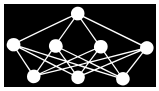

A NEW MULTILAYER FEEDFORWARD NETWORK BASED ON TRIMMED MEAN NEURON MODEL

Ufuk Yolcu[*], *Eren Bas*[†], *Erol Egrioglu*[‡], *Cagdas Hakan Aladag*[§]

Abstract: The multilayer perceptron model has been suggested as an alternative to conventional approaches, and can accurately forecast time series. Additionally, several novel artificial neural network models have been proposed as alternatives to the multilayer perceptron model, which have used (for example) the generalized-mean, geometric mean, and multiplicative neuron models. Although all of these artificial neural network models can produce successful forecasts, their aggregation functions mean that they are negatively affected by outliers. In this study, we propose a new multilayer, feed forward neural network model, which is a robust model that uses the trimmed mean neuron model. Its aggregation function does not depend on outliers. We trained this multilayer, feed forward neural network using modified particle swarm optimization. We applied the proposed method to three well-known time series, and our results suggest that it produces superior forecasts when compared with similar methods.

Key words: *neural networks, neuron model, trimmed mean, particle swarm optimization, outliers, forecast*

Received: June 14, 2013 **DOI:** 10.14311/NNW.2015.25.029
Revised and accepted: May 16, 2015

1. Introduction

Forecasting refers to the process of making an inference related to the future using available knowledge. It is important to the economy and policies of countries. Conventional forecasting models may not be adequate because they require various assumptions. Artificial neural networks (ANNs) do not require the conventional time series assumptions, and have been used extensively and successfully as an

[*]Ufuk Yolcu – Corresponding author, Ankara University, Faculty of Sciences, Department of Statistics, Dogol Street, 06100, Tandogan/Ankara, Turkey, E-mail: uyolcu@ankara.edu.tr, varyansx@hotmail.com

[†]Eren Bas, Giresun University, Faculty of Arts and Sciences, Department of Statistics, Gure Campus, 28000, Center/Giresun, Turkey, E-mail: eren.bas@giresun.edu.tr

[‡]Erol Egrioglu, Marmara University, Faculty of Arts and Sciences, Department of Statistics, Goztepe Campus, 34722, Kadikoy/Istanbul, Turkey, E-mail: erole1977@yahoo.com

[§]Cagdas Hakan Aladag, Hacettepe University, Faculty of Sciences, Department of Statistics, Beytepe Campus, 06800, Beytepe/Ankara, Turkey, E-mail: chaladag@gmail.com

alternative to conventional approaches. Some ANNs that used different neuron models were proposed in [2, 12, 15, 16, 29]. Although there are many types of ANNs for forecasting, multilayer perceptrons (MLPs) are used the most frequently. They were proposed in [18].

When solving real-life problems using a standard ANN such as an MLP, we need many neurons [22]. A neuron with higher order statistics can produce a superior neural network with comparatively less neurons [22]. Higher order neural networks were proposed in [4, 6, 8, 23, 25]. These neurons have been shown to improve the computational power and generalization. However, they are difficult to train because of a combinatorial explosion of the higher order term as the number of inputs to the neuron increases [22]. Additionally, outliers negatively affect MLPs, as shown in [7, 30]. There are various ANN models such as the multilayer feed forward model (MFF), which is based on the generalized-mean neuron model (GMN, [28]), geometric mean neuron model (G-MN, [22]), and the single multiplicative neuron model (SMN, [27]).

There are also many other studies that produced good results in the presence of outliers. In [9], an M-estimator as an evaluation function for training an ANN was used. Robust training algorithms based on least median squares were proposed in [5, 19]. A robust, feed forward, back propagation algorithm was developed for an ANN in [24]. In [26], an approach based on the robust adaptive training of feed forward neural networks was presented. New robust forecasting models for ANNs were proposed in [14] and a robust learning algorithm was developed in [20]. In [1], a robust ANN based on the median neuron model was proposed. Moreover, a robust, feed forward, and recurrent neural network model was presented in [17].

The aggregation functions of most of these models are negatively affected by outliers because they are based on the mean. This means that the predictions do not represent the general characteristic features of the data and that the predictions tend to outliers. Although, the proposed ANN methods in [1] that is based on the median neuron model provides to avoid this problem, because each of neurons uses fifty percent of its inputs, there are an information loss in their ANN model. This situation can affect the performance of model, negatively.

In this study, we propose a new, multilayer, feed forward neural network called the multilayer feed forward network with trimmed mean neuron model (TMNM-MFF). In our method, the aggregation function is not affected by outliers. Using TMNM, an outlier input does not cause an outlier output. On the contrary median neuron model, TMNM uses the trimmed mean of values that are generated by multiplication inputs and weights. Therefore TMNM benefits from most of inputs information. We used modified particle swarm optimization (MPSO) to train the model.

The remainder of this paper is organized as follows. In Section 2, we briefly summarize the MPSO method that was used to train the TMNM-MFF. We introduce TMNM in Section 3. Section 4 contains a description of the TMNM-MFF architecture and its training algorithm. In Section 5, we give the results of applying the TMNM-MFF method to three well-known time series. Finally, we discuss our conclusions in Section 6.

2. Modified particle swarm optimization algorithm

Particle swarm optimization (PSO) is a heuristic algorithm that was first proposed in [11]. Its distinguishing feature is that it simultaneously examines different points in different regions of the solution space to find the global optimum solution. This feature means that it avoids local optimum traps. In this study, we used the modified PSO method to train the TMNM-MFF. This MPSO algorithm has a time-varying inertia weight, as in [21]. In a similar way, the algorithm also has a time-varying acceleration coefficient, as in [13].

Algorithm 1. Modified particle swarm optimization.

Step 1. Randomly determine the positions of each k-th, $(k = 1, 2, \ldots, pn)$ particle, and store them in vector

$$\mathbf{x}_k = \left\{x_1^k, x_2^k, \cdots, x_d^k\right\}, k = 1, 2, \cdots, pn, \tag{1}$$

where $x_i^k, i = 1, 2, \cdots, d$ represents the i-th position of the k-th particle, pn is the number of particles in a swarm, and d is the position. And also determine the maximum number of iterations $(t_{\max})$.

Step 2. Randomly determine the velocities, and store them in a vector

$$\mathbf{v}_k = \left\{v_1^k, v_2^k, \cdots, v_d^k\right\}, k = 1, 2, \cdots, pn. \tag{2}$$

Step 3. According to the evaluation function, determine the P^{best} and G^{best} particles using

$$P_k^{\text{best}} = \left(p_1^k, p_2^k, \cdots, p_d^k\right), k = 1, 2, \cdots, pn \tag{3}$$

and

$$G^{\text{best}} = \left(pg_1, pg_2, \cdots, pg_d\right), \tag{4}$$

where ${P^{\text{best}}}_k$ stores the positions corresponding to the k-th particle's best individual performance, and G^{best} represents the best particle so far, according to the evaluation function.

Step 4. Let c_1 and c_2 represent the cognitive and social coefficients, respectively, and w be the inertia parameter. Let $[c_{1f}, c_{1i}]$, $[c_{2i}, c_{2f}]$ and $[w_1, w_2]$ be intervals that include possible values for c_1, c_2, and w, respectively. In addition, c_{1i} and $c_{1f}\,(c_{1i} > c_{1f})$ are initial and final values of cognitive coefficient, c_{2i} and $c_{2f}\,(c_{2i} < c_{2f})$ are initial and final values of social coefficient. For each iteration, calculate these parameters using

$$c_1 = (c_{1f} - c_{1i})\,\frac{t}{t_{\max}} + c_{1i}, \tag{5}$$

$$c_2 = (c_{2f} - c_{2i})\,\frac{t}{t_{\max}} + c_{2i}, \tag{6}$$

and

$$w = (w_2 - w_1)\,\frac{t_{\max} - t}{t_{\max}} + w_1, \tag{7}$$

where $t_{\max}$ is the maximum number of iterations, and t is the current iteration.

Step 5. Update the velocities and positions using

$$v_{i,d}^{t+1} = \left[w \times v_{i,d}^{t} + c_1 \times \text{rand}_1 \times (p_{i,d} - x_{i,d}) + c_2 \times \text{rand}_2 \times (p_{g,d} - x_{i,d})\right] \quad (8)$$

and

$$x_{i,d}^{t+1} = x_{i,d} + v_{i,d}^{t+1}, \quad (9)$$

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

Step 6. Repeat Steps 3–5 until reaching a predetermined maximum number of iterations ($t_{\max}$).

3. Trimmed mean neuron model (TMNM)

The first ANN model was proposed in [15]. Many different neuron models have been suggested since then. Additionally, the MLP neurons can have different aggregation functions such as

$$\text{net}\,(x_j, w_j) = \left[\sum_{j=1}^{N} w_j x_j + w_0\right], \quad (10)$$

where N is the number of input signals, $x_j\,(j = 1, 2, \ldots, N)$ are the input signals, and $w_j\,(j = 0, 1, 2, \ldots, N)$ are the weights. The neuron model represented by Eq. (10) is affected by input signals that have outliers, because its aggregation function is based on addition. Moreover, although many different models have been proposed (such as GMN and G-MN), they are affected by outliers because their aggregation functions are based on the mean. This means that the forecasts do not agree with the general structure of the data and tend towards outliers. The objective of this study was to overcome this problem. For this purpose, we proposed a TMNM aggregation function, which is based on the trimmed mean *(trimmean)* and is not affected by outliers. When training the ANN, some of the smallest and largest values of $w_j x_j$ and w_0 (the input signals and weights) are ignored at a certain rate. We determine which weighted input signals should be ignored using

$$k = (N + 1)\,\frac{\text{percent}}{2}, \quad (11)$$

where N represents the number of input signals, and percent is a user-defined rate. Suppose that we have data with 19 observations (input signals), and, for example, take percent to be 10/100. Then,

$$k = 20\frac{\left(10/100\right)}{2} = 1. \quad (12)$$

This means that we ignore the smallest and largest of the weighted input signals. The value of k is rounded to the nearest integer, if necessary. The TMNM model with these properties is shown in Fig. 1.

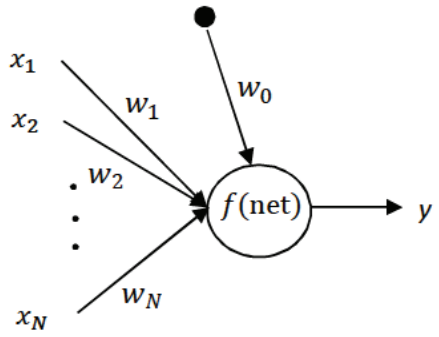

Fig. 1 *Trimmed mean neuron model.*

In Fig. 1, y is the output signal and f is the activation function. The aggregation function value (net) is calculated using

$$\text{net} = \text{Trimmean}\,(w_1x_1w_2x_2, \ldots, w_Nx_N, w_0). \tag{13}$$

Therefore, outlier inputs do not affect the aggregation function.

4. Multilayer feed forward network with trimmed mean neuron model (TMNM-MFF)

The TMNM-MFF proposed in this study is a feed forward ANN that uses TMNM. Its architecture for N inputs and M neurons in the hidden layer is given in Fig. 2.

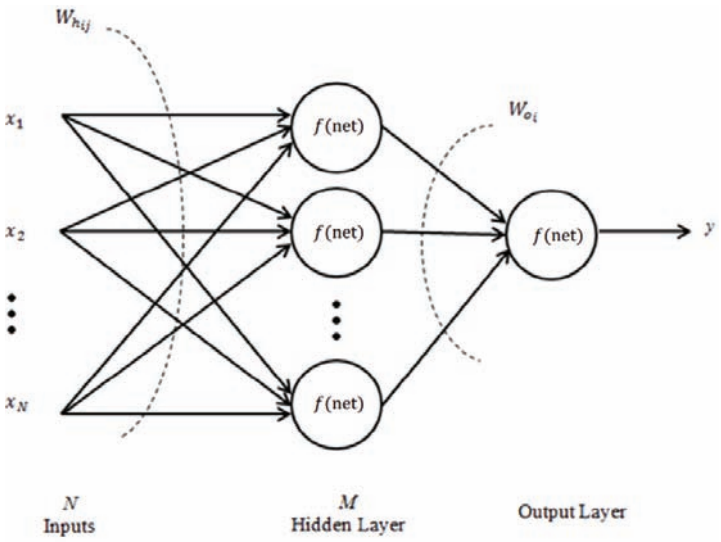

Fig. 2 *The TMNM-MFF architecture.*

In Fig. 2, the input is $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ and the output is $[y]$. If $W_{h_{ij}}$ is a weight that connects the i-th hidden neuron with the j-th input and $w_{h_{i0}}$ is the bias of the i-th hidden neuron, the activation value of the i-th hidden neuron is

$$\text{net}_{h_i} = \text{Trimmean}\left(w_{h_{i1}}x_1, w_{h_{i2}}x_2, \ldots, w_{h_{iN}}x_N, w_{h_{i0}}\right), i = 1, 2, \ldots, M \tag{14}$$

The nonlinear transformation performed by each of the M neurons is

$$y_{h_i} = f\left(\text{net}_{h_i}\right), i = 1, 2, \ldots, M, \tag{15}$$

where f denotes a sigmoid function, w_{o_i} is the weight that connects the i-th neuron of the hidden layer to a neuron in the output layer, and w_{O_0} is the bias of the corresponding output layer neuron. Similarly, the output of the neuron in the output layer is determined by

$$\text{net} = \text{Trimmean}\left(w_{o_1}y_{h_1}w_{o_2}y_{h_2}, \ldots, w_{o_M}y_{h_M}, w_{O_0}\right), \tag{16}$$

and

$$y = f\left(\text{net}\right) \tag{17}$$

We trained the TMNM-MFF using MPSO. The positions of a particle in MPSO are the TMNM-MFF weights. The structure of a particle is given in Fig. 3.

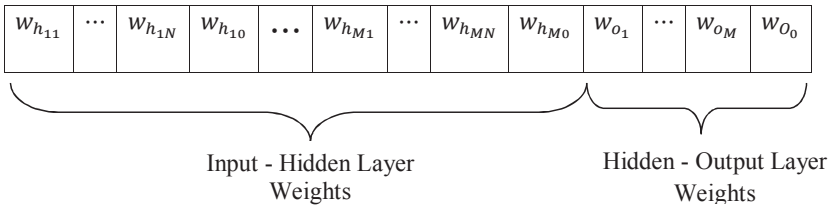

Fig. 3 *The structure of an MPSO particle used to train the TMNM-MFF.*

In the MPSO algorithm, we used the mean square error (MSE) criterion as the evaluation function, which is defined as

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}\left(\text{output}_t - \text{target}_t\right)^2, \tag{18}$$

where T is the number of training samples.

Algorithm 2. The TMNM-MFF algorithm.

Step 1. Determine the PSO parameters and the maximum number of iterations ($t_{\max}$).

We calculate the parameters that direct the modified PSO algorithm (pn, vm, c_{1i}, c_{1f}, c_{2i}, c_{2f}, w_1, and w_2) as described previously.

In the implementations of this study, these values were taken as $pn = 30$, $vm = 1$, $c_{1i} = 3$, $c_{1f} = 2$, $c_{2i} = 2$, $c_{2f} = 3$, $w_1 = 0.4$, $w_2 = 0.8$, and the maximum number of iterations was determined as 100.

Step 2. Determine the initial positions and velocities.

Randomly generate the initial positions (uniformly distributed between (0,1)) and velocities (uniformly distributed between $(-vm, vm)$) of each particle in the swarm.

In order to avoid too large step sizes, velocities can be limited. Different strategies to initialize velocities can be identified in the literature such as zero velocities, uniform random values within the domain of the optimization problem, and small uniform random values. We prefer to use initial velocities sampled from a uniform distribution within the domain of the problem. However, the best clamping threshold referred to as vm, is problem dependent. In the implementations of this study, vm was taken as 1.

Step 3. Compute the values of the evaluation function.

We used the MSE given in Eq. (18) as the evaluation function.

Step 4. Determine ${P^{\text{best}}}_k, (k = 1, 2, \cdots, p)$ and G^{best} according to the evaluation function values from the previous step.

${P^{\text{best}}}_k$ is a vector that contains the positions corresponding to the k-th particle's best individual performance, and G^{best} is the best particle (i.e., it has the best evaluation function value) found to date.

Step 5. Update the parameters.

The updated values of the cognitive coefficient c_1, the social coefficient c_2, and the inertia parameter w are calculated using Eqs. (5)–(7).

Step 6. Calculate the new positions and velocities.

We compute the new velocities and positions using Eqs. (8) and (9) respectively. If we have not reached the maximum number of iterations, go to Step 3; otherwise, go to Step 7.

Step 7. Determine the optimal solution.

The elements of G^{best} are taken as the optimal weighting values of the TMNM-MFF.

5. Applications of TMNM-MFF

We used three well-known, real-time series to analyze the performance of the TMNM-MFF. These time series contain monthly sulfur dioxide measurements from the Ankara capitol of Turkey (ANSO), Box-Jenkins gas furnace data [3], and Australian beer consumption data. The inputs contain lagged variables of related time series, which are used to analyze TMNM-MFF and other ANNs.

To compare the forecasts, we used the root mean square error (RMSE), mean absolute percentage error (MAPE), and median absolute percentage error (MdAPE).

They are defined as

$$\mathrm{RMSE} = \left(\frac{\sum_{t=1}^{T} (y_t - \hat{y}_t)^2}{T} \right)^{1/2}, \tag{19}$$

$$\mathrm{MAPE} = \frac{1}{T} \sum_{t=1}^{T} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \times 100, \tag{20}$$

and

$$\mathrm{MdAPE} = \mathrm{Median} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \times 100, \tag{21}$$

where y_t is the actual value, $\hat{y}_t$ is the predicted value, and T is the number of data.

5.1 Monthly sulfur dioxide measurements for Ankara

The first data set contains measurements of the amount of sulfur dioxide (SO_2) in the air. The data were recorded in Ankara, Turkey, between 1994 and 2006 (see Fig. 4). The last 10 observations of the time series were used for testing, as in previous studies. In this application, we carried out two different analyses using the original time series and the time series with outliers obtained by replacing an original observation with values 5 and 10 times the maximum observation in the data. The aim of this experiment was to show that TMNM-MFF produces superior forecasts regardless if the data contain outliers.

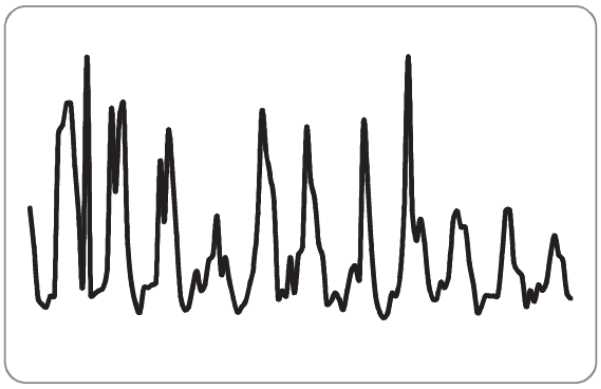

Fig. 4 *Time series data containing the amount of SO_2 in Ankara.*

First, we forecasted the original ANSO data using the seasonal autoregressive integrated moving average method (SARIMA), Winter's multiplicative exponential smoothing method (WMES), and a MLP. We compared the results with those from the proposed TMNM-MFF method. For both the ANN models, we varied the number of neurons in the input layer between 1 and 12, because the period of the time series was 12. Additionally, we did not let the number of neurons in the hidden layer exceed the number of inputs. In other words, we considered 144

different architectures for both of the ANN models. For all these architectures, the *RMSE* values and some statistics are given in Tabs. I and II. These results show that the results obtained from the TMNM-MFF are consistent.

Second, we analyzed the differences in the results obtained from the best MLP and TMNM-MFF methods, for the data with outliers. The results are summarized in Tab. IV.

The results in Tabs. III and IV demonstrate that, although the MLP and TMNM-MFF methods produced similar results when there were no outliers, the TMNM-MFF produced superior results for both the training and test sets when there were outliers.

5.2 Box-Jenkins gas furnace data

We also analyzed the Box-Jenkins gas furnace data [3]. In this data set, $x(t)$ is the gas flow rate and $y(t)$ is the CO_2 concentration. We modeled the furnace output as a function of the output $y(t-1)$ and input $x(t-4)$. We trained the models using 146 samples, and used 150 samples as the test set.There were five neurons in the hidden layer. The results obtained from TMNM-MFF and some different ANN models are given in Table V. The back propagation single multiplicative neuron model (BP-SMN) and MLP ($2\times2\times1$) results were taken from [21], and the results of the particle swarm optimization-single multiplicative neuron model (PSO-SMN), cooperative random learning particle swarm optimization-single multiplicative neuron model (CRPSO-SMN), and genetic algorithm-single multiplicative neuron model (GA-SMN) were taken from [31].

Fig. 5 plots the real observations and forecasts obtained from the TMNM-MFF method. According to this graph, the forecasts obtained from the proposed approach are very accurate.

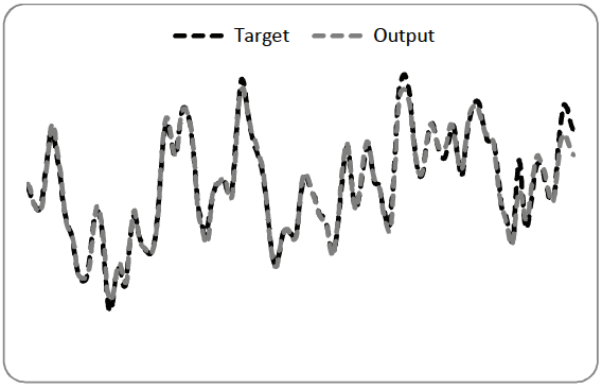

Fig. 5 *Forecasts of the proposed TMNM-MFF for the Box-Jenkins gas furnace data.*

Our results suggest that the proposed method produces better forecasts (in the absence of outliers) than the SMNs trained by various algorithms and the MLP. According to Fig. 5, our forecasts are a good match to the real observations.

		NHLN											
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
NILN	**1**	6.81	6.81	17.26	7.37	7.00	6.66	6.93	6.90	7.31	7.00	6.31	6.74
	2	7.84	17.26	9.10	8.07	17.26	7.54	9.32	7.50	11.38	17.26	7.77	13.72
	3	17.26	11.74	11.99	10.77	12.56	12.29	17.26	8.72	13.39	25.08	11.48	23.06
	4	17.26	11.79	7.56	14.25	19.19	11.01	17.26	21.84	20.73	17.26	20.54	18.59
	5	7.21	9.02	13.27	11.63	25.95	15.98	26.62	17.26	11.69	43.15	17.26	35.75
	6	7.24	16.35	17.26	16.08	17.26	23.04	40.74	23.48	22.49	23.29	25.85	34.50
	7	8.17	8.23	17.26	17.22	27.51	22.23	26.42	23.42	30.79	17.26	23.87	17.26
	8	6.41	11.86	8.18	8.91	20.58	23.89	19.48	16.66	17.26	17.26	30.38	35.61
	9	6.19	17.26	14.89	14.15	7.73	25.08	10.44	10.22	16.36	18.42	21.46	22.09
	10	5.52	8.64	8.33	24.60	15.13	6.14	11.96	14.36	16.62	34.50	17.26	27.50
	11	3.93	5.13	6.46	4.70	7.47	20.50	29.79	13.60	18.38	21.13	17.88	12.51
	12	3.74	8.91	17.26	14.71	17.76	27.82	39.26	11.96	18.32	12.40	16.18	14.75
Min		3.7402											
Max		43.1473											
Mean		15.9573											
Std. Dev.		8.1163											

Tab. I *RMSE values and some statistics for the MLP method with different architectures, for the ANSO test data.* ***NILN****: Number of input layer neurons.* ***NHLN****: Number of hidden layer neurons.*

NIILN \ NHLN	1	2	3	4	5	6	7	8	9	10	11	12
1	7.74	7.88	7.83	7.73	8.00	7.75	7.28	7.68	7.44	7.58	7.58	7.48
2	7.57	7.42	7.18	8.52	8.38	8.02	8.23	7.59	8.06	8.04	7.99	7.73
3	7.75	8.80	7.60	7.72	8.29	7.95	7.30	7.52	7.72	7.31	7.38	7.99
4	8.22	8.40	8.91	8.55	8.23	7.99	8.14	7.78	8.80	8.23	7.94	8.75
5	10.77	9.26	9.00	7.99	9.76	9.75	10.64	10.00	8.17	10.57	10.45	9.12
6	9.18	10.01	10.44	9.55	8.65	8.41	8.15	10.18	10.51	9.74	9.90	9.30
7	11.40	10.80	10.97	9.94	10.53	10.07	9.71	9.97	8.70	9.79	9.54	10.05
8	10.31	12.15	9.17	9.64	9.68	10.11	10.38	8.45	9.44	10.97	9.28	9.11
9	10.51	9.17	8.19	8.96	9.70	9.21	9.84	9.05	8.87	8.51	8.76	8.91
10	7.75	7.50	7.83	7.28	7.75	8.29	7.99	8.12	9.25	7.86	8.05	8.91
11	4.85	10.33	9.98	5.87	6.69	6.21	5.75	5.33	6.62	5.45	6.93	6.98
12	6.26	7.86	6.09	6.32	3.63	6.07	7.93	6.30	6.22	4.92	5.62	5.01
Min	3.6298											
Max	12.1504											
Mean	8.3965											
Std. Dev.	1.4777											

Tab. II *RMSE values and some statistics for the TMNM-MFF method with different architectures, for the ANSO test data.* ***NIILN****: Number of input layer neurons.* ***NHLN****: Number of hidden layer neurons.*

	SARIMA (1,1,0)(0,1,1)	WMES (Trend-Seasonal)	MLP Best Architecture: (12-1-1)	TMNM-MFF TMNM-MFF Best Architecture: (12-5-1)
RMSE	9.6249	7.1062	3.7402	3.6298
MAPE	0.2336	0.2204	0.0995	0.1148
MdAPE	0.1931	0.2478	0.0898	0.1075

Tab. III *Results obtained for the ANSO test data using other methods.*

		MLP Best Architecture: 12-3-1		TMNM-MFF Best Architecture: 12-5-1	
		Train	Test	Train	Test
Five multiple outlier	**RMSE**	56.0856	17.2534	19.3009	11.3851
Ten multiple outlier	**RMSE**	28.9453	17.2549	19.4028	13.5803

Tab. IV *Results of the best methods for the data with outliers.*

		BP-SMN	PSO-SMN	CRPSO-SMN	GA-SMN	MLP	TMNM-MFF
RMSE	Train	0.0400	0.0400	0.0400	0.0400	0.0906	0.0214
	Test	0.0424	0.0436	0.0424	0.0424	0.1503	0.0392

Tab. V *Performance comparison of different methods applied to the Box-Jenkins gas furnace data.*

We also applied the method to a dataset with outliers. First, we replaced the 10-th observation with a value 10 times the size of the maximum. The *RMSE* values obtained for this new time series are given in Tab. VI. It is clear that the MLP was negatively affected by this outlier, and that TMNM-MFF produced superior forecasts.

		MLP	TMNM-MFF
RMSE	Train	0.2706	0.0814
	Test	0.2612	0.0051

Tab. VI *Performance comparison for the Box-Jenkins gas furnace data with an outlier.*

5.3 Australian beer consumption data

Finally, we applied the proposed method to Australian beer consumption data [10]. The time series contains 148 quarterly observations from 1956 to 1994. The first 132 observations were used for training, and the last 16 observations were used

for testing. We compared the proposed method with the WMES, SARIMA, radial basis neural network (R-ANN), MLP, PSO-SMN, Elman neural network (E-ANN), and multiplicative seasonal artificial neural network (MS-ANN) techniques in terms of RMSE criteria for test data. The results are given in Tab. VII.

	WMES	SARIMA	R-ANN	PSO-SMN	MLP	E-ANN	MS-ANN	TMNM-MFF
RMSE	53.3295	47.0367	41.7000	26.7831	24.1052	22.6581	22.1700	21.0623

Tab. VII *Performance comparison of different techniques applied to the Australian beer consumption for test data.*

Fig. 6 plots the real observations and forecasts obtained from the TMNM-MFF. According to this graph, the forecasts obtained from the proposed approach are very accurate.

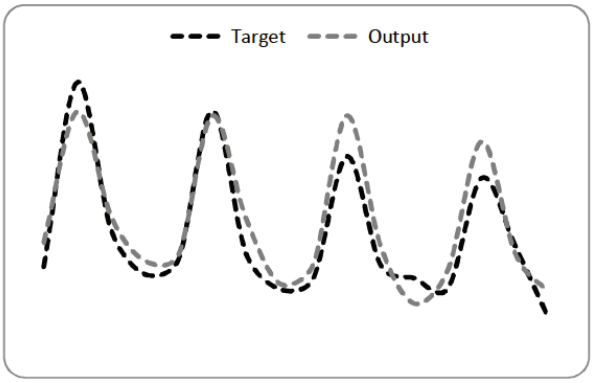

Fig. 6 *Forecasts of the TMNM-MFF for the Australian beer consumption data.*

We also analyzed this dataset after adding an outlier, as with the other datasets. We replaced the 10-th observation with a value 10 times the maximum. The *RMSE* values for this new time series are given in Tab. VIII. It is clear that the MLP was negatively affected by the outlier, and that the TMNM-MFF produced superior forecasts. Although MLP performs better than TMNM-MFF in case of training, this is an expected result because the outlier was injected the training set of data. This situation shows that the predictions of TMNM-MFF for training set approached the outlier less than MLP. That is, TMNM-MFF was less affected by outlier than MLP and the obtained results for test set of data support this phenomenon.

6. Conclusions

Many ANN models have been proposed as an alternative to the MLP, and there are many different neuron models. Although all these ANNs can successfully forecast time series, they are negatively affected by outliers because their aggregation

		MLP	TMNM-MFF
RMSE	Train	493.1830	503.7322
	Test	60.5499	56.1878

Tab. VIII *Performance comparison for the Australian beer consumption data with an outlier.*

functions are the same as MLPs.

In this study, we proposed a new neuron model that uses TMNM as an aggregation function, and developed the TMNM-MFF based on this new neuron model. TMNM is insensitive to outliers in the inputs and prevents outliers in the outputs. We used MPSO to train various ANN models, and applied the proposed model to three real-time series.

Our results suggest that TMNM-MFF is more effective than SMNs with different training algorithms and MLPs. This is particularly true when the data contain outliers.

The obtained findings from this study can be utilized in the future studies, from different viewpoints. On the one hand, the proposed trimmed mean neuron model can be used in the other type of ANNs such as SMN, E-ANN and R-ANN to search the performance of them, on the other side, different artificial intelligent optimization techniques such as genetic algorithm, artificial bee colony algorithm and differential evaluation algorithm can be used in the training of ANNs.

References

[1] ALADAG C.H., YOLCU U., EGRIOGLU E. Robust multilayer neural network based on median neuron model. *Neural Computing and Applications*. 2014, 24, pp. 945−956, doi: 10.1007/s00521-012-1315-5.

[2] BASU M., HO T.K. Learning behavior of single neuron classifiers nonlinearly separable or non-separable inputs. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN '99)*, vol.2, Washington DC, USA. IEEE, 1999, pp. 1259−1264, doi: 10.1109/IJCNN.1999.831142.

[3] BOX G.E.P., JENKINS G.M. *Time series analysis: Forecasting and Control*. San Francisco: Holden-Day, 1976.

[4] CHATURVEDI D.K., MOHAN M., SINGH R.K., KALRA P.K. Improved generalized neuron model for short-term load forecasting. *Soft Computing*. 2003, 8(1), pp. 10–18, doi: 10.1007/s00500-002-0241-3.

[5] EL-MELEGY M.T., ESSAI M.H., ALI A.A. Foundations of Computational Intelligence vol.1, Learning and Approximation. In: A.-E. HASSANIEN, A. ABRAHAM, A.V. VASILAKOS, W. PEDRYCZ, eds. *Robust training of artificial feed forward neural networks*. Volume 201 of the series Studies in Computational Intelligence (series title). Berlin, Heidelberg: Springer, 2009, pp. 217−242, doi: 10.1007/978-3-642-01082-8_9.

[6] GILES C.L., MAXWELL T. Learning invariance and generalization in higher order networks. *Applied Optics*. 1987, 26, pp. 4972–4978, doi: 10.1364/AO.26.004972.

[7] HILL T., MARQUEZ L., O'CONNOR M., REMUS W. Artificial neural networks for forecasting and decision making. *International Journal of Forecasting*. 1994, 10(1), pp. 5–15, doi: 10.1016/0169-2070(94)90045-0.

[8] HOMMA N., GUPTA M.M. Study on general second-order neural units (SONUs). In: *Proceedings of the 5th Biannual World on Automation Congress*. IEEE, 13, 2002, pp. 177–182, doi: 10.1109/WAC.2002.1049541.

[9] HSIAO C.C., CHUANG C.C., JENG J.T. Robust back propagation learning algorithm based on near sets. In: *International Conference on System Science and Engineering (ICSSE)*, Dalian, Liaoning, China. IEEE, 2012, pp. 19–23.

[10] JANACEK G. *Practical Time Series*. New York: Oxford University Press Inc., 2001.

[11] KENNEDY J., EBERHART R. Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks*, Perth, WA, Australia. IEEE, 4, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[12] LABIB R. New single neuron structure for solving non-linear problems. *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'99)*, Washington DC, USA. IEEE, 1, 1999, pp. 617–620, doi: 10.1109/IJCNN.1999.831569.

[13] MA Y., JIANG C., HOU Z., WANG C. The formulation of the optimal strategies for the electricity producers based on the particle swarm optimization algorithm. *IEEE Transactions on Power Systems*. 2006, 21(4), pp. 1663–1671, doi: 10.1109/TPWRS.2006.883676.

[14] MAJHI B., ROUT M., MAJHI R., PANDA G., FLEMING P.J. New robust forecasting models for exchange rates prediction. *Expert Systems with Applications*. 2012, 39(16), pp. 12658–12670, doi: 10.1016/j.eswa.2012.05.017.

[15] McCULLOCH W.S., PITTS W.A. Logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*. 1943, 5, pp. 115–133.

[16] PLATE T.A. Randomly connected sigma–pi neurons can form associator networks. *Network: Computation in Neural Systems*. 2000, 11(4), pp. 321–332, doi: 10.1088/0954-898X_11_4_305.

[17] ROY P., MAHAPATRA G.S., POOJA R., PANDEY S.K., DEY K.N. Robust feed forward and recurrent neural network based dynamic weighted combination models for software reliability prediction. *Applied Soft Computing*. 2014, 22, pp. 629–637, doi: 10.1016/j.asoc.2014.04.012.

[18] RUMELHART E., HINTON G.E., WILLIAMS R.J. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. In: J.A. FELDMAN, P.J. HAYES, D.E. RUMELHART, eds. *Learning Internal Representations by Error Propagation*. Vol. 1: Foundations. Cambridge MA: MIT Press, 1986, pp. 348–352.

[19] RUSIECKI A. Robust learning algorithm based on iterative least median squares. *Neural Processing Letters*. 2012, 36(2), pp. 145–160, doi: 10.1007/s11063-012-9227-z.

[20] RUSIECKI A. Robust learning algorithm based on LTA estimator. *Neurocomputing*. 2013, 120, pp. 624–632, doi: 10.1016/j.neucom.2013.04.008.

[21] SHI Y., EBERHART R.C. Empirical study of particle swarm optimization. In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA. IEEE, 1999, 3, pp. 101–106, doi: 10.1109/CEC.1999.785511.

[22] SHIBBLEE M.D., CHANDRA B., KALRA P.K. Learning of geometric mean neuron model using resilient propagation algorithm. *Expert Systems with Applications*. 2010, 37(12), pp. 7449–7455, doi: 10.1016/j.eswa.2010.04.018.

[23] SINHA M., KUMAR K., KALRA P.K. Some new neural network architectures with improved learning schemes. *Soft Computing*. 2000, 4(4), pp. 214–223, doi: 10.1007/s005000000057.

[24] SRINIVAS Y., RAJ S. A., OLIVER H.D., MUTHURAJ D., CHANDRASEKAR N. A robust behavior of feed forward back propagation algorithm of artificial neural networks in the application of vertical electrical sounding data inversion. *Geoscience Frontiers*. 2012, 3(5), pp. 729–736, doi: 10.1016/j.gsf.2012.02.003.

[25] TAYLOR J.G., COMMBES S. Learning higher order correlations. *Neural Networks*. 1993, 6(3), pp. 423–428, doi: 10.1016/0893-6080(93)90009-L.

[26] XINGJIAN J. Robust adaptive learning of feed forward neural networks via LMI optimizations. *Neural Networks*. 2012, 31, pp. 33–45, doi: 10.1016/j.neunet.2012.03.003.

[27] YADAV R.N., KALRA P.K., JOHN J. Time series prediction with single multiplicative neuron model. *Applied Soft Computing*. 2007, 7(4), pp. 1157–1163, doi: 10.1016/j.asoc.2006.01.003.

[28] YADAV R.N., KUMAR N., KALRA P.K., JOHN J. Learning with generalized-mean neuron model. *Neurocomputing*. 2006, 69(16–18), pp. 2026–2032, doi: 10.1016/j.neucom.2005.10.006.

[29] ZHANG C.N., ZHAO M., WANG M. Logic operations based on single neuron rational model. *IEEE Transactions on Neural Networks*. 2000, 11(3), pp. 739–747, doi: 10.1109/72.846745.

[30] ZHANG G., EDDY PATUWO B., HU Y.M. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*. 1998, 14(1), pp. 35–62, doi: 10.1016/S0169-2070(97)00044-7.

[31] ZHAO T.S., YANG W.W. Encyclopedia of Electrochemical Power Sources. In: J. GARCHE, ed. *Fuel Cells - Direct Alcohol Fuel Cells Modeling*. Vol 1, Elsevier Science, 2009, pp. 436–445.