Evolutionary Ensembles with Negative Correlation Learning

Yong Liu
The University of Aizu
Tsuruga, Ikki-machi, Aizu-Wakamatsu
Fukushima 965-8580
Japan
Email: yliu@u-aizu.ac.jp

Xin Yao
School of Computer Science
The University of Birmingham
Edgbaston, Birmingham B15 2TT
U.K.
Email: x.yao@cs.bham.ac.uk

Tetsuya Higuchi
Evolvable Systems Laboratory
Computer Science Division, mbox 1501
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki 305-8568
Japan
Email: higuchi@etl.go.jp

Abstract

Based on negative correlation learning and evolutionary learning, this paper presents evolutionary ensembles with negative correlation learning (EENCL) to address the issues of automatic determination of the number of individual neural networks (NNs) in an ensemble and the exploitation of the interaction between individual NN design and combination. The idea of EENCL is to encourage different individual NNs in the ensemble to learn different parts or aspects of the training data so that the ensemble can better learn the entire training data. The cooperation and specialization among different individual NNs are considered during the individual NN design. This provides an opportunity for different NNs to interact with each other and to specialize. Experiments on two real-world problems demonstrate that EENCL can produce NN ensembles with good generalization ability.

1 Introduction

Many real-world problems are too large and too complex for a single monolithic system to solve alone. There are many examples from both natural and artificial systems that show that an

integrated system consisting of several subsystems can reduce the total complexity of the system while solving a difficult problem satisfactorily. The success of neural network (NN) ensembles in improving a classifier's generalization is a typical example [1].

NN ensembles adopt the divide-and-conquer strategy. Instead of using a single network to solve a task, an NN ensemble combines a set of NNs that learns to subdivide the task and thereby solve it more efficiently and elegantly. An NN ensemble offers several advantages over a monolithic NN [2]. First, it can perform more complex tasks than any of its components (i.e., individual NNs in the ensemble). Second, it can make an overall system easier to understand and modify. Finally, it is more robust than a monolithic NN and can show graceful performance degradation in situations where only a subset of NNs in the ensemble are performing correctly.

Given the advantages of NN ensembles and the complexity of the problems that are beginning to be investigated, it is clear that the NN ensemble method is and will be an important and pervasive problem-solving technique. However, designing NN ensembles is a very difficult task. It relies heavily on human experts and prior knowledge about the problem.

The idea of designing an ensemble learning system consisting of many subsystems can be traced back to as early as 1958 [3, 4]. Since the early 1990's, algorithms based on similar ideas have been developed in many different but related forms, such as NN ensembles [5, 2], mixtures of experts [6, 7, 8, 9], and various boosting and bagging methods [10, 11, 12]. However, all of these algorithms rely on some manual design, such as the design of individual NNs and/or division of training data. The number of individual NNs in an ensemble system is often predefined and fixed according to human experience and prior knowledge of the problem to be solved. Also, it is generally unclear what subtasks should be performed by which NNs. In the case of some speech and image processing tasks, such task decomposition was done manually. While manual design and a fixed ensemble architecture may be appropriate when there are experienced human experts with sufficient prior knowledge of the problem to be solved, it is certainly not the case for those real-world problems where we do not have much prior knowledge. Tedious trial-and-error processes are often involved in designing NN ensembles in practice.

This paper presents evolutionary ensembles with negative correlation learning (EENCL) for designing NN ensembles automatically based on negative correlation learning [13, 14, 15] and evolutionary learning [16, 1, 17]. EENCL differs from previous work on designing NN ensembles in three major aspects. First, most previous work did not acknowledge or exploit the negative correlation among individual NNs in an ensemble as a driving force in the control of problem-solving activity. The individual NNs were often trained independently or sequentially. One of the disadvantages of such an approach is the loss of interaction among the individual networks during

learning. EENCL emphasizes specialization and cooperation among individual NNs in the ensemble. In EENCL, an evolutionary algorithm based on evolutionary programming [18] has been used to search for a population of diverse individual NNs that together solve a problem (e.g., classify examples). To maintain a diverse population while applying the selection operator, the evolutionary algorithm must incorporate some kind of speciation technique [19, 20, 21] by which individuals in a population can form several species automatically through evolution. Fitness sharing [22, 17] and negative correlation learning [13, 14, 15] have been used to encourage the formation of different species. Fitness sharing refers to a class of speciation techniques in evolutionary computation [23]. The idea of negative correlation learning is to encourage different individual networks in the ensemble to learn different parts or aspects of the training data, so that the ensemble can better learn the entire training data. In negative correlation learning, the individual networks are trained simultaneously rather than independently or sequentially. This provides an opportunity for the individual networks to interact with each other and to specialize. Negative correlation learning can create negatively correlated NNs using a correlation penalty term in the error function to encourage such specialization.

Second, most previous work separated the design of individual NNs from combination procedures, following a two-stage design process: first generating the individual NNs, and then combining them. The possible interactions among them cannot be exploited until the combination stage. There is no feedback from the combination stage to the individual NN design stage. It is possible that some of the independently designed individual NNs do not make much contribution to the whole system. In contrast, negative correlation learning learns and combines individual NNs in the same process. This provides an opportunity for different NNs to interact with each other and to specialize. One approach that does resemble negative correlation learning is the mixtures-of-experts (ME) architectures [6, 7, 8, 24, 9]. Negative correlation learning is different from the ME architecture [9] that consists of a gating network and a number of expert networks, although the ME architecture can also produce biased individual networks whose estimates are negatively correlated [9]. Negative correlation learning does not need a separate gating network. It uses totally different error functions. The strength parameter λ in negative correlation learning provides a convenient way to balance the bias-variance-covariance trade-off [15]. The ME architecture does not provide such control over the trade-off.

Third, the number of NNs in the ensemble is often predefined and fixed in most previous work. Usually the number of NNs is determined manually through a trial-and-error process. In EENCL, the number of NNs in the ensemble is not predefined but determined by the number of species, which are formed automatically through evolution.

The rest of this paper is organized as follows: Section 2 gives ideas of using evolutionary learning to design NN ensembles; Section 3 describes EENCL in detail; Section 4 presents experimental results on EENCL and some discussions; and finally Section 5 concludes with a summary of the paper and a few remarks.

2 Evolutionary Learning

Evolutionary learning is a population-based learning method. It has been used to minimize error functions because of its robust global search capability, as compared with gradient-based algorithms. It is also used because it could incorporate NN's weight learning and structure learning in the same learning process. EPNet is such an evolutionary learning system for designing feedforward NNs [16].

In essence, evolutionary learning has been used as a kind of global optimization algorithm. It is common that the best individual (i.e., the individual with the smallest error) is selected as the output of evolution. No consideration is given to other individuals in the population. Unfortunately, an NN with the smallest training error may not be the NN with best generalization [1].

While there is little we can do in nonpopulation-based learning, there are opportunities for improving population-based evolutionary learning. Since the best individual in learning might not be the individual with best generalization, other individuals in the population might be better and contain useful information. Hence, an ensemble system that combines the entire population is expected to have better generalization than any single individual.

Combining individual NNs in a population into an NN ensemble has a close relationship with the design of NN ensembles. The population of NNs can be regarded as an ensemble. The evolutionary process can be regarded as a natural and automatic way to design NN ensembles. We have tested this idea to form the final result by combining all the individuals in the last generation [1]. However, the interaction between individual NN design and combination was not considered, and the ensemble size was fixed.

3 Evolving Neural Network Ensembles

EENCL is studied to address the following issues: exploitation of the interaction between individual NN design and combination and automatic determination of the number of individual NNs. In EENCL, an evolutionary algorithm based on evolutionary programming [18] has been used to search for a population of diverse individual NNs that solve a problem together. To maintain a diverse population, fitness sharing [22] and negative correlation learning have been used to encourage the

formation of different species. In the current implementation of EENCL, each NN in the ensemble is a feedforward NN with logistic transfer functions. The major steps of EENCL are given as follows:

1. Generate an initial population of M NNs, and set $k = 1$. The number of hidden nodes for each NN, n_h, is specified by the user. The random initial weights are distributed uniformly inside a small range.

2. Train each NN in the initial population on the training set for a certain number of epochs using negative correlation learning. The number of epochs, n_e, is specified by the user.

3. Randomly choose a group of n_b NNs as parents to create n_b offspring NNs by Gaussian mutation.

4. Add the n_b offspring NNs to the population and train the offspring NNs using negative correlation learning while the remaining NNs' weights are frozen.

5. Calculate the fitness of $M + n_b$ NNs in the population and prune the population to the M fittest NNs.

6. Go to the next step if the maximum number of generations has been reached. Otherwise, $k = k + 1$ and go to Step 3.

7. Form species using the k-means algorithm.

8. Combining species to form the ensembles.

There are two levels of adaptation in EENCL: negative correlation learning at the individual level and evolutionary learning based on evolutionary programming (EP) [18] at the population level. Details about each component of EENCL are given in the following sections.

3.1 Negative Correlation Learning

Suppose that we have a training set

$$D = \{(\mathbf{x}(1), d(1)), \cdots, (\mathbf{x}(N), d(N))\}$$

where $\mathbf{x} \in R^p$, d is a scalar, and N is the size of the training set. The assumption that the output d is a scalar has been made merely to simplify exposition of ideas without loss of generality. This section considers estimating d by forming an ensemble whose output is a simple averaging of outputs of a set of neural networks

$$F(n) = \frac{1}{M}\Sigma_{i=1}^{M} F_i(n) \tag{1}$$

where M is the number of the individual neural networks in the ensemble, $F_i(n)$ is the output of network i on the nth training pattern, and $F(n)$ is the output of the ensemble on the nth training pattern.

Negative correlation learning introduces a correlation penalty term into the error function of each individual network in the ensemble so that all the networks can be trained simultaneously and interactively on the same training data set D. The error function E_i for network i in negative correlation learning is defined by

$$\begin{aligned} E_i &= \frac{1}{N}\Sigma_{n=1}^{N} E_i(n) \\ &= \frac{1}{N}\Sigma_{n=1}^{N}\frac{1}{2}(F_i(n)-d(n))^2 + \frac{1}{N}\Sigma_{n=1}^{N}\lambda p_i(n) \end{aligned} \tag{2}$$

where $E_i(n)$ is the value of the error function of network i at the presentation of the nth training pattern. The first term in the right side of Eq.(2) is the empirical risk function of network i. The second term, p_i, is a correlation penalty function. The purpose of minimizing p_i is to negatively correlate each network's error with errors for the rest of the ensemble. The parameter $0 \leq \lambda \leq 1$ is used to adjust the strength of the penalty. The penalty function p_i has the form:

$$p_i(n) = (F_i(n) - F(n))\,\Sigma_{j\neq i}\,(F_j(n) - F(n)) \tag{3}$$

The partial derivative of $E_i(n)$ with respect to the output of network i on the nth training pattern is

$$\begin{aligned} \frac{\partial E_i(n)}{\partial F_i(n)} &= F_i(n) - d(n) + \lambda\frac{\partial p_i(n)}{\partial F_i(n)} \\ &= F_i(n) - d(n) + \lambda\Sigma_{j\neq i}\,(F_j(n) - F(n)) \\ &= F_i(n) - d(n) - \lambda(F_i(n) - F(n)) \\ &= (1-\lambda)(F_i(n) - d(n)) + \lambda(F(n) - d(n)) \end{aligned} \tag{4}$$

where we have made use of the assumption that $F(n)$ has constant value with respect to $F_i(n)$. The standard back-propagation (BP) algorithm [25] has been used for weight adjustments in the mode of pattern-by-pattern updating. That is, weight updating of all the individual networks is performed simultaneously using Eq.(4) after the presentation of each training pattern. One complete presentation of the entire training set during the learning process is called an *epoch*. The negative correlation learning from Eq.(4) is a simple extension to the standard BP algorithm. In fact, the only modification that is needed is to calculate an extra term of the form $\lambda(F_i(n) - F(n))$ for the ith network.

From Eqs.(2), (3), and (4), we may make the following observations:

1. During the training process, all the individual networks interact with each other through their penalty terms in the error functions. Each network F_i minimizes not only the difference between $F_i(n)$ and $d(n)$, but also the difference between $F(n)$ and $d(n)$. That is, negative correlation learning considers errors that all other networks have learned while training a network.

2. For $\lambda = 0.0$, there are no correlation penalty terms in the error functions of the individual networks, and the individual networks are just trained independently. That is, independent training for the individual networks is a special case of negative correlation learning.

3. For $\lambda = 1$, from Eq.(4) we get
$$\frac{\partial E_i(n)}{\partial F_i(n)} = F(n) - d(n) \tag{5}$$
Note that the empirical risk function of the ensemble for the nth training pattern is defined by
$$E_{ens}(n) = \frac{1}{2}(\frac{1}{M}\Sigma_{i=1}^{M}F_i(n) - d(n))^2 \tag{6}$$
The partial derivative of $E_{ens}(n)$ with respect to F_i on the nth training pattern is
$$\begin{aligned}\frac{\partial E_{ens}(n)}{\partial F_i(n)} &= \frac{1}{M}(\frac{1}{M}\Sigma_{i=1}^{M}F_i(n) - d(n)) \\ &= \frac{1}{M}(F(n) - d(n))\end{aligned} \tag{7}$$
In this case, we get
$$\frac{\partial E_i(n)}{\partial F_i(n)} \propto \frac{\partial E_{ens}(n)}{\partial F_i(n)} \tag{8}$$

The minimization of the empirical risk function of the ensemble is achieved by minimizing the error functions of the individual networks. From this point of view, negative correlation learning provides a novel way to decompose the learning task of the ensemble into a number of subtasks for different individual networks.

3.2 Fitness Evaluation

The fitness evaluation in EENCL is carried out by fitness sharing. Explicit and implicit fitness sharing have been proposed to encourage speciation in recent years [26, 27]. Fitness sharing accomplishes speciation by degrading the raw fitness (i.e., the unshared fitness) of an individual according to the presence of similar individuals. Thus this type of speciation requires a distance metric on the phenotype or genotype space of the individuals. Traditionally, such a measurement is based on the Hamming distance between two binary strings. However, this sharing scheme is not suitable for EENCL because the individual networks are not represented by binary strings in EENCL.

Fitness sharing used in EENCL is based on the idea of "covering" the same training patterns by shared individuals. The procedure of calculating shared fitness is carried out pattern-by-pattern over the training set. If one training pattern is learned correctly by p individuals in the population, each of these p individuals receives fitness $1/p$, and the rest of the individuals in the population receive zero fitness. Otherwise, all the individuals in the population receive zero fitness. The fitness is summed over all training patterns. Such fitness evaluation encourages each individual in the population to cover different patterns in the training set.

3.3 Selection Mechanism and Replacement Strategy

There are many different selection mechanisms, such as *roulette wheel selection* [23] and *rank-based selection* [16], that have been used to select individuals to reproduce in evolutionary algorithms. Such selection mechanisms encourage the algorithm to quickly find the best individual. In EENCL, each individual is selected to be mutated with equal probability. This reflects the emphasis of EENCL on evolving a diverse set of individuals.

After mutation, all the $M + n_b$ individuals in the population are evaluated and sorted. The M fittest individuals are selected to replace the current population.

3.4 Mutation

Mutation in EENCL is carried out in two steps: weight mutation and further weight training. In the first step, the selected n_b parent networks create n_b offspring by the following weight mutation:

$$w_{ij}^{'} = w_{ij} + N(0,1) \tag{9}$$

where $w_{ij}^{'}$ and w_{ij} denote the weights of offspring i and parent i, respectively, $i = 1, \cdots, n_b$, j is the index number of weights. $N(0,1)$ denotes a Gaussian random variable with mean zero and standard deviation one.

In the second step, the n_b offspring NNs are first added to the population, and indexed by $M + 1$ to $M + n_b$. Then the n_b offspring NNs are further trained by negative correlation learning, while the weights of the remaining NNs in the population are frozen. The error function for each offspring NN is defined by

$$\begin{aligned} E_i &= \frac{1}{N}\Sigma_{n=1}^{N}[\frac{1}{2}(F_i(n) - y(n))^2 \\ &\quad +\lambda(F_i(n) - F(n))\Sigma_{j=1,j\neq i}^{M+n_b}(F_j(n) - F(n))] \end{aligned} \tag{10}$$

where $i = M+1, \cdots, M+n_b$, N is the number of training patterns, $F_j(n)$ is the output of individual network j on the nth training pattern, $y(n)$ is the desired output of the nth training pattern and

$F(n) = \frac{1}{M+n_b}\Sigma_{m=1}^{M+n_b} F_m(n)$. Error function (10) uses the feedback from the current population to guide the learning of each offspring.

3.5 Forming the Ensembles

A population of NNs is generated after the evolutionary process. A direct method is to use all the individual NNs in the population to form an ensemble. It is interesting to investigate whether or not all the individual NNs in the last generation are useful in forming the ensemble. Can the size of the ensemble be reduced without worsening the performance too much? Two methods for constructing the NN ensembles were tested in EENCL. One used all the individual NNs in the last generation and the other selected one representative from each species in the last generation. The species in the population are determined by clustering the individuals in the population using the *k-means algorithm* [28]. The resulting clusters correspond to different species. The representative for each species is the fittest individual in the species.

There are n_c output nodes in each NN, where n_c is the number of classes in a classification problem. For each NN, its output on N training patterns forms an $(n_c \times N)$-dimensional vector. Given M output vectors $\mathbf{o}_m, m = 1, \cdots, M$, of individual NNs, the k-means algorithm classifies them in k, determined by the user in advance, different clusters. The algorithm proceeds as follows:

1. Choose a set of cluster centers $\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_k\}$ arbitrarily.

2. For all $\mathbf{o}_m, m = 1, \cdots, M$, assign $\mathbf{o}_m$ to cluster C_i using the minimum Euclidean distance rule:

$$\mathbf{o}_m \text{ belongs to cluster } C_i \text{ if } \|\mathbf{o}_m - \mathbf{c}_i\| < \|\mathbf{o}_m - \mathbf{c}_j\|, \ j \neq i$$

3. Compute new cluster centers so as to minimize the cost function

$$J_i = \sum_{\mathbf{o}_n \in C_i} \|\mathbf{o}_n - \mathbf{c}_i\|$$

4. If any cluster center changes, return to step 2; otherwise, stop.

3.6 Combination Methods

Three combination methods for determining the output of the ensemble have been investigated in EENCL. The first is simple averaging. The output of the ensemble is formed by a simple averaging of output of individual NNs in the ensemble. The second is majority voting. The output of the greatest number of individual NNs will be the output of the ensemble. If there is a tie, the output of the ensemble is rejected. The third is winner-takes-all. For each pattern of the testing set,

the output of the ensemble is only decided by the individual NN whose output has the highest activation.

4 Experimental Studies

We have applied EENCL to two benchmark problems: the Australian credit card assessment problem and the diabetes problem. Both data sets were obtained from the UCI machine learning benchmark repository. They are available by anonymous ftp at ics.uci.edu (128.195.1.1) in directory /pub/machine-learning-databases.

The Australian credit card assessment problem is to assess applications for credit cards based on a number of attributes. There are 690 patterns in total. The output has two classes. The 14 attributes include 6 numeric values and 8 discrete ones, the latter having from 2 to 14 possible values.

The diabetes data set is a two-class problem that has 500 examples of class 1 and 268 of class 2. There are 8 attributes for each example. The data set is rather difficult to classify. The so-called "class" value is really a binarised form of another attribute that is itself highly indicative of certain types of diabetes but does not have a one-to-one correspondence with the medical condition of being diabetic.

In order to compare EENCL with previous work, the experimental setup is the same as the previous experimental setup described in [29]. The n-fold cross-validation technique [30] was used to divide the data randomly into n mutually exclusive data groups of equal size. In each train-and-test process, one data group is selected as the testing set, and the other $(n-1)$ groups become the training set. The estimated error rate is the average error rate from these n groups. In this way, the error rate is estimated efficiently and in an unbiased way. The parameter n was set to be 10 for the Australian credit card data set, and 12 for the diabetes data set, respectively.

The parameters used in EENCL were set to be the same for both problems: the population size M (25), the number of generations (200), the reproduction block size n_b (2), the strength parameter λ (0.75), the number of training epochs n_e (5), the minimum number of cluster sets (3), and the maximum number of cluster sets (25). The used NNs in the population are multilayer perceptrons with one hidden layer and five hidden nodes. These parameters were selected after some preliminary experiments. They were not meant to be optimal.

4.1 Experimental Results

4.1.1 A Population as an Ensemble

Tables 1–2 show the results of EENCL for the two data sets, where the ensembles were constructed by the whole population in the last generation. The *accuracy rate* refers to the percentage of correct classifications produced by EENCL. In comparison with the accuracy rates obtained by three combination methods, winner-takes-all outperformed simple averaging and majority voting on both problems. In simple averaging and majority voting, all individuals are treated equally. However, not all individuals are equally important. Because different individuals created by EENCL were able to specialize to different parts of the testing set, only the outputs of these specialists should be considered to make the final decision of the ensemble for this part of the testing set. The winner-takes-all combination method performed better because there are good and poor individuals for each pattern in the testing set and winner-takes-all selects the best individual.

	Simple Averaging		Majority Voting		Winner-Takes-All	
Accuracy Rate	Training	Testing	Training	Testing	Training	Testing
Mean	0.910	0.855	0.917	0.857	0.887	0.865
SD	0.010	0.039	0.010	0.039	0.007	0.028
Min	0.897	0.797	0.900	0.812	0.874	0.812
Max	0.924	0.913	0.928	0.913	0.895	0.913

Table 1: Accuracy rates of EENCL for the Australian credit card data set. The results are averaged on 10-fold cross-validation. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

4.1.2 A Subset of the Population as an Ensemble

For the previous implementation of EENCL, all the individuals in the last generation were used in the ensembles. It is interesting to investigate whether we can reduce the size of the ensembles by selecting representative individuals from the whole population. Such investigation can provide some hints on whether all the individuals in the last generation will contain some useful information.

We used the k-means algorithm [28] to divide the individuals in the population into different clusters. Each cluster corresponds to a species. These species are then used to construct an NN ensemble, in which one representative from each species was selected for combination. The representative for each species is the fittest individual in the species.

	Simple Averaging		Majority Voting		Winner-Takes-All	
Accuracy Rate	Training	Testing	Training	Testing	Training	Testing
Mean	0.795	0.766	0.802	0.764	0.783	0.779
SD	0.007	0.039	0.007	0.042	0.007	0.045
Min	0.783	0.703	0.786	0.688	0.774	0.703
Max	0.805	0.828	0.810	0.828	0.794	0.844

Table 2: Accuracy rates of EENCL for the diabetes data set. The results are averaged on 12-fold cross-validation. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

The number of species, i.e., the number of clusters, starts from 3 to 25 (i.e., the population size). The optimal number of species was determined by measuring the performance of these constructed ensembles on the training set. The accuracy rate was chosen as the measure of performance. The results of the ensemble formed by the representatives from species are given in Table 3. The combination method used is winner-takes-all. The t-test statistics comparing the accuracies of the ensembles using the representatives from species to the ensembles using the whole population are 0.80 for the Australian credit card data set, and -0.36 for the diabetes data set. No statistically significance difference was observed between them for both data sets, which implies that the ensemble does not have to use the whole population to achieve good performance. The size of the ensemble can be substantially smaller than the population size. The reduction in the size of the ensembles can be seen from Table 4 that gives the sizes of the ensembles using the representatives from species.

4.1.3 Comparisons with Other Work

Direct comparison with other evolutionary approaches to designing ensembles is very difficult due to the lack of such results. Instead, the best and latest results available in the literature, regardless of whether the algorithm used was an evolutionary, a BP, or a statistical one, were used in the comparison.

Tables 5–6 compare the results of EENCL, including the results of the ensembles using the whole population and the ensembles using the representatives from species, with those produced by other 23 algorithms tested by Michie *et al.* [29]. These 23 algorithms can be categorized into four groups: statistical algorithms (Discrim, Quadisc, Logdisc, SMART, ALLOC80, k-NN, CASTLE, NaiveBay, Default); decision trees (CART, IndCART, NewID, AC^2, Baytree, Cal5, C4.5); rule-

	Card		Diabetes	
Accuracy Rate	Training	Testing	Training	Testing
Mean	0.887	0.868	0.783	0.777
SD	0.004	0.030	0.009	0.042
Min	0.881	0.812	0.770	0.719
Max	0.890	0.913	0.798	0.844

Table 3: Accuracy rates of the ensemble formed by the representatives from species. The results are averaged on 10-fold cross-validation for the Australian credit card data set, and 12-fold cross-validation for the diabetes data set. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

	Size of the Ensembles			
	Mean	SD	Min	Max
Card	13.2	7.8	5	25
Diabetes	16.3	6.4	5	25

Table 4: Sizes of the ensembles using the representatives from species. The results are averaged on 10-fold cross-validation for the Australian credit card data set, and 12-fold cross-validation for the diabetes data set. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

based methods (CN2, ITrule); neural networks (Backprop, Kohonen, LVQ, RBF, DIPOL92). More details about these algorithms appear in [29]. EENCL used the same data setup as in [29], i.e., 10-fold cross-validation for the Australian credit card data set and 12-fold cross-validation for the diabetes data set. The *error rate* refers to the percentage of wrong classifications on the testing set. For the ensembles using the whole population and the winner-takes-all combination method, the average testing accuracy rates of EENCL are respectively 0.865 and 0.779 for the Australian credit card data set and the diabetes data set. Accordingly, their average testing error rates are respectively 0.135 and 0.221. For the ensembles using the representatives from species, similarly we can get the average testing error rates of EENCL from Table 3 for the two data sets. They are respectively 0.132 and 0.223 for the Australian credit card data set and the diabetes data set. As demonstrated by the results, EENCL have been able to achieve the generalization performance comparable to or better than the best of 23 algorithms tested [29] for both the Australian credit card data set and the diabetes data set.

Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate
EENCL	0.135, 0.132	CART	0.145	ITrule	0.137
Discrim	0.141	IndCART	0.152	Cal5	0.131
Quadisc	0.207	NewID	0.181	Kohonen	FD
Logdisc	0.141	AC^2	0.181	DIPOL92	0.141
SMART	0.158	Baytree	0.171	Backprop	0.154
ALLOC80	0.201	NaiveBay	0.151	RBF	0.145
k-NN	0.181	CN2	0.204	LVQ	0.197
CASTLE	0.148	C4.5	0.155	Default	0.440

Table 5: Comparison among EENCL and others [29] in terms of the average testing error rate for the Australian credit card data set. The results are averaged on 10-fold cross-validation. "FD" indicates Kohonen algorithm failed on that data set. Two error rates are listed for EENCL, which are the results for the ensembles using the whole population and the ensembles using the representatives from species.

5 Conclusions

This paper introduces EENCL for learning and designing of NN ensembles. The negative correlation learning and fitness sharing were adopted to encourage the formation of species in the population. In a sense, EENCL provides an automatic way of designing NN ensembles, where each NN is an

Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate
EENCL	0.221, 0.223	CART	0.255	ITrule	0.245
Discrim	0.225	IndCART	0.271	Cal5	0.250
Quadisc	0.262	NewID	0.289	Kohonen	0.273
Logdisc	0.223	AC^2	0.276	DIPOL92	0.224
SMART	0.232	Baytree	0.271	Backprop	0.248
ALLOC80	0.301	NaiveBay	0.262	RBF	0.243
k-NN	0.324	CN2	0.289	LVQ	0.272
CASTLE	0.258	C4.5	0.270		

Table 6: Comparison among EENCL and others [29] in terms of the average testing error rate for the diabetes data set. The results are averaged on 12-fold cross-validation. Two error rates are listed for EENCL, which are the results for the ensembles using the whole population and the ensembles using the representatives from species.

individual or a representative from each species in the population. EENCL was tested on the Australian credit card assessment problem and the diabetes problem. Very competitive results have been produced by EENCL in comparison with other algorithms [29].

Rosen [31] proposed an ensemble learning algorithm using decorrelated NNs. The idea is that individual networks attempt to not only minimize the error between the target and their output, but also decorrelate their errors from networks trained previously. However, Rosen's algorithm still trains the individual networks sequentially. One major disadvantage of this algorithm is that training a network in an ensemble cannot affect the networks trained previously in the ensemble, so that the errors of the individual networks are not necessarily negatively correlated. Negative correlation learning extends Rosen's work to simultaneous training of negatively correlated NNs. Such extension has produced significant improvement in NN ensembles' performance. Negatively correlated NNs can be obtained in negative correlation learning.

The architecture of each NN in the ensemble is predefined in the current implementation of EENCL. One of the future improvements to EENCL would be to evolve the architectures of NNs by EPNet [16]. Another future improvement would be to form the ensemble output from a linear combination of individual NN outputs, where the linear combination weights would co-evolve with NN ensembles in order to exploit the interaction between NNs and their combined output.

Acknowledgement — The authors are grateful to anonymous referees and Dr David Fogel for their constructive comments and criticism that have helped to improve the paper.

References

[1] X. Yao and Y. Liu.

Making use of population information in evolutionary artificial neural networks.

IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, 28(3):417–425, 1998.

[2] A. J. C. Sharkey.

On combining artificial neural nets.

Connection Science, 8:299–313, 1996.

[3] O. G. Selfridge.

Pandemonium: a paradigm for learning.

In *Mechanisation of Thought Processes: Proc. of a Symp. Held at the National Physical Lab.*, pages 513–526. HMSO, London, 1958.

[4] N. J. Nilsson.

Learning Machines: Foundations of Trainable Pattern-Classifying Systems.

NY: McGraw Hill, New York, 1965.

[5] L. K. Hansen and P. Salamon.

Neural network ensembles.

IEEE Trans. on Pattern Analysis and Machine Intelligence, 12(10):993–1001, 1990.

[6] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton.

Adaptive mixtures of local experts.

Neural Computation, 3:79–87, 1991.

[7] R. A. Jacobs and M. I. Jordan.

A competitive modular connectionist architecture.

In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 767–773. Morgan Kaufmann, San Mateo, CA, 1991.

[8] R. A. Jacobs, M. I. Jordan, and A. G. Barto.

Task decomposition through competition in a modular connectionist architecture: the what and where vision task.

Cognitive Science, 15:219–250, 1991.

[9] R. A. Jacobs.

Bias/variance analyses of mixture-of-experts architectures.

Neural Computation, 9:369–383, 1997.

[10] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik.

Boosting and other ensemble methods.

Neural Computation, 6:1289–1301, 1994.

[11] R. E. Schapire.

The strength of weak learnability.

Machine Learning, 5:197–227, 1990.

[12] H. Drucker, R. Schapire, and P. Simard.

Improving performance in neural networks using a boosting algorithm.

In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 42–49. Morgan Kaufmann, San Mateo, CA, 1993.

[13] Y. Liu and X. Yao.

Negatively correlated neural networks can produce best ensembles.

Australian Journal of Intelligent Information Processing Systems, 4:176–185, 1998.

[14] Y. Liu and X. Yao.

A cooperative ensemble learning system.

In *Proc. of the 1998 IEEE International Joint Conference on Neural Networks (IJCNN'98)*, pages 2202–2207. IEEE Press, Piscataway, NJ, USA, 1998.

[15] Y. Liu and X. Yao.

Simultaneous training of negatively correlated neural networks in an ensemble.

IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, 29(6):716–725, 1999.

[16] X. Yao and Y. Liu.

A new evolutionary system for evolving artificial neural networks.

IEEE Trans. on Neural Networks, 8(3):694–713, 1997.

[17] Y. Liu and X. Yao.

Towards designing neural network ensembles by evolution.

In *Parallel Problem Solving from Nature — PPSN V: Proc. of the Fifth International Conference on Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, pages 623–632. Springer-Verlag, Berlin, 1998.

[18] D. B. Fogel.

Evolutionary Computation: Towards a New Philosophy of Machine Intelligence.

IEEE Press, New York, NY, 1995.

[19] J. Horn, D. E. Goldberg, and K. Deb.

Implicit niching in a learning classifier system: nature's way.

Evolutionary Computation, 2:37–66, 1994.

[20] P. Darwen and X. Yao.

Automatic modularization by speciation.

In *Proc. of the 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC'96), Nagoya, Japan*, pages 88–93. IEEE Press, New York, NY 10017-2394, 1996.

[21] P. Darwen and X. Yao.

Every niching method has its niche: fitness sharing and implicit sharing compared.

In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature (PPSN) IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 398–407. Springer-Verlag, Berlin, 1996.

[22] X. Yao, Y. Liu, and P. Darwen.

How to make best use of evolutionary learning.

In R. Stocker, H. Jelinek, and B. Durnota, editors, *Complex Systems: From Local Interactions to Global Phenomena*, pages 229–242. IOS Press, Amsterdam, 1996.

[23] D. E. Goldberg.

Genetic Algorithms in Search, Optimization, and Machine Learning.

Addison-Wesley, Reading, MA, 1989.

[24] M. I. Jordan and R. A. Jacobs.

Hierarchical mixtures-of-experts and the em algorithm.

Neural Computation, 6:181–214, 1994.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams.

Learning internal representations by error propagation.

In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[26] S. W. Mahfoud.

Niching Methods for Genetic Algorithms.

PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[27] P. J. Darwen and X. Yao.

Speciation as automatic categorical modularization.

IEEE Trans. on Evolutionary Computation, 1(2):101–108, 1997.

[28] J. MacQueen.

Some methods for classification and analysis of multivariate observation.

In *Proc. of the 5th Berkely Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Berkely: University of California Press, 1967.

[29] D. Michie, D. J. Spiegelhalter, and C. C. Taylor.

Machine Learning, Neural and Statistical Classification.

Ellis Horwood Limited, London, 1994.

[30] M. Stone.

Cross-validatory choice and assessment of statistical predictions.

Journal of the Royal Statistical Society, 36:111–147, 1974.

[31] B. E. Rosen.

Ensemble learning using decorrelated neural networks.

Connection Science, 8:373–383, 1996.

Table Captions

Table 1: Accuracy rates of EENCL for the Australian credit card data set. The results are averaged on 10-fold cross-validation. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

Tables 2: Accuracy rates of EENCL for the diabetes data set. The results are averaged on 12-fold cross-validation. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

Table 3: Accuracy rates of the ensemble formed by the representatives from species. The results are averaged on 10-fold cross-validation for the Australian credit card data set, and 12-fold cross-validation for the diabetes data set. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

Table 4: Sizes of the ensembles using the representatives from species. The results are averaged on 10-fold cross-validation for the Australian credit card data set, and 12-fold cross-validation for the diabetes data set. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

Table 5: Comparison among EENCL and others [29] in terms of the average testing error rate for the Australian credit card data set. The results are averaged on 10-fold cross-validation."FD" indicates Kohonen algorithm failed on that data set. Two error rates are listed for EENCL, which are the results for the ensembles using the whole population and the ensembles using the representatives from species.

Table 6: Comparison among EENCL and others [29] in terms of the average testing error rate for the diabetes data set. The results are averaged on 12-fold cross-validation. Two error rates are listed for EENCL, which are the results for the ensembles using the whole population and the ensembles using the representatives from species.

Tables on Individual Pages

	Simple Averaging		Majority Voting		Winner-Takes-All	
Accuracy Rate	Training	Testing	Training	Testing	Training	Testing
Mean	0.910	0.855	0.917	0.857	0.887	0.865
SD	0.010	0.039	0.010	0.039	0.007	0.028
Min	0.897	0.797	0.900	0.812	0.874	0.812
Max	0.924	0.913	0.928	0.913	0.895	0.913

Table 1: Accuracy rates of EENCL for the Australian credit card data set. The results are averaged on 10-fold cross-validation. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

	Simple Averaging		Majority Voting		Winner-Takes-All	
Accuracy Rate	Training	Testing	Training	Testing	Training	Testing
Mean	0.795	0.766	0.802	0.764	0.783	0.779
SD	0.007	0.039	0.007	0.042	0.007	0.045
Min	0.783	0.703	0.786	0.688	0.774	0.703
Max	0.805	0.828	0.810	0.828	0.794	0.844

Table 2: Accuracy rates of EENCL for the diabetes data set. The results are averaged on 12-fold cross-validation. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

	Card		Diabetes	
Accuracy Rate	Training	Testing	Training	Testing
Mean	0.887	0.868	0.783	0.777
SD	0.004	0.030	0.009	0.042
Min	0.881	0.812	0.770	0.719
Max	0.890	0.913	0.798	0.844

Table 3: Accuracy rates of the ensemble formed by the representatives from species. The results are averaged on 10-fold cross-validation for the Australian credit card data set, and 12-fold cross-validation for the diabetes data set. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

	Size of the Ensembles			
	Mean	SD	Min	Max
Card	13.2	7.8	5	25
Diabetes	16.3	6.4	5	25

Table 4: Sizes of the ensembles using the representatives from species. The results are averaged on 10-fold cross-validation for the Australian credit card data set, and 12-fold cross-validation for the diabetes data set. *Mean, SD, Min*, and *Max* indicate the mean value, standard deviation, minimum, and maximum value, respectively.

Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate
EENCL	0.135, 0.132	CART	0.145	ITrule	0.137
Discrim	0.141	IndCART	0.152	Cal5	0.131
Quadisc	0.207	NewID	0.181	Kohonen	FD
Logdisc	0.141	AC^2	0.181	DIPOL92	0.141
SMART	0.158	Baytree	0.171	Backprop	0.154
ALLOC80	0.201	NaiveBay	0.151	RBF	0.145
k-NN	0.181	CN2	0.204	LVQ	0.197
CASTLE	0.148	C4.5	0.155	Default	0.440

Table 5: Comparison among EENCL and others [29] in terms of the average testing error rate for the Australian credit card data set. The results are averaged on 10-fold cross-validation. "FD" indicates Kohonen algorithm failed on that data set. Two error rates are listed for EENCL, which are the results for the ensembles using the whole population and the ensembles using the representatives from species.

Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate
EENCL	0.221, 0.223	CART	0.255	ITrule	0.245
Discrim	0.225	IndCART	0.271	Cal5	0.250
Quadisc	0.262	NewID	0.289	Kohonen	0.273
Logdisc	0.223	AC^2	0.276	DIPOL92	0.224
SMART	0.232	Baytree	0.271	Backprop	0.248
ALLOC80	0.301	NaiveBay	0.262	RBF	0.243
k-NN	0.324	CN2	0.289	LVQ	0.272
CASTLE	0.258	C4.5	0.270		

Table 6: Comparison among EENCL and others [29] in terms of the average testing error rate for the diabetes data set. The results are averaged on 12-fold cross-validation. Two error rates are listed for EENCL, which are the results for the ensembles using the whole population and the ensembles using the representatives from species.