Soft Computing (2023) 27:1797–1808
https://doi.org/10.1007/s00500-021-06093-6

FOCUS

A new hybrid neural network classifier based on adaptive neuron and multiplicative neuron

Erdinç Kolay[1] · Taner Tunç[2]

Accepted: 28 July 2021 / Published online: 6 August 2021

Abstract
Neural network (NN) classifiers are very popular tools for solving classification tasks. Mostly known NN classifier is a multilayer perceptron (MLP). Although MLP has a good correct classification ratio, its structure could be very complex and network training may work for a long time. Pi-sigma NN (PSNN) is higher-order NN (HONN), which used higher-order correlations among the input components to establish a HONN, and the PSNN utilizes the product of neurons as the output units. By contrast with MLP and PSNN, single multiplicative neuron (SMN) is simple concerning its structure and mathematical model. The absence of the hidden layer(s) could be an advantage for easy implementation, and the mathematical model can be easily interpreted. In this paper, we propose a new hybrid NN classifier based on simple adaptive neurons and SMN which form the SMN as a whole, where input units are constituted by adaptive neurons. In contrast with conventional NN, our proposed classifier can use fewer learning parameters. To train this network, we use a modified particle swarm optimization (MPSO) algorithm. For the investigation of the generalization capability of the proposed classifier, we compare this method to other NN classifiers: MLP and PSNN together with other classification procedure classifiers.

Keywords Classification · Multilayer perceptron · Pi-sigma neural network · Multiplicative neuron · Particle swarm optimization

1 Introduction

The classification problem is defined as assigning an object to one of the predefined categories. The oldest known classification method is discriminant analysis (DA) and this method has well enough classification performance when only assumptions are supplied. When the explanatory variables are categorical, DA cannot work well enough. Logistic regression (LR), a special case of generalized linear models, has no assumptions on the explanatory variables or inputs and may take better results than DA. In LR, model parameters are obtained by the maximum likelihood approach, but performance on the training set is not a good indicator of predictive performance on unseen data because of the problem of over-fitting (Bishop 2006). To achieve good prediction performance on unseen data, MLP can be used. MLP is organized by more than two LR models when logistic activation is selected as the activation function. Rumelhart et al. (1988), proposed the backpropagation learning rules for training MLP network. MLP network using this learning rule have been successfully applied to classification tasks. Nevertheless, the training speeds for MLP are extremely slow. Additionally, finding the best architecture design of MLP and decision-making hidden layer numbers and hidden layer units are all by itself problems. Therefore, researchers have attempted to increase the performance of MLP by either modified backpropagation algorithm (BPA) (Nienhuis 1992; Riedmiller 1993; Hagan and Menhaj 1994) or different training algorithms; particle swarm algorithm (Mendes et al. 2002), genetic algorithm (GA) (Montana and Davis 1989), artificial bee colony algorithm (Karaboga et al. 2007),

Communicated by Mu-Yen Chen.

✉ Taner Tunç
ttunc@omu.edu.tr

Erdinç Kolay
ekolay@sinop.edu.tr

[1] Department of Statistics, Sinop University, Sinop, Turkey

[2] Department of Statistics, Ondokuz Mayıs University, Samsun, Turkey

differential evolutional algorithm (Ilonen et al. 2003), grey wolf optimization algorithm (GWO) (Mirjalili 2015), multi-verse optimization algorithm (Faris et al. 2016) and combining of partial least square algorithm and hierarchical cluster analysis (Jia et al. 2015).

Except for MLP, various architectures of NN have been proposed to solve a classification problem. In Yadav (2003), the authors show different neuron models such as Sigma-Pi-Sigma, Sigma-Pi-Radial Basis, and Sigma-Pi neuron model for classification, and they offer using only a single neuron unit model called Sigma-Pi neuron model. On the other hand, the structure of NN is the PSNN which was firstly proposed in Shin and Ghosh (1991). This NN is a higher-order and less variant, fast convergence speed and has a fewer learning parameter than MLP (Kolay et al. 2016; Husaını et al. 2012), and the PSNN have been applied to image coding (Hussain and Liatsis 2003), time series prediction (Ghazali 2009; Li 2003) and function approximation (Nie 2008). In classification problems, (Nayak et al. 2014) proposes to train the PSNN with a hybrid algorithm combining genetic algorithm and PSO algorithm, (Panigrahi et al. 2013) proposes to train this network with the differential evolutional algorithm. However, with an increasing number of higher-order terms for PSNN, training the PSNN will be much more difficult. Also, for multi-class classification problems, the training time of the PSNN becomes unbearable. On the contrary to the MLP and the PSNN, SMN has a simple structure. SMN, based on polynomial architecture, is firstly introduced in Yadav et al. (2007) for time series prediction. Excluding above-mentioned NN models, varied hybrid NN structures for solving classification problems are proposed in Tunc (2012), Shekara et al. (2011), Chuang (2011), Lim et al. (2005) to solve classification problems.

In the training process, the weights of the network elements are adjusted by an optimization procedure. As is known, BPA is based on the gradient descent method and depends on the initialized values, and this method may easily fall into local optima. Recently, different evolutionary optimization algorithms, including PSO, are playing a vital role in solving optimization problems, and these algorithms have been proposed to improve the capability of the neural network. PSO is a population-based stochastic algorithm which was firstly introduced in Eberhart and Kennedy (1995). In the PSO, each particle searches the global optimum point in multi-dimensional search space. Researchers use PSO (Mendes et al. 2002; Kolay et al. 2016; Khan and Sahai 2012; Engelbrecht 2001; Yolcu et al. 2013; Hakan and Erol 2013) or its hybrids (Nayak et al. 2014; Juang 2004; Zhang et al. 2007; Marinakis et al. 2009; Yaghini et al. 2012) for NN training. As with all evolutionary algorithms, PSO may easily fall into local optima, and the convergence rate is significantly decreased in evolutionary processes. To overcome these shortcomings, we use a modified PSO which combines the advantage of the exponent decreasing inertia weight like in Li (2009) and time-varying acceleration coefficient like in Ratnaweera et al. (2004) to avoid premature convergence in the early stages of the search and to enhance convergence to the global optimum solution during the latter stages of the search. In this paper, we propose a new NN structure based on adaptive neurons and multiplicative neurons, and train the proposed NN with MPSO to improve training performance. The proposed NN structure offers an alternative for NN classification methodology with higher classification accuracy and saving training time. Also, the training algorithm is called MPSO is improved to capabilities of PSO in the point of global search ability in multi-dimensional search space.

The paper is organized as follows: In Sect. 2, we sketch in something of conventional NN models. In Sect. 3, we show the MPSO method for optimization of NN parameters. In Sect. 4, the proposed NN classifier is introduced and in the last section of this paper, experiments and results of the proposed method are presented and discussed.

2 Neural networks' structures

NNs are data-based approaches, and one of the most frequently used methods for handling classification problems. The most important task of the NN is optimizing the network parameters, which is called the training network. The process of optimizing the network is based on the minimization of error functions such as *cross-entropy error function* shown in (1). However, over-fitting can arise due to training data more than adequate. To avoid over-fitting, early stopping which stops training when validation error starts going up can be used.

$$E = \sum_{i=1}^{N} \sum_{j=1}^{p} -t_{ij} \log(y_{ij}) \tag{1}$$

Here $t_{ij,}$ taking 1 or 0 values depending on i-th learning sample belonging to j-th class or not, respectively, is the target of the i-th learning sample and y_{ij} is the probability of i-th learning sample belonging to j-th class. N and p are the numbers of training samples and distinct classes, respectively.

2.1 Multilayer perceptron

MLP is the most widely used procedure for solving any problem in artificial NN literature. Figure 1 shows the MLP classifier with one hidden layer and k hidden layer units for multi-class classification for p distinct classes. To

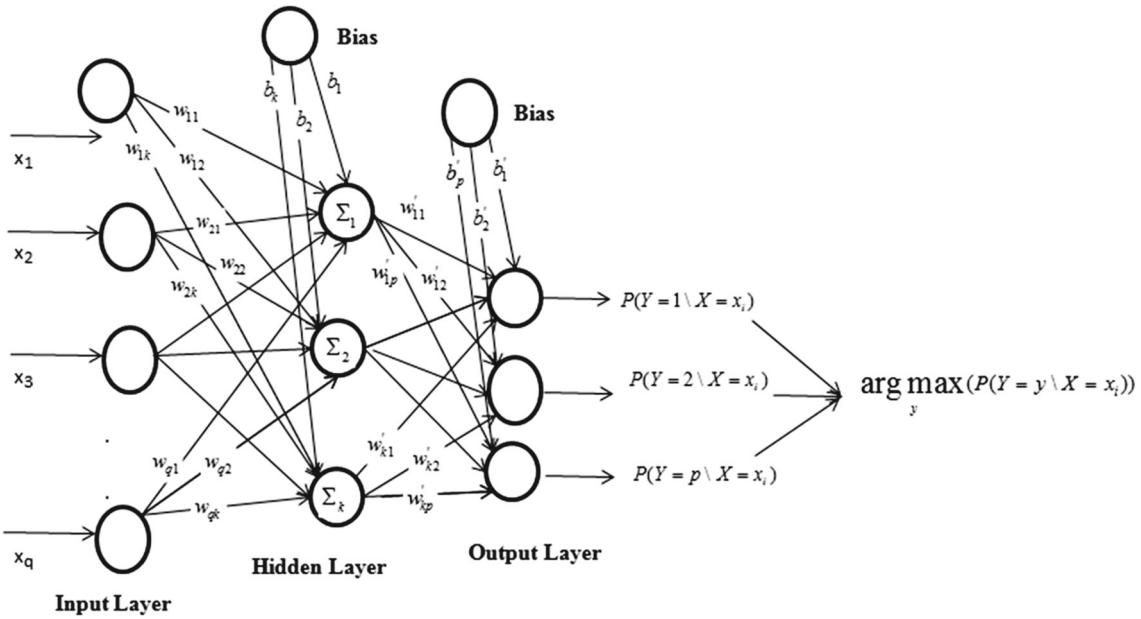

Fig. 1 Multilayer perceptron for p-class classification

obtain an output of i-th learning sample belonging to j-th class (j = 1, 2, ..., p), we use the following equations:

$$\Sigma_i = (x_1 w_{1i} + x_2 w_{2i} + \cdots + x_q w_{qi}) + b_i, \quad i = 1, \ldots, k \tag{2}$$

$$P(Y = j \backslash X = x_i) = g\left(\sum_{i=1}^{k}\left(f(\Sigma_i) w_{ij}^{'} + b_j^{'}\right)\right) \tag{3}$$

where $Y = \{y_1, y_2, ..., y_p\}$ is a set of distinct classes, q is a number of inputs, $w_{ij}^{'}$ is a connection weight of i-th hidden layer unit between j-th output unit, $b_j^{'}$ is a bias for j-th output unit, f is a *logistic* activation function and g is a *softmax* transfer function which provides the occurrence of the probability to ensure that the sum of the components of the output vector is equal to 1. For classifying any observation, a value of output unit with high probability is assigned as a class label.

2.2 Pi-sigma neural network

PSNN is higher-order feed forward NN. The PSNN consists of k summing units instead of hidden layer in the MLP, and the network's output is obtained by the product of k summing units with a fixed weight connection between k summing units and output units. Figure 2 shows the proposed PSNN architecture with one output unit by Shin and Ghosh (1991).

Calculations of the summation $(\Sigma_j, j = 1, ..., k)$ of MLP's hidden layer units and the PSNN's k summing units are the same as shown in (2). To attain output value for i-th learning sample (4) can be used.

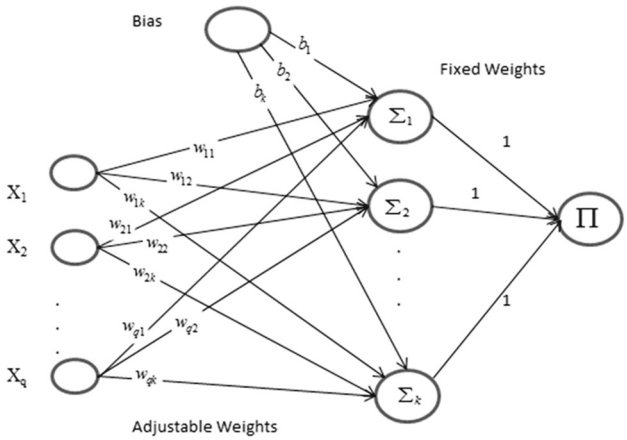

Fig. 2 PSNN with one output unit

$$y^{(i)} = f\left(\prod_{j=i}^{k} \Sigma_j\right) \tag{4}$$

Here k is the number of summing units. If multiple outputs are required, the summing layer is needed for each output unit (Ghosh and Shin 1995). In such a case, the PSNN involves $\sum_{i=1}^{p}(q+1)k_i$ learning parameters for q inputs, where k_i is the number of summing units for i-th output unit. Therefore, the number of parameters of the PSNN will increase when more and more classes exist. Then, the training time of the PSNN will increase too much for a high number of classes. Figure 3 shows PSNN with multiple outputs for p-class classification.

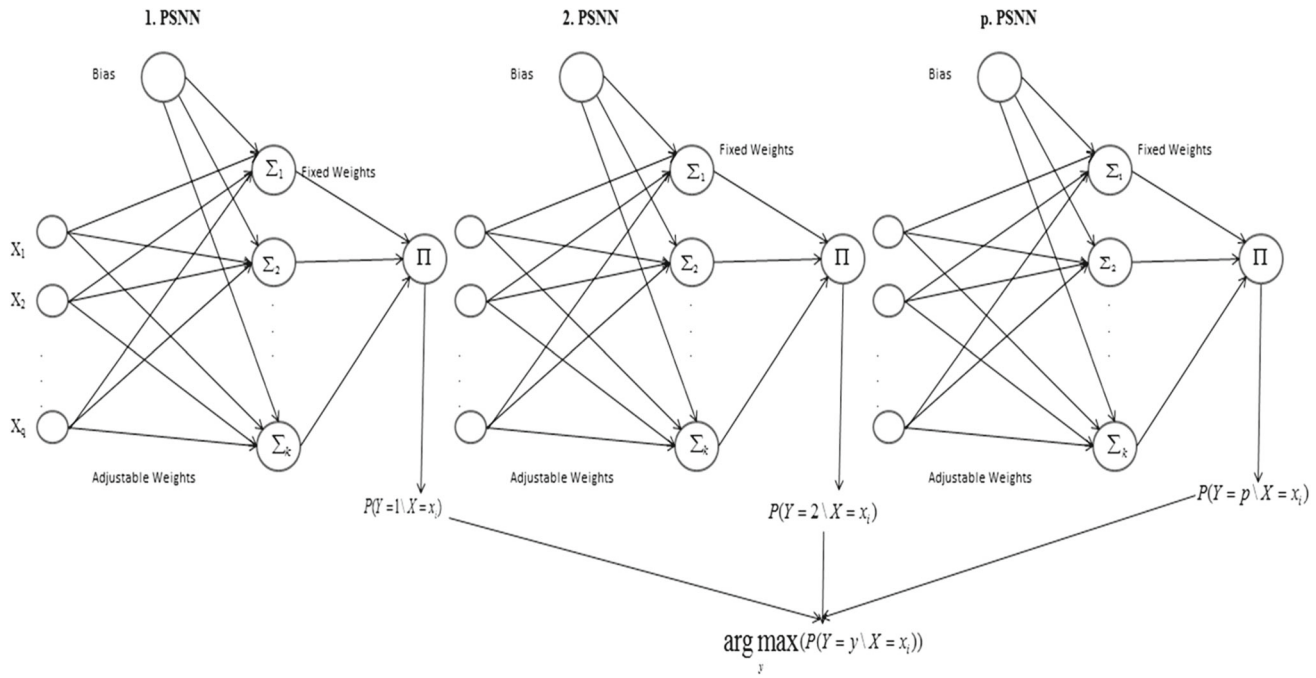

Fig. 3 PSNN with p-class classification

2.3 Single multiplicative neuron

SMN is based on the polynomial structure and is firstly introduced in Yadav et al. (2007). Instead of MLP and PSNN's higher-order terms, SMN uses a simple aggregation function. Figure 4 shows the SMN's structure with an output unit (Yadav et al. 2007). Calculation of activation value of i-th learning sample for q inputs is shown in (5). Yadav et al., use error backpropagation for network training. Due to the slow convergence speed of this algorithm, (Zhao and Yang 2009) proposes the PSO algorithm for training the SMN network. In this NN model, a simple aggregation function, which is considered as a product of linear functions in different dimensions of space, is used. Also, SMN model has reduced the computational time by comparison with MLP and PSNN due to its simple architecture and involving fewer parameters.

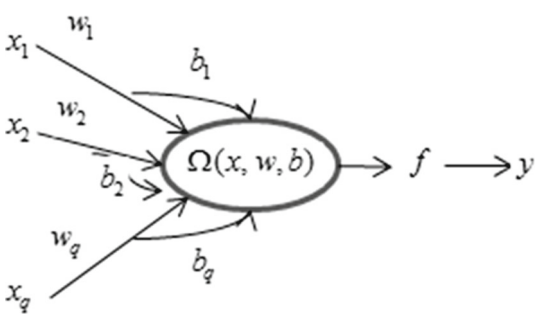

Fig. 4 Single multiplicative neuron

$$\Omega(x, w, b) = \prod_{j=1}^{q} (x_j w_j + b_j) \tag{5}$$

$$y = f(\Omega) = \frac{1}{1 + e^{-\Omega}} \tag{6}$$

In (5), x, w, and b show inputs, weights, and bias, respectively, q is the number of the attribute, and in (6), f is the activation function and y is the class label.

3 The modified particle swarm optimization

PSO is a population-based and stochastic optimization algorithm, and this method is motivated by the simulation of social behavior such as bird flocking or fish schooling. PSO has firstly been introduced in Eberhart and Kennedy (1995) for the optimization of nonlinear functions. In PSO, each element is called a particle and is a candidate for the optimum point in search space. These particles use their own best experience and their neighbor's best experience in the evolutional process. Although it has effective computational performance, PSO may easily fall into local optima as all evolutionary algorithms. Therefore, to improve the PSO performance, researchers must use an effective search strategy. At the early iterations of the PSO, particles are spread over the search space, and these particles are far from the others, comparatively. But, in the next iterations, all particles are gathered and restricted to a small area in the search space. This situation may cause

premature convergence (Hatamlou 2017). For preventing such a situation, we propose the random regenerate particle position once again when the fitness function gives up to evaluation. In this paper, MPSO is used for training NN parameters. The algorithm has a time-varying inertia weight proposed in Li (2009) and also has a varying acceleration coefficient like in Ratnaweera et al. (2004).

Algorithm 1. The MPSO.

Step 1. Initialize i-th (i = 1, 2, …, Npar) particle's position and velocity randomly, and keep in a vector X_i and V_i, respectively, as follows:

$$X_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}, \quad i = 1, \ldots, \text{Npar} \tag{7}$$

$$V_i = \{v_{i1}, v_{i2}, \ldots, v_{iD}\} \tag{8}$$

where x_{id} and v_{id} represent position and velocity of i-th particle at d-th (d = 1, …, D) dimension.

Step 2. According to the evaluation function, personal best position (pbest) and global best position (gbest) particles given in (8) and (9), respectively, are designated.

$$\text{pbest}_i = (p_{i1}, p_{i2}, \ldots, p_{id}) \tag{9}$$

$$\text{gbest} = (p_{g1}, p_{g2}, \ldots, p_{gd}) \tag{10}$$

Step 3. Let c_1 and c_2 represent cognitive and social parameters, respectively, and w is inertia weight parameter. Let (c_{1i}, c_{1f}), (c_{2i}, c_{2f}), and (w_i, w_f) be intervals which include values for c_1, c_2 and w, respectively. During the iteration process, these parameters are updated by using (11), (12), and (13), respectively.

$$c_1^{(t+1)} = (c_{1f}^t - c_{1i}^t)\frac{t}{t_{\max}} + c_{1i}^t \tag{11}$$

$$c_2^{(t+1)} = (c_{2f}^t - c_{2i}^t)\frac{t}{t_{\max}} + c_{2i}^t \tag{12}$$

$$w^{(t+1)} = (w_i - w_f - d_1)\exp\left(\frac{1}{1 + d_2(t/t_{\max})}\right) \tag{13}$$

where t, $t_{\max}$, f, and i represent current iteration number, maximum iteration number, final, and initial, respectively, d_1 and d_2 are control factors to control w between w_i and w_f.

Step 4. Values of velocities are updated by using (14), and values of positions are calculated by using (15).

$$v_{id}^{(t+1)} = wv_{id}^t + c_1 r_1(p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id}) \tag{14}$$

$$x_{id}^{(t+1)} = x_{id}^t + v_{id}^{(t+1)} \tag{15}$$

where r_1 and r_2 are random numbers uniformly distributed in the range (0,1).

Step 5. If there is no improvement in the evaluation function, regenerate particle position vectors randomly for all particles.

Step 6. If stopping criteria is not met, return to step 2.

4 The proposed neural network

For training the neural network, one is usually closely interested in obtaining a network with optimal generalization performance. Improving the generalization capacity of the NN can be based on network structure or network training or both of them. To obtain a better classification performance on unseen data, researchers have attempted to use varied training algorithms or have proposed different NN structures.

MLP is the most used NN classifier and researches on this NN training take part in the artificial intelligence literature. Comparing with MLP, the Pi-Sigma NN is often used for handling classification problems, too. However, training these NN classifiers can be very tough, and computational time can increase because of the high dimension of data and/or a large amount of sample data regarding their architecture. The proposed NN, comparing with MLP and PSNN, can provide a simpler structure, have higher classification accuracy, and consume a short time for training, especially for the multi-class classification.

We propose to use SMN whose inputs are constituted by adaptive neurons for solving binary and multi-class classification problems. This NN is HONN according to its structure, and we should limit the number of adaptive neurons to avoid hard training. Figure 5 shows the proposed NN architecture for multi-class classification. The product of j-th output unit is calculated as (16).

$$\Pi_j^{(i)} = \prod_{t=1}^{k} (\Sigma_t w_{tj}^{'} + b_{tj}^{'}) \tag{16}$$

Here k is the number of the adaptive neurons. The proposed NN is similar to the PSNN with respect to structures of NN. The proposed model has adjustable weights substituted for fixed weights between the hidden layer and the output layer. This may provide an advantage for values of outputs in order not to be affected negatively by any emerging errors in the hidden layers or summing units. Additionally, the proposed NN has fewer parameters than MLP and PSNN for multi-class classification and fewer parameters than MLP for binary classification.

5 Algorithm 2. Training the proposed NN with MPSO

Step 1. Initialize adjustable network parameters and assign as particle's position like in (17), and initialize the velocity and keep in a vector V_i like in (18).

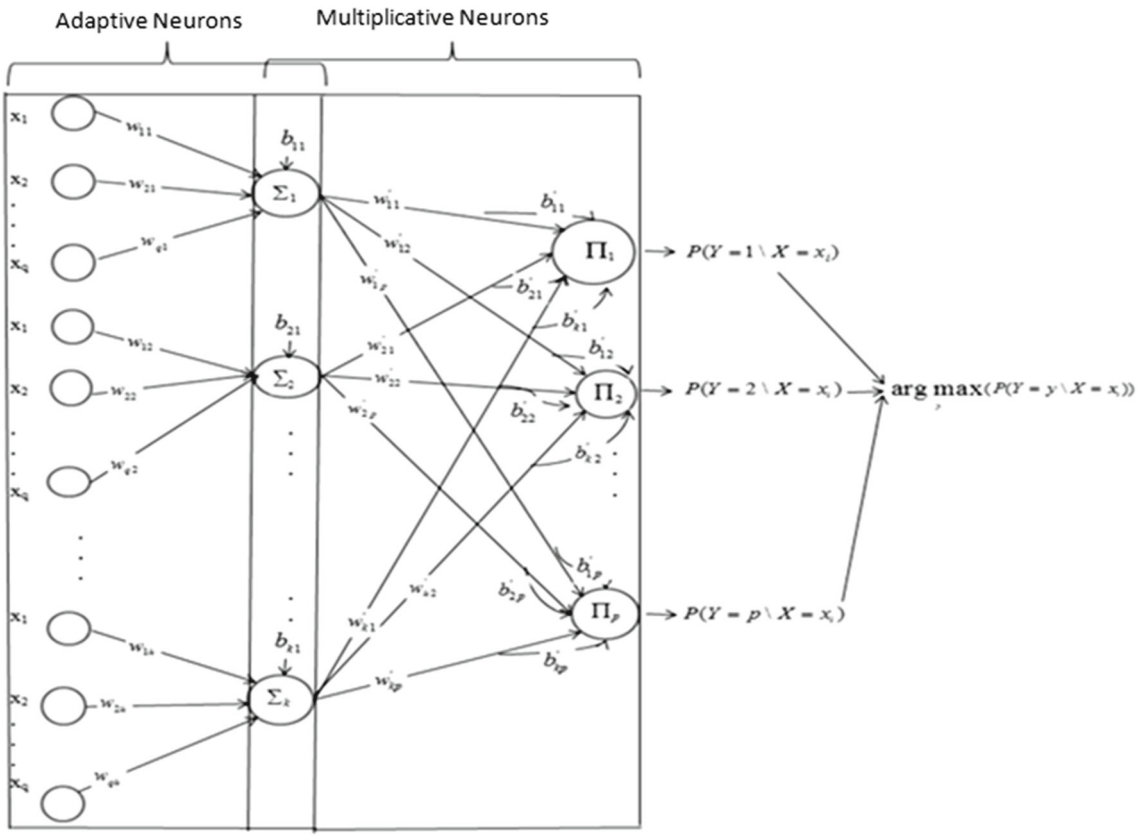

Fig. 5 The proposed neural network structure for p-class classification

$$X_{id} = \Big[(w_{11},\ldots,w_{1q}),\ldots,(w_{1k},\ldots,w_{qk}),(w'_{11},\ldots,w'_{1p}),\ldots,(w'_{k1},\ldots,w'_{kp}),(b'_{11},\ldots,b'_{k1}),\ldots,(b'_{1p},\ldots,b'_{kp})\Big] \tag{17}$$

$$V_{id} = [v_{id}]_{k(q+1)+2pk} \tag{18}$$

Step 2. Data are divided into three independent groups randomly; training, test, and validation samples, and convert these data to a range of (0,1).

Step 3. For the training sample to obtain outputs, use (16) and then softmax activation function can be used to attain probabilities. According to the error function, personal best position (*pbest*) and global best position (*gbest*) particles are calculated.

Step 4. PSO parameters are identified by using (11), (12), and (13).

Step 5. Values of velocities are updated by using (14) and values of network parameters are calculated by using (15). Additionally, a validation error is calculated to control over-fitting.

Step 6. If there is no improvement in the error function, regenerate the particle's position like in Step 1.

Step 7. The iterative process is terminated when the validation error starts to go up.

Step 8. The obtained model will be used for unseen test data to specify the accuracy of the model.

6 Experimental design and results

6.1 Design of experiments

To investigate the generalization performance of the proposed NN model on unseen data, we use well-known classification problems including binary and multi-class classification problems in the UCI repository (Blake and Merz 1998) which is shown in Table 1. For comparing above-mentioned NN classifiers and selection of the number of using adaptive neurons (k) in the proposed classifier, we apply tenfold cross-validation 30 times on Table 1 datasets. However, it should not be forgotten that with reference to the "No-Free-Lunch" theorem (Wolpert 1996), the number of k can fluctuate for different datasets for the proposed method, and we limit the number of k to 4 for avoiding hard training and time consumption. Likewise, some hidden layer units of the MLP or the number of summing units for the Pi-Sigma NN can show a change. For preserving this complication, we choose the number of hidden layer units to be 10 for the MLP and the number of

Table 1 Datasets from UCI repository

Dataset	No. of attributes	No. of classes	No. of instances
Australian credit	14	2	690
Balance scale	4	3	625
Statlog heart disease	13	2	270
Iris	4	3	150
Wisconsin (orj.)	9	2	699

the summing units to be 3 for the Pi-Sigma NN. In the design of the experiment, the maximum number of iterations is set to 200, and the number of particles used for the optimization is set to 30 for all above-mentioned NN classifiers. Also, inertia weight is set to a range of (0.95, 0.4), and cognitive and social parameters are set to a range of (1.4995, 1.2). Additionally, control factors of inertia weight are $d_1 = 0.2$ and $d_2 = 7$. We compare this proposed NN with other supervised methods by tenfold cross-validation; Linear support vector machine (SVM) with 100 support vectors note that we use one-vs-all classification method for multi-class classification; Weighted k Nearest Neighbors using Euclidean distance measure with 10 neighbors and weight distances obtained by squared inverse; Complex Tree classifier which special case of Decision Trees with the maximum number of split is 100 and split criterion is Ginis Diversity Index, and Bagged Trees with the number of learners are 200.

6.2 Experimental results

Firstly, for the investigation of the training process of the above-mentioned NN classifiers, we can check on convergence curves. Figures 6 and 7 show the convergence curves for training NN classifiers with choosing 70% of data for training on Iris and Balance Scale datasets,

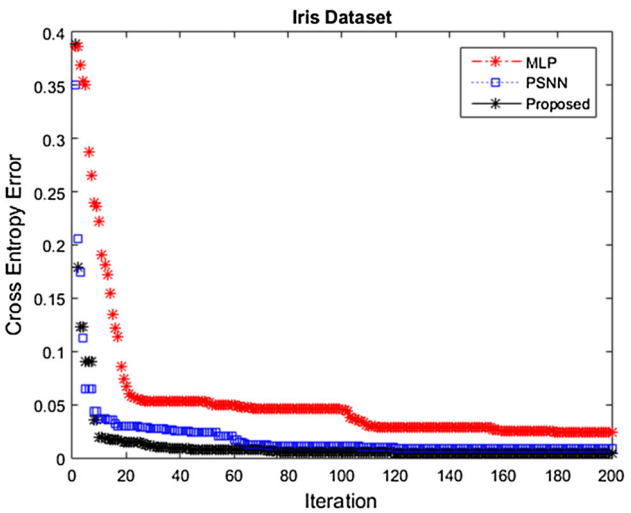

Fig. 6 Cross-entropy error curves of the NN classifiers for Iris dataset

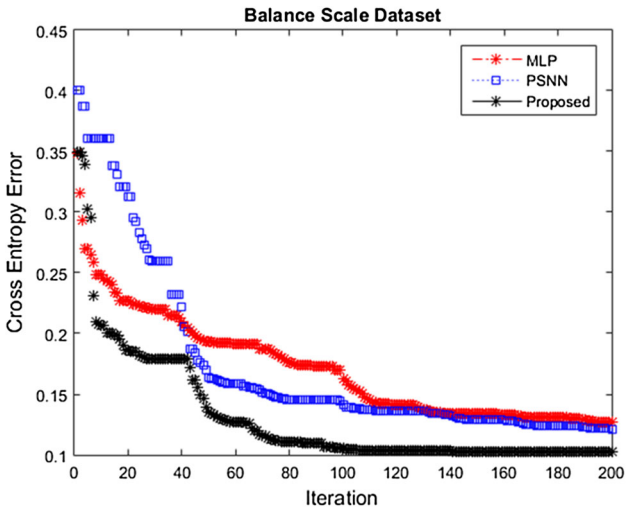

Fig. 7 Cross-entropy error curves of the NN classifiers for balance scale dataset

respectively. Our proposed classifier can converge to the solution faster than the other NN classifiers. Table 2 consisting of binary classification problems and Table 3 consisting of multi-class classification problems show the number of network parameters, mean of the prediction accuracies with standard deviation and the training times, and these tables show that our proposed NN classifier has fewer parameters and consume little time for training data in particular for multi-class classifications. In these tables, we summarize the comparison of the generalization performances of these NN classifiers using the Analysis of Variance (ANOVA) test. According to Tukey's method, which is used for multiple comparisons in statistic literature, we indicate the better NN classifier(s), which is/are signed bold, for corresponding classification problem. Obtained results have shown that using two adaptive neurons for constituting NN structure is provided equal or better classification performance than using more than two adaptive neurons, and using two adaptive neurons enables faster training too. Finally, Table 4 shows the comparison of the proposed NN classifier, which includes two adaptive neurons, with other supervised classifiers. In Table 4, the best classifier(s) for corresponding classification problems is/are signed bold according to Tukey's multiple comparison method. Hereunder this situation, our proposed

Table 2 Results of classification accuracies of tenfold cross-validation for binary classification

Australian dataset	No. of network parameters	Mean of tenfold cross validations	Time (s)
MLP-MPSO	161	0.85268 ± 0.0067	1482
PSNN-MPSO	45	0.85323 ± 0.0093	651
Proposed NN-MPSO (k = 2)	**34**	**0.85980 ± 0.0062**	**445**
Proposed NN-MPSO (k = 3)	**51**	**0.85781 ± 0.0068**	**639**
Proposed NN-MPSO (k = 4)	68	0.85517 ± 0.0076	893
Statlog (heart) disease			
MLP-MPSO	151	0.81938 ± 0.0137	489
PSNN-MPSO	42	0.81827 ± 0.0158	259
Proposed NN-MPSO (k = 2)	**32**	**0.83340 ± 0.0099**	**189**
Proposed NN-MPSO (k = 3)	**48**	**0.83198 ± 0.0100**	**272**
Proposed NN-MPSO (k = 4)	64	0.82556 ± 0.0126	349
Wisconsin B. cancer dataset			
MLP-MPSO	111	0.96175 ± 0.0039	1158
PSNN-MPSO	**30**	**0.96848 ± 0.0030**	**650**
Proposed NN-MPSO (k = 2)	**24**	**0.97077 ± 0.0024**	**476**
Proposed NN-MPSO (k = 3)	36	0.96686 ± 0.0034	681
Proposed NN-MPSO (k = 4)	48	0.96636 ± 0.0036	877

Table 3 Results of classification accuracies of tenfold cross-validation for multi-class classification

Balance dataset	No. of network parameters	Mean of tenfold cross validations	Time (s)
MLP-MPSO	83	0.89104 ± 0.008	3452
PSNN-MPSO	45	0.88672 ± 0.007	1844
Proposed NN-MPSO (k = 2)	**22**	**0.89872 ± 0.006**	**1219**
Proposed NN-MPSO (k = 3)	33	0.89249 ± 0.006	1837
Proposed NN-MPSO (k = 4)	44	0.89211 ± 0.006	2142
Iris dataset	**No. of network parameters**	**Mean of tenfold cross validations**	**Time (s)**
MLP-MPSO	83	0.95378 ± 0.011	923
PSNN-MPSO	45	0.95266 ± 0.011	470
Proposed NN-MPSO (k = 2)	**22**	**0.96778 ± 0.007**	**205**
Proposed NN-MPSO (k = 3)	**33**	**0.96711 ± 0.011**	**281**
Proposed NN-MPSO (k = 4)	44	0.96220 ± 0.013	356

Table 4 Results of tenfold cross-validation for the supervised classifiers

Supervised methods	Australian	Statlog (heart)	Wisconsin	Balance	Iris
Linear SVM	0.8539 ± 0.0020	**0.83690 ± 0.0047**	0.96743 ± 0.003	0.8780 ± 0.0047	0.9263 ± 0.0092
Weighted KNN	0.8338 ± 0.0047	**0.83757 ± 0.0077**	0.96847 ± 0.002	0.89247 ± 0.0063	0.9578 ± 0.0049
Complex tree	0.8374 ± 0.0106	0.756 ± 0.0171	0.94910 ± 0.006	0.77423 ± 0.001	0.9505 ± 0.0071
Bagged tree	**0.8684 ± 0.0054**	**0.8304 ± 0.009**	**0.97203 ± 0.002**	0.8435 ± 0.004	0.9500 ± 0.0075
Proposed method	0.8598 ± 0.0062	**0.8340 ± 0.0100**	**0.97077 ± 0.002**	**0.89872 ± 0.007**	**0.9678 ± 0.0074**

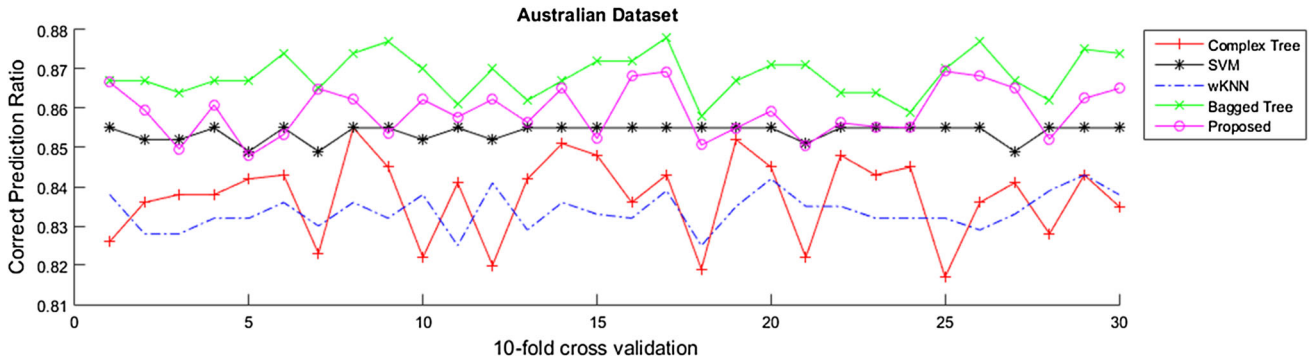

Fig. 8 Comparison of cross-validation results for Australian dataset for supervised methods

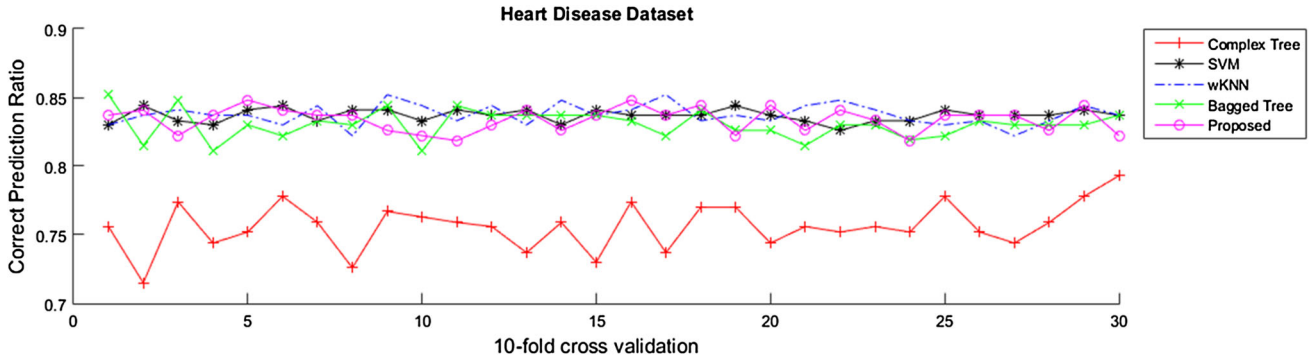

Fig. 9 Comparison of cross-validation results for heart disease for supervised methods

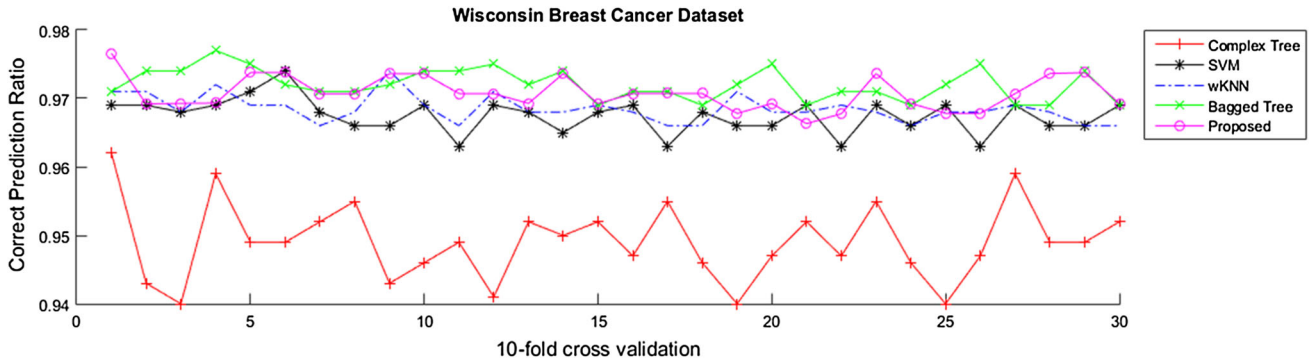

Fig. 10 Comparison of cross-validation results for Wisconsin for supervised methods

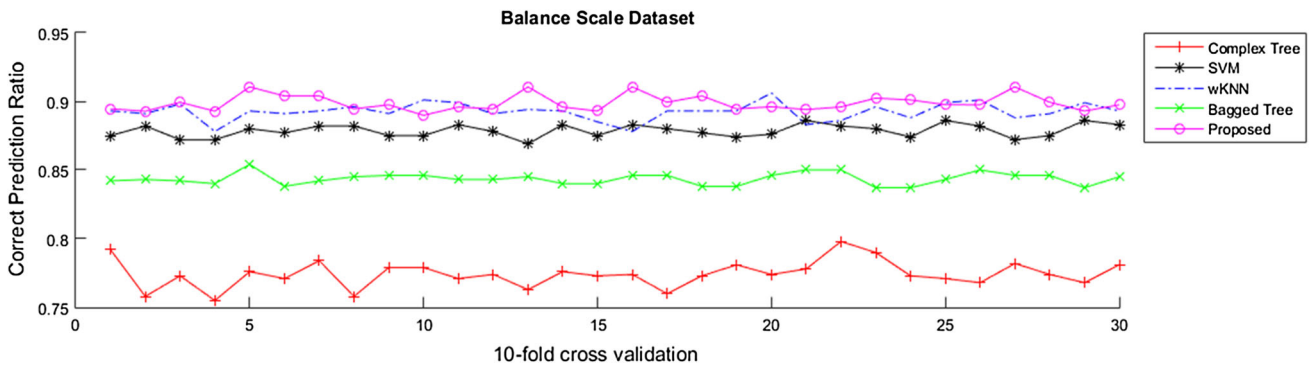

Fig. 11 Comparison of cross-validation results for balance scale for supervised methods

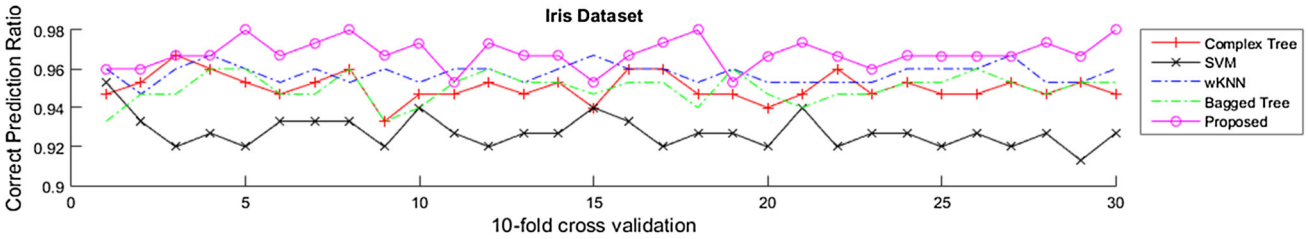

Fig. 12 Comparison of cross-validation results for Iris dataset for supervised methods

Table 5 Interpretation of Phi and Cramer's V

Phi and Cramer's V	Interpretation
> 0.25	Very strong
> 0.15	Strong
> 0.10	Moderate
> 0.05	Weak
> 0	No or very weak

method has equal or better performance than conventional NN classifier and other machine learning classifiers for Heart Disease Dataset, Wisconsin Breast Cancer Dataset, Balance Scale Dataset, and Iris Dataset. For the Australian Credit Approval dataset, the best classification method is Bagged Tree algorithm. Also, Figs. 8, 9, 10, 11, and 12 show the results of tenfold cross-validation for the above-mentioned machine learning classifiers in Table 1 datasets. Regarding these figures, our proposed method has equal or higher accuracies than other supervised methods for datasets excluding the Australian Credit Approval dataset in Table 1. These results show that our proposed NN can provide higher generalization performance compared with above-mentioned supervised classifiers and the proposed training algorithm MPSO can avoid premature convergence using decreasing inertia weights and time-varying acceleration coefficients with regeneration strategy.

7 Explainable results of the proposed method

In this paper, we used Statlog Heart Disease medical data to investigate the explainability of the proposed method. Firstly, the classes to which each observation belongs for each variable/attribute were estimated using the proposed method by keeping the other variables constant. In this way, it was revealed whether each attribute is statistically significantly correlated to the response variable for the diagnosis of heart disease.

Phi coefficient is a measure for the strength of an association between two categorical variables in a 2×2 contingency table while Cramer's V is used for bigger than 2×2 tables (Akoglu 2018). Interpretations of these correlation coefficients, especially in the medical field, are shown in Table 5 which is referred to by Akoglu (2018), and the results of classification are given in Table 6.

In Table 6, it can be said that for some variables, it has a very strong correlation (Phi > 0.25 and $P < 0.05$) with the response variable. A contingency table could not be created because all observations were classified in the same class for the 6th variable. In addition, it is observed that the correct classification rate of these variables is higher than other variables. However, the interpretation of the correlation coefficient may differ in each scientific area. Therefore, authors should pay attention to these changes according to their field of study.

Table 6 Interplatation of statlog heart disease data

No	Phi	*P*-value	Accuracy	No	Phi	*P*-value	Accuracy
1	0.0951	0.3921	0.4321	8	0.3270	0.0033*	0.6667
2	0.1603	0.1490	0.5802	9	0.5597	4.72e-07*	0.7778
3	0.5805	1.75e-07*	0.7901	10	0.4167	1.76e-04*	0.7160
4	0.3016	0.0066*	0.6049	11	0.4577	3.80e-05*	0.7160
5	0.1337	0.2287	0.6049	12	0.4962	7.97e-06*	0.7531
6	NaN	NaN	0.5556	13	0.5225	2.57e-06*	0.7654
7	0.2771	0.0126*	0.6173				

*$P < 0.05$

8 Conclusions

Neural Networks are one of the most popular methods for solving classification problems. The most used classifier is the multilayer perceptron. However, it has some disadvantages like the slow convergence rate or determining its topology. There are many studies available in the literature to train neural networks and improve training performance.

In all machine learning methods, the black-box approach brings with it some uncertainty. It is obvious that the machine learning method used must be interpretable.

In this study, we proposed a new neural network classifier trained by a modified particle swarm optimization algorithm for handling classification problems. Also, we tried to explain the explainability of the method we used with an example using the correlation coefficient. We found that some variables have a high correlation with classifier variables. In future studies, correlation coefficients can be used to interpret artificial intelligence. But it should not be ignored that the strength of correlation for each field of study can be interpreted differently.

Author contribution Taner Tunç conceived of the presented idea. Erdinç Kolay performed the computation. Both Taner Tunç ve Erdinç Kolay authors contributed to the final version of the manuscript.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Akoglu H (2018) User's guide to correlation coefficients. Turk J Emerg Med 18:91–93. https://doi.org/10.1016/j.tjem.2018.08.001

Bishop CM (2006) Pattern recognition and machine learning. Academic Press, New York

Blake CL, Merz CJ (1998) UCI Repository of machine learning databases. University of California. http://archive.ics.uci.edu/ml/

Chuang C (2011) Huang S A hybrid neural network approach for credit scoring. Exp Syst 28:185–196. https://doi.org/10.1111/j.1468-0394.2010.00565.x

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the MHS'95 sixth ınternational symposium micro machine human science, pp 39–43. doi:https://doi.org/10.1109/MHS.1995.494215.

Engelbrecht AP (2001) Cooperative learning in neural networks using particle swarm optimizers. Annu Res Conf South Afr Inst Comput Sci Inf Technol, pp 84–90

Faris H, Aljarah I, Mirjalili S (2016) Training feedforward neural networks using multi-verse optimizer for binary classification problems. Appl Intell. https://doi.org/10.1007/s10489-016-0767-1

Ghazali R (2009) Application of pi-sigma neural networks and ridge polynomial neural networks to financial time series prediction, 271–273.

Ghosh J, Shin Y (1995) Ecient higher-order neural networks for classification and function approximation. Int J Neural Syst 03:323–350

Hagan MT, Menhaj MB (1994) Training feedforward networks with the marquardt algorithm. IEEE Trans Neural Netw 5:989–993. https://doi.org/10.1109/72.329697

Hakan C, Erol A (2013) Robust multilayer neural network based on median neuron model. Neural Comput Appl. https://doi.org/10.1007/s00521-012-1315-5

Hatamlou A (2017) A hybrid bio-inspired algorithm and its application. Appl Intel. https://doi.org/10.1007/s10489-017-0951-y

Husaını NA, Ghazalı R, Nawı NM, Ismaıl LH (2012) The effect of network parameters on pi-sigma neural network for temperature forecasting. Int J Mod Phys Conf Ser 09:440–447. https://doi.org/10.1142/S2010194512005521

Hussain AJ, Liatsis P (2003) Recurrent pi-sigma networks for DPCM image coding. Neurocomputing 55:363–382. https://doi.org/10.1016/S0925-2312(02)00629-X

Ilonen J, Kamarainen J-K, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17:93–105. https://doi.org/10.1023/A:1022995128597

Jia W, Zhao D, Shen T, Ding S, Zhao Y, Hu C (2015) An optimized classification algorithm by BP neural network based on PLS and HCA. Appl Intel. https://doi.org/10.1007/s10489-014-0618-x

Juang C-F (2004) A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern B Cybern 34:997–1006. https://doi.org/10.1109/TSMCB.2003.818557

Karaboga D, Akay B, Ozturk C (2007) Artificial bee colony (ABC) optimization algorithm for training feed-forward neural. Int Conf Model Decis Artif Intel 4617:318–329

Khan K, Sahai A (2012) A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context. Int J Intell Syst Appl 4(2012):23–29. https://doi.org/10.5815/ijisa.2012.07.03

Kolay E, Tunç T, Eğrioğlu E (2016) Classification with some artificial neural network classifiers trained a modified particle swarm optimization. Am J Intel 6:59–65. https://doi.org/10.5923/j.ajis.20160603.01

Li CK (2003) A sigma-pi-sigma neural network (SPSNN). Neural Process Lett 17:1–19. https://doi.org/10.1023/A:1022967523886

Li H (2009) Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation, pp 66–69. doi:https://doi.org/10.1109/ICIC.2009.24.

Lim C, Leong J, Kuan M (2005) A hybrid neural network system for pattern classification tasks with missing features. IEEE Trans Pattern Anal Mach Intel 27:648–653

Marinakis Y, Marinaki M, Doumpos M, Zopounidis C (2009) Ant colony and particle swarm optimization for financial classification problems. Exp Syst Appl 36:10604–10611. https://doi.org/10.1016/j.eswa.2009.02.055

Mendes R, Cortez P, Rocha M, Neves J (2002) Particle swarms for feedforward neural network training. In: Proceedings of the 2002 International Joint Conference Neural Networks IJCNN02 Cat No02CH37290 6: 1895–1899. doi:https://doi.org/10.1109/IJCNN.2002.1007808.

Mirjalili S (2015) How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl Intell 43:150–161. https://doi.org/10.1007/s10489-014-0645-7

Montana DJ, Davis L (1989) Training feedforward neural networks using genetic algorithms. 762–767.

Nayak J, Naik B, Behera HS (2014) A hybrid PSO-GA based Pi sigma neural network (PSNN) with standard back propagation gradient descent learning for classification. In: Proceedings of the 2014 International Conference on Controlled Instrumentation, Communication Computer Technology ICCICCT 2014. pp 878–885. doi:https://doi.org/10.1109/ICCICCT.2014.6993082.

Nie Y (2008) A hybrid genetic learning algorithm for Pi-sigma neural network and the analysis of its convergence. pp 19–23. doi:https://doi.org/10.1109/ICNC.2008.896.

Nienhuis B (1992) Improving the convergence of the back-propagation algorithm. Neural Netw 5:465–471

Panigrahi S, Bhoi AK, Karali Y (2013) A modified differential evolution algorithm trained pi-sigma neural network for pattern classification. Int J Soft Comput Eng 3:133–136

Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8:240–255. https://doi.org/10.1109/TEVC.2004.826071

Riedmiller M (1993) A direct adaptive method for faster backpropagation learning: the RPROP algorithm. IEEE Int Conf Neural Netw

Rumelhart DE, Hinton GE, Williams RJ (1988) Learning ınternal representations by error propagation. Read Cogn Sci. https://doi.org/10.1016/B978-1-4832-1446-7.50035-2

Shekara S, Reddy S, Wang L, Devabhaktuni V, Nelapati P (2011) A hybrid neural network model and encoding technique for enhanced classification of energy consumption data. pp 1–8.

Shin Y, Ghosh J (1991) The pi-sigma network: an efficient higher-order neural network for\npattern classification and function approximation. IJCNN-91-Seattle Int Jt Conf Neural Netw. doi:https://doi.org/10.1109/IJCNN.1991.155142.

Tunc T (2012) A new hybrid method logistic regression and feedforward neural network for lung cancer data. Math Problem Eng. https://doi.org/10.1155/2012/241690

Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. Neural Comput 8:1341–1390. https://doi.org/10.1162/neco.1996.8.7.1341

Yadav RN (2003) Classification using single neuron. doi:https://doi.org/10.1109/INDIN.2003.1300258

Yadav RN, Kalra PK, John J (2007) Time series prediction with single multiplicative neuron model. Appl Soft Comput 7:1157–1163. https://doi.org/10.1016/j.asoc.2006.01.003

Yaghini M, Khoshraftar MM, Fallahi M (2012) Engineering applications of artificial ıntelligence a hybrid algorithm for artificial neural network training. Eng Appl Artif Intell. https://doi.org/10.1016/j.engappai.2012.01.023

Yolcu U, Egrioglu E, Aladag CH (2013) A new linear and nonlinear artificial neural network model for time series forecasting. Decis Support Syst 54:1340–1347. https://doi.org/10.1016/j.dss.2012.12.006

Zhang J, Zhang J, Lok T, Lyu MR (2007) A hybrid particle swarm optimization: back-propagation algorithm for feedforward neural network training. Appl Math Comput 185:1026–1037. https://doi.org/10.1016/j.amc.2006.07.025

Zhao L, Yang Y (2009) PSO-based single multiplicative neuron model for time series prediction. Exp Syst Appl 36:2805–2812. https://doi.org/10.1016/j.eswa.2008.01.061