Bagging and Boosting Negatively Correlated Neural Networks

Bagging and Boosting Negatively Correlated Neural Networks

Md. Monirul Islam, Xin Yao, *Fellow, IEEE*, S. M. Shahriar Nirjon, Muhammad Asiful Islam, and Kazuyuki Murase

***Abstract*—In this paper, we propose two cooperative ensemble learning algorithms, i.e., NegBagg and NegBoost, for designing neural network (NN) ensembles. The proposed algorithms incrementally train different individual NNs in an ensemble using the negative correlation learning algorithm. Bagging and boosting algorithms are used in NegBagg and NegBoost, respectively, to create different training sets for different NNs in the ensemble. The idea behind using negative correlation learning in conjunction with the bagging/boosting algorithm is to facilitate interaction and cooperation among NNs during their training. Both NegBagg and NegBoost use a constructive approach to automatically determine the number of hidden neurons for NNs. NegBoost also uses the constructive approach to automatically determine the number of NNs for the ensemble. The two algorithms have been tested on a number of benchmark problems in machine learning and NNs, including Australian credit card assessment, breast cancer, diabetes, glass, heart disease, letter recognition, satellite, soybean, and waveform problems. The experimental results show that NegBagg and NegBoost require a small number of training epochs to produce compact NN ensembles with good generalization.**

***Index Terms*—Bagging, boosting, constructive approach, diversity, generalization, negative correlation learning, neural network (NN) ensemble design.**

I. Introduction

THE GENERALIZATION ability of neural networks (NNs) is an important property for their practical applications to many real-world problems. This generalization ability can significantly be improved through NN ensembles, i.e., training several individual NNs and combining their outputs [1]–[3]. An ensemble approach has two major components, i.e., a method for creating individual NNs and a method for combining NNs. Theoretical and experimental studies have shown that when the NNs in the ensemble are accurate and their errors are negatively correlated, an improved generalization ability can be obtained by voting or averaging the outputs of NNs [4]–[11]. There is little to be gained by combining NNs whose errors are positively correlated.

There are a number of alternative approaches that can be used for producing negatively correlated NNs for the ensemble. These include varying the initial random weights of NNs, varying the topology of NNs, varying the algorithm employed for training NNs, and varying the training sets of NNs. It is argued that training NNs using different training sets is likely to produce more uncorrelated errors than other approaches [12]. This is because it is the training data on which a network is trained that determine the function it approximates. Data sampling or variation of examples in training data is a common and effective technique for producing negatively correlated NNs.

The two most popular algorithms for constructing ensembles that independently and sequentially train individual NNs, respectively, using different training sets are the bagging [13] and boosting [14] algorithms. Recently, Liu and Yao [15]–[17] proposed negative correlation learning that simultaneously trains NNs in the ensemble. While bagging and boosting explicitly create different training sets for different NNs by probabilistically changing the distribution of the original training data, negative correlation learning implicitly creates different training sets by encouraging different NNs to learn different parts or aspects of the training data.

As explicit and implicit approaches for creating training sets have complementary strengths and limitations (as described in Section II), the question arises as to whether their integration could lead to an improved and more powerful ensemble learning scheme. We studied this question and propose two new ensemble learning algorithms, i.e., NegBagg and NegBoost, for designing NN ensembles. The NegBagg and NegBoost algorithms incorporate negative correlation learning in bagging and boosting, respectively, for incrementally training NNs. The reason for using negative correlation learning in conjunction with bagging and boosting algorithms is to improve the interaction and cooperation among NNs in ensembles. This approach is quite different from others, which independently [13], sequentially [14], [18], or simultaneously [15]–[17] train NNs in ensembles by explicitly or implicitly dividing the training data. It also differs from these other approaches in its automatic construction of the ensemble architecture.

In [19], an ensemble learning algorithm is proposed, which sequentially trains NNs in the ensemble. The algorithm attempts not only to minimize the NN error but also to decorrelate the error from previously trained NNs. Since this algorithm does not provide any mechanism for interaction among NNs, sequentially training an NN cannot affect the previously trained

Manuscript received August 17, 2007; revised February 11, 2008. The work of M. M. Islam was supported by the Japanese Society for the Promotion of Sciences (JSPS). The work of X. Yao was supported in part by the Engineering and Physical Sciences Research Council (U.K.) under Grant GR/T10671/01 and by the Fund for Foreign Scholars in University Research and Teaching Programs (China) under Grant B07033. This paper was recommended by Associate Editor N. Chawla.

M. M. Islam is with the Bangladesh University of Engineering and Technology (BUET), Dhaka 1000, Bangladesh, and also with the University of Fukui, Fukui 910-8507, Japan.

X. Yao is with the University of Birmingham, B15 2TT Birmingham, U.K., and also with the University of Science and Technology of China, Hefei 230026, China.

S. M. Shahriar Nirjon is with the Bangladesh University of Engineering and Technology (BUET), Dhaka 1000, Bangladesh.

M. A. Islam is with the State University of New York at Stony Brook, Stony Brook, NY 11794-4400 USA.

K. Murase is with the University of Fukui, Fukui 910-8507, Japan.

Digital Object Identifier 10.1109/TSMCB.2008.922055

NNs. The errors of the NNs are, therefore, not necessarily negatively correlated [16]. Our algorithms, i.e., NegBagg and NegBoost, incrementally train each NN in the ensemble using a different training set and negative correlation learning. The negatively correlated NNs can easily be obtained by using NegBagg and NegBoost. Such a training strategy has produced a significant improvement in the NN ensemble's performance, as shown by our study.

The NegBagg and NegBoost algorithms are different from the mixtures-of-experts (ME) architecture [20]–[22], which consists of a gating network and a number of expert networks. Although the ME architecture can also produce biased NNs whose estimates are negatively correlated, it uses a gating network. The NegBagg and NegBoost algorithms do not require such a gating network. They use different training sets and negative correlation learning to ensure negative correlation among NNs in an ensemble. The advantage of this approach is that the λ parameter of negative correlation learning provides a convenient way to balance the bias–variance–covariance tradeoff. The ME architecture does not provide such control over this tradeoff.

Recently, a constructive algorithm for training cooperative NN ensembles (CNNEs) has been proposed [23]. The algorithm automatically determines the ensemble architecture and uses incremental training based on negative correlation learning for training NNs. However, it does not use different data for training NNs. One major disadvantage of such an approach is that CNNE solely relies on negative correlation learning for producing negatively correlated NNs. Since all the NNs in an ensemble are trained using the same training data, there will be competitions during the training of NNs [23]. We extend CNNE by incrementally training individual NNs using different training sets that are created using bagging and boosting algorithms. Such an extension has produced a significant performance gain for the NN ensembles.

The rest of this paper is organized as follows. Section II discusses various methods to create NNs for ensembles and explains why we combine negative correlation learning with bagging and boosting algorithms. Section III describes our NegBagg and NegBoost algorithms in detail. Section IV presents the results of our experimental study. Finally, Section V concludes this paper with a brief summary and a few remarks.

II. Previous Work

The idea of designing an ensemble learning system can be traced back to as early as 1958 [24], [25]. Since the early 1990s, algorithms based on similar ideas have been developed in many different but related forms such as NN ensembles and ME. An NN ensemble approach can be viewed as comprised of two different methods, i.e., a method for creating individual NNs and a method for combining the outputs of NNs. One important feature in creating NNs is that the networks should be accurate yet diverse, i.e., produce uncorrelated errors. The accuracy of NNs is dependent on their architectures, whereas the diversity is dependent on the error correlation among NNs. This section briefly describes some work toward creating NNs for an ensemble.

The most prevalent methods for creating NNs for an ensemble are the bagging and boosting algorithms. Bagging [13] creates M NNs for the ensemble by independently training these M NNs on M different training sets, which are generated by forming bootstrap replicates of the original training data. This approach is particularly attractive when there is not much training data available. Another advantage of bagging is that all individual NNs in the ensemble can independently be trained in parallel. The boosting algorithm was proposed by Schapire [14] and improved by Freund and Schapire [26]. This algorithm generates a series of NNs whose training sets are determined by the performance of the former ones. Training examples that were wrongly predicted by former NNs will play more important roles in the training of later NNs. The main advantage of boosting is that the number of NNs in the ensemble can easily be automatically determined because the NNs are trained one after another. However, the main problem of both bagging and boosting is that the NNs created by them are not necessarily negatively correlated [16], [17].

The negative correlation learning algorithm proposed by Liu and Yao [15]–[17] encourages different NNs in an ensemble to learn different parts or aspects of the training data, so that the ensemble can efficiently learn the entire training data. The algorithm trains NNs simultaneously and interactively rather than independently or sequentially by introducing a correlation penalty term into the error function of NNs. The advantage of negative correlation learning is that the algorithm can produce biased NNs whose errors tend to be negatively correlated. However, NNs may engage in competition during their training because all NNs are concerned with the same remaining ensemble error in the training scheme of negative correlation learning [23]. There is no chance for a such competition in a bagging or boosting algorithm because the algorithm independently trains each NN in the ensemble on a separate training set.

There are also many other approaches to create NNs for ensembles. Raviv and Intrator [27] present a method for creating NNs that uses a combination of bootstrap sampling of data, variable amounts of noise to inputs, and weight decay. Cherkauer [28] creates NNs with different numbers of hidden nodes. Maclin and Shavlik [29] initialize NNs at different points in the weight space. Krogh and Vedelsby [6] employ cross validation to create NNs. Opitz and Shavlik [8] exploit the genetic algorithm to train diverse knowledge-based NNs. Oza and Tumer [30] present a method that seeks to reduce the correlations among NNs using different subsets of input features.

Recently, the DECORATE algorithm [31] was proposed to create diverse NNs for an ensemble in a simple and straightforward manner. The algorithm creates different training sets for different NNs in the ensemble by adding different artificially produced examples to the training sets. Granitto *et al.* [32] proposed a stepwise ensemble construction algorithm that seeks a new NN that should be at least partially anticorrelated with the previous NNs in the ensemble. This is achieved by applying a late-stopping method in the learning phase of NNs, leading to a controlled level of overtraining of the ensemble members. The algorithm retains the simplicity of independent network training, although, if necessary, it can avoid the computational burden by saving intermediate NNs.

Liu [33] uses the bagging algorithm to create different training sets for ensembles. In [33], each training set is used once for training all the NNs in an ensemble using negative correlation learning. The aim of creating separate training sets is to use them for the cross-validation purpose. One basic difference between NegBagg and the work in [33] is that NegBagg uses negative correlation learning to establish training time interaction among NNs in the ensemble trained by different training sets. In addition, NegBagg considers the accuracy and diversity of NNs in the ensemble, whereas the work in [33] considers diversity among NNs. Both accuracy and diversity are important for improving the performance of ensembles. These are also basic differences between NegBagg/NegBoost and most existing algorithms in the literature. More developments in NN ensembles can be found in the survey paper [34].

III. NegBagg and NegBoost

This section describes our NegBagg and NegBoost algorithms in detail. Both algorithms incrementally train individual NNs in an ensemble using different training sets. The algorithms use a constructive approach to automatically determine the number of hidden nodes in NNs. Furthermore, NegBoost also uses the constructive approach to automatically determine the number of individual NNs in the ensemble. To facilitate training time interactions among NNs, the two algorithms use negative correlation learning [16], [17] to train the NNs in the ensemble.

In comparison with bagging, boosting, and negative correlation, the major advantages of the proposed algorithms include the following: 1) the use of a single-stage ensemble learning approach, i.e., the creation and combination of individual NNs are accomplished in the same learning phase; 2) the use of incremental training based on negative correlation learning and different training sets; and 3) maintaining both accuracy and diversity among NNs in ensembles.

Three-layered feedforward networks are used as individual NNs for constructing ensembles in NegBagg and NegBoost. The sigmoid transfer function is employed for nodes in the hidden and output layers of NNs. This is, however, not an inherent constraint. In fact, any type of NNs (e.g., CCA [35], CNNDA [36], and RBF [37]) and transfer functions (e.g., threshold function [37] and Hermite polynomial [38]) can be used in the proposed algorithms.

A. NegBagg Algorithm

The major steps of NegBagg can be summarized as follows.

Step 1) Create an ensemble architecture consisting of M individual NNs. Here, M is a user-specified parameter, and it is a positive integer number. The number of nodes in the input and output layers of NNs is defined by the problem. Since NegBagg uses a constructive approach to determine the architecture of NNs, the hidden layer of NNs initially contains only one node, similar to other constructive approaches (see the review paper [39]). If the initial architecture contains more than one node, it is necessary to know how many nodes will be for the initial architecture, which is problem dependent and difficult to determine in advance. All the connection weights of each NN in the ensemble are initialized at random within a small range.

Step 2) Create M training sets by forming bootstrap replicates of the original training data as used in bagging [13].

Step 3) Partially train M NNs in the ensemble, each with a different set, for a certain number of training epochs using negative correlation learning [16]. The number of training epochs τ is specified by the user. The term partial training was first introduced in [40]. It means that an NN is to be trained for a fixed number of epochs regardless of whether it has converged or not.

Step 4) Compute the ensemble error on the validation set. If the termination criterion is not satisfied, go to the next step. It is assumed here that the NNs in the ensemble are not sufficiently trained or the architectures of NNs are too small. Otherwise, stop the ensemble training and determine the accuracy of the existing ensemble architecture using the testing set of the given problem. According to [23], the ensemble error E is defined as

$$E = 100\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\left(\frac{1}{M}\sum_{i=1}^{M}F_i(n) - d(n)\right)^2 \quad (1)$$

where N is the number of patterns in the validation set, $F_i(n)$ is the output of the network i for the nth example in the validation set, and $d(n)$ is the desired output of the nth validation example.

Step 5) Add one hidden node to any NN in the ensemble if the NN satisfies the node addition criterion, and go to Step 3).

It is now clear that NegBagg can automatically add hidden nodes to NNs during the training process of the ensemble. The algorithm adds a hidden node by splitting an existing node in an NN. The process of a node splitting is called "cell division" by Odri *et al.* [41]. Two nodes created by splitting an existing node have the same number of connections as the existing node. According to [41], the weights of the new nodes are calculated as

$$w_i^1 = (1+\beta)w_i \quad (2)$$

$$w_i^2 = -\beta w_i \quad (3)$$

where w_i represents the ith weight of the existing node, and w_i^1 and w_i^2 are the corresponding weights of the two new nodes. β is a mutation parameter whose value may be a fixed or random one. The advantage of this addition process is that it does not require random initialization of the weight vector of the newly added node. As a result, the new NN can better maintain the behavioral link with its predecessor [40].

Although the design of NN ensembles is generally formulated as a two-stage process (first training individual NNs and

then combining them), NegBagg trains and combines NNs in the same learning process using a simple cost function. The idea behind NegBagg is simple and straightforward. NegBagg minimizes the ensemble error first by training NNs and then by adding hidden nodes to existing NNs. Details about the node addition criterion used in NegBagg will be given in a later section.

B. NegBoost Algorithm

The major steps of NegBoost can be summarized as follows.

Step 1) Initialize an ensemble architecture with one NN. The number of nodes in the input and output layers of the NN is defined by the problem. Initially, the hidden layer contains only one node. All connection weights of the NN are initialized at random within a small range. Label the individual NN with I.

Step 2) Create a new training set for the newly added NN in the ensemble by changing the distribution of examples in the training data as used in boosting [14].

Step 3) Partially train the I-labeled NN in the ensemble on the new training set for a certain number of training epochs using negative correlation learning. The number of training epochs τ is specified by the user.

Step 4) Compute the ensemble error on the validation set. If the termination criterion is not satisfied, it is assumed that the I-labeled NN is not sufficiently trained or the ensemble architecture is too small, and go to the next step. Otherwise, stop the ensemble training and determine the accuracy of the existing ensemble architecture using the testing set of the given problem.

Step 5) Check the criteria for stopping the network construction and node addition. If the I-labeled NN does not satisfy the criterion for stopping network construction or node addition, it is assumed that the NN is not sufficiently trained, and go to Step 3) for further training. Otherwise, go to the next step for the modification of the existing ensemble architecture.

Step 6) Replace the labeling of the I-labeled NN by F if it satisfies the stopping criterion for network construction.

Step 7) If all the individual NNs in the ensemble have been marked with the label F, add one new individual NN to the ensemble. Initialize the new NN in the same way as that described in Step 1) and go to Step 2). Otherwise, go to the next step.

Step 8) Add one hidden node to the I-labeled NN and go to Step 3). The hidden node is also added here, as in NegBagg, by splitting an existing node of the I-labeled NN.

Although NegBoost uses the original boosting algorithm in creating different training sets for NNs in ensembles, it can also use adaboost.M1 [26] or adaboost.M2 [42]. This is because the boosting algorithm is used here only for creating training sets. Similar to NegBagg, NegBoost also trains and combines individual NNs in the same learning process using a simple cost function, i.e., the ensemble error E. However, NegBoost automatically determines not only the number of hidden nodes in NNs but also the number of NNs in the ensemble. The following section gives more details about our algorithms.

1) Criterion for Node Addition: Both NegBagg and NegBoost use the same criterion for deciding when to add hidden nodes to existing NNs in an ensemble. The criterion is based on the contribution of individual NNs to the ensemble. According to [23], the contribution C_m of the mth NN to the ensemble at any training epoch is

$$C_m = 100\left(\frac{1}{E} - \frac{1}{E_m}\right) \tag{4}$$

where E is the ensemble error *including* the network m, and E_m is the ensemble error *excluding* the network m. The errors are computed according to (1). The criterion is very simple and does not require any extra computational cost because E_m is a part of E. Thus, one could extract the value of E_m during the computation of E and save it for future use.

According to the node addition criterion, NegBagg and NegBoost add one node to any network m when its contribution C_m to the ensemble does not improve by a threshold ϵ after a certain number of training epochs τ. The parameters ϵ and τ are specified by the user. The addition criterion is tested in NegBagg and NegBoost for every τ epochs and is described as [23]

$$C_m(t+\tau) - C_m(t) < \epsilon, \qquad t = \tau, 2\tau, 3\tau, \ldots. \tag{5}$$

2) Criterion for Stopping Network Construction: Another simple criterion is used in NegBoost for deciding when to stop the construction of individual NNs. The NegBoost algorithm stops the construction of an I-labeled NN when its contribution C to the ensemble, measured after the addition of each hidden node, failed to improve after the addition of a certain number of hidden nodes n_h. The parameter n_h is specified by the user, and it is a positive integer number. In other words, the construction of the I-labeled NN is stopped when the following is true:

$$C(i+n_h) \leq C(i), \qquad i = 1, 2, \ldots. \tag{6}$$

If $n_h = 0$, then all individual NNs can consist of only one hidden node. In NegBoost, no nodes are added to the NNs after their construction processes have been stopped. The salient feature of using the individual NN's contribution as a criterion for node addition and stopping network construction reflects our emphasis on the *cooperative* training of NNs for designing ensembles. This is because the contribution of any NN in an ensemble will increase only when the NN produces accurate but uncorrelated outputs with respect to other NNs in the ensemble.

As with NegBoost, NegBagg does not stop the construction process of NNs during the training process of the ensembles. Instead, the algorithm rather continuously adds hidden nodes to an NN one after another if the node addition criterion is satisfied and the problem is remain unsolved. The following is the reason for such a continuous addition of hidden nodes in the training scheme of NegBagg. It is clear from Section III-A that NegBagg does not add NNs during the training process of an ensemble. Hence, when the performance of the ensemble

does not improve after training, the algorithm can only try to improve the ensemble performance by increasing its computational power, i.e., by adding hidden nodes to existing NNs.

3) Termination Criterion: It can be seen from Sections III-A and B that both NegBagg and NegBoost use the validation error for stopping ensemble training. Since the two algorithms use a constructive approach in designing ensembles, the training error of the ensembles will gradually decrease as their construction processes progress. However, at some point, usually in the later stages of training, ensembles may start to take advantage of idiosyncrasies in the training data, and their generalization performance, i.e., performance on a testing set, may start to deteriorate although the training error continues to decrease. This type of overfitting due to overtraining is described in [43].

One common approach to avoid overfitting is to estimate the validation error during training and stop the training process when the error begins to increase. The simplest method to achieve this goal is to divide the training data into a training set and a validation set. The training set is used to modify the weights of NNs and the validation set to determine when the training process is to be stopped. In other words, the validation set is used to anticipate the behavior of the testing set, assuming that the error on both validation and testing sets is similar. However, the real situation is a lot more complex because the real generalization curves almost always have more than one local minima. It is shown for linear networks with K inputs and K outputs that up to K such local minima are possible; for multilayer NNs, the situation is even worse [44]. In general, it is impossible to know whether an increase in the validation error indicates real overfitting or is just intermittent.

In NegBagg or NegBoost, a very simple criterion that relies on the sign of the changes in the validation error is used for deciding when the ensemble training is to be stopped. The criterion stops the training process of the ensemble when its validation error increases for N successive times. The idea behind this definition is that when the validation error is increased not just once but during N consecutive times, it can be assumed that such increases indicate the beginning of final overfitting, independent of how large the increases actually are. A similar criterion is described in [45] for automatically stopping the training process of a single NN.

C. NegBagg/NegBoost and Variance Reduction

One of the main explanations for the improvements achieved by ensembles is based on separating the expected error into a bias term and a variance term. Several methods have been suggested for decomposing the error into bias and variance. The bias term basically measures how closely a learning algorithm's average guess (over all the possible training sets of the given training set size) matches the target. The variance term measures how much the learning algorithm's guess fluctuates for the different training sets of the given size.

The bias of an ensemble is simply the average of the biases of individual NNs. The introduction of negative correlation learning in bagging and boosting helps reduce the ensemble bias. When an NN in the ensemble fails to solve some parts or aspects of a given task, other NNs become aware of this situation in the training scheme of negative correlation learning, and the NNs can try to solve the unsolved parts of the task. Eventually, the unsolved parts of the task get solved. If the interaction among NNs in the ensemble were completely missing as in pure bagging [13] or boosting [14], it might be the case that some portion of the task remain unsolved, and the bias term does not significantly reduce with training. However, NegBagg or NegBoost overcomes this problem by facilitating interaction through negative correlation learning [16], [17] and can successfully keep the bias of the error significantly reduced.

The variance $V(x)$ of the ensemble can be defined in terms of the deviations of individual NN predictions $f_k(x)$. According to [7], $V(x)$ can be expressed as

$$V(x) = E_D\left[\left(\sum_k w_k \Delta f_k(x)\right)^2\right] \tag{7}$$

where w_k is the weight for the kth NN, which are positive constants and sum to 1, and $\Delta f_k(x)$ denotes the deviation of $f_k(x)$. The expectation is taken over all training sets D. The ensemble variance in (7) can be related to the variance of the individual NN, which is $E_D[(\Delta f_k)^2]$. Since $V(x)$ cannot be more than the average of the variances of all NNs in the ensemble, we get the inequality denoting the upper bound of $V(x)$, which is

$$V(x) \leq \sum_k w_k E_D\left[(\Delta f_k(x))^2\right]. \tag{8}$$

This bound is saturated when the fluctuations of the networks' outputs are fully correlated and have equal variance [7]. This situation occurs when all NNs in the ensemble are identical, which means each NN in the ensemble reacts very similarly to different training sets. On the other hand, when the fluctuations of NNs are completely uncorrelated, the variance of the ensemble can be expressed as

$$V(x) = \sum_k w_k^2 E_D\left[(\Delta f_k(x))^2\right]. \tag{9}$$

This shows that, when NNs are given equal weights (i.e., $w_k = 1/k$), one can have a reduction of variance by a factor of $O(1/k)$. It is therefore proved that the ensemble variance can be reduced by a large factor when individual NNs are uncorrelated. Now the question is whether the introduction of negative correlation learning in bagging or boosting can reduce the correlation among NNs in the ensemble. The answer is affirmative, and the following few paragraphs explain how negative correlation learning reduces the correlation among NNs.

The bagging algorithm creates separate training sets for the different NNs of an ensemble by forming bootstrap replicates of the original training data T. Let T consist of N training examples. The algorithm constructs training sets by uniformly drawing N examples at random (with replacement) from T. It can easily be shown that there is correlation among NNs although they are trained with separate training sets created by bagging.

Suppose that two new training sets T_i and T_j of size N are created from T using bagging. Let D_i and D_j be the set of distinct examples in T_i and T_j, respectively. Whenever $T_i \bigcap T_j \neq \phi$,

two NNs trained by T_i and T_j will produce correlated outputs for any example $x \in (T_i \bigcap T_j)$. If $(|D_i| + |D_j|) > N$, then surely such x exists. Let $|D_i| = r_i N$ and $|D_j| = r_j N$, where r_i and r_j are positive fractions. We now investigate the probability $\mathrm{P}((|D_i| + |D_j|) <= N)$. By inserting the value of D_i and D_j, and replacing the value of r_i and r_j with their expected value $E[r]$, the probability term becomes $\mathrm{P}(E[r] \leq 1/2)$. For an ensemble consisting of M NNs, the probability term becomes $\mathrm{P}(E[r] \leq 1/M)$. This probability decreases as M increases. It is therefore proved that bagging cannot completely eliminate the correlation among NNs in the ensemble.

The introduction of negative correlation learning in bagging can completely remove the correlation. This is because negative correlation learning facilitates interaction among NNs by introducing a penalty term into the error function of NNs. In negative correlation learning, the error $E_i(n)$ of the ith NN for the nth training example is expressed by the following equation [15]:

$$E_i(n) = \frac{1}{2}\left(F_i(n) - d(n)\right)^2 + \lambda p_i(n) \tag{10}$$

where $F_i(n)$ and $d(n)$ are the actual and desired outputs of the ith NN for the nth training example, respectively. The first term in (10) is the empirical risk function of the ith NN. The second term p_i is the correlation penalty function, which can be expressed as

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

The interaction among NNs of the ensemble is controlled by the value of λ. The penalty function keeps the NNs negatively correlated. Once a training example $x \in T$ is learned by an NN, negative correlation learning inherently forces other NNs in the ensemble to learn from $T - \{x\}$. The NegBagg algorithm, being a hybrid of negative correlation learning and bagging, uses negative correlation learning to keep the NNs in the ensemble negatively correlated. Since bagging is incorporated, each NN in an ensemble of NegBagg has a different training set. The common examples (i.e., $T_i \bigcap T_j \neq \phi$) were a cause for correlation among NNs in bagging. Negative correlation learning works on these common examples and implicitly divides the learning task $(T_i \bigcap T_j)$ among NNs. It thereby removes the correlations that were present even after bagging. The explicit division of task by bagging is therefore complemented by the negative correlation's implicit task division, and hence, the correlation is eliminated. Since the ensemble variance is related to the correlation among NNs, NegBagg reduces the ensemble variance, and it thereby reaches an ensemble variance given by (9). A similar analysis is also applicable for NegBoost.

IV. Experimental Studies

In this section, we evaluate the performance of NegBagg and NegBoost on several classification problems from the University of California, Irvine, Machine Learning Repository and one regression problem used in [17]. Nine classification problems are considered for evaluating the performance. They are the Australian credit card, breast cancer, diabetes, glass, heart disease, letter recognition, satellite image, soybean, and waveform problems. Detailed descriptions of the classification problems are available at ics.uci.edu in directory/pub/machine-learning-databases. The characteristics of the classification problems are summarized in Table I, which shows considerable diversity in the number of examples, attributes, and classes among problems.

TABLE I
Characteristics of Nine Classification Data Sets

Data set	No. of		
	examples	input attributes	output classes
Card	690	14	2
Cancer	699	9	2
Diabetes	768	8	2
Glass	214	9	2
Heart	303	13	2
Letter	20,000	16	26
Satellite	6435	36	7
Soybean	683	35	19
Waveform	5000	40	3

A. Experimental Setup

The data sets representing nine classification problems were preprocessed by rescaling the input attribute values to between 0 and 1. The outputs were encoded by $1 - of - c$ for c classes, where the output node with the highest activation was designated as the network output. The first 50%, the following 25%, and the final 25% examples in all data sets except the letter were used for the training, validation, and testing sets. Such partition follows previous suggestions on benchmarking [46], [47]. For the letter data set, 16 000 and 2000 examples were randomly selected for the training and validation sets, and the remaining 2000 examples in the data set were used for the testing set.

For each individual NN, one bias node with a fixed input 1 was used for the hidden and output layers. The logistic sigmoid function $f(y) = 1/(1 + exp(-y))$ was used as an activation function for all nodes in the hidden and output layers. The initial connection weights for individual NNs in an ensemble were randomly chosen in the range of -0.3 to 0.3. The learning rate and momentum of backpropagation were chosen in the range of 0.10–0.50 and 0.5–0.9, respectively. The values used for τ and n_h were 10 and 2, respectively. The threshold value ϵ was chosen between 0.10 and 0.25. The value of M, i.e., the number of individual NNs in NegBagg, was chosen between 5 and 10 for all problems except for the letter, where it was chosen between 10 and 15. The value of N used in the termination criterion to stop the ensemble construction process was set to 2. The same experimental settings were used in our previous studies [23]. It has been shown in [23] that the values of τ, n_h, and ϵ do not appreciably affect the performance of ensembles when a constructive approach is used for designing ensembles.

Two sets of experiments were performed in our study. In the first set of experiments, the value of the correlation parameter λ was set to 1 in both NegBagg and NegBoost. In the second set of experiments, it was set to 0, which is equivalent to using plain bagging and boosting in association with optimization of the

TABLE II
PERFORMANCE OF NEGBAGG WITH $\lambda = 0$ AND $\lambda = 1$ FOR DIFFERENT CLASSIFICATION PROBLEMS. THE RESULTS WERE AVERAGED OVER 30 INDEPENDENT RUNS

Data set	Value of	No. of HNs		Error rate		No. of epochs	
	λ	Mean	S.D.	Mean	S.D.	Mean	S.D.
Card	0	5.2	1.4	0.129	0.014	50.3	7.8
	1	4.3	1.2	0.112	0.012	42.5	7.0
Cancer	0	4.0	1.2	0.030	0.014	48.4	6.0
	1	3.7	1.1	0.026	0.013	39.3	6.7
Diabetes	0	4.9	1.3	0.214	0.014	54.3	6.4
	1	4.2	1.2	0.193	0.015	48.5	5.6
Glass	0	5.0	1.0	0.242	0.014	73.3	6.8
	1	4.6	0.9	0.215	0.012	66.6	6.9
Heart	0	3.9	1.1	0.121	0.014	50.2	8.3
	1	3.4	1.0	0.102	0.012	42.1	7.9
Letter	0	12.7	1.3	0.093	0.013	129.5	12.1
	1	9.1	1.5	0.079	0.014	100.7	10.1
Satellite	0	5.9	0.8	0.098	0.015	74.3	5.6
	1	5.5	0.3	0.080	0.012	60.4	4.8
Soybean	0	5.1	1.3	0.066	0.012	64.2	5.6
	1	4.6	1.3	0.059	0.011	59.1	5.3
Waveform	0	5.5	1.2	0.176	0.203	66.7	4.2
	1	4.6	1.4	0.163	0.205	57.0	4.1

TABLE III
PERFORMANCE OF NEGBOOST WITH $\lambda = 0$ AND $\lambda = 1$ FOR DIFFERENT CLASSIFICATION PROBLEMS. THE RESULTS WERE AVERAGED OVER 30 INDEPENDENT RUNS

Data set	Value of	No. of NNs		No. of HNs		Error rate		No. of epochs	
	λ	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
Card	0	6.1	1.5	4.5	1.3	0.104	0.070	229.6	10.5
	1	5.3	1.6	4.0	1.1	0.085	0.010	210.3	10.1
Cancer	0	3.6	1.3	3.3	1.3	0.017	0.017	98.3	5.3
	1	3.3	1.3	2.6	1.1	0.013	0.018	94.4	5.2
Diabetes	0	4.9	1.4	4.3	1.4	0.208	0.050	163.2	6.3
	1	4.2	1.4	3.8	1.3	0.187	0.033	150.3	6.5
Glass	0	5.0	1.3	4.5	1.3	0.271	0.036	191.6	5.7
	1	4.5	1.2	3.7	1.1	0.238	0.028	182.1	5.5
Heart	0	5.0	1.4	5.2	1.3	0.159	0.085	233.4	7.5
	1	4.5	1.3	4.1	1.4	0.137	0.066	221.2	7.3
Letter	0	13.3	1.3	10.3	1.8	0.089	0.061	899.1	10.1
	1	10.8	1.4	8.3	1.9	0.041	0.045	854.5	10.7
Satellite	0	8.0	1.4	6.3	1.3	0.096	0.033	449.2	7.5
	1	7.3	1.3	5.7	1.1	0.076	0.020	430.3	7.3
Soybean	0	4.1	1.5	5.0	1.6	0.065	0.019	181.3	6.0
	1	3.8	1.4	4.3	1.2	0.061	0.016	176.3	5.3
Waveform	0	8.1	1.5	6.3	1.5	0.177	0.021	458.1	6.1
	1	7.0	1.4	5.4	1.4	0.156	0.020	435.7	6.2

ensemble architecture. Other parameters, such as the training algorithm, learning rate, and the heuristics for adding hidden nodes, were chosen the same in both sets of experiments. The use of the same experimental settings will provide a fair chance to compare the results of the two sets of experiments. The comparison may give the answer to a crucial question: whether is it beneficial to use NegBagg and NegBoost or is it sufficient to use bagging and boosting in association with optimization of the ensemble architecture?

B. Experimental Results

Tables II and III summarize the average results of NegBagg and NegBoost, respectively, over 30 independent runs on nine classification problems. The error rate in the tables refers to the percentage of wrong classifications produced by the trained ensemble on the testing set. The number of epochs refers to the total number of iterations required for training all individual NNs in an ensemble.

It is clear from Tables II and III that the ensemble architectures produced and the number of training epochs required by both NegBagg and NegBoost were influenced by the value of λ. For example, for the card problem, the average numbers of hidden nodes in the NNs of an ensemble produced by NegBagg with $\lambda = 0$ and $\lambda = 1$ were 5.2 and 4.3, respectively. The average numbers of epochs required by NegBagg with $\lambda = 0$ and $\lambda = 1$ were 50.3 and 42.5, respectively. These results indicate that a value of unity for λ in NegBagg helps produce compact NN architectures that require a small number of training epochs. This is reasonable because when $\lambda = 1$, NNs in the ensemble can communicate with each other during training. Thus, different NNs in the ensemble can cooperatively learn separate parts or aspects of training data. In contrast, when $\lambda = 0$, NNs cannot communicate with each other during training. Consequently, different NNs in the ensemble may learn the same parts or aspects of the training data because there are overlaps among examples in different training sets created by bagging [48]. This means that the ensemble will learn redundant information when $\lambda = 0$. It is natural to require more computational resources, i.e., large architectures and training epochs, to learn redundant information. Similar phenomena were also observed for NegBoost.

The positive effect of using $\lambda = 1$ in NegBagg and NegBoost is also seen from the classification accuracies. For the card problem, for example, the average testing error rates of NegBagg with $\lambda = 0$ and $\lambda = 1$ were 0.129 and 0.112, respectively. For the diabetes problem, the average error rates of NegBoost with $\lambda = 0$ and $\lambda = 1$ were 0.208 and 0.187, respectively. As shown in Tables II and III, both NegBagg and NegBoost with $\lambda = 0$ produced large ensemble architectures, i.e., NNs with more hidden nodes. It is well known that small NN architectures are more likely to have better generalization ability, i.e., smaller testing error rates [23], [40]. In summary, when the same architectural optimization procedure and experimental setup were used, NegBagg and NegBoost with $\lambda = 1$ were found better than their counterparts with $\lambda = 0$ for producing compact ensemble architectures with small testing error rates and epochs. This indicates the essence of using negative correlation learning in conjunction with bagging or boosting for producing better ensembles.

In this paper, the t-test was used to determine whether the performance difference between two different sets of experiments is statistically significant or not. It was found that NegBagg or NegBoost with $\lambda = 1$ was significantly better than its counterpart with $\lambda = 0$ at 95% confidence level, with the exception of the cancer problem. The *t-test* based on error rate

indicated that the performance of NegBagg (or NegBoost) with $\lambda = 0$ and $\lambda = 1$ was similar for the cancer problem, which is the easiest problem in the realm of machine learning.

The *t-test* between NegBoost and NegBagg with $\lambda = 1$ indicated that the error rate of NegBagg is significantly lower than that of NegBoost for the glass and heart problems. It is shown in Table I that the number of training examples for these two problems is small. Furthermore, the heart problem contains many noisy examples or missing attributes. It is known that boosting is not very effective for small and noisy data [48]–[51]. Neither NegBagg nor NegBoost was found superior for the soybean problem. For the other problems, when the number of training examples was large or the examples were nonnoisy, the error rate of NegBoost was significantly better than that of NegBagg. However, NegBagg required a significantly smaller number of epochs than that of NegBoost for all problems. This is because NegBagg trained all the NNs of an ensemble in parallel, whereas NegBoost sequentially trained them one after another. The difference in average training epochs between NegBagg and NegBoost is at least an order of magnitude three for all problems (Tables II and III). The order is high for problems with a large number of training examples, e.g., the letter problem. The experimental results indicate that NegBagg could be a good choice when the training time is an important issue, or there are few examples in the data set, or the examples in the data set contains noise. It is known that both the error rate and the training time are important in many application areas; improving one at the expense of the other becomes a crucial decision [52]. It would be highly desirable to devise some efficient parallel implementation mechanism of NegBoost.

Although it was found that NegBoost is better than boosting, an interesting question arises as to whether NegBoost is better than adaboost.M1 [26] in terms of the ability to deal with noise. To investigate this issue, these two algorithms were applied on the following regression function [21]:

$$f(\mathbf{x}) = \frac{1}{13}\left[10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5\right] - 1 \quad (12)$$

where $\mathbf{x} = [x_1, \ldots, x_5]$ is an input vector whose components lie between 0 and 1. The value of $f(\mathbf{x})$ lies in the interval $[1, -1]$. The training and testing sets consisted of 500 and 1024 input–output examples, respectively. Here, the components of the input vectors were independently sampled from a uniform distribution over the interval (0, 1). The target outputs were created by adding moderate noise ($\sigma^2 = 0.1$) and large noise ($\sigma^2 = 0.2$) sampled from a Gaussian distribution with a mean of 0 and a variance of σ^2 to the function. The same noise condition is used in [16] to determine the effect of their algorithm on noisy data. To make a fair comparison, adaboost used the same ensemble architectures and epochs that were obtained by NegBoost with $\lambda = 1$. It is important to note here that the suggestions provided in [53] were used for conducting experiments on the regression problem using NegBoost and adaboost.M1.

TABLE IV
COMPARISON BETWEEN NEGBOOST AND ADABOOST WITH NO, MODERATE, AND HIGH NOISE CONDITIONS

Method	Mean squared error on		
	no noise	moderate noise	high noise
NegBoost	0.0094	0.0112	0.0165
AdaBoost	0.0147	0.0182	0.0401

Table IV compares the performance of NegBoost and adaboost for the regression function with no, moderate, and high noise conditions. The results show that NegBoost outperformed adaboost in terms of the mean squared error. It is seen that the performance of adaboost is more affected by noise when compared with NegBoost. Since adaboost puts more weights on training examples that were misclassified, this leads adaboost to overfit very badly in noisy conditions. In contrast, NegBoost does not put more weights on misclassified examples so its performance is not appreciably affected due to noise.

C. Analyses

A number of measures have been proposed in the literature with the aim of gaining a better understanding of the ensemble performance. The three most widely used measures are correlation, margin, and bias–variance decomposition. The purpose of this section is to analyze the performance of our proposed algorithms based on these three measures.

1) Correlation: Correlation is one of the most common and most useful statistics that describes the degree of relationship between two variables. A number of criteria have been proposed in the literature to estimate the correlation between different pairs of NNs in an ensemble [54]. However, there has not been a conclusive study showing which measure is the best. In our study, the correlation between two networks i and j is calculated as follows [17]:

$$Cor(i,j) = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K}\left(F_i^{(k)}(n) - \overline{F}_i(n)\right)}{\sqrt{\sum_{n=1}^{N}\sum_{k=1}^{K}\left(F_i^{(k)}(n) - \overline{F}_i(n)\right)^2}} \times \frac{\sum_{n=1}^{N}\sum_{k=1}^{K}\left(F_j^{(k)}(n) - \overline{F}_j(n)\right)}{\sqrt{\sum_{n=1}^{N}\sum_{k=1}^{K}\left(F_j^{(k)}(n) - \overline{F}_j(n)\right)^2}} \quad (13)$$

where $F^{(k)}(n)$ and $F_i^{(k)}(n)$ are the outputs of the ensemble and network i, respectively, of the nth pattern in the testing set from the kth simulation, and K is the number of simulations. When the ensemble consisted of M NNs, there are C_2^M correlations in total between different pairs of NNs.

Twenty-five simulations of an ensemble architecture obtained by NegBagg and NegBoost were conducted for three different problems. The same number of simulations was conducted as in the previous study to determine correlations among NNs in the ensemble [17]. In each simulation, the same

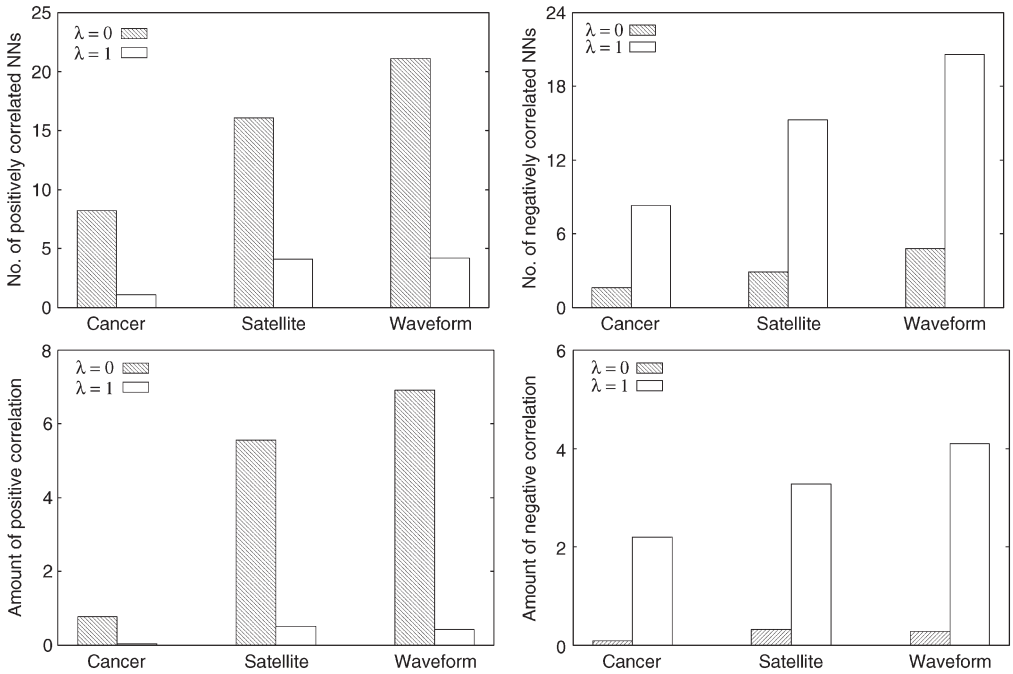


Fig. 1. Correlation among individual NNs produced by NegBagg with $\lambda = 0$ and $\lambda = 1$ for three different classification problems.

ensemble architecture was trained on a different training set from the same random initial weights. The training algorithm employed in each simulation was also the same. In other words, different simulations of the same architecture could yield different performances only due to the use of different training sets. Such an experimental setup follows the suggestion from [21]. The architectures of ensembles for different problems were chosen from our previous experiments. There were five, seven, and eight NNs, respectively, in the ensemble architectures for the cancer, satellite, and waveform problems. These values were the same for both NegBagg and NegBoost. The reason for choosing the same value for NegBagg and NegBoost is to make a fair comparison between the two algorithms.

Figs. 1 and 2 summarize our experimental results for representing correlations between NNs in ensemble for three different problems. It is observed that NegBagg and NegBoost with $\lambda = 1$ tended to produce more negatively correlated NNs for the ensembles than their counterparts with $\lambda = 0$. The number of negatively correlated NNs and the summation of negative correlations were large for NegBagg and NegBoost with $\lambda = 1$. In contrast, the number of positively correlated NNs and the summation of positive correlations were large for NegBagg and NegBoost with $\lambda = 0$. The t-test indicated that NegBagg and NegBoost with $\lambda = 1$ were superior to their counterparts with $\lambda = 0$ for producing negative correlation. However, NegBagg and NegBoost with $\lambda = 1$ were inferior to their counterparts with $\lambda = 0$ for producing positive correlation, which is not beneficial for the performance of ensembles.

It is also shown in Figs. 1 and 2 that NegBoost with $\lambda = 1$ produces a larger number of negatively correlated NNs than NegBagg with $\lambda = 1$. In addition, the summation of negative correlations produced by NegBoost with $\lambda = 1$ was larger than that produced by NegBagg with $\lambda = 1$. A possible reason for such difference might be a nonoptimal M (i.e., number of NNs in an ensemble) we used in NegBagg. The t-test indicated that NegBoost with $\lambda = 1$ produced a significantly larger amount of negative correlation than that of NegBagg with $\lambda = 1$ with 95% confidence level for the cancer, satellite, and waveform problems. However, when the t-test was conducted based on the number of negatively correlated NNs, NegBoost with $\lambda = 1$ was found to be significantly better than that of NegBagg with $\lambda = 1$ for the satellite and waveform problems. For the cancer problem, the performance of the two algorithms was found to be similar.

2) Margin: The margin of an example is defined as the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label. Schapire *et al.* [55] proposed an analysis of the generalization performance of bagging [13], [56], [57] and boosting [14], [26] using margin. Margin is generally expressed as a number between -1 and $+1$, and is positive if the example is correctly classified. Furthermore, the absolute value of the margin represents the confidence of classification.

It is customary to plot the margin as a cumulative distribution graph, that is, $f(z)$ versus z, where z is the margin, and $f(z)$ is its cumulative value. A margin distribution curve that moves to the right is indicative of a more confident classification. It has also been shown that large margins are associated with superior upper bounds on the generalization error [55]. The idea that maximizing the margin could improve the generalization error of a classifier was also studied earlier [58], [59]. However, other more recent results show that improving the whole margin distribution can also result in a degraded generalization [57].

The margin distribution curve of NegBagg and NegBoost on cancer and waveform problems is shown in Fig. 3. It is clear from the figure that the number of examples with high margin increased when $\lambda = 1$ was used in NegBagg and NegBoost. For example, for the cancer problem, all examples had a margin greater than 0.5 for NegBoost with $\lambda = 1$. In contrast, some

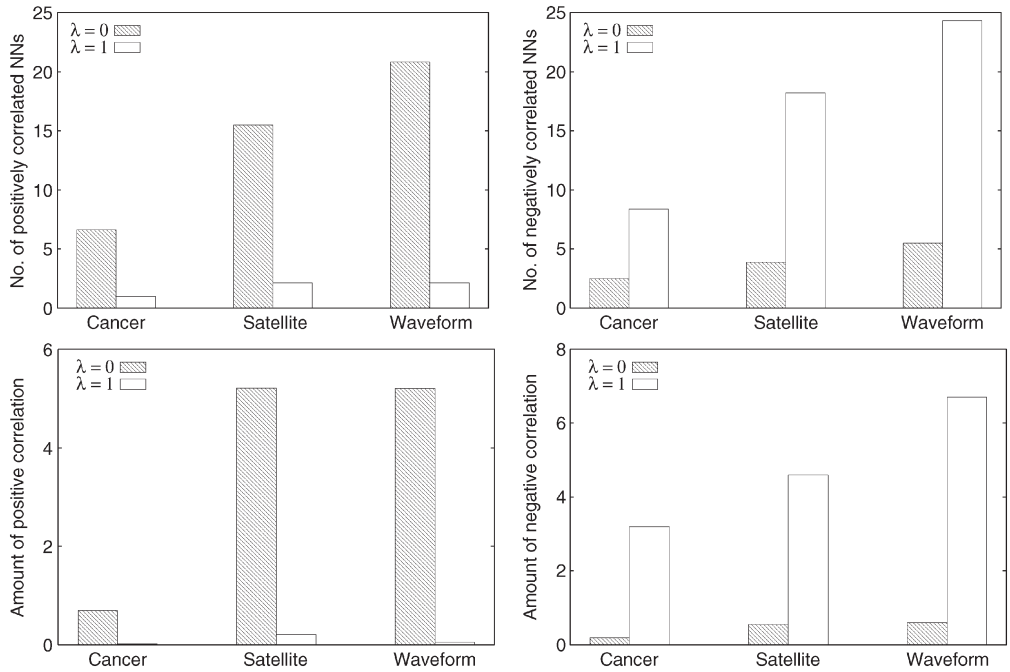


Fig. 2. Correlation among individual NNs produced by NegBoost with $\lambda = 0$ and $\lambda = 1$ for three different classification problems.

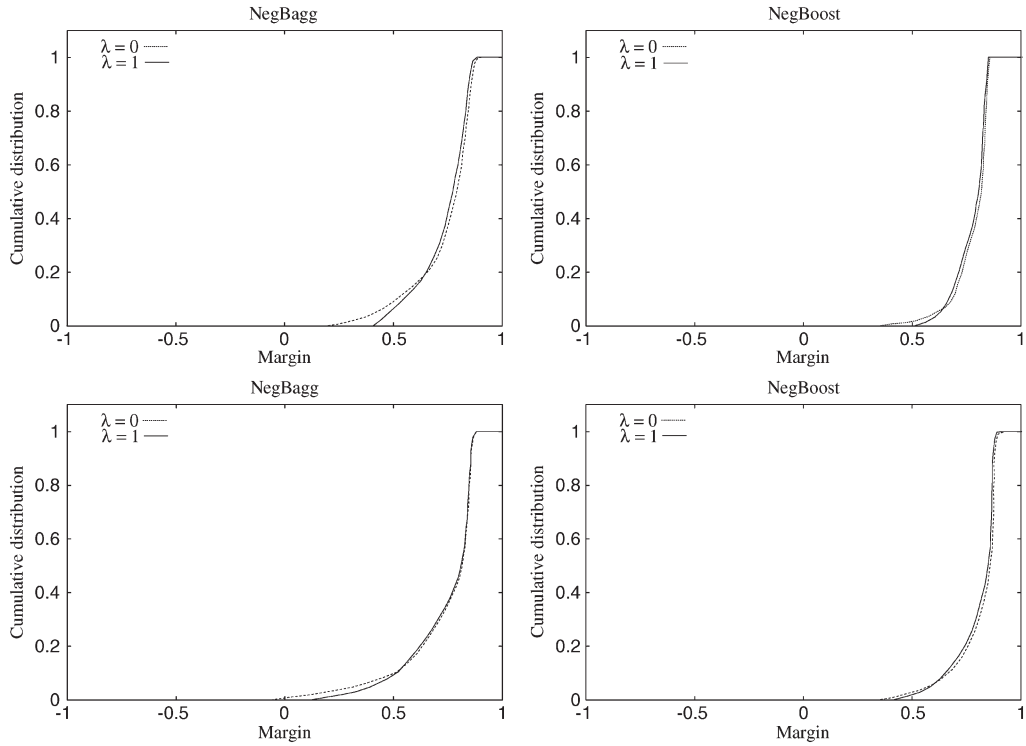


Fig. 3. Margin distributions of NegBagg and NegBoost for (top row) cancer and (bottom row) waveform problems.

examples had a margin below 0.5 when $\lambda = 0$ was used in NegBoost. The achievement of high margin could be attributed to the fact that when $\lambda = 1$, NNs in an ensemble need to learn a smaller number of examples. This is because NegBagg or NegBoost with $\lambda = 1$ can produce more negatively correlated NNs for ensembles than NegBagg or NegBoost with $\lambda = 0$ (Figs. 1 and 2). It is natural that an NN can concentrate more on increasing the margins when it learns a smaller number of examples. The margin distribution curve also indicates that the use of negative correlation learning in bagging or boosting did not change the nature of the curve.

3) Bias and Variance: In the ensemble literature, a number of ideas have been suggested for decomposing the classification error into bias and variance. The decomposition proposed by Kohavi and Wolport [60] is used in this paper to estimate the bias and variance. A two-stage sampling procedure is used in [60]. First, a test set is split from the training set. Then, the remaining data D are repeatedly sampled to estimate the bias

TABLE V
RESULTS OF BIAS–VARIANCE EXPERIMENTS OF NEGBAGG AND NEGBOOST ON THREE DIFFERENT CLASSIFICATION PROBLEMS

Data set		NegBagg with		NegBoost with	
		$\lambda = 0$	$\lambda = 1$	$\lambda = 0$	$\lambda = 1$
Cancer	bias	0.005	0.003	0.003	0.002
	var	0.055	0.046	0.044	0.040
Waveform	bias	0.131	0.112	0.109	0.085
	var	0.051	0.048	0.050	0.045
Satellite	bias	0.087	0.076	0.071	0.061
	var	0.056	0.038	0.035	0.029

and variance on the testing set. According to [60], D is to be chosen in such a way that it is twice the size of the desired training set. We create ten training sets from D using uniform random sampling without replacement. The whole process is repeated three times to get stable estimates.

The results of bias–variance experiments on three different problems are summarized in Table V. Both NegBagg and NegBoost show a reduced variance with an increased λ. For example, for the satellite problem, the variance of NegBagg was 0.056 when $\lambda = 0$, and it was reduced to 0.038 when $\lambda = 1$. The effect of using $\lambda = 1$ for producing compact NN architectures is evident from the results in Tables II and III. These two results are related in the sense that NNs with small architectures could reduce the variance of the ensemble. The reduction of variance can also be explained by the fact that with increased λ, the NNs in the ensemble are more communicative to reduce correlation (Figs. 1 and 2). It is theoretically shown in Section III-C that the reduction of correlation between NNs in the ensemble is beneficial for reducing the variance.

The effect of λ on the bias of an ensemble is also shown in Table V. The bias is reduced when the NNs in the ensemble are more communicative in learning the task. For example, the bias is reduced from 0.131 $(\lambda = 0)$ to 0.112 $(\lambda = 1)$ when NegBagg is applied to the waveform problem. Since NNs in the ensemble focus on solving different portions of a task when $\lambda = 1$, they are now more specialized and will make less error on their portions of the task. The overall effect is that the bias is reduced. Similar results were also observed for NegBoost.

D. Comparison With Other Work

There are many ensemble learning algorithms in the literature that could be compared. However, it is not feasible and not necessary to conduct an exhaustive comparison with all the ensemble learning algorithms. The goal of our experimental comparison here is to evaluate and understand the strengths and weaknesses of NegBagg and NegBoost as applied to different problems. The proposed algorithms are compared here against nine most relevant works. They are bagging [13], arcboosting [57], and adaboosting.M1 [26] tested by Opitz and Maclin [48]; bagging and adaboosting.M2 tested by Schwenk and Bengio [42]; bagging and adaboosting.M1 tested by Dietterich [50]; CNNE [23]; and evolutionary ensemble with negative correlation learning (EENCL) [15].

TABLE VI
COMPARISON BETWEEN NEGBAGG, CNNE [36], BAGGED NN [48], BAGGED NN [42], BAGGED C4.5 [50], AND EENCL [15]. NN AND C4.5 INDICATE THE TYPE OF CLASSIFIER USED WITH DIFFERENT APPROACHES. "—" MEANS NOT AVAILABLE

Data set	NN					C4.5
	NegBagg	CNNE	EENCL	Bagged [48]	Bagged [42]	Bagged
Card	0.112	0.092	0132	0.138	-	0.141
Cancer	0.026	0.013	-	0.034	-	0.042
Diabetes	0.193	0.198	0221	0.228	-	0.236
Glass	0.215	0.268	-	0.331	-	0.270
Heart	0.102	0.134	-	0.170	-	0.203
Letter	0.079	0.062	-	0.105	0.043	0.075
Satellite	0.080	-	-	0.106	0.087	-
Soybean	0.059	0.076	-	0.069	-	0.758
Waveform	0.163	-	-	-	-	0.197

The aforementioned algorithms represent a wide range of ensemble approaches. CNNE and EENCL, for example, employ constructive and evolutionary approaches, respectively, for automatically designing ensembles. However, the remaining algorithms do not design the ensembles, but rather use predefined and fixed ensemble architectures. Two types of base classifiers are used in different algorithms. Dietterich [50] used C4.5 as a base classifier with bagging and adaboosting.M1, whereas NN was used as a base classifier in all the other algorithms. A ten-fold cross validation was used in all the algorithms except [42] and the proposed algorithms. The data sets were partitioned into training and testing sets in [42], whereas they were partitioned into training, validation, and testing sets in this paper.

Table VI compares the average testing error rates of NegBagg with those of CNNE, bagged NN, or C4.5 and EENCL. Table VI shows that NegBagg achieved the smallest error rates for six out of nine problems and the second smallest (next to CNNE) for two problems. For one problem, the bagged NN tested in [42] achieved the smallest error rate, whereas NegBagg achieved the third smallest error rate.

The improved performance of NegBagg could be attributed to a couple of factors. First, bagged NN or C4.5 independently trained individual NNs or constructed decision trees for an ensemble. The NNs or decision trees could not communicate with each other during their training or construction process. In contrast, NegBagg with $\lambda = 1$ cooperatively trains NNs in the ensemble using negative correlation. It is clear from our previous experiments that the communication among NNs in an ensemble is helpful for achieving small testing error rates (Tables II and III).

Second, EENCL and bagged NN manually determined the architecture of individual NNs, whereas NegBagg automatically determined the architecture. It is well known that the performance of any NN is greatly dependent on its architecture. While manual determination of NN architectures might be appropriate for problems where rich prior knowledge and an experienced expert exist, it is inappropriate when such knowledge and expertise are unavailable.

The reason that NegBagg was outperformed by CNNE and bagged NN [42] on two and one problems, respectively, might be its nonoptimal setting of M, i.e., the number of NNs in the

TABLE VII
COMPARISON BETWEEN NEGBOOST, CNNE [36], ADABOOSTED.M1 C4.5 [51], ARCBOOSTED [48], AND ADABOOSTED.M1 NN [48] BASED ON THE AVERAGE TESTING ERROR RATE. NN AND C4.5 INDICATE THE TYPE OF CLASSIFIER USED WITH DIFFERENT APPROACHES. "—" MEANS NOT AVAILABLE

Data set	NN					C4.5
	NegBoost	CNNE	Arcboosted	Adaboosted. M1	Adaboosted. M2	Adaboosted. M1
Card	0.085	0.092	0.158	0.157	-	0.156
Cancer	0.013	0.013	0.038	0.040	-	0.040
Diabetes	0.187	0.198	0.244	0.233	-	0.281
Glass	0.238	0.268	0.320	0.311	-	0.2355
Heart	0.137	0.134	0.207	0.211	-	0.210
Letter	0.041	0.062	0.057	0.046	0.015	0.046
Satellite	0.076	-	0.099	0.100	0.081	-
Soybean	0.061	0.076	0.067	0.063	-	0.071
Waveform	0.156	-	-	-	-	0.185

ensemble. CNNE can automatically determine the number of NNs in the ensemble, whereas NegBagg cannot. In [42], a large ensemble architecture was used for the letter problem, which might be suitable for this problem. Furthermore, the training strategies used in CNNE, bagged NN [42], and NegBagg were different, and they might contribute to the performance difference.

Table VII compares the average testing error rates of NegBoost with those of CNNE, arcboosted, adaboosted.M1, and adaboosted.M2. It is shown in Table VII that NegBoost was able to achieve the smallest average testing error rate among all the algorithms on six out of nine problems. For two problems, the error rate of NegBoost and CNNE was similar. There are three possible reasons why NegBoost performed so well. First, NegBoost automatically determined the ensemble architecture, whereas adaboosted.M1 NN or C4.5, arcboosted NN, and adaboosted.M2 NN manually determined the architecture, which implies that NegBoost could search for more appropriate and different ensemble architectures for different problems. Second, NegBoost trained all NNs in the ensemble for an appropriate (and potentially different) number of training cycles, automatically determined by the algorithm, whereas arcboosted NN, adaboosted.M1 NN, and adaboosted.M2 NN trained NNs with a user-specified and fixed number of training cycles. Third, NegBoost used both negative correlation learning and boosting for training NNs in the ensemble, whereas the others used only one of the two algorithms for training NNs.

It is observed in Tables VI and VII that the error rate of NegBagg and NegBoost was worse than that of bagging and boosting tested in [42] for the letter problem, which was the largest problem among all the problems considered in this paper. One reason might be that adaboost.M2 was used in [42], which is superior to the boosting algorithm used in NegBoost. Furthermore, the ensemble architecture used in [42] for the letter problem was much larger than that automatically obtained by NegBagg and NegBoost. This raises a question as to whether the proposed algorithms scale well for large real-world problems. Since NegBagg and NegBoost used a nonevolutionary constructive approach for determining ensemble architectures, the architecture determination process of these algorithms might trap into the local optima. However, this is not an inherent problem because NegBagg and NegBoost can easily be modified to accommodate any kind of evolutionary or nonevolutionary approach (e.g., [40], [61]–[63]) for determining ensemble architectures. One of our future research projects will be to study algorithms and techniques suitable for designing ensembles for large problems.

The use of the constructive approach [64] in this paper gives us an opportunity for making a direct comparison with our previous work CNNE [23]. The effect of using negative correlation in conjunction with bagging or boosting is therefore clearly understood. It is straightforward for an inexperienced user to specify the initial conditions for the constructive approach [36], [39], [65]. Furthermore, the constructive algorithm is computationally more efficient because it always searches small solutions first [66]–[68], so an algorithm can be tested on several problems within a reasonable amount of time.

V. CONCLUSION

Bagging, boosting, and negative correlation learning are popular approaches to train NN ensembles. Bagging [13], [56], [57] and boosting [14], [26] rely on the construction of different training sets for different NNs in an ensemble to encourage error decorrelation among NNs. Negative correlation learning [16], [17], [69], [70] incorporates a penalty term in NN's error function to explicitly promote an error negative correlation. In this paper, we have proposed and studied two algorithms (i.e., NegBagg and NegBoost) that combine the complementary strengths of both negative correlation and bagging/boosting. Our experimental results on nine different problems have shown that NegBagg and NegBoost are able to produce compact ensembles that generalize better than either negative correlation or bagging/boosting by itself.

Our algorithms NegBagg and NegBoost also have the added advantage that the algorithms can automatically determine individual NN architectures during ensemble learning. The number of epochs needed for training each individual NN can also be automatically determined. Furthermore, NegBoost is able to automatically determine the number of individual NNs in an ensemble, relieving the user from a potentially tedious trial-and-error process of manually selecting an optimal number of NNs in the ensemble.

There are several future research directions that logically follow this study. First, an automatic technique for deciding the number of NNs in an ensemble for NegBagg will be very useful in improving NegBagg's performance and its practical use. Second, a more in-depth analysis of NegBagg and NegBoost is needed to gain more insights into when NegBagg and NegBoost are most likely to perform well and for what kind of problems. Third, while NegBagg has concentrated on the classification problems, it would be interesting to study how well NegBagg would perform on regression problems.

ACKNOWLEDGMENT

The authors would like to thank the associate editor and anonymous reviewers for their constructive comments.

REFERENCES

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[2] S. Hashem, "Optimal linear combinations of neural networks," *Neural Netw.*, vol. 10, no. 4, pp. 599–614, Jun. 1997.

[3] S. Hashem and B. Schmeiser, "Improving model accuracy using optimal linear combinations of trained neural networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 792–794, May 1995.

[4] K. M. Ali, "Learning probabilistic relational concept descriptions," Ph.D. dissertation, Univ. California, Irvine, CA, 1996.

[5] K. M. Ali and M. J. Pazzani, "Error reduction through learning multiple descriptions," *Mach. Learn.*, vol. 24, no. 3, pp. 173–202, Sep. 1996.

[6] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 231–238.

[7] A. Krogh and P. Sollich, "Statistical mechanics of ensemble learning," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 55, no. 1, pp. 811–825, Jan. 1997.

[8] D. W. Opitz and J. W. Shavlik, "Actively searching for an effective neural-network ensemble," *Connect. Sci.*, vol. 8, no. 3, pp. 337–353, Dec. 1996.

[9] M. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," in *Neural Networks for Speech and Image Pressing*, R. J. Mammone, Ed. London, U.K.: Chapman & Hall, 1993, pp. 126–142.

[10] A. J. C. Sharkey, "On combining artificial neural nets," *Connect. Sci.*, vol. 8, no. 3, pp. 299–314, Dec. 1996.

[11] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connect. Sci.*, vol. 8, no. 3, pp. 385–404, Dec. 1996.

[12] A. J. C. Sharkey and N. E. Sharkey, "Combining diverse neural nets," *Knowl. Eng. Rev.*, vol. 12, no. 3, pp. 1–17, Sep. 1997.

[13] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[14] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, Jun. 1990.

[15] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 380–387, Nov. 2000.

[16] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Netw.*, vol. 12, no. 10, pp. 1399–1404, Dec. 1999.

[17] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 6, pp. 716–725, Dec. 1999.

[18] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," *Neural Comput.*, vol. 6, no. 6, pp. 1289–1301, 1994.

[19] B. E. Rosen, "Ensemble learning using decorrelated neural networks," *Connect. Sci.*, vol. 8, no. 3, pp. 373–383, Dec. 1996.

[20] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.

[21] R. A. Jacobs, "Bias/variance analyses of mixture-of-experts architectures," *Neural Comput.*, vol. 9, no. 2, pp. 369–383, Feb. 1997.

[22] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Comput.*, vol. 6, no. 2, pp. 181–214, Mar. 1994.

[23] M. M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 820–834, Jul. 2003.

[24] N. J. Nilsson, *Learning Machines: Foundation of Trainable Pattern-Classifying System*. New York: McGraw-Hill, 1965.

[25] O. G. Selfridge, "Pandemonium: A paradigm for learning," in *Proc. Symp. Mechanization Thought Process*, London, U.K., 1958, pp. 513–526.

[26] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 146–148.

[27] Y. Raviv and N. Intrator, "Bootstrapping with noise: An effective regularization technique," *Connect. Sci.*, vol. 8, no. 3, pp. 356–372, Dec. 1996.

[28] K. J. Cherkauer, "Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks," in *Proc. AAAI Workshop Integrating Multiple Learned Models Improving Scaling Mach. Learn. Algorithms*, P. Chan, S. Stolfo, and D. Wolpert, Eds., Portland, OR, 1996, pp. 15–21.

[29] R. Maclin and J. Shavlik, "Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, 1995, pp. 524–530.

[30] N. C. Oza and K. Tumer, "Input decimation ensembles: Decorrelation through dimensionality reduction," in *Proc. Int. Workshop Multiple Classifier Syst.* Cambridge, U.K.: Springer-Verlag, 2001, vol. 2096, pp. 238–247.

[31] P. Melville and R. J. Mooney, "Creating diversity in ensembles using artificial data," *Inf. Fusion*, vol. 6, no. 1, pp. 99–111, Mar. 2005.

[32] P. M. Granitto, P. F. Verdes, and H. A. Ceccatto, "Neural network ensembles: Evaluation of aggregation algorithms," *Artif. Intell.*, vol. 163, no. 2, pp. 139–162, Apr. 2005.

[33] Y. Liu, "Generate different neural networks by negative correlation learning," in *Advances in Natural Computation*, vol. 3610. New York: Springer-Verlag, 2005, pp. 149–156.

[34] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: A survey and categorisation," *Inf. Fusion*, vol. 6, no. 1, pp. 5–20, Mar. 2005.

[35] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing System 2*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524–532.

[36] M. M. Islam and K. Murase, "A new algorithm to design compact two-hidden-layer artificial neural networks," *Neural Netw.*, vol. 14, no. 9, pp. 1265–1278, Nov. 2001.

[37] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[38] L. Ma and K. Khorasani, "Constructive feedforward neural networks using Hermite polynomial activation functions," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 821–833, Jul. 2005.

[39] T. Y. Kwok and D. Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 630–645, May 1997.

[40] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 694–713, May 1997.

[41] S. V. Odri, D. P. Petrovacki, and G. A. Krstonosic, "Evolutional development of a multilevel neural network," *Neural Netw.*, vol. 6, no. 4, pp. 583–595, 1993.

[42] H. Schwenk and Y. Bengio, "Boosting neural networks," *Neural Comput.*, vol. 12, no. 8, pp. 1869–1887, Aug. 2000.

[43] Y. Chauvin, "Generalization performance of overtrained backpropagation networks," in *Proc. EUROSZP Workshop*, Feb. 1990, pp. 46–55.

[44] P. Baldi and Y. Chauvin, "Temporal evolution of generalization during learning in linear networks," *Neural Comput.*, vol. 3, no. 4, pp. 589–603, 1991.

[45] L. Prechelt, "Automatic early stopping using cross validation: Quantifying the criteria," *Neural Netw.*, vol. 11, no. 4, pp. 761–767, Jun. 1998.

[46] L. Prechelt, "Proben1—A set of neural network benchmark problems and benchmarking rules," Fakultät für Informatik, Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, Sep. 1994.

[47] L. Prechelt, "Some notes on neural learning algorithm benchmarking," *Neurocomputing*, vol. 9, no. 3, pp. 343–347, Dec. 1995.

[48] D. W. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, 1999.

[49] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learn.*, vol. 36, no. 1/2, pp. 105–139, Jul./Aug. 1999.

[50] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, Aug. 2000.

[51] J. R. Quinlan, "Bagging, boosting, and C4.5," in *Proc. 13th Nat. Conf. Artif. Intell., 8th Innovative Appl. Artif. Intell. Conf.*, Menlo Park, CA, Aug. 4–8, 1996, pp. 725–730.

[52] K. Jim, C. L. Giles, and B. G. Horne, "An analysis of noise in recurrent neural networks: convergence and generalization," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1424–1438, Nov. 1996.

[53] H. Drucker, "Improving regressors using boosting techniques," in *Proc. 14th Int. Conf. Mach. Learn.*, 1997, pp. 107–115.

[54] L. Kuncheva and C. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, May 2003.

[55] R. E. Schapire, P. Bartlett, Y. Freund, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Stat.*, vol. 26, no. 5, pp. 1651–1686, Oct. 1998.

[56] L. Breiman, "Stacked regressions," *Mach. Learn.*, vol. 24, no. 1, pp. 49–64, Jul. 1996.

[57] L. Breiman, "Prediction games and arcing algorithms," *Neural Comput.*, vol. 11, no. 7, pp. 1493–1518, Oct. 1999.

[58] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. ACM Workshop Comput. Learn. Theory*, 1992, pp. 144–152.

[59] C. Cortes and V. N. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[60] R. Kohavi and D. H. Wolport, "Bias plus variance decomposition for zero one loss function," in *Proc. 3rd Int. Conf. Mach. Learn.*, 1996, pp. 275–283.
[61] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 740–747, Sep. 1993.
[62] R. Setiono, "Extracting rules from neural networks by pruning and hidden—Unit splitting," *Neural Comput.*, vol. 9, no. 1, pp. 205–225, Jan. 1997.
[63] Z.-H. Zhou, S. Chen, and Z. Chen, "FANRE: A fast adaptive neural regression estimator," in *Proc. 12th Australian Joint Conf. Artif. Intell.*, 1999, vol. 1747, pp. 48–59.
[64] T. Ash, "Dynamic node creation in backpropagation networks," *Connect. Sci.*, vol. 1, no. 4, pp. 365–375, 1989.
[65] Y. Hirose, K. Yamashita, and S. Hijiya, "Back-propagation algorithm which varies the number of hidden units," *Neural Netw.*, vol. 4, no. 1, pp. 61–66, 1991.
[66] T. Y. Kwok and D. Y. Yeung, "Objective functions for training new hidden units in constructive neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1131–1148, Sep. 1997.
[67] M. Lehtokangas, "Modeling with constructive backpropagation," *Neural Netw.*, vol. 12, no. 4/5, pp. 707–716, Jun. 1999.
[68] T. K. Nicholas and T. D. Gedeon, "Exploring constructive cascade networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1335–1350, Nov. 1999.
[69] Y. Liu and X. Yao, "A cooperative ensemble learning system," in *Proc. IEEE IJCNN*, 1998, pp. 2202–2207.
[70] Y. Liu and X. Yao, "Negatively correlated neural networks can produce best ensembles," *Aust. J. Intell. Inf. Process. Syst.*, vol. 4, no. 3/4, pp. 176–185, 1997.

Md. Monirul Islam received the B.E. degree from the Khulna University of Engineering and Technology (KUET), Khulna, Bangladesh, in 1989, the M.E. degree from the Bangladesh University of Engineering and Technology (BUET), Dhaka, in 1996, and the Ph.D. degree from the University of Fukui, Fukui, Japan, in 2002.

From 1989 to 2002, he was a Lecturer and an Assistant Professor with KUET. Since 2003, he has been with BUET, where he was an Assistant Professor of Computer Science and Engineering Department, and is currently an Associate Professor. He is also a Visiting Associate Professor with the University of Fukui, supported by the Japanese Society for the Promotion of Sciences. He has more than 50 refereed publications. His major research interests include evolutionary robotics, evolutionary computation, neural networks, machine learning, pattern recognition, and data mining.

Dr. Islam won the First Prize in the Best Paper Award Competition of the Joint 3rd International Conference on Soft Computing and Intelligent Systems, and the 7th International Symposium on Advanced Intelligent Systems.

Xin Yao (M'91–SM'96–F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, in 1985, and the Ph.D. degree from USTC, in 1990.

From 1985 to 1990, he was an Associate Lecturer and a Lecturer with USTC. In 1990, he took up a Postdoctoral Fellowship with the Computer Sciences Laboratory, Australian National University, Canberra, and continued his work on simulated annealing and evolutionary algorithms. In 1991, he joined the Knowledge-Based Systems Group, Commonwealth Scientific and Industrial Research Organisation, Division of Building, Construction and Engineering, Melbourne, Australia, where he primarily worked on an industrial project on the automatic inspection of sewage pipes. In 1992, he returned to Canberra to take up lectureship in the School of Computer Science, University College, University of New South Wales, Kensington, Australia, and the Australian Defence Force Academy, Canberra, where he was later promoted to a Senior Lecturer and Associate Professor. In 1999, he was a Professor of computer science with the University of Birmingham, Birmingham, U.K. He is currently the Director of the Centre of Excellence for Research in Computational Intelligence and Applications, and a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) with USTC. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. He has more than 250 refereed publications. His major research interests include evolutionary artificial neural networks, automatic modularization of machine learning systems, evolutionary optimization, constraint handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner's dilemma, data mining, and real-world applications.

Dr. Yao is the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, an Associate Editor or Editorial Board Member of ten other journals, and the Editor of the World Scientific Book Series on Advances in Natural Computation. He was awarded the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. work on simulated annealing and evolutionary algorithms in 1989. He won the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.

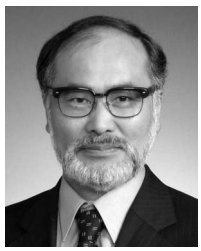
S. M. Shahriar Nirjon received the B.Sc. degree in computer science and engineering in 2006 from the Bangladesh University of Engineering and Technology (BUET), Dhaka, where he is currently working toward the M.Sc. degree.

He is currently a Lecturer with the Department of Computer Science and Engineering, BUET. His research interests include machine learning, neural networks, computer vision, and biorobotics.

Mr. Shahriar Nirjon has won a number of awards in national and international programming contests including the Association for Computing Machinery International Collegiate Programming Contest. He is also a winner of the famous $2.56 check of Dr. D. E. Knuth.

Muhammad Asiful Islam received the B.Sc. degree in computer science and engineering from the Bangladesh University of Engineering and Technology, Dhaka, in 2006. He is currently working toward the Ph.D. degree at the State University of New York at Stony Brook, Stony Brook, NY.

His research interests include machine learning, web mining, and application of machine learning techniques to other areas of computer science.



Kazuyuki Murase received the M.E. degree in electrical engineering from Nagoya University, Nagoya, Japan, in 1978, and the Ph.D. degree in biomedical engineering from Iowa State University, Ames, in 1983.

In 1984, he was a Research Associate with the Department of Information Science, Toyohashi University of Technology, Toyohashi, Japan. He was an Associate Professor in 1988 and a Professor in 1992 with the Department of Information Science, University of Fukui, Fukui, Japan. Since 1999, he has been a Professor with the Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui.

Dr. Murase is a member of The Institute of Electronics, Information and Communication Engineers, The Japanese Society for Medical and Biological Engineering, the Japan Neuroscience Society, the International Neural Network Society, and the Society for Neuroscience. He serves as a member of the Board of Directors in the Japan Neural Network Society, a Councilor of the Physiological Society of Japan, and a Councilor of the Japanese Association for the Study of Pain.