Evolving Neural Network Ensembles by Minimization of Mutual Information

Xin Yao[1] **and Yong Liu**[2]

[1]School of Computer Science
The University of Birmingham
Edgbaston, Birmingham, U.K.
X.Yao@cs.bham.ac.uk
[2]The University of Aizu
Aizu-Wakamatsu, Fukushima 965-8580, Japan
yliu@u-aizu.ac.jp

Abstract. Learning and evolution are two fundamental forms of adaptation. There has been a great interest in combining learning and evolution with neural networks in recent years. This paper presents a hybrid learning system for learning and designing of neural network ensembles based on negative correlation learning and evolutionary learning. The idea of the hybrid learning system is to regard the population of neural networks as an ensemble, and the evolutionary process as the design of neural network ensembles. Two fitness sharing techniques have been used in the evolutionary process. One is based on the covering set. The other is to use the concept of mutual information. The effectiveness of such hybrid learning approach was tested on two real-world problems.

Keywords: hybrid learning system, evolutionary learning, negative correlation learning, fitness sharing

1 Introduction

Learning and evolution are two fundamental forms of adaptation. There has been a great interest in combining learning and evolution with neural networks (NNs) in recent years. Evolutionary NNs refer to a special class of NNs in which evolution is another fundamental form of adaptation in addition to learning (Yao 1991, Yao 1993, Yao 1994, Yao 1995). In evolutionary NNs, evolutionary algorithms are used to perform various tasks, such as connection weight training, architecture design, learning rule adaptation, input feature selection, connection weight initialization, rule extraction from NNs, etc. One distinct feature of evolutionary NNs is their adaptability to a dynamic environment. In other words, evolutionary NNs can adapt to an environment as well as changes in the environment. The two forms of adaptation, i.e., evolution and learning in evolutionary NNs, make their adaptation to a dynamic environment much more effective and efficient. In a broader sense, evolutionary NNs can be regarded as a general framework for adaptive systems, i.e., systems that can change their architectures and learning rules appropriately without human intervention.

In evolutionary NNs, it is practical that the NN with the minimum error on a training data set or a validation data set is chosen to be the final learning system. However, learning is different from optimization in practice because we want the learned system to have best generalization, which is different from minimizing an error function on the training data set or the validation data set. The NN with the minimum error on a training data set may not have best generalization unless there is an equivalence between generalization and the error on the training data. Unfortunately, measuring generalization quantitatively and accurately is almost impossible in practice (Wolpert 1990) although there are many theories and criteria on generalization, such as the minimum description length (Rissanen 1978), Akaike information criteria (Akaike 1974), and minimum message length (Wallace and Patrick 1991). In practice, these criteria are often used to define better error functions in the hope that minimizing the functions will maximize generalization. While these functions often lead to better generalization of learned systems, there is no guarantee.

Since the maximum fitness or the minimum error may not be equivalent to best generalization in evolutionary learning, the best individual with the maximum fitness in a population may not be the desired one. Other individuals in the population may contain some useful information that will help to improve generalization of learned systems. It is thus beneficial to make use of the whole population rather than any single individual. A population always contains at least as much information as any single individual. Hence, combining different individuals in the population to form an integrated system is expected to produce better results. Such a population of NNs is called an NN ensemble in this paper. There have been some very successful experiments which

show that evolutionary learning can be used to evolve NN ensembles (Liu et al. 2000, Liu et al. 2001, Liu and Yao 2002, Liu and Yao 2002).

In this paper, a hybrid learning system based on evolutionary learning and negative correlation learning (Liu and Yao 1999a, Liu and Yao 1999b, McKay and Abbass 2001) is presented to learn and evolve NN ensembles. 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 which learn to subdivide the task and thereby solve it more efficiently and elegantly (Liu and Yao 1999a, Liu and Yao 1999b). However, designing NN ensembles is a very difficult task. It relies heavily on human experts and prior knowledge about the problem. The idea of the hybrid learning system is to regard the population of NNs as an ensemble, and the evolutionary process as the design of NN ensembles.

The negative correlation learning (Liu and Yao 1999a, Liu and Yao 1999b, McKay and Abbass 2001) and fitness sharing (Liu and Yao 1998) were adopted in the hybrid learning system to encourage the formation of species in the population. 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 (Liu and Yao 1999a, Liu and Yao 1999b), 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.

Fitness sharing refers to a class of speciation techniques in evolutionary algorithms (Goldberg 1989). Two fitness sharing techniques were used in the hybrid learning system. One is based on the idea of the covering set that consists of the same training patterns correctly classified by the shared individuals. The other is based on the mutual information between one individual and the rest of population. Although the covering set could roughly show the similarity between two NNs, a more accurate similarity measurement between two NNs in a population can be defined by the explicit mutual information of output variables extracted by two NNs. The mutual information between two variables, output F_i of network i and output F_j of network j, is given by

$$I(F_i; F_j) = H(F_i) + H(F_j) - H(F_i, F_j) \tag{1}$$

where $H(F_i)$ is the entropy of F_i, $H(F_j)$ is the entropy of F_j, and $H(F_i, F_j)$ is the joint differential entropy of F_i and F_j. The equation shows that joint differential entropy can only have high entropy if the mutual information between two variables is low, while each variable has high individual entropy. That is, the lower mutual information two variables have, the more different they are. By minimizing the mutual information between variables extracted by two NNs, two NNs are forced to convey different information about some features of their input.

In this paper, negative correlation learning is firstly analyzed in terms of minimization of mutual information on a regression task. Secondly, two fitness sharing techniques based on mutual information and the idea of covering set are introduced into the hybrid learning system. Through negative correlation learning and evolutionary learning, a diverse and cooperative population of NNs can be evolved. The effectiveness of such hybrid learning approach was tested on two real-world problems.

The rest of this paper is organized as follows: Section 2 explores the connections between the mutual information and the correlation coefficient, and explains how negative correlation learning can be used to minimize mutual information. Section 3 analyzes negative correlation learning via the metrics of mutual information on a regression task. Section 4 describes the hybrid learning system for evolving NN ensembles. Section 5 presents experimental results of the hybrid learning system. Finally, Section 6 concludes with a summary of the paper.

2 Minimizing Mutual Information by Negative Correlation Learning

2.1 Minimization of Mutual Information

Suppose the output F_i of network i and the output F_j of network j are Gaussian random variables. Their variances are σ_i^2 and σ_j^2, respectively. The mutual information between F_i and F_j can be defined by Eq.(1). The differential entropy $h(F_i)$ and $h(F_j)$ are given by $h(F_i) = \frac{1}{2}[1 + \log(2\pi\sigma_i^2)]$ and $h(F_j) = \frac{1}{2}[1 + \log(2\pi\sigma_j^2)]$. The joint differential entropy $h(F_i, F_j)$ is given by

$$h(F_i, F_j) = 1 + \log(2\pi) + \frac{1}{2}\log[\sigma_i^2\sigma_i^2(1 - \rho_{ij}^2)] \tag{2}$$

where ρ_{ij} is the correlation coefficient of F_i and F_j

$$\rho_{ij} = \frac{E[(F_i - E[F_i])(F_j - E[F_j])]}{\sigma_i^2 \sigma_j^2} \tag{3}$$

By substituting F_i, F_j, and Eq. (2) in (1), we get

$$I(F_i; F_j) = -\frac{1}{2}\log(1 - \rho_{ij}^2) \tag{4}$$

From Eq.(4), we may make the following statements:

1. If F_i and F_j are uncorrelated, the correlation coefficient ρ_{ij} is reduced to zero, and the mutual information $I(F_i; F_j)$ becomes very small.
2. If F_i and F_j are highly positively correlated, the correlation coefficient ρ_{ij} is close to 1, and mutual information $I(F_i; F_j)$ becomes very large.

Both theoretical and experimental results (Clemen and Winkler 1985) have indicated that when individual networks in an ensemble are unbiased, average procedures are most effective in combining them when errors in the individual networks are negatively correlated and moderately effective when the errors are uncorrelated. There is little to be gained from average procedures when the errors are positively correlated. In order to create a population of NNs that are as uncorrelated as possible, the mutual information between each individual NN and the rest of population should be minimized. Minimizing the mutual information between each individual NN and the rest of population is equivalent to minimizing the correlation coefficient between them.

2.2 Negative Correlation Learning

We consider estimating y by forming an NN ensemble whose output is a simple averaging of outputs F_i of a set of NNs. Given the training data set $D = \{(\mathbf{x}(1), y(1)), \cdots, (\mathbf{x}(N), y(N))\}$, all the individual networks in the ensemble are trained on the same training data set D

$$F(n) = \frac{1}{M}\Sigma_{i=1}^{M} F_i(n) \tag{5}$$

where $F_i(n)$ is the output of individual network i on the nth training pattern $\mathbf{x}(n)$, $F(n)$ is the output of the NN ensemble on the nth training pattern, and M is the number of individual networks in the NN ensemble.

The idea of negative correlation learning (Liu and Yao 1999a, Liu and Yao 1999b) is to introduce a correlation penalty term into the error function of each individual network so that the mutual information among the ensemble can be minimized. The error function E_i for individual i on the training data set $D = \{(\mathbf{x}(1), y(1)), \cdots, (\mathbf{x}(N), y(N))\}$ in negative correlation learning is defined by

$$E_i = \frac{1}{N}\Sigma_{n=1}^{N} E_i(n) = \frac{1}{N}\Sigma_{n=1}^{N}\left[\frac{1}{2}(F_i(n) - y(n))^2 + \lambda p_i(n)\right] \tag{6}$$

where N is the number of training patterns, $E_i(n)$ is the value of the error function of network i at presentation of the nth training pattern, and $y(n)$ is the desired output of the nth training pattern. The first term in the right side of Eq.(6) is the mean-squared error of individual network i. The second term p_i is a correlation penalty function. The purpose of minimizing p_i is to negatively correlate each individual's error with errors for the rest of the ensemble. The parameter λ is used to adjust the strength of the penalty.

The penalty function p_i has the form

$$p_i(n) = -\frac{1}{2}(F_i(n) - F(n))^2 \tag{7}$$

The partial derivative of E_i with respect to the output of individual i on the nth training pattern is

$$\begin{aligned}\frac{\partial E_i(n)}{\partial F_i(n)} &= F_i(n) - y(n) - \lambda(F_i(n) - F(n)) \\ &= (1 - \lambda)(F_i(n) - y(n)) + \lambda(F(n) - y(n))\end{aligned} \tag{8}$$

where we have made use of the assumption that the output of ensemble $F(n)$ has constant value with respect to $F_i(n)$. The value of parameter λ lies inside the range $0 \leq \lambda \leq 1$ so that both $(1 - \lambda)$ and λ have nonnegative values. The standard back-propagation (BP) (Rumelhart et al. 1986) algorithm 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.(8) after the presentation of each training pattern. One complete presentation of the entire training set during the learning process is called an *epoch*. Negative correlation learning from Eq.(8) 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 NN.

From Eq. (8), we may make the following observations. 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 $y(n)$, but also the difference between $F(n)$ and $y(n)$. That is, negative correlation learning considers errors what all other NNs have learned while training an NN.

3 Simulation Results

In order to understand how negative correlation learning minimizes mutual information, this section analyzes it through measuring mutual information on a regression task in three cases: noise free condition, small noise condition, and large noise condition.

3.1 Simulation Setup

The regression function investigated here is

$$f(\mathbf{x}) = \frac{1}{13}\left[10sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5\right] - 1 \tag{9}$$

where $\mathbf{x} = [x_1, \ldots, x_5]$ is an input vector whose components lie between zero and one. The value of $f(\mathbf{x})$ lies in the interval $[-1, 1]$. This regression task has been used by Jacobs (Jacobs 1997) to estimate the bias of mixture-of-experts architectures and the variance and covariance of experts' weighted outputs.

Twenty-five training sets, $(\mathbf{x}^{(k)}(l), y^{(k)}(l))$, $l = 1, \cdots, L$, $L = 500$, $k = 1, \cdots, K$, $K = 25$, were created at random. Each set consisted of 500 input-output patterns in which the components of the input vectors were independently sampled from a uniform distribution over the interval (0,1). In the noise free condition, the target outputs were not corrupted by noise; in the small noise condition, the target outputs were created by adding noise sampled from a Gaussian distribution with a mean of zero and a variance of $\sigma^2 = 0.1$ to the function $f(\mathbf{x})$; in the large noise condition, the target outputs were created by adding noise sampled from a Gaussian distribution with a mean of zero and a variance of $\sigma^2 = 0.2$ to the function $f(\mathbf{x})$.

A testing set of 1024 input-output patterns, $(\mathbf{t}(n), d(n))$, $n = 1, \cdots, N$, $N = 1024$, was also generated. For this set, the components of the input vectors were independently sampled from a uniform distribution over the interval (0,1), and the target outputs were not corrupted by noise in all three conditions.

Each individual network in the ensemble is a multilayer perceptron with one hidden layer. All the individual networks have five hidden nodes in an ensemble architecture. The hidden node function is defined by the logistic function. The network output is a linear combination of the outputs of the hidden nodes.

For each estimation of mutual information among an ensemble, twenty-five simulations were conducted. In each simulation, the ensemble was trained on a different training set from the same initial weights distributed inside a small range so that different simulations of an ensemble yielded different performances solely due to the use of different training sets. Such simulation setup follows the suggestions from Jacobs (Jacobs 1997).

3.2 Measurement of Mutual Information

The average outputs of the ensemble and the individual network i on the nth pattern in the testing set, $(\mathbf{t}(n), d(n))$, $n = 1, \cdots, N$, are denoted respectively by $\overline{F}(\mathbf{t}(n))$ and $\overline{F}_i(\mathbf{t}(n))$, which are given by

$$\overline{F}(\mathbf{t}(n)) = \frac{1}{K}\Sigma_{k=1}^{K} F^{(k)}(\mathbf{t}(n)) \tag{10}$$

and

$$\overline{F}_i(\mathbf{t}(n)) = \frac{1}{K}\Sigma_{k=1}^{K} F_i^{(k)}(\mathbf{t}(n)) \tag{11}$$

where $F^{(k)}(\mathbf{t}(n))$ and $F_i^{(k)}(\mathbf{t}(n))$ are the outputs of the ensemble and the individual network i on the nth pattern in the testing set from the kth simulation, respectively, and $K = 25$ is the number of simulations. The

correlation coefficient between network i and network j is given by

$$\rho_{ij} = \frac{\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}\left(F_i^{(k)}(\mathbf{t}(n)) - \overline{F}_i(\mathbf{t}(n))\right)\left(F_j^{(k)}(\mathbf{t}(n)) - \overline{F}_j(\mathbf{t}(n))\right)}{\sqrt{\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}\left(F_i^{(k)}(\mathbf{t}(n)) - \overline{F}_i(\mathbf{t}(n))\right)^2 \Sigma_{n=1}^{N}\Sigma_{k=1}^{K}\left(F_j^{(k)}(\mathbf{t}(n)) - \overline{F}_j(\mathbf{t}(n))\right)^2}} \tag{12}$$

From Eq.(4), the integrated mutual information among the ensembles can be defined by

$$E_{mi} = -\frac{1}{2}\Sigma_{i=1}^{M}\Sigma_{j=1,j\neq i}^{M} log(1-\rho_{ij}^2) \tag{13}$$

The integrated mean-squared error (MSE) on the testing set can also defined by

$$E_{test\_mse} = \frac{1}{N}\Sigma_{n=1}^{N}\frac{1}{K}\Sigma_{k=1}^{K}\left(F^{(k)}(\mathbf{t}(n)) - d(n)\right)^2 \tag{14}$$

3.3 Results of Negative Correlation Learning

The results of negative correlation learning for the different values of λ at epoch 2000 are given in Table 1. For the noise free condition, the results suggest that E_{test_mse} appeared to decrease with increasing value of λ. The mutual information E_{mi} among the ensemble decreased as the value of λ increased when $0 \leq \lambda \leq 0.5$. However, when λ increased further to 0.75 and 1, the mutual information E_{mi} had larger values. The reason of having larger mutual information at $\lambda = 0.75$ and $\lambda = 1$ is that some correlation coefficients had negative values and the mutual information depends on the absolute values of correlation coefficients.

For the small noise (variance $\sigma^2 = 0.1$) and large noise (variance $\sigma^2 = 0.2$) conditions, the results show that there were same trends for E_{mi} and E_{test_mse} in both noise free and noise conditions when $\lambda \leq 0.5$. That is, E_{mi} and E_{test_mse} appeared to decrease with increasing value of λ. However, E_{test_mse} appeared to decrease first and then increase with increasing value of λ. Choosing a proper value of λ is important, and also problem dependent. For the noise conditions used for this regression task and the ensemble architectured used, the performance of the ensemble was optimal for $\lambda = 0.5$ among the tested values of λ in the sense of minimizing the MSE on the testing set.

Table 1. The results of negative correlation learning for different λ values at epoch 2000.

	Noise free		Small noise ($\sigma^2 = 0.1$)		Large noise ($\sigma^2 = 0.2$)	
λ	E_{mi}	E_{test_mse}	E_{mi}	E_{test_mse}	E_{mi}	E_{test_mse}
0	0.3706	0.0016	6.5495	0.0137	6.7503	0.0249
0.25	0.1478	0.0013	3.8761	0.0128	3.9652	0.0235
0.5	0.1038	0.0011	1.4547	0.0124	1.6957	0.0228
0.75	0.1704	0.0007	0.3877	0.0126	0.4341	0.0248
1	0.6308	0.0002	0.2431	0.0290	0.2030	0.0633

4 Evolving Neural Network Ensembles

The major steps of the proposed hybrid learning system (HLS) 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. Stop the evolution process if the maximum number of generations has been reached. Otherwise, $k = k + 1$ and go to Step 3.

There are two levels of adaptation in HLS: negative correlation learning at the individual level and evolutionary learning based on evolutionary programming (EP) (Fogel 1995) at the population level. In HLS, an evolutionary algorithm based on evolutionary programming (Fogel 1995) has been used to search for a population of diverse individual NNs that solve a problem together.

The fitness evaluation in HLS is carried out by fitness sharing. Explicit and implicit fitness sharing have been proposed to encourage speciation in recent years (Mahfoud 1995, Darwen and Yao 1997). 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 HLS because the individual networks are not represented by binary strings in HLS.

Two fitness sharing techniques were used in HLS. One fitness sharing 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 fitness zero. Otherwise, all the individuals in the population receive fitness zero. 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.

The other fitness sharing is based on the minimization of mutual information. In order to create a population of NNs that are as uncorrelated as possible, the mutual information between each individual NN and the rest of population should be minimized. The fitness f_i of individual network i in the population can therefore be evaluated by the mutual information:

$$f_i = \frac{1}{\sum_{j \neq i} I(F_i, F_j)} \tag{15}$$

Minimization of mutual information has the similar motivations as fitness sharing. Both of them try to generate individuals that are different from others, though overlaps are allowed.

5 Experimental Studies of HLS

This section investigates HLS on two benchmark problems: the Australian credit card assessment problem and the diabetes problem. Both data sets were obtained from the UCI Machine Learning 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 binarized 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 tell the difference between HLS with minimization of the size of covering sets and HLS with minimization of mutual information. We name them HLSCS and HLSMI, respectively. The experimental setup is the same as the previous experimental setup described in (Michie et al. 1994, Liu et al. 2000). The n-fold cross-validation technique (Stone 1974) 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.

All parameters used in HLS except for the number of training epochs were set to be the same for both the Australian credit card assessment problem and the diabetes problem: the population size M (25), the number of generations (200), the reproduction block size n_b (2), and the strength parameter λ (0.5). The number of training epochs n_e was set to 3 for the Australian credit card data set, and 15 for the diabetes data set. 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.

5.1 Comparisons between HLSMI and HLSCS

Tables 2–3 show the results of HLSMI for the two data sets, where the ensembles were constructed by the whole population in the last generation. Three combination methods for determining the output of the ensemble have been investigated in HLSMI. 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. The *accuracy rate* refers to the percentage of correct classifications produced by HLSMI. In comparison with the accuracy rates obtained by three combination methods, majority voting and winner-takes-all outperformed simple averaging on both problems. Simple averaging is more suitable to the regression type of tasks. Because both problems studied in this section are classification tasks, majority voting and winner-takes-all are better choices.

Table 2. Comparison of accuracy rates between HLSCS and HLSMI for the Australian credit card data set. The results are averaged on 10-fold cross-validation. *Mean* and *SD* indicate the mean value and standard deviation, respectively.

	Simple Averaging		Majority Voting		Winner-Takes-All	
Methods	Mean	SD	Mean	SD	Mean	SD
HLSMI	0.864	0.038	0.870	0.040	0.868	0.039
HLSCS	0.855	0.039	0.857	0.039	0.865	0.028

Table 3. Comparison of accuracy rates between HLSCS and HLSMI for the diabetes data set. The results are averaged on 12-fold cross-validation. *Mean* and *SD* indicate the mean value and standard deviation, respectively.

	Simple Averaging		Majority Voting		Winner-Takes-All	
Methods	Mean	SD	Mean	SD	Mean	SD
HLSMI	0.771	0.049	0.777	0.046	0.773	0.051
HLSCS	0.766	0.039	0.764	0.042	0.779	0.045

Tables 2–3 compare the results produced by HLSMI and HLSCS using three combination methods. Majority voting supports HLSMI, while winner-takes-all favors HLSCS. Since the only difference between HLSMI and HLSCS is the fitness sharing scheme used, the results suggest that combination methods and fitness sharing are closely related to each other. Further studies are needed to probe the relationship of these two.

5.2 Evolution Process of Mutual Information

In order to observe the evolution process of the mutual information among the populations evolved by HLSMI and HLSCS, Figure 1 shows the evolution of the mean of sum of the mutual information among the population for the Australian credit card data set. The sum of the mutual information among the population is calculated by

$$I_{population} = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} I(F_i, F_j) \tag{16}$$

where F_i is the vector formed by the output of network i on the training set, and F_j is the vector formed by the output of network j on the training set. The mean of $I_{population}$ is averaged on 10-fold cross-validation.

In Figure 1, the values of mutual information of populations evolved by both HLSMI and HLSCS reduced quickly at the initial generations. After that, the values of mutual information of population evolved by HLSCS decreased very slowly. However, HLSMI was able to steadily decrease the mutual information among the population through the whole evolution process.

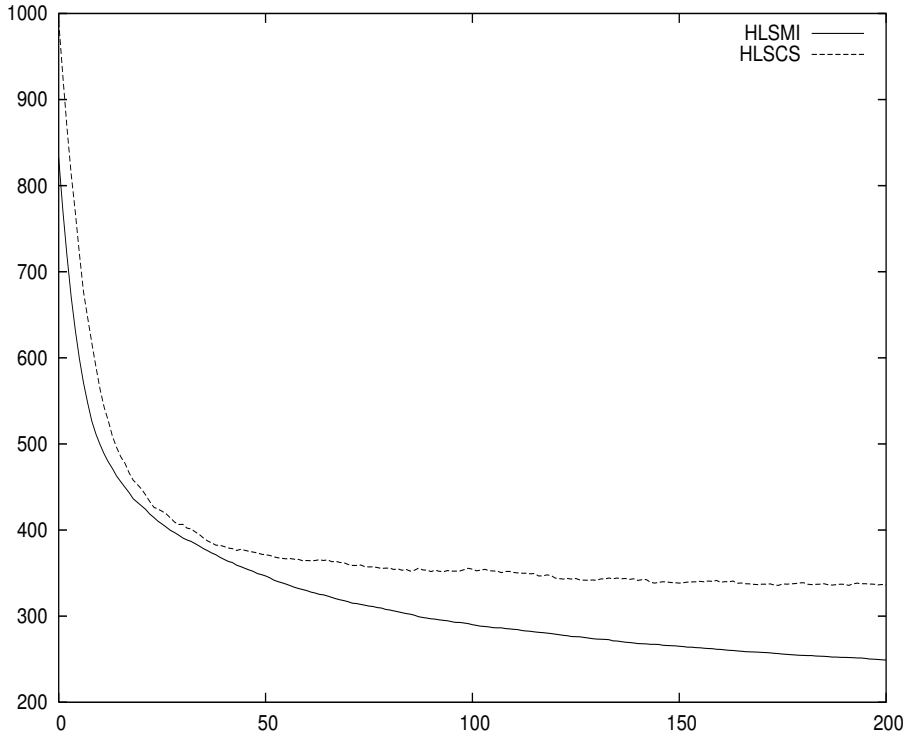

Figure 1. The evolution of the mean of sum of the mutual information among the populations evolved by HLSMI ans HLSCS for the Australian credit card data set. The mean is averaged on 10-fold cross-validation. The vertical axis is the mutual information value and the horizontal axis is the number of generations.

Table 4. Comparison among HLSMI, HLSCS, and others (Michie et al. 1994) 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. The error rates of HLSMI using majority voting and HLSCS using the winner-takes-all are shown in the table.

Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate	Algorithm	Error Rate
HLSMI	0.130	ALLOC80	0.201	AC^2	0.181	Cal5	0.131
HLSCS	0.135	k-NN	0.181	Baytree	0.171	Kohonen	FD
Discrim	0.141	CASTLE	0.148	NaiveBay	0.151	DIPOL92	0.141
Quadisc	0.207	CART	0.145	CN2	0.204	Backprop	0.154
Logdisc	0.141	IndCART	0.152	C4.5	0.155	RBF	0.145
SMART	0.158	NewID	0.181	ITrule	0.137	LVQ	0.197

5.3 Comparisons with EENCL and Other Work

Tables 4–5 compare the results produced by HLSMI, HLSCS, and other 22 algorithms tested by Michie *et al.* (Michie et al. 1994). These 22 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-based methods (CN2, ITrule); NNs (Backprop, Kohonen, LVQ, RBF, DIPOL92). More details about these algorithms appear in (Michie et al. 1994). HLSMI and HLSCS used the same data setup as in (Michie et al. 1994), 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. Only the results of HLSMI using majority voting and HLSCS using the winner-takes-all combination method are shown in Tables 4–5. The average testing error rates of HLSMI and HLSCS are 0.130 and 0.135 for the Australian credit card data set, and 0.223 and 0.221 for the diabetes data set. As demonstrated by the results, HLSMI and HLSCS have been able to achieve the generalization performance comparable to or better than the best of 22 algorithms tested (Michie et al. 1994) for both the

Australian credit card data set and the diabetes data set.

Table 5. Comparison among HLSMI, EENCL, and others (Michie et al. 1994) in terms of the average testing error rate for the diabetes data set. The results are averaged on 12-fold cross-validation. The error rates of HLSMI using majority voting and HLSCS using the winner-takes-all are shown in the table.

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

6 Conclusions

Fitness sharing is important to evolve a diverse and cooperative population of NNs that make a good NN ensemble. Two fitness sharing techniques have been introduced and compared in the hybrid learning system. Because of being more accurate measurement of similarity, mutual information not only provides a better fitness sharing scheme, but also leads to study on the connections between fitness sharing and mutual information concept.

There are a large number of combination methods which have been proposed in the fields of NNs, machine learning and statistics (Jacobs 1995). Three combination methods are adopted in the hybrid learning system, including simple averaging, majority voting and winner-takes-all, because of their simplicity and effectiveness. It would be desirable to study a more complex model, such as a *supra Bayesian* methods (Jacobs 1995), which are theoretically well-motivated and normative. The problems existing in such complex models are that they may be impractical on some real-world tasks and involve expensive computation. It will be an important topic of how to apply these complex models in our hybrid learning system for real-world applications.

References

Akaike, H. (1974), "A new look at the statistical model identification," *IEEE Trans. Appl. Comp.*, AC-19:716-723.

Clemen, R. T. and R. .L Winkler, R. L. (1985), "Limits for the precision and value of information from dependent sources," *Operations Research*, 33:427-442.

Darwen, P. J. and Yao, X. (1997), "Speciation as automatic categorical modularization," *IEEE Trans. on Evolutionary Computation*, 1(2):101-108.

Fogel, D. B. (1995), *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, IEEE Press, New York, NY.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

Jacobs, R. A. (1995), "Methods for combining experts' probability assessments," *Neural Computation*, 7:867-888.

Jacobs, R. A. (1997), "Bias/variance analyses of mixture-of-experts architectures," *Neural Computation*, 9:369-383.

Liu, Y. and Yao, X. (1998), "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*, Springer-Verlag, Berlin, pp. 623-632.

Liu, Y. and Yao, X. (1999a), "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(6):716-725.

Liu, Y. and Yao, X. (1999b), "Ensemble learning via negative correlation," *Neural Networks*, 12:1399-1404.

Liu, Y., Yao, X., and Higuchi, T. (2000), "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, 4(4):380-387.

Liu, Y., Yao, X., Zhao, Q., and Higuchi, T. (2001), "Evolving a cooperative population of neural networks by minimizing mutual information," *Proc. of the 2001 Conference on Evolutionary Computation*, IEEE Press, pp. 384-389.

Liu, Y. and Yao, X. (2002), "Maintaining population diversity by minimizing mutual information," *Proceedings of the 2002 Genetic and Evolutionary Computation Conference (GECCO2002)*, Morgan Kaufmann, pp. 652-656.

Liu, Y. and Yao, X. (2002), "Learning and evolution by minimization of mutual information," *Parallel Problem Solving from Nature – PPSN VII: Proc. of the 7th International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, Vol. 2439, Springer, pp. 495-504.

Lubbe, J. C. A. van der (1999), *Information Theory*, Prentice-Hall International, Inc., 2nd edition.

Mahfoud, S. W. (1995), *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign.

McKay, R. I. and Abbass, H. A. (2001), "Analyzing anti correlation in ensemble learning," *Proceedings of the 2001 Conference on Artificial Neural Networks and Expert Systems*, pp. 22-27.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Limited, London.

Rissanen, J. (1978), "Modeling by shortest data description," *Automatica*, 14:465-471.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986), "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*, MIT Press, Cambridge, MA, pp. 318-362.

Stone, M. (1974), "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society*, 36:111-147.

Wallace, C. S. and Patrick, J. D. (1991), "Coding decision trees," Technical Report 91/153, Dept. of Computer Science, Monash University, Clayton, Victoria 3168, Australia.

Wolpert, D. H. (1990), "A mathematical theory of generalization," *Complex Systems*, 4:151-249.

Yao, X. (1991), "Evolution of connectionist networks," In T. Dartnall, editor, *Preprints of the Int'l Symp. on AI, Reasoning & Creativity*, Griffith University, Queensland, Australia, pp. 49-52.

Yao, X. (1993), "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, 8(4):539-567.

Yao, X. (1994), "The evolution of connectionist networks," In T. Dartnall, editor, *Artificial Intelligence and Creativity*, Kluwer Academic Publishers, Dordrecht, pp. 233-243.

Yao, X. (1995), "Evolutionary artificial neural networks," In A. Kent and J. G. Williams, editors, *Encyclopedia of Computer Science and Technology*, volume 33, Marcel Dekker Inc., New York, NY 10016, pp. 137-170.