Rabieyan, Reza; Pohl, Philipp

Article — Published Version

Improving a fuzzy neural network for predicting storage usage and calculating customer value

Journal of Revenue and Pricing Management

Provided in Cooperation with:
Springer Nature

Suggested Citation: Rabieyan, Reza; Pohl, Philipp (2020) : Improving a fuzzy neural network for predicting storage usage and calculating customer value, Journal of Revenue and Pricing Management, ISSN 1477-657X, Palgrave Macmillan UK, London, Vol. 19, Iss. 5, pp. 292-301, https://doi.org/10.1057/s41272-020-00253-3

This Version is available at:
https://hdl.handle.net/10419/288629

Journal of Revenue and Pricing Management (2020) 19:292–301
https://doi.org/10.1057/s41272-020-00253-3

RESEARCH ARTICLE

Improving a fuzzy neural network for predicting storage usage and calculating customer value

Reza Rabieyan[1] · **Philipp Pohl**[1]

Received: 8 March 2020 / Accepted: 22 May 2020 / Published online: 2 July 2020

Abstract
Predicting the behavior of customers plays a crucial role in the quality of resource management and customer services. In this article, a fuzzy neural network model for predicting the customer storage usage is identified. The identified fuzzy neural network is improved and finally the result of the improved fuzzy neural network is compared with some other fuzzy neural network and other prediction methods.

Keywords Machine learning · Fuzzy neural networks · Resource management · Storage usage · Adaptive second-order algorithm

Introduction

According to the report of the 12th annual Cisco Visual Networking Index Complete Forecast,[1] the number of internet users will increase from 3.3 billion to 4.6 billion by 2021. From now to 2021 the improvement equates to 61% of the global population using the internet.

Easy access has led to an ever-increasing use of the internet. Today, more and more information is uploaded from websites, and users download this information. Therefore, web hosting customers request more from web host providers. The inevitable result is that the controllability and manageability of the web host resources will be enhanced.

Web hosting companies provide service for individuals and organizations to create their websites. Web host companies provide space on a server so that the customer can upload the files of their website such as documents, videos, music, and images. The amount of this space is called storage usage. Web hosting providers have thousands of servers and millions of customers, and predicting the customer storage usage plays an important role in the quality of their service, customers satisfaction, energy saving, and maintenance management. Another great advantage of this prediction methodology is that the web hosting company can predict the achievable future cash flows of customers and it helps them to determine and analyze customer's portfolio, by calculating customer value. Customer value is defined as sum of customers' discounted future cash flows.

The first step for predicting the storage usage of customers applies Poisson distribution theory which approximate to the Markov process. For instance, the forecasting model based on the autoregressive method is proposed by Barba and Rodríguez (2015). In reference to the autocorrelation properties of customer storage usage, applying the Poisson process is not a reliable method (Iliev and Bedzhev 2015; Morikawa and Tsuneda 2014). The linear model such as the Markov-modulated Poisson process and the moving average model are applied efficiently for short-term forecasting (Mai et al. 2014; Borchers and Langrock 2015) but by increasing the forecasting step the forecasting error will increase slowly (Hou et al. 2018). In fact, because of the strong non-linear behavior of web hosting customers, the customer storage usage model is a non-linear system and applying the classical prediction methods is not a reliable strategy for our purpose.

One of the most popular tools for predicting the non-linear and time varied behavior is the artificial neural network. The artificial neural network has some processing functions such as learning, memorizing, and computing,

✉ Reza Rabieyan
r.rabieyan@gmail.com

Philipp Pohl
philipp.pohl@dhbw-karlsruhe.de

[1] Baden-Wuerttemberg Cooperative State University, Karlsruhe, Germany

[1] https://it-online.co.za/2017/06/09/the-growth-and-growth-of-internet-usage/.

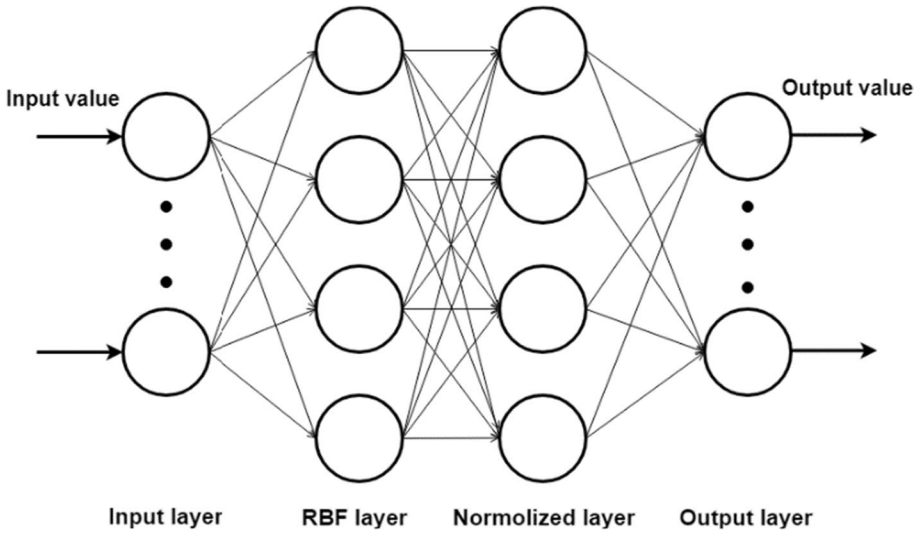


Fig. 1 The structure of fuzzy neural network (Han et al. 2016)

which make the artificial neural network a strong tool for solving non-linear problems (Hou et al. 2018). The other advantages of artificial neural network are memorizing the signal in a non-linear way, distributing processing and adapting abilities (Moretti et al. 2015).

The most important difference of previous research, including this research, is the complexity of the behavior of customers. On the one hand, in this problem the customers have three different types of behavior:

1. Customers have a positive storage usage (they upload data on server)
2. Customers have a negative storage usage (they delete the uploaded data on server)
3. Customers do nothing.

On the other hand, the variation of storage usage between customers is very large. The minimum storage usage is 4 KB and the maximum is 12.7 GB.

The purpose of this article is to apply fuzzy artificial neural network (FNN) for identifying a model that is optimal for predicting the absolute customer storage usage for each server. Therefor different training algorithms are compared and improved. Finally, it will be shown that improved algorithm is significantly better than the other algorithms.

This article is structured as follows: The FNN architecture is described in "The fuzzy neural network architecture" section. "Learning algorithm" section discusses learning algorithms. The results of experiments are presented in "Results of the experiment" section. "Significance test" section contains the applied significance test and finally in "Conclusion" section the conclusion is presented.

The fuzzy neural network architecture

The radial basis function (RBF) neural network is one of the eminently appropriate methods which can model the non-linear systems and is applied in different types of research studies. In the following paragraphs, a brief history of the application of RBF will be presented.

In 2012, Fei and Ding proposed a new adaptive RBF neural network to control dynamic systems. The proposed adaptive RBF neural network is applied to train the upper bound of model uncertainties and external disturbances. Finally, the results of the experiment illustrate the stability of the closed-loop system (Fei and Ding 2012).

Huang and Ben-Hsiang (2014) presented a motion detection approach based on the RBF artificial neural networks to divide moving objects in a dynamic sense. The final evaluation results show that the recommended method succeeds in complete and accurate detection in both static and dynamic scenes (Hagan and Menhaj 1994).

Han et al. (2016) applied RBF neural network for non-linear system modeling. In contrast to the previously mentioned methods, a prediction approach, which is based on the RBF artificial neuron network, is applied to predict the wastewater treatment process.

Figure 1 illustrates the architecture of FNN, which Han proposed in 2016 (Han et al. 2016). The model consists of four layers. Each one has the number of neurons (nodes), and input and output values. The input value for each neuron is defined as is I_b^L and the output value is defined as O_b^L. L is set as the name of the layer and b as the number of neurons.

First layer:

The input value in the first layer is the amount of storage usage for each server in one week. The applied FNN structure predicts the amount of absolute storage usage for each server according to the input value. For instance, the input value can be the weekly storage usage during the last 70 days (10 input values), last 7 months (30 input values), or last year (52 input values). A collection of the servers during a specific time duration is monitored to predict the future customer storage usage.
Because of the huge amount of the storage usage for different servers, all the input values are normalized and de-normalized between 0 and 1 before and after applying the model. The input and output values in the first layer are assigned as follows:

$$I_h^{\text{Input}L} = x_h \tag{1}$$

$$O_h^{\text{Input}L} = I_h^{\text{Input}L} \tag{2}$$

In this equation $h = 1, 2, \ldots, k$, k is the number of node (neuron) in the first layer (input layer) and x is $[x_1, x_2, \ldots, x_k]$.

The second layer:

The second layer (fuzzy layer) is the RBF and the Gauss function which is applied for the fuzzy system.
The definition of membership value $\mu(x)$ for the Gaussian membership function is as follows:

$$\mu(x) = e^{\frac{-(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}} \tag{3}$$

The input and output values of the RBF layer are defined as follows:

$$I_p^{\text{RBF}L} = O_p^{\text{Input}L} \tag{4}$$

$$O_p^{\text{RBF}L} : \varphi_j(t) = \prod_{i=1}^{k} e^{\frac{-(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}} = e^{-\sum_{i=1}^{k}\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$
$$i = 1, 2, \ldots, k; j = 1, 2, \ldots, p \tag{5}$$

where $\varphi_j(t)$ is the output of the jth RBF nodes. $\mu_j = [\mu_{1j}, \mu_{2j}, \ldots, \mu_{kj}]$ and $\sigma_j = [\sigma_{1j}, \sigma_{2j}, \ldots, \sigma_{kj}]$ are the vectors of the membership and width of the jth RBF node.

The third layer:

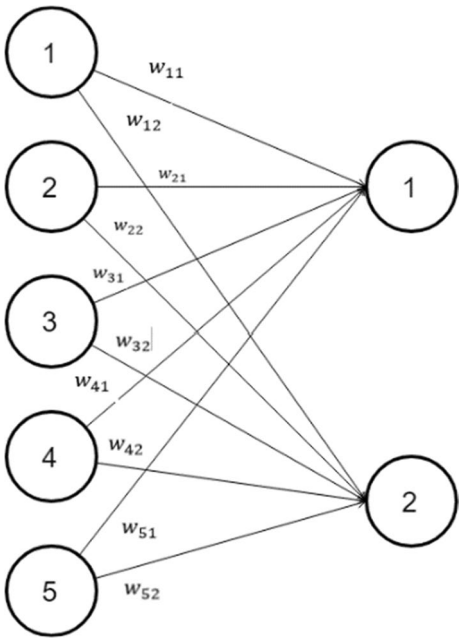


Fig. 2 The weight between third and fourth layer

The third layer is the normalized layer and the number of nodes here is equal to the number of nodes in the second RBF layer

$$I_p^{\text{Normalized}L} = \varphi_j(t) \quad j = 1, 2, \ldots, p \tag{6}$$

$$O_p^{\text{Normalized}L} : \vartheta_j(t) = \frac{\varphi_j(t)}{\sum_{j=1}^{p}\varphi_j(t)} \quad j = 1, 2, \ldots, p \tag{7}$$

The fourth layer:

The fourth layer is the output layer.

$$I_q^{\text{Output}L} = \vartheta_j(t) \tag{8}$$

$$O_q^{\text{Output}L} : y_i = \sum_{j=1}^{p} w_{ji}\vartheta_j(t), \quad i = 1, 2, \ldots, q \tag{9}$$

where w_{ji} is the weight between the jth node (neuron) in the third layer (normalized layer) and ith node (neuron) in the fourth layer (output layer), the number of neurons in this layer (number of output value) depends on the operator and it can be 1, 2,….
Finally, y_i is the output of the ith node in the fourth layer, which is calculated as in the equation below (Fig. 2):

$$y_i = \frac{\sum_{j=1}^{p} w_{ji} e^{-\sum_{i=1}^{k} \frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}}{\sum_{j=1}^{p} e^{-\sum_{i=1}^{k} \frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}}, \quad i = 1, 2, \ldots, q \tag{10}$$

Learning algorithm

The adaptive second-order algorithm

The optimization method has an important role in the efficiency of the training process of neural networks and it affects the ability of neural networks, which depend on the size and the architecture of the network (Hornik et al. 1989) and thus, the limitation of training algorithms has a strong effect on the performance of neural networks (Reed and Marks 1999).

One of the most popular training algorithms is the Levenberg–Marquardt (LM) method (Hagan and Menhaj 1989). The proposed algorithm is the combination of the Gauss Newton method and gradient descent. Hagan and Menhaj (1989) tested the LM algorithm on several function-approximation problems and the results are compared with the conjugated gradient algorithm and with the variable learning rate backpropagation. The results show that LM is more efficient than backpropagation and conjugating the gradient in medium- to large-scale problems.

In 2002, the Levenberg–Marquardt algorithm is proposed with an adaptive momentum for training of feedforward neural network by Ampazis and Perantonis (2002). The algorithm is tested on learning tasks that are known for their difficulty. The final results show that the proposed algorithm can solve this task very successfully.

Following the computation procedure of Levenberg–Marquardt which is proposed by Ampatis and Perantonis (2002),

$$j_q(t) = \left[\frac{\partial e_q(t)}{\partial \mu^1(t)}, \frac{\partial e_q(t)}{\partial \mu^2(t)}, \cdots \frac{\partial e_q(t)}{\partial \mu^Q(t)}, \frac{\partial e_q(t)}{\partial c^1(t)}, \frac{\partial e_q(t)}{\partial c^2(t)}, \cdots \frac{\partial e_q(t)}{\partial c^Q(t)}, \frac{\partial e_q(t)}{\partial w^1(t)}, \frac{\partial e_q(t)}{\partial w^2(t)}, \cdots \frac{\partial e_q(t)}{\partial w^Q(t)}\right] \tag{18}$$

the learning rule of the adaptive second-order algorithm (ASOA) is given by

$$\Theta(t+1) = \Theta(t) + (\Psi(t) + \lambda(t) \times I)^{-1} * \Omega(t) \tag{11}$$

Ψ(t) is the quasi-Hessian matrix in this formula, $\Omega(t)$ is the gradient vector, I is the unit matrix, and $\lambda(t)$ is the adaptive learning rate.

In regard to the RBF FNN which is proposed in Sect. 2, $\Theta(t)$ there are three types of variables: the weight parameters (which are defined between the normalized layer and the output layer), the center (μ_{ij}) and the width (σ_{ij}) of membership function (which are defined between the input layer and RBF layer) and the connection weight between the third and fourth layers w_{ji}, therefore, the $\Theta(t)$ is defined as (Han et al. 2016)

$$\Theta(t) = [\mu^1(t), \ldots, \mu^Q(t), \sigma^1(t), \ldots, \sigma^Q(t), w^1(t), \ldots, w^Q(t)] \tag{12}$$

The weight parameters, the center of membership function, and the width of membership function can be optimized concurrently by ASOA-FNN.

The $\Psi(t)$ (quasi-Hessian matrix) is defined as the summation of submatrices:

$$\Psi(\mathbf{t}) = \sum_{q=1}^{Q} \Psi_q(t) \tag{13}$$

where the related submatrices are

$$\Psi_q(t) = j_q^T(t) j_q(t) \tag{14}$$

The Gradient vector ω(t), which is proposed by Han et al. (2016), is

$$\omega\,(\mathrm{t}) = \sum_{q=1}^{Q} \omega_q(t) \tag{15}$$

where the related subvectors are

$$\omega_q(t) = j_q(t) e_q(t) \tag{16}$$

$e_q(t)$ is the error of the qth neuron and it is the difference between the output layer and the expected value of qth neuron:

$$e_q(t) = y_q(t) - \widehat{g_q(t)} \tag{17}$$

$j_q(t)$ is accumulated as

$\lambda(t)$ is the adaptive learning rate and formalized as (Han et al. 2016)

$$\lambda(t) = \mu(t)\lambda(t-1), \quad 0 < \lambda(t) < 1 \tag{19}$$

Han et al. (2016) propose $\mu(t)$ as

$$\mu(t) = (\tau^{\min}(t) + \lambda(t-1))/(\tau^{\max}(t) + 1), \quad 0 < \tau^{\min}(t) < \tau^{\max}(t) \tag{20}$$

$\tau^{\min} \wedge \tau^{\max}$ which are defined as the maximum and minimum eigenvalue of $\Theta(t)$ (Han et al. 2016).

Improved ASOA-FNN

The adaptive learning rate plays an important role in the basic ASOA-FNN process in that the small learning rate can improve the algorithm's local searching ability, while the bigger adaptive learning rate can enhance the ability of the global search. A continuous decrease of $\lambda(t)$ helps ASOA-FNN to have an effective global and local search and it avoids local extremum. Therefore, we need to stabilize the decrease rate of learning in order to maintain a productive learning rate. Thus, this article adopts a new learning rate as is shown in Formulae (21) and (22).

$$\tau_{\text{average}} = (\tau^{\min}(t) + \tau^{\max}(t))/2, \quad 0 < \tau^{\min}(t) < \tau^{\max}(t) \tag{21}$$

The following, τ_{average} represents the average of maximum and minimum eigenvalue of $\Theta(t)$ and $\mu(t)$ is defined as

$$\mu(\text{t}) = (\tau_{\text{average}})/(\tau_{\text{average}} + 1) \tag{22}$$

Differential evolution

Differential Evolution (DE) algorithm is a type of Genetic algorithm, which is proposed by Storn and Price (2005). The value of variables in the DE algorithm is represented by a real number. The search technique in DE is based on population evolution. DE randomly chooses the initial population ($X^0=[x_1^0, x_2^0, \ldots, x_{NP}^0]$), in which NP is the size of pool. Pool is a set of problem variables. The variables of this model are the center of membership function, the width of membership function, and the connection weight between the third and fourth layer. After a series of operations (mutation, crossover, and selection), the pool of the jth generation improves to $x_i^t=[x_{i,1}^j, x_{i,2}^j, \ldots, x_{i,NP}^j]$.

The three important types of operation in DE are mutation, crossover, and selection.

Mutation operation

Avoiding evolution from local optimal solution is the most important role of mutation. Constant mutation operator has some disadvantages. For example, with too high a mutation rate, the algorithm search is too random and it results in a massive decrease of searching efficiency. Low accuracy of the globally optimal solution is the underlying cause of too randomness of an algorithm search.

Consequently, Hou et al. (2018) proposed the adaptive mutation rate as

$$F = F_{\min} + \frac{g_{\max} - g}{g_{\max}} * F_{\max}, \tag{23}$$

where $F_{\min}=0.1$ and $F_{\max}=0.9$

Crossover operation

Maintaining the diversity of population is the main goal of a crossover operation. To generate the trial vector in a crossover operation, two vectors will be chosen. The trial vector is the child of vectors, which is calculated by mutation operation and the vector, which is chosen randomly from the pool. The child vector inherits the parameters from parents with the crossover constant probability (CR). For instance, when the crossover constant is equal to 0, the trial vectors come from the parent, which is chosen randomly from the pool. On the other hand, when the crossover constant is equal to 1, the trial vectors inherit from X_m.

The size of the crossover probability factor plays a critical role in the DE algorithm. On one hand, the algorithm's local searching ability is decreased through a small crossover rate and on the other hand, the diversity of the pool and the global convergence of algorithms is improved through a big crossover rate.

Hou et al. (2018) proposed an adaptive crossover rate, which is defined as

$$\text{CR} = \text{CR}_{\min} + \frac{g}{g_{\max}} * \left(\text{CR}_{\max} - \text{CR}_{\min}\right) \tag{24}$$

They defined $\text{CR}_{\max}$, $\text{CR}_{\min}$ as preset numbers and $\text{CR}_{\max}=0.9, \text{CR}_{\min}=0$.

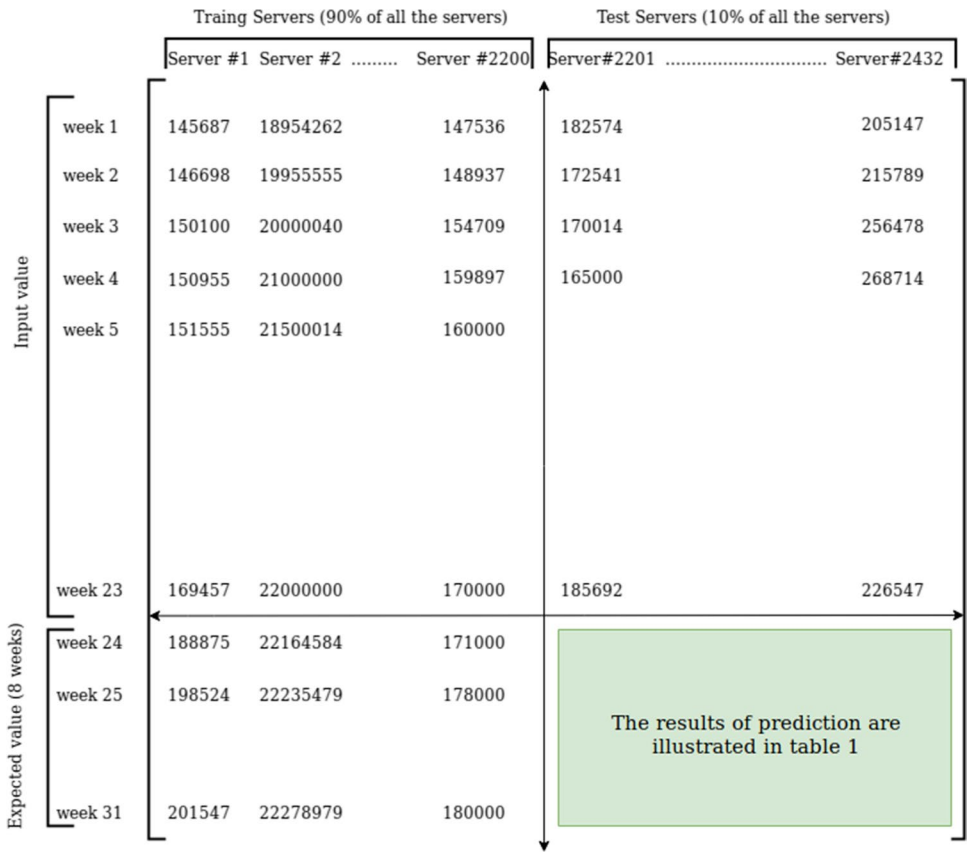
Selection operation

The value of the objective function of the target vector and the trial vector is compared. If the trial vector has the lower objective function, it will be replaced with the target vector.

Backpropagation algorithm

Backpropagation (BP) algorithm is the wildly popular learning algorithm for the neural networks with more than one hidden layer because of its simplicity and effectiveness (Rumelhart et al. 1986). It is used to determine a gradient which is needed in the calculation of the weights between the hidden layers in the network. The amount of learning rate has a crucial role in searching for the local optimal solution of neural network parameters. For instance, a slow convergence is the result of a low learning rate and divergence is the result of a high learning rate. Duffner and Garcia (2007) proposed an online BP algorithm with an adaptive global learning rate. The "blod driver" method is the main idea of the adaption of the learning rate (Battiti 1989). In this article, an online BP algorithm with an adaptive global learning rate, which is proposed by Duffner and Garcia (2007), is applied.

Fig. 3 The input value



Differential evolution–Backpropagation algorithm

Hou et al. (2018) proposed a new FNN training algorithm to optimize the FNN parameters. In this algorithm, the improved DE algorithm and BP algorithm are combined. They used the improved DE algorithm to gain the suboptimal solution or global optimal solution of the FNN parameters. The BP algorithm is applied to improve the local optimal solution of FNN (Han et al. 2016).

In this article, improved DE algorithm (Han et al. 2016) and BP algorithm (Duffner and Garcia 2007) are combined and the results of improved ASOA, ASOA, DE, BP, DE–BP algorithms and other prediction methods such as AR, ARMA, and Mackey glass time series are compared.

The applied objective function in all the algorithms is the Sphere function:

$$f(x) = \sum_{j=1}^{N} \sum_{i=1}^{npw} \left(\text{expected}_{\text{value}} - \text{output}_{\text{value}}\right)_i^2 \tag{25}$$

Results of the experiment

Experimental data of the project are from a famous web hosting database. Monitoring time is from May 5, 2018 to February 20, 2019 (31 weeks), and every week the average storage usage for each server is collected. The 2432 servers, that have a complex behavior in using the storage, are taken as the sample for the experiment. For the 2432 servers, the first 2200 servers (90% of all the servers) are used for the training process, and the remaining 232 servers (10% of the servers) are used for the testing process. We run the algorithms for node 11 and the number of nodes in output layer is defined as 8 (Fig. 3).

Performance of each algorithm is assessed by the root mean square error (RMSE), the average percentage error (APE), and the running time duration. They are presented as (26)–(27):

$$\text{RMSE} = \sqrt{\frac{1}{N * npd} \sum_{i=1}^{\text{number of customer}} \sum_{i=1}^{npd} \left(\text{output}_{\text{value}} - \text{expected}_{\text{value}}\right)^2} \tag{26}$$

$$\text{APE}(t) = \sum_{t=1}^{N} \frac{\|e(t)\|}{\|y(t)\|} * 100\% \tag{27}$$

Table 1 The result of RMSE for fuzzy neural network algorithms

	Improved ASOA-FNN			ASOA-FNN			Differential evolution-FNN		
	RMSE	APE	Time	RMSE	APE	Time	RMSE	APE	Time
1	0.052918	0.262103	33,272.6	0.092713	0.553029	33,897.3	0.210451	1.42117	5818.6
2	0.044515	0.229996	33,529.6	0.076374	0.424813	33,882.4	0.248240	1.56580	5828.8
3	0.055042	0.308270	33,560.1	0.067147	0.405723	33,468.4	0.210096	1.02780	5819.3
4	0.038686	0.186657	33,832.4	0.087654	0.601053	33,571.3	0.269212	1.71826	5353.0
5	0.056859	0.298585	29,952.6	0.075914	0.489524	33,120.9	0.265727	1.65369	5348.2
6	0.057300	0.341216	31,466.5	0.083657	0.569361	30,616.1	0.264530	1.76038	5359.3
7	0.059499	0.364398	33,227.8	0.067739	0.460109	70,567.4	0.222404	1.38350	5341.4
8	0.038677	0.189270	33,474.0	0.080202	0.490355	33,520.5	0.228837	1.53863	5364.6
9	0.050956	0.273833	33,852.8	0.081003	0.502207	34,202.0	0.244301	1.50700	5365.3
10	0.060215	0.316136	33,782.9	0.065839	0.407165	33,457.3	0.252491	1.72679	5323.3
11	0.052016	0.249054	33,523.8	0.073611	0.433029	33,859.4	0.256565	1.86490	5333.7
12	0.044130	0.198994	33,629.4	0.068618	0.399666	36,316.9	0.273712	1.81902	5523.8
13	0.054556	0.312821	36,867.3	0.076638	0.499870	36,369.6	0.263638	1.74305	5519.9
14	0.060754	0.340892	33,893.5	0.083845	0.503672	70,744.1	0.275498	1.60468	5519.6
15	0.053718	0.309730	53,885.3	0.086072	0.538686	36,399.29	0.279547	1.81980	5526.7
16	0.054854	0.290840	30,668.8	0.109835	0.750936	36,289.3	0.272288	1.74444	5332.7
17	0.056527	0.313702	32,937.4	0.074212	0.458074	70,594.1	0.276727	1.71754	5511.0
18	0.061180	0.344348	31,024.1	0.072546	0.454730	36,347.1	0.270511	1.61183	5527.4
19	0.054278	0.280791	26,312.3	0.075648	0.472290	36,413.7	0.273857	1.85694	5522.92
20	0.043334	0.204978	36,349.4	0.078266	0.498800	36,305.6	0.214074	1.39583	5521.0

	Differential evolution–back propagation-FNN		Back propagation-FNN	
	RMSE	Time	RMSE	Time
1	0.288204	7232.9	0.503287	6124.9
2	0.345253	7322.8	0.479736	6141.8
3	0.303116	7095.7	0.486390	6119.2
4	0.308845	7207.4	0.428263	6445.1
5	0.306213	7139.40	0.386645	6485.4
6	0.289191	7109.4	0.509079	6476.5
7	0.278877	7221.0	0.490810	6506.4
8	0.306075	7192.8	0.472320	6480.3
9	0.302166	7137.75	0.480040	6484.3
10	0.301214	7590.4	0.487667	6153.9
11	0.301734	7558.3	0.471530	6185.8
12	0.283162	7216.6	0.488372	6152.8
13	0.295977	7349.8	0.456622	6139.4
14	0.318921	7126.4	0.485130	6158.8
15	0.306804	7129.1	0.466852	6150.1
16	0.268998	7281.5	0.426317	6182.9
17	0.244742	7275.0	0.444901	6156.1
18	0.256048	7350.7	0.483375	5744.0
19	0.269642	7407.1	0.465989	6155.6
20	0.330233	7110.2	0.460492	6166.4

The ASOA initial leaning rate is assumed as 0.999. The learning iteration for each algorithm is 2000. In DE–BP algorithm the 1000 iterations for each algorithm is set. The details are presented in Table 1.

The results show, the predicting values gained from the improved ASOA-FNN model is more accurate than other algorithms. The predicting values from the improved ASOA-FNN are compared with those from ASOA-FNN

Table 2 The results of other prediction methods and fuzzy neural network

	Other prediction methods			Fuzzy neural network				
	Autoregression (AR)	Autoregression moving average (ARMA)	Mackey glass time series	Improved ASOA-FNN	ASOA-FNN	DE-FNN	DE–BP-FNN	BP-FNN
RMSE	0.697819	0.895476	0.380734	0.061180	0.109835	0.279547	0.330233	0.509079

(Han et al. 2016), DE-FNN (Hou et al. 2018), BP-FNN (Duffner and Garcia 2007), and DE–BP-FNN (Hou et al. 2018; Duffner and Garcia 2007).

The results of experiments show that the improved ASOA-FNN has the smaller RMSE and APE than ASOA-FNN, DE-FNN, BP-FNN, and DE–BP-FNN, but the running time of improved ASOA-FNN and ASOA-FNN is higher than the other algorithms. Another important result is that the DE-FNN has smaller RMSE than DE–BP-FNN. In this problem, the RMSE of the DE algorithm with 2000 iteration is less than the RMSE of a combined algorithm with 1000 DE iterations and 1000 BP iterations.

In Table 2, there is a comparison regarding the predicting values of other methods such as Mackey Glass time series, autoregression (AR), autoregressive moving average (ARMA), with the biggest RMSE of improved ASOA-FNN, ASOA-FNN, DE-FNN, BP-FNN, DE–BP-FNN.

The results show that the improved ASOA, ASOA, DE, and DE–BP algorithms have a smaller RMSE than other prediction methods (AR, ARMA, and Mackey Glass time series) and the Mackey Glass time series has a smaller RMSE than the BP-FNN.

Significance test

In this part, the predictive performance of improved ASOA-FNN with other algorithms (ASOA, DE, BP, and DE–BP) will be compared by significance test methods. The main point of evaluating the predictive performance of a model is as follows (Raschka 2018):

1. Estimating the generalization performance and the predictive performance of an improved and identified algorithm.
2. Identifying the machine learning algorithm, that is suitable for our model and has the best performance.

The Wilcoxon signed-rank test, *F* test, and Morgan–Gragner–Newbald test (Diebold and Mariano 1995) are implemented at the 0.01, 0.05, and 0.1 significance level in one-tailed test to confirm the performance of the improved ASOA. The test helps us to evaluate the predictive ability

Table 3 Significance test between improved ASOA and ASOA

Sample size	Wilcoxon signed-rank test *p* value	Morgan–Gragner–Newbald test *p* value	F test *p* value
5	0.021562**	0.004584***	0.16056
10	0.002531***	0.000153***	0.106879
15	0.000328***	<0.00001***	0.066646*
20	0.000044***	<0.00001***	0.038187**

Table 4 Significance test between improved ASOA and DE

Sample size	Wilcoxon signed-rank test *p* value	Morgan–Gragner–Newbald test *p* value	*F* test *p* value
5	0.021562**	0.001771**	0.003253**
10	0.002531***	0.000018***	0.000017***
15	0.000328***	<0.00001***	<0.00001***
20	0.000044***	<0.00001***	<0.00001***

Table 5 Significance test between improved ASOA and DE–BP

Sample size	Wilcoxon signed-rank test *p* value	Morgan–Gragner–Newbald test *p* value	*F* test *p* value
5	0.021562**	0.000168***	0.000538***
10	0.002531***	<0.00001***	<0.00001***
15	0.000328***	<0.00001***	<0.00001***
20	0.000044***	<0.00001***	<0.00001***

Table 6 Significance test between improved ASOA and BP

Sample size	Wilcoxon signed-rank test *p* value	Morgan–Gragner–Newbald test *p* value	*F* test *p* value
5	0.021562**	0.000081***	<0.00001***
10	0.002531***	<0.00001***	<0.00001***
15	0.000328***	<0.00001***	<0.00001***
20	0.000044***	<0.00001***	<0.00001***

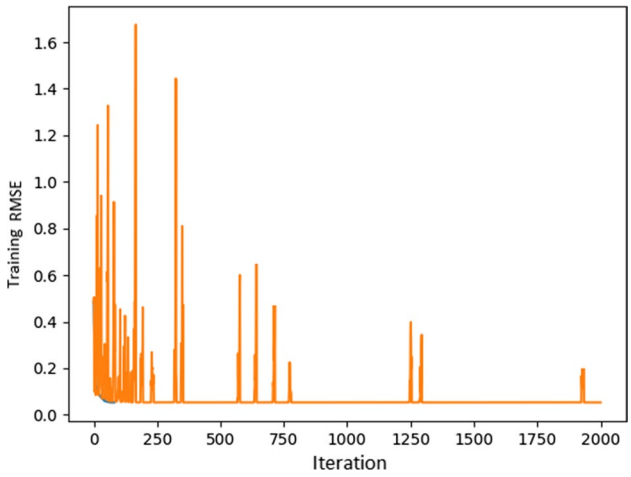


Fig. 4 Evolution of training RMSE in improved ASOA-FNN

of two different learning algorithms and the statistically significant difference between the results is considered.

The performance metric RMSE is selected to carry out the non-parametric test for making a comparison of the forecasting performance of improved ASOA, ASOA, DE, BP, and DE–BP. The comparison results are illustrated in Tables 3, 4, 5, 6. The end results show that the improved ASOA achieves statistical significance in contrast to ASOA, DE, BP, and DE–BP at the 0.01,[2] 0.05[3], and 0.1[4] level. In Table 3, the *F* test between improved ASOA and ASOA with sample size 5 and 10 is not significant but by increasing the sample size to 15 and 20, it is significant. The results of this significance test illustrate that the predict performance of improved ASOA is better than the other algorithms.

Figure 4 shows the typical evolution of RMSE during the training process for node 11 in improved ASOA-FNN by using the applied learning algorithms. As shown in the figure, at the beginning of the learning process the RMSE is high but after some iterations it drops and finally converges to a minimum value.

Conclusion

Storage usage forecasting will provide reliable data support for resource management and resource planning, and storage usage forecasting technology is an effective means to optimize the resources. So, for the web hosting experts who work in the field of web hosting resources, they must first give attention to a forecasting method. On this basis, customer value can be determined, and customer portfolio will be optimized.

In this article a RBF-FNN architecture has been applied and some learning algorithms such as ASOA, DE, BP, and DE–BP have been applied. The applied ASOA-FNN, which is proposed by Han et al. (2016) has been improved. Three significance tests have been implemented to confirm the performance of the improved ASOA and other applied algorithms. The end results show that the improved ASOA outperforms ASOA, DE, BP, and DE–BP by statistic significance tests.

The other classical prediction methods such as autoregressive, autoregressive moving average, and Mackey Glass time series have been implemented and the results have been compared with the algorithms which are applied to FNN. The comparisons of the classical prediction methods and FNN demonstrate that the learning efficiency and performance of the improved ASOA-FNN are better than others.

Considering an intelligent data mining algorithm, which can cluster the customers in different types of lifecycle in relation to predicting the storage usage and other resources of servers such as CPU usage will be vital in future.

Funding Open access funding provided by Projekt DEAL.

References

Ampazis, N., and S. Perantonis. 2002. Two highly efficient second-order algorithms for trainingfeedforward networks. *IEEE Transactions on Neural Networks.* 13: 1064–1074. https://doi.org/10.1109/tnn.2002.1031939.

Barba, L., and N. Rodríguez. 2015. Traffic accidents forecasting using singular value decomposition and an autoregressive neural network based on PSO. *Polibits* 51: 33–38. https://doi.org/10.17562/pb-51-5.

Battiti, R. 1989. Accelerated backpropagation learning: Two optimization methods. *Complex Systems* 3: 331–342.

Borchers, D.L., and R. Langrock. 2015. Double-observer line transect surveys with Markov-modulated Poisson process models for animal availability. *Biometrics* 71: 1060–1069. https://doi.org/10.1111/biom.12341.

Diebold, F., and R. Mariano. 1995. Comparing predictive accuracy. *Journal of Business and Economic Statistics*. https://doi.org/10.3386/t0169.

[2] ***Significance level 0.01.

[3] **Significance level 0.05.

[4] *Significance level 0.1.

Duffner, S., and C. Garcia. 2007. An online backpropagation algorithm with validation error-based adaptive learning rate. In *Lecture notes in computer science*, 249–258. https://doi.org/10.1007/978-3-540-74690-4_26.

Fei, J., and H. Ding. 2012. Adaptive sliding mode control of dynamic system using RBF neural network. *Nonlinear Dynamics* 70 (2): 1563–1573. https://doi.org/10.1007/s11071-012-0556-2.

Hagan, M., and M. Menhaj. 1994. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks.* 5: 989–993. https://doi.org/10.1109/72.329697.

Han, H., L.M. Ge, and J.F. Qiao. 2016. An adaptive second order fuzzy neural network for nonlinear system modeling. *Neurocomputing* 214: 837–847. https://doi.org/10.1016/j.neucom.2016.07.003.

Hornik, K., M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2 (5): 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

Hou, Y., L. Zhao, and H. Lu. 2018. Fuzzy neural network optimization and network traffic forecasting based on improved differential evolution. *Future Generation Computer Systems* 81: 425–432. https://doi.org/10.1016/j.future.2017.08.041.

Huang, S., and D. Ben-Hsiang. 2014. Radial basis function based neural network for motion detection in dynamic scenes. *IEEE Transactions on Cybernetics* 44 (1): 114–125. https://doi.org/10.1109/tcyb.2013.2248057.

Iliev, M., and B. Bedzhev. 2015. An algorithm for synthesis of binary phase manipulated signals with optimal periodic auto-correlation properties. In *2015 IEEE international black sea conference on communications and networking (BlackSeaCom)*. https://doi.org/10.1109/blackseacom.2015.718511.

Mai, T., B. Ghosh, and S. Wilson. 2014. Short-term traffic-flow forecasting with auto-regressive moving average models. *Proceedings of the Institution of Civil Engineers—Transport* 167 (4): 232–239. https://doi.org/10.1680/tran.12.00012.

Moretti, F., S. Pizzuti, S. Panzieri, and M. Annunziato. 2015. Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling. *Neurocomputing* 167: 3–7. https://doi.org/10.1016/j.neucom.2014.08.100.

Morikawa, K., and A. Tsuneda. 2014. On auto-correlation properties of random bit sequences by post-processing based on chaos theory. In *2014 14th international symposium on communications and information technologies (ISCIT)*. https://doi.org/10.1109/iscit.2014.7011879.

Price, K.V., R.M. Storn, and J.A. Lampinen. 2005. *Differential evolution a practical approach to global optimization*. Berlin: Springer.

Raschka, S. 2018. Model evaluation, model selection, and algorithm selection in machine learning.

Reed, R.D., and R.J. Marks. 1999. *Neural Smithing*. https://doi.org/10.7551/mitpress/4937.001.0001.

Rumelhart, D.E., J.L. McClelland, and CORPORATE PDP Research Group. 1986. *Parallel distributed processing: Explorations in the micro structure of cognition, vol. 1: Foundations*. Cambridge, MA: MIT Press.