Pattern Recognition 44 (2011) 97–106

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Sparse ensembles using weighted combination methods based on linear programming

Li Zhang *, Wei-Da Zhou

Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, China

ARTICLE INFO

Article history:
Received 11 March 2010
Received in revised form
2 July 2010
Accepted 18 July 2010

Keywords:
Classifier ensemble
Linear weighted combination
Linear programming
Sparse ensembles
k nearest neighbor

ABSTRACT

An ensemble of multiple classifiers is widely considered to be an effective technique for improving accuracy and stability of a single classifier. This paper proposes a framework of sparse ensembles and deals with new linear weighted combination methods for sparse ensembles. Sparse ensemble is to sparsely combine the outputs of multiple classifiers by using a sparse weight vector. When the continuous outputs of multiple classifiers are provided in our methods, the problem of solving sparse weight vector can be formulated as linear programming problems in which the hinge loss or/and the 1-norm regularization are exploited. Both the hinge loss and the 1-norm regularization are techniques inducing sparsity used in machine learning. We only ensemble classifiers with nonzero weight coefficients. In these LP-based methods, the ensemble training error is minimized while the weight vector of ensemble learning is controlled, which can be thought as implementing the structure risk minimization rule and naturally explains good performance of these methods. The promising experimental results over UCI data sets and the radar high-resolution range profile data are presented.

1. Introduction

Recently, combining multiple classifiers has been a very active research technique. It is widely accepted that combining multiple classifiers can achieve better classification performance than a single (best) classifier, supported by experimental results [1–3]. An ensemble means combining multiple versions of a single classifier or multiple various classifiers. One classifier used in an ensemble is called an individual or component classifier. There are two important issues in combining multiple classifiers. One is that an ensemble of classifiers must be both diverse and accurate in order to get better performance. Diversity can ensure that all the individual classifiers make uncorrelated errors. If classifiers get the same errors which will be propagated to the ensemble, no improvement can be achieved in combining multiple classifiers. In ensemble learning, there are two schemes to implement diversity [4]. One scheme is to seek diversity explicitly (i.e., to define a diversity measure and optimize it), and the other is to seek diversity implicitly. Here we consider the scheme of seeking diversity implicitly. One common way is to train individual classifiers by using different (randomly selected) training sets [5–7]. Bagging [5] and Boosting [6] are well known examples of successfully iterative methods for reducing a generalization error. The other way is to train multiple classifiers by using different feature sets [8,9]. In addition, accuracy of individual classifiers is also important, since too many poor classifiers can suppress correct predictions of good classifiers.

The other issue is about combination rules or fusion rules, which is regarding how to combine the outputs of individual classifiers. So far, many combination rules have been proposed [10–16]. If the labels are available, a simple (majority) voting (SV) rule can be used [10]. If the continuous outputs like posteriori probabilities are supplied, an average, linear or nonlinear combination rules can be employed [10,12,16]. Linear weighted voting is the most frequently used rule [11,12,15]. Work on weighted voting have addressed the problem of weights estimation, in a regression setting [11,14,15], or in a classification setting [12,17,18]. A linear weighted voting based on the minimum classification error (WV-MCE) criterion is presented in [12], which is solved by using gradient descent methods. In [17], a genetic algorithm (GA) is used to select the best subset of classifiers and the corresponding weight coefficients in neural network ensembles. Grove et al. [18] suggest that we should make the minimum margin of learned ensembles as large as possible by minimizing training set error. They propose the LP-Adaboost method to find the sparse weight vector.

The LP-Adaboost method in [18] and the GA-based method in [17] are the beginning of sparse ensembles. By sparse ensembles, we mean combining the outputs of all classifiers by a sparse weight vector. Each classifier model has its own weight value, zero or nonzero. Only classifiers corresponding to nonzero coefficients play a role in the ensemble. As it is known, a sparse

* Corresponding author.
E-mail address: lizhang.ml@gmail.com (L. Zhang).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.07.021

model representation in machine learning is expected to improve the generalization performance and computational efficiency [19–21]. The mechanism to maximize the sparseness of a model representation can be thought of as an approximative form of the minimizing description length principle which can be used to improve the generalization performance [7]. The sparsity in machine learning can be measured by the number of nonzero coefficients in a decision function.

The above combination rules except LP-Adaboost and GA-based methods are to try to combine all classifiers in an ensemble. In general classifier ensembles, it is necessary to combine all individual classifiers to ensure good performance. It results in a large memory requirement and a slow classification speed [22]. Selective ensembles, also called pruned ensembles, are designed to remedy the drawbacks of general classifier ensembles. Only a fraction of individual classifiers is selected and combined by using simple or weighted voting in selective ensembles. In [22], some methods are introduced for selecting a subset of individual classifiers, and the performance of these methods are compared in several benchmark classification tasks. The problem of selecting the optimal subset of classifiers is a combinatorial search assuming that the generalization performance can be estimated in terms of some quantity measured on the training set [22]. Recently, global optimization methods, e.g., GA [23] and semi-definite programming [24] are used to solve the combinatorial search problem. Since the global methods cost a lot, some suboptimal ensemble pruning methods based on ordered aggregation are proposed, including reduce-error pruning [25], margin distance minimization (MDM) [26], orientation ordering [27], boosting-based ordering [28], expectation propagation [29], and so on. Among the pruning techniques, MDM and boosting-based ordering methods provide similar or better classification performance [22]. Actually the concept of pruned ensembles is identical with that of sparse ensembles. In pruned ensemble, the coefficients of selected classifiers are nonzero, and unselected are zero, which generates a sparse weight vector. Generally, pruned ensembles use simple voting or weighted voting. The nonzero coefficients take the value one in simple voting [22], and a value proportional to the classification accuracy of the corresponding classifier [30,31], or found by some optimization methods [23,24,29] in weighted voting.

This paper gives a framework of sparse ensemble learning, and proposes new weighted combination methods for sparse ensembles. The key problem in sparse ensembles is to find a sparse weight vector. Grove and Schuurmans use a linear programming method to find a sparse weight vector. The objective function of LP-Adaboost is to minimize maximum margin in [18]. Here, our goal is to find a sparse weight vector by minimizing the ensemble training error and simultaneously controlling the weight vector of ensemble learning, which can be taken as implementing the structural risk minimization rule from the view of machine learning. In our methods, the continuous outputs (estimated posteriori probabilities or discriminant function values) of individual classifier are required. This learning problem can also be formulated as linear programming problems in which sparseness techniques the hinge loss or/and the 1-norm regularization are used. In our experiments, we consider the k NN classifier as an individual classifier and apply the new linear weighted combination rule to combine the multiple k NN classifiers.

The rest of this paper is organized as follows. In Section 2, we propose the framework of sparse ensembles and review the related work including some classical combination rules. Section 3 presents new linear weighted voting based on LP. We compare our methods with the single k NN classifier and the k NN ensemble classifiers based on other seven combination rules on the UCI data sets and the radar high-resolution range profile (HRRP) data in Section 4. Section 5 concludes this paper.

2. Sparse ensembles and other related work

In this section, we firstly propose the framework of classifier sparse ensembles and then introduce some other combination methods used in our experiments.

2.1. Framework of sparse ensembles

Sparse ensembles mean that we combine the outputs of all classifiers using a sparse weight vector. Each classifier model has its own weight value, zero or nonzero. Only classifiers corresponding to nonzero coefficients play a role in the ensemble. To reduce memory demand and improve test speed, it is required to select an optimal sub-ensemble (or a subset of classifiers) in pruned (or selective) ensembles [22,30–32]. Actually the concept of pruned ensembles is identical with that of sparse ensembles. In pruned ensemble, the coefficients of selected classifiers are nonzero, and unselected are zero, which creates a sparse weight vector.

Now consider a multi-class classification problem. Let a training sample set be $X=\{(\mathbf{x}_i,y_i)|\mathbf{x}_i\in\mathbb{R}^D,\ y_i\in\ \{1,2,\ldots,c\},\ i=1,2,\ldots,\ell\}$, where y_i are labels of $\mathbf{x}_i$, D is the dimensionality of the sample space (or the number of sample features), c is the number of classes, and ℓ is the total number of training samples. Hereafter we use ω_m to denote class m, $m=1,\ldots,c$. If $\mathbf{x}_i\in\omega_m$, then $y_i=m$. The framework of sparse ensembles is shown in Fig. 1. The whole ensemble process is divided into two phases: training phase and test phase. In training phase, $X_1,X_2,\ldots,X_N$ are the training sets of N individual classifiers, respectively. In this phase, we need to find the sparse weight vector $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\ldots,\alpha_N]^T\in\mathbb{R}^N$ by using some methods, such as LP-Adaboost. In the test phase, the goal is to estimate the label of a given test sample $\mathbf{x}$. Assume the j-th classifier would generate an output vector $\mathbf{f}_j=[f_{j1}(\mathbf{x}),f_{j2}(\mathbf{x}),\ldots,f_{jc}(\mathbf{x})]^T\in\mathbb{R}^c$, where $f_{jm}(\mathbf{x})$ are the output of the j-th classifier for the sample $\mathbf{x}$ associated with class ω_m, which could be posteriori probabilities or just only discriminant values normalized to the interval [0,1]. The ensemble output of $\mathbf{x}$ for class ω_m is

$$f_m^*=\sum_{j=1}^{N}\alpha_j f_{jm}(\mathbf{x}) \tag{1}$$

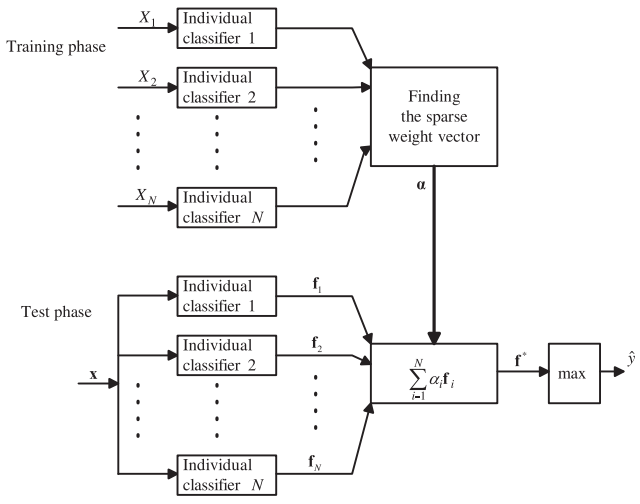

Fig. 1. Framework of classifier ensembles.

The estimated label for $\mathbf{x}$ can be estimated by

$$\hat{y} = \arg \max_{m=1,\ldots,c} f_m^* \tag{2}$$

2.1.1. Diversity of individual classifiers

In sparse ensembles, we first consider about the diversity of individual classifiers, or generating different classification outputs. There are two schemes to implement diversity [4], we only adopt the scheme of seeking diversity implicitly for its simpleness and popularity.

- Using different individual classifiers and the same training set, such as k NN, decision tree, neural networks, etc. [30,31]. Here, $X_i = X$, $i = 1,\ldots,N$.
- Using randomness or different parameters of some algorithms, e.g., initialization for neural networks [12].
- Using different data subsets and the same individual classifier, such as bootstrap samples [5–7], and feature subsets [8,9].

2.1.2. Weighed voting

In sparse ensembles, the combination rule adopts weighted voting. The key is how to find a sparse weight vector, not just a weight vector. There are many methods for finding a weight vector [11,12,14,15,18]; most of them would get a nonsparse weight vector except that in [18]. Pruned ensembles can result in sparse weight vector [22–24,29–31]. From (2) and (1), the weighted voting can be described as follows [12]:

$$\text{assign } \mathbf{x} \longrightarrow \omega_q \quad \text{if } f_q^* = \max_{m=1,\ldots,c} \sum_{j=1}^{N} \alpha_j f_{jm}(\mathbf{x}) \tag{3}$$

In (1), if all weight coefficients $\alpha_j = 1/N$, $j = 1,\ldots,N$, then simple weighted voting (SWV) (also called simple averaging) is resulted.

Another weighted combination formula is presented in [12].

$$f_m(\mathbf{x}) = \sum_{j=1}^{N} \alpha_{jm} f_{jm}(\mathbf{x}) \tag{4}$$

where α_{jm} is the weight coefficient of the jth classifier for class ω_m. Ref. [12] shows that weighted voting based on the MCE criterion using the combination formula (4) has the best performance in his experimental comparison. However, a probabilistic descent method is used to minimize the MCE criterion. As it is known, gradient descent methods often run into local minima.

2.2. Other combination methods

In the following, we briefly review some classical classifier combination methods including the naive Bayes combination methods and simple voting.

2.2.1. Naive Bayes combination methods

In these rules, assume that individual classifiers are mutually independent; hence the name "naive" [1,10]. Now the outputs of all individual classifiers should be posterior probabilities or their estimates, or $f_{jm}(\mathbf{x}) = P_j(\omega_m|\mathbf{x})$ which is the posterior probability of the test sample $\mathbf{x}$ belonging to class ω_m obtained from the j-th classifier. Obviously, these outputs $0 \le f_{jm}(\mathbf{x}) \le 1$. Two naive Bayes combination rules are given as follows. The interested reader should refer to [10] for detailed information.

- Product rule

  $\text{assign } \mathbf{x} \longrightarrow \omega_q$

  $$\text{if } [P(\omega_q)]^{-(N-1)} \prod_{j=1}^{N} f_{jq}(\mathbf{x}) = \max_{m=1,\ldots,c} [P(\omega_m)]^{-(N-1)} \prod_{j=1}^{N} f_{jm}(\mathbf{x}) \tag{5}$$

  where $P(\omega_m)$ are priori probabilities for class ω_m.

- Sum rule

  $\text{assign } \mathbf{x} \longrightarrow \omega_q$

  $$\text{if } (1-N)P(\omega_q) + \sum_{j=1}^{N} f_{jq}(\mathbf{x}) = \max_{m=1,\ldots,c} \left[(1-N)P(\omega_m) + \sum_{j=1}^{N} f_{jm}(\mathbf{x}) \right] \tag{6}$$

2.2.2. Simple voting

The output vectors $\mathbf{f}_j$ of models should be c-dimensional binary vectors

$$[f_{j1}(\mathbf{x}), f_{j2}(\mathbf{x}), \ldots, f_{jc}(\mathbf{x})]^T \in \{0,1\}^c, \quad j = 1,\ldots,N$$

where $f_{jm}(\mathbf{x}) = 1$ if and only if $\mathbf{x}$ is classified as class ω_m by using the j-th classifier, and $f_{jm}(\mathbf{x}) = 0$ otherwise. Thus the SV method can be described as

$$\text{assign } \mathbf{x} \longrightarrow \omega_q \quad \text{if } \sum_{j=1}^{N} f_{jq}(\mathbf{x}) = \max_{m=1,\ldots,c} \sum_{j=1}^{N} f_{jm}(\mathbf{x}) \tag{7}$$

In pruned ensembles, SV is also a common combination method [22,26,27].

3. New weighted combination methods based on linear programming

In this section, we propose new weighted combination methods based on LP to yield the sparse weight coefficients for sparse ensembles.

Suppose there are c class samples and a training sample set $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$, where $\mathbf{x}_i \in \mathbb{R}^D$, $y_i \in \{1,2,\ldots,c\}$. Let ω_m denote class m, and N the ensemble size. $X_j \subseteq \mathbb{R}^d$ is the training set utilized in the j-th classifier, where $d \le D$ is the dimensionality of the training set X_j. We only consider the simple linear combination formula (1). If a training sample $\mathbf{x}_i \in \omega_q$, (3) can be expressed as the following constraint

$$f_q(\mathbf{x}_i) > f_m(\mathbf{x}_i), \quad m = 1,\ldots,c, \ m \ne q \tag{8}$$

where $f_q(\mathbf{x}_i)$ is the ensemble output of sample $\mathbf{x}_i$ on class ω_q. Substituting (1) into (8), we obtain

$$\sum_{j=1}^{N} \alpha_j f_{jq}(\mathbf{x}_i) > \sum_{j=1}^{N} \alpha_j f_{jm}(\mathbf{x}_i), \quad m = 1,\ldots,c, \ m \ne q \tag{9}$$

When the output f_{jq} are regarded as class posterior probabilities or their estimates, the inequality (9) can be explained from the view of Bayesian theory. If $\mathbf{x}_i$ belongs to class ω_q, the weighted posterior probability (or ensemble output) on class ω_q should be the largest, otherwise $\mathbf{x}_i$ would be misclassified. Obviously, the better the classifier performance should be obtained, the larger the term $\sum_{j=1}^{N} \alpha_j f_{jq}(\mathbf{x}_i)$ in the left hand of (9) compared with that in the right hand is. Thus, we introduce a positive constant ε and have

$$\sum_{j=1}^{N} \alpha_j f_{jq}(\mathbf{x}_i) - \sum_{j=1}^{N} \alpha_j f_{jm}(\mathbf{x}_i) \ge \varepsilon, \quad m = 1,\ldots,c, \ m \ne q \tag{10}$$

Since these class posterior probabilities or discriminant function values are obtained from the training results of multiple classifiers, they might not be so accurate. We relax this inequality constraint by introducing positive slack variables ξ_{im}^q, and we

rewritten (10) as

$$\sum_{j=1}^{N}\alpha_j f_{jq}(\mathbf{x}_i)-\sum_{j=1}^{N}\alpha_j f_{jm}(\mathbf{x}_i)\geq\varepsilon-\xi_{im}^q,\quad \xi_{im}^q\geq 0,\ m=1,\ldots,c,\ m\neq q \tag{11}$$

If only $\xi_{im}^q>\varepsilon$, the sample $\mathbf{x}_i$ is misclassified. Thus, it is required to minimize the sum of ξ_{im}^q to reduce the ensemble training error. For this problem, we would obtain three different LP formulations based on ways of controlling weight vector.

3.1. LP1 method

Similar to the way of processing weight vector in LP-Adaboost, we first formulate the above problem as the following LP problem:

$$\begin{aligned}\text{LP1}:\quad &\min_{\boldsymbol{\alpha},\boldsymbol{\xi}}\quad \sum_{i=1}^{\ell}\sum_{\substack{m=1,\\ m\neq q}}^{c}\xi_{im}^q\\ \text{s.t.}\quad &\sum_{j=1}^{N}\alpha_j=1\\ &1(\mathbf{x}_i\in\omega_q)\left[\sum_{j=1}^{N}\alpha_j f_{jq}(\mathbf{x}_i)-\sum_{j=1}^{N}\alpha_j f_{jm}(\mathbf{x}_i)\right]\geq\varepsilon-\xi_{im}^q,\\ &\alpha_j\geq 0,\quad \xi_{im}^q\geq 0,\ m\neq q,\ m=1,\ldots,c,\ i=1,\ldots,\ell\end{aligned} \tag{12}$$

where $\varepsilon\geq 0$ is a constant chosen by users, $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\ldots,\alpha_N]^T$, $\boldsymbol{\xi}$ is the column vector spanned by ξ_{im}^q, $i=1,\ldots,\ell$, $m=1,\ldots,c$, $m\neq q$, and the function $1(\mathbf{x}_i\in\omega_q)$ is defined as

$$1(\mathbf{x}_i\in\omega_q)=\begin{cases}1 & \text{if } \mathbf{x}_i\in\omega_q\\ 0 & \text{otherwise}\end{cases} \tag{13}$$

In LP1 (12), the objective function can be regarded as the hinge loss. Namely if $1(\mathbf{x}_i\in\omega_q)=1$, then slack variables $\xi_{im}^q=\max\{0,\varepsilon+(f_m(\mathbf{x}_i)-f_q(\mathbf{x}_i))\}$, $m\neq q$. The hinge loss has a zero-valued interval of $[\varepsilon,+\infty)$, on which the loss takes zero value. In other words, if $f_q(\mathbf{x})_i-f_m(\mathbf{x})_i\geq\varepsilon$, then $\xi_{im}^q=0$, which means $\boldsymbol{\xi}$ would be sparse. A typical example of implementing the hinge loss to get a sparse model representation is SVMs for classification [33,34]. The learned models in SVMs exhibit obvious sparseness [35,36], the decision function is only dependent on support vectors. At the same time, $\boldsymbol{\alpha}$ in the optimal solution is also sparse as the case of LP-Adaboost. Note that α_j is the weight coefficient of the j-th individual classifier. If and only if $\alpha_j>0$, the corresponding individual classifier is selected to be one effective individual classifier. Thus we implement sparse ensembles by combining classifiers with only positive weight coefficients.

Now we make a comparison between LP-Adaboost and LP1. As mentioned before, LP-Adaboost is also a combination method based on LP for sparse ensembles [18]. But the goal of LP-Adaboost is to minimize maximum margin in [18], which is different from ours. In [18], the margin of the training sample $\mathbf{x}_i$ is defined as $\gamma_i=\sum_{j=1}^{N}\alpha_j z_{ij}=\boldsymbol{\alpha}^T\mathbf{z}_i$, where $z_{ij}=1$ if $h_j(\mathbf{x}_i)=y_i$ and $z_{ij}=-1$ if $h_j(\mathbf{x}_i)\neq y_i$, and $h_j(\mathbf{x}_i)$ are the classification results of the j-th classifier on $\mathbf{x}_i$. LP-Adaboost is to maximize γ, subject to $\boldsymbol{\alpha}^T\mathbf{z}_i\geq\gamma$, $\sum_{j=1}^{N}\alpha_j=1$ and $\alpha_j\geq 0$, $j=1,\ldots,N$. The margin γ_i can be regarded as a measurement for classification performance of all classifiers on $\mathbf{x}_i$. Thus, LP-Adaboost is to find the weight vector by maximizing the classification performance of the hardest sample. In LP1, we put focus on the total ensemble training error instead of the classification performance of individual classifiers. For each training sample, its ensemble output on its own class should be the largest among the ensemble outputs on all classes. Thus, the weight vector is adjusted to get good ensemble outputs for training samples.

3.2. LP2 method

If we put weights α_j into the objective function and delete the equality constraint in LP1, we can obtain another LP formula as follows:

$$\begin{aligned}\text{LP2}:\quad &\min_{\boldsymbol{\alpha},\boldsymbol{\xi}}\quad \sum_{j=1}^{N}\alpha_j+C\sum_{i=1}^{\ell}\sum_{\substack{m=1,\\ m\neq q}}^{c}\xi_{im}^q\\ \text{s.t.}\quad &1(\mathbf{x}_i\in\omega_q)\left[\sum_{j=1}^{N}\alpha_j f_{jq}(\mathbf{x}_i)-\sum_{j=1}^{N}\alpha_j f_{jm}(\mathbf{x}_i)\right]\geq\varepsilon-\xi_{im}^q\\ &\alpha_j\geq 0,\quad \xi_{im}^q\geq 0,\ m\neq q,\ m=1,\ldots,c,\ i=1,\ldots,\ell\end{aligned} \tag{14}$$

where $C>0$ is the penalty factor and $\varepsilon>0$ is any constant.

In LP2 (14), the first term $\sum_{j=1}^{N}\alpha_j$ is the 1-norm regularization and the second term $\sum_{i=1}^{\ell}\sum_{m=1,\ m\neq q}^{c}\xi_{im}^q$ is the hinge loss. Both of them are sparseness techniques. In fact, the 0-norm regularization is the desirable one to obtain sparseness, but the 0-norm regularization is so discontinuous that it is difficult to optimize the objective function. As an approximation of the 0-norm regularization, the 1-norm regularization can also induce sparseness and is segment-wise differentiable to make the optimization possible. A good example of using both two sparseness techniques to implement a sparse model representation is 1-norm SVMs [37–42]. It has been shown that 1-norm SVMs have better sparseness than SVMs due to the adoption of two sparseness techniques [21]. Clearly, the solution of LP2 (14) is sparse. We can also implement sparse ensembles by combining the individual classifiers with positive weight coefficients (or $\alpha_j>0$).

When we employ LP2 (14) to find the coefficients of N individual classifiers, we have the following theorem about the selection of the constant ε.

Theorem 1. *When ε takes two positive constants, say $\varepsilon_1>0$ and $\varepsilon_2>0$, LP2 (14) gives two optimal solutions $((\boldsymbol{\alpha})_1^*,(\boldsymbol{\xi})_1^*)$ and $((\boldsymbol{\alpha})_2^*,(\boldsymbol{\xi})_2^*)$, respectively, then $((\boldsymbol{\alpha})_1^*,(\boldsymbol{\xi})_1^*)$ and $((\boldsymbol{\alpha})_2^*,(\boldsymbol{\xi})_2^*)$ are rescalings of the same optimal solution.*

The proof of Theorem 1 is given in Appendix A. Theorem 1 shows that the various values of ε have no effect on the classification results. An unseen sample $\mathbf{x}$, for example, is assigned to class ω_q if the optimal solution $(\boldsymbol{\alpha})_1^*$ is taken as the coefficients of individual classifiers. This sample is also assigned to the same class ω_q if $(\boldsymbol{\alpha})_2^*$ is adopted to combining N individual classifiers.

3.3. LP3 method

While in LP problems (12) and (14), weights are constrained to be nonnegative. In ensemble learning, it is required that individual classifiers are good weak ones whose performance is better than that of random guess. Poor weak classifiers do not perform better than random guess and affect the performance of ensemble learning. In order to avoid this, we expect the coefficients of poor individual classifiers to be negative. In doing so, poor individual classifiers would play a positive role in ensembles. Hence, we construct a LP formula in which weights are unrestricted in sign. Let $\alpha_j=\beta_j^+-\beta_j^-$. Then we can get

$$\begin{aligned}\text{LP3}:\quad &\min_{\boldsymbol{\alpha},\boldsymbol{\xi}}\quad \sum_{j=1}^{N}(\beta_j^++\beta_j^-)+C\sum_{i=1}^{\ell}\sum_{\substack{m=1,\\ m\neq q}}^{c}\xi_{im}^q\\ \text{s.t.}\quad &1(\mathbf{x}_i\in\omega_q)\left[\sum_{j=1}^{N}(\beta_j^+-\beta_j^-)f_{jq}(\mathbf{x}_i)-\sum_{j=1}^{N}(\beta_j^+-\beta_j^-)f_{jm}(\mathbf{x}_i)\right]\geq\varepsilon-\xi_{im}^q\end{aligned}$$

$$\beta_j^+ \geq 0, \quad \beta_j^- \geq 0, \ \xi_{im}^q \geq 0, \ m \neq q, \ m = 1,\ldots,c, \ i = 1,\ldots,\ell \tag{15}$$

In LP3 (15), the first term is also the 1-norm regularization term and the second term is the hinge loss. Similar to LP2, LP3 also uses two sparseness techniques. In LP3, however, weight coefficients may be negative. Thus individual classifiers with nonzero weight coefficients (or $\alpha_j \neq 0$) are considered in the ensemble. We have the similar theorem about the constant ε in LP3 (15).

Theorem 2. *When ε takes two positive constants, say $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$, LP3 (15) gives two optimal solutions $((\boldsymbol{\beta}^+)_1^*,(\boldsymbol{\beta}^-)_1^*,(\boldsymbol{\xi})_1^*)$ and $((\boldsymbol{\beta}^+)_2^*,(\boldsymbol{\beta}^-)_2^*,(\boldsymbol{\xi})_2^*)$, respectively, then $((\boldsymbol{\beta}^+)_1^*,(\boldsymbol{\beta}^-)_1^*,(\boldsymbol{\xi})_1^*)$ and $((\boldsymbol{\beta}^+)_2^*,(\boldsymbol{\beta}^-)_2^*,(\boldsymbol{\xi})_2^*)$ are rescalings of the same optimal solution.*

The proof of Theorem 2 is similar to that of Theorem 1, so it is omitted. However, it is argued whether the unrestricted weights can result good performance. In theory, it could be better using unrestricted in sign, but the weights cannot be reliably estimated in most cases [16]. We will see in Section 4, the experimental results of LP3 is not so good as we expected.

From the three LP problems (12), (14) and (15), we can see the ensemble training error is minimized while simultaneously the capacity of ensemble learning (or the weight vector) is controlled. Therefore, these methods can be roughly thought as implementing the structure risk minimization rule. LP1 (12), LP2 (14) and LP3 (15) can be solved by classical methods such as Newton method, the column generation algorithm, and the simplex method [43]. We will not develop this topic further here. Interested readers may refer to [43] for details.

4. Simulation

In order to validate the performance of our linear weighted combination methods, experiments on UCI data sets [44] and radar target images [45,46] are performed. All numerical experiments are performed on the personal computer with a 1.8 GHz Pentium III and 1 G bytes of memory. This computer runs on Windows XP, with Matlab 7.1 installed.

4.1. Individual classifier and combination methods

The k NN classifier which employs the Euclidean distance as a distance measurement is considered as an individual classifier in the ensemble. It turns out that sampling the training set is not effective in k NN classifier ensembles [5,6]. However, the k NN methods are sensitive to input features [9], and to the chosen distance metric [47,48]. Bay [9] proposes an efficient way to combining k NN classifiers through multiple feature subsets (MFS). Here, we use MFS to get the diversity of k NN classifiers. Moreover, experimental results in [9] showed that both sampling with replacement and sampling without replacement have the similar performance. In our experiments, the random subset of features X_j are selected by sampling with replacement from the original set X, all d_j, $j=1,\ldots,N$ are equal to each other, and smaller than or equal to D. Namely k NN classifiers share the same value of d. Define the outputs of the jth individual k NN classifier to be

$$f_{jm}(\mathbf{x}) = \frac{k_m}{k}, \quad m = 1,\ldots,c \tag{16}$$

where k is the number of nearest neighbors, k_m is the number of nearest neighbors belonging to the class ω_m, and $\sum_{m=1}^{c} k_m = k$. Actually, the expression k_m/k can also be regarded as the discriminant function. To adopt the SV rule, we have

$$f_{jq}(\mathbf{x}) = \begin{cases} 1 & \text{if } k_q = \max\limits_{m=1,\ldots,c} k_m \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

We compare the accuracy of linear weighted averaging methods based on LP1 (12) (WV-LP1), LP2 (14) (WV-LP2) and LP3 (15) (WV-LP3) with the following methods.

1. Single k NN method with parameter k. There is no combination rule used in this method. So we call this method "None" in terms of combination rules.
2. Ensembles with two naive Bayes combination rules including the product rule (5) and the sum rule (6), with parameters N, k and d.
3. Ensembles with SV (7) with parameters N, k and d.
4. Ensembles with two linear combination rules, including SWA with parameters N, k and d, and the MCE criterion ([12]) with parameters N, k, d, η and ζ.
5. Sparse ensemble with LP-Adaboost [18] in which N, k and d are parameters.
6. Pruned ensemble with MDM [26] in which parameters are N, k and d.

WV-LP1 has parameters N, k, d, and ε, while both WV-LP2 and WV-LP3 have parameters N, k, d, and C.

For all ensembles in our experiments, the number of classifiers is set to $N=100$ as a reasonable trade-off between computational complexity and accuracy [9]. Other parameters, such as the size of the feature subsets and the value of k are selected by the cross-validation method on the training set [7]. The setting of other parameters is given in the following.

1. The value of k is selected from {1, 4, 7, 10, 13, 16, 19}.
2. In the classifier ensembles, the size of the feature subsets are closely related to the dimension of data. Let the size of original features be D. The size of feature subsets is selected from $\{\lfloor 0.1(D-1)\rfloor,\lfloor 0.2(D-1)\rfloor,\ldots,\lfloor 0.9(D-1)\rfloor,(D-1)\}$, where $\lfloor\cdot\rfloor$ is a floor function.
3. In WV-MCE, the parameter η is selected from $\{2^{-3},2^{-2},\ldots,2^3\}$ and ζ is selected from {10,20,30,40,50,60}.
4. In the WV-LP1 method, ε is selected from $\{2^{-9},2^{-8},\ldots,2^0\}$.
5. For LP2 and LP3 methods, penalty factor C is selected from $\{2^{-5},2^{-4},\ldots,2^4\}$.

Theorems 1 and 2 tell us that the value of ε in LP2 and LP3 is not so important. Thus we take $\varepsilon = 0.1$ in our experiments.

4.2. Experiments on UCI data sets

We use 14 data sets from the UCI database [44]. The second column in Table 1 presents some attribute of these data sets, where ℓ is the number of samples, D is the feature number of samples, and c is the number of classes. These data sets are normalized so that continuous features ranged in the interval [0,1]. For each data set, we run 10 trials where the training set contains $\frac{2}{3}$ of samples (randomly selected) of each class, and the test set contains the remaining $\frac{1}{3}$. In each trial, the 10-fold cross-validation method is applied to the training set to choose optimal parameters. Although we have the optimal parameter d^*, we do not know which features should be chosen. Hence we randomly select d^* features for both training and test set in each trial and perform 10 random selection.

Table 1 also gives the average ensemble classifiers numbers of all ensemble methods. Note that the ensemble size $N=100$. We can see that the first five methods use all 100 classifiers in the

ensemble. Moreover, the WV-MCE method uses $c \times N$ nonzero weight coefficients in the ensemble. The other methods LP-Adaboost, MDM, and our three methods get much smaller numbers of nonzero weights than 100. These methods only utilize a small part of classifiers in the ensemble. From the index of the total average, the LP-Adaboost method has the best sparsity among them, followed by our LP methods.

The mean and standard deviation of classification error rates on test sets are reported in Table 2. The WV-LP1 method has the best performance on all sets except Liver, Pima, Wdbc and Wine data sets. The WV-LP2 has the best performance on Liver and Pima sets, and WV-MCE on the Wdbc and Wine sets. Fig. 2 shows the average classification errors on all 14 data sets. From this figure, we can see that the best average performance is obtained by the WV-LP1 method, followed by WV-MCE and WV-LP2 methods. By observing the classification results of three LP methods, we can see that WA-LP1 is the best. Originally, we expected WA-LP3 would be a good one because weight coefficients are not constrained to be positive. However, empirical results show it is unreliable.

Linear weighted averaging methods in both sparse or non-sparse ensembles need time to find the weight coefficients, and pruned ensembles also need additional time to select optimal sub-ensembles. As we stated before, pruned ensembles can be taken as a special sparse ensemble in which the weight coefficients of selected classifiers have the value one. Thus, for the sake of convenience, the additional time is called time for finding the weight coefficients. Table 3 reports the additional time for some methods in our experiments. WV-MCE has good performance, but we can see it takes a long time to find weight

Table 1
Comparison of ensemble classifier numbers.

Data set	$\ell/D/c$	Product Sum SV SWA WV-MCE	LP-Adaboost	MDM	WV-LP1	WV-LP2	WV-LP3
Breast	699/9/2	100	1.05	55.31	5.49	9.87	12.29
Glass	213/9/6	100	3.88	40.33	17.61	16.81	17.33
Heart-Cleveland	303/13/2	100	8.46	19.44	13.11	13.16	20.47
Hepatitis	155/19/2	100	12.70	17.09	14.64	7.54	5.12
Ionosphere	351/32/2	100	6.28	20.37	14.64	19.93	28.87
Liver	345/6/2	100	2.70	43.59	8.90	6.00	4.85
Musk	476/166/2	100	19.65	22.00	22.49	23.20	34.56
Pima	768/8/2	100	5.93	21.61	11.61	5.66	8.85
Sonar	208/60/2	100	12.16	25.85	16.74	16.33	28.76
Vehicle	846/18/4	100	22.71	15.94	23.59	17.81	65.61
Vote	435/16/2	100	2.28	9.52	8.93	8.81	13.63
Wdbc	569/30/2	100	1.34	8.26	7.27	8.78	20.21
Wine	178/13/3	100	9.84	14.46	5.95	7.38	8.47
Wpbc	198/33/2	100	1.00	9.34	4.56	7.26	20.05
Total average		100	7.86	23.08	12.54	12.04	20.65

Note: ℓ is the number of total training sample, D is the dimensionality of sample space, and c is the class number.

Table 2
Mean and standard deviation of classification error rates (%) on test sets of UCI database.

Combination rule	Breast	Glass	Heart-Cleveland	Hepatitis	Ionosphere	Liver	Musk
None	3.49 ± 1.03	30.29 ± 5.40	22.00 ± 3.53	37.65 ± 7.44	14.53 ± 2.58	38.51 ± 3.11	14.62 ± 3.61
Product	7.96 ± 3.36	35.91 ± 6.85	33.05 ± 16.38	36.63 ± 7.38	19.09 ± 3.70	36.84 ± 4.90	28.97 ± 7.19
Sum	3.10 ± 0.78	30.74 ± 5.86	17.83 ± 4.03	36.86 ± 7.34	13.19 ± 2.65	36.93 ± 4.31	10.46 ± 3.78
SV	3.28 ± 0.94	35.45 ± 6.98	22.77 ± 9.08	36.49 ± 7.58	13.58 ± 2.61	38.24 ± 1.692	10.54 ± 4.02
SWA	3.15 ± 0.81	30.62 ± 5.70	18.03 ± 3.91	37.14 ± 7.43	13.18 ± 2.53	37.04 ± 4.39	10.43 ± 3.80
WV-MCE	3.12 ± 1.23	21.91 ± 5.52	18.30 ± 3.30	34.86 ± 5.58	6.10 ± 1.44	34.10 ± 4.17	9.97 ± 3.41
LP-Adaboost	5.79 ± 4.18	43.09 ± 11.26	22.96 ± 3.17	36.24 ± 6.68	13.61 ± 2.60	38.13 ± 3.43	10.92 ± 2.24
MDM	3.33 ± 1.37	20.70 ± 4.03	19.73 ± 3.10	37.55 ± 3.90	6.62 ± 1.92	33.18 ± 2.18	9.23 ± 2.21
WV-LP1	$\mathbf{2.90} \pm 1.02$	$\mathbf{18.35} \pm 3.73$	$\mathbf{16.78} \pm 3.65$	$\mathbf{34.78} \pm 7.90$	$\mathbf{5.71} \pm 1.24$	32.99 ± 3.40	$\mathbf{7.85} \pm 3.42$
WV-LP2	3.93 ± 1.23	20.61 ± 4.45	17.78 ± 3.97	35.92 ± 5.26	7.56 ± 1.30	$\mathbf{32.22} \pm 3.77$	9.44 ± 2.45
WA-LP3	3.82 ± 1.32	22.45 ± 4.47	19.73 ± 3.18	34.84 ± 6.79	8.74 ± 2.57	32.95 ± 4.00	12.25 ± 2.22
	Pima	Sonar	Vehicle	Vote	Wdbc	Wine	Wpbc
None	26.27 ± 1.91	15.22 ± 2.75	31.46 ± 2.16	7.31 ± 1.27	3.02 ± 1.39	2.24 ± 1.42	22.46 ± 2.83
Product	24.65 ± 2.42	50.13 ± 23.47	32.40 ± 4.86	8.74 ± 1.74	5.97 ± 4.31	2.93 ± 2.38	25.09 ± 9.18
Sum	25.71 ± 3.04	16.04 ± 4.60	29.44 ± 2.90	19.68 ± 14.25	3.44 ± 1.16	2.31 ± 1.41	21.09 ± 3.73
SV	25.87 ± 2.66	19.42 ± 5.96	30.93 ± 2.54	13.94 ± 13.14	3.52 ± 1.72	2.57 ± 1.26	21.80 ± 4.28
SWA	25.95 ± 3.34	15.88 ± 4.26	29.50 ± 2.99	19.63 ± 14.31	3.53 ± 1.12	2.28 ± 1.40	21.18 ± 3.63
WA-MCE	25.50 ± 2.17	13.78 ± 4.00	28.03 ± 1.56	6.70 ± 1.65	$\mathbf{2.34} \pm 1.45$	$\mathbf{0.93} \pm 1.16$	19.43 ± 2.38
LP-Adaboost	27.87 ± 3.11	18.86 ± 6.05	39.22 ± 10.89	15.48 ± 10.68	4.59 ± 1.41	3.38 ± 2.09	27.06 ± 3.85
MDM	24.76 ± 2.16	14.77 ± 3.87	27.20 ± 1.45	6.23 ± 1.22	2.98 ± 1.29	1.90 ± 1.44	19.63 ± 1.17
WV-LP1	24.62 ± 1.98	$\mathbf{10.88} \pm 3.07$	$\mathbf{25.38} \pm 2.09$	$\mathbf{5.23} \pm 1.55$	2.38 ± 1.31	2.19 ± 2.08	$\mathbf{16.65} \pm 1.11$
WV-LP2	$\mathbf{24.20} \pm 1.60$	14.07 ± 3.45	27.49 ± 0.84	5.86 ± 0.59	2.79 ± 1.10	2.31 ± 2.51	20.95 ± 1.69
WV-LP3	25.10 ± 1.58	15.75 ± 3.68	27.43 ± 2.09	6.08 ± 1.58	3.02 ± 1.41	2.34 ± 2.69	22.89 ± 2.85

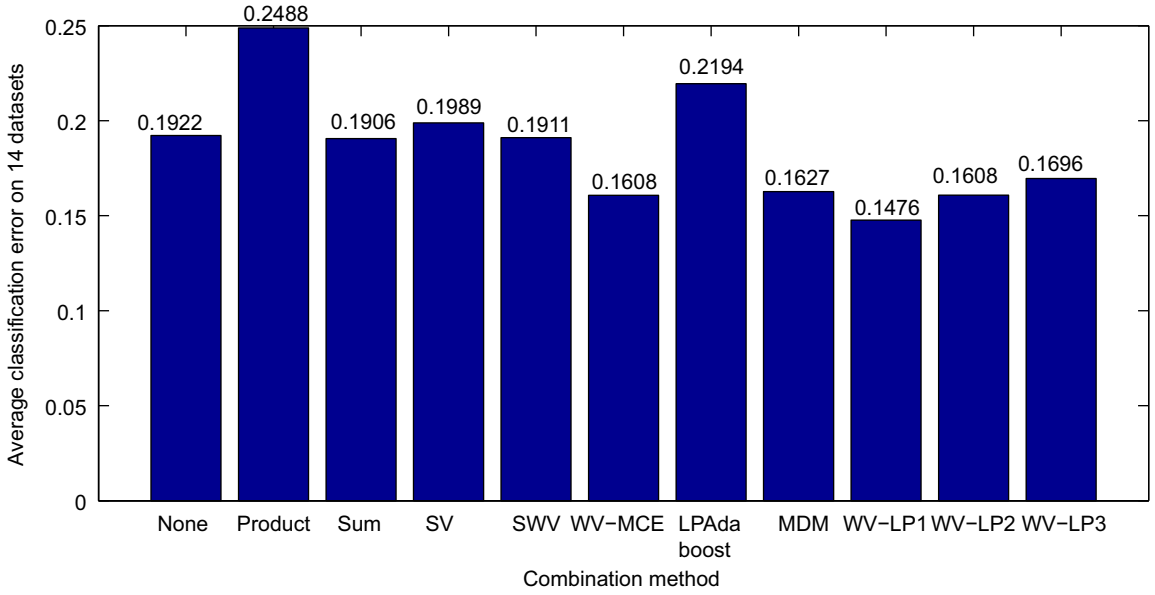

Fig. 2. Average classification error on all 14 UCI data sets.

Table 3
Comparison of average time for finding weight coefficients.

Data set	WV-MCE	LP-Adaboost	MDM	WV-LP1	WV-LP2	WV-LP3
Breast	18.46	0.019	0.46	0.53	0.53	0.67
Glass	49.15	0.0095	0.28	1.33	1.35	1.56
Heart-Cleveland	12.71	0.016	0.21	0.26	0.25	0.37
Hepatitis	0.85	0.0092	0.041	0.034	0.020	0.02
Ionosphere	4.65	0.041	0.36	0.12	0.15	0.49
Liver	3.83	0.032	0.11	0.087	0.066	0.10
Musk	9.29	0.076	0.43	0.24	0.34	0.52
Pima	14.05	0.097	0.37	0.32	0.21	0.36
Sonar	3.53	0.027	0.15	0.066	0.061	0.13
Vehicle	85.56	0.19	0.23	5.75	4.04	10.42
Vote	1.91	0.048	0.051	0.14	0.11	0.24
Wdbc	7.07	0.058	0.26	0.18	0.19	0.53
Wine	1.52	0.015	0.17	0.11	0.13	0.30
Wpbc	3.66	0.017	0.083	0.046	0.041	0.081

coefficients. LP-Adaboost is fast, but its performance is bad. Other methods, MDM, WV-LP1, WV-LP2, and WV-LP3 take a time between LP-Adaboost and WV-MCE for finding weight coefficients. For instance, WV-LP1 has a slower speed and a better performance than LP-Adaboost, and has a faster speed than WV-MCE on UCI data sets.

4.3. Experiments on radar high-resolution range profile data

High-resolution range profile (HRRP) is the amplitude of the coherent summations of the complex time returns from target scatterers in each range cell, which represents the projection of the complex returned echoes from the target scattering centers onto the radar line-of-sight (LOS) [45,46]. HRRP contains the target structure signatures, including target size, scatterer distribution, etc. Here, the HRRP data is measured airplane data as in [45,46]. There are three airplanes including Yark-42, An-26 and Cessna Citation S/II. The parameters of three airplanes and radar are presented in Table 4, and the projections of airplane trajectories onto ground plane are shown in Fig. 3 from which we can know that the measured data is segmented, and can estimate the aspect angle of an airplane according to its relative position to radar. Training samples and test samples are selected from

Table 4
Parameters of planes and radar in the inverse synthetic aperture radar experiment.

Radar parameters	Center frequency Bandwidth		5520 MHz 400 HMz
Planes	Length (m)	Width (m)	Height (m)
Yark-42	36.38	34.88	9.83
An-26	23.80	29.20	9.83
Cessna citation S/II	14.40	15.90	

different data segments. The selection scheme is the same as that in [45,46]. The training samples are from the second and fifth segments of the Yark-42, the fifth and sixth segments of the An-26, and the sixth and seventh segments of the Cessna Citation S/II. The remaining data segments are taken as the test samples. The training samples cover almost all of the target-aspect angles, but their elevation angles are different from those of the test data.

In the training process, we adopt the 10-fold cross-validation method to choose the optimal parameters. These resulting optimal parameters are applied to the test procedure. In the same way, we randomly select d features in both training and test sets and perform 10 random runs. The results (the average test error rate over these 10 runs, the number of ensemble classifiers and the time for finding weight coefficients) are given in Table 5. The observation on Table 5 indicates that the WV-LP1 has the best classification performance, followed by WV-LP2 and WV-LP3. Our methods are comparable to other two methods in sparsity. The time for finding weight coefficients is about one second in this experiment.

5. Conclusions

This paper deals with linear weighted combination methods based on LP, which are applied to sparse ensembles. In ensembles, we consider minimizing the ensemble training error in terms of all learned individual classifiers. The problem can be cast into linear programming problems in which the hinge loss or/and the 1-norm regularization are adapted. Both techniques can induce a sparse solution. The optimization goal of these LP-based methods

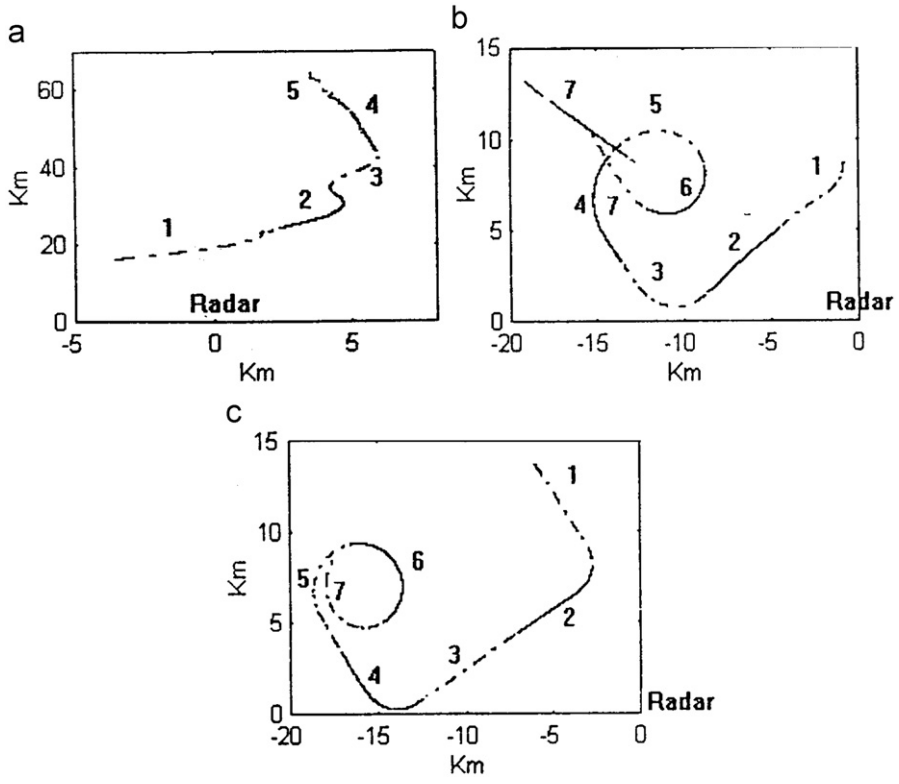

Fig. 3. Projections of three plane trajectories onto ground plane: (a) Yark-42, (b) An-26, and (c) Cessna Citation S/II.

Table 5
Comparison of average ensemble classifier numbers.

Combination rule	Test error rate (%)	♯ Ensemble classifier	Time for finding weight coefficients (s)
None	36.08 ± 0.00	–	–
Product	83.16 ± 0.43	100.00	–
Sum	18.71 ± 0.81	100.00	–
SV	18.80 ± 0.75	100.00	–
SWA	18.79 ± 0.57	100.00	–
WV-MCE	18.76 ± 0.55	100.00	4.1184
LP-Adaboost	22.33 ± 2.72	17.40	0.0546
MDM	15.63 ± 1.68	8.10	0.0172
WV-LP1	$\mathbf{14.56} \pm 1.08$	14.10	0.7472
WV-LP2	14.96 ± 1.08	17.30	0.7847
WV-LP3	15.27 ± 1.97	17.10	1.2199

is to minimize the ensemble training error and to control the weight vector of ensemble learning, which accounts for their good performance. Our combination rules can be applied to the ensemble of any classifiers if posterior probabilities or discriminant values of these classifiers can be obtained. In experiments, we compare our methods with other methods by ensembling k NN classifiers. Experimental results on UCI data sets and the radar high-resolution range profile data confirm the validity of our rules. Our methods have a promising sparseness and generalization performance. Especially the WV-LP1 method behaves very well in the most of data sets investigated here.

Acknowledgments

The authors would like to thank Journal Manager S. Doman and the two anonymous reviewers for their valuable comments and suggestions, which have helped to improve the quality of this paper significantly. This work was supported in part by the National Natural Science Foundation of China under Grant nos. 60970067, 60602064 and 60872135.

Appendix A. The proof of Theorem 1

Proof. Assume that $((\boldsymbol{\alpha})_1^*,(\boldsymbol{\xi})_1^*)$ is the optimal solution of the following LP for $\varepsilon=\varepsilon_1$.

$$\begin{aligned}\min\quad & L((\boldsymbol{\alpha})_1,(\boldsymbol{\xi})_1)=\sum_{j=1}^{N}(\alpha_j)_1+C\sum_{i=1}^{\ell}\sum_{m=1,\ m\neq q}^{c}(\xi_{im}^q)_1\\ \text{s.t.}\quad & 1(\mathbf{x}_i\in\omega_q)\left[\sum_{j=1}^{N}(\alpha_j)_1P_j(\omega_q|(\mathbf{x}_i)^j)-\sum_{j=1}^{N}(\alpha_j)_1P_j(\omega_m|(\mathbf{x}_i)^j)\right]\geq\varepsilon_1-(\xi_{im}^q)_1\\ & (\alpha_j)_1\geq 0,\quad (\xi_{im}^q)_1\geq 0,\ m\neq q,\ m=1,\ldots,c,\ i=1,\ldots,\ell\end{aligned}\tag{18}$$

where $(\alpha_j)_1$ and $(\xi_{im}^q)_1$ are components of $(\boldsymbol{\alpha})_1$ and $(\boldsymbol{\xi})_1$, respectively.

For $\varepsilon=\varepsilon_2$, LP2 can be rewritten as

$$\begin{aligned}\min\quad & L((\boldsymbol{\alpha})_2,(\boldsymbol{\xi})_2)=\sum_{j=1}^{N}(\alpha_j)_2+C\sum_{i=1}^{\ell}\sum_{m=1,\ m\neq q}^{c}(\xi_{im}^q)_2\\ \text{s.t.}\quad & 1(\mathbf{x}_i\in\omega_q)\left[\sum_{j=1}^{N}(\alpha_j)_2P_j(\omega_q|(\mathbf{x}_i)^j)-\sum_{j=1}^{N}(\alpha_j)_2P_j(\omega_m|(\mathbf{x}_i)^j)\right]\geq\varepsilon_2-(\xi_{im}^q)_2\\ & (\alpha_j)_2\geq 0,\quad (\xi_{im}^q)_2\geq 0,\ m\neq q,\ m=1,\ldots,c,\ i=1,\ldots,\ell\end{aligned}\tag{19}$$

Now we multiply two sides of inequality constraints of (19) by $\varepsilon_1/\varepsilon_2$, and multiply the objective function of (19) by $\varepsilon_1\varepsilon_2/\varepsilon_2\varepsilon_1$.

There get

$$\begin{aligned}\min\quad & L((\boldsymbol{\alpha})_2,(\boldsymbol{\xi})_2)=\frac{\varepsilon_2}{\varepsilon_1}\left(\sum_{j=1}^{N}\frac{\varepsilon_1}{\varepsilon_2}(\alpha_j)_2+C\sum_{i=1}^{\ell}\sum_{m=1,\ m\neq q}^{c}\frac{\varepsilon_1}{\varepsilon_2}(\xi_{im}^{q})_2\right)\\ \text{s.t.}\quad & (\mathbf{x}_i\in\omega_q)\left[\sum_{j=1}^{N}\frac{\varepsilon_1}{\varepsilon_2}(\alpha_j)_2P_j(\omega_q|(\mathbf{x}_i)^j)-\sum_{j=1}^{N}\frac{\varepsilon_1}{\varepsilon_2}(\alpha_j)_2P_j(\omega_m|(\mathbf{x}_i)^j)\right]\\ & \geq\varepsilon_1-\frac{\varepsilon_1}{\varepsilon_2}(\xi_{im}^{q})_2\quad(\alpha_j)_2\geq 0,\\ & (\xi_{im}^{q})_2)\geq 0,\ m\neq q,\ m=1,\ldots,c,\ i=1,\ldots,\ell\end{aligned}\tag{20}$$

Observing LPs (18) and (20), we get the optimal solution of (20)

$$\frac{\varepsilon_1}{\varepsilon_2}(\alpha_j)_2^*=(\alpha_j)_1^*,\quad j=1,\ldots,N\tag{21}$$

and

$$\frac{\varepsilon_1}{\varepsilon_2}(\xi_{im}^{q})_2^*=(\xi_{im}^{q})_1^*,\quad m\neq q,\ m=1,\ldots,c,\ i=1,\ldots,\ell,\tag{22}$$

That is,

$$(\boldsymbol{\alpha})_2^*=\frac{\varepsilon_2}{\varepsilon_1}(\boldsymbol{\alpha})_2^*\tag{23}$$

and

$$(\boldsymbol{\xi})_2^*=\frac{\varepsilon_2}{\varepsilon_1}(\boldsymbol{\xi})_1^*\tag{24}$$

where $((\boldsymbol{\alpha})_2^*,(\boldsymbol{\xi})_2^*)$ is the optimal solution of (19) in matrix form. Hence $((\boldsymbol{\alpha})_1^*,(\boldsymbol{\xi})_1^*)$ and $((\boldsymbol{\alpha})_2^*,(\boldsymbol{\xi})_2^*)$ are rescalings of the same optimal solution.

In addition the optimal objective function value

$$L((\boldsymbol{\alpha})_2^*,(\boldsymbol{\xi})_2^*)=\frac{\varepsilon_2}{\varepsilon_1}L((\boldsymbol{\alpha})_1^*,(\boldsymbol{\xi})_1^*)\tag{25}$$

This completes the proof of Theorem 1. □

References

[1] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, Hoboken, NJ, 2004.
[2] T. Windeatt, F. Roli (Eds.), Multiple Classifier Systems, in: Lecture Notes in Computer Science, vol. 2709, Springer, 2003.
[3] F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems, in: Lecture Notes in Computer Science, vol. 3077, Springer, 2004.
[4] E.K. Tang, P.N. Suganthan, X. Yao, An analysis of diversity measures, Machine Learning 65 (2006) 247–271.
[5] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123–140.
[6] Y. Freund, R. Shapire, Experiments with a new boosting algorithm, in: 13th International Conference on Machine Learning, Bary, Italy, Morgan Kaufmann, 1996, pp. 148–156.
[7] R. Duda, P. Hart, D. Stork, Pattern Classification, second ed., John Wiley & Sons, 2000.
[8] T.K. Ho, J.J. Hull, S.N. Srihari, Decision combination in multiple classifier systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (1) (1994) 66–75.
[9] S.D. Bay, Combining nearest neighbor classifiers through multiple feature subsets, in: Proceedings of the 15th International Conference on Machine Learning, Madison, WI, 1998, pp. 37–45.
[10] J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3) (1998) 226–239.
[11] J.A. Benediktsson, J.R. Sveinsson, O.K. Ersoy, P.H. Swain, Parallel consensual neural networks, IEEE Transactions on Neural Networks 8 (1) (1997) 54–64.
[12] N. Ueda, Optimal linear combination of neural networks for improving classification performance, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2) (2000) 207–215.
[13] L.I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 281–286.
[14] V. Tresp, M. Taniguchi, Combining estimators using non-constant weighting functions, Advances in Neural Information Processing Systems, vol. 7, 1995, pp. 419–426.
[15] L. Breiman, Stacked regressions, Machine Learning 24 (1996) 49–64.
[16] G. Fumera, F. Roli, A theoretical and experimental analysis of linear combiners for multiple classifier systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (6) (2005) 942–956.
[17] X. Yao, Y. Liu, Making use of population information in evolutionary artificial neural networks, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 28 (3) (1998) 417–425.
[18] A.J. Grove, D. Schuurmans, Boosting in the limit: maximizing the margin of learned ensembles, in: Proceedings of the 15th National Conference on Artificial Intelligence, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1998, pp. 692–699.
[19] T. Graepel, R. Herbrich, J. Shawe-Taylor, Generalization error bounds for sparse linear classifiers, in: 13th Annual Conference on Computational Learning Theory, 2000, pp. 298–303.
[20] S. Floyd, M. Marmuth, Sample compression learnability, and the Vapnik–Chervonenkis dimension, Machine Learning 21 (3) (1995) 269–304.
[21] L. Zhang, W.D. Zhou, On the sparseness of 1-norm support vector machines, Neural Networks 23 (3) (2010) 373–385.
[22] G. Martínez-Muñoz, D. Hernández-Lobato, A. Suárez, An analysis of ensemble pruning techniques based on ordered aggregation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 245–259.
[23] Z.H. Zhou, J.X. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artificial Intelligence 137 (1–2) (2002) 239–263.
[24] Y. Zhang, S. Burer, W.N. Street, Ensemble pruning via semi-definite programming, Journal of Machine Learning Research 7 (2006) 1315–1338.
[25] D.D. Margineantu, T.G. Dietterich, Pruning adaptive boosting, in: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 211–218.
[26] G. Martínez-Muñoz, A. Suárez, Aggregation ordering in bagging, in: Proceedings of the International Conference on Artificial Intelligence and Applications, 2004, pp. 258–263.
[27] G. Martínez-Muñoz, A. Suárez, Pruning in ordered bagging ensembles, in: Proceedings of the 23th International Conference on Machine Learning, 2006, pp. 609–616.
[28] G. Martínez-Muñoz, A. Suárez, Using boosting to prune bagging ensembles, Pattern Recognition Letters 28 (1) (2007) 156–165.
[29] H. Chen, P. Tino, X. Yao, Predictive ensemble pruning by expectation propagation, IEEE Transactions on Knowledge and Data Engineering 21 (7) (2009) 999–1013.
[30] G. Tsoumakas, I. Katakis, I.P. Vlahavas, Effective voting of heterogeneous classifiers, Proceedings of the 11th European Conference on Machine Learning, Lecture Notes in Artificial Intelligence, vol. 3201, 2004, pp. 465–476.
[31] G. Tsoumakas, L. Angelis, I. Vlahavas, Selective fusion of heterogeneous classifiers, Intelligent Data Analysis 9 (2005) 511–525.
[32] Z.H. Zhou, Y. Yu, Adapt bagging to nearest neighbor classifier, Journal of Computer Science and Technology 20 (2005) 48–54.
[33] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.
[34] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.
[35] F. Girosi, M. Jones, T. Poggio, Regularization theory and neural networks architectures, Neural Computation 7 (2) (1995) 219–269.
[36] F. Girosi, An equivalence between sparse approximation and support vector machines, Neural Computation 10 (6) (1998) 1455–1480.
[37] O. Mangasarian, Exact 1-norm support vector machines via unconstrained convex differentiable minimization, Journal of Machine Learning Research 7 (2006) 1517–1530.
[38] J. Zhu, S. Rosset, T. Hastie, R. Tibshirani, 1-norm support vector machines, Advances in Neural Information Processing Systems, Lecture Notes in Computer Science, vol. 16, MIT Press, Cambridge, MA, 2004, pp. 49–56.
[39] J. Bi, Y. Chen, J. Wang, A sparse support vector machine approach to region-based image categorization, in: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Lecture Notes in Computer Science, MIT Press, Cambridge, MA, 2005, pp. 1121–1128.
[40] J. Bi, K.P. Bennett, M. Embrechts, C.M. Breneman, M. Song, Dimensionality reduction via sparse support vector machines, Journal of Machine Learning Research 4 (2003) 1229–1243.
[41] G.M. Fung, O.L. Mangasarian, A feature selection Newton method for support vector machine classification, Computational Optimization and Applications 28 (2004) 185–202.
[42] W.D. Zhou, L. Zhang, L.C. Jiao, Linear programming support vector machines, Pattern Recognition 35 (12) (2002) 2927–2936.
[43] R. Venderbei, Linear Programming: Foundation and Extension, Academic Press, New York, 1996.
[44] P. Murphy, D. Aha, UCI machine learning repository, from 〈http://www.ics.uci.edu/∼mlearn/MLRepository.html〉, 1992.
[45] L. Du, H.W. Liu, Z. Bao, J.Y. Zhang, A two-distribution compounded statistical model for radar HRRP target recognition, IEEE Transactions on Signal Processing 54 (6) (2006) 2226–2238.
[46] B. Chen, H. Liu, Z. Bao, Optimizing the data-dependent kernel under a unified kernel optimization framework, Pattern Recognition 41 (6) (2008) 2107–2119.

[47] C. Domeniconi, J. Peng, D. Gunopulos, Locally adaptive metric nearest neighbor classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (9) (2002) 1281–1285.
[48] Y. Bao, N. Ishii, X. Du, Combining multiple k-nearest neighbor classifiers using different distance functions, Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, vol. 3177, Springer-Verlag, Berlin, 2004, pp. 634–641.

Li Zhang received the B.S. degree in 1997 and the Ph.D. degree in 2002 in Electronic Engineering from Xidian University, Xi'an, China. From 2003 to 2005, she was a Postdoctor at the Institute of Automation of Shanghai Jiao Tong University, Shanghai, China. Now, she is an Associate Professor at Xidian University. Her research interests have been in the areas of machine learning, pattern recognition, neural networks and intelligent information processing.

Weida Zhou received the B.S. in 1996 and the Ph.D. degree in 2003 in Electronic Engineering from Xidian University, Xi'an, China. He has been an Associate Professor at the School of Electronic Engineering at Xidian University, Xi'an, China since 2006. His research interests include machine learning, learning theory and intelligent information processing.