ELSEVIER

Artificial Intelligence 137 (2002) 239–263

Artificial Intelligence

www.elsevier.com/locate/artint

Ensembling neural networks: Many could be better than all ☆

Zhi-Hua Zhou *, Jianxin Wu, Wei Tang

National Laboratory for Novel Software Technology, Nanjing University, Hankou Road 22, Nanjing 210093, PR China

Received 16 November 2001

Abstract

Neural network ensemble is a learning paradigm where many neural networks are jointly used to solve a problem. In this paper, the relationship between the ensemble and its component neural networks is analyzed from the context of both regression and classification, which reveals that it may be better to ensemble *many* instead of *all* of the neural networks at hand. This result is interesting because at present, most approaches ensemble *all* the available neural networks for prediction. Then, in order to show that the appropriate neural networks for composing an ensemble can be effectively selected from a set of available neural networks, an approach named GASEN is presented. GASEN trains a number of neural networks at first. Then it assigns random weights to those networks and employs genetic algorithm to evolve the weights so that they can characterize to some extent the fitness of the neural networks in constituting an ensemble. Finally it selects some neural networks based on the evolved weights to make up the ensemble. A large empirical study shows that, compared with some popular ensemble approaches such as Bagging and Boosting, GASEN can generate neural network ensembles with far smaller sizes but stronger generalization ability. Furthermore, in order to understand the working mechanism of GASEN, the bias-variance decomposition of the error is provided in this paper, which shows that the success of GASEN may lie in that it can significantly reduce the bias as well as the variance.

Keywords: Neural networks; Neural network ensemble; Machine learning; Selective ensemble; Boosting; Bagging; Genetic algorithm; Bias-variance decomposition

☆ This is an extended version of the paper presented at IJCAI 2001, Seattle, WA, USA.

* Corresponding author.

E-mail addresses: zhouzh@nju.edu.cn (Z.-H. Zhou), wujx@ai.nju.edu.cn (J. Wu), tangwei@ai.nju.edu.cn (W. Tang).

0004-3702/02/$ – see front matter
PII: S0004-3702(02)00190-X

1. Introduction

Neural network ensemble is a learning paradigm where a collection of a finite number of neural networks is trained for the same task [42]. It originates from Hansen and Salamon's work [20], which shows that the generalization ability of a neural network system can be significantly improved through ensembling a number of neural networks, i.e., training many neural networks and then combining their predictions. Since this technology behaves remarkably well, recently it has become a very hot topic in both neural networks and machine learning communities [40], and has already been successfully applied to diversified areas such as face recognition [16,22], optical character recognition [9,19,30], scientific image analysis [5], medical diagnosis [6,47], seismic signals classification [41], etc.

In general, a neural network ensemble is constructed in two steps, i.e., training a number of component neural networks and then combining the component predictions.

As for training component neural networks, the most prevailing approaches are Bagging and Boosting. Bagging is proposed by Breiman [3] based on bootstrap sampling [10]. It generates several training sets from the original training set and then trains a component neural network from each of those training sets. Boosting is proposed by Schapire [39] and improved by Freund et al. [11,12]. It generates a series of component neural networks whose training sets are determined by the performance of former ones. Training instances that are wrongly predicted by former networks will play more important roles in the training of later networks. There are also many other approaches for training the component neural networks. Examples are as follows. Hampshire and Waibel [17] utilize different object functions to train distinct component neural networks. Cherkauer [5] trains component networks with different number of hidden units. Maclin and Shavlik [29] initialize component networks at different points in the weight space. Krogh and Vedelsby [28] employ cross-validation to create component networks. Opitz and Shavlik [34] exploit genetic algorithm to train diverse knowledge based component networks. Yao and Liu [46] regard all the individuals in an evolved population of neural networks as component networks.

As for combining the predictions of component neural networks, the most prevailing approaches are plurality voting or majority voting [20] for classification tasks, and simple averaging [33] or weighted averaging [35] for regression tasks. There are also many other approaches for combining predictions. Examples are as follows. Wolpert [45] utilizes learning systems to combine component predictions. Merz and Pazzani [31] employs principal component regression to determine the appropriate constraint for the weights of the component networks in combining their predictions. Jimenez [24] uses dynamic weights determined by the confidence of the component networks to combine the predictions. Ueda [43] exploits optimal linear weights to combine component predictions based on statistical pattern recognition theory.

Note that there are some approaches using a number of neural networks to accomplish a task in the style of *divide-and-conquer* [23,25]. However, in those approaches, the neural networks are in fact trained for different subtasks instead of for the same task, which makes those approaches usually be categorized into mixture of experts instead of ensembles, and the discussion of them is beyond the scope of this paper.

It is worth mentioning that when a number of neural networks are available, at present most ensemble approaches employ all of those networks to constitute an ensemble. Yet the goodness of such a process has not been formally proved. In this paper, from the viewpoint of prediction, i.e., regression and classification, the relationship between the ensemble and its component neural networks is analyzed, which reveals that ensembling *many* of the available neural networks may be better than ensembling *all* of those networks. Then, in order to show that those "*many*" neural networks can be effectively selected from a number of available neural networks, an approach named GASEN (Genetic Algorithm based Selective ENsemble) is presented. This approach selects some neural networks to constitute an ensemble according to some evolved weights that could characterize the fitness of including the networks in the ensemble. An empirical study on twenty big data sets show that in most cases, the performance of the neural network ensembles generated by GASEN outperform those generated by some popular ensemble approaches such as Bagging and Boosting in that GASEN utilizes far less component neural networks but achieves stronger generalization ability. Moreover, this paper employs the bias-variance decomposition to analyze the empirical results, which shows that the success of GASEN may owe to its ability of significantly reducing the bias along with the variance.

The rest of this paper is organized as follows. In Section 2, the relationship between the ensemble and its component neural networks is analyzed. In Section 3, GASEN is presented. In Section 4, a large empirical study is reported. In Section 5, the bias-variance decomposition of the error is provided. Finally in Section 6, contributions of this paper are summarized and several issues for future works are indicated.

2. Should we ensemble all the neural networks?

In order to know whether it is a good choice to ensemble all the available neural networks, this section analyzes the relationship between the ensemble and its component neural networks. Note that since regression and classification have distinct characteristics, the analyses are separated into two subsections.

2.1. Regression

Suppose the task is to use an ensemble comprising N component neural networks to approximate a function $f:\mathbb{R}^m \to \mathbb{R}^n$, and the predictions of the component networks are combined through weighted averaging where a weight w_i $(i = 1, 2, \ldots, N)$ satisfying both Eqs. (1) and (2) is assigned to the ith component network f_i

$$0 \leqslant w_i \leqslant 1, \tag{1}$$

$$\sum_{i=1}^{N} w_i = 1. \tag{2}$$

The lth output variable of the ensemble is determined according to Eq. (3) where $f_{i,l}$ is the lth output variable of the ith component network

$$\hat{f}_l = \sum_{i=1}^{N} w_i f_{i,l}. \tag{3}$$

For convenience of discussion, here we assume that each component neural network has only one output variable, i.e., the function to be approximated is $f : \mathbb{R}^m \to \mathbb{R}$. But note that the following derivation can be easily generalized to situations where each component neural network has more than one output variables.

Now suppose $x \in \mathbb{R}^m$ is sampled according to a distribution $p(x)$, the expected output of x is $d(x)$, and the actual output of the ith component neural network is $f_i(x)$. Then the output of the ensemble on x is:

$$\hat{f}(x) = \sum_{i=1}^{N} w_i f_i(x). \tag{4}$$

The generalization error $E_i(x)$ of the ith component neural network on x and the generalization error $\widehat{E}(x)$ of the ensemble on x are respectively:

$$E_i(x) = \big(f_i(x) - d(x)\big)^2, \tag{5}$$

$$\widehat{E}(x) = \big(\hat{f}(x) - d(x)\big)^2. \tag{6}$$

Then the generalization error of the ith component neural network and that of the ensemble, i.e., E_i and $\widehat{E}$, on the distribution $p(x)$ are respectively:

$$E_i = \int \mathrm{d}x\, p(x) E_i(x), \tag{7}$$

$$\widehat{E} = \int \mathrm{d}x\, p(x) \widehat{E}(x). \tag{8}$$

Now we define the *correlation* between the ith and the jth component neural networks as:

$$C_{ij} = \int \mathrm{d}x\, p(x)\big(f_i(x) - d(x)\big)\big(f_j(x) - d(x)\big). \tag{9}$$

It is obvious that C_{ij} satisfies both Eqs. (10) and (11):

$$C_{ii} = E_i, \tag{10}$$

$$C_{ij} = C_{ji}. \tag{11}$$

Considering Eqs. (4) and (6) we get:

$$\widehat{E}(x) = \left(\sum_{i=1}^{N} w_i f_i(x) - d(x)\right)\left(\sum_{j=1}^{N} w_j f_j(x) - d(x)\right). \tag{12}$$

Then considering Eqs. (8), (9), and (12) we get:

$$\widehat{E} = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j C_{ij}. \tag{13}$$

For convenience of discussion, here we assume that all the component neural networks have equal weights, i.e., $w_i = 1/N$ ($i = 1, 2, \ldots, N$). In other words, here we assume that the component predictions are combined via simple averaging. Then Eq. (13) becomes:

$$\widehat{E} = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}/N^2. \tag{14}$$

Now suppose that the kth component neural network is excluded from the ensemble. Then the generalization error of the new ensemble is:

$$\widehat{E}' = \sum_{\substack{i=1\\ i\neq k}}^{N}\sum_{\substack{j=1\\ j\neq k}}^{N} C_{ij}/(N-1)^2. \tag{15}$$

From Eqs. (14) and (15) we can derive that if Eq. (16) is satisfied then $\widehat{E}$ is not smaller than $\widehat{E}'$, which means that the ensemble excluding the kth component neural network is better than the one including the kth component neural network

$$\widehat{E} \leqslant \left(2\sum_{\substack{i=1\\ i\neq k}}^{N} C_{ik} + E_k\right)\Big/(2N-1). \tag{16}$$

Then considering Eq. (16) along with Eq. (14), we get the constraint on the kth component neural network that should be excluded from the ensemble:

$$(2N-1)\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} \leqslant 2N^2\sum_{\substack{i=1\\ i\neq k}}^{N} C_{ik} + N^2 E_k. \tag{17}$$

It is obvious that there are cases where Eq. (17) is satisfied. For an extreme example, when all the component neural networks are the duplication of the same neural network, Eq. (17) indicates that the size of the ensemble can be reduced without sacrificing the generalization ability.

Now we reach the conclusion that in the context of regression, when a number of neural networks are available, ensembling many of them may be better than ensembling all of them, and the networks that should be excluded from the ensemble satisfy Eq. (17).

2.2. Classification

Suppose the task is to use an ensemble comprising N component neural networks to approximate a function $f : \mathbb{R}^m \to \mathcal{L}$ where $\mathcal{L}$ is the set of class labels, and the predictions of the component networks are combined through majority voting where each component network votes for a class and the class label receiving the most number of votes is regarded as the output of the ensemble. For convenience of discussion, here we assume that $\mathcal{L}$ contains only two class labels, i.e., the function to be approximated is $f : \mathbb{R}^m \to \{-1, +1\}$.[1]

[1] The set of two class labels are often denoted as $\{0, 1\}$. However, using $\{-1, +1\}$ here is more helpful for following derivation.

But note that the following derivation can also be generalized to situations where $\mathcal{L}$ contains more than two class labels.

Now suppose there are m instances, the expected output, i.e., D, on those instances is $[d_1, d_2, \ldots, d_m]^{\mathrm{T}}$ where d_j denotes the expected output on the jth instance, and the actual output of the ith component neural network, i.e., f_i, on those instances is $[f_{i1}, f_{i2}, \ldots, f_{im}]^{\mathrm{T}}$ where f_{ij} denotes the actual output of the ith component network on the jth instance. D and f_i satisfy that $d_j \in \{-1, +1\}$ $(j = 1, 2, \ldots, m)$ and $f_{ij} \in \{-1, +1\}$ $(i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, m)$ respectively. It is obvious that if the actual output of the ith component network on the jth instance is correct according to the expected output then $f_{ij}d_j = +1$, otherwise $f_{ij}d_j = -1$. Thus the generalization error of the ith component neural network on those m instances is:

$$E_i = \frac{1}{m}\sum_{j=1}^{m} \mathit{Error}(f_{ij}d_j), \tag{18}$$

where $\mathit{Error}(x)$ is a function defined as:

$$\mathit{Error}(x) = \begin{cases} 1 & \text{if } x = -1, \\ 0.5 & \text{if } x = 0, \\ 0 & \text{if } x = 1. \end{cases} \tag{19}$$

Now we introduce a vector Sum as $[\mathit{Sum}_1, \mathit{Sum}_2, \ldots, \mathit{Sum}_m]^{\mathrm{T}}$ where Sum_j denotes the sum of the actual output of all the component neural networks on the jth instance,[2] i.e.,

$$\mathit{Sum}_j = \sum_{i=1}^{N} f_{ij}. \tag{20}$$

Then the output of the neural network ensemble on the jth instance is:

$$\hat{f}_j = \mathit{Sgn}(\mathit{Sum}_j), \tag{21}$$

where $\mathit{Sgn}(x)$ is a function defined as:

$$\mathit{Sgn}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases} \tag{22}$$

It is obvious that $\hat{f}_j \in \{-1, 0, +1\}$ $(j = 1, 2, \ldots, m)$. If the actual output of the ensemble on the jth instance is correct according to the expected output then $\hat{f}_j d_j = +1$; if it is wrong then $\hat{f}_j d_j = -1$; otherwise $\hat{f}_j d_j = 0$, which means that there is a tie on the jth instance, e.g., three component networks vote for $+1$ while other three networks vote for -1. Thus the generalization error of the ensemble is:

$$\widehat{E} = \frac{1}{m}\sum_{j=1}^{m} \mathit{Error}(\hat{f}_j d_j). \tag{23}$$

[2] Here the class labels, i.e., -1 and $+1$, are regarded as integers, which is the profit of using $\{-1, +1\}$ instead $\{0, 1\}$ in denoting the class labels.

Now suppose that the kth component neural network is excluded from the ensemble. Then the output of the new ensemble on the jth instance is:

$$\hat{f}'_j = Sgn(Sum_j - f_{kj}) \tag{24}$$

and the generalization error of the new ensemble is:

$$\widehat{E}' = \frac{1}{m}\sum_{j=1}^{m} Error(\hat{f}'_j d_j). \tag{25}$$

From Eqs. (23) and (25) we can derive that if Eq. (26) is satisfied then $\widehat{E}$ is not smaller than $\widehat{E}'$, which means that the ensemble excluding the kth component neural network is better than the one including the kth component neural network.

$$\sum_{j=1}^{m}\left\{Error\big(Sgn(Sum_j)d_j\big) - Error\big(Sgn(Sum_j - f_{kj})d_j\big)\right\} \geqslant 0. \tag{26}$$

Then considering that the exclusion of the kth component neural network won't impact the output of the ensemble on the jth instance where $|Sum_j| > 1$, and considering the properties of the combination of the functions $Error(x)$ and $Sgn(x)$ when $x \in \{-1, 0, +1\}$ and $y \in \{-1, +1\}$, i.e.,

$$Error\big(Sgn(x)\big) - Error\big(Sgn(x - y)\big) = -\frac{1}{2}Sgn(x + y) \tag{27}$$

we get the constraint on the kth component neural network that should be excluded from the ensemble:

$$\sum_{\substack{j=1 \\ j\in\{j||Sum_j|\leqslant 1\}}}^{m} Sgn\big((Sum_j + f_{kj})d_j\big) \leqslant 0. \tag{28}$$

It is obvious that there are cases where Eq. (28) is satisfied. For an extreme example, when all the component neural networks are the duplication of the same neural network, Eq. (28) indicates that the size of the ensemble can be reduced without sacrificing the generalization ability.

Now we reach the conclusion that in the context of classification, when a number of neural networks are available, ensembling many of them may be better than ensembling all of them, and the networks that should be excluded from the ensemble satisfy Eq. (28).

3. Selective ensemble of neural networks

In Section 2 we have proved that ensembling *many* of the available neural networks may be better than ensembling *all* of those networks in both regression and classification, and the networks that should not be included in the ensemble satisfy Eqs. (17) and (28) respectively. However, excluding those "bad" neural networks from the ensembles is not an easy task as we may have imagined.

Let's look around Eqs. (17) and (28) again. It is obvious that even with the assumptions such as there is only one output variable in regression and there are only two class labels in classification, the computational cost required by those equations for identifying the neural networks that should not join the ensembles is still too extensive to be met in real-world applications.

In this section we present a practical approach, i.e., GASEN, to find out the neural networks that should be excluded from the ensemble. The basic idea of this approach is a heuristics, i.e., assuming each neural network can be assigned a weight that could characterize the fitness of including this network in the ensemble, then the networks whose weight is bigger than a pre-set threshold λ could be selected to join the ensemble.

Here we explain the motivation of GASEN from the context of regression. Suppose the weight of the ith component neural network is w_i, which satisfies both Eqs. (1) and (2). Then we get a weight vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)$. Since the optimum weights should minimize the generalization error of the ensemble, considering Eq. (13), the optimum weight vector $\boldsymbol{w}_{opt}$ can be expressed as:

$$\boldsymbol{w}_{opt} = \arg\min_{\boldsymbol{w}} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j C_{ij} \right). \tag{29}$$

$\boldsymbol{w}_{opt.k}$, i.e., the kth ($k = 1, 2, \ldots, N$) variable of $\boldsymbol{w}_{opt}$, can be solved by *Lagrange multiplier*, which satisfies:

$$\frac{\partial(\sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j C_{ij} - 2\lambda(\sum_{i=1}^{N} w_i - 1))}{\partial \boldsymbol{w}_{opt.k}} = 0. \tag{30}$$

Eq. (30) can be simplified to:

$$\sum_{\substack{j=1 \\ j \neq k}}^{N} \boldsymbol{w}_{opt.k} C_{kj} = \lambda. \tag{31}$$

Considering that $\boldsymbol{w}_{opt.k}$ satisfies Eq. (2), we get:

$$\boldsymbol{w}_{opt.k} = \frac{\sum_{j=1}^{N} C_{kj}^{-1}}{\sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij}^{-1}}. \tag{32}$$

It seems that we can solve $\boldsymbol{w}_{opt}$ from Eq. (32). But in fact, this equation rarely works well in real-world applications. This is because when a number of neural networks are available, there are often some networks that are quite similar in performance, which makes the correlation matrix $(C_{ij})_{N \times N}$ be an irreversible or ill-conditioned matrix so that Eq. (32) cannot be solved.

However, although we cannot solve the optimum weights of the neural networks directly, we can try to approximate them in some way. Look at Eq. (29) again, we may find that it could be viewed as defining an optimization problem. Considering that genetic algorithm has been shown as a powerful optimization tool [15], GASEN is developed. GASEN assigns a random weight to each of the available neural networks at first. Then it employs genetic algorithm to evolve those weights so that they can characterize to some

extent the fitness of the neural networks in joining the ensemble. Finally it selects the networks whose weight is bigger than a pre-set threshold λ to make up the ensemble. It is worth noting that if every evolved weight is bigger than λ, then all the available neural networks will join the ensemble. We believe that this corresponds to the situation where all the component networks satisfy neither Eq. (17) in regression nor Eq. (28) in classification.

Note that GASEN can be applied to not only regression but also classification because the aim of the evolving of the weights is only to select the component neural networks. In particular, the component predictions for regression are combined via simple averaging instead of weighted averaging. This is because we believe that using the weights both in the selection of the component neural networks and in the combination of the component predictions is easy to suffer overfitting, which is supported by experiments described in Section 4.

Here GASEN is realized by utilizing the standard genetic algorithm [15] and a floating coding scheme that represents each weight in 64 bits. Thus each individual in the evolving population is coded in $8N$ bytes where N is the number of the available neural networks. Note that GASEN can also be realized by employing other kinds of genetic algorithms and coding schemes. In each generation of the evolution, the weights are normalized so that they can compare with the pre-set threshold λ. Currently GASEN uses a quite simple normalization scheme, i.e.,

$$w_i' = w_i / \sum_{i=1}^{N} w_i. \tag{33}$$

In order to evaluate the goodness of the individuals in the evolving population, a validation data set bootstrap sampled from the training set is used. Let $\widehat{E}_{\boldsymbol{w}}^{V}$ denote the estimated generalization error of the ensemble corresponding to the individual $\boldsymbol{w}$ on the validation set V. It is obvious that $\widehat{E}_{\boldsymbol{w}}^{V}$ can express the goodness of $\boldsymbol{w}$, i.e., the smaller $\widehat{E}_{\boldsymbol{w}}^{V}$ is, the better $\boldsymbol{w}$ is. So, GASEN uses $f(\boldsymbol{w}) = 1/\widehat{E}_{\boldsymbol{w}}^{V}$ as the fitness function.

It is worth mentioning that with the help of Eq. (14), $\widehat{E}_{\boldsymbol{w}}^{V}$ can be evaluated efficiently for regression tasks. But since we have not such kind of intermediate result in the derivation presented in Section 2.2, the evaluation of $\widehat{E}_{\boldsymbol{w}}^{V}$ for classification tasks is relatively time-consuming.

The GASEN approach is summarized in Fig. 1, where T bootstrap samples $S_1, S_2, \ldots, S_T$ are generated from the original training set and a component neural network N_t is trained from each S_t, an ensemble N^* is built from $N_1, N_2, \ldots, N_T$ whose output is the average output of the component networks in regression, or the class label received the most number of votes in classification.

4. Empirical study

In order to know how well GASEN works, a large empirical study is performed. This section briefly introduces the approaches used to compare with GASEN, then presents the information on the data sets, then describes the experimental methodology, and finally reports on the experimental results.

Input: training set S, learner L, trials T, threshold λ

Procedure:
1. for $t = 1$ to T {
2. $S_t =$ bootstrap sample from S
3. $N_t = L(S_t)$
4. }
5. generate a population of weight vectors
6. evolve the population where the fitness of a weight vector $\boldsymbol{w}$ is measured as $f(\boldsymbol{w}) = 1/E_{\boldsymbol{w}}^{V}$
7. $\boldsymbol{w}^* =$ the evolved best weight vector

Output: ensemble N^*

$$N^*(x) = \text{Ave} \sum_{\boldsymbol{w}_t^* > \lambda} N_t(x) \qquad \text{for regression}$$

$$N^*(x) = \arg\max_{y \in Y} \sum_{\boldsymbol{w}_t^* > \lambda:\ N_t(x) = y} 1 \qquad \text{for classification}$$

Fig. 1. The GASEN approach.

4.1. Bagging and Boosting

In our experiments, GASEN is compared with two prevailing ensemble approaches, i.e., Bagging and Boosting.

The Bagging algorithm [3] employs bootstrap sampling [10] to generate many training sets from the original training set, and then trains a neural network from each of those training sets. The component predictions are combined via simple averaging for regression tasks and majority voting for classification tasks. In classification tasks, ties are broken arbitrarily.

The Boosting algorithms used for classification and regression are AdaBoost [12] and AdaBoost.R2 [8] respectively. Both algorithms sequentially generate a series of neural networks, where the training instances that are wrongly predicted by the previous neural networks will play more important role in the training of later networks. The component predictions are combined via weighted averaging for regression tasks and weighted voting for classification tasks, where the weights are determined by the algorithms themselves. Note that there are two ways, i.e., resampling [13] and reweighting [36], in determining the training sets used in Boosting. In our experiments resampling is employed because neural networks cannot explicitly support weighted instances. Moreover, it is worth mention that Boosting requires that a *weak learning* algorithm whose error is bounded by a constant strictly less than 0.5. In practice, this requirement cannot be guaranteed especially when dealing with multiclass tasks. In our experiments, instead of aborting the learning process when the error bound is breached, we generate a bootstrap sample from the original training set and continue up to a limit of 20 such samples at a given trial. Such an option has been adopted by Bauer and Kohavi [1] before.

4.2. Data sets

Twenty big data sets are used in our experiments, each of which contains at least 1000 instances. Among those data sets, ten are used for regression while the remains are used for classification.

Table 1
Data sets used for regression

Data set	Function	Variable	Size
2-d Mexican Hat	$y = \sin c\|x\| = \frac{\sin\|x\|}{\|x\|}$	$x \sim \boldsymbol{U}[-2\pi, 2\pi]$	5000
3-d Mexican Hat	$y = \sin c\sqrt{x_1^2 + x_2^2} = \frac{\sin\sqrt{x_1^2+x_2^2}}{\sqrt{x_1^2+x_2^2}}$	$x \sim \boldsymbol{U}[-4\pi, 4\pi]$	3000
Friedman #1	$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$	$x_i \sim \boldsymbol{U}[0, 1]$	5000
Friedman #2	$y = \sqrt{x_1^2 + (x_2 x_3 - (\frac{1}{x_2 x_4}))^2}$	$x_1 \sim \boldsymbol{U}[0, 100]$ $x_2 \sim \boldsymbol{U}[40\pi, 560\pi]$ $x_3 \sim \boldsymbol{U}[0, 1]$ $x_4 \sim \boldsymbol{U}[1, 11]$	5000
Friedman #3	$y = \tan^{-1}\frac{x_2 x_3 - \frac{1}{x_2 x_4}}{x_1}$	$x_1 \sim \boldsymbol{U}[0, 100]$ $x_2 \sim \boldsymbol{U}[40\pi, 560\pi]$ $x_3 \sim \boldsymbol{U}[0, 1]$ $x_4 \sim \boldsymbol{U}[1, 11]$	3000
Gabor	$y = \frac{1}{2}\pi\exp[-2(x_1^2 + x_2^2)]\cos[2\pi(x_1 + x_2)]$	$x_i \sim \boldsymbol{U}[0, 1]$	3000
Multi	$y = 0.79 + 1.27x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5$	$x_i \sim \boldsymbol{U}[0, 1]$	4000
Plane	$y = 0.6x_1 + 0.3x_2$	$x_i \sim \boldsymbol{U}[0, 1]$	1000
Polynomial	$y = 1 + 2x + 3x^2 + 4x^3 + 5x^4$	$x_i \sim \boldsymbol{U}[0, 1]$	3000
SinC	$y = \frac{\sin(x)}{x}$	$x \sim \boldsymbol{U}[0, 2\pi]$	3000

The information on the data sets used for regression is tabulated in Table 1. *2-d Mexican Hat* and *3-d Mexican Hat* have been used by Weston et al. [44] in investigating the performance of support vector machines. *Friedman #1*, *Friedman #2*, and *Friedman #3* have been used by Breiman [3] in testing the performance of Bagging. *Gabor*, *Multi*, and *SinC* have been used by Hansen [18] in comparing several ensemble approaches. *Plane* has been used by Ridgeway et al. [37] in exploring the performance of boosted naive Bayesian regressors.

In our experiments, the instances contained in those data sets are generated from the functions listed in Table 1. The constraints on the variables are also shown in Table 1, where "$\boldsymbol{U}[x, y]$" means a uniform distribution over the interval determined by x and y. Note that in our experiments some noise terms have been added to the functions, but we have not shown them in Table 1 because the focus of our experiments is on the relative performance instead of the absolute performance of the compared approaches.

All the data sets used for classification are from UCI machine learning repository [2], which has been extensively used in testing the performance of diversified kinds of classifiers. Here the data sets are selected according to the criterion that after the removal of instances with missing values, each data set should contain at least 1,000 instances.

The *Credit (German)* we used is the numerical version donated by Strathclyde University. In *Image segmentation*, a constant attribute is removed. In *Allbp* and *Sick*, seven useless nominal attributes are removed. In *Hypothyroid* and *Sick-euthyroid*, six useless nominal attributes are removed. Besides, in *Allbp*, *Sick*, *Hypothyroid*, and *Sick-euthyroid*, a

Table 2
Data sets used for classification

Data set	Class	Attribute		Size
		Nominal	Continuous	
Allbp	3	15	6	2643
Chess	2	36	0	3196
Credit (German)	2	0	24	1000
Hypothyroid	2	12	6	2000
Image segmentation	7	0	18	2310
LED-7	10	7	0	2000
LED-24	10	24	0	1000
Sick	2	15	6	2643
Sick-euthyroid	2	12	6	2000
Waveform-40	3	0	40	5000

continuous attribute that has a great number of missing values is removed. The information on the data sets used in our experiments is tabulated in Table 2.

4.3. Experimental methodology

In our experiments, 10-fold cross validation is performed on each data set, where ten neural network ensembles are trained by each compared approach in each fold. For Bagging and Boosting, each ensemble contains twenty neural networks. But for GASEN, the component networks are selected from twenty neural networks, that is, the number of networks in an ensemble generated by GASEN is far less than twenty.

The training sets of the ensembles are bootstrap sampled from the training set of the fold. In order to increase the diversity of those ensembles, the size of their training sets is roughly half of that of the fold. For example, for a data set with 1,000 instances, the training set of each fold comprises 900 instances, and each of the training sets of the ensembles contains 450 instances that are bootstrap sampled from those 900 instances. The training sets of the neural networks used to constitute the ensembles are bootstrap sampled from the training set of the ensembles. Such a methodology is helpful in estimating the bias and variance [14] of the ensemble approaches, which will be described in Section 5.

Here the genetic algorithm employed by GASEN is realized by the GAOT toolbox developed by Houck et al. [21]. The genetic operators, including select, crossover, and mutation, and the system parameters, including the crossover probability, the mutation probability, and the stopping criterion, are all set to the default values of GAOT. The pre-set threshold λ used by GASEN is set to 0.05. The validation set used by GASEN is bootstrap sampled from its training set.

The neural networks in the ensembles are trained by the implementation of Backpropagation algorithm [38] in MATLAB [7]. Each network has one hidden layer that comprises five hidden units. The parameters such as the learning rate are set to the default values of MATLAB. Here we do not optimize the architecture and the parameters of those networks because we care the relative performance of the compared ensemble approaches instead of their absolute performance. During the training process, the generalization error of each network is estimated in each epoch on a validation set. If the error does not change in five

consecutive epochs, the training of the network is terminated in order to avoid overfitting. The validation set used by a neural network is bootstrap sampled from its training set.

In order to know how well the compared ensemble approaches work, i.e., how significant the generalization ability is improved by utilizing those ensemble approaches, in our experiments we also test the performance of single neural networks. For each data set, in each fold, ten single neural networks are trained. The training sets, the architecture, the parameters, and the training process of those neural networks are all crafted in the same way as that of the networks used in ensembles.

4.4. Results

The result of an approach in each fold is the average result of ten learning systems (ensembles or single neural networks) generated by the approach in the fold, and the reported result is the average result of ten folds, i.e., the 10-fold cross validation results. For regression tasks, the error is measured as mean squared error on test instances. For classification tasks, the error is measured as the number of the test instances correctly predicted divided by the number of the test instances.

The comparison results on regression and classification are shown in Figs. 2 and 3 respectively. Note that since we care relative performance instead of absolute performance, the error of Bagging, Boosting, and GASEN has been normalized according to that of the single neural networks. In other words, the error of single neural networks is regarded as 1.0, and the reported error of Bagging, Boosting, and GASEN is in fact the ratio against the error of the single neural networks. Moreover, in each of those two figures there is a subfigure titled "*average*" which shows the average relative error of the compared approaches on all those regression/classification tasks.

Fig. 2 shows that all the three ensemble approaches are consistently better than single neural networks in regression. Pairwise two-tailed t-tests indicate that GASEN is significantly better than both Bagging and Boosting in most regression tasks, i.e., *2-d Mexican Hat*, *Friedman #1*, *Friedman #2*, *Gabor*, *Multi*, *Polynomial*, and *SinC*. As for the remaining three tasks, in *Friedman #3* and *Plane* all the three ensemble approaches obtain similar performance, in *3-d Mexican Hat* GASEN is better than Bagging but worse than Boosting. Note that in half of those ten tasks, i.e., *2-d Mexican Hat*, *Friedman #2*, *Gabor*, *Polynomial*, and *SinC*, the performance of GASEN is so good that the relative error is reduced to the degree close to zero. So, we believe that GASEN is better than both Bagging and Boosting when utilized in regression, which is supported by the subfigure titled "*average*" in Fig. 2.

Fig. 3 shows that GASEN is consistently better than single neural networks in classification. Moreover, pairwise two-tailed t-tests indicate that GASEN is significantly better than both Bagging and Boosting in half tasks, i.e., *Chess*, *Credit* (*German*), *Hypothyroid*, *Sick*, and *Sick-euthyroid*. As for the remaining tasks, in *LED-7* all the three ensemble approaches obtain similar performance, in *Image segmentation* and *Waveform-40*, GASEN is far better than Boosting but comparable to Bagging, in *LED-24* GASEN is far better than Boosting but slightly worse than Bagging, and in *Allbp* GASEN is worse than Boosting but comparable to Bagging. So, we believe that GASEN is better than both

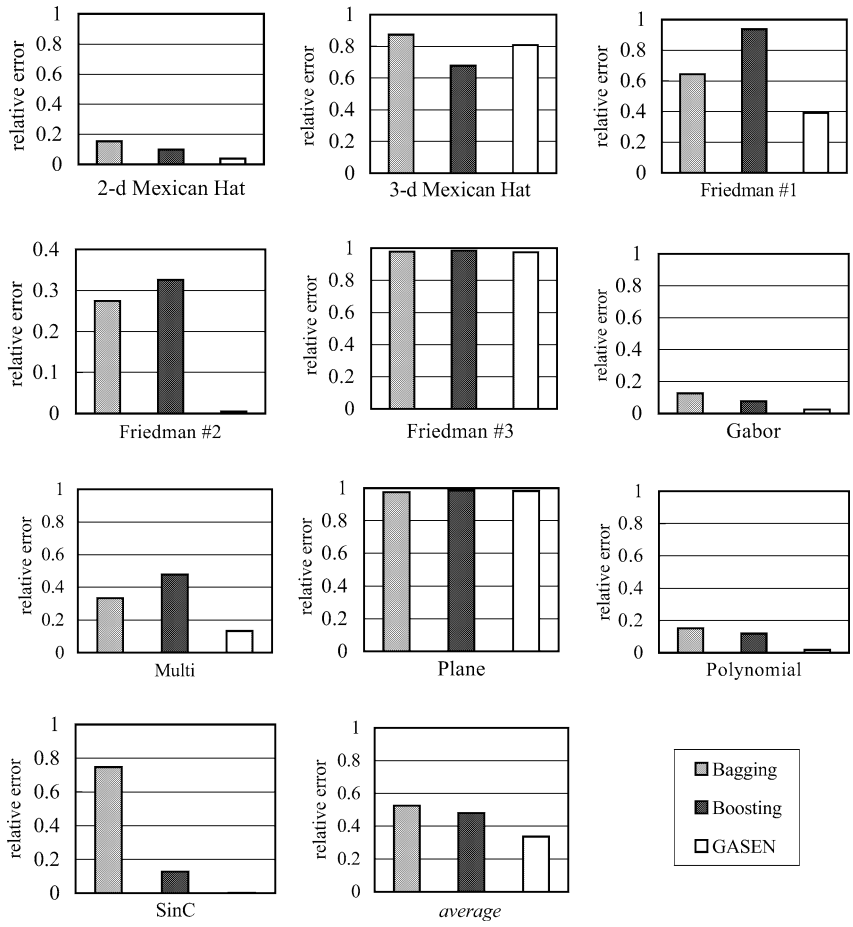

Fig. 2. Comparison of the relative error of Bagging, Boosting, and GASEN on regression tasks.

Bagging and Boosting when utilized in classification, which is supported by the subfigure titled "*average*" in Fig. 3.

In summary, Figs. 2 and 3 show that GASEN is superior to both Bagging and Boosting in both regression and classification, which strongly supports our theory formally proved in Section 2 that it may be a better choice to ensemble *many* instead of *all* neural networks at hand.

Figs. 2 and 3 also show that Bagging is consistently better than a single neural network in both regression and classification, but the performance of Boosting is not so stable. There are tasks such as *3-d Mexican Hat* and *Allbp* where Boosting obtains the best performance, but there are also tasks such as *Credit (German)*, *LED-24*, and *Waveform-40* where the performance of Boosting is even worse than that of single neural networks. Such observation is accordant with those reported in previous works [1,32].

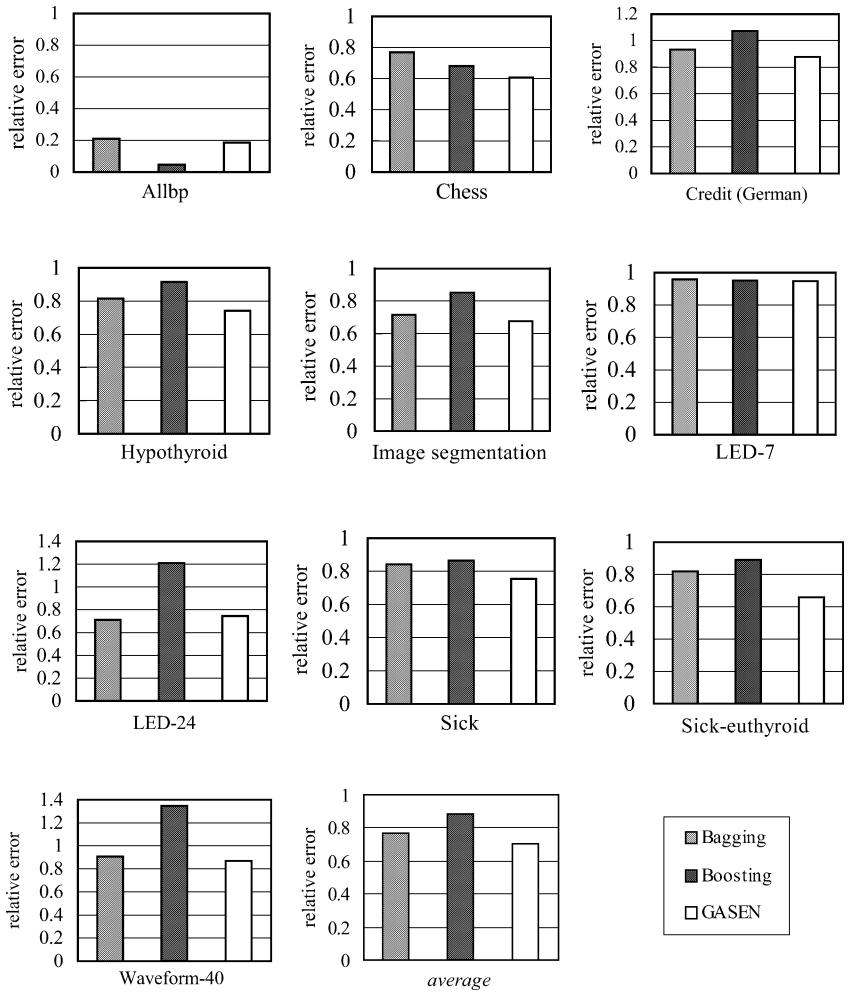

Fig. 3. Comparison of the relative error of Bagging, Boosting, and GASEN on classification tasks.

We also compare GASEN with its two variants on those twenty data sets with 10-fold cross validation. The first variant is GASEN-w that uses the evolved weights to select the component neural networks but combines the predictions of the selected networks with the normalized version of their evolved weights. In other words, weighted averaging or weighted voting is used instead of simple averaging or majority voting for combining the predictions of the selected networks. The second variant is GASEN-wa that also uses genetic algorithm to evolve the weights but does not select the component neural networks according to the evolved weights. In other words, all the available neural networks are kept in the ensembles and their predictions are combined via weighted averaging or weighted voting with the normalized version of their evolved weights. Note that the computational cost of GASEN-w and GASEN-wa is similar to that of

Table 3
Comparison of the relative error of GASEN, GASEN-w, and GASEN-wa on regression tasks

Data set	GASEN	GASEN-w	GASEN-wa	Num. of networks used by GASEN
2-d Mexican Hat	0.038	0.038	0.035	3.82
3-d Mexican Hat	0.809	0.808	0.804	5.20
Friedman #1	0.390	0.392	0.387	3.42
Friedman #2	0.005	0.005	0.005	2.07
Friedman #3	0.974	0.973	0.973	4.82
Gabor	0.025	0.027	0.028	4.10
Multi	0.131	0.127	0.129	4.50
Plane	0.982	0.982	0.981	4.32
Polynomial	0.016	0.013	0.014	2.57
SinC	0.001	0.001	0.001	2.29
Average	0.337	0.337	0.336	3.71

Table 4
Comparison of the relative error of GASEN, GASEN-w, and GASEN-wa on classification tasks

Data set	GASEN	GASEN-w	GASEN-wa	Num. of networks used by GASEN
Allbp	0.186	0.418	0.210	4.76
Chess	0.607	0.705	0.597	5.83
Credit (German)	0.878	0.949	0.876	7.78
Hypothyroid	0.741	0.886	0.759	6.08
Image segmentation	0.676	0.764	0.665	7.55
LED-7	0.947	0.984	0.943	8.27
LED-24	0.745	0.771	0.739	10.66
Sick	0.751	0.877	0.755	5.92
Sick-euthyroid	0.659	0.781	0.652	5.36
Waveform-40	0.871	0.927	0.870	8.76
Average	0.706	0.806	0.707	7.10

GASEN because the main difference of those approaches only lies in the utilization of the evolved weights. The comparison results on regression and classification are shown in Tables 3 and 4 respectively. Note that since we care relative performance instead of absolute performance, the error of GASEN, GASEN-w, and GASEN-wa has been normalized according to that of the single neural networks. It is also worth mention that each ensemble generated by GASEN-wa contains twenty component neural networks, but each ensemble generated GASEN-w contains the same number of component networks as that generated by GASEN, which is far less than twenty. The average number of component neural networks used by GASEN in constituting an ensemble is also shown in Tables 3 and 4.

Pairwise two-tailed t-tests indicate that GASEN is significantly better than GASEN-w on almost all the classification data sets. We believe that this is because using the evolved weights both in the selection of the component neural networks and the combination of the component predictions is easy to suffer overfitting. There is no significant difference

between GASEN and GASEN-w in regression. We believe that this is because the regression data sets we used are artificially generated while most of the classification data sets we used are from real-world tasks, which leads to that the noise in the regression data sets is far less than that in the classification data sets. So, overfitting is easier to happen to the classification data sets than the regression data sets in our experiments.

Pairwise two-tailed t-tests also indicate that there is no significant difference between the generalization ability of the ensembles generated by GASEN and those generated by GASEN-wa. We believe that this is because GASEN-wa does not use the evolved weights to select component neural networks, overfitting may not be so serious as in GASEN-w. But since the size of the ensembles generated by GASEN is about only 19% (3.71/20.0) in classification and 36% (7.10/20.0) in regression of the size of the ensembles generated by GASEN-wa, and those two approaches are with similar computational cost, we believe that GASEN is better than GASEN-wa.

5. Bias-variance decomposition

In order to explore the reason of the success of GASEN, the bias-variance decomposition is employed to analyze the empirical results of Bagging, Boosting, and GASEN. This section briefly introduces the bias-variance decomposition and then presents the decomposition results.

5.1. Bias and variance

The bias-variance decomposition [14] is a powerful tool for investigating the working mechanism of learning approaches. Given a learning target and the size of training set, it breaks the expected error of a learning approach into the sum of three non-negative quantities, i.e., the intrinsic noise, the bias, and the variance. The intrinsic noise is a lower bound on the expected error of any learning approach on the target. The bias measures how closely the average estimate of the learning approach is able to approximate the target. The variance measures how much the estimate of the learning approach fluctuates for the different training sets of the same size.

At present there are several kinds of bias-variance decomposition schemes [4,26,27]. Here we adopt the one proposed by Kohavi and Wolpert [26]. Let Y_H be the random variable representing the label of an instance in the hypothesis space, and Y_F be the random variable representing the label of an instance in the target. Then the bias and the variance are expressed as Eqs. (34) and (35) respectively

$$\text{bias}_x^2 = \frac{1}{2}\sum_{y\in Y}\left[P(Y_F = y \mid x) - P(Y_H = y \mid x)\right]^2, \tag{34}$$

$$\text{variance}_x = \frac{1}{2}\left(1 - \sum_{y\in Y} P(Y_H = y \mid x)^2\right). \tag{35}$$

According to Kohavi and Wolpert [26], for estimating the bias and variance of a learning approach, the original data set is split into two parts, that is, D and E. Then, N training

sets are sampled from D, whose size is roughly half of that of D to guarantee that there are not many duplicate training sets in those N training sets even for small D. After that, the learning approach is ran on each of those training sets and the bias and variance are estimated with Eqs. (34) and (35). The whole process can be repeated several times to improve the estimates.

Since it is difficult to estimate the intrinsic noise in practice, the actual bias-variance decomposition scheme of Kohavi and Wolpert [26] generates a bias term that includes the intrinsic noise. Therefore the bias plus the variance should be equal to the average error. However, if an ensemble approach employs majority voting in classification, then the sum of the bias and the variance generated by such a decomposition scheme may not be strictly equal to the average error. Nevertheless, this is no a serious problem in our scenarios because such a problem also occurs in some other bias-variance decomposition schemes [4] and the generated bias and variance are still useful in exploring the reason of the success of GASEN.

5.2. Results

With the experimental methodology described in Section 4.3, it is easy for us to estimate the bias and variance of the compared approaches according to Kohavi and Wolpert [26]'s decomposition scheme. In detail, in our experiments, 90% data of the original data set is used as the original training set while the remaining 10% data is used as the test set. From the original training set, ten training sets whose size is roughly half of the original training set are sampled. Then, the ensemble approaches are ran on each of those ten training sets and their bias and variance are estimated with Eqs. (34) and (35). Such a process is repeated for ten times to improve the estimates.

The bias of the compared ensemble approaches on regression and classification are shown in Figs. 4 and 5 respectively, and the variance of them are shown in Figs. 6 and 7. Note that since we care relative performance instead of absolute performance, the bias/variance of Bagging, Boosting, and GASEN has been normalized according to that of single neural networks. In other words, the bias/variance of single neural networks is regarded as 1.0, and the reported bias/variance of Bagging, Boosting, and GASEN is in fact the ratio against the bias/variance of the single neural networks. Moreover, in each of those figures there is a subfigure titled "*average*" which shows the average relative bias/variance of the compared approaches on all those regression/classification tasks.

Fig. 4 shows that in most regression tasks, i.e., *2-d Mexican Hat*, *3-d Mexican Hat*, *Friedman #2*, *Gabor*, *Multi*, *Polynomial*, and *SinC*, Boosting can significantly reduce the bias, and the degree of its reduction is bigger than that of Bagging except in *Multi*. As for the remaining three tasks, in *Friedman #3* both Boosting and Bagging cannot reduce the bias, in *Friedman #1* and *Plane* Boosting even increases the bias. Therefore it seems that Boosting is better than Bagging in reducing the bias but its performance is not very stable, which is accordant with the observations reported in previous works [1,4].

Pairwise two-tailed t-tests indicate that in almost all the regression tasks except *3-d Mexican Hat*, *Friedman #3*, and *Plane*, GASEN is significantly better than Boosting in reducing the bias. In particular, GASEN's ability of reducing the bias is so good that in *Friedman #2*, *Polynomial*, and *SinC*, the relative bias is even reduced to the degree

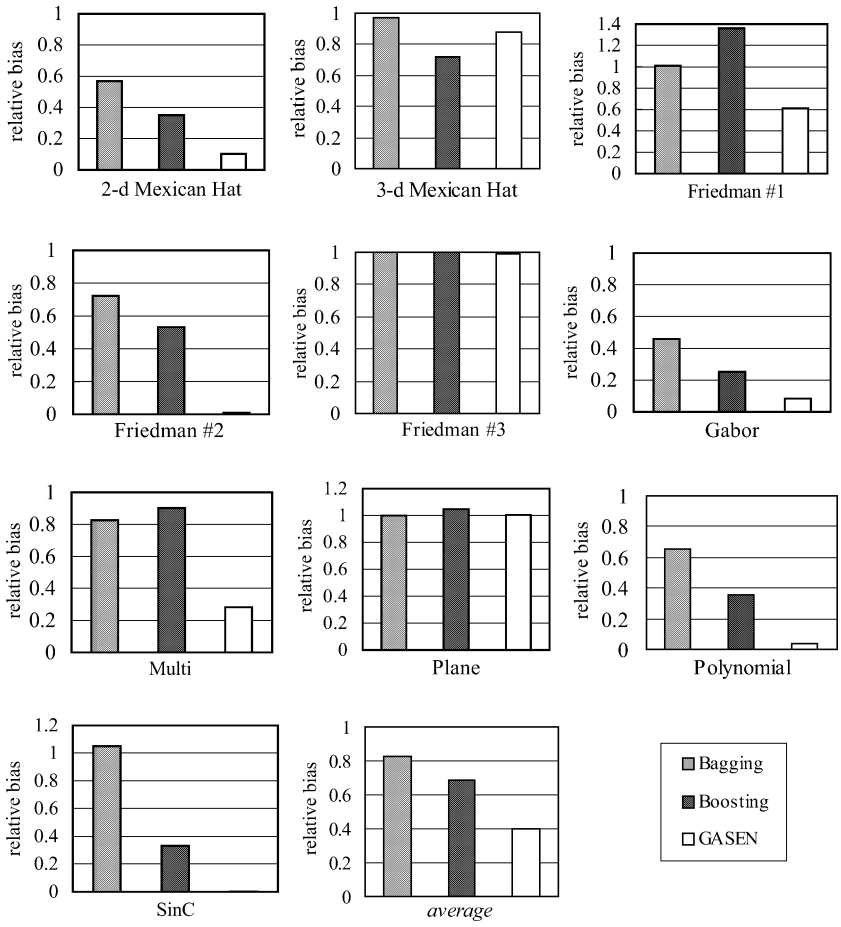

Fig. 4. Comparison of the relative bias of Bagging, Boosting, and GASEN on regression tasks.

close to zero. Therefore we believe that in regression tasks, GASEN is the best among the compared ensemble approaches in reducing the bias, which is supported by the subfigure titled "*average*" in Fig. 4.

Fig. 5 shows that in majority classification tasks, i.e., *Allbp*, *Chess*, *Hypothyroid*, *Image segmentation*, *Sick*, and *Sick-euthyroid*, Boosting can reduce the bias, but Bagging can only reduce the bias in *Allbp* and *Chess*. Moreover, when Boosting cannot reduce the bias, such as in *Credit (German)*, *LED-7*, *LED-24*, and *Waveform-40*, neither can Bagging. Therefore it seems that Boosting is more effective than Bagging in reducing the bias, which is accordant with the observations reported in previous works [1,4].

Pairwise two-tailed t-tests indicate that when Boosting can significantly reduce the bias in the classification tasks, GASEN can also do so although the degree of its reduction may not be so large as that of Boosting. Therefore we believe that in classification tasks,

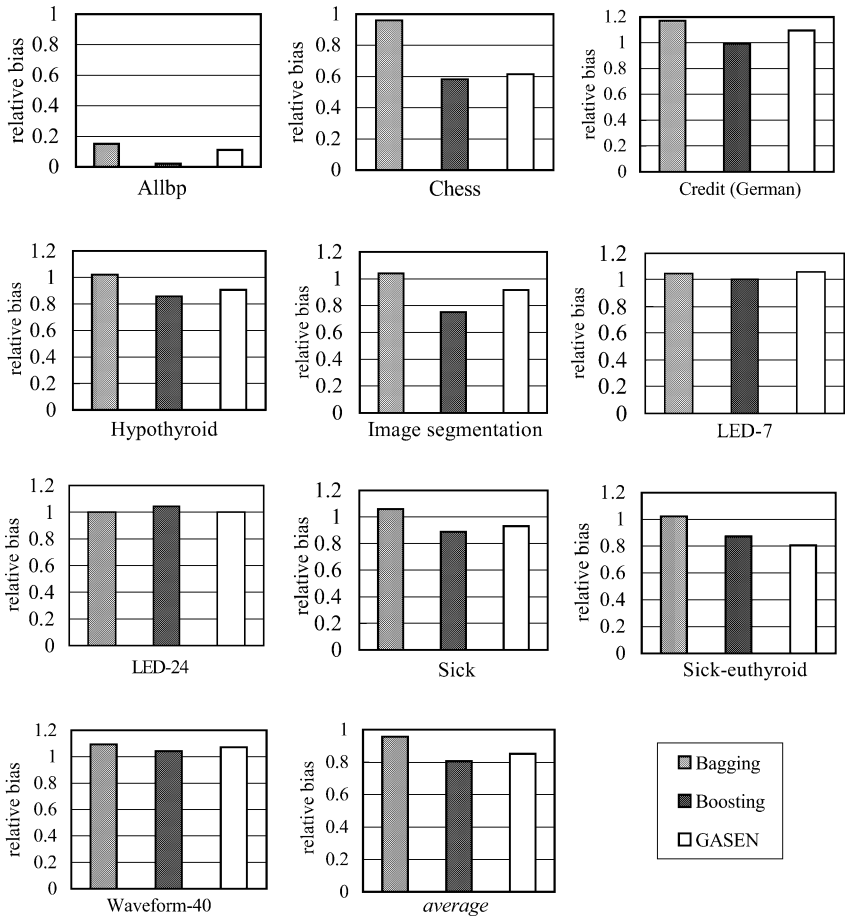

Fig. 5. Comparison of the relative bias of Bagging, Boosting, and GASEN on classification tasks.

although GASEN's ability of reducing the bias is not so good as that of Boosting, it is still better than that of Bagging, which is supported by the subfigure titled "*average*" in Fig. 5.

So, from Figs. 4 and 5, we believe that the success of GASEN may partially owe to its ability of significantly reducing the bias.

Fig. 6 shows that Bagging can significantly reduce the variance in all regression tasks, but the performance of Boosting is not so stable. There are tasks such as *2-d Mexican Hat*, *Gabor*, and *SinC* where Boosting reduces the variance more significantly than Bagging, but there are also tasks such as *Plane* where Boosting greatly increases the variance.

Pairwise two-tailed t-tests indicate that GASEN can also significantly reduce the variance in all the regression tasks. Moreover, GASEN's ability in reducing the variance is even significantly better than that of Bagging in almost half of those tasks, i.e., *Friedman #2*, *Gabor*, *Polynomial*, and *SinC*. Therefore we believe that in regression tasks, GASEN

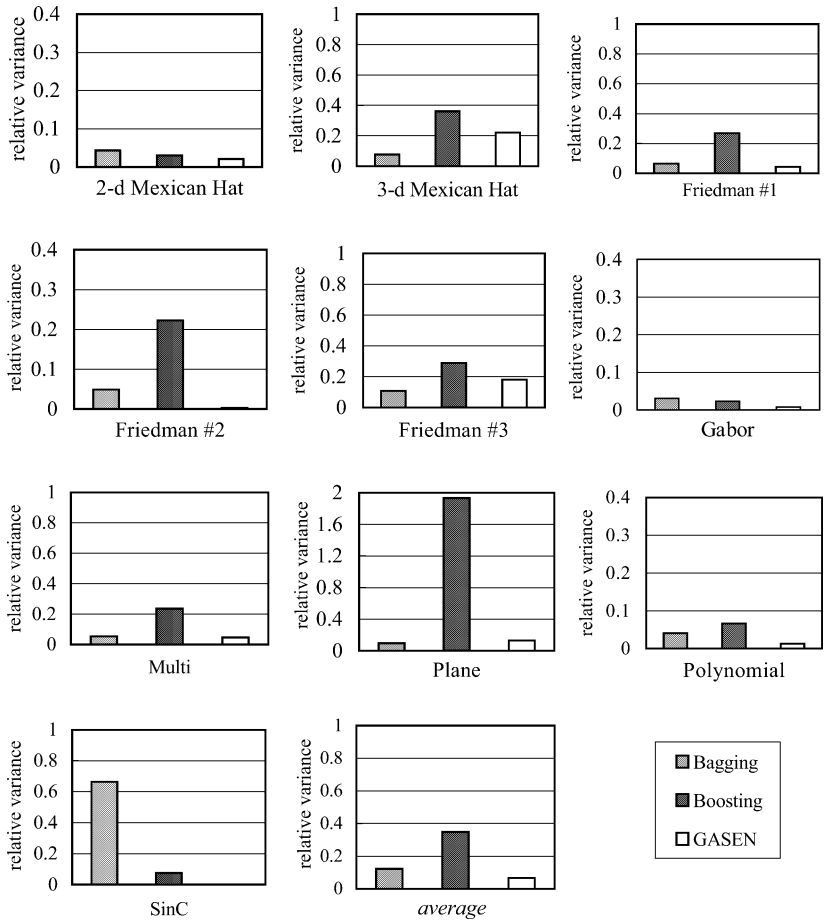

Fig. 6. Comparison of the relative variance of Bagging, Boosting, and GASEN on regression tasks.

is the best among the compared ensemble approaches in reducing the variance, which is supported by the subfigure titled "*average*" in Fig. 6.

Fig. 7 shows that Bagging can significantly reduce the variance in all classification tasks, but the performance of Boosting is not so stable. There are tasks such as *Allbp*, *Chess*, *LED-7*, and *Sick* where Boosting greatly reduces the variance, but there are also tasks such as *Credit (German)*, *LED-24*, and *Waveform-40* where Boosting greatly increases the variance.

Pairwise two-tailed t-tests indicate that when Bagging can significantly reduce the variance in the classification tasks, GASEN can also do so although the degree of its reduction may not be so large as that of Bagging. Therefore we believe that in classification tasks, although GASEN's ability of reducing the variance is not so good as that of Bagging, it is still better than that of Boosting, which is supported by the subfigure titled "*average*" in Fig. 7.

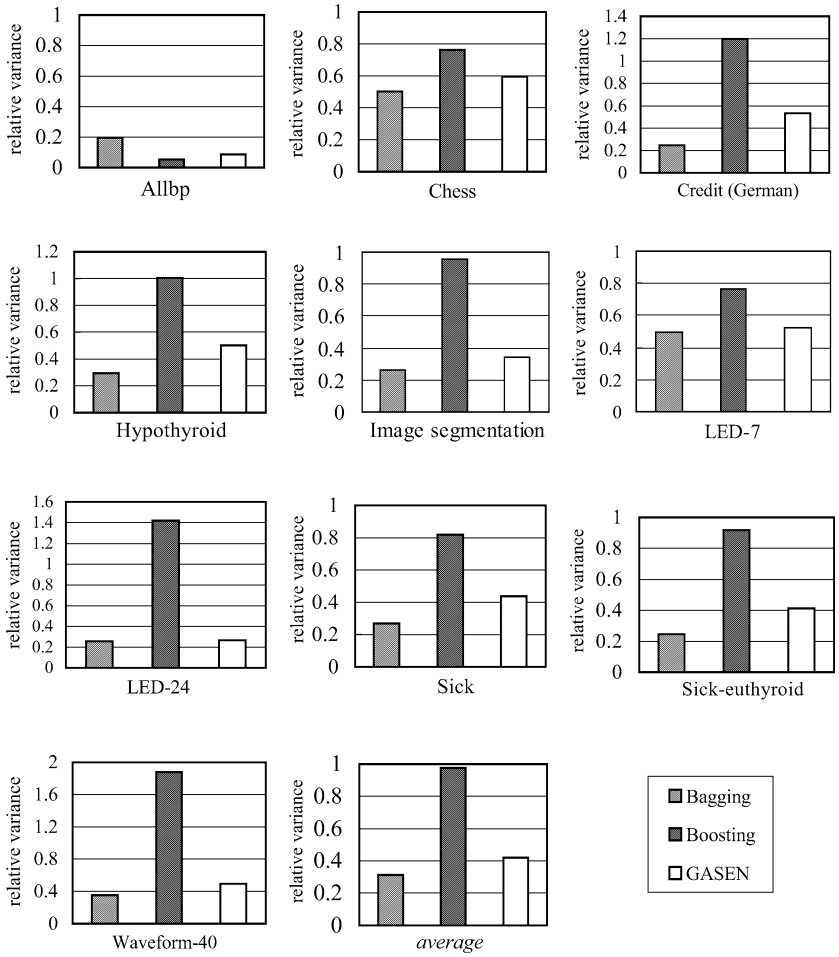

Fig. 7. Comparison of the relative variance of Bagging, Boosting, and GASEN on classification tasks.

So, from Figs. 6 and 7, we believe that the success of GASEN may partially owe to its ability of significantly reducing the variance.

In summary, from Figs. 4–7 we find that in regression tasks GASEN can do better than both Bagging and Boosting in reducing both the bias and the variance, and in classification tasks GASEN is better than Bagging in reducing the bias and is better than Boosting in reducing the variance. So, we believe that the success of GASEN may lie in that it has the ability of significantly reducing both the bias and the variance simultaneously.

We guess that GASEN can reduce the bias because it efficiently utilizes the training data in that it employs a validation set that is bootstrap sampled from the training set, and it can reduce the variance because it combines multiple versions of the same learning approach. However, those guesses should be justified by rigorous theoretical analysis.

6. Conclusions

At present, most neural network ensemble approaches utilize all the available neural networks to constitute an ensemble. However, the goodness of such a process has not yet been formally proved. In this paper, the relationship between the ensemble and its component neural networks is analyzed, which reveals that it may be a better choice to ensemble *many* instead of *all* the available neural networks. This theory may be useful in designing powerful ensemble approaches. Then, in order to show the feasibility of the theory, an ensemble approach named GASEN is presented. A large empirical study shows that GASEN is superior to both Bagging and Boosting in both regression and classification because it utilizes far less component neural networks but achieves stronger generalization ability.

Note that although GASEN has obtained impressive performance in our empirical study, we believe that there are approaches that could do better than GASEN along the way that GASEN goes, i.e., ensembling *many* instead of *all* available neural networks under certain circumstances. The reason is that GASEN has not been finely tuned because its aim is only to show the feasibility of our theory. In other words, the aim of GASEN is just to show that the networks appropriate for constituting the ensemble could be effectively selected from a collection of available neural networks. So, its performance might at least be improved through utilizing better fitness functions, coding schemes, or genetic operators. In the future we hope to use some other large-scale data sets such as NIST to test GASEN and tune its performance, and then apply it to real-world applications. Moreover, it is worth mention that finding stronger ensemble approaches based on the recognition that *many could be better than all* is an interesting issue for future works.

In order to explore the reason of the success of GASEN, the bias-variance decomposition is employed in this paper to analyze the empirical results. It seems that the success of GASEN mainly lies in that GASEN could reduce the bias as well as the variance. We guess that GASEN can reduce the bias because it efficiently utilizes the training data in that it employs a validation set bootstrap sampled from the training set, and it can reduce the variance because it combines multiple version of the same learning approach. Rigorous theoretical analysis may be necessary to justify those guesses, which is another interesting issue for future works.

Acknowledgements

The comments and suggestions from the anonymous reviewers greatly improved this paper. The authors wish to thank the AI Lab of Nanjing University for the usage of about 3000 CPU hours on SGI Z $\times$ 10s (2 CPUs, 900 MHz, 512 MB RAM) for the empirical study. The National Natural Science Foundation of P.R. China and the Natural Science Foundation of Jiangsu Province, P.R. China, supported this research.

References

[1] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, Boosting, and variants, Machine Learning 36 (1–2) (1999) 105–139.
[2] C. Blake, E. Keogh, C.J. Merz, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, CA, 1998, http://www.ics.uci.edu/~mlearn/MLRepository.htm.
[3] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[4] L. Breiman, Bias, variance, and arcing classifiers, Technical Report 460, Statistics Department, University of California, Berkeley, CA, 1996.
[5] K.J. Cherkauer, Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks, in: P. Chan, S. Stolfo, D. Wolpert (Eds.), Proc. AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, AAAI Press, Menlo Park, CA, 1996, pp. 15–21.
[6] P. Cunningham, J. Carney, S. Jacob, Stability problems with artificial neural networks and the ensemble solution, Artificial Intelligence in Medicine 20 (3) (2000) 217–225.
[7] H. Demuth, M. Beale, Neural Network Toolbox for use with MATLAB, The MathWorks, Natick, MA, 1998.
[8] H. Drucker, Boosting using neural nets, in: A. Sharkey (Ed.), Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, Springer, London, 1999, pp. 51–77.
[9] H. Drucker, R. Schapire, P. Simard, Improving performance in neural networks using a boosting algorithm, in: S.J. Hanson, J.D. Cowan, C.L. Giles (Eds.), Advances in Neural Information Processing Systems 5, Denver, CO, Morgan Kaufmann, San Mateo, CA, 1993, pp. 42–49.
[10] B. Efron, R. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, New York, 1993.
[11] Y. Freund, Boosting a weak algorithm by majority, Inform. and Comput. 121 (2) (1995) 256–285.
[12] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: Proc. EuroCOLT-94, Barcelona, Spain, Springer, Berlin, 1995, pp. 23–37.
[13] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proc. ICML-96, Bari, Italy, Morgan Kaufmann, San Mateo, CA, 1996, pp. 148–156.
[14] S. German, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Comput. 4 (1) (1992) 1–58.
[15] D.E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[16] S. Gutta, H. Wechsler, Face recognition using hybrid classifier systems, in: Proc. ICNN-96, Washington, DC, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 1017–1022.
[17] J. Hampshire, A. Waibel, A novel objective function for improved phoneme recognition using time-delay neural networks, IEEE Trans. Neural Networks 1 (2) (1990) 216–228.
[18] J.V. Hansen, Combining predictors: Meta machine learning methods and bias/variance and ambiguity decompositions, Ph.D. Dissertation, Department of Computer Science, University of Aarhus, Denmark, 2000.
[19] L.K. Hansen, L. Liisberg, P. Salamon, Ensemble methods for handwritten digit recognition, in: Proc. IEEE Workshop on Neural Networks for Signal Processing, Helsingoer, Denmark, IEEE Press, Piscataway, NJ, 1992, pp. 333–342.
[20] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Machine Intelligence 12 (10) (1990) 993–1001.
[21] C.R. Houck, J.A. Joines, M.G. Kay, A genetic algorithm for function optimization: A Matlab implementation, Technical Report NCSU-IE-TR-95-09, North Carolina State University, Raleigh, NC, 1995.
[22] F.J. Huang, Z.-H. Zhou, H.-J. Zhang, T.H. Chen, Pose invariant face recognition, in: Proc. 4th IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France, IEEE Computer Society Press, Los Alamitos, CA, 2000, pp. 245–250.
[23] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptively mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87.
[24] D. Jimenez, Dynamically weighted ensemble neural networks for classification, in: Proc. IJCNN-98, Vol. 1, Anchorage, AK, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 753–756.

[25] M.I. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Comput. 6 (2) (1994) 181–214.
[26] R. Kohavi, D.H. Wolpert, Bias plus variance decomposition for zero-one loss functions, in: Proc. ICML-96, Bari, Italy, Morgan Kaufmann, San Mateo, CA, 1996, pp. 275–283.
[27] E.B. Kong, T.G. Dietterich, Error-correcting output coding corrects bias and variance, in: Proc. ICML-95, Tahoe City, CA, Morgan Kaufmann, San Mateo, CA, 1995, pp. 313–321.
[28] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems 7, Denver, CO, MIT Press, Cambridge, MA, 1995, pp. 231–238.
[29] R. Maclin, J.W. Shavlik, Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks, in: Proc. IJCAI-95, Montreal, Quebec, Morgan Kaufmann, San Mateo, CA, 1995, pp. 524–530.
[30] J. Mao, A case study on bagging, boosting and basic ensembles of neural networks for OCR, in: Proc. IJCNN-98, Vol. 3, Anchorage, AK, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 1828–1833.
[31] C.J. Merz, M.J. Pazzani, Combining neural network regression estimates with regularized linear weights, in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems 9, Denver, CO, MIT Press, Cambridge, MA, 1997, pp. 564–570.
[32] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, J. Artificial Intelligence Res. 11 (1999) 169–198.
[33] D.W. Opitz, J.W. Shavlik, Actively searching for an effective neural network ensemble, Connection Science 8 (3–4) (1996) 337–353.
[34] D.W. Opitz, J.W. Shavlik, Generating accurate and diverse members of a neural network ensemble, in: D.S. Touretzky, M.C. Mozer, M.E. Hasselmo (Eds.), Advances in Neural Information Processing Systems 8, Denver, CO, MIT Press, Cambridge, MA, 1996, pp. 535–541.
[35] M.P. Perrone, L.N. Cooper, When networks disagree: Ensemble method for neural networks, in: R.J. Mammone (Ed.), Artificial Neural Networks for Speech and Vision, Chapman & Hall, New York, 1993, pp. 126–142.
[36] J.R. Quinlan, Bagging, Boosting, and C4.5, in: Proc. AAAI-96, Portland, OR, AAAI Press, Menlo Park, CA, 1996, pp. 725–730.
[37] G. Ridgeway, D. Madigan, T. Richardson, Boosting methodology for regression problems, in: Proc. AISTATS-99, Fort Lauderdale, FL, Morgan Kaufmann, San Mateo, CA, 1999, pp. 152–161.
[38] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.
[39] R.E. Schapire, The strength of weak learnability, Machine Learning 5 (2) (1990) 197–227.
[40] A. Sharkey (Ed.), Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, Springer, London, 1999.
[41] Y. Shimshoni, N. Intrator, Classification of seismic signals by integrating ensembles of neural networks, IEEE Trans. Signal Process. 46 (5) (1998) 1194–1201.
[42] P. Sollich, A. Krogh, Learning with ensembles: How over-fitting can be useful, in: D.S. Touretzky, M.C. Mozer, M.E. Hasselmo (Eds.), Advances in Neural Information Processing Systems 8, Denver, CO, MIT Press, Cambridge, MA, 1996, pp. 190–196.
[43] N. Ueda, Optimal linear combination of neural networks for improving classification performance, IEEE Trans. Pattern Anal. Machine Intelligence 22 (2) (2000) 207–215.
[44] J.A.E. Weston, M.O. Stitson, A. Gammerman, V. Vovk, V. Vapnik, Experiments with support vector machines, Technical Report: CSD-TR-96-19, Royal Holloway University of London, London, 1996.
[45] D.H. Wolpert, Stacked generalization, Neural Networks 5 (2) (1992) 241–259.
[46] X. Yao, Y. Liu, Making use of population information in evolutionary artificial neural networks, IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics 28 (3) (1998) 417–425.
[47] Z.-H. Zhou, Y. Jiang, Y.-B. Yang, S.-F. Chen, Lung cancer cell identification based on artificial neural network ensembles, Artificial Intelligence in Medicine 24 (1) (2002) 25–36.