Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble

Yong Liu, *Student Member, IEEE,* and Xin Yao, *Senior Member, IEEE*

***Abstract*— This paper presents a new cooperative ensemble learning system (CELS) for designing neural network ensembles. The idea behind CELS is to encourage different individual networks in an ensemble to learn different parts or aspects of a training data so that the ensemble can learn the whole training data better. In CELS, 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. CELS can create negatively correlated neural networks using a correlation penalty term in the error function to encourage such specialization. This paper analyzes CELS in terms of bias-variance-covariance tradeoff. CELS has also been tested on the Mackey–Glass time series prediction problem and the Australian credit card assessment problem. The experimental results show that CELS can produce neural network ensembles with good generalization ability.**

***Index Terms*— Generalization, negative correlation learning, neural network ensembles.**

I. Introduction

NEURAL network ensembles [1], [2] have been used increasingly in recent years to improve classifier's generalization. Both theoretical and experimental results [3], [4] 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.

There are several methods of designing neural network ensembles. Most of them follow the two-stage design process [1]; first generating individual networks, and then combining them. Usually, the individual networks are trained independent of each other. One of the disadvantages of such an approach is the loss of interaction among the individual networks during learning. There is no feedback from the combination stage to the individual design stage. It is possible that some of the independently designed individual networks do not make much contribution to the whole ensemble.

This paper proposes a new cooperative ensemble learning system (CELS). The idea behind CELS is to encourage different individual networks to learn different parts or aspects of a training data so that the ensemble can learn the whole training data better. CELS is different from previous work on designing neural network ensembles. It emphasizes interaction and cooperation among the individual networks in the ensemble, and uses an unsupervised penalty term in the error function to produce biased individual networks whose errors tend to be negatively correlated. This approach is quite different from existing ones [4], [5] which train the individual networks independently or sequentially.

Rosen [6] proposed an ensemble learning algorithm using decorrelated neural networks. 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 previously trained networks. 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 previously trained networks in the ensemble so that the errors of the individual networks are not necessarily negatively correlated. CELS extends Rosen's work to simultaneous training of negatively correlated neural networks. Such extension has produced significant improvement in neural network ensembles' performance. Negatively correlated neural networks can be easily obtained in CELS. Theoretical and empirical studies will be carried out in this paper to show why and how CELS works.

CELS is also different from the mixtures-of-experts (ME) architecture [7] that consists of a gating network and a number of expert networks although ME architecture can also produce biased individual networks whose estimates are negatively correlated. CELS does not need a separate gating network. It uses a totally different error function. The λ parameter in CELS provides a convenient way to balance the bias-variance-covariance tradeoff. ME architecture does not provide such control over the tradeoff.

CELS attempts to train and combine individual networks in the same learning process. That is, the goal of each individual training is to generate the best result for the whole ensemble. Such an approach is quite different from other ensemble approaches which separate individual design from average procedures.

CELS has been analyzed in terms of bias-variance-covariance tradeoff. It has also been tested on the Mackey–Glass time series prediction problem and the Australian credit card assessment problem. The experimental results obtained by CELS are better than those obtained by other algorithms in terms of generalization.

Manuscript received June 19, 1998; revised January 15, 1999. This paper was recommended by Associate Editor P. Willett.

Y. Liu is with the Evolvable Systems Laboratory, Computer Science Division, Electrotechnical Laboratory, Ibaraki 305-8568, Japan (e-mail: yliu@etl.go.jp).

X. Yao is with the School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

Publisher Item Identifier S 1083-4419(99)08052-8.

The rest of this paper is organized as follows: Section II briefly overviews the neural network learning and the bias-variance tradeoff. Section III describes CELS and gives its motivation based on the bias-variance-covariance tradeoff. Section IV analyzes CELS via the metrics of bias, variance, and covariance on a regression task. Section V presents the experimental results on CELS and some discussions. Finally, Section VI concludes with a summary of the paper and a few remarks.

II. Neural Network Learning

Suppose that we have a training set

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

where $\boldsymbol{x} \in R^p$, 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.

The functional relationship between $\boldsymbol{x}$ and d can be expressed as

$$d = g(\boldsymbol{x}) + \epsilon \tag{1}$$

where $g(\boldsymbol{x})$ is some function of vector $\boldsymbol{x}$, and ϵ is a random variable that reflects the fact that simultaneous specification of a set of input vectors, $\{\boldsymbol{x}(1), \cdots, \boldsymbol{x}(N)\}$, does not uniquely specify an output value (unless ϵ is a constant). The statistical model described by (1) is called a *regressive model*. In this model the function $g(\boldsymbol{x})$ is defined by [8]

$$g(\boldsymbol{x}) = E[d|\boldsymbol{x}] \tag{2}$$

where E is the statistical expectation operator.

The exact functional relationship between $\boldsymbol{x}$ and d is usually unknown. The purpose of neural network learning is to use $\boldsymbol{x}$ to explain or predict d. It does so by encoding the empirical knowledge represented by the training set D into a set of synaptic weights, $\boldsymbol{w}$. One criterion for optimizing weights is the minimization of the mean-square error

$$J(\boldsymbol{w}) = E[(d - F(\boldsymbol{x}, \boldsymbol{w}))^2] \tag{3}$$

where $F(\boldsymbol{x}, \boldsymbol{w})$ is the actual response of the network.

The error function $J(\boldsymbol{w})$ may be expressed as the sum of two terms [9]

$$J(\boldsymbol{w}) = E[(d - g(\boldsymbol{x}))^2] + E[(g(\boldsymbol{x}) - F(\boldsymbol{x}, \boldsymbol{w}))^2]. \tag{4}$$

Note that the first term of (4) is independent of $\boldsymbol{w}$. It is sufficient to minimize the second term. To be explicit about dependence on the training set D, the approximating function may be rewritten as $F(\boldsymbol{x}, D)$. Consider then the mean-squared error of the function $F(\boldsymbol{x}, D)$ as an estimator of the regression function $g(\boldsymbol{x}) = E[d|\boldsymbol{x}]$, which is defined by

$$E_D[(E[d|\boldsymbol{x}] - F(\boldsymbol{x}, D))^2]$$

where the expectation operator E_D represents the average over all the training sets D of given size N. Taking expectations with respect to the training set D, we can get the well-known separation of the mean-squared error [9], [10]

$$\begin{aligned} E_D[(E[d|\boldsymbol{x}] - F(\boldsymbol{x}, D))^2] &= (E_D[F(\boldsymbol{x}, D)] - E[d|\boldsymbol{x}])^2 \\ &\quad + E_D[(F(\boldsymbol{x}, D) - E_D[F(\boldsymbol{x}, D)])^2]. \end{aligned} \tag{5}$$

The first term of (5) is the square of the bias of the approximating function $F(\boldsymbol{x}, D)$ measured with respect to the regression function $g(\boldsymbol{x}) = E[d|\boldsymbol{x}]$, and the second term represents the variance of the approximating function $F(\boldsymbol{x}, D)$.

Accordingly, (5) states that the mean-square value of the estimation error between the regression function $g(\boldsymbol{x})$ and approximating function $F(\boldsymbol{x}, D)$ consists of the sum of two terms: bias squared and variance. To achieve good performance, the bias and variance of the approximating function $F(\boldsymbol{x}, D)$ should both be small. In the case of a training set with finite size, although there can be neural networks with both small bias and variance, usually there is a tradeoff between the two: attempts to decrease bias by introducing more parameters in the network often tend to increase variance; attempts to reduce variance by reducing parameters in the network often tend to increase bias.

III. Ensemble Network Learning

A. Bias-Variance-Covariance Tradeoff

In this section, we consider estimating $g(\boldsymbol{x}) = E[d|\boldsymbol{x}]$ by forming a simple averaging of a set of $F_i(\boldsymbol{x}, D)$ which are trained on the same training data set D

$$F(\boldsymbol{x}, D) = \frac{1}{M} \Sigma_{i=1}^{M} F_i(\boldsymbol{x}, D) \tag{6}$$

where M is the number of neural network estimators. Consequently, the expected mean-squared error of the combined system can be written in terms of individual network output [11], [7]

$$\begin{aligned} E_D[(E[d|x] - F(\boldsymbol{x}, D))^2] &= (E_D[F(\boldsymbol{x}, D)] - E[d|x])^2 \\ &\quad + E_D\left[\frac{1}{M^2} \Sigma_{i=1}^{M} (F_i(\boldsymbol{x}, D) - E_D[F_i(\boldsymbol{x}, D)])^2\right] \\ &\quad + E_D\left[\frac{1}{M^2} \Sigma_{i=1}^{M} \Sigma_{j \neq i} (F_i(\boldsymbol{x}, D) - E_D[F_i(\boldsymbol{x}, D)]) \right. \\ &\quad \left. \times (F_j(\boldsymbol{x}, D) - E_D[F_j(\boldsymbol{x}, D)])\right] \end{aligned} \tag{7}$$

where the first term is the square of the bias of the combined system, the second and third terms are the variance and covariance of the outputs of the individual networks, respectively. Similar to the bias-variance tradeoff for a single network, there is the bias-variance-covariance tradeoff for neural network ensembles.

While the variance in (7) can be seen to decay at $1/M$, the covariance is finite unless the covariances between individual networks are very small. It has been found that the combining results are weakened if the errors of individual networks are positively correlated [3], [4]. Common approaches to dealing with this issue are to obtain unbiased estimators whose estimation errors are as weakly correlated as possible [5]. In contrast, this paper describes a new approach to create biased estimators whose estimation errors are negatively correlated.

B. Simultaneous Learning of Negatively Correlated Neural Networks

CELS introduces a correlation penalty term into the error function of each individual network so that the individual networks can be trained simultaneously and interactively. The error function E_i for individual network i in CELS is defined by

$$\begin{aligned} E_i &= \frac{1}{N}\,\Sigma_{n=1}^{N} E_i(n) \\ &= \frac{1}{N}\,\Sigma_{n=1}^{N}\left[\frac{1}{2}\,(d(n)-F_i(n))^2+\lambda p_i(n)\right] \end{aligned} \tag{8}$$

where N is the number of training patterns, $E_i(n)$ is the value of the error of network i at presentation of the nth training pattern, $F_i(n)$ is the output of network i on the nth training pattern, and 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 $0 \le \lambda \le 1$ is used to adjust the strength of the penalty. The function p_i can be chosen as

$$p_i(n) = (F_i(n)-g(n))\Sigma_{j\neq i}\,(F_j(n)-g(n)) \tag{9}$$

where $g(\boldsymbol{x}) = E[d|\boldsymbol{x}]$. For the noise free data, i.e., $g(n) = d(n)$, we have

$$p_i(n) = (F_i(n)-d(n))\Sigma_{j\neq i}\,(F_j(n)-d(n)). \tag{10}$$

Unfortunately, the value of g is unknown for noisy data. In such cases, the function p_i can be chosen as

$$p_i(n) = (F_i(n)-F(n))\Sigma_{j\neq i}(F_j(n)-F(n)) \tag{11}$$

where $F(n)$ is the output of the combined system on the nth training pattern. For the sake of convenience, the following discussion of CELS is for the noise free data.

In CELS, the standard back-propagation (BP) algorithm [12] with pattern-by-pattern updating has been used for weight adjustments. The partial derivative of E_i with respect to the output of network i on the nth training pattern is

$$\frac{\partial E_i(n)}{\partial F_i(n)} = F_i(n)-d(n)+\lambda\Sigma_{j\neq i}(F_j(n)-d(n)). \tag{12}$$

Weight updating of all the individual networks is performed simultaneously using (12) after the presentation of each training pattern. Note that the correlation penalty term is very easy to implement since it requires only a small change in the independent training without the correlation penalty term. One complete presentation of the entire training set during the learning process is called an *epoch.*

The sum of $E_i(n)$ over all i is

$$\begin{aligned} E(n) &= \Sigma_{i=1}^{M} E_i(n) \\ &= (\tfrac{1}{2}-\lambda)\Sigma_{i=1}^{M}(d(n)-F_i(n))^2 \\ &\quad + \lambda(\Sigma_{i=1}^{M}F_i(n)-Md(n))^2. \end{aligned} \tag{13}$$

From (8), (10)–(13), 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.

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 using BP. That is, independent training using BP for the individual networks is a special case of CELS.

3) For $\lambda = \frac{1}{2}$, (13) can be rewritten as

$$E(n) = \tfrac{1}{2}(\Sigma_{i=1}^{M}F_i(n)-Md(n))^2. \tag{14}$$

The right term in (14), denoted by E_{ave}, is the error function of the ensemble. From this point of view, CELS provides a novel way to decompose the learning task of the ensemble into a number of subtasks for the different individual networks.

4) For $\lambda = 1$, the following equality holds

$$\frac{\partial E_{\text{ave}}}{\partial F_i(n)} = \frac{\partial E_i(n)}{\partial F_i(n)}. \tag{15}$$

The minimization of the error function of the ensemble is achieved by minimizing the error functions of the individual networks.

IV. Bias-Variance-Covariance Estimation

This section analyzes CELS in terms of the bias-variance-covariance tradeoff on a regression task in order to understand why and how CELS works. The regression function is

$$\begin{aligned} f(\boldsymbol{x}) = \frac{1}{13}[&10\,\sin(\pi x_1 x_2)+20(x_3-\tfrac{1}{2})^2 \\ &+10x_4+5x_5]-1 \end{aligned} \tag{16}$$

where $\boldsymbol{x} = [x_1,\cdots,x_5]$ is an input vector whose components lie between zero and one. The value of $f(\boldsymbol{x})$ lies in the interval $[-1, 1]$. This regression task has been used by Jacobs [7] to estimate the bias of ME architectures and the variance and covariance of experts' weighted outputs.

The ensemble architecture used in our experiments consisted of a set of networks. Each individual network was a multilayer perceptron with one hidden layer. All the individual networks had the same number of hidden nodes in an ensemble architecture. The hidden node function was defined by the logistic function

$$\varphi(y) = \frac{1}{1+\exp(-y)}. \tag{17}$$

The network output was a linear combination of the outputs of the hidden nodes.

Twenty-five training sets 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). The target outputs were not corrupted by noise for Experiments 1 and 2. In Experiment 3, the target outputs were created by adding noise sampled from a Gaussian distribution with a mean of zero and a variance of σ^2 to the function $\boldsymbol{x}$. A testing set of 1024 input–output patterns, $(\boldsymbol{t}(n), y(n))$, $n = 1, \cdots, N = 1024$, 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. The correlation penalty term given in (10) was used in Experiments 1 and 2. The correlation penalty term given in (11) was used in Experiment 3.

Twenty-five simulations of each ensemble architecture were conducted. In each simulation, the architecture was trained on a different training set from the same initial weights distributed inside a small range so that different simulations of an architecture yielded different performances solely due to the use of different training sets. Such experimental setup follows the suggestions from Jacobs [7].

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

$$\overline{F}(n) = \frac{1}{K}\,\Sigma_{k=1}^{K} F^{(k)}(n) \tag{18}$$

and

$$\overline{F}_i(n) = \frac{1}{K}\Sigma_{k=1}^{K} F_i^{(k)}(n) \tag{19}$$

where $F^{(k)}(n)$ and $F_i^{(k)}(n)$ are the outputs of the ensemble and network i on the nth pattern $\boldsymbol{t}(n)$ in the testing set from the kth simulation, respectively, and K is the number of simulations. The integrated bias E_{bias}, integrated variance E_{var}, and integrated covariance E_{cov} of the ensemble are defined by, respectively,

$$E_{\text{bias}} \equiv \frac{1}{N}\Sigma_{n=1}^{N}(\overline{F}(n) - y(n))^2 \tag{20}$$

and

$$E_{\text{var}} \equiv \Sigma_{i=1}^{M}\frac{1}{N}\,\Sigma_{n=1}^{N}\frac{1}{K}\,\Sigma_{k=1}^{K}\frac{1}{M^2}(F_i^{(k)}(n) - \overline{F}_i(n))^2 \tag{21}$$

and

$$E_{\text{cov}} \equiv \Sigma_{i=1}^{M}\Sigma_{j=1,j\neq i}^{M}\frac{1}{N}\Sigma_{n=1}^{N}\frac{1}{K}\Sigma_{k=1}^{K} \cdot \frac{1}{M^2}(F_i^{(k)}(n) - \overline{F}_i(n))(F_j^{(k)}(n) - \overline{F}_j(n)). \tag{22}$$

We may also define the integrated mean-squared error (MSE) E_{mse} as

$$E_{\text{mse}} \equiv \frac{1}{N}\Sigma_{n=1}^{N}\frac{1}{K}\Sigma_{k=1}^{K}(F^{(k)}(n) - y(n))^2. \tag{23}$$

It is clear that the following equality holds

$$E_{\text{mse}} = E_{\text{bias}} + E_{\text{var}} + E_{\text{cov}}. \tag{24}$$

TABLE I

RESULTS OF INTEGRATED BIAS, INTEGRATED VARIANCE, INTEGRATED COVARIANCE, AND INTEGRATED MEAN-SQUARED ERROR ON THE TESTING SET IN CELS FOR DIFFERENT λ VALUES AT EPOCH 2000

	$\lambda = 0.0$	$\lambda = 0.25$	$\lambda = 0.5$
E_{bias}	0.001310	0.000464	0.000263
E_{var}	0.000151	0.000451	0.001004
E_{cov}	0.000140	−0.000264	−0.000827
$E_{var} + E_{cov}$	0.000291	0.000188	0.000177
E_{mse}	0.001601	0.000653	0.000440
	$\lambda = 0.75$	$\lambda = 0.875$	$\lambda = 1.0$
E_{bias}	0.000209	0.000172	0.001828
E_{var}	0.002145	0.003536	0.134295
E_{cov}	−0.001969	−0.003353	−0.129785
$E_{var} + E_{cov}$	0.000176	0.000183	0.004510
E_{mse}	0.000385	0.000355	0.006338

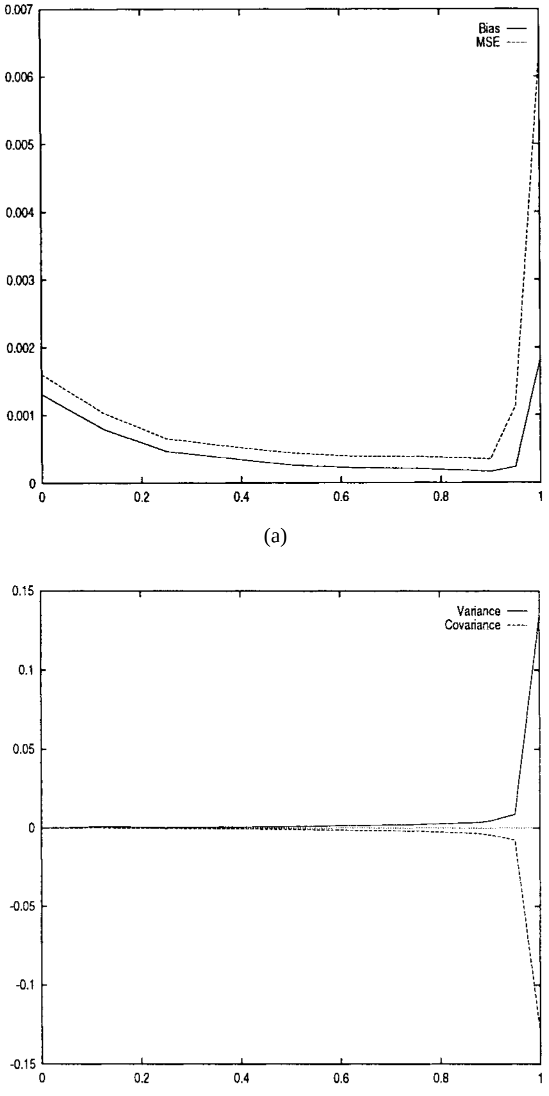
A. Experiment 1

The first experiment is aimed to investigate the dependence of E_{bias}, E_{var}, and E_{cov} on the strength parameter λ. The architecture of the ensemble was composed of eight individual networks. Each individual network had five hidden nodes. The learning-rate η in BP was set to 0.1. The results of CELS for the different values of λ at epoch 2000 are given in Table I and Fig. 1. The results suggest that E_{bias} appeared to decrease first and then increase with increasing value of λ. It was found that E_{bias} for $\lambda \approx 0.9$ seemed to be the minimum value for different values of λ. However, when λ became too big, E_{bias} increased dramatically. In such cases the individual learnings mainly minimized the correlation penalty terms in the error functions rather than the error of the ensemble. It seems that E_{var} increased as the value of λ increased, and E_{cov} decreased as the value of λ increased. It is important to note that the integrated covariance became negative during the training procedure.

It is interesting that CELS controls not only the variance and covariance of the individual networks, but also the bias of the combined system. Compared with independent training using BP (i.e., $\lambda = 0.0$ in CELS), although CELS created larger variance, the sum of the variance and covariance in CELS was smaller because of the negative covariance. At the same time, CELS reduced the bias of the ensemble significantly. From (24), the integrated MSE consists of the sum of three terms: the integrated bias, variance, and covariance. Therefore, CELS provides a control of bias, variance, and covariance through the choice of λ value to achieve good performance. For this regression task and the ensemble architecture used, it was observed that the bias-variance-covariance tradeoff was optimal for $\lambda \approx 0.9$ in the sense of minimizing the MSE.

B. Experiment 2

There are two aims of Experiment 2. The first is to investigate the dependence of E_{bias}, E_{var}, and E_{cov} on the individual network size. We used three different ensemble architectures, denoted by A_5, A_{10}, and A_{15}, which were composed of 8 individual networks. Each individual network in A_5, A_{10}, and A_{15} had 5, 10, and 15 hidden nodes, respectively. The second



(a)

(b)

Fig. 1. (a) Progress of integrated bias and integrated MSE on the testing set at epoch 2000 for different λ values. The vertical axis is the value of the integrated bias and integrated MSE and the horizontal axis is the value of λ. (b) Progress of integrated variance and integrated covariance on the testing set at epoch 2000 for different λ values. The vertical axis is the value of the integrated variance and integrated covariance and the horizontal axis is the value of λ.

is to compare the correlations among the individual networks and the squared bias of the individual networks created by CELS and independent training, respectively. The learning rate η in BP and strength parameter λ were set to 0.1 and 0.5, respectively.

The results of CELS for A_5, A_{10}, and A_{15} at epoch 2000 are given in Table II. As expected, A_{10} and A_{15}, which had more computational resources, performed slightly better than A_5. It is worth pointing out that larger individual network size does not necessarily improve the performance of the ensemble system. The choice of individual network size is problem dependent. Interestingly, although the sum of the variance and covariance did not change much among A_5, A_{10}, and A_{15}, their variance and covariance were quite different.

TABLE II
COMPARISON BETWEEN CELS ($\lambda = 0.5$) AND INDEPENDENT TRAINING (i.e., $\lambda = 0.0$ IN CELS) FOR THE DIFFERENT INDIVIDUAL NETWORK SIZES. THE RESULTS OF INTEGRATED BIAS, INTEGRATED VARIANCE, INTEGRATED COVARIANCE, AND INTEGRATED MSE ON THE TESTING SET ARE AT EPOCH 2000

	$\lambda = 0.5$		
	A_5	A_{10}	A_{15}
E_{bias}	0.000263	0.000236	0.000244
E_{var}	0.001004	0.000390	0.000467
E_{cov}	−0.000827	−0.000291	−0.000356
$E_{var} + E_{cov}$	0.000177	0.000099	0.000111
E_{mse}	0.000440	0.000335	0.000354
	$\lambda = 0.0$		
	A_5	A_{10}	A_{15}
E_{bias}	0.001310	0.000498	0.000415
E_{var}	0.000151	0.000049	0.000032
E_{cov}	0.000140	0.000053	0.000056
$E_{var} + E_{cov}$	0.000291	0.000102	0.000087
E_{mse}	0.001601	0.000600	0.000503

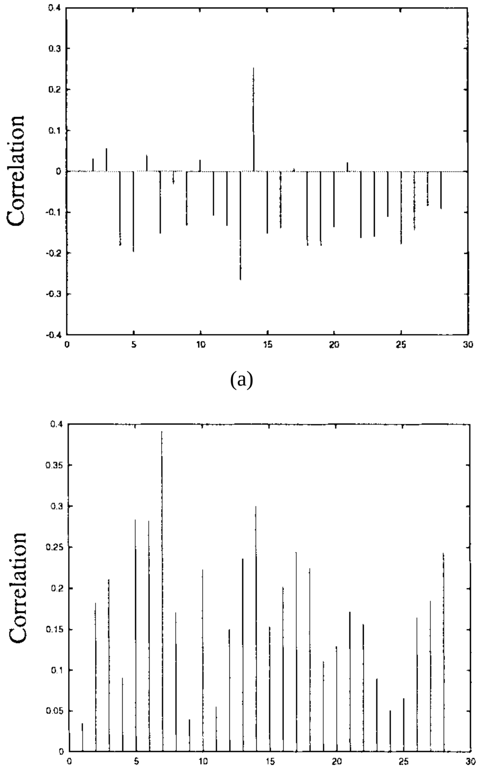
A_5, A_{10}, and A_{15} were also independently trained using BP without the correlation penalty terms (i.e., $\lambda = 0.0$ in CELS). The results are shown in Table II. It is apparent that the architectures trained with the correlation penalty terms performed better in terms of the integrated MSE values.

In order to observe the effect of the correlation penalty terms, Fig. 2 shows the correlations among the individual networks in A_{10} trained with and without the correlation penalty terms, respectively. The correlation between network i and network j is given by

$$\mathrm{Cor}(i,j) = \frac{\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}(F_i^{(k)}(n) - \overline{F}_i(n))}{\sqrt{\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}(F_i^{(k)}(n) - \overline{F}_i(n))^2}} \times \frac{(F_j^{(k)}(n) - \overline{F}_j(n))}{\sqrt{\Sigma_{n=1}^{N}\Sigma_{k=1}^{K}(F_j^{(k)}(n) - \overline{F}_j(n))^2}}. \quad (25)$$

There are $C_8^2 = 28$ correlations in total between different networks in A_{10}. For the simultaneous training with the correlation penalty terms, 21 correlations among them had negative values. The results suggest that the correlation penalty terms tend to lead to negatively correlated individual networks. In contrast, for the independent training without the correlation penalty terms, all the correlations were positive. Because every individual network learns the same task in the independent training, the correlations among them are generally positive. In CELS, each individual network learns different parts or aspects of the training data so that the problem of correlated errors can be removed or alleviated.

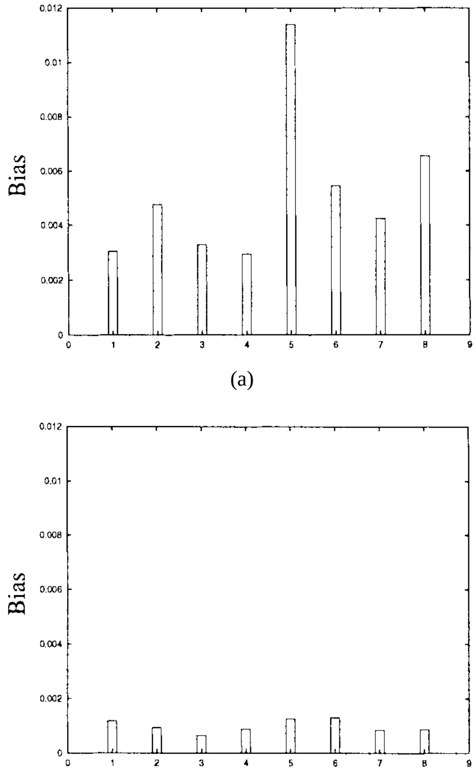
Fig. 3 shows the squared bias of the individual networks in A_{10} using CELS and independent training. For the individual networks created by CELS, they were biased whose values were much larger than those of the individual networks created by independent training. Although the individual networks in CELS were biased, the ensemble was unbiased at the end of training.



(a)

(b)

Fig. 2. (a) Correlations among the individual networks in A_{10} using CELS $(\lambda = 0.5)$. (b) Correlations among the individual networks in A_{10} using independent training (i.e., $\lambda = 0.0$ in CELS). A correlation is represented by a vertical bar with one end at zero and the other end point at the value of the represented correlation. Different bars correspond to correlations between different individuals.



(a)

(b)

Fig. 3. (a) Squared biases of the individual networks in A_{10} using CELS $(\lambda = 0.5)$. (b) Squared biases of the individual networks in A_{10} using independent training (i.e., $\lambda = 0.0$ in CELS). The vertical axis is the value of the estimated bias of an individual network and different bars correspond to different individuals.

C. Experiment 3

This experiment investigated the effects of adding the noise to the target function. Moderate noise (variance $\sigma^2 = 0.1$) and large noise (variance $\sigma^2 = 0.2$) conditions were studied. The architecture of the ensemble was composed of eight individual networks. Each network had five hidden nodes. The learning-rate η in BP was set to 0.1.

Tables III and IV compare the performance of CELS for different strength parameters in both the moderate noise condition and the large noise condition. The results show that there were similar trends for E_{bias}, E_{var}, and E_{cov} in both the noise free condition and the noise conditions. That is, E_{var} seemed to increase as the value of λ increased, and E_{cov} seemed to decrease as the value of λ increased. E_{mse} appeared to decrease first and then increase with increasing value of λ. However, in the large noise condition, E_{bias} appeared to decrease with increasing value of λ. In the moderate noise condition, E_{bias} appeared to decrease first and then increase with increasing value of λ. As demonstrated by the results, CELS with $0 < \lambda \leq 0.75$ outperformed independent training in terms of the integrated MSE values.

Choosing a proper value of λ is important in CELS, and also problem dependent. For the noise conditions used for this regression task and the ensemble architecture used, the bias-variance-covariance tradeoff was optimal for $\lambda = 0.5$ among the tested values of λ in the sense of minimizing the MSE on the testing set. Tables V and VI compare the results of CELS with $\lambda = 0.5$ with those produced by ME architectures [7] and by Rosen's algorithm [6]. For the moderate noise case, the integrated MSE of ME architectures was about 0.018, while the integrated MSE of CELS was 0.012. The integrated MSE achieved by ME architectures was about 0.038 for the large noise case, while the integrated MSE for CELS was 0.023. Although both ME architectures and CELS tend to

TABLE III

RESULTS OF CELS WITH DIFFERENT λ VALUES IN THE MODERATE NOISE CONDITION (VARIANCE $\sigma^2 = 0.1$). THE VALUES OF THE INTEGRATED BIAS, INTEGRATED VARIANCE, INTEGRATED COVARIANCE, AND INTEGRATED MSE ON THE TESTING SET ARE AT EPOCH 2000

	$\lambda = 0.0$	$\lambda = 0.25$	$\lambda = 0.5$
E_{bias}	0.004595	0.004048	0.003766
E_{var}	0.001751	0.002002	0.002653
E_{cov}	0.007351	0.006752	0.005935
$E_{var} + E_{cov}$	0.009102	0.008754	0.008588
E_{mse}	0.013698	0.012802	0.012353
	$\lambda = 0.75$	$\lambda = 0.875$	$\lambda = 1.0$
E_{bias}	0.002934	0.002214	0.002287
E_{var}	0.004589	0.008262	0.210839
E_{cov}	0.005114	0.003342	−0.184092
$E_{var} + E_{cov}$	0.009703	0.011604	0.026747
E_{mse}	0.012637	0.013818	0.029033

TABLE IV
RESULTS OF CELS WITH DIFFERENT λ VALUES IN THE LARGE NOISE CONDITION (VARIANCE $\sigma^2 = 0.2$). THE VALUES OF THE INTEGRATED BIAS, INTEGRATED VARIANCE, INTEGRATED COVARIANCE, AND INTEGRATED MSE ON THE TESTING SET ARE AT EPOCH 2000

	$\lambda = 0.0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 0.875$	$\lambda = 1.0$
E_{bias}	0.008261	0.007414	0.00609	0.005106	0.004686	0.004128
E_{var}	0.003103	0.003533	0.00479	0.008712	0.016792	0.326957
E_{cov}	0.013517	0.012610	0.01196	0.010866	0.008491	−0.267828
$E_{var} + E_{cov}$	0.016620	0.016143	0.01675	0.019578	0.025283	0.059129
E_{mse}	0.024880	0.023557	0.02284	0.024684	0.029969	0.063257

TABLE V
COMPARISON AMONG CELS, ME ARCHITECTURES [7], AND ROSEN'S ALGORITHM [6] IN THE MODERATE NOISE CONDITION

Method	E_{bias}	E_{var}	E_{cov}	E_{mse}
CELS	0.004	0.003	0.006	0.012
ME	0.008	0.030	−0.020	0.018
Rosen's algorithm	0.005	0.005	0.004	0.014

TABLE VI
COMPARISON AMONG CELS, ME ARCHITECTURES [7], AND ROSEN'S ALGORITHM [6] IN THE LARGE NOISE CONDITION

Method	E_{bias}	E_{var}	E_{cov}	E_{mse}
CELS	0.006	0.005	0.012	0.023
ME	0.013	0.065	−0.040	0.038
Rosen's algorithm	0.010	0.008	0.009	0.028

create negatively correlated networks, CELS can achieve good performance by controlling bias, variance, and covariance through the choice of λ value.

CELS's results are also better than those produced by Rosen's algorithm [6]. In Rosen's algorithm, the error function for an individual network j is

$$E_j = \sum_{n=1}^{N} [(d(n) - F_j(n))^2 + \sum_{i=1}^{j-1} \lambda(t) c(i,j) P(\boldsymbol{x}(n), d(n), F_i, F_j)] \quad (26)$$

where $\lambda(t)$ is a (possibly) time-dependent scaling function, $c(i,j)$ is an indicator function for decorrelation between networks i and j, and P is a correlation penalty function. The correlation penalty function P used in [6] is the product of the jth and ith network error

$$P(\boldsymbol{x}(n), d(n), F_i, F_j) = (d(n) - F_i(n))(d(n) - F_j(n)). \quad (27)$$

One indicator function used in [6] is to penalize an individual network for being correlated with the previously trained network

$$c(i,j) = \begin{cases} 1, & \text{if } i = j-1 \\ 0, & \text{otherwise.} \end{cases} \quad (28)$$

There are two essential differences between CELS and Rosen's algorithm. First, in CELS, each network was simultaneously trained to negatively correlate with the rest of networks in the ensemble. In Rosen's algorithm, each network in the ensemble was sequentially trained to decorrelate with the previously trained networks. Second, CELS used a different correlation penalty function for noisy data. These two differences led to different performance between CELS and Rosen's algorithm. Tables V and VI summarize the comparison results. The $\lambda(t)$ in (26) was set to the constant 1. We also tested Rosen's algorithm with different λ. There was not much improvement in the performance.

V. EXPERIMENTAL STUDIES

A. The Mackey–Glass Chaotic Time Series Prediction Problem

This section describes CELS's application to a time series prediction problem. The Mackey–Glass time series investigated here is generated by the following differential equation

$$\dot{x}(t) = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x^{10}(t-\tau)} \quad (29)$$

where $\alpha = 0.2$, $\beta = -0.1$, $\tau = 17$ [13], [14]. As mentioned by Martinetz *et al.* [15], $x(t)$ is quasiperiodic and chaotic with a fractal attractor dimension 2.1 for the above parameters.

1) Experimental Setup: The input consists of four past data points, $x(t)$, $x(t-6)$, $x(t-12)$, and $x(t-18)$. The output is $x(t+6)$. In order to make multiple step prediction (i.e., $\Delta t = 90$) during testing, iterative predictions of $x(t+6)$, $x(t+12), \cdots, x(t+90)$ will be made. During training, the true value of $x(t+6)$ is used as the target value. Such experimental setup is the same as that used by Martinetz *et al.* [15].

In the following experiments, the data for the Mackey–Glass time series was obtained by applying the fourth-order Runge–Kutta method to (29) with initial condition $x(0) = 1.2$, $x(t-\tau) = 0$ for $0 \le t < \tau$, and the time step is 1. The training set consisted of points 118 to 617 (i.e., 500 training patterns). The following 500 data points (starting from point 618) were used as testing set. The values of training and testing data were rescaled linearly to between 0.1 and 0.9. Such experimental setup was adopted in order to facilitate comparison with other existing work.

The normalized root-mean-square (RMS) error E was used to evaluate the performance of CELS, which is determined by the RMS value of the absolute prediction error for Δt, divided by the standard deviation of $x(t)$ [13], [15]

$$E = \frac{\langle [x_{\text{pred}}(t, \Delta t) - x(t + \Delta t)]^2 \rangle^{1/2}}{\langle (x - \langle x \rangle)^2 \rangle^{1/2}} \quad (30)$$

where $x_{\text{pred}}(t, \Delta t)$ is the prediction of $x(t + \Delta t)$ from the current state $x(t)$ and $\langle x \rangle$ represents the expectation of x. As indicated by Farmer and Sidorowich [13], "If $E = 0$, the predictions are perfect; $E = 1$ indicates that the performance is no better than a constant predictor $x_{\text{pred}}(t, \Delta t) = \langle x \rangle$."

TABLE VII
AVERAGE RESULTS PRODUCED BY CELS OVER 25 RUNS FOR THE MACKEY-GLASS TIME-SERIES PREDICTION PROBLEM. THE "TESTING RMS" IN THE TABLE REFERS TO THE ERROR DEFINED BY (30) ON THE TESTING SET. *MEAN, SD, MIN,* AND *MAX* INDICATE THE MEAN VALUE, STANDARD DEVIATION, MINIMUM AND MAXIMUM VALUE, RESPECTIVELY

	Testing RMS			
	Mean	SD	Min	Max
$\Delta t = 6$	0.0100	0.0006	0.0090	0.0116
$\Delta t = 84$	0.0326	0.0028	0.0279	0.0385
$\Delta t = 90$	0.0368	0.0032	0.0305	0.0436

TABLE VIII
GENERALIZATION RESULTS COMPARISON AMONG CELS, EPNET [16], BP, AND CC LEARNING [17] FOR THE MACKEY–GLASS TIME-SERIES PREDICTION PROBLEM. THE "TESTING RMS" IN THE TABLE REFERS TO THE ERROR DEFINED BY (30) ON THE TESTING SET

Method	Testing RMS	
	$\Delta t = 6$	$\Delta t = 84$
CELS	0.01	0.03
EPNet	0.02	0.06
BP	0.02	0.05
CC Learning	0.06	0.32

The ensemble architecture used in the experiments was composed of 20 individual networks. Each individual network was a multilayer perceptron with one hidden layer. Both hidden node function and output node function were defined by the logistic function in (17). All the individual networks had 6 hidden nodes. The number of training epochs was set to 10 000. The learning rate η in BP and strength parameter λ were set to 0.25 and 1.0, respectively. The correlation penalty term given in (10) was used in this experiment.

2) Experimental Results and Comparisons: Table VII shows the average results of CELS over 25 runs. Each run of CELS was from different initial weights. Table VIII compares CELS's results with those produced by EPNet [16], BP and the cascade-correlation (CC) learning [17]. It is obvious that CELS was able to achieve the generalization performance better than that of others.

For a large time span $\Delta t = 90$, CELS's results also compared favorably with those produced by Martinetz *et al.* [15] which had been shown to be better than Moody and Darken [18]. For the same training set size of 500 data points, the smallest prediction error achieved by "neural-gas" networks [15] was about 0.06. The smallest prediction error among 25 CELS runs was 0.003 05, while the average prediction error was 0.0368.

B. The Classification Problems

This section describes CELS's application to a real-world problem, i.e., the Australian credit card assessment problem. The problem is to assess applications for credit cards based on a number of attributes [20]. There are 690 cases in total. The output has two classes. The 14 attributes include six numeric values and eight discrete ones, the latter having from 2–14 possible values. The Australian credit card assessment problem is a classification problem which is different from the previous regression tasks whose outputs are continuous. It is used to illustrate that CELS is applicable to a wide range of problems since it does not assume any *a priori* knowledge of the problem domain. The data set was obtained from the UCI machine learning benchmark repository. It is available by anonymous ftp at ics.uci.edu (128.195.1.1) in directory /pub/machine-learning-databases.

TABLE IX
COMPARISON OF ERROR RATES BETWEEN CELS ($\lambda = 1.0$) AND INDEPENDENT TRAINING USING BP (i.e., $\lambda = 0.0$ IN CELS) ON THE AUSTRALIAN CREDIT CARD ASSESSMENT PROBLEM. THE RESULTS WERE AVERAGED OVER 25 RUNS. "SIMPLE AVERAGING" AND "WINNER-TAKE-ALL" INDICATE TWO DIFFERENT COMBINATION METHODS USED IN CELS

		Simple Averaging		Winner-Take-All
		Training	Test	Test
$\lambda = 1.0$	Mean	0.0938	0.1337	0.1195
	SD	0.0031	0.0068	0.0052
	Min	0.0869	0.1163	0.1105
	Max	0.0985	0.1454	0.1279
$\lambda = 0.0$	Mean	0.0883	0.1386	0.1384
	SD	0.0308	0.0048	0.0049
	Min	0.0792	0.1279	0.1279
	Max	0.0965	0.1454	0.1512

1) Experimental Setup: The data set was partitioned into two sets: a training set and a testing set. The first 518 examples were used for the training set, and the remaining 172 examples for the testing set. The testing set was not seen by any neural network during the training phase. It was only used for testing the generalization of neural network ensembles after they were trained.

The input attributes were rescaled to between 0.0 and 1.0 by a linear function. The output attributes of all the problems were encoded using a 1-of-m output representation for m classes. The output with the highest activation designated the class.

The ensemble architecture used in the experiments consisted of four multilayer perceptrons with one hidden layer. All the individual networks had ten hidden nodes in the hidden layer. Both hidden node function and output node function were defined by the logistic function in (17). The number of training epochs was set to 250. The learning rate η and strength parameter λ were set to 0.1 and 1.0, respectively. These parameters were chosen after limited preliminary experiments. They are not meant to be optimal. The correlation penalty term given in (11) was used in CELS.

2) Experimental Results: Table IX shows CELS's results over 25 runs. Each run of CELS was from different initial weights. The same architecture with the same initial weight setup was also independently trained using BP without the correlation penalty terms (i.e., $\lambda = 0.0$ in CELS). Results are also shown in Table IX. For classification problems, the same combination method used in the regression task, i.e., the simple averaging defined in (6), was first applied to decide the output of the ensemble system. For the simple averaging, it was surprising that the results of CELS with $\lambda = 1.0$ were similar to those of independent training. This phenomenon seems contradictory to the claim that the effect of the correlation penalty term is to encourage different individual networks in an ensemble to learn different parts or aspects of

TABLE X
SIZES OF THE CORRECT RESPONSE SETS OF INDIVIDUAL NETWORKS CREATED RESPECTIVELY BY CELS ($\lambda = 1.0$) AND INDEPENDENT TRAINING (i.e., $\lambda = 0.0$ IN CELS) ON THE TESTING SET AND THE SIZES OF THEIR INTERSECTIONS FOR THE AUSTRALIAN CREDIT CARD ASSESSMENT PROBLEM. THE RESULTS WERE OBTAINED FROM THE FIRST RUN AMONG THE 25 RUNS

$\lambda = 1.0$	$\Omega_1 = 147$	$\Omega_2 = 143$	$\Omega_3 = 138$
	$\Omega_4 = 143$	$\Omega_{12} = 138$	$\Omega_{13} = 124$
	$\Omega_{14} = 141$	$\Omega_{23} = 116$	$\Omega_{24} = 133$
	$\Omega_{34} = 123$	$\Omega_{123} = 115$	$\Omega_{124} = 133$
	$\Omega_{134} = 121$	$\Omega_{234} = 113$	$\Omega_{1234} = 113$
$\lambda = 0.0$	$\Omega_1 = 149$	$\Omega_2 = 147$	$\Omega_3 = 148$
	$\Omega_4 = 148$	$\Omega_{12} = 147$	$\Omega_{13} = 147$
	$\Omega_{14} = 147$	$\Omega_{23} = 147$	$\Omega_{24} = 146$
	$\Omega_{34} = 146$	$\Omega_{123} = 147$	$\Omega_{124} = 146$
	$\Omega_{134} = 146$	$\Omega_{234} = 146$	$\Omega_{1234} = 146$

the training data. In order to verify and quantify this claim, we compared the outputs of the individual networks trained with the correlation penalty term to those of the individual networks trained without the correlation penalty term.

Two notions were introduced to analyze CELS. They are the correct response sets of individual networks and their intersections. The correct response set S_i of individual network i on the testing set consists of all the patterns in the testing set which are classified correctly by the individual network i. Let Ω_i denote the size of set S_i, and $\Omega_{i_1 i_2 \cdots i_k}$ denote the size of set $S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_k}$. Table X shows the sizes of the correct response sets of individual networks and their intersections on the testing set, where the individual networks were, respectively, created by CELS and independent training. It is evident from Table X that different individual networks created by CELS were able to specialize to different parts of the testing set. For instance, in Table X the sizes of both correct response sets S_2 and S_4 at $\lambda = 1.0$ were 143, but the size of their intersection $S_2 \cap S_4$ was 133. The size of $S_1 \cap S_2 \cap S_3 \cap S_4$ was only 113. In contrast, the individual networks in the ensemble created by independent training using BP were quite similar. The sizes of correct response sets S_1, S_2, S_3, and S_4 at $\lambda = 0.0$ were from 146–149, while the size of their intersection set $S_1 \cap S_2 \cap S_3 \cap S_4$ reached 146. There were only three different patterns correctly classified by the four individual networks in the ensemble.

In simple averaging, all the individual networks have the same combination weights and are treated equally. However, not all the networks are equally important. Because different individual networks created by CELS 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. In this experiment, a winner-take-all method was applied to select such networks. For each pattern of the testing set, the output of the ensemble was only decided by the network whose output had the highest activation. Table IX shows the average results of CELS over 25 runs using the winner-take-all combination method. The winner-take-all combination method improved CELS significantly because there were good and poor networks for each case in the testing set and winner-take-all selected the best one. However it did not improved the independent training much because the individual networks created by the independent training were all similar to each other.

TABLE XI
COMPARISON AMONG CELS, EPNet [16], AN EVOLUTIONARY ENSEMBLE LEARNINIG ALGORITHM (Envo-En-RLS) [19], AND OTHERS [20] IN TERMS OF THE AVERAGE TESTING ERROR RATE FOR THE AUSTRALIAN CREDIT CARD ASSESSMENT PROBLEM. TER STANDS FOR TESTING ERROR RATE IN THE TABLE

Algorithm	TER	Algorithm	TER
CELS	0.120	DIPOL92	0.141
EPNet	0.115	Discrim	0.141
Evo-En-RLS	0.095	Logdisc	0.141
Cal5	0.131	CART	0.145
ITrule	0.137	RBF	0.145
CASTLE	0.148	NaiveBay	0.151
IndCART	0.152	BP	0.154

Table XI compares CELS's results with those produced by other neural and nonneural algorithms, where EPNet is an evolutionary system for designing neural networks [16] and Evo-En-RLS forms the final results by combining all the individuals in the last generation in EPNet based on the recursive least-square algorithm [19]. The other algorithms represent the best 11 out of 23 algorithms tested in [20]. Although CELS performed slightly worse than EPNet and Evo-En-RLS, it was significantly faster in terms of training time. CELS performed better than all other algorithms although they used ten-fold cross-validation.

VI. CONCLUSIONS

This paper describes a new approach to designing neural network ensembles for both regression and classification problems with noise. The approach can be regarded as one way of decomposing a large problem into smaller and specialized ones, so that each subproblem can be dealt with by an individual neural network relatively easily. A correlation penalty term in the error function was proposed to encourage the formation of specialists in the ensemble. The Mackey–Glass time series prediction problem and the Australian credit card assessment problem were used as examples to demonstrate the effectiveness of this approach.

This paper has also analyzed CELS in terms of the bias-variance-covariance tradeoff in both the noise free condition and the noise conditions. Unlike other ensemble approaches which try to create unbiased individual networks whose errors are uncorrelated, CELS can produce biased individual networks whose errors tend to be negatively correlated. Very competitive results have been produced by CELS in comparison with independent training, Rosen's algorithm [6] and ME architectures [7].

There are, however, some issues that need resolving. The architectures of individual neural networks and the size of ensemble are predefined at the moment. It would be desirable to develop a learning algorithm which can vary the ensemble architectures dynamically.

ACKNOWLEDGMENT

The authors are grateful to anonymous referees for their constructive comments which have helped to improve the paper.

REFERENCES

[1] A. J. C. Sharkey, "On combining artificial neural nets," *Connect. Sci.*, vol. 8, pp. 299–313, 1996.

[2] X. Yao and Y. Liu, "Ensemble structure of evolutionary artificial neural networks," in *Proc. 1996 IEEE Int. Conf. Evolutionary Computation (ICEC'96)*, 1996, pp. 659–664.

[3] R. T. Clemen and R. L. Winkler, "Limits for the precision and value of information from dependent sources," *Oper. Res.*, vol. 33, pp. 427–442, 1985.

[4] M. Perrone and L. N. Cooper, "When networks disagree: ensemble methods for hybrid neural networks," in *Neural Networks for Speech and Image Processing*, R. J. Mammone, Ed. London, U.K.: Chapman & Hall, 1993.

[5] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," *Neural Computat.*, vol. 6, pp. 1289–1301, 1994.

[6] B. E. Rosen, "Ensemble learning using decorrelated neural networks," *Connect. Sci.*, vol. 8, pp. 373–383, 1996.

[7] R. A. Jacobs, "Bias/variance analyzes of mixture-of-experts architectures," *Neural Computat.*, vol. 9, pp. 369–383, 1997.

[8] H. White, "Learning in artificial neural networks: A statistical perspective," *Neural Computat.*, vol. 1, pp. 425–464, 1989.

[9] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994, pp. 151–152.

[10] S. Geman, E. Bienenstock and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computat.*, vol. 4, pp. 1–58, 1992.

[11] R. Meir, "Bias, variance, and the combination of least squares estimators," in *Advances in Neural Information Processing System 7 (NIPS-7)*, G. Tesauro, D. Touretzky and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 295–302.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[13] J. D. Farmer and J. J. Sidorowich, "Predicting chaotic time series," *Phys. Rev. Lett.*, vol. 59, pp. 845–847, 1987.

[14] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, p. 287, 1977.

[15] T. M. Martinetz and S. G. Berkovich, and K. J. Schulten, "'Neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Networks*, vol. 4, pp. 558–569, 1993.

[16] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Networks*, vol. 8, pp. 694–713, 1997.

[17] R. S. Crowder, "Predicting the Mackey–Glass timeseries with cascade-correlation learning," in *Proc. 1990 Connectionist Models Summer School*, 1990, pp. 117–123.

[18] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computat.*, vol. 1, pp. 281–294, 1989.

[19] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 417–425, June 1998.

[20] D. Michie and D. J. Spiegelhalter and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. London, U.K.: Ellis Horwood, 1994.

Yong Liu (S'97) received the B.Sc. degree from Wuhan University, China, in 1988 and the M.Sc. degree from Huazhong University of Science and Technology in 1991. He received Ph.D. degrees from Wuhan University in 1994 and the University of New South Wales, Australia, in 1999.

He is currently a Research Fellow with the Evolvable Systems Laboratory, Computer Scinece Division, Electrotechnical Laboratory, Ibaraki, Japan. From 1994 to 1996, he was a Lecturer at Wuhan University. Between the end of 1994 to 1995, he was a Visiting Fellow at the School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy. He has published a number of papers in international journals and conferences, and is the first author of the book *Genetic Algorithms* (Beijing, China: Science Press, 1995). His research interests include neural networks, evolutionary algorithms, parallel computing, and optimization.

Dr. Liu received the Third Place Award in the 1997 IEEE Region 10 Postgraduate Student Paper Competition.

Xin Yao (M'91–SM'96) received the B.Sc. degree from the University of Science and Technology of China (USTC) in 1982, the M.Sc. degree from the North China Institute of Computing Technologies (NCI) in 1985, and the Ph.D. degree from USTC in 1990.

He is currently a Professor with the School of Computer Science, University of Birmingham, Birminham, U.K. He was an Associate Professor, School of Computer Science, University College, the University of New South Wales, Australian Defence Force Academy (ADFA). He held postdoctoral fellowships in the Australian National University (ANU) and the Commonwealth Scientific and Industrial Research Organization (CSIRO) before joining ADFA in 1992. He is an Associate Editor for *Knowledge and Information Systems* and a member of the editorial board of *Journal of Cognitive Systems Research*.

Dr. Yao was the Program Committee Cochair for IEEE ICEC'97, Indianapolis, IN, and CEC'99, Washington, DC, and Co-Vice-Vhair for IEEE ICEC'98, Anchorage, AK. He was Program Committee Cochair for SEAL'96, SEAL'98, and ICCIMA'99. He is an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, a member of the IEEE NNC Technical Committee on Evolutionary Computation, and the second vice-president of the Evolutionary Programming Society.