrid learning algorithm based on PSO and gradient descent. [15] suggested a memory based SPNN. Moreover, there are some recurrent Pi-Sigma neural networks (R-PS-ANN) in the literature. Ghazali et al. [8] introduced an R-PS-ANN that the output of NN is connected to input layer as one-step-lagged and forms a new input. [16] and Nayak et al. [17] presented Jordan type R-PS-ANN.

In this study, PS-ANN proposed by Shin and Ghosh [7] is transformed into a recurrent structure by making some modifications in neural network architecture. The proposed new ANN is called as autoregressive moving average type PS-ANN (ARMATPS-ANN). The proposed ARMATPS-ANN has an architecture structure in which the error terms of the NN are fed-back to input layer as lagged variables. Training of the proposed ANN is carried out by PSO. The second and third Sections of the study give a summary info about PS-ANN and PSO, respectively. In the Section 4, the proposed ARMATPS-ANN is introduced and training principle of ANN realized by PSO is given in the form of an algorithm. The obtained implementation results based on analysis of real-life and simulated time series data sets are summarized in the Section 5. In the last Section, obtained findings are emphasized and discussed.

2 Pi-Sigma Artificial Neural Network

PS-ANN was firstly proposed by Shin and Ghosh [7]. In a PS-ANN structure, the multiplications of different linear combinations are the output of the network. The numbers of linear combinations are the degree of the PS-ANN. The increasing of the degree of PS-ANN causes that the function describes the relation between the input and output depends more parameters and more complex network structure. Although this situation makes certain of more forecasting results, more parameters in the network structure causes overfitting problem and as a result of this, it is needed to more computation time for the learning algorithm. For this reason, it has to be careful of learning of the network without memorization by using cross validation methods. The architecture of PS-ANN with kth order and nthinput is given in Figure 1.

The linear combinations of inputs are obtained via, $w_{ij}(\ i=1,2,\ldots,N\ ,\ j=1,2,\ldots,K)$ weights, and biases $\theta_j\ (j=1,2,\ldots,K)$. w_{ij} shows the weight from ith input to jthhidden layer. $\theta_j$shows the biases value for jth hidden layer. Linear combinations as much as the number of hidden layers are formed the outputs of the hidden layers through the linear activation function. h_j shows the output of the hidden layer and formulate as follows

$$h_j = f_1\left(\sum_{i=1}^{N} w_{ij}x_i + \theta_j\right)\ ,\quad j=1,2,\ldots,K. \qquad (1)$$

In this Equation (1), $f_1(x) = x$ shows the linear activation function. The output of the network is calculated in Equation 2.

$$\hat{y} = f_2\left(\prod_{j=1}^{K} h_j\right) = \frac{1}{1+\exp(-\prod_{j=1}^{K} h_j)}. \quad (2)$$

In this Equation, $f_2(x) = \frac{1}{1+\exp(-x)}$ is logistic activation function.

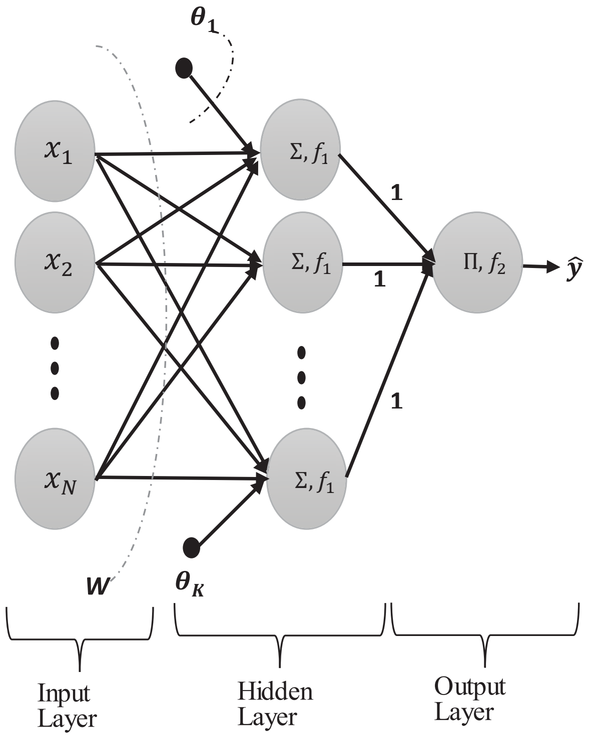


Figure 1. The architecture of PS-ANN

3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was firstly proposed by Kennedy and Eberhart [18]. PSO is a stochastic optimization algorithm based on population. It is adapted solving of many problems because it does not need differentiation of the objective function. There are many modifications of PSO in the literature. Aladag et al. [19] proposed modified PSO (MPSO) for the training of multiplicative seasonal artificial seasonal artificial neural network. How the MPSO algorithm is implemented is given below.

Algorithm 1. MPSO algorithm

Step 1. Each k^{th} $(k = 1,2,\ldots,pn)$ particles' positions are randomly determined and kept in a vector X_k given as follows

$$X_k = \{x_{k,1}, x_{k,2}, \ldots, x_{k,d}\},\ k = 1,2,\ldots,pn, \quad (3)$$

where $x_{k,i}$ (i=1,2,...,d) represents i th position of k th particle. pn and d represents the number of particles and positions, respectively.

Step 2. Velocities are randomly determined and kept in a vector V_k given below.

$$V_k = \{v_{k,1}, \ldots, v_{k,d}\},\ k = 1,2,\ldots,pn. \quad (4)$$

Step 3. According to the evaluation function, *Pbest* and *Gbest* particles given in (5) and (6), respectively, are determined.

$$Pbest_k = \{p_{k,1}, \ldots, p_{k,d}\},\ k = 1,2,\ldots,pn, \quad (5)$$

$$Gbest = \{P_{g,1}, \ldots, P_{g,d}\}, \quad (6)$$

where $Pbest_k$ is a vector stores the positions corresponding to the k^{th} particle's best individual performance, and *Gbest* represents the best particle, which has the best evaluation function value, found so far.

Step 4. Let c_1 and c_2 represents cognitive and social coefficients, respectively, and w is the inertia parameter. Let (c_{1i}, c_{1f}), (c_{2i}, c_{2f}), and (w_1, w_2) be the intervals which includes possible values for c_1, c_2 and w, respectively. At each iteration, these parameters are calculated by using the formulas given in (7), (8) and (9).

$$c_1(t) = (c_{1f} - c_{1i})\frac{t}{\max t} + c_{1i}, \quad (7)$$

$$c_2(t) = (c_{2f} - c_{2i})\frac{t}{\max t} + c_{2i}, \quad (8)$$

$$w(t) = (w_2 - w_1)\frac{maxt - t}{maxt} + w_1, \quad (9)$$

where *maxt* and t represent maximum iteration number and current iteration number, respectively.

Step 5. Values of velocities and positions are updated by using the formulas given in (10) and (11), respectively.

$$\begin{aligned} v_{i,d}^{t+1} = &[w \times v_{i,d}^{t} + \\ &+ c_1 \times rand_1 \times (p_{i,d} - x_{i,d}) + \\ &+ c_2 \times rand_2 \times (p_{g,d} - x_{i,d})], \end{aligned} \quad (10)$$

$$x_{i,d}^{t+1} = x_{i,d} + v_{i,d}^{t+1}, \tag{11}$$

where $rand_1$ and $rand_2$ are random values from the interval [0, 1].

Step 6. Steps 3 to 5 are repeated until a predetermined maximum iteration number (*maxt*) is reached.

4 Autoregressive Moving Average Type Pi Sigma Neural Network

Recurrent artificial neural networks produces more successful forecasting results compared with feed forward artificial neural networks under favor of their recurrent mechanism. The error calculated as the difference between target and output or the output produced by the network in recurrent artificial neural networks returns as an input to the hidden or input layer of the network. There are different types of recurrent ANNs in the literature. In Elman type recurrent ANN, the output of the network is given as input to the context layer which employs as a hidden layer. In Jordan type recurrent ANN, the output of the network feedbacks to the input layer as an input. The output produced by network and the error value are different from each other and as a superiority of error value, it considers the target value. In Egrioglu et al. [2], the lagged variables obtained from the error of the network feed backed to the input layer and from this aspect it is a recurrent artificial network like in linear ARMA models. Therefore recurrent type proposed by Egrioglu et al. [2] has similar structure with ARMA model. In the literature, there are several recurrent architectures for PS-ANN. [8, 16, 17] proposed Jordan Type recurrent PS-ANN. In these methods, the output of recurrent PS-ANN is linked to the input layer as one step lagged and shown as a new input. In this paper, an architecture that is a feedback of the error of the network used in Egrioglu et al. [2] is applied to PS-ANN and a new ARMATPS-ANN is proposed. The proposed architecture is given in Figure 2. When Table 2 is examined, it is seen that the lagged variables of $e_t = y_t - \hat{y}_t$ series is used as the inputs of artificial neural networks. The lagged variables are obtained via "B" back shift operator. The lagged error variables are linked to the hidden layers with *We* weights. Thus, the artificial neural network has a model as non-linear ARMA model. The output of ARMATPS-ANN, *g* shows a non-linear function, is a function of the weights, lagged variables of error series and time series.

$$\begin{aligned}\hat{y}_t =& g(y_{t-1},\dots,y_{t-p},e_{t-1},..,e_{t-q};\\ & Wy, We, \theta_1,..,\theta_K)\end{aligned} \tag{12}$$

$e_{t-1},..,e_{t-q}$ lagged variables are taken as zero for the first q observations when they cannot calculated. Some of them can be calculated numerically after second observation. $Wy, We, \theta_1,..,\theta_K$ weights are totally $(p+q)K$ and optimal weight values are obtained with learning algorithm.

How the output is calculated for a learning sample in ARMATPS-ANN network is given in steps in Algorithm 2.

Algorithm 2. Calculation of An output of ARMATPS-ANN for a learning sample

Step 1. Let $\mathrm{wy_{ij}}(\mathrm{i} = 1,2,\dots,\mathrm{p}\ ,\ \mathrm{j} = 1,2,\dots,\mathrm{K})$ gives the weights in the linear combinations for the lagged variables of the time series, $\mathrm{we_{ij}}(\mathrm{i} = 1,2,\dots,\mathrm{q}\ ,\ \mathrm{j} = 1,2,\dots,\mathrm{K})$ gives the weights in linear combinations for the lagged variables of the error series and θ_j $(\mathrm{j} = 1,2,\dots,\mathrm{K})$ be the values of the biases. $\mathrm{wy_{ij}}$ and $\mathrm{we_{ij}}$ shows the weights between the input i^{th} and j^{th} hidden layer unit, θ_j shows the biases for j^{th} hidden layer unit. The linear combination as the number of hidden layer units passes through the linear activation function to form the outputs of the hidden layers. $\mathrm{h_j}$ shows the output of the hidden layer unit and it is calculated by the following formula.

$$\begin{aligned}&h_j = f_1\left(\sum_{i=1}^{p} wy_{ij}y_{t-i} + \sum_{m=1}^{q} we_{mj}e_{t-m} + \theta_j\right),\\ &j = 1,2,\dots,K.\end{aligned} \tag{13}$$

In this Equation, $f_1(x) = x$ shows linear activation function. In formula (13), the values of the error series are calculated after the corresponding estimation is obtained, and in the case of not being calculated, the error value is taken as zero.

Step 2. The output of the network is $f_2(x) = \frac{1}{1+\exp(-x)}$ logistic activation function and it is calculated as follows.

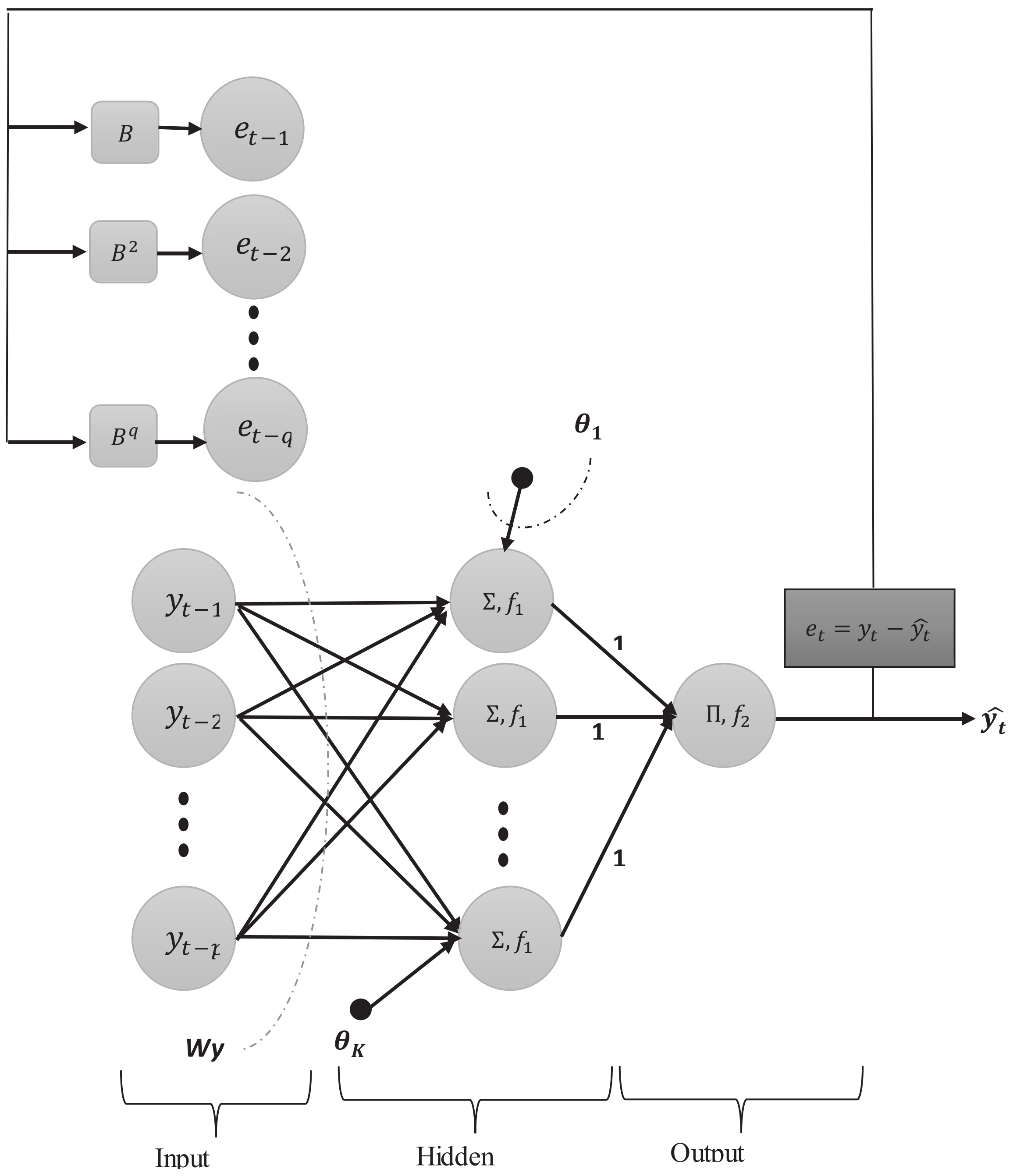


Figure 2. The architecture of ARMATPS-ANN

$$\hat{y}_t = f_2\left(\prod_{j=1}^{K} h_j\right) = \frac{1}{1+\exp(-\prod_{j=1}^{K} h_j)}. \quad (14)$$

The output of the network can be calculated for learning or testing samples by using Algorithm 2. In order to calculation, there is need to weight and biases values. The optimal values of weights and biases are obtained with Algorithm 3 given by MPSO algorithm.

Algorithm 3. The training of ARMATPS-ANN with MPSO

Step 1. The parameters of MPSO are determined. These parameters are pn, c_{1i}, c_{1f}, c_{2i}, c_{2f}, w_1, w_2 and vm, given below.

Let c_1 and c_2 represents cognitive and social coefficients, respectively, and w is the inertia parameter. Let (c_{1i}, c_{1f}), (c_{2i}, c_{2f}), and (w_1, w_2) be the intervals which includes possible values for c_1, c_2 and w, respectively.

Step 2. Initial values of positions and velocities are determined. The initial positions and velocities of each particle in a swarm are randomly generated from uniform distribution (0,1) and (*-vm,vm*), respectively.

The positions of a particle are composed of the values of weights and biases of ARMATPS-ANN and it is shown in Table 1. A particle has totally $(p+q)K+K$ positions.

Table 1. Presentation of a particle

Positions	***Weights***
1	wy_{11}
...	...
pK	wy_{pK}
pK+1	we_{11}
...	...
(p+q)K	we_{qK}
(p+q)K +1	θ_1
...	...
(p+q)K +K	θ_K

Step 3. Evaluation function values are computed.

Evaluation function values for each particle are calculated by using Algorithm 2 for these particles. Root mean square error (RMSE) given in (15) is used as evaluation function.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \quad (15)$$

where y_i, $\hat{y}_i$, n represent real values, forecasts, and length of the test set, respectively.

Step 4. $Pbest_k$ (k = 1,2, ..., pn) and *Gbest* are determined due to evaluation function values calculated. According to the evaluation function, *Pbest* and *Gbest* particles given in (5) and (6), respectively, are determined.

Step 5. The parameters are updated.

The updated values of cognitive coefficient c_1, social coefficient c_2, and inertia parameter w are calculated using the formulas given in (7), (8), and (9).

Step 6. New values of positions and velocities are calculated.

New values of positions and velocities for each particle are computed by using the formulas given in (10) and (11). If maximum iteration number is reached, the algorithm goes to Step 3; otherwise, it goes to Step 7.

Step 7. The optimal solution is determined.

Algorithm 3 is used in the learning of ARMATPS-ANN. The data set was firstly separated two parts as learning and testing set for the using of learning algorithm. That deal with time series data, the testing test was selected at the end of the time series as block and so it is provided that the trained network is tested on the most recent data. Although the choice of the number of particles and the other parameters in the training of the network may vary from case to case according to data, there is generally a priori knowledge of the values of the parameters.

The choice of the number of inputs of the network, i.e. the determination of p and q, is determined by the user through trial and error. However, it can be used as prior knowledge in the results obtained from classical time series analysis. For instance, if the fit linear model is *ARMA(3,2)* for the data; the p and q values in ARMATPS-ANN can be taken as 3 and 2 respectively. And also, it is possible to use an optimization algorithm to determine these values, but the choice of the parameters by optimization method is not discussed in this study.

5 Applications

To evaluate the performance of the proposed ARMATPS-ANN, three real-life time series and a simulation study based on Istanbul Stock Exchange (BIST100) time series data were carried out. The first real-world time series is Australian beer consumption [20] between 1956 Q1 and 1994 Q1 is used to examine the performance of proposed method. The last 16 observations of the time series were used for test set. The graph of time series is given in Figure 3.

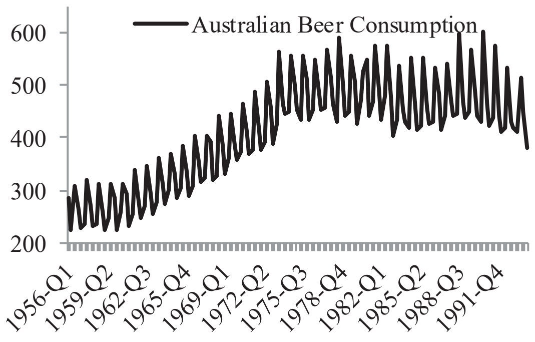


Figure 3. The time series graph of Australian beer consumption data

The input numbers were changed from 1 to 8, model order was changed from 2 to 5 when PS-ANN was applied to data. The input numbers were changed from 1 to 8 for time series and error series, model order were changed from 2 to 5 when ARMATPS-ANN was applied to data. The forecast results obtained from architecture which produces the best result for the test data for PS-ANN and ARMATPS-ANN are given in Table 2. The results of the methods in Table 2; multiplicative seasonal neural network proposed by [19], linear and nonlinear neural network proposed by [21], radial basis multiplicative neuron model neural network proposed by [3], recurrent multiplicative neuron model neural network proposed by [2] are taken related papers. Table 2 gives the values of RMSE and mean of absolute percentage error (MAPE) criteria for forecasts. RMSE and MAPE are calculated from the Equations 15 and 16.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \widehat{y}_i}{y_i}\right|. \qquad (16)$$

Table 2. Forecasting results for Australian beer consumption

Method	RMSE	MAPE
[19]	22.1700	0.0394
[21]	18.7888	0.0357
[3]	15.8378	0.0300
[2]	17.8573	0.0329
PS-ANN	20.0886	0.0352
ARMATPS-ANN	15.7100	0.0282

Table 3. Forecasting results for Australian beer consumption

Wy	1.488314	2.47328	0.62898
	-0.89127	0.071054	0.693617
	1.088966	-0.08612	-0.84267
	-0.36162	-0.00751	0.784274
	0.948041	0.143927	0.190732
	1.479005	0.622298	-0.11653
	-0.37843	0.210134	-0.77739
	0.300648	-0.35385	0.583702
We	0.397623	0.416693	1.112363
	0.897966	1.007118	-0.94232
	0.479383	0.19388	1.889696
	0.812074	0.703009	0.517137
	0.422582	1.765491	1.691538
	0.809445	2.050321	0.053912
	1.858237	0.267968	3.080951
	0.809625	-0.26929	0.959872
θ	-1.60533	-0.94613	-0.31296

The best test result of PS-ANN was produced with 8 inputs and 2nd order network. The best test result of ARMATPS-ANN was produced with 8 lagged variables of the time series $(p = 8)$, 8 lagged variables of the error series $(q = 8)$ and 3th order architecture $(K = 3)$.

Optimal weights of ARMATPS-ANN and forecast values for the test set are given in Tables 3 and 4. The second real-world time series is the amount of carbon dioxide measured monthly in Ankara capitol of Turkey (ANSO) between March 1995 and April 2006. The graph of ANSO time series is presented in Figure 4.

This time series has both trend and seasonal components and its period is 12. The first 124 observations were used for training and the last 10 observations are used for test set. Elman type recurrent ANN ([22]), MNM-ANN, the methods [19, 2, 21], PS-ANN and ARMATPS-ANN methods were applied to the time series. The results of Elman type recurrent ANN ([22]), MNM-ANN, [19, 2, 21] were taken from Egrioglu et al. [2]. The number of lagged variables of time series and error series were changed from 1 to 12 when PS-ANN and ARMATPS-ANN methods were applied to data. The best architecture of the PS-ANN is architecture with 8 inputs and third order. The best architecture of ARMATPS-ANN is an architecture with two lagged variables of time series (p=2), 12 lagged variables of error series (q=12) and third order. The MSE (the square of RMSE) and MAPE values for all methods are given in Table 5.

Table 4. Forecasts of ARMATPS-ANN and test data

Test Data	ARMATPS-ANN
430.50	436.24
600.00	588.94
464.50	467.09
423.60	407.44
437.00	461.93
574.00	563.25
443.00	451.96
410.00	409.08
420.00	422.74
532.00	560.97
432.00	427.05
420.00	406.23
411.00	423.26
512.00	506.39
449.00	422.50
382.00	409.69

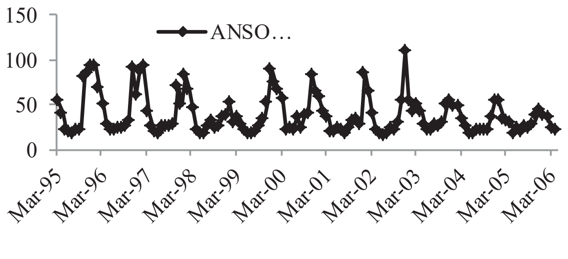


Figure 4. Time series graph of ANSO

Table 5. The forecast results for ANSO time series

Method	RMSE	MAPE
[22]	13.4821	0.099
MNM-ANN	40.2006	0.1822
[19]	91.0100	0.0887
[21]	12.7263	0.0944
[2]	8.6289	0.0761
PS-ANN	14.8600	0.1119
ARMATPS-ANN	5.6233	0.067

The third real world time series data is Turkey Electricity consumption data which is obtained from Turkish Energy Ministry. It was monthly obtained from January 2002 to December 2013. The graph of time series is given in Figure 5. The last twelve observations were taken as test set and others are training set.

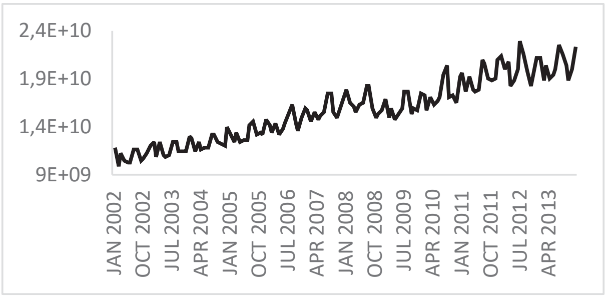


Figure 5. Turkey Electricity Consumption data

MLP-ANN, SARIMA, MNM-ANN and [21] methods were applied to the time series as benchmarks. The obtained forecast results are given in Table 6.

Table 6. The forecast results for Turkey Electricity Consumption time series

Method	RMSE	MAPE
MLP-ANN	1065870606	0.0398
SARIMA (0,1,1)(0,1,1)12	917321409	0.0388
MNM-ANN	813259007	0.0301
[21]	820978567	0.0254
PS-ANN	697763267	0.0281
ARMATPS-ANN	651383248	0.0227

Finally, a simulation study was carried out to investigate the performance of the proposed method. In the simulation study, the BIST100 index time series was used according to the closing prices calculated between 2.1.2009 and 12.8.2015. The time series is a 5-day series and the total numbers of observations are 1661 and the graph of time series is given in Figure 6. In the simulation study, 200 different time series were randomly drawn from the whole series with a size between 10% and 12% of the total number of observations. 10% of the number of observations is 166 and 12% is 200.

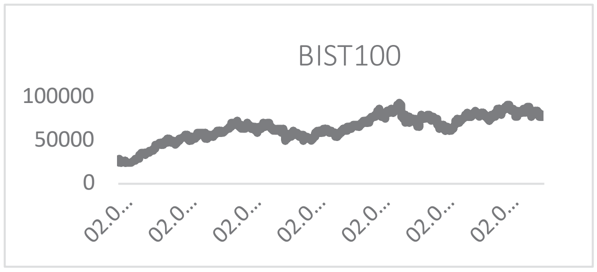


Figure 6. BIST100 closing price between 2.1.2009 and 12.8.2015

Table 7. The statistics for RMSE values which are obtained from simulated series

	[2]	**PSNN**	**ARMATPSNN**
Mean	2986.90	1272.37	1189.03
Median	2437.31	1106.78	1016.59
Standard Dev.	1989.05	714.87	640.87
Inter Quantile Range	2530.59	719.63	644.90
Minimum	377.41	306.97	319.88
Maximum	9902.74	3831.18	3559.26

In order to extract the time series from the data by the simulation method, the length of the data is firstly obtained by generating a random number (ns) from 166 to 200. Second, the number of the first observation of the time series was determined by drawing a random number from the range of 1 to 1661-*ns*. These processes were repeated 200 times and 200 different time series were obtained from the BIST 100 time series.

200 different time series were analyzed by taking the last 7 observations as test set with [2], PS-ANN and ARMATPS-ANN methods. Descriptive statistics for the RMSE values obtained as a result of the analysis are given in Table 7.

Three methods were compared by using Kruskal Wallis H-test. T-test result shows that $p<0.001$. It means there is an important difference between three methods. If Table 7 is examined, the smallest median and interquartile range were obtained from the proposed ARMATPS-ANN. The similar results were obtained from the other statistics.

6 Conclusion and Discussion

ANN has been commonly used to obtain forecasts for non-linear time series. Researchers have proposed several ANN types for forecasting problem in recent years. It is proved that PS-ANN which is a high order ANN type gives successful forecasting results for time series forecasting problem. PS-ANN is a feed-forward ANN type. The contribution of this paper is to propose a recurrent PS-ANN type. The inputs of the proposed ARMATPS-ANN are time series like linear ARMA model and the lagged variables of error series. In the proposed method, recurrent is performed with error values which are the difference between the output and the target value, not with the output values. Thus, the lagged variables of the error series given as inputs to the network allow adapting itself for the previous observations of ANN.

The proposed new ANN is compared with several ANN types such as PS-ANN, MLP-ANN, MNM-ANN, MS, RMNM and Elman. When real-life time series applications and simulation study results are examined it is seen that ARMATPS-ANN gives better prediction results than other ANNs according to both RMSE and MAPE criteria. In particular, ARMATPS-ANN gives better predictive results than PS-ANN; this case is a proof that adding a feedback mechanism like this paper is useful. Future work will focus on a training algorithm that will not be affected by outliers for ARMATPS-ANN and hybridization of ARMATPS-ANN with classical linear time series models.

References

[1] R.N. Yadav, P.K. Kalra, J. John, Time series prediction with single multiplicative neuron model, Applied Soft Computing, 7, 2007, 1157-1163.

[2] E. Egrioglu, C.H. Aladag, U. Yolcu, and E. Bas, Recurrent multiplicative neuron model artificial neural network for non-linear time series forecasting, Neural Processing Letters 41(2), 2015, 249-258.

[3] O. Gundogdu, E. Egrioglu, C.H. Aladag, and U. Yolcu, Multiplicative neuron model artificial neural

network based on gauss activation function, Neural Computing and Applications 27(4), 2015, 927-935

[4] D.E. Rumelhart, and J.L. Mcclelland, Parallel distributed processing: explorations in the microstructure of cognition, Cambridge (Britian): MIT Press, 1986.

[5] C.L. Giles, and T. Maxwell, Learning, invariance, and generalization in a high-order neural network, Appl Opt, 26(23), 1978, 4972–8.

[6] R. Durbin, and D.E. Rumelhart, Product units: a computationally powerful and biologically plausible extension to back propagation networks, Neural Computation, 1, 1989:133–42.

[7] Y. Shin, and J. Gosh, The Pi-sigma Network: An efficient higher-order neural network for pattern classification and function approximation. In Proceedings of the International Joint Conference on Neural Networks, 1991.

[8] R. Ghazali. A. Husaini, L.H. Ismail, T. Herawan, and Y.M. Hassim, The performance of a recurrent HONN for temperature time series prediction, 2014 International Joint Conference on Neural Networks (IJCNN), July 6-11, Proceeding Book, page 518-524, Beijing, China, 2014.

[9] R. Ghazali. A. Husaini, and W. El-Deredy, Application of ridge polynomial neural networks to financial time series prediction. In: 2006 International joint conference on neural networks; July, 16–21, 2006, 913–20.

[10] R. Ghazali, A.J. Hussain, P. Liatsis, and H. Tawfik, The application of ridge polynomial neural network to multi-step ahead financial time series prediction, Neural Computing & Applications, 17(3), 2008, 311–323.

[11] H. Tawfik, and P. Liatsis, Prediction of non-linear time-series using higher-order neural networks, Proceeding IWSSIP'97 Conference, Poznan, Poland, 1977.

[12] N. Yong, and D. Wei, A hybrid genetic learning algorithm for Pi– sigma neural network and the analysis of its convergence, In: IEEE fourth international conference on natural computation, 19–23, 2008

[13] J. Nayak, B. Naik, and H.S. Behera, A hybrid PSO-GA based Pi sigma neural network (PSNN) with standard back propagation gradient descent learning for classification. International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014, art. no. 6993082, 878-885, 2014b.

[14] J. Nayak, B. Naik, and H.S. Behera, and A. Abraham, Particle swarm optimization based higher order neural network for classification, Smart Innovation, Systems and Technologies, 31, 2015, 401-414.

[15] L. Chien-Kuo, Memory-based Sigma–Pi–Sigma neural network, IEEE SMC, TP1F5; 2002, 112–8.

[16] A.J. Hussain, and P. Liatsis, Recurrent Pi–Sigma networks for DPCM image coding, Neurocomputing, 55, 2002, 363–82.

[17] J. Nayak, D.P. Kanungo, B. Naik, and H.S. Behera, A higher order evolutionary Jordan Pi-sigma neural network with gradient descent learning for classification , 2014 International Conference on High Performance Computing and Applications, ICHPCA 2014, Article number 7045328.

[18] J. Kennedy, R. Eberhart, Particle swarm optimization, In Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, USA, IEEE Press., 1995, 1942–1948.

[19] C.H. Aladag, U. Yolcu, and E. Egrioglu, A new multiplicative seasonal neural network model based on particle swarm optimization, Neural Processing Letters 37(3), 2013, 251-262.

[20] G. Janacek, Practical time series. Oxford University Press Inc., New York, 156, 2001.

[21] U. Yolcu, E. Egrioglu, C.H. Aladag, A new linear & nonlinear artificial neural network model for time series forecasting, Decision Support Systems, 2013, 1340–1347.

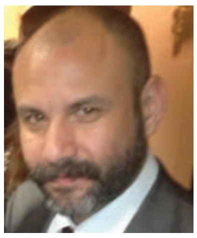
[22] J.L. Elman, Finding structure in time, Cognitive Science, 14 (2), 1990, 179–211.

Esra Akdeniz is an associate professor at Marmara University, Turkey. She received the PhD degree in statistics from Gazi University, Turkey, Her research interests include statistical analysis, regression analysis, and time series.

Erol Egrioglu is a professor at Giresun University, Turkey. He received the BSc degree in statistics from the University of Ondokuz Mayis Turkey, in 1998. He received the MSc degree (MSc thesis title: Bayesian analysis of ARMA model and an application) in statistics at the Faculty of Science and Arts of the University of Ondokuz

Mayis, Turkey, in 2002 and the PhD degree in statistics at the Faculty of Science, of the University of Hacettepe, Turkey, in 2006. His research interests include time series analysis, bayesian approaches, fuzzy time series and artificial neural networks, heuristic and evolutionary algorithms, robust methods and bootstrap methods.

Eren Bas is an assistant professor at Giresun University, Turkey. He received the BSc, the MSc (MSc thesis title: A new approach based on the genetic algorithm for fuzzy time series analysis) and the PhD (A new robust learning algorithm with multiplicative neuron model in artificial neural networks) degrees in statistics from the University of Ondokuz Mayis, Turkey, in 2009, 2011 and 2014, respectively. His research interests include fuzzy time series, artificial neural networks, heuristic and evolutionary algorithms and robust methods.



Ufuk Yolcu is an associate professor at Giresun University, Turkey. He received the BSc, the MSc (MSc thesis title: High order fuzzy time series forecasting model based on artificial neural networks) and the PhD (PhD thesis title: Multivariate analysis in fuzzy time series) degrees in statistics from the University of Ondokuz Mayis, Turkey, in 2003, 2008 and 2011, respectively. His research interests include fuzzy systems, fuzzy time series, artificial neural networks, heuristic and evolutionary algorithms, robust methods and bootstrap methods.