Combinations of Weak Classifiers

Chuanyi Ji and Sheng Ma

***Abstract*— To obtain classification systems with both good generalization performance and efficiency in space and time, we propose a learning method based on combinations of weak classifiers, where weak classifiers are linear classifiers (perceptrons) which can do a little better than making random guesses. A randomized algorithm is proposed to find the weak classifiers. They are then combined through a majority vote. As demonstrated through systematic experiments, the method developed is able to obtain combinations of weak classifiers with good generalization performance and a fast training time on a variety of test problems and real applications. Theoretical analysis on one of the test problems investigated in our experiments provides insights on when and why the proposed method works. In particular, when the strength of weak classifiers is properly chosen, combinations of weak classifiers can achieve a good generalization performance with polynomial space- and time-complexity.**

***Index Terms*—Weak classifiers, combinations of classifiers, supervised learning, generalization error, space-complexity, time-complexity.**

I. Introduction

TWO of the most important issues for supervised learning can be specified as generalization performance and efficiency. The former addresses the problem of how to develop an adaptive pattern recognition system to achieve optimal performance on samples that are not included in a training set with a finite number of training samples. The latter deals with the complexity of a pattern recognition system in both space and time. The space complexity refers to the size of a system, and the time complexity characterizes the computational time needed to develop such systems. These two issues are interrelated.

The generalization performance of a supervised learning system is characterized by its generalization error. For the classification problems which will be the focus of this work, the generalization error of a classifier is the probability of misclassification of a random sample by this classifier. Since it is well known that the size (space-complexity) of a classifier is pertinent to the generalization performance, one of the key issues on performance is how to find the appropriate size of a pattern recognition system so that the desired generalization error can be achieved.

Tremendous efforts have been made on estimating and finding the optimal architecture of a classification system using a finite number of training samples. These approaches include computational learning theory in both the machine learning and neural-network community [4], [23], [32], [37], [41], [42] and various statistical methods including cross-validation and model selection [1], [3], [10]. Since it is very difficult to find the best architecture [18], methods are proposed to combine different architectures. Specifically, in the pattern recognition community, combinations of classifiers are proposed to improve the classification performance of a single classifier [20], [26], [40]. In machine learning, combinations of experts have shown to be able to work at least as well as the best expert in the pool of experts on predicting binary strings [8], [28]. Combinations of neural networks [19], [33], [44], [19], [38], [39], [22], [29] and statistical regressors [7] have also been used for both classification and function approximation problems. Although the generalization performance often improves when classifiers are combined, it becomes computationally costly to combine well-trained classifiers.

In terms of space-complexity, it has been found that compact classification systems like feedforward neural nets provide efficient implementations of nonlinear adaptive systems [2]. The space-complexity of neural networks when used as nonparametric adaptive systems scales nicely (polynomially) to the dimension of feature vectors. Localized models such as radial basis function networks [30], nearest neighbor classifiers [9], and Parzen windows [13] are easier to obtain, but may suffer from the "curse of dimensionality" which is an exponential scaling of their space-complexity with the dimension of feature vectors [2], [21], [35].[1] In terms of time-complexity, training feedforward neural networks is slow, and in general, is NP-complete [6], [24], but training aforementioned localized classifiers can be done quickly. This presents a dilemma which exists between performance, space-, and time-complexity.

The problem we will tackle is how to develop a classifier with both good generalization performance and efficiency in space and time in a supervised learning environment. The method we propose to tackle this problem is based on combinations of weak classifiers [36], [26], where the weak classifiers are the classifiers which can do a little better than random guessing. The motivation to use weak classifiers instead of well-trained classifiers is that they are computationally cheap and easy to obtain. If combinations of weak classifiers can also achieve good generalization performance, they may provide a feasible solution for achieving good performance and efficiency in space and time. To explore this possibility, we will develop a randomized algorithm to obtain combinations of weak classifiers. The algorithm will be tested systematically on both synthetic problems and real applications to gain experimental evidence on capabilities of combinations of weak classifiers. Then we will provide theoretical analysis on one

Manuscript received January 5, 1996; revised June 27, 1996. This work was supported by the National Science Foundation (ECS-9 312 594 and (CAREER) IRI-9 502 518).

The authors are with the Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA.

Publisher Item Identifier S 1045-9227(97)00240-3.

[1] The space-complexity of K-nearest neighbor classifiers is characterized by the number of stored training samples.

1045–9227/97$10.00 © 1997 IEEE

of the test problems used in our experiments to show when and why such methods can work.

This paper is organized as follows. In Section II, we provide related background knowledge and review closely related work. In Section III, the concept of weak classifiers is introduced. In Section IV, a learning algorithm for combinations of weak classifiers is described. Extensive experimental results are given in Section V to show the effectiveness of the proposed method. Rigorous analysis on a simple problem which has been used in our experiments is given in Section VI to provide theoretical basis for the proposed approach. Conclusions are given in Section VII.

II. Background

A. Performance and Efficiency

To formally define the performance and efficiency of a classifier, we begin by considering a set of Nlabeled training samples $(x_1, t_1), \cdots, (x_N, t_N)$, where $x_n \in R^d$are feature vectors drawn from some underlying distribution $p(x)$. $t_n$is a class label for x_n. For two-class classification problems, $t_n = 1$ if x_n belongs to Class 1; and $t_n = 0$, otherwise. For M-class classification problems $(M > 2), t_n \in \{0,1\}^M$, where the jth element of t_n is one if x_n belongs to Class j (denoted as Ω_j), and the rest of the elements of t_n are zeros, for $1 \leq j \leq M$. Let $C(x)$ denote a classifier obtained through learning the training set, where x is a new sample drawn from the same distribution. $C(x) \in \{0,1\}^M$ is the actual label assigned to x by the classifier. Let t be the true class label for x. Then the generalization error, which measures the generalization performance of $C(x)$, is defined as the probability of incorrect classification of x by the classifier $C(x)$, which is $\Pr(C(x) \neq t)$.

The space- and time-complexities of $C(x)$ are defined to be the size of $C(x)$ and the average amount of training time[2] needed to obtain $C(x)$ in terms of the dimension d of the feature vectors, and other parameters in the training algorithm. If the space-complexity and the time-complexity all scale polynomially with respect to these parameters, the classifier is said to be efficient both in space and time. Otherwise, if an exponential scaling is observed, the classifier is considered to be inefficient.

B. Previous Work

The method we develop in this work is motivated by the ideas addressed in the boosting algorithm [36] and in stochastic discriminations [26], [27]. In the boosting algorithm, Shapire [36] shows that if a classification problem is solvable in the probabilistic approximately correct (PAC) learning framework [4], [10], [42], [41], the problem can be solved through combinations of weak classifiers, where the number of weak classifiers needed is polynomial in the parameters of interest. The way to accomplish this is to force the weak classifiers to learn harder and harder parts of a problem, and then to combine those weak classifiers in a tree structure. Freund later generalizes the algorithm so that it can be applied for learning a fixed training set, and shows that a simple majority vote among weak classifiers can be used to combine weak classifiers when an optimal weighting scheme is used to select weak classifiers [17]. These results show that in principle it is possible for combinations of weak classifiers to achieve comparable performance to that of a well-trained classifier. However, the theory does not provide information on the type of weak classifiers that can be used and how to find them. The question on whether using combined weak classifiers can provide additional advantages over a well-trained classifier was not dealt with either. In practice, the ideas proposed in boosting algorithm have been applied with success in handwritten character recognition to boost the performance of an already well-trained classifier by combining three[3] such classifiers [11], [12]. But the original idea on combining a large number of weak classifiers has never been used in solving real problems.

In an independent work by Kleinberg [26], combinations of weak classifiers are also proposed to solve pattern classification problems. This work suggests that using combinations of weak classifiers does not only lead to a good generalization performance, but also provides advantages in computation time, since weak classifiers are computationally easier to obtain than well-trained classifiers. Random sampling is suggested to generate weak classifiers, which are either randomly selected subinput spaces or randomly located hyperspheres [5], [26]. However, since the proposed method is based on an assumption which is difficult to realize, discrepancies have been found between the theory and the experimental results [27]. Nevertheless, the method has been applied successfully to handwritten digit recognition [26], [27] and has shown promise in getting both a good performance and fast training time.

C. Problems To Tackle

We investigate the following open questions regarding combinations of weak classifiers.

1) How to find weak classifiers?
2) What are the performance and efficiency of combinations of weak classifiers?
3) What are the advantages of using combined weak classifiers compared with other pattern classification methods?

We will first provide answers to these questions experimentally, and then give theoretical analysis based on one of the problems investigated in our experiments.

III. Weak Classifiers

Three factors determine a set of weak classifiers: the architecture of each weak classifier, the strength of individual weak classifiers, and the computational power of a set of weak classifiers.

Intuitively, the architecture of weak classifiers should be simple enough so that they are easy to get, but not too localized so that they maintain an efficient space-complexity. The

[2] The definition will be given later.

[3] As reported in [11], combining more than three classifiers did not seem to be able to improve the performance even further due to using the tree structure.

architecture should also be general enough for the combined classifiers to be used for a variety of applications. In the present work, we start with a simple architecture and choose linear classifiers (perceptrons) as weak classifiers. The need for a more complicated architecture will be investigated in the future.

The strength of weak classifiers can be characterized by their generalization error. Let $\frac{1}{2}-1/\nu$ be the required generalization error of a classifier, where $\nu \geq 2$, is called the weakness factor which is used to characterize the strength of a classifier. The larger the ν, the weaker the weak classifier. A set of weak classifiers should satisfy the following two conditions: 1) each weak classifier should do better than random guessing and 2) the set of classifiers should have enough computational power to learn a problem. The first condition is to ensure that each weak classifier possess a minimum computational power. The second condition suggests that individual weak classifiers should learn different parts of a problem so that a collection of weak classifiers can learn an entire problem. If all the weak classifiers in a collection learned the same part of a problem, their combination would not do better than the individual classifiers.

Since the true generalization error is unknown, to implement the first condition, we can require that the training error on a training set of N samples is no bigger than $\frac{1}{2}-1/\nu-\epsilon$, where $\epsilon > 0$ is small. Then from well-known results in computational learning theory [41], [42], the generalization error is within the ϵ tolerance of the training error with high probability when N is sufficiently large.[4] The second condition can be satisfied by forcing weak classifiers to learn different parts of a training set as will be described in the algorithm given in the next section.

IV. Algorithm

We developed our algorithm for combinations of weak classifiers by combining the strengths of the boosting algorithm and stochastic discriminations and circumventing their drawbacks. In particular, the method consists of two steps: 1) generating individual weak classifiers through a simple randomized algorithm and 2) combining a collection of weak classifiers through a simple majority vote. We describe our algorithm for two-class classification problems, and discuss an extension to multiple classes at the end of this section.

Three parameters need to be chosen *a priori* for the algorithm: a weakness factor ν, a number θ $(\frac{1}{2} \leq \theta < 1)$ which will be used as a threshold to partition the training set, and the number of weak classifiers $2L+1$ to be generated, where L is a positive integer.

A. Partitioning the Training Set

The method we use to partition a training set is motivated by what given in [17]. Suppose a combined classifier consists of K $(K \geq 1)$ weak classifiers already. In order to generate a (new) weak classifier, the entire training set of N training samples is partitioned into two subsets: a set of M_1 samples which contain all the misclassified samples and a small fraction of samples correctly classified by the existing combined classifier; and the remaining $N - M_1$ training samples. The set of M_1 samples are called "cares," since they will be used to select a new weak classifier, while the rest of the samples are the "don't-cares."

The threshold θ is used to determine which samples should be assigned as "cares." For instance, for the nth training sample $(1 \leq n \leq N)$, the performance index $a(n)$ is recorded, where $a(n)$ is the fraction of the weak classifiers in the existing combined classifier which classify the nth sample correctly. If $a(n) < \theta$, this sample is assigned to the "cares." Otherwise, it is a "don't-care." This is done for all N samples.

The motivation for partitioning a training set in this way is to be able to force a newly generated weak classifier to learn the samples which have not been learned by the existing weak classifiers. In the meantime, a properly chosen θ can ensure that enough samples are used to obtain each weak classifier. For $\theta = \frac{1}{2}$, only misclassified samples are used as "cares." When $\theta = 1$, all the training samples are cares. When $\theta = 0$, all the sample will be dont-cares. Usually, θ is chosen to be between $\frac{1}{2}$ and $\frac{1}{2} + 1/\nu$. We will discuss various choices of θ in Section V.

B. Random Sampling

In order for combinations of weak classifiers to possess an efficient time-complexity, weak classifiers have to be generated in a computationally easy and cheap way. To do this, we take advantage of simple randomized methods as in [26]. The training process we consider consists of random sampling from the classifier-space which consists of all possible linear classifiers. This is different from conventional training methods which find parameters of a learning system through minimizing some cost function on a training set.

Assume that a feature vector x is distributed over a compact region D. The direction of a hyperplane characterized by a linear classifier with a weight vector, is first generated by randomly selecting the elements of the weight vector based on a uniform distribution over $(-1, 1)^d$. Then the threshold of the hyperplane is determined by randomly picking an $x \in D$, and letting the hyperplane pass through x. This will generate random hyperplanes which pass through the region D, and whose directions are randomly distributed in all directions. Such a randomly selected classifier will then be tested on all the cares. If it misclassifies a fraction of cares no more than $\frac{1}{2} - 1/\nu - \epsilon$ ($\epsilon > 0$ and small), the classifier is kept and will be used in the combination. Otherwise, it is discarded. This process is repeated until a weak classifier is finally obtained.

When and why can such a randomized method be efficient in selecting a weak classifier? If weak classifiers are weak enough, i.e., the weakness factor ν is large enough, there will be many such weak classifiers. Therefore, the chance of getting a weak classifier is high at each sampling. That is, the number of times needed to sample the classifier space until a weak classifier is accepted is small. Choosing ν turns out to be a crucial factor in determining whether the time-complexity of the algorithm is polynomial or exponential. This will be shown later in our theoretical analysis.

[4] e.g., $N = O(d_v/\epsilon^2)$, where d_v is the so-called VC-dimension and $O(\cdot)$ represents a quantity in the order of $(\cdot)$ for an upper bound.

C. Combination of Weak Classifiers

The newly selected weak classifier is incorporated into a combined classifier through a simple majority vote. The combined classifier $C_K(x)$ with K weak classifiers can be expressed as

$$C_K(x) = I\left(\sum_{m=1}^{K} I(w_m^T x) - \left\lfloor \frac{K}{2} \right\rfloor\right) \tag{1}$$

where $I(z)$ is an indicator function: $I(z) = 1$ if $z \geq 0$, and $I(z) = 0$ otherwise. w_m is the weight vector of the mth weak classifier for $1 \leq m \leq K$. $\lfloor K/2 \rfloor$ represents the largest integer no bigger than $K/2$.

Such a combined classifier can be implemented as a two layer feedforward neural network with K hard threshold hidden units. The weights between the input and hidden layer are the weights of the weak classifiers, while the weights connecting the hidden units with the output units are all equal to one. Such a network is also called a committee machine as discussed in [31].

After a newly selected weak classifier is added into the combination, the entire training set will be tested on the combined classifier to result in a new set of "cares" and "don't-cares." Then the whole process will be repeated until the total number $2L+1$ of weak classifiers are generated.

It is noted that in order to investigate the asymptotic performance and efficiency of a combined classifier, $2L+1$ is always chosen to be sufficiently large *a priori* in this work. Important issues on how to automatically decide the number of weak classifiers needed for a given problem will be investigated in future research.

D. Summary of The Algorithm

The entire algorithm can be summarized as follows.

Choose the parameters for the algorithm: ν, θ and $2L+1$.

1) Initial step: $i = 1$.[5]
2) Partition a training set into "cares" and "don't-cares" by evaluating the performance index $a(n)$ of each sample, and assigning that sample to the cares if $a(n)$ is below θ.
3) Generate a weight vector w of a weak classifier by randomly generating a direction and a threshold of the corresponding hyperplane. If a newly generated weak classifier misclassifies no more than a $\frac{1}{2} - 1/\nu - \epsilon$ fraction of "cares," accept the weak classifier; otherwise, discard it. Repeat the process until a weak classifier is accepted.
4) Incorporate the newly generated classifier into the combined classifier through majority vote.
5) $i = i + 1$. If $i = 2L+1$, stop; otherwise, go back to Step 2).

E. Extension of The Algorithm to Multiple Classes

To extend our algorithm to more than two classes, we simply convert the M-class classification problem to M two-class classification problems $(M > 2)$. For instance, suppose $M = 3$. Then the original problem will be converted into three two-class classification problems: Class 1 versus the other classes, Class 2 versus the other classes, and Class 3 versus the other classes. Our algorithm for two-class classification problems can then be applied directly.

[5] All the training samples are used as "cares" when the very first weak classifier is generated. Then Step 2) can be skipped when $i = 1$.

V. Experimental Results

Extensive simulations have been carried out on both synthetic problems and real applications using our algorithm. The goal is to test both the performance and efficiency of the proposed method. In particular, two synthetic problems are chosen to test the space- and time-complexity of our method. Real applications from standard data bases are selected to compare the generalization performance of combinations of weak classifiers (CW) with that of other methods such as K-nearest neighbor classifiers (K-NN)[6], artificial neural networks (ANN's), combinations of neural networks (CNN), and stochastic discriminations (SD).[7]

A. Synthetic Problems

Two synthetic problems were designed to test the scaling properties on the performance of combined classifiers in terms of the dimension d of feature vectors.

1) A Perceptron: This problem is designed to learn an underlying perceptron with a d-dimensional binary weight vector w_o which consists of all $(+1)$'s. Feature vectors x's are uniformly distributed in $\{-1, 1\}^d$. If an x satisfies $w_o^T x > 0, x$ belongs to Class 1 with a label $t = 1$; otherwise, x belongs to Class 2 with a label $t = 0$. Our algorithm is used to learn a set of 2000 randomly drawn samples when the dimension d of feature vectors is made to be larger and larger. That is, the size of the training set is made to be fixed, while the dimension of feature vectors is increasing. Ten different runs are conducted for each d, and the resulting classifiers are tested on another 4000 randomly drawn samples. For comparison, the k-NN classifiers also learn the same data sets. The resulting average generalization error as well as the standard deviations are given in Fig. 1.

As shown in Fig. 1, as d increases, the performance of k-NN decrease rapidly, thereby indicating the occurrence of the so-called curse-of-dimensionality as discussed in [35]. The degradation in performance is moderate with combinations of weak classifiers.

2) Two Overlapping Gaussians: To further test the scaling properties of combinations of weak classifiers, a nonlinearly separable problem is chosen from a standard database called ELENA [14], [15]. This database has been used by both neural-network and machine learning communities to test and compare algorithms.

The problem is a two-class classification problem, where the distributions of samples in both classes are multivariate Gaussians. Each dimension of the samples corresponds to an independent Gaussian random variable with zero mean, but

[6] The best result of different k is reported.

[7] We always test our method using exact data when compared with other methods. The details on the results due to other methods can be found in the related references.

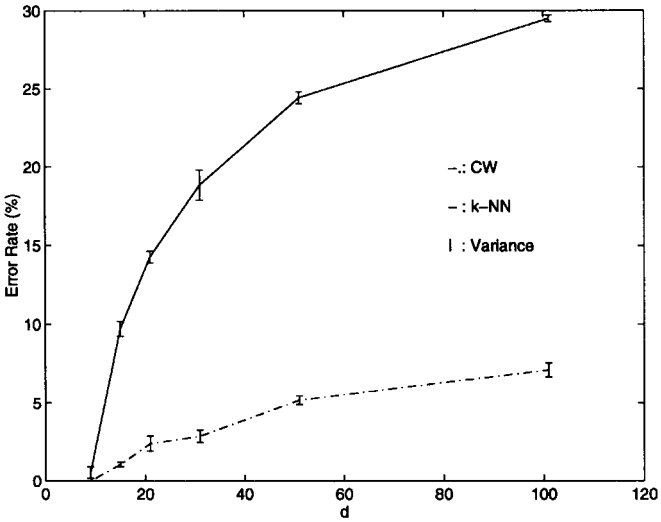

Fig. 1. Performance on learning a perceptron.

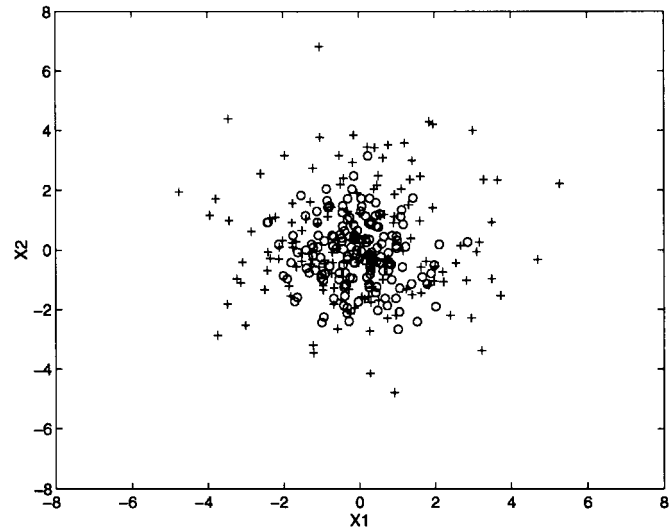

Fig. 2. Two overlapping Gaussians.

samples in different classes have different standard deviations: one for samples in Class 1 and two for samples in Class 2. As shown in Fig. 2 (for $d = 2$), there is a considerable amount of overlap between the samples in two classes, therefore the problem is nonlinearly separable. The average generalization error and the standard deviations are given in Fig. 3 for our algorithm based on 20 runs, and for other classifiers. The Bayes error is also given to show the theoretical limit.[8]

Once again, the results show that the performance of kNN degrades very quickly. The performance of ANN is better than that of kNN but still deviates more and more from the Bayes error as d gets large. The combination of weak classifiers continues to follow the trend of the Bayes error.

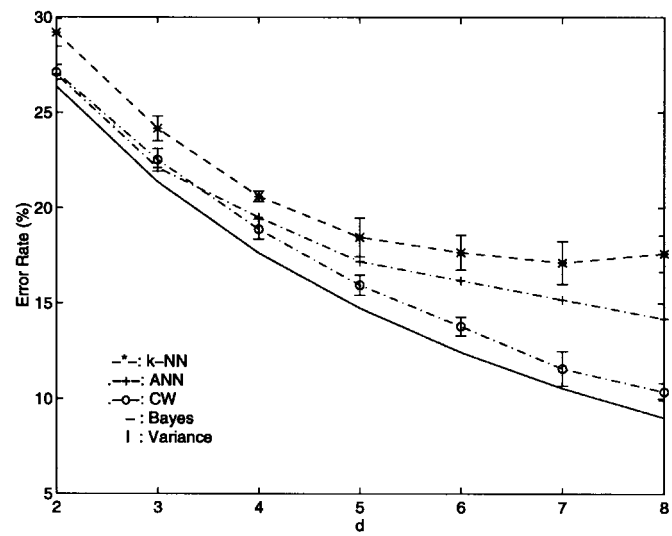

Fig. 3. Performance versus the dimension of the feature vectors.

B. Real Applications

1) Proben1 Data Sets: Three data sets, Card1, Diabetes1, and Gene1 were selected to test our algorithm from Proben1 databases which contain data sets from real applications [34].[9]

Card1 data set is for a problem on determining whether a credit-card application from a customer can be approved based on information given in 51-dimensional feature vectors. Out of 690 examples, 345 are used for training and the rest for testing. Diabetes1 data set is for determining whether diabetes is present based on eight-dimensional input patterns. Three hundred eighty-four examples are used for training and the same number of samples for testing. Gene1 data set is for deciding whether a DNA sequence is from a donor, an acceptor or neither from 120-dimensional binary feature vectors. Out of total of 3175 samples, 1588 were used for training and the rest for testing.

The average generalization error as well as the standard deviations are reported in Table I. The results from combinations of weak classifiers are based on 25 runs. The results of neural networks and combinations of well-trained neural networks are from [34] and [38].[10] As demonstrated by the results, combinations of weak classifiers have been able to achieve the generalization performance comparable to or better than that of combinations of well-trained neural networks.

TABLE I
PERFORMANCE ON CARD1 DIABETES1 AND GENE1. σ: STANDARD DEVIATION

Algorithms	Card1	Diabetes1	Gene1
	(%) Error /σ	(%) Error /σ	(%) Error /σ
Combined Weak Classifiers	11.3/ 0.85	22.70 / 0.70	11.80 / 0.52
k Nearest Neighbor	15.67	25.8	22.87
Neural Networks	13.64/ 0.85	23.52/ 0.72	13.47/ 0.44
Combined Neural Networks	13.02/0.33	22.79 /0.57	12.08 / 0.23

[8] The results of ANN and the Bayes error are from what reported in ELENA [14]. For ANN's, a one hidden lay feedforward neural network with 10 or 20 hidden nodes was trained by backpropagation (BP) algorithm.

[9] Available by anonymous ftp from ftp.ira.uka.de, as/pub/papers/techreports/1994/1994-21.ps.z

[10] In [34] and [38], neural networks with many different architectures were trained by BP and several its variations. In [38], three to seven well-trained neural networks are combined by majority vote. Only the best reported results are listed in Table I.

TABLE II
PERFORMANCE ON HANDWRITTEN DIGIT RECOGNITION

Algorithms	(%) Error/σ
Combined Weak Classifiers	4.23 / 0.1
k Nearest Neighbor	4.84
Neural Networks	5.33
Stochastic Discriminations	3.92

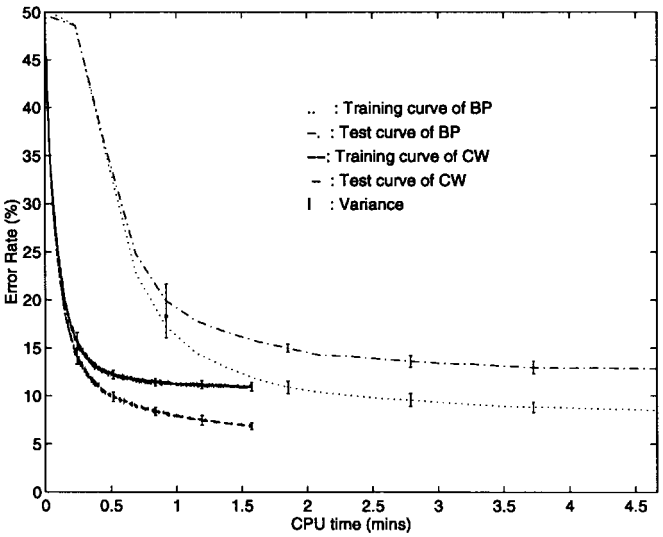

Fig. 4. Performance versus the number of weak classifiers for different ν. nu: ν.

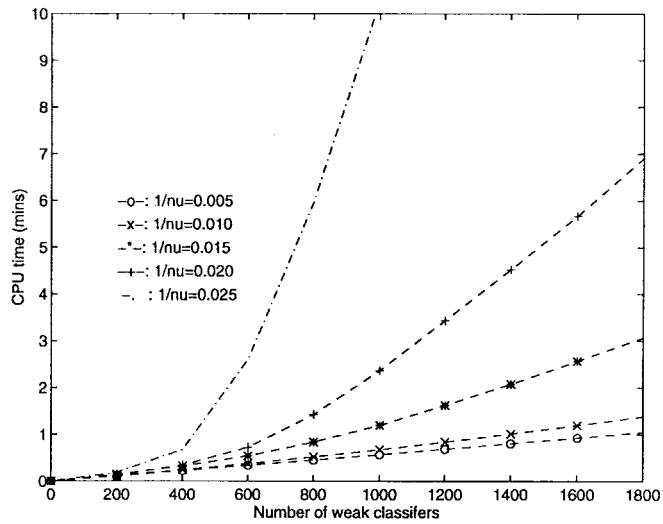

Fig. 5. Training time versus the number of weak classifiers for different ν.

TABLE III
PARAMETERS USED IN OUR EXPERIMENTS

Parameters	Perceptron	Gaussians	Card1	Diabetes1	Gene1	Digits
$1/2+1/\nu$	0.51	0.51	0.51	0.51	0.55	0.54
θ	0.51	0.51	0.51	0.54	0.54	0.53
2L+1	2000	2000	1000	1000	4000	20000
Average Tries	2∼10	2	3	7	4	2

2) Handwritten Digit Recognition: Handwritten digit recognition is chosen to test our algorithm, since one of the previously developed method on combinations of weak classifiers (stochastic discrimination [26]) was applied to this problem. For the purpose of comparison, the same set of data as used in [26] (from the NIST data base) is utilized to train and to test our algorithm. The data set contains 10 000 digits written by different people. Each digit is represented by 16 by 16 black and white pixels. The first 4997 digits are used to form a training set, and the rest are for testing. Performance of our algorithm, k-NN, neural networks, and stochastic discriminations are given in Table II. The results for our methods are based on five runs, while the results for the other methods are from [26].

The results show that the performance of our algorithm is slightly worse (by 0.3%) than that of stochastic discriminations, which uses a different method for multiclass classification by converting an M-class classification problem into $M(M-1)/2$ two-class classification problems [26].

C. Effects of Parameters

There are two parameters used in our algorithm: the threshold θ and the weakness factor ν. For most of the experiments described in this work, good performance can be obtained if θ is chosen to satisfy $\frac{1}{2} \leq \theta \leq \frac{1}{2} + 1/\nu$. When the data is noisy, a slightly larger θ can be chosen to incorporate more samples into a set of cares. From our experience, the performance of a combined classifier is not very sensitive to the choice of θ as long as it is chosen within these guidelines.

The choice of weakness factor ν, strongly affects the size and training time of a combined classifier. Experiments on the problem of two eight-dimensional overlapping Gaussians given in Section V-A2 are done to test the effects of ν. The performance and the average training time (central processing unit or CPU-time on Sun Spac-10) of combined weak classifiers based on 10 runs are given for different ν's in Figs. 4 and 5, respectively.[11] The results indicate as ν increases an individual weak classifier is obtained more quickly, but more weak classifiers are needed to achieve good performance.

A record of the parameters used in all the experiments on real applications are provided in Table III. The average tries, which are the average number of times needed to sample the classifier space to obtain an acceptable weak classifier, are also given in the table to characterize the training time for these problems.

D. Training Time

To compare learning time with off-line BP,[12] feedforward two-layer neural network with ten sigmoidal hidden units are trained by gradient-descent to learn the problem on the two eight-dimensional overlapping Gaussians. There were 2500

[11] θ is chosen to be $\frac{1}{2} + 1/\nu$.

[12] Since our algorithm requires off-line learning, off-line BP is used to make comparison. However, it should be mentioned that on-line BP can be faster than the off-line BP.

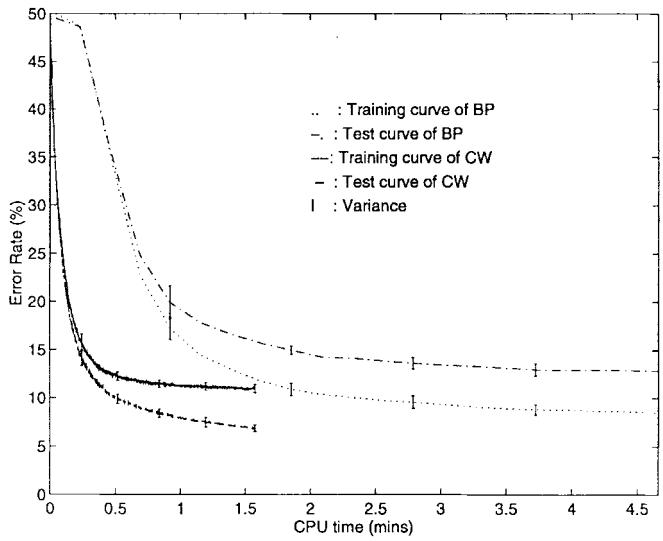

Fig. 6. Performance versus CPU time.

training samples used. The performance versus CPU time[13] are plotted for both our algorithm and BP in Fig. 6. For our algorithm, 2000 weak classifiers are combined. For BP, 1000 epoches are used. The figure shows that our algorithm is much faster than the BP algorithm. Moreover, when several well-trained neural networks are combined to achieve a better performance, the cost on training time will be even higher. For instance, in Card1, Diabetes1, and Gene1 problems, seven well trained neural networks are combined to obtain an almost comparable performance to that of combinations of weak classifiers (see Table I). Then the cost on training time would be seven times that spent on training a single neural network. Therefore, compared to combinations of well-trained neural networks, combining weak classifiers is computationally much cheaper.

E. Discussions of Experimental Results

What have the experimental results shown us? First, we observe that our algorithm for combinations of weak classifiers has been able to achieve the best performance on all the synthetic and real problems except for handwritten digit recognition problem. More specifically, the performance of the combined weak classifiers is comparable or even better than combinations of well-trained classifiers, and out-performs individual neural network classifiers and k-NN classifiers. In the meantime whereas the k-NN classifiers suffer from the curse of dimensionality, a nice scaling property in terms of the dimension of feature vectors has been observed for combined weak classifiers.

Another important observation obtained from the experiments is that the weakness factor directly impacts the size of a combined classifier and the training time.

As the experimental results provide positive results, they also pose the following questions for us to answer:

1) how to characterize the performance and efficiency of a combined classifier?
2) how to choose the weakness factor ν?

[13] Both algorithms are run on a Sun Sparc-10 sun workstation

Theoretical analysis will be carried out to provide answers to these questions. The analysis will be based on the same problem of learning a perceptron given in Section V-A1.[14] The insights gained can shed light on how to analyze our algorithm for more general problems in the future.

VI. Theoretical Analysis

A. The Results

Our main theoretical results can be summarized as follows. A bound for the generalization error of a combined classifier is obtained. It provides a polynomial rate at which the generalization error of the combined classifier approaches zero when more and more weak classifiers are used. A polynomial space-complexity of the combined classifier is derived directly from the bound. In evaluating the time-complexity, it is discovered that the choice of the weakness factor is critical. When the weakness factor is chosen to be no smaller than a quantity in the order of $\sqrt{d/\ln d}$, the time-complexity is polynomial for the combined classifier, otherwise it is exponential. The price paid to accomplish this though, is the corresponding space-complexity, which can not be smaller than $O(d \ln d)$.

We will describe our analytical results next, and details on the derivations can be found in the Appendixes.

B. A Condition for Weak Classifiers

The generalization error of a given classifier with binary weights w can be evaluated as

$$\Pr(I(w^T x) \neq t|w) = \frac{\displaystyle\sum_{k=0}^{d/2} \sum_{j=0}^{(k/2)-(d-2d_h/4)} \binom{d_h}{j}\binom{d-d_h}{k-j}}{\displaystyle\sum_{j=0}^{d/2}\binom{d}{j}} \tag{2}$$

where x is a randomly drawn binary feature vector with a class label t, and d_h is the Hamming distance between w and the perceptron w_0 given in Section V-A1. Let the desired generalization error of the weak classifiers be $\frac{1}{2}-1/\nu$. Then by setting $\Pr(I(w^T x) \neq t|w) = \frac{1}{2}-1/\nu$ for ν large[15], a condition can be obtained to characterize a set of weak classifiers

$$d_h = \frac{d}{2}\left(1 - \frac{1}{\nu'}\right) \tag{3}$$

where $\nu' = O(\nu)$. The total number of weak classifiers satisfying this condition is $\binom{d}{(\frac{1}{2}-(1/\nu'))d}$. For d and ν' large, by the Stirling formula, this number is approximately $\sqrt{2/\pi}(2^d/\sqrt{d})e^{-(d/\nu'^2)}$. Therefore, there are many such weak classifiers.

[14] Since many methods with good performance and efficiency can be used to learn a binary perceptron [43], our intention is not to investigate the problem of how to learn a perceptron itself but to use it as an example for analyzing combinations of weak classifiers.

[15] To simplify the analysis, we only consider weak classifiers with the classification error equal to $\frac{1}{2} - 1/\nu$.

C. A Bound on Generalization Error

To analyze the performance of a combined classifier, we first assume that the weak classifiers used in a combination all have the required generalization error.[16] When $2L+1$ such weak classifiers are combined, the generalization error of a combined classifier can be bounded through a theorem given below.

Theorem 1: Assume $2L+1$ weak classifiers are randomly drawn (with replacement) from a set of weak classifiers which satisfy the condition given in (3). For $2L+1$ large and $L \gg \nu$, the generalization error, $\Pr(C_{2L+1}(x)t<0)$ of a combined classifier can be bounded as

$$\Pr(C_{2L+1}(x)t<0) \leq O\left(\frac{\nu' \ln(2L+1)}{\sqrt{2L+1}}\right). \tag{4}$$

The proof of the theorem can be found in Appendix A. This result suggests that the performance of a combined classifier improves at a rate of at least $O(\nu' \ln(2L+1)/\sqrt{2L+1})$ when the number of weak classifier increases. This polynomial rate explains why a good performance as well as a nice scaling property were observed in our experiments.

D. Space-Complexity

The space-complexity of a combined classifier is defined as the number of weak classifiers needed to achieve a given generalization error ϵ_g, where $\epsilon_g > 0$ and is small. By setting the upper bound to be equal to ϵ_g, the space-complexity S of a combined classifier can be easily obtained as

$$S \leq O\left(\frac{(\nu' \ln \nu')^2}{\epsilon_g^2}\right). \tag{5}$$

Therefore, S is polynomial in both ν' and ϵ_g.

E. A Critical Weakness Factor and Time-Complexity

How large should ν be? A critical value of ν can be obtained through evaluating the time-complexity of a combined classifier. The time-complexity T of a combined classifier is the average number of samplings needed to obtain a combined classifier with a desired generalization error ϵ. Let $\overline{K}$ be the average number of times to sample the classifier space until a weak classifier is obtained. Then $T = S\overline{K}$. An expression of T is given in Theorem 2 below.

Theorem 2: For d and ν large but $\nu \ll d$

$$T = O\left(\frac{(\nu' \ln \nu')^2}{\epsilon_g^2}\sqrt{d}e^{d/\nu'^2}\right). \tag{6}$$

The proofs can be found in Appendix B. Therefore, when $\sqrt{d/\ln d} \leq O(\nu')$, the time complexity T is polynomial in the dimension d of feature vectors; otherwise, T is an exponential function of d. In the meantime, when this condition is satisfied, the space-complexity $S = O(d \ \ln \ d/\epsilon_g^2)$, is also polynomial in d.

[16] This is equivalent to assuming that there are an infinite number of cares used to select a weak classifiers. The effect of finite samples will be discussed later.

The existence of the critical value for ν' explains what observed in experiments, and provides a guideline on how to choose the weakness factor.

F. Tradeoffs Between Performance, Space-, and Time-Complexity

Since the larger the weakness factor ν, the larger the space-complexity S, but the smaller the time-complexity T, a tradeoff can be made between the space- and time-complexity by finding an optimal ν'. Specifically, let $dT/d\nu = 0$ and assume d large, an optimal ν'_o can be obtained as $\nu'_o = O(\sqrt{d})$. Since the corresponding space-complexity is $O(d \ \ln^2 \ d/\epsilon_g^2)$, the polynomial time-complexity is obtained at a cost of a larger size classifier compared to that of a well-trained classifier. The cost, however, is theoretically tolerable, since it is polynomial in the dimension d of feature vectors.

G. Effect of Finite Samples

When the number of training sample is finite and equal to N, the generalization given in (4) should be replaced by the probability given N training samples. If the training samples are assumed to be randomly drawn, commonly used methods as in [1] and [10] can be used to show that the corresponding inequality to what is given by (4) will be true with a probability at least $1-\eta$, where $\eta = (2L+1)e^{-2\epsilon^2 N}$, and ϵ is a bound for the difference between the error rate on a training set and the true generalization error[17] for each individual weak classifier. Then for $0 < \epsilon \leq 1/\nu$ and $\nu = O(\sqrt{d})$, the number of samples needed to make η small should satisfy the relation $d \ \ln(2L+1) \ll N$. Then N is polynomial in d and $2L+1$. More elaborate analysis on the effect of finite samples will be done in future research.

VII. Conclusions

We have developed a training method for combinations of weak classifiers. A randomized algorithm is proposed to find the weak classifiers. As shown in our experiments, the algorithm has been able to obtain combinations of weak classifiers with a very good generalization and fast training time on both the test problems and the real applications. The combined classifiers show a good scaling property which indicates efficiency in space-complexity. Theoretical analysis has been done on one of the test problems investigated in experiments to show when and why the proposed method works. Specifically, when the weakness factor is chosen according to the critical value given by the theory, the combinations of weak classifiers can achieve a good generalization performance with polynomial space- and time-complexity. The price paid to achieve this is a larger space-complexity compared to that of a well-trained classifier, but such a large space-complexity is tolerable, since it is polynomial in the dimension of feature vectors.

Areas for future research include analysis of the algorithm for nonlinear classification problems, and investigation of better methods for selecting weak classifiers.

[17] Here we can assume that all the training samples are used as cares to select weak classifiers.

APPENDIX A
PROOF OF THEOREM 1

Proof: The generalization error satisfies $\Pr(C_{2L+1}(x) \neq t) = \Pr(C_{2L+1}(x) < 0 | x \in \Omega_1)$ for this problem if the prior probabilities are assumed to be equal, i.e., $\Pr(\Omega_1) = \Pr(\Omega_2)$. The probability $\Pr(C_{2L+1}(x) < 0 | x \in \Omega_1)$ can be rewritten as

$$\begin{aligned}&\Pr(C_{2L+1}(x) < 0 | x \in \Omega_1)\\ &\quad = \sum_x \Pr(C_{2L+1}(x) < 0 | x, x \in \Omega_1) p(x)\end{aligned} \tag{7}$$

where $\Pr(C_{2L+1}(x) < 0 | x, x \in \Omega_1)$ is the conditional probability that $C_{2L+1}(x)$ misclassifies a given x, and $p(x) = 2^{-d}$ is the probability of x.

Assume a given $x (x \in \Omega_1)$ contains $i(-1)$'s, i.e., $\Sigma_{j=1}^{d} x_j = d - 2i$, for $0 \leq i \leq d/2$. Let w be a randomly drawn weak classifier with binary weights with a Hamming distance d_h satisfying (13). Let

$$P_w(i) = \Pr\left(w^T x < 0 | \sum_{j=1}^{d} x_j = d - 2i, 0 \leq i \leq \frac{d}{2} \right)$$

which is the probability that w misclassifies x. Assuming each classifier is drawn with an equal probability, $P_w(i)$ can be obtained as

$$P_w(i) = \frac{1}{A} \sum_{d_i=0}^{(i/2)-(d-2d_h/4)} \binom{i}{d_i} \binom{d-i}{d_h - d_i} \tag{8}$$

where $A = \binom{d}{d_h}$ is the total number of weak classifiers, and

$$\sum_{d_i=0}^{(i/2)-(d-2d_h/4)} \binom{i}{d_i} \binom{d-i}{d_h - d_i}$$

is the total number of weak classifiers which misclassifies x. Since randomly drawn weak classifiers mis-classify a given x independently, we have

$$\begin{aligned}&\Pr(C_{2L+1}(x) < 0 | x, x \in \Omega_1)\\ &\quad = \Pr\left(C_{2L+1}(x) < 0 | \sum_{j=1}^{d} x_j = d - 2i, 0 \leq i \leq \frac{d}{2} \right)\\ &\quad = \sum_{l=0}^{L} \binom{2L+1}{l} (1 - P_w(i))^l [P_w(i)]^{2L+1-l}.\end{aligned} \tag{9}$$

Putting this equation into (7), we can obtain

$$\begin{aligned}&\Pr(C_{2L+1}(x) < 0 | x \in \Omega_1)\\ &\quad = 2^{-d} \sum_{i=0}^{d/2} \binom{d}{i} \sum_{l=0}^{L} \binom{2L+1}{l} (1 - P_w(i))^l\\ &\qquad \cdot [P_w(i)]^{2L+1-l}.\end{aligned} \tag{10}$$

Let Δ be a quantity which satisfies $\Delta \ll d$. Then the summation can be divided into two terms, T_1 and T_2, where

$$\begin{aligned}T_1 = &2^{-d} \sum_{i=0}^{(d/2)-\Delta} \binom{d}{i} \sum_{l=0}^{L} \binom{2L+1}{l}\\ &\cdot (1 - P_w(i))^l [P_w(i)]^{2L+1-l}\end{aligned} \tag{11}$$

and

$$\begin{aligned}T_2 = &2^{-d} \sum_{(d/2)-\Delta}^{d/2} \binom{d}{i} \sum_{l=0}^{L} \binom{2L+1}{l}\\ &\cdot (1 - P_w(i))^l [P_w(i)]^{2L+1-l}.\end{aligned} \tag{12}$$

Since $\max_{0 \leq i \leq (d/2)-\Delta} P_w(i) = P_w(d/2 - \Delta)$, $P_w(d/2 - \Delta)$ can be obtained as

$$\begin{aligned}P_w\left(\frac{d}{2} - \Delta\right) = &\sum_{d_i=0}^{(d/4)-(\Delta/2)-(d-2d_h/4)}\\ &\cdot \frac{\binom{\frac{d}{2} - \Delta}{d_i} \binom{\frac{d}{2} + \Delta}{d_h - d_i}}{\binom{d}{d_h}}\\ \approx &\frac{1}{2} - \sqrt{\frac{2}{\pi}} \frac{\Delta}{\nu' \sqrt{d}}\end{aligned} \tag{13}$$

where (13) is obtained using a Gaussian approximation to the hypergeometric distribution

$$\frac{\binom{\frac{d}{2} - \Delta}{d_i} \binom{\frac{d}{2} + \Delta}{d_h - d_i}}{\binom{d}{d_h}}$$

when d and d_h are large [16]. Meanwhile

$$\begin{aligned}T_1 \leq &\sum_{l=0}^{L} \binom{2L+1}{l} \left(1 - P_w\left(\frac{d}{2} - \Delta\right)\right)^l\\ &\cdot \left[P_w\left(\frac{d}{2} - \Delta\right)\right]^{2L+1-l}.\end{aligned} \tag{14}$$

The bound can be further bounded using Chernoff bound, i.e.,

$$\begin{aligned}&\sum_{l=0}^{L} \binom{2L+1}{l} \left(1 - P_w\left(\frac{d}{2} - \Delta\right)\right)^l \left[P_w\left(\frac{d}{2} - \Delta\right)\right]^{2L+1-l}\\ &\quad \leq O(e^{-2\gamma^2(2L+1)})\end{aligned} \tag{15}$$

where $\gamma^2 = (8/\pi)(\Delta^2/\nu'^2 d)$. Then

$$T_1 \leq O(e^{-2\gamma^2(2L+1)}). \tag{16}$$

In the meantime,

$$\begin{aligned}T_2 &\leq 2^{-d} \sum_{(d/2)-\Delta}^{d/2} \binom{d}{i}\\ &\leq \sqrt{\frac{2}{\pi}} \frac{\Delta}{\sqrt{d}}.\end{aligned} \tag{17}$$

Therefore, when $2L + 1, d$ and ν large but $\nu \ll d, \Delta$ is chosen to be $\nu' \sqrt{d} / \sqrt{2L+1} \ \ln(2L+1)$. Then $T_1 + T_2 \leq O(\nu' / \sqrt{2L+1} \ \ln(2L+1))$. That is

$$\Pr(C_{2L+1}(x) < 0 | x \in \Omega_1) \leq O\left(\frac{\nu'}{\sqrt{2L+1}} \ \ln(2L+1) \right). \tag{18}$$

Q.E.D.

APPENDIX B
PROOFS OF THEOREM 2

Proof: Let K be the number of samplings needed to obtain one weak classifier. Let p_1 be the probability that a weak model is obtained at each sampling. Since each classifier is equally likely

$$p_1 = \frac{\binom{d}{\left(\frac{1}{2} - \frac{1}{\nu'}\right)d}}{2^d} \tag{19}$$

where ν' is given in (3). For d and ν' large, by the Stirling formula

$$p_1 \approx \sqrt{\frac{2}{\pi}} \frac{1}{\sqrt{d}} e^{-d/\nu'^2}. \tag{20}$$

Then the average number of samplings needed, $\overline{K}$, can be evaluated through a simple model for a sequence of Bernoulli trials with the number of trials as random

$$\begin{aligned} \overline{K} &= \sum_{k=1}^{+\infty} k(1-p_1)^{(k-1)} p_1 \\ &\approx \sqrt{\frac{\pi}{2}} \sqrt{d} e^{d/\nu'^2}. \end{aligned} \tag{21}$$

Then the time complexity, T, is

$$\begin{aligned} T &= S\overline{K} \\ &\approx O\left(\frac{(\nu' \ \ln \ \nu')^2}{\epsilon_g^2} \sqrt{d} e^{d/\nu'^2}\right). \end{aligned} \tag{22}$$

Q.E.D.

ACKNOWLEDGMENT

The authors give special thanks to T. K. Ho for providing NIST data, related references, and helpful discussions. The authors would like to thank H. T. Demiral, E. Kleinberg, G. Nagy, and H. Baird for providing reference and helpful discussions; and C. Hood, G. Nagy, and S. Epstein for carefully reading the manuscript and making comments.

REFERENCES

[1] A. Barron, "Approximation and estimation bounds for artificial neural networks," *Machine Learning*, vol. 14, pp. 113–143, 1994.
[2] ——, "Universal approximation bounds for artificial neural networks," *IEEE Trans. Inform. Theory*, vol. 39, pp. 930–944, 1993.
[3] ——, "Predicted squared error: A criterion for automatic model selection," in *Self-Organizing Methods in Modeling*, S. J. Farlow, Ed. New York: Marcel Dekker, 1984.
[4] E. Baum and D. Haussler, "What size net gives valid generalization?" *Neural Computa.*, vol. 1no. 1, pp. 151–160, 1989.
[5] R. S. Berlind, "an alternative method of stochastic discrimination with applications to pattern recognition," Ph.D. dissertation, State Univ. New York, Buffalo, 1994.
[6] A. L. Blum and R. L. Rivest, "Training a three-node neural network is NP-complete," *Neural Networks*, vol. 5, pp. 117–127, 1992.
[7] L. Breiman, "Stacked generalization," Dep. Statist., Univ. California, Berkeley, Tech. Rep., TR-367, Aug. 1992.
[8] N. Cesa-Bianchi, Y. Freud, D. P. Helmbold, D. Haussler, R. E. Shapire, and M. K. Warmuth, "How to use expert advice," in *Proc. Found. Comput. Sci.*, pp. 382–391, 1993.
[9] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 21–27, 1967.
[10] L, Devroye, "Automatic pattern recognition— A study of the probability of error, " *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-10, pp. 530–543, 1988.
[11] H. Drucker, R. Schapire, and P. Simard, "Improving performance in neural networks using a boosting algorithm," in *Proc. Neural Inform. Processing Symp.*, 1993, pp. 42–49.
[12] H. Drucker and C. Cortes, "Boosting decision trees," in *Neural Information Processing Symp.*, 1995.
[13] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[14] F. Blayo, Y. Cheneval, J. Madrenas, M. Moreno, and J. L. Voz, "Deliverable R3-B4-P Task B4: Benchmarks," Enhanced learning for evolutive neural architecture, ESPRIT-Basic res. project 6891, June 1995 [Online]. Available FTP:/pub/neural-nets/ELENA/databases/Benchmarks.ps.Z on ftp.dice.ucl.ac.be
[15] C. Aviles-Cruz, A. Guerin-Dugue, J. L. Voz, and D. Van Cappel, "Deliverable R3-B1-P Task B1: Databases," Enhanced learning for evolutive neural architecture, ESPRIT-Basic res. project 6891, June 1995. Available FTP:/pub/neural-nets/ELENA/databases/Benchmarks.ps.Z on ftp.dice.ucl.ac.be
[16] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. New York: Wiley, vol. 1, 1968, p. 195.
[17] Y. Freund, "Boosting a weak learning algorithm by majority," in *Proc. 3rd Wkshp. Computa. Learning Theory*. San Mateo, CA: Morgan Kaufmann, 1990, pp. 202–216.
[18] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias variance dilemma," *Neural Computa.* vol. 4, no. 1, pp. 1–58, Jan. 1992.
[19] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 993–1001, 1990.
[20] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems, " *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 66–75, 1994.
[21] S. J. Randy and A. K. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13), pp. 252–264, 1994.
[22] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computa.*, vol. 3, pp. 79–87, 1991.
[23] C. Ji and D. Psaltis, "The VC-dimension versus the statistical capacity for two layer network with binary weights," in *Proc. 4th Wkshp. Computa. Learning Theory*, 1991.
[24] S. Judd, *Neural Network Design and The Complexity of Learning*. Cambridge, MA: MIT Press, 1990.
[25] A. Stwart and J. K. Ord, *Kandell's Advanced Theory of Statis*. Oxford, U.K.: Oxford Univ. Press, 1987.
[26] E. M. Kleinberg, "Stochastic discrimination," *Ann. Math. Artificial Intell.*, vol. 1, pp. 207–239, 1990.
[27] E. M. Kleinberg and T. Ho, "Pattern recognition by stochastic modeling," in *Proc. 3rd Int. Wkshp. Frontiers in Handwriting Recognition*, Buffalo, NY, May 1993, pp. 175–183.
[28] N. Little and M. Warmuth, "The weighted majority algorithm," in *Proc. 3rd Wkshp. Computa. Learning Theory*, 1989.
[29] R. Meir, "Bias, variance, and the combination of estimators: The case of linear least squares," Dep. Electr. Eng., Technion, Haifa, Israel, 1994.
[30] J. Moody and C. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computa.*, vol. 1, 281–294, 1989.
[31] N. J. Nilsson, *Learning Machines*. San Mateo, CA: Morgan Kaufmann, 1990.
[32] S. J. Nowlan, and G. E. Hinton, "Simplifying neural networks by soft weight sharing," *Neural Computa.*, vol. 4, 1992, pp. 473–493.
[33] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble method for neural networks," *Artificial Neural Networks for Speech and Vision*, 1993, ch. 10.
[34] L. Prechelt, "PROBN1-A Set of Benchmarks and benchmarking rules for neural-network training algorithms," Fakultät für Informatik, Universität Karlsuhe, D-76128 Karlsruhe, Germany, Tech. Rep. 21/94, Sept. 1994 [Online]. Anonymous FTP:/pub/papers/techreports/1994/1994-21.ps.z on ftp.ira.uka.de
[35] D. Psaltis, R. R. Snap, and S. S. Venkatesh, "On the finite sample performance of nearest neighbor classifiers," *IEEE Trans. Inform. Theory*, vol. 40, pp. 820–837, May 1994.
[36] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197–227, 1990.

[37] N. Tishby, E. Levin, and S. Solla, "Consistent inference of probability in layered networks: Prediction and generalization," in *Proc. Int. Joint Conf. Neural Networks*, 1989.

[38] K. Tumer and J. Ghosh, "Theoretical foundations of linear and order statistics combiners for neural pattern classifiers." *IEEE Trans. Neural Networks*, 1995.

[39] L. Xu, M. I. Jordan, and G. E. Hinton, "An alternative model for mixtures of experts," *Advances in Neural Information Processing System 7*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Cambridge, MA: MIT Press, 1995, pp. 633–640.

[40] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 418–435, 1992.

[41] L. G. Valient, "A theory of learnable," *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[42] V. Vapnik, *Estimation Of Dependences Based On Empirical Data*. New York: Springer-Verlag, 1982.

[43] S. S. Venkatesh, "Directed drift—A new linear threshold algorithm for learning binary weights online," *J. Comput. Syst.* vol. 46, no. 2, pp. 198–217, Apr. 1993.

[44] D. Wolpert, "Staked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

Chuanyi Ji received the B.S. degree with honors from Tsinghua University, Beijing, China, in 1983, the M.S. degree from the University of Pennsylvania. Philadelphia, in 1986, and the Ph.D. degree from the California Institute of Technology (Caltech), Pasadena, in 1992, all in electrical engineering.

Since 1991, she has been an Assistant Professor in the Department of Electrical Computer and System Engineering at Rensselaer Polytechnic Institute, Troy, NY. Her research interests include pattern recognition, machine learning, and computer communication networks.

Dr. Ji received the Ming Li scholarship in 1989 at Caltech and the Early Career Development (CAREER) Award from the National Science Foundation in 1995.

Sheng Ma received the B.S. degree from Tsinghua University, Beijing, China, in 1992 and the M.S. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1995, both in electrical engineering. He is currently a Ph.D candidate in electrical engineering at Rensselaer Polytechnic Institute.

His current research interests include pattern recognition, machine learning, and computer communication networks.