Threshold Single Multiplicative Neuron Artificial Neural Networks for Non-linear Time Series Forecasting

Asiye Nur Yildirim[a] Eren Bas[a] and Erol Egrioglu[a,b]

[a]*Department of Statistics, Faculty of Arts and Science, Giresun University, Giresun, Turkey*
[b]*Department of Management Science, Lancaster University, Lancaster, UK*

ABSTRACT
Single multiplicative neuron artificial neural networks have different importance than many other artificial neural network models because they do not have complex architecture problem, too many parameters and they need more computation time to use. In single multiplicative neuron artificial neural network, it is assumed that there is a one data generation process for the time series. However, using a single model for all observations of time series may have misleading results. Many time series need an assumption that they have two data generation process or more. Based on this idea, the threshold model structure can be employed in a single multiplicative neuron model artificial neural network for taking into considering data generation processes problem. In this study, a new artificial neural network type is proposed and it is called a threshold single multiplicative neuron artificial neural network. It is assumed that time series have two data generation processes according to the architecture of single multiplicative neuron artificial neural network. Training algorithms are proposed based on harmony search algorithm and particle swarm optimization for threshold single multiplicative neuron artificial neural network. The proposed method is tested by various time series data sets and compared with well-known forecasting methods by considering different error measures. Finally, the performance of the proposed method is evaluated by a simulation study.

Keywords: threshold; multiplicative neuron model; harmony search algorithm; particle swarm optimization; forecasting

1. Introduction

Artificial neural networks (ANNs), one of the commonly used artificial intelligence methods; are based on mathematical modelling of the learning process inspired by the human brain. ANNs are the structures formed by artificial neural neurons coming together. In general, ANNs have three layers: the input layer, hidden layer and the output layer. All layers have a significant effect on the performance of the network and especially the hidden layer has special importance since its number cannot be determined precisely. Based on this problem, Yadav et al. (2007) proposed the single multiplicative neuron model artificial neural networks (SMNM-ANN). In the studies about SMNM-ANN, the model obtained from the network is only a single model and this situation can be accepted when time series is stationary. But a time series is not always stationary. It is not possible to encounter a stationary time series in real life. The time series include some components such as trend or/and seasonality. It is not realistic to explain and analyse this

CONTACT Eren BAS eren.bas@giresun.edu.tr Department of Statistics, Faculty of Arts and Science, Giresun University, Giresun, Turkey

type of time series with a single model. Besides, using a single model assumption may have misleading results.

In this study, unlike the other studies about SMNM-ANN, a threshold value is used to obtain the output of the SMNM-ANN. With this obtained threshold value, it is determined which weight and bias values will be used to obtain the output of the system. The proposed new artificial neural network is called a threshold single multiplicative neuron artificial neural network (TS-SMNM-ANN). In the training of TS-SMNM-ANN, to find the optimum weight, bias and threshold values, harmony search algorithm (HSA) proposed by Geem et al. (2001) and particle swarm optimization (PSO) proposed by Kennedy and Eberhart (1995) are used separately.

Although ANN is a machine learning algorithm, it is a nonlinear regression model. Just like nonlinear models, the parameters of it can be estimated with the Levenberg Marquart algorithm. In this study, the obtained model is a piecewise non-linear regression model and it has been focused on point estimation.

One of the novelties of the proposed method is that two different models were obtained with the proposed method, unlike many ANN models. Moreover, the performance measures show that using two models give better analyse results than many artificial neural network models with a single model.

To analyse the performance of the proposed method, Australian Beer Consumption (AUST) time series data between the years 1956 and 1994, Turkey Electricity Consumption (TEC) data observed monthly between the first month of 2002 and last month of 2013 and Taiwan Future Exchange (TAIFEX) data with observations between 03.08.1998 and 30.09.1998 are analysed with both proposed methods and many methods proposed in the literature. Finally, a simulation study is made for the evaluation of the proposed method.

The rest part of the paper can be outlined as below: The literature review of SMNM-ANN is given in the second section of the paper. The third section is about HSA and the fourth section is about PSO. The proposed two training algorithms named threshold single multiplicative neuron model artificial neural networks with HSA (TS-SMNM-ANN-HSA) and threshold single multiplicative neuron model artificial neural network with PSO (TS-SMNM-ANN-PSO) are given step by step with details in section five. Section six presents the results obtained from both proposed methods with some other methods proposed in the literature and also a simulation study is made in section seven. Finally section eight presents conclusions and discussions.

2. Literature Review

As mentioned before, SMNM-ANN was proposed by Yadav et al. (2007) and it is frequently used in many fields and especially in forecasting problems in recent years. SMNM-ANN does not include the determination of hidden layer unit number problem and it has both reasonable calculation time and superior forecasting performance. Popular areas where SMNM-ANN is used and the studies about these areas were summarized below.

There are many studies use for forecasting aim, which is the most frequently used area, for SMNM-ANN. Zhao and Yang (2009) proposed a method that the training of SMNM-ANN was done by PSO for the time series forecasting problem. Worasucheep and Chongstitvatana (2009) used a multi-strategy differential evolution algorithm (DEA) for financial prediction with SMNM-ANN. Wu et al. (2013) used online training algorithms based on SMNM-ANN for energy consumption estimation. Henao et al. (2013) proposed a hybrid method uses a combination of the SARIMA and SMNM-ANN to estimate

electricity demand. Ilter et al. (2014) showed the effects of differencing and transforming in SMNM-ANN for forecasting. Wu et al. (2015) used SMNM-ANN for hourly wind speed estimation. Cui et al. (2015) proposed a new training algorithm for SMNM-ANN with firefly optimization algorithm for time series forecasting problem. Bas (2016) used differential evolution algorithm (DEA) for the training of SMNM-ANN in time series forecasting problem. Bas et al. (2016a) used the artificial bat algorithm (ABA) for the training of SMNM-ANN in time series forecasting problem. Egrioglu et al. (2017) used a hybrid forecasting method based on exponential smoothing and SMNM-ANN for stock exchange data sets. Cagcag Yolcu et al. (2018) proposed an SMNM-ANN with the autoregressive coefficient for time series forecasting. In Cagcag Yolcu et al. (2018), the weights and biases of SMNM-ANN were obtained by way of autoregressive equations. Kolay (2019) used sine cosine algorithm for the training of SMNM-ANN for time series forecasting problem.

There are some studies use SMNM-ANN in fuzzy time series methods. Aladag (2013) used SMNM-ANN to determine fuzzy relations in a fuzzy time series model. Cagcag Yolcu (2013) proposed a hybrid fuzzy time series approach based on fuzzy clustering and SMNM-ANN. Egrioglu et al. (2014) proposed a fuzzy time series method on SMNM-ANN and membership values. Yolcu (2017) proposed a new high order multivariate fuzzy time series forecasting model. In this model, SMNM-ANN was used to define multivariate fuzzy relations. Cagcag Yolcu and Lam (2017) proposed a combined robust fuzzy time series method uses SMNM-ANN to determine fuzzy relations. Cagcag Yolcu and Alpaslan (2018) proposed a hybrid fuzzy time series model uses SMNM-ANN.

There are also some studies use SMNM-ANN for different aims. Burse et al. (2010) proposed an improved backpropagation (BP) algorithm to avoid local minima in SMNM-ANN. Kolay et al. (2016) used SMNM-ANN for the classification of some data sets and compared it with some ANNs types. Kandpal and Ashish (2017) compared SMNM-ANN with multilayer perceptron for classification. Basiouny et al. (2017) used SMNM-ANN and PCA algorithms for Wi-Fi fingerprinting for the indoor positioning system. Kandpal and Mehta (2019) and Sharma et al. (2019) used SMNM-ANN for classification. Nigam (2019) used SMNM-ANN in reinforcement learning.

In all these studies mentioned by the SMNM-ANN, the model obtained at the end of the training process is only a single model and root mean square error (RMSE) and/or mean absolute percentage error (MAPE) criteria are generally used as error metrics in these studies. The advantages and disadvantages of the proposed method can be given as follows.

- The proposed method in this paper is the first method for SMNM-ANN uses a threshold mechanism.
- The proposed method can be used when a time series has a trend or/and seasonality.
- The proposed method uses HSA and PSO artificial intelligence optimization techniques for the training of SMNM-ANN instead of using a derivative-based method such as Levenberg Marquardt.
- The proposed threshold mechanism can be adapted to the many artificial neural network models.
- Using two models instead of a single model increase the number of parameters to be optimized.

3. Harmony Search Algorithm

HSA was proposed by Geem et al. (2001) and it is an optimization algorithm based on the principle of obtaining the best harmonic melody with the notes played by musicians in an orchestra. While the aesthetic quality of notes and tones played with different instruments is improved by practising in music studies, this improvement in function solution is achieved by successive iterations. The HSA aims to investigate whether the solution vector (harmony) obtained by certain updates is better than the worst solution in the memory or not. If the obtained solution vector is better than the worst solution vector in the harmony memory according to the objective function, the obtained solution vector is replaced by the worst solution vector, otherwise, it is held in the harmony memory. The HSA is given step by step in Algorithm 1.

Algorithm 1. The steps of HSA

Step 1. Determination of algorithm parameters
The parameters to be used in HSA are as follows.
HM: Harmony Memory
HMS: Harmony Memory Search
$HMCR$: Harmony Memory Considering Rate
PAR: Pitch Adjusting Rate
n: The number of variables
Step 2. Creating harmony memory

HM for the HSA is created as in Equation (1).

$$HM = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{HMS1} & x_{HMS2} & x_{HMS3} & x_{HMSn} \end{bmatrix} = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_{HMS}' \end{bmatrix} \quad (1)$$

Here, $x_{ij},\ i = 1,2, \ldots HMS\ ; j = 1,2, \cdots, n$ is expressed as the note value in the musician's memory and generated randomly. For example; x_{12} is the value of the first note of the second musician's memory. In HSA, each solution vector is represented by $x_i', i = 1, 2, \cdots, HMS$. There are HMS solution vectors in HSA. The representation of the first solution vector is given in Equation 2.

$$x_1' = [x_{11}, x_{12}, \cdots, x_{1n}] \quad (2)$$

Step 3. Calculation of objective function values

The objective function values are calculated for each randomly generated solution vector as in Equation (2). Here, $f(x_1')$ represents the value of the objective function calculated for the first solution vector.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{HMS1} & x_{HMS2} & x_{HMS3} & x_{HMSn} \end{bmatrix} = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_{HMS}' \end{bmatrix} = \begin{bmatrix} f(x_1') \\ f(x_2') \\ \vdots \\ f(x_{HMS}') \end{bmatrix} \quad (3)$$

Step 4. Developing the new harmony

The probability of $HMCR$ takes a value between 0 and 1 and $HMCR$ is the probability of selecting a value from the existing values in the HM, $(1 - HMCR)$ is the ratio of selecting a random value from the possible value ranges. The value of the first decision variable (x_{i1new}) for the new vector is selected from any value within the specified HM range. Accordingly, the new harmony is obtained with the help of Equation (4).

$$x_{ijnew} = \begin{cases} x_{ijnew} \in \{x_{ij}; \mathrm{i} = 1,,2,\cdots,\mathrm{HMS}\} & \mathrm{rnd} < \mathrm{HMCR} \\ U(0,1)x(max - min) + min & \mathrm{otherwise} \end{cases} \quad (4)$$

The PAR parameter determines whether the tone adjustment can be applied to each decision variable selected from memory with the possibility of $HMCR$ or not.

$$x_{ijnewpitch} = \begin{cases} Yes & \mathrm{rnd} < \mathrm{PAR} \\ No & \mathrm{otherwise} \end{cases} \quad (5)$$

In Equation (5), rnd is a randomly generated number between $U(0,1)$. If this generated random number is smaller than the PAR ratio, this value is replaced with the value closest to it. If tone adjustment is to be made for each x_{ijnew} decision variable and it is assumed that x_{ijnew} is the next value in the value vector that the decision variable can take, the new value of $x_{ijnew}(k)$ is set as in Equation (6);

$$x_{ij} \leftarrow x_{ij}(k \pm m) \quad (6)$$

In Equation (6), $m \in \{\cdots, -2, -1, 1, 2, \cdots\}$ is the neighbourhood depth index. In the literature, values between 0.7-0.95 for $HMCR$ and 0.05-0.7 for PAR are possible values. (Geem, 2006)

Step 5. Updating of harmony memory

If the new harmony vector is better than the worst vector in the HM according to the objective function, the worst vector is removed from memory and replaced with the new harmony vector HM.

Step 6. Checking stopping condition

Steps 4 - 6 are repeated until the termination criterion is met. The flowchart of HSA was given in Figure 1.

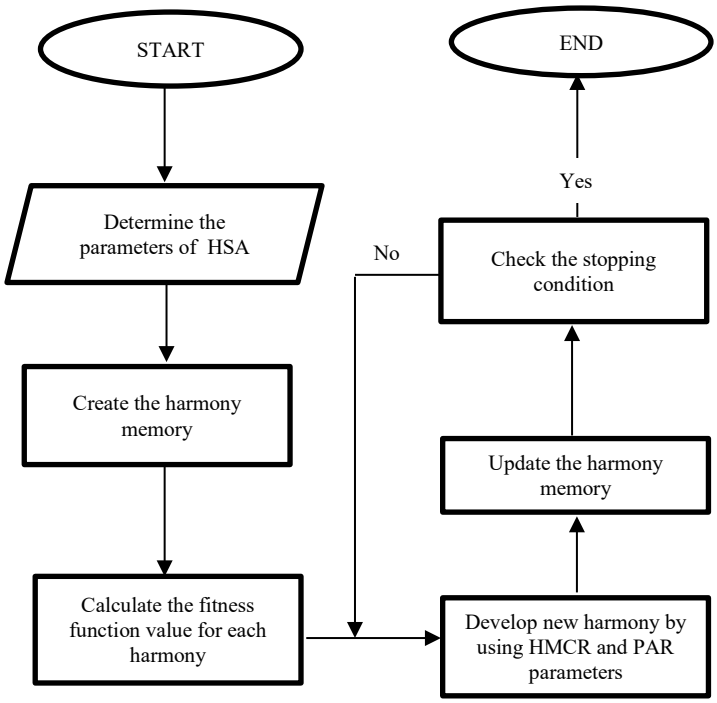

Figure 1. Flowchart for HSA

4. Particle Swarm Optimization

PSO is an optimization algorithm proposed by Kennedy and Eberhart (1995), inspired by the behaviour of bird flocks. One of the most important advantages of PSO is that it does not require derivative information in the optimization process. Searching food for birds is similar to searching a solution for a problem. Each solution called as a particle in PSO is a bird in search space. The value of the coordinates in the function that means the success value of the particle is a measure of the distance of a bird to food. A particle has to keep its coordinates, speed and coordinates with the highest success achieved. The change of the coordinate and velocity values in the solution space is a combination of the best coordinates of their neighbours and their best coordinates. Although it has been achieved successful results, many contributions have been made to increase the speed of convergence. The first contribution is using the inertia weight parameter proposed by Shi and Eberhart (1998). With this approach, the velocity value in the previous iteration is included in the velocity update. Another contribution is the linear change of cognitive, social coefficient values and inertia weight within iterations. The improved PSO (IPSO) with contributions is given below in steps with Algorithm 2.

Algorithm 2. The steps of IPSO

Step 1. Determination of algorithm parameters.
The parameters to be used in the algorithm are as follows and are initially determined.
The number of Particles (Ps)
Particle positions (PP)
Particle velocities (V)
Cognitive coefficient (c_1)

Social coefficient (c_2)
Inertia weight (w)
Number of variables(n)

Step 2. The determination of particle positions and velocities.
Considering the number of particles and variables in the PSO, the particle size (PP) with $Psxn$ dimension and velocity (v) values are generated as in Equations (7-8), respectively.

$$PP = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{Ps1} & P_{Ps2} & \cdots & w_{Psn} \end{bmatrix} \quad (7)$$

$$V = \begin{bmatrix} V_{11} & V_{12} & \cdots & V_{1n} \\ V_{21} & V_{22} & \cdots & V_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ V_{Ps1} & V_{Ps2} & \cdots & V_{Psn} \end{bmatrix} \quad (8)$$

Here; P_{ij} is ith position value of jth particle and V_{ij} ith velocity value of jth particle and they are generated randomly. ($i = 1,2, \dots, Ps$; $j = 1,2, \cdots, n$).

Step 3. Determination of P_{best} and G_{best} depending on the objective function value.
The objective function values are calculated for each randomly generated solution vector. The best positions of the particles are stored in the P_{best} vectors. The best state of all particles is indicated by P_{best_g} or G_{best}. P_{best_g} and G_{best} vectors are given in Equations (9-10).

$$P_{best\,i} = [P_{i1} \quad P_{i2} \quad \dots \quad P_{in}];\ i = 1,2,..,Ps \quad (9)$$

$$P_{best_g} = G_{best} = [P_{g1} \quad P_{g2} \quad \dots \quad P_{gn}] \quad (10)$$

Step 4. Updating the inertia weight (w), cognitive coefficient (c_1) and social coefficient (c_2)
To increase the convergence speed of PSO, the coefficients w, c_1 and c_2 are updated in the iterations as in Equation (11-13)

$$w = (w_{max} - w_{min}) \times \frac{maxk-k}{maxk} + w_{min} \quad (11)$$

$$c_1 = \left(c_{1f} - c_{1i}\right) \times \frac{k}{maxk} + c_{1i} \quad (12)$$

$$c_2 = \left(c_{2f} - c_{2i}\right) \times \frac{k}{maxk} + c_{2i} \quad (13)$$

Here, (c_{1f}, c_{1i}), (c_{2f}, c_{2i}) and (w_{max}, w_{min}) are the maximum and minimum values for the cognitive coefficient, social coefficient and inertia weight, respectively. $maxk$ is the maximum number of iterations and k is the number of iterations.

Step 5. Calculation and update of new velocities and positions. The new velocity and position values are calculated by Equations (14-15).

$${V_{ij}}^{k+1} = w \times {V_{ij}}^{k} + c_1 \times rand_1 \times \left({P_{best}}_{ij}^{k} - P_{ij}^{k}\right) + c_2 \times rand_2 \times \left({G_{best}}_{ij}^{k} - P_{ij}^{k}\right) \quad (14)$$

$$P_{ij}^{k+1} = P_{ij}^{k} + {V_{ij}}^{k+1} \quad (15)$$

Step 6. Checking stopping condition
Repeat Step 3 - Step 5 until the termination criterion is met. The flowchart of IPSO was given in Figure 2.

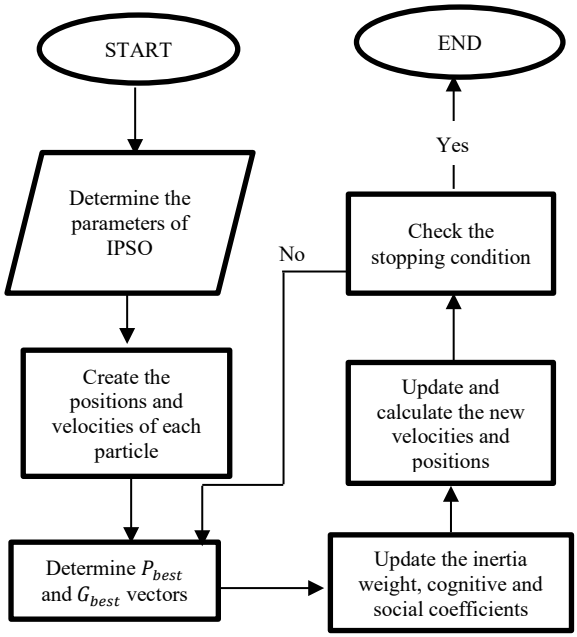

Figure 2. Flowchart for IPSO

5. Threshold Single Multiplicative Neuron Model Artificial Neural Network and Training Algorithms

The threshold SMNM-ANN (TS-SMNM-ANN) has a dual structure in its calculation formulas. The architecture of the TS-SMNM-ANN is given in Figure 3.

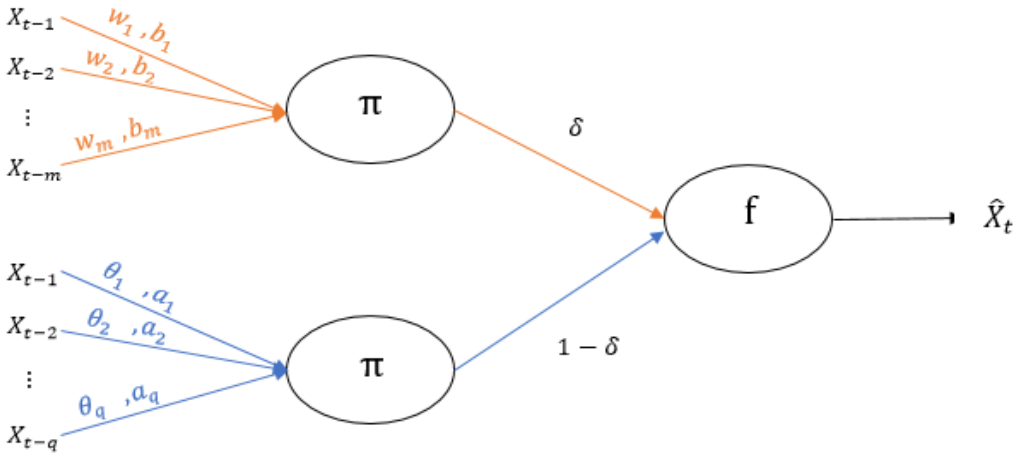

Figure 3. The architecture of TS-SMNM-ANN

In Figure 3, $X_{t-1}, X_{t-2}, \cdots, X_{t-m}$ and $X_{t-1}, X_{t-2}, \cdots, X_{t-q}$ are the inputs for the first and second multiplicative neuron model. Which model's weight and bias values should be used to obtain the output of the network is determined by a parameter δ given in Equation (16).

$$\delta = \begin{cases} 1 & X_{t-d} < c \\ 0 & X_{t-d} \geq c \end{cases} \tag{16}$$

In Equation (16), c is the threshold value, d is a randomly generated integer value and X_{t-d} is the lag value compared with the threshold. According to the δ value obtained from this equation, the net value of the network determined according to the number of inputs of the model to be used are calculated with Equation (17) and the output of the network is obtained with Equation (18).

$$net = \begin{cases} \prod_{j=1}^{m} w_j X_{t-j} + b_j & \delta = 1 \\ \prod_{j=1}^{q} \theta_j X_{t-j} + a_j & \delta = 0 \end{cases} \tag{17}$$

$$\hat{X}_t = \frac{1}{1+\exp(-net)} \tag{18}$$

The training algorithms used in the training of threshold SMNM-ANN were given in Algorithms 3 and 4, respectively.

Algorithm 3: The training of threshold multiplicative neuron model with HSA (TS-SMNM-ANN- HSA)

Step 1. Determine the parameters of TS-SMNM-ANN-HSA

m: the number of inputs for the model of observations smaller than the threshold value.
q: the number of inputs for the model of observations bigger than the threshold value.
HM: harmony memory
HMS: harmony memory capacity
$HMCR$: Harmony memory consideration rate
PAR: tone adjustment ratio
c: threshold value
d: the lag compared with the threshold

Step 2. Creating of the harmony memory

The harmony memory was created as in Equation (19)

$$HM = \begin{bmatrix} w_{11} & \cdots & w_{1m} & b_{11} & \cdots & b_{1m} & \theta_{11} & \cdots & \theta_{1q} & a_{11} & \cdots & a_{1q} & c_1 & d_1 \\ w_{21} & \cdots & w_{2m} & b_{21} & \cdots & b_{2m} & \theta_{21} & \cdots & \theta_{2q} & a_{21} & \cdots & a_{2q} & c_2 & d_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{HMS1} & \cdots & w_{HMSm} & b_{HMS1} & \cdots & b_{HMSm} & \theta_{HMS1} & \cdots & \theta_{HMSq} & a_{HMS1} & \cdots & a_{HMSq} & c_{HMS} & d_{HMS} \end{bmatrix} \tag{19}$$

Step 3. Calculation of the objective function value

It is determined which weight and bias values to be used to obtain the forecast of the system according to the relation between threshold value c and the lag value X_{t-d_i} given in Equation (18). The objective function value of each harmony is calculated with RMSE criterion given in Equation (20) by using the obtained output values $(\hat{X}_t)$and target values X_t.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(X_t - \hat{X}_t)^2}{n}} \tag{20}$$

Step 4. Development of the new harmony

In this Step, the new harmony is developed based on HMCR and PAR parameters.

Step 4.1. All weight and bias values in the harmony memory, the threshold value and the lag value to be compared with the threshold value are updated by using Equations (21).

$$HM_{ij} = \begin{cases} HM_{ij} \in \{HM_{ij}; \mathrm{i} = 1,,2,\cdots,\mathrm{HMS}, \mathrm{j} = 1,,2,\cdots,\mathrm{m}\} & \mathrm{rnd} < \mathrm{HMCR} \\ U(0,1)x(\max(HM_{ij}) - \min(HM_{ij})) + \min(HM_{ij}); \mathrm{i} = 1,,2,\cdots,\mathrm{HMS}, \mathrm{j} = 1,,2,\cdots,\mathrm{m} & otherwise \end{cases} \quad (21)$$

The new harmony created for the model with (m, q) input is given in Figure 4.

w'_1	$\cdots$	w'_m	b'_1	$\cdots$	b'_m	θ'_1	$\cdots$	θ'_q	a'_1	$\cdots$	a'_q	c'	d'

Figure 4. New harmony created for the model with (m, q) input

Step 4.2. In the new harmony obtained as a result of the HMCR process, the tone adjustment process made by PAR parameter is performed for the variable values selected from the harmony memory by using Equation (22) and new adjusted value of $HM'_j(k)$ is given in Equation (23).

$$HM'_j = \begin{cases} HM_j \pm depth(PAR \times HMCR) & \mathrm{rnd} < \mathrm{PAR} \\ w'_j & otherwise \end{cases} \quad (22)$$

$$wp'_j \leftarrow w'_j \pm k(PAR \times HMCR) \quad (23)$$

As a result of the PAR process, the updated harmony is given in Figure 5.

wp'_1	$\cdots$	wp'_m	bp'_1	$\cdots$	bp'_m	$\theta p'_1$	$\cdots$	$\theta p'_q$	ap'_1	$\cdots$	ap'_q	cp'	dp'

Figure 5. Updated harmony for the model with (m, q) input as a result of PAR operation

Step 5. Updating harmony memory

If the RMSE value of the harmony obtained by HMCR and PAR operations is smaller than the harmony with the worst RMSE value in the harmony memory, the harmony in the harmony memory is removed from the memory and the harmony obtained by HMCR and PAR operations is replaced of it; if not, no changes are made to the HM.

Step 6. Stop condition check

If it is reached to the maximum number of iterations or the RMSE value is less than a certain error (ε) value, the stop condition is met. Otherwise, Step 4 - Step 6 is repeated.

Algorithm 4: The training of threshold multiplicative neuron model with PSO (TS-SMNM-ANN- PSO)

Step 1. Determine the parameters used in the training process of TS-SMNM-ANN- PSO.

m: the number of inputs for the model created before the threshold

q: the number of inputs for the model created after the threshold
Ps: the number of particles
PP: particle position matrix
V: particle velocity matrix
C: threshold
d: the lag compared with the threshold

Step 2. Determine the particle positions and velocities

The PP matrix with $Psx[2(m+q)+2]$ positions is created as in Equation (24).

$$PP = \begin{bmatrix} w_{11} & \cdots & w_{1m} & b_{11} & \cdots & b_{1m} & \theta_{11} & \cdots & \theta_{1q} & a_{11} & \cdots & a_{1q} & C_1 & d_1 \\ w_{21} & \cdots & w_{2m} & b_{21} & \cdots & b_{2m} & \theta_{21} & \cdots & \theta_{2q} & a_{21} & \cdots & a_{2q} & C_2 & d_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{Ps1} & \cdots & w_{Psm} & b_{Ps1} & \cdots & b_{Psm} & \theta_{Ps1} & \cdots & \theta_{Psq} & a_{Ps1} & \cdots & a_{Psq} & C_{Ps} & d_{Ps} \end{bmatrix} = \begin{bmatrix} P_{1j} \\ P_{2j} \\ \vdots \\ P_{ij} \end{bmatrix} \quad (24)$$

Step 3. Determine P_{best} and G_{best} depending on the objective function value. The objective function value is calculated for each particle by using similar calculations in Step 3 of the previous algorithm.

Step 4. Update w inertia weight, $c_1$cognitive coefficient and c_2 social coefficients by using the linear increase or decrease formulas as given in the PSO algorithm.

Step 5. Calculate and update the new velocities and positions.

The new velocity and position values are calculated by using Equations (25-26). Here c_1 and c_2 are the cognitive and social coefficients respectively, in many studies, it is seen that good results are obtained when $c_1 = c_2 = 2$. $rand_1$ and $rand_2$ are randomly selected values generated by $U(0,1)$.

$$V_{ij}^{k+1} = w \times V_{ij}^{k} + c_1\, x\, rand_1 x \left(P_{best}{}_{ij}^{k} - P_{ij}^{k}\right) + c_2\, x\, rand_2 x \left(G_{best}{}_{ij}^{k} - P_{ij}^{k}\right) \quad (25)$$

$$P_{ij}^{k+1} = P_{ij}^{k} + V_{ij}^{k+1} \quad (26)$$

Step 6. Stop condition check**.** If it is reached to the maximum number of iterations or the RMSE value is less than a certain error (ε) value, the stop condition is met. Otherwise, Step 4 - Step 6 is repeated.

6. Applications

In the application part of the paper, Australian beer consumption data (AUST) time series data with 148 observations between the years 1956 and 1994, Turkey Electricity Consumption (TEC) data observed monthly between the first month of 2002 and last month of 2013 and Taiwan Future Exchange (TAIFEX) data with observations between 03.08.1998 and 30.09.1998 were analysed. To compare the performance of the proposed methods, the proposed methods were compared with many methods in terms of both RMSE and mean absolute percentage error (MAPE) criteria given in Equations (20) and (27) respectively.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \quad (27)$$

The number of observations of each data set and the number of test data set of each data set analysed in this paper, the number of inputs for the model created before threshold (m) and the number of inputs for the model created after the threshold (q) and the optimal m and q values were given in Table 1 for both TS-SMNM-ANN-PSO and TS-SMNM-ANN-HSA.

Table 1. Some features for data sets

			Tried Values		TS-SMNM-ANN-PSO		TS-SMNM-ANN-HSA	
Data	The number of observations	ntest	m	q	Optimal m, q values		Optimal m, q values	
AUST	125	16	2,4,8	2,4,8	4	4	8	4
TEC	135	12	5:15	5:15	14	6	12	10
TAIFEX	145	16	1:10	1:10	10	2	1	1

In this part of the paper, firstly, AUST time series data set was analysed with Multi-layer feed-forward neural network (ML-FF-ANN), Multilayer neural network based on PSO (ML-PSO-ANN), SMNM-ANN based on BP (BP-SMNM-ANN), SMNM-ANN based on PSO (PSO-SMNM-ANN), SMNM-ANN based on DEA (DEA-SMNM-ANN) proposed by Bas (2016), Radial basis artificial neural network (RB-ANN), Elman ANN (E-ANN), multiplicative seasonal ANN (MS-ANN) proposed by Aladag et al. (2013), SMNM-ANN based on ABA (ABA-SMNM-ANN) proposed by Bas et al. (2016a), multilayer feed-forward network based on trimmed mean neuron model (TMNM-MFF) proposed by Yolcu et al. (2015) and Pi-Sigma ANN (PS-ANN) except TS-SMNM-ANN-HSA and TS-SMNM-ANN-PSO methods. The RMSE and MAPE results obtained from the methods mentioned above for AUST test data were given in Table 2.

Table 2. The RMSE and MAPE values obtained from all methods for AUST test data

Methods	**RMSE**	**MAPE**
BP-SMNM-ANN	74.2551	0.0983%
ML-PSO-ANN	44.7780	0.0856%
RB-ANN	41.7000	0.0686%
TS-SMNM-ANN-HAS	33.3777	0.0633%
PSO-SMNM-ANN	26.7831	0.0489%
ML-FF-ANN	24.1052	0.0476%
E-ANN	22.6581	0.0436%
MS-ANN	22.1700	0.0394%
TMNM-MFF	21.0623	0.0399%
PS-ANN	20.0886	0.0352%
DEA-SMNM-ANN	19.7819	0.0372%
TS-SMNM-ANN-PSO	18.8777	0.0331%
ABA-SMNM-ANN	17.7054	0.0323%

Table 2 shows that although TS-SMNM-ANN-PSO method is the second-best method in terms of both RMSE and MAPE criteria the performance of TS-SMNM-ANN-HSA method is not good as TS-SMNM-ANN-PSO method for AUST test data.
Secondly, TEC time series data was analysed with BP-SMNM-ANN, MLP-ANN, ABA-SMNM-ANN proposed by Bas et al. (2016a), ML-PSO-ANN, SMNM-ANN, PS-ANN and PSO-SMNM-ANN methods except for TS-SMNM-ANN-HSA and TS-SMNM-

ANN-PSO methods. The RMSE and MAPE results obtained from the methods mentioned above for TEC test data were given in Table 3.

Table 3. The RMSE and MAPE values obtained from all methods for TEC test data

Methods	RMSE	MAPE
BP-SMNM-ANN	4243944603	19.54%
MLP-ANN	1065900000	3.98%
ABA-SMNM-ANN	924149635	3.73%
ML-PSO-ANN	915190000	3.39%
SMNM-ANN	813260000	3.01%
TS-SMNM-ANN-HSA	772262044	3.15%
PS-ANN	697763268	2.81%
PSO-SMNM-ANN	672790000	2.89%
TS-SMNM-ANN-PSO	605542432	2.38%

Table 3 shows that TS-SMNM-ANN-PSO method is the best method in terms of both RMSE and MAPE criteria. Besides, the performance of TS-SMNM-ANN-HSA method is not good as TS-SMNM-ANN-PSO method for TEC test data.

Finally, TAIFEX data was analyzed by BP-SMNM-ANN, PS-ANN, Lee et al. (2007), robust learning algorithm for SMNM-ANN (R-SMNM-ANN) proposed by Bas et al. (2016b), Lee et al. (2008), PSO-SMNM-ANN, Hsu et al. (2010), ABA-SMNM-ANN proposed by Bas et al. (2016a), DEA-SMNM-ANN proposed by Bas (2016), Aladag et al. (2009) methods except for TS-SMNM-ANN-HSA and TS-SMNM-ANN-PSO methods.

The RMSE and MAPE results obtained from the methods mentioned before for TAIFEX test data are given in Table 4.

Table 4. The RMSE and MAPE values obtained from all methods for TAIFEX test data

Methods	RMSE	MAPE
BP-SMNM-ANN	108.1627	1.17%
Lee et al. (2008)	102.9600	1.14%
PS-ANN	94.1439	1.01%
Lee et al. (2007)	93.4900	1.09%
R-SMNM-ANN	89.9655	0.90%
PSO-SMNM-ANN	88.5839	0.87%
Aladag et al. (2009)	83.5800	0.96%
Hsu et al. (2010)	80.0200	0.87%
ABA-SMNM-ANN	79.8623	0.95%
DEA-SMNM-ANN	79.2514	0.95%
TS-SMNM-ANN-HSA	77.4153	0.88%
TS-SMNM-ANN-PSO	77.0761	0.87%

Table 4 shows that TS-SMNM-ANN-PSO method is the best method in terms of both RMSE and MAPE criteria and TS-SMNM-ANN-HSA method is the second-best method in terms of both MAPE criteria for TAIFEX test data.

7. Simulation Study

To emphasize the efficiency of the proposed methods in the paper, a simulation study was performed. In the simulation study, the performance of the TS-SMNM-ANN-HSA and

TS-SMNM-ANN-PSO methods were compared with BP-SMNM-ANN, DEA-SMNM-ANN, PSO-SMNM-ANN and ML-PSO-ANN methods. In the simulation study, a data set for TS-SMNM-ANN was generated.

Algorithm 3. Creating a data set for TS-SMNM-ANN

Step 1. Let $nmı$ and $nqı$ be the number of the inputs, first observations of time series $X_0 = (x_1, x_2, \dots, x_{\max(nm1,nq1)})$ generated with U $(0, 0.01)$.

Step 2. Since the number of inputs is determined as $nmı$ and $nqı$, the total number of biases and weights are $2(nm1 + nq1)$. The bias and weight values are generated with U $(0, 1)$ respectively.

Step 3. The time series with n elements is calculated by using Equation 28 with the help of Equation (16).

$$X_t = \begin{cases} \prod_{j=1}^{nmı} w_j X_{t-j} + b_j\,; & if\ \delta = 1 \\ \prod_{j=1}^{nqı} \theta_j X_{t-j} + a_j\,; & if\ \delta = 0 \end{cases} \quad (28)$$

The time series to be generated can have n observations by discarding the first $\max(nm1, nq1)$ observation values from the data set (x_t) by considering Equations (16) and (28).

Step 4. n error values corresponding to these n observations are generated by $N(0, \sigma^2)$ that means (ε_t) and the time series y_t is created by the sum of both.

As a result, a time series y_t for TS-SMNM-ANN is generated by this algorithm. The analysis results obtained with the simulated data were given in Table 5. A time series with 250 observations were generated with Algorithm 3 and the last 45 observations of this generated time series were taken as the test set. Besides, the optimal number of for both inputs so the number of bias and weight values were taken as 5.

Table 5. The RMSE and MAPE values obtained from some methods for the simulated test data

Methods	**RMSE**	**MAPE**
BP-SMNM-ANN	2.7713	3.43%
PSO-SMNM-ANN	1.9204	1.00%
ML-PSO-ANN	1.8369	1.01%
PS-ANN	1.8800	1.20%
TS-SMNM-ANN-HSA	1.8355	0.98%
TS-SMNM-ANN-PSO	1.8366	0.97%

Table 5 shows that TS-SMNM-ANN-PSO method is the best method and TS-SMNM-ANN-HSA method is the second-best method in terms of MAPE criteria for the simulated test data. TS-SMNM-ANN-HSA method is the best method and TS-SMNM-ANN-PSO method is the second-best method in terms of RMSE criteria for the simulated test data when compared with some methods in the literature.

8. Conclusions and Discussions

In this study, two different models were obtained by using a threshold value instead of a single model, and the parameters obtained as a result of the training of the SMNM-ANN were obtained by HSA and PSO methods. When the performance of the proposed methods is examined; it is seen that TS-SMNM-ANN-PSO method gives better results than TS-SMNM-ANN-HSA method. Besides, the proposed methods are proof that using two models improve forecasting performance. The proposed methods can be used when a time series has a trend or/and seasonality. In the future studies, it is planned to apply the logic of threshold to many artificial neural network models and to obtain different models by using multiple threshold values instead of a single threshold value and also work on different time-series data.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

[1] C. Worasucheep and P. Chongstitvatana, A multi-strategy differential evolution algorithm for financial prediction with single multiplicative neuron, Neural Information Processing (2009), pp. 122-130

[2] C.H. Aladag, M.A. Basaran, E. Egrioglu, U. Yolcu and V.R. Uslu, Forecasting in high order fuzzy time series by using neural networks to define fuzzy relations, Expert Systems with Applications 36 (2009), pp. 4228-4231.

[3] 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), pp. 251-262.

[4] C.H. Aladag, Using multiplicative neuron model to establish fuzzy logic relationships, Expert Systems with Applications 40(3) (2013), pp. 850-853.

[5] D. Ilter, E. Karaahmetoglu, O. Gundogdu and A.Z. Dalar, An experimental study for transforming and differencing effects in multiplicative neuron model artificial neural network for time series forecasting, 8th International Days of Statistics and Economics, Prague, Czech Republic, 2014, pp. 598-607,

[6] E. Bas, The training of multiplicative neuron model based artificial neural networks with differential evolution algorithm for forecasting, Journal of Artificial Intelligence and Soft Computing Research, 6(1) (2016), pp. 5-11.

[7] E. Bas, U. Yolcu, E. Egrioglu, O. Cagcag Yolcu and A.Z. Dalar, Single multiplicative neuron model artificial neuron network trained by bat algorithm for time series forecasting, American Journal of Intelligent Systems 6(3) (2016a), pp. 74-77.

[8] E. Bas, V.R. Uslu and E. Egrioglu, Robust learning algorithm for multiplicative neuron model artificial neural networks, Expert Systems with Applications 56 (2016b), pp. 80-88.

[9] E. Egrioglu, C.H. Aladag, U. Yolcu, B. Corba and O. Cagcag Yolcu, Fuzzy time series method based on multiplicative neuron model and membership values, American Journal of Intelligent Systems (2014), pp. 33-39.

[10] E. Egrioglu, U. Yolcu, E. Bas, and A.Z. Dalar, A hybrid forecasting method based on exponential smoothing and multiplicative neuron model artificial neural network. 3th International Researchers, Statisticians And Young Statisticians (2017), pp. 63.

[11] E. Kolay, A novel multıplıcatıve neuron model based on sıne cosıne algorıthm for tıme serıes predıctıon, Anadolu University Journal Of Science And Technology A - Applied Sciences and Engineering 20(2) (2019), pp. 153-160.

[12] E. Kolay, T. Tunc and E. Egrioglu, Classification with some artificial neural network classifiers trained a modified particle swarm optimization, American Journal of Intelligent Systems 6 (3) (2016), pp. 59-65.

[13] H. Cui, J. Feng, J. Guo and T. Wang, A novel single multiplicative neuron model trained by an improved glowworm swarm optimization algorithm for time series prediction, Knowledge 88 (2015), pp. 195-209.

[14] J. Kennedy and R.C. Eberhart, Particle swarm optimization, In Proceedings Of IEEE International Conference On Neural Networks, 1995, pp. 1942-1948.

[15] J.D.V. Henao, V.M.R. Mejia and C.J.F. Cardona, Electricity demand forecasting using a SARIMA-multiplicative single neuron hybrid model, Dyna 80(180) (2013), pp. 04-08.

[16] K. Burse, M. Manoria and V.P.S. Kirar, Improved back propagation algorithm to avoid local minima in multiplicative neuron model, International Journal of Electrical and Computer Engineering 4(12) (2010), pp. 1776-1779.

[17] K.K. Sharma, Pankaj and M. Ashish, Effect of dispersion on classification of different datasets using multiplicative neuron model, Journal of Computational and Theoretical Nanoscience (2019), pp. 4431-4437.

[18] L. Zhao and Y. Yang, PSO-based single multiplicative neuron model for time series prediction, Expert Systems with Applications 36(2) (2009), pp. 2805-2812.

[19] L.W. Lee, L.H. Wang and S.M. Chen, Temperature prediction and TAIFEX forecasting based on high-order fuzzy logical relationships and genetic algorithms, Expert Systems with Applications 33 (2007), pp. 539–550.

[20] L.W. Lee, L.H. Wang and S.M. Chen, Temperature prediction and TAIFEX forecasting based on high-order fuzzy logical relationships and genetic simulated annealing techniques, Expert Systems with Applications 34 (2008), pp. 328–336.

[21] L.Y. Hsu, S.J. Horng, T.W. Kao, Y.H. Chen, R.S. Run, R.J. Chen, J.L. Lai and I.H. Kuo, Temperature prediction and TAIFEX forecasting based on fuzzy relationships and MTPSO techniques, Expert Systems with Applications 37(4) (2010), pp. 2756-2770.

[22] O. Cagcag Yolcu and F. Alpaslan, Prediction of TAIEX based on hybrid fuzzy time series model with single optimization process, Applied Soft Computing 66 (2018), pp. 18-33.

[23] O. Cagcag Yolcu, A hybrid fuzzy time series approach based on fuzzy clustering and artificial neural network with single multiplicative neuron model, Mathematical Problems in Engineering (2013), Article ID 560472, 9 pages.

[24] O. Cagcag Yolcu, E. Bas, E. Egrioglu and U. Yolcu, Single multiplicative neuron model artificial neural network with autoregressive coefficient for time series modelling, Neural Processing Letters 47(3) (2018), pp. 1133-1147.

[25] O. Cagcag Yolcu, H. K. Lam, A combined robust fuzzy time series method for prediction of time series, Neurocomputing 247 (2017), pp. 87-101.

[26] P. K. Kandpal, and B. A. Mehta, Comparative study between multiplicative neuron and spiking neuron model, 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Ghaziabad, India, 2019, pp. 1-8.

[27] P.K. Kandpal and M. Ashish, Comparison and analysis of multiplicative neuron and multilayer perceptrons using three different datasets, International Journal of Engineering Research and General Science Volume 5 (3) (2017), pp. 58-70.

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

[29] S. Nigam, Single multiplicative neuron model in reinforcement learning. Harmony Search and Nature Inspired Optimization Algorithms. Springer, pp. 889-895, 2019.

[30] U. Yolcu, A new high order multivariate fuzzy time series forecasting model, Advances in Time Series Forecasting 2 (2017), pp. 127-143.

[31] U. Yolcu, E. Bas, E. Egrioglu and C.H. Aladag, A new multilayer feed forward network model based on trimmed mean neuron model, Neural Network World 25(6) (2015), pp. 587-602.

[32] X. Wu, J. Mao, Z. Du and Y. Chang, Online training algorithms based single multiplicative neuron model for energy consumption forecasting, Energy 59 (2013), pp. 126-132.

[33] X. Wu, Z. Zhu, X. Su, S. Fan, Z. Du, Y. Chang, and Q. Zeng, A study of single multiplicative neuron model with nonlinear filters for hourly wind speed prediction, Energy 88 (2015), pp. 194-201.

[34] Y. Basiouny, M. Arafa and A. M. Sarhan, Enhancing Wi-Fi fingerprinting for indoor positioning system using single multiplicative neuron and PCA algorithm, 12th International Conference on Computer Engineering and Systems (ICCES), (2017) Cairo, pp. 295-305.

[35] Y. Shi and R.C. Eberhart, A modified particle swarm optimizer. Proceedings of the IEEE Conference on Evolutionary Computation, (1998), pp. 69-73.

[36] Z.W. Geem, J.H. Kim and G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76(2) (2001), pp. 60-68.

[37] Z.W. Geem, Optimal cost design of water distribution networks using harmony search, Engineering Optimization 38(3) (2006), pp. 259-277.