Neurocomputing 247 (2017) 173–182

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A niching evolutionary algorithm with adaptive negative correlation learning for neural network ensemble

Weiguo Sheng[a,b,*], Pengxiao Shan[b], Shengyong Chen[c,b], Yurong Liu[d,e], Fuad E. Alsaadi[e]

[a] Department of Computer Science, Hangzhou Normal University, Hangzhou 310036, PR China
[b] College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou 310023, PR China
[c] College of Computer Science, Tianjin University of Technology, Tianjing 300384, PR China
[d] Department of Mathematics, Yangzhou University, Yangzhou 225002, PR China
[e] Communication Systems and Networks (CSN) Research Group, Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia

ARTICLE INFO

Article history:
Received 13 October 2016
Revised 19 March 2017
Accepted 23 March 2017
Available online 29 March 2017

Communicated by Ma Lifeng Ma

Keywords:
Neural network ensemble
Evolutionary algorithm
Negative correlation learning
Adaptation strategy
Diversity measure

ABSTRACT

This paper proposes a niching evolutionary algorithm with adaptive negative correlation learning, denoted as NEA_ANCL, for training the neural network ensemble. In the proposed NEA_ANCL, an adaptive negative correlation learning, in which the penalty coefficient λ is set to dynamically change during training, has been developed. The adaptation strategy is based on a novel population diversity measure with the purpose of appropriately controlling the trade-off between the diversity and accuracy in the ensemble. Further, a modified dynamical fitness sharing method is applied to preserve the diversity of population during training. The proposed NEA_ANCL has been evaluated on a number of benchmark problems and compared with related ensemble learning algorithms. The results show that our method can be used to design a satisfactory NN ensemble and outperform related works.

1. Introduction

The term “ensemble” is a paradigm that combines a set of machine learning algorithms to solve a problem. In the field of machine learning, ensemble learning [1] has attracted much attention as it can improve the generalization ability of classifiers. Neural network (NN) ensemble is the process of creating a set of NNs, which are then trained and combined, as opposed to create just one single NN. This process has two goals, that is, to generate accurate yet diverse NN individuals utilizing training algorithms and to improve the generalization performance by combining the individuals. Many studies [2,3] have shown that NN ensemble can lead to a considerable improvement in generalization performance. This is due to the complementary among NN individuals for the same decision. Therefore, NN ensemble has been widely applied in many applications, such as scientific image analysis [4], face recognition [5] and medical diagnosis [6].

Both theoretical and empirical studies [7-10] have demonstrated that the performance of NN ensemble depends greatly on the accuracy as well as diversity of NN individuals. Here, the diversity is regarded as the degree of different decision made by ensemble members. Generally, a better performance of NN ensemble can be achieved with more diverse NN individuals in the ensemble. Hence, maintaining the diversity among NN individuals during training is essential [51]. The key relies on how to generate a population of diverse NN individuals and how to select diverse NN individuals from the population during training.

A number of algorithms [11-17] are existed for producing diverse and accurate NN individuals in the ensemble. These methods can be broadly divided into two categories according to whether NN individuals in an ensemble are trained independently or sequentially. Bagging [11] and boosting [12,13] are two well-known ensemble learning algorithms, which are based on independent and sequential training, respectively. The bagging algorithm firstly creates M NN individuals, and then generates M different training sets by forming bootstrap replicates of the original training data. These M individuals are trained independently on the M different training sets. By contrast, the boosting algorithm trains a series of NN individuals sequentially on different training sets which are determined by the performance of previous training sets. These two algorithms can produce uncorrelated error among NN individuals. However, they are not necessarily to produce negatively correlated errors due to the internal correlation among individual NNs are not exploited [18,19].

* Corresponding author at: Department of Computer Science, Hangzhou Normal University, Hangzhou 310036, PR China.
E-mail address: weiguouk@hotmail.com (W. Sheng).

http://dx.doi.org/10.1016/j.neucom.2017.03.055
0925-2312/

Unlike the above algorithms, Liu and Yao [14,18] proposed the negative correlated learning (NCL) algorithm to simultaneously train NN individuals in an ensemble. This is achieved by introducing a correlation penalty term to the error function of each individual such that individuals can interactively and cooperatively learn different aspects of training examples, thus producing negatively correlated errors among NN individuals. Since NCL is capable of explicitly creating diverse errors in the ensemble, it has been widely combined with evolutionary algorithms (EAs) to train NN ensembles [17,20,21]. In these methods, during evolutionary process, NN individuals are generally evaluated based on multi-objective functions. These functions can be used to promote the diversity and, as the same time, encourage collaboration among the individuals. However, a key issue of NCL is that its performance depends crucially on the setting of penalty coefficient λ ($0 \leq \lambda \leq 1$) [21-23]. The coefficient is used to control the negative correlated strength among individuals in the ensemble. Set $\lambda = 1$ or a large positive value corresponds to treating the ensemble as a single estimator and considering the empirical training error without negative correlated learning. While set $\lambda = 0$ or a small positive value corresponds to training the individuals independently without negatively correlated learning. To achieve a satisfactory performance, the value of λ should be set appropriately. Unfortunately, there is no consistent approach to set such a crucial value. By employing cross-validation as usually adopted in existing methods is extremely expensive.

In this work, we present an adaptive NCL to address the above issue by dynamically adjusting the value of λ during training. The idea behind the adaptive strategy is to control the trade-off between the accuracy and diversity in the ensemble. In addition, a dynamical fitness sharing technique is introduced to create different species, therefore preserving the population diversity further during training. As a result, an algorithm called niching evolutionary algorithm with adaptive negative correlation learning has been developed for NN ensemble learning. Our experimental results show the proposed adaptive NCL is able to significantly improve the effectiveness of NN ensemble learning while the resulting algorithm can deliver satisfactory NN ensembles.

The key contributions of this work are summarized as follows:

- An adaptive negative correlation learning mechanism, in which the penalty coefficient λ is set to dynamically change based on a novel population diversity measure, is developed to appropriately control the trade-off between the diversity and accuracy in the ensemble;
- A dynamical fitness sharing is introduced and incorporated into the resulting algorithm to preserve the population diversity.

The remainder of this paper is organized as follows. In Section 2, we discuss related work to train NN ensembles. Section 3 presents our proposed algorithm. This is followed by a description of adaptive NCL in Section 4. Section 5 evaluates the proposed algorithm and compares it with related ensemble learning algorithms. Finally, Section 6 concludes the paper with a brief summary.

2. Related work

NN ensemble is first introduced by Hansen and Salamon [9]. It has been shown that the generalization ability of a NN can be significantly improved through combining a number of NNs. Many studies [24,25] reveal that one of important factors in constructing NN ensembles is that the diversity among NN individuals, whose errors should be uncorrelated. A lot of methods [12-14,17-19,21,23,26,27,50] have been proposed to create diverse NN individuals in an ensemble. These methods can be roughly divided into two categories depending on whether the NN ensembles are trained by varying training data or algorithms.

Bagging [26] and boosting [12,13] are perhaps the two most popular methods, which are based on varying the data for training NN ensembles. The bagging is first proposed by Breiman [11] and has been widely used for ensemble learning. While the original version of boosting is developed by Freund [28]. Based on this method, many improved versions have also been designed. For example, in [29], an method called Adaboost was developed for alleviating overfitting issue of the original boosting. In [12,30], Freund proposed two improved versions of boosting, which can obtain a better performance than the original one. Both bagging and boosting algorithms can be used to train NN individuals in the ensemble where the errors are uncorrelated. However, the NN individuals created by these methods are not necessarily negative correlated.

To address the above issue, Liu and Yao [18] proposed the NCL, which falls into the second category, to train the NN ensemble. NCL encourages NN individuals in an ensemble to simultaneously learn different aspects of training examples. This learning algorithm trains NN individuals by incorporating a correlation penalty term into the error function and can produce biased NN individuals whose errors are negative correlated. In [31], Rosen et al. showed that de-correlated NN ensembles can achieve a good performance. Based on this observation, Liu et al. [18] presented a cooperative ensemble learning system (CELS) to train NN ensembles. In CELS, NCL is used to create negative correlated NNs. Liu et al. [15] also designed evolutionary ensembles with negative correlation learning, which encourages NNs in ensembles to interactively and cooperatively learn different aspects of the training data. Following this approach, Chandra et al. [20] developed a diverse and accurate ensemble learning algorithm (DIVACE) which is based on a multi-objective evolutionary algorithm. DIVACE tries to control the trade-off between diversity and accuracy in the ensemble by optimizing two objective functions (i.e., empirical error function and correlation penalty function). In [17], an algorithm combining cooperative coevolution with a multi-objective evolutionary algorithm was designed to train NN ensembles. In this algorithm, NCL is considered as the objective of diversity preservation. This algorithm considers to encourage collaboration among NNs and to improve the combination of trained NNs.

The NCL has also been incorporated into constructive algorithms to train ensembles. For example, in [27], Islam et al. presented a constructive algorithm for training cooperative NN ensembles. In this method, NCL with different training epochs is used to promote the diversity among NNs in the ensemble. In [19], Islam et al. designed two cooperative ensemble learning algorithms, called NegBagg and NegBoost, to train NN ensembles. In these two algorithms, NCL is used as partial training to train NNs in the ensemble, with the purpose of facilitating interaction and cooperation among NNs.

To avoid overfitting of noise in training data, additional regularizations has also been introduced into NCL. For example, Chen et al. [23] devised a regularized NCL algorithm for training NN ensembles. This algorithm is based on a multi-objective evolutionary algorithm, which uses Bayesian inference to address the problem of parameter setting of regularization. In this method, the penalty coefficient λ in NCL is set by employing the cross validation. Chen et al. [21] developed another multi-objective regularized NCL algorithm, which can avoid the setting of penalty coefficient. This is achieved by simultaneously optimizing three objective functions (i.e., empirical error function, correlation penalty function and regularized function).

When training NN ensembles by employing NCL, setting an appropriate penalty coefficient λ is crucial for the performance of NCL. However, there is no consistent methodology for determining the value of λ, which is usually arbitrarily set using the cross

Step 1: Generate an initial population with M NN individuals. Each NN individual is encoded using a real-number representation. The number of hidden neurons, connections and connection weights in the NN individuals are initialized randomly.

Step 2: Calculate the fitness value according to the fitness function for each NN individual in the initial population.

Step 3: Repeat the following (a) to (e) steps until the maximal number of generation is reached.

(a) Select paired parents using a modified dynamical fitness sharing method.

(b) Apply crossover and mutation on paired parents to generate intermediate offspring.

(c) Employ the adaptive NCL to update the connection weight w_{ij} of offspring according to the following equation:

$$E = \sum_{n=1}^{N}(f_i(x_n) - y_n)^2 - \lambda\sum_{n=1}^{N}(f_i(x_n) - f_{ens}(x_n))^2$$

$$\frac{\partial E}{\partial w_{ij}} = 2\sum_{n=1}^{N}(f_i(x_n) - y_n)\frac{\partial f_i}{w_{ij}} - 2\lambda\sum_{n=1}^{N}(f_i(x_n) - f_{ens}(x_n))(1-\frac{1}{M})\frac{\partial f_i}{\partial w_{ij}}$$

(d) Calculate the fitness value of each offspring according to the fitness function.

(e) Create a new population of size M from the individuals of previous population and their offspring, and rank the individuals in the current population based on their fitness values.

Step 4: Combine all NN individuals in the population to form the ensemble.

Fig. 1. The procedure of niching evolutionary algorithm with adaptive negative correlated learning for training NN ensemble.

validation. In this paper, we proposed an adaptive NCL, which can dynamically adjust the penalty coefficient depending on problem instances and training process.

3. Proposed algorithm

In this section, we propose a niching evolutionary algorithm with adaptive negative correlated learning, denoted as NEA_ANCL, for training the NN ensemble. Our proposed algorithm is distinguishable from most ensemble learning algorithms and emphasizes on controlling the trade-off between diversity and accuracy in the ensemble during training. In our algorithm, an adaptive NCL is developed to dynamically adjust the penalty coefficient depending on problem instances and training process. The adaptation strategy is based on a novel population diversity measure. Further, a modified dynamical fitness sharing method is employed to preserve the population diversity during training. The procedure of the proposed algorithm is given in Fig. 1.

In following subsections, we first describe the component of proposed algorithm including the representation of NN individuals, genetic operators, fitness function and modified fitness sharing method. Then, the details of adaptive NCL are presented in Section 4.

3.1. Representation of individuals

Three-layer feedforward neural network, which consists of one input layer, one output layer and a set of hidden neurons, is adopted to represent the NN individual, as shown in Fig. 2. Each network has N_i+N_h+N_o neurons, where N_i, N_h and N_o are the numbers of input, hidden, and output neurons, respectively. The maximum number of hidden neurons N_{max} allowable is a user-specified parameter, which is set to be $N_{max}=10$ here, whereas the number of input and output neurons is problem-dependent. W^1 represents connection weights, which is from the input layer to hidden layer, and W^2 denotes connection weights, which is from the hidden layer to output layer. B^1 and B^2 are biases for the hidden and output layers, respectively. Each NN individual is coded using the real values [32]. For example, the chromosome of a NN individual $s_0 = w_{11}^1 w_{21}^1 w_{11}^2 w_{12}^2 \ |w_{12}^1 w_{22}^1 w_{21}^2 w_{22}^2 \ |w_{13}^1 w_{23}^1 w_{31}^2 w_{32}^2 \ |b_1^1 b_2^1 b_3^1 b_1^2 b_2^2$ encodes two input neurons, two output neurons and three hidden neurons. In this chromosome, $|w_{11}^1 w_{21}^1 w_{11}^2 w_{12}^2|$ represents all

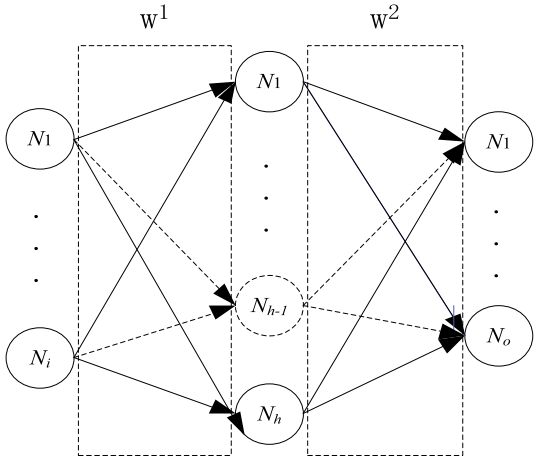


Fig. 2. A three-layer feed-forward neural network architecture.

associated connection weights of the first hidden neurons. A connection weight w_{ij} of zero means that the associated connection does not exist in the network.

3.2. Crossover and mutation

The single-point crossover operator that recombines one part of a network with another part of a network may cause the permutation problem, and hence destroys both networks [33]. To avoid such a problem, we employ a crossover operator, which is analogous to the traditional two-point crossover [34]. The operator works by selecting two individuals and then choosing one hidden neuron from each individual to exchange. For example, suppose paired parents P_1 and P_2, to be crossed, having k_1 and k_2 hidden neurons respectively. The cross points x_1 and x_2 in P_1 will be generated using functions: $x_1 = RandInt(0, N_{max}) \times (N_i + N_o)$ and$x_2 = x_1 + (N_i + N_o)$, respectively. If the value of x_2 is greater than the length of connection weights, then it is recalculated as $x_2 = x_1 - (N_i + N_o)$. This means that one hidden neuron in P_1 is chosen randomly. Similarly, x_3 and x_4 in P_2 are calculated as: $x_3 = RandInt(0, N_{max}) \times (N_i + N_o)$ and $x_4 = x_3 + (N_i + N_o)$. Then, the hidden neuron in P_2 is exchanged with the chosen hidden neu-

ron in P_1. By employing the above crossover operation, we intend to preserve the permutation of both networks. All paired parents in the population will be crossed using the above rule to generate offspring during training.

For mutation, the following three types of operator have been employed with the purpose of increasing the diversity of individual architecture.

- Gaussian mutation: the connection weights are adjusted using a low probability of Gaussian mutation, which adds a unit Gaussian-distributed random value to the chosen connection weight.
- Connection deletion: randomly selecting and deleting and existing connection.
- Connection addition: Randomly selecting a non-existing connection and a new connection is added.

3.3. Fitness function

In our method, the fitness function is defined as follows:

$$F = \sum_{n=1}^{N} (f_i(x_n) - y_n)^2 - \sum_{n=1}^{N} (f_i(x_n) - f_{ens}(x_n))^2 + \sum_{ij} \frac{{w_{ij}}^2}{1 + {w_{ij}}^2} \tag{1}$$

where the first term of equation is used to calculate the mean square error on training set. While, the second term is used to encourage NN individuals to produce negative correlated errors. The purpose of this term is to generate diverse NN individuals in the ensemble. Here, $f_{ens}(x_n)$ denotes the average fitness of combined NN individuals on the n^{th} training pattern. The third term is a weight decay, which is used to estimate the complexity of NN architectures and improve the generalization of NNs [35]. In this term, the w_{ij} denotes the connection weight from neuron i to j.

3.4. Dynamical fitness sharing

Maintaining the diversity among ensemble members is also critical when training the NN ensemble [24]. Fitness sharing [36] is a promising method to maintain the population diversity and has been used for designing NN ensembles [16]. In this paper, we introduce a modified dynamical fitness sharing [37] and integrate it into our algorithm to preserve the population diversity. The modified fitness sharing method is able to 1) adaptively determine the niche radius σ during evolution; 2) preserve the diversity with respect to the number of hidden neurons encoded in individuals; 3) encourage similar NN individuals for reproduction.

The original dynamical fitness sharing presented in [37] is based on two assumptions: there are no less than two individuals in each species and selected individuals belong to only one species. This method can dynamically identify the species and exploit the information available in the population. However, it suffers drawbacks that the value of niche radius needs to be specified beforehand [38]. In most real applications, since there is no a *priori* knowledge about fitness landscape available, it is difficult to set the niche radius appropriately. Here, we try to adaptively determine the niche radius σ in the fitness sharing method by employing the following formula:

$$\sigma = \frac{1}{2M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{ij} \tag{2}$$

where M is the population size, d_{ij} is the distance between the ith and jth networks.

When searching for solutions within the fitness landscape, it is desirable to preserve the diversity of solutions with respect to the number of hidden neurons. This is due to the performance of NNs depends crucially on their architectures [39], especially the number of hidden neurons. The individuals with different number of hidden neurons possess different learning capability, and thus produce diverse NN individuals. For this reason, during the crossover, we only allow the NN individuals with different number of hidden neurons to exchange the information of hidden neurons. By employing such a scheme, the number of hidden neurons encoded in the individuals will be preserved, so as the population diversity with respect to the number of hidden neurons. Additionally, since mating among dissimilar individuals often produces low-performance offspring, our fitness sharing method is designed to encourage mating between similar individuals. Specifically, during selection, the parent P_1 with the highest fitness value in population is selected, its mate P_2, which has different number of hidden neurons is sequentially selected based on its fitness rank. These two similar individuals are then used to generate offspring. This procedure terminates when all individuals in the population are selected.

4. Adaptive negative correlation learning

Many theoretical studies regarding to the ensemble learning have been carried out in literature [10,40,41]. These studies reveal that the diversity among population in an ensemble is critical for its performance. Many methods have been proposed to achieve a high diversity of individuals in the ensemble, among which the NCL is perhaps the most promising one. However, NCL suffers from a serious issue that its performance depends significantly on the setting of its penalty coefficient λ. To tackle such an issue, here we present an adaptive NCL that can dynamically adjust the value of λ depending on problem instances and training process. The adaptation strategy of determining λ is designed based on a novel population diversity measure.

In the following subsections, we first provide an overview of theoretical study of ensemble learning in Section 4.1. Then, a brief description of the NCL and its associate penalty coefficient is given in Section 4.2. After that, Section 4.3 presents a novel diversity measure of the population diversity. Finally, we design an adaptation strategy to determine the penalty coefficient λ in Section 4.4.

4.1. Theoretical analysis of ensemble learning

Many theoretical studies have been carried out, which show that the performance of an ensemble is closely related with the diversity among ensemble members. In [41], Geman et al. proposed a bias-variance decomposition and showed that the generalization error can be broken down into the terms of bias and variance. In this decomposition, there is a trade-off between the two terms: reducing the bias will cause an increase in variance, and vice versa. This decomposition is often applied to evaluate the performance of a single network. On the other hand, bias-variance-covariance decomposition [40] illustrates that the performance of NN ensemble depends on both the bias and variance as well as the amount of error correlation of ensemble members. Based on the above two decompositions, Krogh et al. [10] proposed an ambiguity decomposition, which clearly reveals the relationship between the diversity and accuracy in the ensemble learning: increasing the diversity in ensemble tends to reduce the accuracy, and vice versa. This decomposition has been commonly used for regression ensembles. Further, Chen et al. [25] showed that the ambiguity decomposition is applicable to classifier ensembles such as NN ensembles with zero-one loss function, and extended it to multi-class cases in [42]. In overall, the above theoretical studies indicate that the performance of ensemble methods depend greatly on how to balance

the tradeoff between the diversity and accuracy in the ensemble during training.

4.2. NCL

NCL introduces a correlation penalty term into the error function of each NN individual in the ensemble such that the NNs can learn interactively different aspects of training examples. Suppose a training data set $\{x_n, y_n\}_{n=1}^{N}$, NCL combines M NN individuals to constitute the NN ensemble according to the following equation:

$$F(x_n) = \frac{1}{M}\sum_{i=1}^{M} F_i(x_n) \tag{3}$$

where M is the number of NN individuals in the ensemble, $F_i(x_n)$ is the fitness of network i on nth training pattern and $F(x_n)$ is the fitness of the ensemble on nth training pattern. The error function e_i for ith individual in NCL is defined by

$$e_i = \sum_{n=1}^{N} (F_i(x_n) - y_n)^2 + \lambda\rho_i \tag{4}$$

where the first term in the equation is the empirical training error of ith individual. The second term ρ_i is a correlation penalty function, which can be written as:

$$\rho_i = \sum_{n=1}^{N}\left\{(F_i(x_n) - F(x_n))\sum_{j\neq i}\left(F_j(x_n) - F(x_n)\right)\right\} \tag{5}$$

The purpose of minimizing ρ_i is to negatively correlate each individual's error with the error of rest individuals in the ensemble. The penalty coefficient λ is used to balance the error term and penalty term. Here, $\lambda = 0$ means that the NN individuals are trained independently. While $\lambda = 1$ indicates the training of NCL would correspond to treating the entire ensemble as a single estimator and considering only the empirical training error. In NCL, the penalty coefficient λ is set to control the negative correlated strength of the error between an individual and the rest ones in the ensemble. A too large value λ will lead to excessive diversity among individuals in the ensemble, thus reducing its accuracy. While, a too small value λ will result in a low diversity in the ensemble and thus poor performance. Hence, setting the penalty coefficient λ appropriately is crucial.

4.3. Diversity measure

Many diversity measures have been proposed in the literature. These diversity measures can be roughly divided into two categories: pairwise based diversity measures such as Q statistic [43], Kappa statistic [44], correlation coefficient [45] and non-pairwise based diversity measures such as entropy [46] and Kohavi–Wolpert variance [47].

The pairwise diversity is based the measurement of pairwise classifier. In most cases, this type of diversity measure ignores the genetic diversity of individuals in the ensemble. Employing such a diversity measure, although the NN ensemble shows a low diversity, NN individuals can be spread widely in the architecture space. This is due to NN individuals with different NN architectures can share similar fitness value. However, NN individuals with different architectures can be trained to learn different aspects of training examples, and help to promote the population diversity. Here, we present a pairwise based diversity measure, which considers the genetic diversity as well as fitness diversity simultaneously.

Specifically, since a NN ensemble can be regarded as a population, we divide the population into b subpopulations according to the number of hidden neurons encoded in NN individuals, such that each subpopulation contains NN individuals with the same number of hidden neurons. For qth subpopulation S_q^k, $S_q^k = (s_1, s_2, \ldots, s_c)$, let c denotes the number of NN individuals in the subpopulation, k ($k \geq 2$) the number of hidden neurons encoded in its individuals. The procedure, as shown in Fig. 3, is then implemented to calculate the population diversity. In this calculation, the diversity contribution of each set of two individuals is based on their fitness as well as distance. Consequently, two individuals with high fitness that are far away with each other will have a large contribution to the population diversity. As both genetic and fitness diversity are accounted, the resulting diversity measure can therefore be used to appropriately indicate the population diversity.

Table 1
Eight benchmark classification problems.

Data set	No. of		
	examples	attributes	classes
Australian credit card	690	14	2
Diabetes	768	8	2
Glass	214	9	6
Breast cancer	699	9	2
Image	2310	18	7
Heart	303	13	2
Satellite	6435	36	7
Soybean	683	35	19

4.4. Adaptation strategy

In this subsection, we design an adaptation strategy for addressing the crucial issue of setting the penalty coefficient λ in NCL. The proposed adaptation strategy can dynamically adjust the value of λ depending on problem instances and training process. The core idea behind the proposed adaptation strategy is to determine an appropriate penalty coefficient value of λ based on the above population diversity measure, thus control the trade-off between the diversity and accuracy in the ensemble. Specifically, the adaptation strategy works as follows. At the initial training phase, a large penalty coefficient value, $\lambda = 1$, is used to create a large diversity among NN individuals in the population. This phase is terminated when the population diversity ceases to increase within a specified number of generations and the maximum population diversity of D^{max} is then recorded. After that, an adaptation strategy based on the current population diversity takes over to determine the penalty coefficient λ as follow:

$$\lambda' = \lambda_{max} - \frac{D}{D^{max}} \times (\lambda_{max} - \lambda_{min}) \tag{6}$$

where λ' denotes the current penalty coefficient value. λ_{max} and λ_{min} correspond to the upper and lower bound values of λ, which are set to be 0.9 and 0.1, respectively. So, the value of λ' will vary from λ_{min} to λ_{max}.

5. Experiments

In this section, we evaluate the performance of our proposed algorithm and compared it with related ensemble learning algorithms on eight benchmark classification problems. These data sets are from the University of California at Irvine (*UCI*) machine learning repository [48], and have different characteristics for classification. The data sets are listed in Table 1. We first evaluate our proposed algorithm on the benchmark problems. Then, we investigate the property of adaptive NCL. Finally, comparisons are carried out with related ensemble learning algorithms. The Test Error Rate (TER), commonly used to measure the performance, refers to the percentage of classification error produced by the trained NN ensembles on the testing set.

Step 1: Calculate the output distance OD_{ij} between each two individuals in the subpopulation using

$$OD_{ij} = \frac{1}{N}\sum_{n=1}^{N}\left|\left(O_i(x_n) - O_j(x_n)\right)\right|$$

where N is the number of training samples, $O_i(.)$ and $O_j(.)$ are the output of network i and j, respectively, on the n^{th} training sample.

Step 2: Compute the genetic distance GD_{ij} of each two NN individuals in the subpopulation using

$$GD_{ij} = \sqrt{\frac{1}{L}\sum_{l=1}^{L}(w_i^l - w_j^l)^2}$$

where L is the length of the chromosome, w_i^l and w_j^l are the weights of network i and j, respectively.

Step 3: Compute each two individuals' contribution to the diversity of subpopulation as

$$SD_{ij} = \frac{f_i + f_j}{2 \times f_{ave}} \times GD_{ij} \times OD_{ij}$$

where f_{ave} is the average fitness of the subpopulation, f_i and f_j are the fitness of network i and j, respectively.

Step 4: Calculate the subpopulation's diversity by summing average values of the individuals' contribution matrix, which can be written as

$$SD = \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} SD_{ij}$$

Step 5: Compute the population diversity by summing all subpopulations' diversity contributions as

$$D = \sum_{i=1}^{C} SD^i$$

Fig. 3. The calculation procedure of population diversity.

5.1. Experimental setup

In the experiments, we employ k-fold-cross-validation for deriving a good estimation of generalization error. The k-fold-cross-validation partitions a sample of data set into k subsets, and then k rounds of cross-validation are performed. In each round of cross-validation, different combination of $k-1$ subsets is used to train NN individuals in the ensemble, and the rest subset is used to test the generalization error of NN ensemble. Specifically, 10-fold-cross-validation is employed to evaluate the performance of our proposed algorithm. Unless otherwise stated, all simulation results are averaged over 30 independent runs. The input attributes of all problems are normalized to between 0 and 1 by a linear function. The output attributes of all problems are encoded using a *1-of-m* output representation for m classes. We employ a logistic sigmoid activation function for hidden and output layers. The output of NN ensemble is determined as the average output of all NN individuals.

The proposed algorithm involves a few control parameters that need to be set. The population size M is set to be 30 and the maximize number of generations is set as 100. The number of hidden neurons for each NN individual is initialized randomly but restricted in the range 3–10, and the initial connection weights are randomly chosen in the range between -1 and 1. In NCL, the learning rate and momentum for training NN individuals are 0.3 and 0.8, respectively. These values are experimentally determined and they are not meant to be the best.

First, we evaluate the performance of our proposed algorithm on eight benchmark problems. In the experiments, we use different number of adaptive NCL epochs, τ, to train NN ensembles. The results in terms of average TER and average number of hidden neurons of the identified NN ensembles are shown in Table 2. The results show that our proposed algorithm is generally capable of delivering satisfactory NN ensembles for different problems. For example, on the Diabetes and Glass data set using $\tau = 10$, the NN ensembles provided by the proposed algorithm have average number of hidden neurons of 4.839 and 5.517 and obtain the average TER of 0.204 and 0.259, respectively. It can also be observed from the results that the average TER and number of hidden neurons are influenced by the number of epochs. Typically, the results show that using a higher number of epochs will result in NN ensembles with better TER performance and also more compact NN architectures. For example, on the Australian credit card data set, using $\tau = 5$ and $\tau = 10$ result in average number of hidden neurons of 5.923 and 5.567 with average TER of 0.115 and 0.108, respectively. The reason is mainly due to that NN individuals in the ensemble have more opportunity to cooperatively and interactively learn different aspects of training data, thus generating ensembles with better performance.

Table 2
The performance of NEA_ANCL with different number of epochs on eight benchmark classification problems.

Data sets	No. of epochs	No. of hidden neurons (Std. Dev.)	TER (Std. Dev.)
Australian credit card	5	5.923(1.901)	0.115(0.0168)
	10	5.567(1.912)	0.108(0.0143)
Diabetes	5	5.220(1.307)	0.224(0.0168)
	10	4.839(1.344)	0.204(0.0158)
Glass	5	5.681(1.269)	0.280(0.258)
	10	5.517(1.845)	0.259(0.361)
Breast cancer	5	6.026(1.093)	0.0258(0.00512)
	10	5.901(1.301)	0.0229(0.00524)
Image	5	6.548(1.892)	0.0247(0.0247)
	10	6.842(1.610)	0.0251(0.0208)
Heart	5	6.381(2.423)	0.168(0.0152)
	10	6.512(2.122)	0.164(0.0163)
Satellite	5	6.924(1.435)	0.0851(0.0547)
	10	6.547(1.501)	0.0922(0.0522)
Soybean	5	6.601(1.952)	0.118(0.0102)
	10	6.807(1.482)	0.114(0.0103)

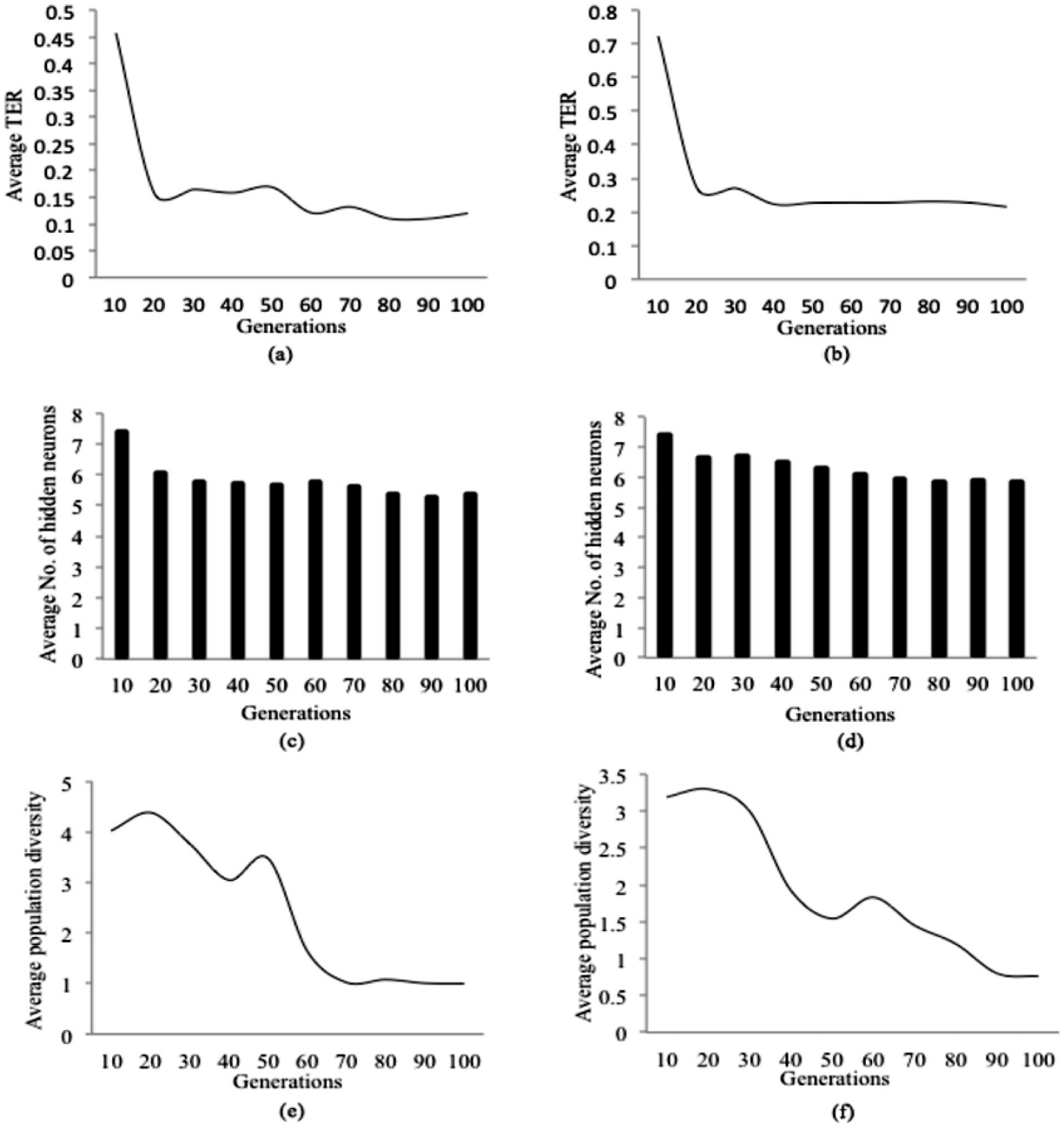


Fig. 4. The training process of NEA_ANCL. The variance of TER on (a) Australian credit problem and (b) diabetes problem. The variance of number of hidden neurons on (c) Australian credit problem and (d) diabetes problem. The variance of population diversity on (e) Australian credit problem and (f) diabetes problem.

Table 3
Comparing the performance of NEA_ANCL, NEA_NCL$_{\lambda=0}$ and NEA_NCL$_{\lambda=1}$ on three benchmark classification problems.

Data sets	Algorithm	No. of hidden neurons	TER
Breast cancer	NEA_NCL$_{\lambda=0}$	6.080(1.440)	0.0235(0.0238)
	NEA_ANCL	5.90(1.301)	0.0229(0.0254)
	NEA_NCL$_{\lambda=1}$	5.833(1.793)	0.0237(0.0214)
Australian credit card	NEA_NCL$_{\lambda=0}$	5.912(1.414)	0.128(0.0156)
	NEA_ANCL	5.567(1.412)	0.108(0.0143)
	NEA_NCL$_{\lambda=1}$	6.411(1.594)	0.116(0.0189)
Diabetes	NEA_NCL$_{\lambda=0}$	5.372(1.283)	0.221(0.0143)
	NEA_ANCL	4.839(1.344)	0.204(0.0158)
	NEA_NCL$_{\lambda=1}$	5.342(1.545)	0.209(0.0154)

In order to gain some insight of how our proposed algorithm works, we have also reported the variance of average TER, average number of hidden neurons and population diversity during the training process. The results on the Australian credit card and diabetes data sets are shown in Fig. 4. On the Australian credit card and diabetes data, it can be observed from Fig. 4(a) and (b) that, the average TER reduces quickly at the beginning of training process. However, this is not the case in terms of hidden neurons and population diversity. In fact, for the population diversity, it decreases much slower (see Fig. 4(e) and (f)) than that of TER. This could reveal that our proposed algorithm is capable of preserving the population diversity during training. For the average hidden neurons, although it fluctuates slightly during training (see Fig. 4(c)

Table 4
Comparing the performance of six algorithms: NEA_ANCL, NEA_NCL$_{CV}$, EPNet, bagging, adaboosting and MRNCL, on eight benchmark classification problems.

Data sets	Methods					
	NEA_ANCL	NEA_NCL$_{CV}$	EPNet	Bagging	Adaboosting	MRNCL
Australian credit card	**0.108 (0.0143)**	0.112 (0.0144)	0.129 (0.0147)	0.142 (0.0148)	0.152 (0.0153)	0.115 (0.0152)
Diabetes	**0.204 (0.0158)**	0.212 (0.0158)	0.231 (0.0152)	0.0232 (0.0160)	0.241 (0.0161)	0.227 (0.0153)
Glass	**0.259 (0.361)**	0.266 (0.368)	0.445 (0.412)	0.335 (0.387)	0.332 (0.391)	0.301 (0.385)
Breast cancer	**0.0229 (0.0052)**	0.0238 (0.0067)	0.0361 (0.0072)	0.0341 (0.0063)	0.0369 (0.0066)	0.0238 (0.0068)
Image	**0.0251 (0.0208)**	0.0271 (0.0233)	–	0.0338 (0.0267)	0.0312 (0.0261)	0.263 (0.0241)
Heart	0.164 (0.0163)	0.168 (0.0169)	0.172 (0.0181)	0.184 (0.0160)	0.211 (0.0168)	**0.159 (0.0149)**
Satellite	**0.0922 (0.0522)**	0.104 (0.0531)	–	0.120 (0.0568)	0.114 (0.0563)	0.0994 (0.0592)
Soybean	0.114 (0.0103)	0.118 (0.0106)	–	0.0824 (0.0042)	**0.0782 (0.0041)**	0.119 (0.0108)

and (d)), a large standard deviation of the number of hidden neurons exists in the ensemble. This turns out that the diversity with respect to the number of hidden neurons can also be maintained during the training process.

Next, we conduct experiments to investigate the effect of adaptive NCL in our proposed algorithm, using $\tau = 10$. For this purpose, we compared NEA_ANCL with its two variants: NEA_NCL$_{\lambda=0}$ with a fixed penalty coefficient $\lambda=0$ and NEA_NCL$_{\lambda=1}$ with a fixed penalty coefficient $\lambda=1$. Table 3 shows that, as compared with the two variants, NEA_ANCL generally performs the best in terms of the average TER and number of hidden neurons. On the Australian credit card and diabetes data, NEA_NCL$_{\lambda=1}$ achieves a better performance than NEA_NCL$_{\lambda=0}$. This is due to the NEA_NCL$_{\lambda=1}$ is able to interactively train NN individuals in the ensemble. While, the NEA_ANCL, which employs adaptive NCL, can deliver an even better performance than the NEA_NCL$_{\lambda=1}$. This is mainly due to the employment of adaptive NCL, which can dynamically and appropriately adjust the penalty coefficient during training. On the breast cancer data, all three algorithms have comparable performance. This is because the breast cancer data is relatively easy for classification.

Finally, we compare the proposed NEA_ANCL with the NEA_NCL$_{CV}$, EPNet [49], Bagging [11], Adaboosting [29] and MRNCL [21]. Before discussing the results, we first briefly describe the algorithms to be compared. The NEA_NCL$_{CV}$ is a variant of the proposed NEA_ANCL, in which the penalty coefficient λ is set using the cross validation. In NEA_NCL$_{CV}$, the range of λ is set to be {0.1, 0.2, 0.3…, 0.9}. The Bagging and Adaboosting are the popular ensemble learning algorithms, which train NN ensemble by varying the training data. The MRNCL employs evolutionary multiobjective algorithm to train the NN ensemble along with the NCL to preserve diversity among NN individuals. In MRNCL, an additional regularization is incorporated for addressing the noisy problem. To show the advantages of our proposed algorithm, the EPNet, which is used to design single NN, has also been included for comparison. In all experiments, *10*-fold cross validation is employed except the EPNet, which is examined using the training, validation and testing sets. For both Bagging and Adaboosting, 100 NN individuals are used to constitute the NN ensemble for comparison purpose.

Table 4 shows the results of average TER of the six algorithms to be compared. It can be observed from Table 4 that NEA_ANCL achieves the lowest average TER on six out of the eight data sets. Specifically, compared with NEA_NCL$_{CV}$, the NEA_ANCL can deliver ensembles with better average TER on all data sets. This is mainly due to the employment of adaptive NCL, in which the penalty coefficient is set to dynamically adjust depending on problem instances and training process. Similar results can also be found by comparing the NEA_ANCL with EPNet, which is a popular method used for designing single NN. This may indicate that ensembles can have better generalization performance than single NN. Compared with Adaboosting and Bagging, NEA_ANCL gives lower average TERs on all data sets except the soybean data. This is due to NEA_ANCL trains NN individuals interactively and cooperatively, thus producing negative correlated errors among NN individuals. Compared with MRNCL, NEA_ANCL also achieves lower average TERs on all data sets except the heart data. Based on the above results, we therefore conclude that our proposed algorithm is generally able to outperform the related methods to be compared.

6. Conclusions

As NN ensemble can improve the generalization of NN, it has become a popular research topic in the NN community. The negative correlation learning (NCL) is a promising NN ensemble algorithm. However, its performance depends highly on the setting of its penalty coefficient λ, while how to set this critical parameter remains an open issue. This work firstly tries to address this issue by presenting an adaptive NCL, in which the value of λ is set to dynamically change based on a novel population diversity measure. Then, in order to improve the population diversity during the evolutionary NN ensemble learning, a dynamic sharing fitness technique is further introduced and incorporated. The significance of the resulting algorithm as well as the adaptive NCL has been clearly showed in the experiments. Certainly, the proposed algorithm can be further extended and justified to design, e.g., deep neural network architectures [52], a work we would be interest to carry out in the future.

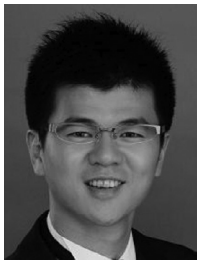
Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61573316, U1509207, 61325019, 61374010).

Reference

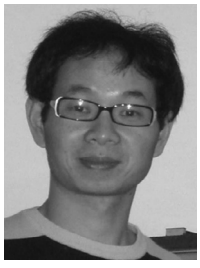
[1] T.G. Dietterich, Ensemble learning, The Handbook of Brain Theory and Neural Networks, 2, 2002, pp. 110–125.
[2] Z.H. Zhou, J. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artif. Intell. 137 (1) (2002) 239–263.

[3] Z.H. Zhou, S. Chen, Neural network ensemble, Chin. J. Comput. Chin. Edit. 25 (1) (2002) 1–8.
[4] F. Moretti, S. Pizzuti, S. Panzieri, Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling, Neurocomputing 167 (2015) 3–7.
[5] W.A. Yang, W. Zhou, Autoregressive coefficient-invariant control chart pattern recognition in autocorrelated manufacturing processes using neural network ensemble, J. Intell. Manuf. 26 (6) (2015) 1161–1180.
[6] R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing 157 (2015) 231–242.
[7] D.W. Opitz, J.W. Shavlik, Generating accurate and diverse members of a neural-network ensemble, Adv. Neural Inform. Process. Syst. (1996) 535–541.
[8] D.W. Opitz, J.W. Shavlik, Actively searching for an effective neural network ensemble, Connect. Sci. 8 (3-4) (1996) 337–354.
[9] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 10 (1990) 993–1001.
[10] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, Adv. Neural Inform. Process. Syst. 7 (1995) 231–238.
[11] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[12] Y. Freund, and R.E. Schapire, "Experiments with a new boosting algorithm." pp. 148–156, 1996.
[13] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Mach. Learn. 37 (3) (1999) 297–336.
[14] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Netw. 12 (10) (1999) 1399–1404.
[15] Y. Liu, X. Yao, T. Higuchi, Evolutionary ensembles with negative correlation learning, IEEE Trans. Evolut. Comput. 4 (4) (2000) 380–387.
[16] K.J. Kim, S.B. Cho, Evolutionary ensemble of diverse artificial neural networks using speciation, Neurocomputing 71 (7) (2008) 1604–1618.
[17] N. García-Pedrajas, C. Hervás-Martínez, D. Ortiz-Boyer, Cooperative coevolution of artificial neural network ensembles for pattern classification, IEEE Trans. Evolut. Comput. 9 (3) (2005) 271–302.
[18] Y. Liu, X. Yao, Simultaneous training of negatively correlated neural networks in an ensemble, IEEE Trans. Syst., Man, Cybern. Part B 29 (6) (1999) 716–725.
[19] M.M. Islam, X. Yao, S.S. Nirjon, M.A. Islam, K. Murase, Bagging and boosting negatively correlated neural networks, IEEE Trans. Syst., Man, Cybern. Part B 38 (3) (2008) 771–784.
[20] A. Chandra, X. Yao, Diverse and accurate ensemble learning algorithm, Intelligent Data Engineering and Automated Learning-IDEAL 2004, Springer Berlin Heidelberg, 2004, pp. 619–625.
[21] H. Chen, X. Yao, Multiobjective neural network ensembles based on regularized negative correlation learning, IEEE Trans. Knowl. Data Eng. 22 (12) (2010) 1738–1751.
[22] M. Eastwood, B. Gabrys, The dynamics of negative correlation learning, J. VLSI Signal Process. Syst. Signal, Image, Video Technol. 49 (2) (2007) 251–263.
[23] H. Chen, X. Yao, Regularized negative correlation learning for neural network ensembles, IEEE Trans. Neural Netw. 20 (12) (2009) 1962–1979.
[24] G. Brown, Diversity in Neural Network Ensembles, Citeseer, 2004.
[25] H. Chen, Diversity and regularization in neural network ensembles, School of Computer Science University of Birmingham, PhD Thesis October, 2008.
[26] T.G. Dietterich, Ensemble methods in machine learning, Multiple Classifier Systems, Springer, 2000, pp. 1–15.
[27] M.M. Islam, X. Yao, K. Murase, A constructive algorithm for training cooperative neural network ensembles, IEEE Trans. Neural Netw. 14 (4) (2003) 820–834.
[28] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, Comput. Learn. Theory (1995) 23–37.
[29] G. Rätsch, T. Onoda, K.R. Müller, An improvement of adaboost to avoid overfitting, in: Process of the International Conference on Neural Information Processing, 1998.
[30] Y. Freund, Boosting a weak learning algorithm by majority, Inf. Comput. 121 (2) (1995) 256–285.
[31] B.E. Rosen, Ensemble learning using decorrelated neural networks, Connect. Sci. 8 (3-4) (1996) 373–384.
[32] A.H. Wright, Genetic algorithms for real parameter optimization, Found. Genet. Algorithms 1 (1991) 205–218.
[33] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (9) (1999) 1423–1447.
[34] D.E. Golberg, Genetic Algorithms in Search, Optimization, and Machine Learning, 1989, Addion wesley, 1989.
[35] J. Moody, S. Hanson, A. Krogh, J.A. Hertz, A simple weight decay can improve generalization, Adv. Neural Inf. Process. Syst. 4 (1995) 950–957.
[36] B. Sareni, L. Krähenbühl, Fitness sharing and niching methods revisited, IEEE Trans. Evolut. Comput. 2 (3) (1998) 97–106.
[37] A. Della Cioppa, C. De Stefano, A. Marcelli, Where are the niches? Dynamic fitness sharing, IEEE Trans. Evolut. Comput. 11 (4) (2007) 453–465.
[38] A. Della Cioppa, C. De Stefano, A. Marcelli, On the role of population size and niche radius in fitness sharing, IEEE Trans. Evolut. Comput. 8 (6) (2004) 580–592.
[39] M.M. Islam, M.A. Sattar, M.F. Amin, X. Yao, K. Murase, A new constructive algorithm for architectural and functional adaptation of artificial neural networks, IEEE Trans. Syst. Man Cybern. Part B 39 (6) (2009) 1590–1605.
[40] N. Ueda, R. Nakano, Generalization error of ensemble estimators, in: IEEE International Conference on Neural Network, 1996, pp. 90–95.
[41] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Comput. 4 (1) (1992) 1–58.
[42] S. Wang, H. Chen, X. Yao, Negative correlation learning for classification ensembles, in: IEEE International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–8.
[43] G.U. Yule, On the association of attributes in statistics: with illustrations from the material of the childhood society, &c., Philos. Trans. R. Soc. Lond. Ser. A, Contain. Papers Math. Phys. Character 194 (1900) 257–319.
[44] D.D. Margineantu and T.G. Dietterich, "Pruning adaptive boosting." pp. 211–218, 1997.
[45] K Dai, J Zhao, F Cao, A novel decorrelated neural network ensemble algorithm for face recognition, Knowl. Based Syst. 89 (2015) 541–552.
[46] P. Cunningham, J. Carney, Diversity versus quality in classification ensembles based on feature selection, Machine Learning: ECML 2000, Springer, 2000, pp. 109–116.
[47] R. Kohavi and D.H. Wolpert, "Bias plus variance decomposition for zero-one loss functions." pp. 275–83, 1997.
[48] C. Blake, C.J. Merz, UCI Repository of machine learning databases, Department of Information and Computer Science, 55, University of California, Irvine, CA, 1998.
[49] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Trans. Neural Netw. 8 (3) (1997) 694–713.
[50] Z.S. Zhao, X. Feng, Y. Lin, Evolved neural network ensemble by multiple heterogeneous swarm intelligence, Neurocomputing 149 (2015) 29–38.
[51] M.S. Cao, L.X. Pan, Y.F. Gao, Neural network ensemble-based parameter sensitivity analysis in civil engineering systems, Neural Comput. Appl. (2015) 1–8.
[52] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.



Weiguo Sheng received the M.Sc. degree in information technology from the University of Nottingham, U.K., in 2002 and the Ph.D. degree in computer science from Brunel University, U.K., in 2005. Since then, he has worked as a Researcher at the University of Kent, U.K. and Royal Holloway, University of London, U.K. In March 2011, he moved to Zhejiang University of Technology, China, where he is now a Professor in Computer Science. His research interests include evolutionary computations, data mining/clustering and machine learning.

Pengxiao Shan received the B.Sc. degree in software engineering in 2013 and M.Sc. degree in information technology in 2016, both from Zhejiang University of Technology, Hangzhou, China. He is currently worked at Industrial and Commercial Bank of China Ltd. as Data Engineer. His main research interests focus on neural network, evolutionary computation and data mining/classification.



Shengyong Chen (M'01, SM'10) received the Ph.D. degree in computer vision from City University of Hong Kong, Hong Kong, in 2003. He joined Zhejiang University of Technology, China, in Feb. 2004, where he is currently a Professor in the Department of Computer Science. He received a fellowship from the Alexander von Humboldt Foundation of Germany and worked at University of Hamburg in 2006–2007. His research interests include computer vision, robotics, and image analysis. Dr. Chen is a Fellow of IET, a senior member of IEEE, and a committee member of IET Shanghai Branch. He has published over 100 scientific papers in international journals and conferences. He received the National Outstanding Youth Foundation Award of China in 2013.

Yurong Liu was born in China in 1964. He received his B.Sc. degree in Mathematics from Suzhou University, Suzhou, China, in 1986, the M.Sc. degree in Applied Mathematics from Nanjing University of Science and Technology, Nanjing, China, in 1989, and the Ph.D. degree in Applied Mathematics from Suzhou University, Suzhou, China, in 2001.

Dr. Liu is currently a professor with the Department of Mathematics at Yangzhou University, China. He also serves as an Associate Editor of Neurocomputing. So far, he has published more than 80 papers in refereed international journals. His current interests include stochastic control, neural networks, complex networks, nonlinear dynamics, time-delay systems, multi-agent systems, and chaotic dynamics.

Fuad E. Alsaadi received the B.S. and M.Sc. degrees in electronic and communication from King AbdulAziz University, Jeddah, Saudi Arabia, in 1996 and 2002. He then received the Ph.D. degree in Optical Wireless Communication Systems from the University of Leeds, Leeds, UK, in 2011. Between 1996 and 2005, he worked in Jeddah as a communication instructor in the College of Electronics & Communication. He was a lecturer in the Faculty of Engineering in King AbdulAziz University, Jeddah, Saudi Arabia in 2005. He is currently an assistant professor of the Electrical and Computer Engineering Department within the Faculty of Engineering, King Abdulaziz University, Jeddah, Saudi Arabia. He published widely in the top IEEE communications conferences and journals and has received the Carter award, University of Leeds for the best PhD. He has research interests in optical systems and networks, signal processing, Synchronization and Systems Design.