RC22600 (W0210-128) October 22, 2002
Computer Science

IBM Research Report

Combination of classifers for supervised learning: A Survey

Sheng Ma
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Chuanyi Ji
Dept. of ECSE
Rensselaer Polytechnic Institute
Troy, NY 12180

PATTERN RECOGNITION AND STRING MATCHING
D. Chen and X. Cheng (Eds.) pp. – – –
©2002 Kluwer Academic Publishers

Combination of Classifiers for Supervised Learning: A Survey

Sheng Ma
IBM T.J. Watson Research Center
Hawthorne, NY 10532
E-mail: `shengma@us.ibm.com`

Chuanyi Ji
Dept. of ECSE
Rensselaer Polytechnic Institute
Troy, NY 12180
E-mail: `chuanyi@ecse.rpi.edu`

Contents

1 Introduction

In many important application areas such as signal processing, pattern recognition, control and communication, nonlinear adaptive systems are needed to approximate underlying mappings through learning from examples. In order for approximations to be sufficiently accurate, a good performance is required for nonlinear adaptive systems. Meanwhile, many applications, especially those in emerging areas of wireless communication and networking [11][20][42][52][50], require the learning to be done in real-time in order to adapt to a rapidly changing environment. Other applications such as data mining and searching the web need to deal with very large data sets [19][36][55], and thus the learning time must scale nicely with respect to the size of data . Since the size of learning machines determines the memory required for implementation, a learning machine with a compact structure is also preferred. Therefore, a challenging problem is how to develop a learning system with a compact structure that can achieve a good performance, and be adapted in real time rapidly. The goal of this paper is to address the important issues of performance and real-time learning for nonlinear adaptive learning machines by first reviewing recent work in combinations of classifiers, and then detailing some of our work in this area.

In supervised learning, performance addresses the problem of how to develop a learning machine to achieve optimal performance on examples that are not included in a training set. Efficiency deals with the complexity of a learning machine in both space and training time. Specifically, the space-complexity of a classifier refers to its size, and the time-complexity characterizes the computational time needed to develop such a classifier. These three issues are interrelated.

The performance of a supervised learning system is characterized by its generalization error, which measures the distance between the output function of a trained model and an underlying target function. Most of the existing training methods for classifiers in supervised learning suffer from an intrinsic problem in pattern recognition: the bias and variance dilemma [25][21]. That is, if a classifier is too large, [1] it may overfit a particular training set and thereby fail to maintain small generalization error. A small classifier, however, may be insufficient to approximate an optimal solution. In addition, one algorithmic problem is how to find a good classifier, which can learn a complex optimization problem with possibly many local minima.

The size of a learning machine can be characterized by the space-complexity, which is related to the number of free parameters (for instance, the number of weights of a neural-network classifier). A learning machine is considered to be efficient in space if its space-complexity scales as a polynomial function in terms of the dimension of feature vectors.[2] It has been found that compact classifiers like multiplayer feedforward neural networks are efficient approximators of a wide class of smooth functions. They possess a polynomial space-complexity [5]. Other learning machines, which consist of localized models such as nearest neighbor classifiers [13], Parzen windows

[1]For instance, feedforward neural networks as classifiers with too many neurons.

[2]Here the dimension of feature vectors is used as a measure of the complexity of a problem.

[18], and linear combinations of localized basis functions [43][5], may suffer from the so-called "curse of dimensionality." That is, their space-complexity scales exponentially with the dimension of the feature vectors [5][45][35].

Learning time can be characterized by the time-complexity as a scaling property with respect to the dimension of feature vectors. An adaptive learning system is considered efficient in time if the time complexity is polynomial. For example, training feedforward neural-network classifiers is slow, and in general, NP-complete [7][31]; while training the aforementioned localized classifiers can be done quickly. For example, nearest-neighbor classifiers simply remember all training samples, and therefore do not require training at all. This presents a dilemma existing between performance, space-, and time-complexity. These three fundamental issues – performance, efficiency, and space complexity – motivate the search for new solutions for adaptive learning machines, from training an individual model, to recent activities on combining well-trained models, and of many weak models.

Numerous methods have been developed in training an individual classifier. However, finding an optimal sized classifier is still difficult if not impossible. To address the performance issue, the combination of well-trained models has been proposed. The basic idea is to pick a slightly oversized model as a base for combination. Since the base model is oversized, it has a low bias but a high variance. The algorithm, therefore, relies on combinations to reduce the overall variance, and thus the generalization error.

Although the combination of well-trained models relaxes the need to optimize the size of a classifier, the learning time increases with the number of models combined. The combination of weak models is proposed to improve the efficiency of the combination of well-trained models while preserving the nice performance property. In particular, this approach uses the similar combination scheme but chooses a weak classifier as a base model. A weak classifier has a performance only slightly better than that due to random guessing, and thus has a large bias but a small variance. From the performance perspective, since different weak models are forced to learn different parts of a problem, combinations of many such weak models can reduce the overall bias and thus achieve good generalization. From the time-efficiency perspective, since the base model performs only slightly better than random guessing, it can be obtained efficiently. Furthermore, this approach uses an incremental learning procedure, and the whole training process can be very efficient.

The goal of this paper is to provide an overview of different approaches in combining weak classifiers. We would like to identify a general framework for various approaches with different grounds. Further, we would like to provide an overview of both theoretical and empirical results that demonstrate the advantages of combining weak classifiers in terms of both performance and efficiency.

The paper is organized as follows. Section 2 gives the background knowledge on performance and efficiency. Section 3 reviews the research on combining well-trained models to improve the performance. Section 4 discusses combinations of weak classifiers and how combined weak classifiers trade off among performance, time- and space-complexity. Section 5 discusses the empirical results of combining week classifiers. Section 6 discusses the theoretical results. Section 7 concludes the paper. As there is a large volume of the related work, we apologize for possible omissions.

2 Background: Performance and Efficiency

2.1 Notation

In a supervised learning environment, let $D = \{\mathbf{x}_n, t_n\}_{n=1}^N$ denote N training samples pairs, where $\mathbf{x}_n \in R^d$ is a d-dimensional feature vector of the n-th sample and t_n is the corresponding target. For simplicity, t_n is assumed to be a scalar in this paper. Furthermore, we assume that there is a function $f(\cdot)$ so that $t_n = f(\mathbf{x}_n)$.[3] For the function approximation problem, t_n is a real number. For classification, t_n indicates which class the n-th sample belongs to.

2.2 Performance

Let $f_D(\mathbf{x};\mathbf{w})$ be a model with a set of parameters $\mathbf{w}$ and trained on training set D. The performance of $f_D(\mathbf{x};\mathbf{w})$ can be measured in terms of the difference between a function $f(\mathbf{x})$ to be approximated and its approximation $f_D(\mathbf{x};\mathbf{w})$ through the squared norm

$$\int \mid f(\mathbf{x}) - \mathbf{f_D}(\mathbf{x};\mathbf{w}) \mid^{\mathbf{2}} \mathbf{p}(\mathbf{x})\mathbf{dx}, \tag{1}$$

where $p(\mathbf{x})$ is the probability density function of $\mathbf{x}$. If training samples are drawn randomly, the expected squared norm is

$$\mathbf{E}_D(\| f(\mathbf{x}) - \mathbf{f_D}(\mathbf{x};\mathbf{w}) \|^{\mathbf{2}}) = \mathbf{E_D}(\int \mid \mathbf{f}(\mathbf{x}) - \mathbf{f_D}(\mathbf{x};\mathbf{w}) \mid^{\mathbf{2}} \mathbf{p}(\mathbf{x})\mathbf{dx}), \tag{2}$$

[3]For simplicity, this paper does not address possible noise terms.

where the expectation ($\mathbf{E}$) is done over all possible training sets D of the same size. The quantity, $\mathbf{E}_D(|| \, f(\mathbf{x}) - \mathbf{f_D}(\mathbf{x};\mathbf{w}) \, ||^{\mathbf{2}})$ can be considered as a measure of the performance. It is also called the generalization error of $f_D(\mathbf{x};\mathbf{w})$, since it measures the average performance of $f_D(\mathbf{x};\mathbf{w})$ in approximating the unknown function $f(\mathbf{x})$.

Pattern classification is considered as a special case of nonlinear regression and its generalization error can be defined as probability of incorrect classification, which is $\mathbf{E_D}(\Pr(\mathbf{f_D}(\mathbf{x};\mathbf{w}) \neq \mathbf{t}))$.

2.3 Performance Issues

Two important issues are related to the performance: the bias and variance dilemma, and the relation of which to possibly a complex objective function with many local minima.

The first issue is intrinsic, and is independent of algorithms used. The second issue is algorithm- and problem-dependent.

2.3.1 Bias and Variance Dilemma

The generalization error is affected by two factors: Bias and variance. Let $f(\mathbf{x};\hat{\mathbf{w}})$ be the best model in the model space. That is, $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \int_{\mathbf{x}} (f(\mathbf{x}) - \mathbf{f}(\mathbf{x};\mathbf{w}))^{\mathbf{2}} \mathbf{p}(\mathbf{x}) \mathbf{dx}$. Note that $\hat{\mathbf{w}}$ does not depend on the training data. The bias $Bias(\mathbf{w})$ and variance $Var(\mathbf{w})$ for a model can be defined as:

$$Bias(\mathbf{w}) = \mathbf{E}(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x};\hat{\mathbf{w}}))^{\mathbf{2}}, \tag{3}$$

where the expectation is on a randomly drawn $\mathbf{x}$, and

$$Var(\mathbf{w}) = \mathbf{E_D}(\mathbf{E}(\mathbf{f_D}(\mathbf{x};\mathbf{w}) - \mathbf{f}(\mathbf{x};\hat{\mathbf{w}}))^{\mathbf{2}}). \tag{4}$$

The inner expectation is on $\mathbf{x}$, and the outer expectation is on randomly drawn training sets of the same size. Therefore, the generalization error can be decomposed into, and bounded by, a sum of the bias and variance [5][21][10]

$$\mathbf{E}_D(|| \, f(\mathbf{x}) - \mathbf{f_D}(\mathbf{x};\mathbf{w}) \, ||^{\mathbf{2}}) \leq \mathbf{2}(\mathbf{Bias}(\mathbf{w}) + \mathbf{Var}(\mathbf{w})). \tag{5}$$

The first term of the right-hand side is the error due to the bias because of an inappropriate choice of the size of a class of models, when the number of training samples is assumed to be infinite. The latter is the error due to the variance because of the finite number of training samples[4].

[4]which is equivalent to the space-complexity versus the number of training samples.

For the special case when a class of models are chosen to be two-layer feed-forward neural-network classifiers with L sigmoid hidden units,[5] the bias and variance were bounded explicitly by Barron [5]. In particular, $O(\frac{1}{L})$ and $O(\frac{Ldlog N}{N})$ are upper bounds for the bias and variance respectively, where $O(z)$ represents a quantity in the order of z. The fact that $O(\frac{1}{L})$ decreases with the number of hidden units suggests that a large neural network with many free parameters is less biased. As it has been shown by Barron [5] that feedforward neural networks are capable of approximating a wide class of smooth functions when the number of hidden units is sufficiently large, a larger set of neural networks (with a larger L) certainly contains a better approximation of $f(\mathbf{x})$ than a smaller set of neural networks (with L being smaller). That is why the bias is smaller when L is larger. However, when the number of training samples is finite, a network with an excessively large space-complexity will overfit the training set. This can be understood from the concept of information capacity of neural networks [1][14][6][28][38] in information theory. The information capacity is the maximum number of training samples that can be memorized by a neural network given a certain space-complexity. When the space-complexity exceeds the information capacity, learning machines are so large that they only memorize training samples. Since there are many such neural networks with the same space-complexity but different choices of weights which can memorize training samples, their (generalization) performance varies and thus leads to a large variance. That is, the average performance can decrease as L gets larger. This can be observed from the term $O(\frac{Ldlog N}{N})$, which increases with respect to L. Therefore, a trade-off needs to be made between bias and variance.

2.3.2 Bias-variance and Local Minima

An issue which needs to be clarified is whether the bias and the variance are related to local minima, or the effectiveness of an algorithm at finding a good solution. In our view, the bias by definition only depends on the structure of a class of classifiers but not the choice of values of their parameters (such as weights of neural-net classifiers), and is therefore independent of any training algorithms. To understand how the variance relates to local minima, we may consider the sample variance. For any given (randomly drawn) training set D of size N, a classifier corresponding to $f_D(\mathbf{x};\mathbf{w})$ can be found, which contributes to one sample in the sample variance. The true variance can be

[5]The number of free parameters is thus approximately $L(d+1)$.

estimated through a sufficiently large number of such samples. The definition given by Barron [5] assumed that each $f_D(\mathbf{x};\mathbf{w})$ corresponded to the global minimum of the error function, e.g. the best neural-network (classifier) that can be obtained given a training set. If the variance depended on a specific algorithm or applications, the theoretical result may lose its generality. This definition (given by Barron) simply suggests that the bias and variance dilemma is intrinsic for nonparametric regression (even if one had the most powerful algorithm to obtain the globally optimal classifier). Therefore, such a definition on the variance can also be considered to be algorithm independent.

In practice, for a given training set and a chosen error function, different algorithms result in different classifiers corresponding to different $f_D(\mathbf{x};\mathbf{w})$'s. If an algorithm is always better at finding a globally optimal solution, and another algorithm (e.g. a gradient-descent algorithm) gets stuck at a local minimum more often, then the former would have a smaller sample variance than the latter.

3 Combinations of Well-Trained Models

To alleviate the difficulties in finding an optimal structure of a single nonlinear classifier, methods are proposed to combine different models. The main idea is to train multiple models, such as neural networks, decision trees, or other classifiers individually, and then combine them as an ultimate model. The hope for using a combination to improve the generalization performance is that if individual models make classification errors differently, a combined model would be able to improve upon the performance of individuals.

Tremendous efforts have been made to investigate whether a combined model can indeed improve the performance, and how to combine models to achieve good performance [49][2]. Specifically, 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 [12][37]. Combinations of neural networks [44][57] and regressors [9] have also been used for both classification and function approximation problems. Somewhat similar ideas were also used for collective agents like the classifier systems and genetic algorithms [22][26][27].

As a good review was offered by Dietterich [16] on this subject; in this section, we do not intend to survey all aspects of combinations of well-trained models. Rather, we focus on the general issues that are relevant to the next subject we discuss on combinations of weak classifiers.

3.1 Key Issues

A combined model can be represented in a simple form. Let $h_k(\mathbf{x})$'s be a set of total K models defined on a feature vector $\mathbf{x} \in \mathbf{R^d}$, where a $h_k(\mathbf{x})$ can be a neural-network classifier (or a single neuron). Let w_k be the weighting factor for the k-th model. The combination of these K models can be

represented as

$$f_K(\mathbf{x}) = \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{w_k h_k(x)}. \tag{6}$$

$f_K(\mathbf{x})$ is essentially a linear combination of the so-called base models $h_k(\mathbf{x})$'s.

When a (two-class) classification is considered, the combined model becomes a combined classifier $C(\mathbf{x})$, where

$$C(\mathbf{x}) = \mathbf{I(f_K(x))}. \tag{7}$$

$I(z)$ is an indicator function, where $I(z) = 1$ for $z > 0$, and $I(z) = 0$ otherwise. $C(\mathbf{x})$ is a label assigned to a feature vector $\mathbf{x}$ by a combined classifier. When $h_k(\mathbf{x})$'s are also classifiers, $C(\mathbf{x})$ is a combination of classifiers.

There are two key issues in combinations of models.

(1) How to obtain a set of base models, $\{h_k(\mathbf{x})\}_{\mathbf{k=1}}^{\mathbf{K}}$?

(2) Given a set of base models, how to choose an optimal set of weighting factors $\{w_k\}_{k=1}^{K}$ so that the generalization error of the combined model is minimized?

3.2 Obtaining Base Models

Numerous algorithms have been developed to explore the first question. These approaches are distinct in two ways. One type of approaches utilize base models with different architectures, and/or obtained through different algorithms. Another type of approaches use different ways to "perturb" the training process so that the base models obtained can have diverse error patterns. Having classifiers with diverse error patterns were shown to be crucial to effectively improving the performance through combinations of models [34][32]. This can be understood through two (extreme) examples. At one extreme, if these classifiers are identical, there will be no gain from any combination. At the other extreme, if these classifiers make independent errors with a probability less than 0.5, the overall number of errors made after a combination can decrease exponentially as the number of such classifiers increases [32]. However, the "diversity" is not easy to achieve because all models are trained to do essentially similar tasks, and thus more or less dependent. Therefore, a certain randomness should be introduced to "perturb" the learning procedure so that models can learn different parts of a problem, and thereby make errors as differently as possible. The common methods used to perturb training were summarized by Dietterich [17][16].

3.3 Combination

The other important issue for combination is how to determine the weighting factors. Intuitively, the outputs of base models $\{h_k(\mathbf{x})\}_{\mathbf{k=1}}^{\mathbf{K}}$ can be used to "train" the weighting factors $\{w_k\}_{k=1}^{K}$. However, the straightforward minimization approach often results in overfitting [41][39] and is therefore not applicable. This is because the base classifiers are highly correlated since they are designed to solve similar tasks. One simplest algorithm called majority vote uses the equal weighting for base models, i.e., $w_k = 1/K$ for all k. Clearly, this scheme is not optimal because it does not take full consideration of differences among base models. But majority vote has been used widely due to its simplicity [30][8][10]. More elaborate algorithms have been investigated for combination. Wolpert [57] proposed combining base models through minimizing the squared error using cross-validation. Breiman [9] later imposed proper constraints on the weights w_k's. Freund et al. [24][23] proposed using the confidence of a base model as the weight for that model. Principle component analysis [40] was also investigated to explore and thus discount correlation among individual base models.

3.4 Discussion

Both theoretical and empirical results [8][10] suggest that combinations of well-trained models can ease the bias-variance dilemma and improve the performance substantially. That is, we can select a base model with a relatively large size so that it has a small bias but a large variance. A combination scheme can be responsible for reducing the overall variance of a combined model [8][10][44][40]. Therefore, finding an optimal structure of a model is no longer important. Very often, however, the price paid for the gain in performance is a larger space-complexity. In addition, as several models are combined, each of which may take a long-time to train, an even longer training time results in for a combination. Since training time is critical for real-time applications and is more difficult to tackle than the space-complexity, a natural question to ask is whether combinations of models can be used to improve the time-complexity at a reasonable cost of the space-complexity. Combining *weak* models provides a promising answer to this question.

4 Performance and Efficiency: Combinations of Weak Classifiers

Although individual base models in a combination can be quite general, we focus on classifiers as base models in this section.

4.1 Three Approaches

There are three main approaches developed on combinations of

weak classifiers: the boosting algorithm called Adaboost by Freund and Shapire [23][24], the stochastic discrimination (SD) by Kleinberg [32][33], and the combination of weak perceptrons (CWP) by Ji and Ma [15][29][30].

The concept of weak learning was first introduced by Kearns and Valiant as a theoretical question in the context of the probabilistic approximately correct (PAC) learning theory [6][54]. The question can be informally stated as " Does the existence of a weak learner imply the existence of an efficient strong learner?" Schapire [51] first proved the existence with a "yes" through a recursive and constructive approach. He showed that if a classification problem is solvable in the PAC framework, the problem can be solved through combinations of weak classifiers that can do a little better than random guessing. Later, Freund [23] showed in theory that a combination of weak classifiers through the simple majority vote can combine the weak classifiers into a strong classifier. The Adaboost [24] was further proposed as a practical algorithm for combination. Although this work [23][24] illustrated that weak classifiers could be combined to achieve what a strong classifier could do, it did not address whether combinations of weak classifiers had any advantage in performance or efficiency compared with either training an individual (strong) classifier or combining well-trained individual strong classifiers.

In an independent work, Kleinberg [32] proposed stochastic discrimination and showed that combining a large number of weak classifiers could improve the (training) performance monotonically. In addition, he also showed that the time-complexity of combined weak classifiers was polynomial. The theory, however, was built on an assumption that weak classifiers made independent classification errors. Such an assumption cannot be achieved in real situations.

Ji and Ma [30][29][15] proposed combinations of weak perceptrons (or neurons). Through experiments, this work showed that a very simple algorithm for generating and combining weak classifiers may achieve better efficiency and performance than training a strong classifier or combining well-trained classifiers. Such a simple procedure is to randomly generate weak perceptrons as weak classifiers and combine these weak perceptrons through a majority vote. Time- and space-complexity of a combined classifier was formally defined, and shown to be polynomial for a "special case" when the strength of weak classifiers was properly chosen. A trade-off was explicitly shown between time-complexity and space-complexity.

In the following, we further review these three algorithms and the corresponding results.

4.2 Weak Classifiers

4.2.1 Definition

The strength of a classifier $C(\mathbf{x})$ can be characterized by ν, the so-called weakness factor, where $\nu > 2$. Let $\frac{1}{2} - \frac{1}{\nu}$ be the required (generalization) error of classifier $C(\mathbf{x})$, i.e., $\Pr(C(\mathbf{x}) \neq \mathbf{t}) = \mathbf{\frac{1}{2}} - \mathbf{\frac{1}{\nu}}$. If $\nu >> 2$, the classifier is considered to be a weak classifier because it only performs a little better than random guessing. The larger the ν is, the weaker the weak classifier. The time- and space-complexity of $C(\mathbf{x})$ follows the general definition given in Section 2, but should take into account the weakness factor.

If the space- and time-complexity of a combined classifier scales polynomially with respect to both the dimension of feature vectors and parameters such as the weakness factor and a desired generalization error, the classifier is said to be efficient. Otherwise, if an exponential scaling is observed, they are considered to be inefficient.

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 ensures that each weak classifier possesses 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 were to learn the same part of a problem, their combination would not do better than individual classifiers.

4.2.2 The Structure of Weak Classifiers

Both local and non-local classifiers have been proposed. Kleinberg et al. [33] proposed using a hyper-sphere as a weak classifier. Such a weak classifier dichotomizes all the samples that fall into the sphere as one class and those outside as the other class. This classifier is local because samples close to the center of a sphere are in the same class. This choice of weak classifiers, however, may suffer from the "curse of dimensionality" and thereby is difficult to handle high dimensional problems.

Kleinberg [33] and Freund et al. [24] independently proposed to use a half-space as a weak classifier. A half-space classifier classifies input features based on only one selected dimension (or feature) and ignores other dimensions. Therefore, the decision boundary of a half-space classifier is a hyperplane perpendicular to the selected dimension. Such a classifier is global and may be expected to have a nice (polynomial) scaling property with respect to the dimension of feature vectors, avoiding the curse of dimensionality for some cases. As shown by experimental results [24], the set of half-space weak classifiers is limited for representing an arbitrary decision boundary. As a result, the performance is not as good as that of combinations of stronger classifiers. To increase the representation power of the half-space classifiers, the union of two or more half spaces were also considered.

To generate a global classifier with a stronger representation power, Ji and Ma proposed using a perceptron (or a neuron) as a weak classifier [15][30]. Similar to a half-space classifier, the decision boundary of this classifier is also a hyperplane, and therefore global. However, the orientation of the hyperplane can be arbitrary. Therefore, a hyperplane classifier is much more flexible than a half-space classifier. In fact, the combination of these perceptrons (or neurons) forms a two-layer neural network that has been shown to be able to approximate arbitrary functions [5].

4.3 Algorithms for Combinations of Weak Classifiers

Once the structure of weak classifiers is determined, qualified weak classifiers can be generated and then combined.

4.3.1 Generation of a Qualified Weak Classifier

There are two fundamentally different approaches to generate a qualified weak classifier. One is to apply a training algorithm to find the "best" weak classifier. The other is to randomly generate a weak classifier from the set of all weak classifiers until a qualified classifier is obtained.

Freund et al. [24] used the first scheme, where a feature is first selected randomly and the best threshold is then obtained through exhaustive search. Both stochastic discrimination (SD) [33] and combinations of weak classifiers [30] generated a weak classifier through the second approach. The algorithm can be described as follows.

(1) Partition the training data into "cares" and "don't-cares."

(2) Randomly generate a candidate classifier from the classifier space,

(3) Test classification error rate of the candidate classifier on the "cares."

(4) If the error rate on the "cared" samples is less than a threshold $0.5 - 1/\nu$, then keep this classifier and go to step (1) to generate another weak classifier. Otherwise go to step (2), where ν is the weakness factor.[6]

In this algorithm, a random classifier is first chosen as a candidate classifier. The strength of this candidate classifier is then tested on a set of "cares," which refer to those training samples incorrectly classified. If the candidate passes the threshold, it is accepted and combined with previous qualified classifier. Otherwise, it is rejected and the procedure is repeated until a qualified candidate is found. Therefore, this approach of generating a qualified classifier can also be called trial-until-qualified (TUQ).

To generate a perceptron classifier randomly [30], the direction of a hyperplane is first generated randomly and uniformly. Then a feature vector is picked randomly to place the hyperplane in the feature space.

Three considerations motivate the choice of a randomized (TUQ) approach over the training

approach. First, if a qualified classifier is weak enough so that it can be obtained through only several trials, the TUQ may be computational cheaper than training a classifier. Second, the random sampling to obtain a weak classifier may reduce overfitting, which is the problem inherent associated with any training algorithm. Third, random sampling may facilitate theoretical analysis as we are to discuss later.

4.3.2 Combinations of Qualified Weak Classifiers

In general, the combination schemes developed to combine well-trained

models can be used to combine weak classifiers. Specifically, Freund [24] used the weighted majority vote and chose the confidence of the corresponding classifier as the weighting factors. Both combinations of weak perceptrons [30] and stochastic discrimination [33] used the the simple majority vote to combine weak classifiers.

[6]A typical value of ν used in [30] is between 50 and 200. A typical number of combined weak classifiers is from 500 to 5,000.

4.3.3 Adaptively Reweighing and Combining (ARC)

Several algorithms for combining either weak or strong models have been discussed. They are bagging [8], AdaBoost [23], stochastic discrimination [32][33], and a combination of weak hyperplanes [30][15]. Although these algorithms result from different theories and serve different purposes, they share common characteristics. Breiman [10] proposed an adaptively reweighing and combining (ARC) framework to capture the common characteristics in these algorithms, where ARC characterizes a combination algorithm as an iterative three-step procedure (assuming $k-1$ ($k \geq 1$) classifiers have been obtained in the current combination):

(1) Resampling (or reweighing) the original training set to obtain the k-th training data set.

Bagging algorithm uniformly resamples the original training set. Boosting adaptively resamples the original training set. Stochastic discrimination and combinations of weak perceptrons resample uniformly on dynamically chosen "cares."

(2) Generating a qualified k-th classifier based on the k-th training data.

Bagging algorithm was designed to combine well-trained models. In practice, both neural networks and decision trees (CART) [8] have been used as base classifiers. Although Adaboost was originally designed to combine weak classifiers, it has been widely used to effectively boost the performance of strong classifiers [48][10]. SD and CWP all used the trial-until-qualified algorithm to generate randomly either a hyper-sphere or a perceptron but only keep those which exceed a certain performance threshold.

(3) Determining the weight for voting.

Adaboost uses the confidence over the performance of a model as the corresponding weight for voting. All other algorithms use the simple majority vote combination scheme.

5 Empirical results of combination of weak classifiers

In this section, empirical results are used to provide quantitative evaluation of combination of weak classifiers against other common approaches including combination of well-trained classifiers and single classifiers. In the latter cases, we use a neural network as a classifier.

Two synthetic problems are chosen to test the space- and time-complexity. Real applications from standard data bases are selected to compare the gen-

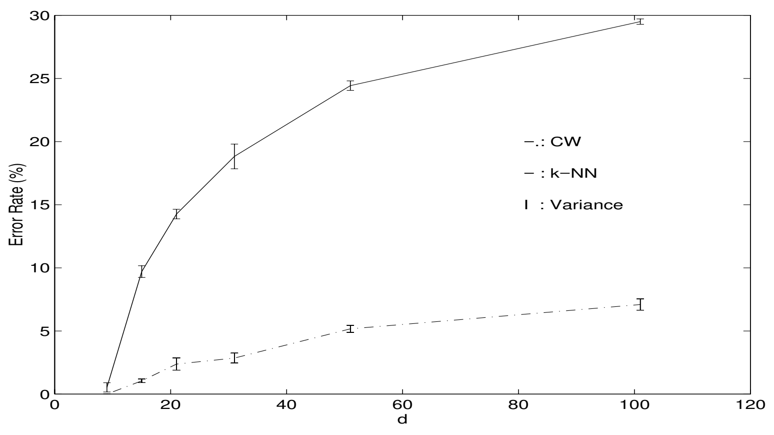

Figure 1: Performance on learning a perceptron

eralization performance of combinations of weak classifiers (CW) with that of other methods such as K-Nearest-Neighbor classifiers (K-NN)[7], artificial neural networks (ANN), combinations of neural networks (CNN), and stochastic discriminations (SD)[8].

5.1 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.

5.1.1 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. 10 different runs are conducted for each d, and the resulting classifiers are tested on another 4000 randomly-drawn

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

[8]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.

samples. For comparison, the k-Nearest Neighbor classifiers also learn the same data sets. The resulting average generalization error as well as the standard deviations are given in Figure 1.

As shown in Figure 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 [45]. The degradation in performance is moderate with combinations of weak classifiers.

5.1.2 Two Overlapping Gaussians

To further test the scaling properties of combinations of weak classifiers, a non-linearly separable problem is chosen from a standard database called ELENA [3][4]. 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 multi-variant Gaussians. Each dimension of the samples corresponds to an independent Gaussian random variable with zero mean, but samples in different classes have different standard deviations: 1 for samples in Class 1, and 2 for samples in Class 2. As shown in Figure 2 (for $d = 2$), there is a considerable amount of overlap between the samples in two classes, therefore the problem is non-linearly separable. The average generalization error and the standard deviations are given in Figure 3 for our algorithm based on 20 runs, and for other classifiers. The Bayes error is also given to show the theoretical limit [9].

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.

5.2 Real Applications

5.2.1 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[46][10].

[9]The results of ANN and the Bayes error are from what reported in ELENA [4]. For ANN, a one hidden lay feedforward neural network with 10 or 20 hidden nodes was trained by Backpropagation algorithm.

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

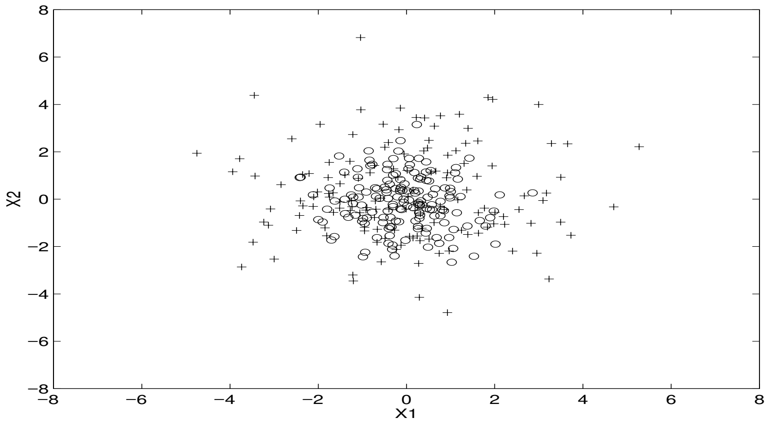

Figure 2: Two overlapping Gaussians

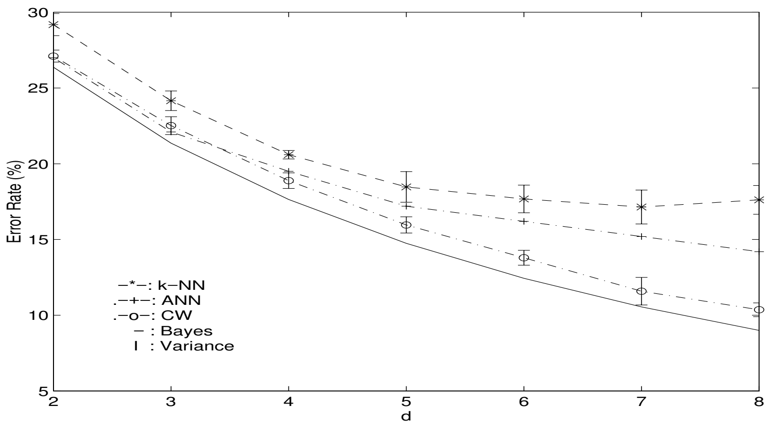

Figure 3: Performance versus the dimension of the feature vectors

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

Table 1: Performance on Card1, Diabetes1 and Gene1. σ: standard deviation

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. 345 out of 690 examples are used for training and the rest for testing. Diabetes1 data set is for determining whether diabetes is present based on 8-dimensional input patterns. 384 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. 1588 samples out of total of 3175 were used for training, and the rest for testing.

The average generalization error as well as the standard deviations are reported in Table 1. 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 [46][53] [11]. 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.

5.2.2 Hand-written Digit Recognition

Hand-written digit recognition is chosen to test our algorithm, since one of the previously developed method on combinations of weak classifiers (stochastic discrimination[32]) was applied to this problem. For the purpose of comparison, the same set of data as used in [32](from the NIST data base) is utilized to train and to test our algorithm. The data set contains 10000 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

[11]In [46][53], neural networks with many different architectures were trained by Back-propagation and several its variations. In [53], 3 to 7 well trained neural networks are combined by majority vote. Only the best reported results are listed in table 1.

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

Table 2: Performance on handwritten digit recognition.

networks, and stochastic discriminations are given in Table 2. The results for our methods are based on 5 runs, while the results for the other methods are from [32].

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 multi-class classification by converting an M-class classification problem into $\frac{M(M-1)}{2}$ two-class classification problems[32].

5.3 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 $\frac{1}{2} \leq \theta \leq \frac{1}{2} + \frac{1}{\nu}$. 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 8-dimensional overlapping Gaussians given in Section 5.1.2 are done to test the effects of ν. The performance and the average training time (CPU-time on Sun Spac-10) of combined weak classifiers based on 10 runs are given for different ν's in Figures 4 and 5, respectively[12]. 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 3. 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.

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

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

Table 3: Parameters used in our experiments.

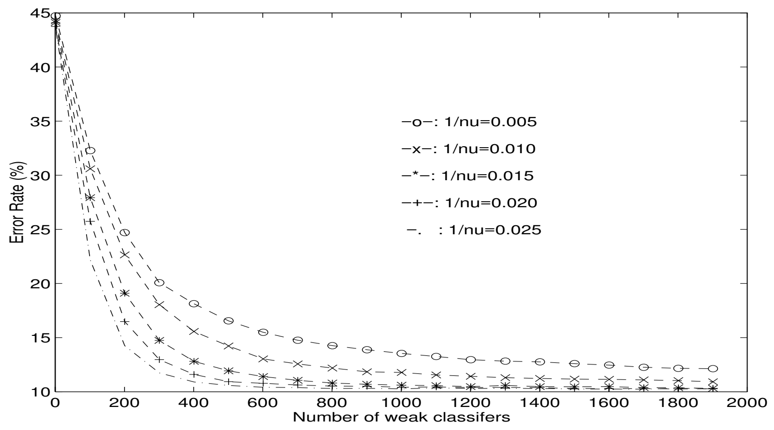

Figure 4: Performance versus the number of weak classifiers for different ν. nu: ν.

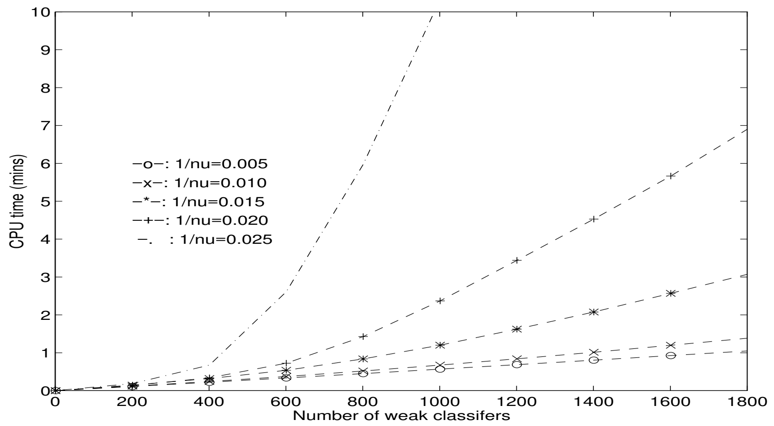

Figure 5: Training time versus the number of weak classifiers for different ν.

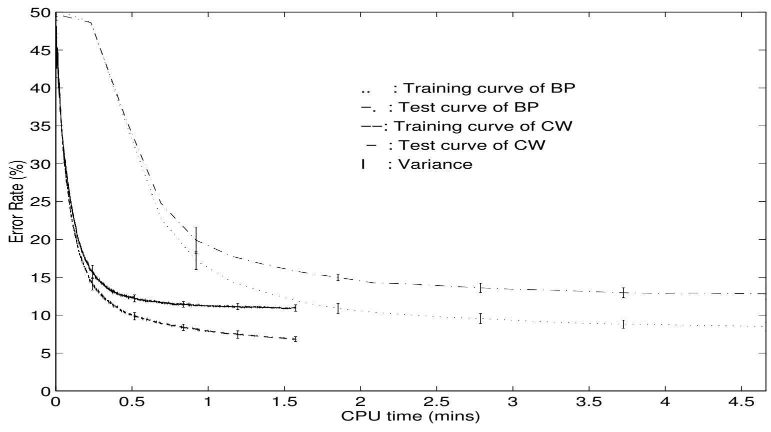

Figure 6: Performance versus CPU time

5.4 Training Time

To compare learning time with off-line BackPropagation (BP)[13], feedforward two layer neural network with 10 sigmoidal hidden units are trained by gradient-descent to learn the problem on the two 8-dimensional overlapping Gaussians. 2500 training samples are used. The performance versus CPU time[14] are plotted for both our algorithm and BP in Figure 6. For our algorithm, 2000 weak classifiers are combined. For BP, 1000 epochs 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 1). Then the cost on training time would be 7 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.

5.5 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

[13]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.

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

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-Nearest Neighbor classifiers. In the meantime whereas the k-nearest neighbor 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?

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 5.1.1[15]. The insights gained can shed light on how to analyze our algorithm for more general problems in the future.

6 Theoretical results: Performance of Combinations of Weak Classifiers

Since the performance of Adaboost [24] when used to combine a large number of weak classifiers were found to be either comparable or somewhat inferior to combining strong classifiers, Adaboost algorithms have been used mostly to combine a small number of well-trained classifiers [24][10][48]. Through extensive experimental comparison s[10][48][24], Adaboost (adaptive boosting algorithm) for combining strong models has been shown to perform consistently better than training an individual model or a bagging combination algorithm.

Can combinations of weak classifiers do better in performance than training a strong classifier? Although the independent assumption used in the theory for stochastic discrimination is difficult to satisfy in reality, better performance has been obtained consistently when a large number of weak

[15]Since many methods with good performance and efficiency can be used to learn a binary perceptron[56], 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.

classifiers are combined and used in handwritten digit recognition [33]. In particular, the resulting combined classifiers were shown to be less sensitive to overfitting. This means the performance of a combined classifier was usually improved when more and more weak classifiers were combined. Similar to combinations of well-trained classifiers, this in fact shows that combinations of weak classifiers reduce the variance when more and more weak classifiers are combined.

Combinations of weak perceptrons [30] were tested extensively on various synthetic and real data sets. The generalization performance of the combined weak perceptrons has been shown to be slightly better than combinations of well-trained classifiers, and outperforms individual neural network classifiers and k-nearest neighbor classifiers. Meanwhile, as will soon be shown, interesting phenomena have been observed that a trade-off can be made between performance and efficiency by combinations of weak perceptrons.

6.1 Efficiency of Combinations of Weak Classifiers

As the performance of combinations of weak classifiers is comparable to that of combinations of well-trained classifiers, what really are the advantages of using weak classifiers?

6.1.1 Stochastic Discrimination

Stochastic discrimination [32] first suggested that combinations of weak classifiers can be used to improve the training time. A theory was derived to show that when weak classifiers were assumed to make independent classification errors, the computational time for selecting a weak classifier was polynomial in terms of the dimension of feature vectors. As the space-complexity was also shown to be polynomial, the resulting time-complexity of a combined classifier was polynomial. Empirically, the training time needed to obtain a combination of weak classifiers was shown to be magnitudes faster than that of conventional training methods such as back-propagation.

Three factors were not included when the time-complexity was derived: the structure of weak classifiers, the weakness factor, and the statistical dependence among outputs of weak classifiers. As discussed in Section 5.2, the structure of weak classifiers relates directly to the space-complexity of a combined classifier, and thus determine its space efficiency. The weakness factor also affects the space-complexity of a combined classifier, since the weaker the weak classifiers are, the more weak classifiers may be needed to learn a problem, and the larger the space-complexity. This may in turn influence the time-complexity, since the time-complexity of a combined classifier can be regarded as the average training time needed to obtain a weak classifier multiplied by the space-complexity.[16]

6.1.2 Combinations of Weak Perceptrons

To shed light on whether and why combinations of weak perceptrons are efficient, theoretical analysis was carried out on a simple example when combinations of weak classifiers are used to learn underlying perceptrons [30].

The generalization error $\Pr(C_{2L+1}(\mathbf{x})\mathbf{t} < \mathbf{0})$ of the combined classifier $C_{2L+1}(\mathbf{x})$ with $2L+1$ weak perceptrons was derived and bounded above by a quantity in the order of $\frac{\nu ln(2L+1)}{\sqrt{2L+1}}$, i.e.,

$$\Pr(C_{2L+1}(\mathbf{x})\mathbf{t} < \mathbf{0}) \leq \mathbf{O}(\frac{\nu \mathbf{ln(2L+1)}}{\sqrt{\mathbf{2L+1}}}), \tag{8}$$

where ν is the weakness factor. $O(z)$ stands for a quantity in the order of z.

Such a bound shows that the generalization error decreases at a polynomial rate in terms of the number of weak perceptrons. By setting the upper bound to be equal to ϵ_g, which is a bound on the desired generalization error, the space-complexity S of a combined classifier[17] can be obtained easily as

$$S \leq O(\frac{(\nu ln \nu)^2}{\epsilon_g^2}). \tag{9}$$

Therefore, S is polynomial in both the weakness of factor ν and ϵ_g.

The time-complexity T of a combined classifier is defined as the average number of samplings needed to obtain a combined classifier when a desired generalization error at most ϵ_g. Such a time-complexity was shown to satisfy

$$T = O(\frac{(\nu ln \nu)^2}{\epsilon_g^2}\sqrt{d}e^{\frac{d}{\nu^2}}), \tag{10}$$

for d and ν large but $\nu \ll d$.

Since the larger the weakness factor ν, the larger the space-complexity S, but the smaller the time-complexity T, a trade-off can be made between the space- and time-complexity by finding an optimal ν. Specifically, let $\frac{dT}{d\nu} = 0$

[16]The number of weak classifiers by definition.

[17]S is the number of weak classifiers needed to achieve a certain generalization error.

and assume d large, an optimal weakness factor ν_o can be obtained as $\nu_o = O(\sqrt{d})$. Therefore, when $\sqrt{\frac{d}{\ln d}} \leq O(\nu)$, 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 $S = O(\frac{d ln d}{\epsilon_g^2})$, is also polynomial in d. The existence of such a critical value for the weakness factor suggests that the polynomial time-complexity may be obtained at a cost of a larger size classifier compared to that of a well-trained classifier with a fixed structure. The cost, however, is theoretically tolerable, since it scales polynomially in the dimension d of feature vectors.

Discussion

Through analyzing the time-complexity, an intuitive explanation can be drawn on when and why combinations of randomly selected weak perceptrons are efficient. 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 qualified weak classifier is accepted is small. As soon as weak classifiers are not too weak to destroy the polynomial space-complexity, the efficiency can be achieved both in time and space for combinations of weak perceptrons.

The theory provides an explicit relationship between performance and efficiency but is limited to a special case for learning a linear decision boundary. There are no similar results derived so far on nonlinear classification problems.

6.2 Open Issues

Due to the intrinsic difficulties of tackling performance and efficiency of nonlinear classifiers, many open issues need to be investigated on combinations of weak classifiers. Some of these open questions are

(1) how to show the optimality of a (randomized) algorithm for choosing and combining weak classifiers?

Assuming there were an infinite number of weak classifiers usable in a combination, this question essentially asks whether an algorithm is optimal in approximating a Bayes classifier. As an elegant analysis showed that the nearest-neighbor classifiers [13] are asymptotically Bayesian optimal, it remains open as to whether or not similar results could be obtained for a randomized algorithm using weak classifiers. The difficulty in obtaining such a result is that weak classifiers are statistically dependent.

(2) For what problems do a large number of weak classifiers exist to achieve a desired performance?

Even when an optimal algorithm is used to obtain a combination of weak classifiers, it is not clear whether a large enough number of weak classifiers exist so that the combined classifier can achieve a desired performance for a given structure (perceptron, for example) and a weakness factor. If the classification problem is too difficult, there may not exist any weak classifier, since a set of weak classifiers with the chosen structure may not have enough capacity [13][38][6][28] to do better than random guessing. (3) For what problems, do a large number of weak classifiers exist so that the time-complexity can be polynomial?

Not all problems can be solved in polynomial time (e.g. training non-linear neural classifiers is NP-complete). A polynomial complexity for randomly selecting a weak classifier is obtained based on the assumption that there exist a large number of weak classifiers. For example, if there exist at least a polynomial fraction of classifiers that can do better than random guessing, the average number of tries required to get one weak classifier is polynomial. Otherwise, choosing one acceptable weak classifier from an exponentially small fraction of all classifiers would take an exponential number of tries on the average. Since the number of acceptable weak classifiers depends on the problem, it is important to characterize problems for which a large number of weak classifiers do exist. The empirical results, however, are encouraging. That is, for many classification problems including those shown in this paper, a sufficiently large number of weak classifiers exist, and result in a powerful combined classifier.

7 Conclusion

We have reviewed several general techniques to improve efficiency and performance. In particular, we have discussed three different approaches , which improve the performance through making a trade-off between the bias and variance: (1) searching for an optimal structure for a single network. This was studied mainly for improving the performance of single model; (2) training several oversized models that have a low bias but a high variance, and then reducing the overall variance through combining these models. Combinations of well-trained models improve the performance through this approach; (3) training a large set of weak models that have a large bias but a small variance, and then reducing the overall bias and thus the generalization error by combining these weak models. Combinations of weak classifiers improve the performance through this scheme.

Combinations of weak classifiers, which use an incremental combination scheme and a randomized algorithm, have shown the potential to achieve time-efficiency as well as a good generalization performance. The cost is a polynomial space-complexity for benchmark problems, which is theoretically acceptable. Explicit relationships have been provided to illustrate the interrelation and the trade-off between performance and efficiency through combinations of weak classifiers. Randomized algorithms play an important role for combinations of (weak) classifiers.

Although much progress has been made in performance and efficiency of nonlinear adaptive systems, a lot of problems are still wide open for possible future research.

Acknowledgment

The authors would like to thank Leo Breiman and Andrew Barron for pointing out the issue of the existence of weak classifiers, and helpful discussions. The authors would like to thank T.K. Ho, Gene Kleinberg, George Nagy and Guohua Zhao for relevant references and helpful discussion. The support from the National Science Foundation (ECS-9312594 and (CAREER) IRI-95025 18) is gratefully acknowledged.

References

[1] Y.S. Abu-Mostafa, "Information theory, complexity and neural networks," *IEEE Communications Magazine*, vol.27, no.11, 25-28, Nov. 1989.

[2] J.A. and P.H. Swain. Consensus theoretic classification methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:688-704, 1992.

[3] 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 Research Project Number 6891, June, 1995. Anonymous FTP: /pub/neural-nets/ELENA/databases/Benchmarks.ps.Z on ftp.dice.ucl.ac.be.

[4] 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 Research Project Number 6891, June, 1995. Anonymous FTP: /pub/neural-nets/ELENA/databases/Benchmarks.ps.Z on ftp.dice.ucl.ac.be.

[5] A. Barron, "Universal Approximation Bounds for Artificial Neural Networks," *IEEE Trans. on Information Theory*, vol. IT-39, 930-944, 1993.

[6] E. Baum and D. Haussler,"What Size Net Gives Valid Generalization?"*Neural Computation*, 1(1), 151-160, 1989.

[7] A.L. Blum and R.L. Rivest, "Training A 3-Node Neural Network Is NP-Complete," *Neural Networks*, vol.5, 117-127, 1992.

[8] L. Breiman. Bagging predictors. *Machine Learning*, 24:132–140, 1996.

[9] L. Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996.

[10] L. Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–849, 1998.

[11] T. X Brown, H. Tong, S. Singh, "Optimizing admission control while ensuring quality of service in multimedia networks via reinforcement learning," to appear in Advances in Neural Information Processing Systems, ed. M. Kearns et al., MIT Press, 1999.

[12] N. Cesa-Bianchi, Yoav Freund, D.P. Helmbold, D. Haussler R.E. Shapire and M.K. Warmuth, " How to Use Expert Advice," *Proceeding of Foundation of Computer Science*, 382-391, 1993.

[13] T.M. Cover and P.E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inform. Theory,*, IT-13, 21-27, 1967.

[14] T.M. Cover, "Capacity Problems for Linear Machines," in *Pattern Recognition*, Thompson Book Co., 1968. ed. by L. Kanal.

[15] H.T. Demiral, S. Ma, and C. Ji. Combined power of weak classifiers. In *Proceeding of World Congress on Neural Networks*, pages 591–595, 1995.

[16] T.G. Dietterich. Machine-learning research. *AI Magazine*, 18:97–136, 1997.

[17] T.G. Dietterich. Discussion of Arcing classifiers. *Annals of Statistics*, 26:838–841, 1998.

[18] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[19] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "From Data Mining to Knowledge Discovery: An Overview, " In *Advances in Knowledge Discovery and Data Mining*, Menlo Park, CA: The AAAI Press/The MIT Press, 1996.

[20] T.L . Fine, S. B. Wicker, T. Berger, and J. Halpern, "Sensor-Assisted ALOHA for Wireless Networks, " *Proc. 1998 IEEE International Symposium on Information Theory*, Aug., 1999.

[21] T.L. Fine, *Feedforward Neural Network Methodology* ,Springer-Verlag, 1999.

[22] S. Forest, " Emergent Computation: Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks," *Physica D*, vol.42, 1-11, 1990.

[23] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285, 1995.

[24] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[25] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, 4, 1-58, 1992.

[26] J.H. Holland, “A Mathematical Framework for Studying Learning Classifier Systems,” *Physica*, vol. 22D, 307-317, 1986.

[27] J.H. Holland, “Genetic Algorithms,” *Scientific American*, vol.267(1), 66-72, 1992.

[28] C. Ji and D. Psaltis, “Capacity of Two-Layer Networks with Binary Weights,” *IEEE Trans. Information Theory*, Vol. 44, No.1, 256-268, Jan., 1998.

[29] C. Ji and S. Ma. Combined weak classifiers. In *NIPS*, pages 494–500, 1996.

[30] C. Ji and S. Ma. Combinations of weak classifiers. *IEEE Transactions on Neural Networks*, 8:32–42, 1997.

[31] S. Judd, *Neural Network Design and The Complexity of Learning*, MIT Press, 1990.

[32] E.M. Kleinberg, “Stochastic Discrimination,” *Annals of Mathematics and Artificial Intelligence*, vol.1, 207-239, 1990.

[33] E.M. Kleinberg and T. Ho, “Pattern Recognition by Stochastic Modeling,” *Proceeding of Third International Workshop on Frontiers in Handwriting Recognition*, 175-183, Buffalo, May 1993.

[34] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *NIPS*, pages 231–238, 1995.

[35] S.R. Kulkarni, G. Lugosi, and S. S. Venkatesh, “Learning pattern classification-a survey,” *IEEE Transactions on Information Theory*, Vol.44, No.6, 2178-2206, Oct. 1998.

[36] S. Lawrence, C.L. Giles, “Searching the Web: general and scientific information access,” *IEEE Communications Magazine*, vol.37, no.1, 116-122, Jan. 1999.

[37] N. Little and M. Warmuth, “The Weighted Majority Algorithm,” *The Third Workshop on Computational Learning Theory*, 1989.

[38] R.J . McEliece, E.C . Posner, E.R . Rodemich, S.S .Venkatesh, “The Capacity of the Hopfield Associative Memory,” *IEEE Trans. Inform. Theory*, Vol. IT-33, No. 4, 461-482, July 1987.

[39] R. Meir. Bias, variance and the combination of least squares estimators. In *NIPS*, pages 295–302, 1995.

[40] C. Merz and M.J. Pazzani. Handling redundancy in ensembles of learned models sing principal components. In *Proceedings of the National Conference on Artificial Intelligence*, 1996.

[41] C. Merz and M.J. Pazzani. Combining neural network regression estimates with regularized linear weights. In *NIPS*, 1997.

[42] U. Mitra and H.V. Poor, "Neural Network Techniques for Adaptive Multiuser Demodulation," *IEEE Journal on Selected Areas in Communications*, vol.12, no.9, 1460-1470, Dec. 1994.

[43] J. Moody and C. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol.1, 281-294, 1989.

[44] M.P. Perrone and L.N. Cooper, "When Networks Disagree: Ensemble Method for Neural Networks," Chap 10, *Artificial Neural Networks for Speech and Vision*, 1993.

[45] D. Psaltis, R.R. Snap and S.S. Venkatesh, "On The Finite Sample Performance of Nearest Neighbor Classifiers," *IEEE Trans. Inform. Theory*, IT-40, 820-837, May 1994.

[46] L. Prechelt, "PROBN1-A Set of Benchmarks and benchmarking rules for Neural Network Training Algorithms," *Technical Report* 21/94, *Fakultat fur Informatik, Universitat Karlsuhe*, D-76128 Karlsruhe, Germany, September 1994. Anonymous FTP:/pub/papers/techreports/1994/1994-21.ps.z on ftp.ira.uka.de.

[47] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman, 1993.

[48] J.R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

[49] S. Rangarajan, P. Jalote, and S.K. Tripathi. Capacity of voting systems. *IEEE Transactions on Software Engineering*, 19:698–705, 1993.

[50] I. Rish, M. Brodie and S. Ma. Accuracy versus efficiency in probabilistic diagnosis. *to be appeared in AAAI conference*, 2002.

[51] R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[52] M. Thottan and C. Ji, "Proactive Anomaly Detection Using Distributed Intelligent Agents," *Special Issue on Network Management-Today and Tomorrow, IEEE Net work, IEEE Network Magazine*, September 1998.

[53] K. Tumer and J. Ghosh. "Theoretical Foundations of linear and Order Statistics Combiners for Neural Pattern Classifiers." To be appeared in *IEEE Transactions on Neural Networks*, 1995.

[54] L.G. Valient, "A Theory of Learnable," *Communications of The ACM*, 27(11), 1134-1142, 1984.

[55] R. Vilalta, C. Apte, J.L. Hellerstein, S. Ma and S. Weiss, "Fault Prediction in Computer Networks," *IBM System Journal, Special Issue on AI*, August 2002.

[56] S. S. Venkatesh, "Directed Drift - A New Linear Threshold Algorithm for Learning Binary Weights Online," *J. Comput. Sys.* 46(2), 198-217, April, 1993.

[57] D. Wolpert, "Stacked Generalization," *Neural Networks*, vol. 5(2), 241-259, 1992.