Ensembling Local Learners Through Multimodal Perturbation

Zhi-Hua Zhou, *Member, IEEE*, Yang Yu

***Abstract*— Ensemble learning algorithms train multiple component learners and then combine their predictions. In order to generate a strong ensemble, the component learners should be with high accuracy as well as high diversity. A popularly used scheme in generating accurate but diverse component learners is to perturb the training data with resampling methods, such as the bootstrap sampling used in Bagging. However, such a scheme is not very effective on local learners such as nearest neighbor classifiers because a slight change in training data can hardly result in local learners with big differences. In this paper, a new ensemble algorithm named FASBIR is proposed for building ensembles of local learners, which utilizes multimodal perturbation to help generate accurate but diverse component learners. In detail, FASBIR employs the perturbation on the training data with bootstrap sampling, the perturbation on the input attributes with attribute filtering and attribute subspace selection, and the perturbation on the learning parameters with randomly configured distance metrics. A large empirical study shows that FASBIR is effective in building ensembles of nearest neighbor classifiers, whose performance is better than that of many other ensemble algorithms.**

***Index Terms*— Machine Learning, Data Mining, Ensemble Learning, Multimodal Perturbation, Local Learner, Nearest Neighbor Classifier, Stable Base Learner.**

I. INTRODUCTION

ENSEMBLE learning algorithms train multiple component learners and then combine their predictions. Since the generalization ability of an ensemble could be significantly better than that of a single learner, ensemble learning has been a hot topic during the past years [13].

In general, an ensemble is built in two steps, that is, generating multiple component learners and then combining their predictions. According to the styles of training the component learners, current ensemble learning algorithms can be roughly categorized into two classes, that is, algorithms where component learners must be trained sequentially, and algorithms where component learners could be trained in parallel. The representative of the first category is AdaBoost [16], which sequentially generates a series of component learners where the training instances wrongly predicted by a component learner will play more important role in the training of its subsequent learner. Other representatives of this category include Arc-$x4$ [7], LogitBoost [17], etc. The representative of the second category is Bagging [6], which generates many samples from the original training set via *bootstrap sampling* [15] and then trains a component learner from each of these samples, whose predictions are combined via *majority voting*. Other representatives of this category include Random Forest [9], Randomization [12], etc.

The Bagging algorithm has achieved great success in building ensembles of decision trees and neural networks. A recent empirical study [18] on ensembles of C4.5 decision trees disclosed that although many ensemble learning algorithms where component learners could be trained in parallel have been developed, few of them significantly outperforms Bagging. However, as Breiman indicated [6], although Bagging could work well on unstable base learners such as decision trees and neural networks, it could hardly work on stable base learners such as nearest neighbor classifiers.

Actually, it is known that in order to build a strong ensemble, the component learners should be with high accuracy as well as high diversity [22]. The reason why Bagging can hardly work on nearest neighbor classifiers lies in the fact that Bagging works through perturbing the training data with the help of bootstrap sampling. Although this scheme is effective in generating accurate but diverse component learners on unstable base learners, it is useless on stable base learners such as nearest neighbor classifiers. This is not difficult to understand because as Breiman indicated [6], given N training examples, the probability that the i-th training example is selected 0, 1, 2, $\cdots$ times is approximately Poisson distributed with $\lambda = 1$. The probability that the i-th example will occur at least once is $1 - (1/e) \approx 0.632$. If there are t bootstrap samples in a 2-class problem, then a test instance may change classification only if at least one of its nearest neighbors in the training set does not appear in at least $t/2$ samples. This probability is given by the probability that the number of heads in t tosses of a coin with probability 0.632 of heads is less than $0.5t$. As t gets larger, this probability gets very small.

Since nearest neighbor classifiers are very useful in real-world applications [1] [11], it will be desirable if the powerful Bagging algorithm can be adapted to these local learners. Considering that the sole perturbation on the training data is not sufficient in generating accurate but diverse component local learners, a possible way to adapt Bagging to nearest neighbor classifiers may be the employment of other kinds of perturbations.

In fact, besides perturbing the training data, there are other kinds of perturbation which can be exploited in ensemble construction [13]. For example, the perturbation on the input attributes has been used in building ensembles of BP neural

Manuscript received xxx xx, 200x; revised xxx xx, 200x. This work was supported by the the National Outstanding Youth Foundation of China under the Grant No. 60325207, the Fok Ying Tung Education Foundation under the Grant No. 91067, and the Excellent Young Teachers Program of MOE of China.

The authors are with the National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China (e-mail: zhouzh@nju.edu.cn; yuy@lamda.nju.edu.cn).

networks [10], C4.5 decision trees [19], and nearest neighbor classifiers [20]; the perturbation on the learning parameters has been used with BP neural networks [21], C4.5 decision trees [12], and FOIL rule inducers [2]; the perturbation on the output targets has been used with C4.5 decision trees and BP neural networks [14], and CART decision trees [8]; while the combination of the perturbation on the training data and that on the input attributes has been used in building ensembles of C4.5 decision trees [26].

In this paper, through utilizing multimodal perturbation, i.e. perturbing the training data, input attributes and learning parameters together, a new variant of Bagging called as *FASBIR* (Filtered Attribute Subspace based Bagging with Injected Randomness) is proposed. A large empirical study involving twenty data sets shows that this algorithm is effective in building ensembles of k-nearest neighbor (k-NN) classifiers, whose performance is better than many other ensemble algorithms.

The rest of this paper is organized as follows. Section II presents the FASBIR algorithm. Section III reports on the empirical study and explores why FASBIR works. Section IV concludes and raises several issues for future work.

II. FASBIR

Krogh and Vedelsby [22] have derived a famous equation $E = \bar{E} - \bar{A}$ in the case of regression, where E is the generalization error of an ensemble, while $\bar{E}$ and $\bar{A}$ are the average generalization error and average *ambiguity* of the component learners, respectively. The ambiguity was defined as the variance of the component predictions around the ensemble prediction, which measures the disagreement among the component learners [22]. This equation discloses that the more accurate and the more diverse the component learners are, the better the ensemble is. Unfortunately, although diversity has been recognized as a very important characteristic in ensemble construction in both regression and classification [22] [25], as a recent study [23] reveals, measuring diversity is not straightforward because there is no generally accepted formal definition, and so using it effectively for building better ensembles is still a problem to be solved. Therefore, although generating accurate but diverse component learners is the key of most ensemble learning algorithms, it remains a trick at present. Since perturbing the training data through bootstrap sampling is not sufficient in generating accurate but diverse component k-NN classifiers, FASBIR exploits the perturbation on input attributes and learning parameters together with bootstrap sampling, which will be explained in this section.

A. Perturbing Input Attributes

A data set is usually described with a set of attributes while different attribute subspaces, i.e. attribute subsets, might provide different views on the data. Therefore, component learners trained from different attribute subspaces might be quite diverse. This fact has been exploited well by the Random Subspace algorithm [19], which has obtained success in building ensembles of decision trees [19] and has been applied to nearest neighbor classifiers [20].

Random Subspace works through randomly choosing a subset of attributes to train a component learner. In nearest neighbor classifiers, when a new instance is compared to a training example, only the selected attributes have nonzero contributions to the distance. Geometrically this is equivalent to projecting all the instances to the selected subspace, and then identifying the nearest neighbors using the projected distances. Each component classifier in the ensemble corresponds to a random subspace, in which a set of nearest neighbors of the new instance are identified. The identified neighboring training examples are then combined via majority voting to determine the label of the new instance.

As the empirical study to be reported in Section III discloses, the success of Random Subspace mainly lies in that it could generate quite diverse component k-NN classifiers. However, the accuracy of the component classifiers is often insufficient. This might be because in an original training set there are usually some attributes irrelevant to the learning target, which might interfere with the learning on the relevant attributes. Since the input attributes are randomly chosen to form subsets to train the component learners, the influence of irrelevant attributes might increase to some degree such that the accuracy of the component learners is harmed. So, FASBIR utilizes attribute filtering to remove some irrelevant attributes and then exploits the merit of attribute subspace.

Here a simple attribute filtering process is used, which is based on the *information gain* criterion. In detail, the information gain of all the original input attributes are evaluated on the original training set. Then, the attributes whose information gain is less than a threshold f are deemed as irrelevant and removed. Finally, a subset of the remaining attributes are randomly chosen for the training of each component learner, where the size of the subset is controlled by a parameter s in such a way that $(d \times s)$ attributes are included in the subset if there are d attributes passing the filtering process.

Note that in general, Random Subspace performed on the entire attribute set should be better than that on an attribute subset, even though there are many irrelevant attributes. This is because although utilizing the irrelevant attributes may degenerate the component learners, the magnitude of the increase of the diversity may be bigger than that of the decrease of the accuracy. However, when the attribute subspace is not the only channel to achieve the diversity, as the case of FASBIR, removing these attributes may result in the improvement of the accuracy of the component learners, while the lost diversity may be got back from other channels. This is confirmed by the empirical study reported in Section III, which shows that the introduction of the attribute filtering mechanism is beneficial to FASBIR.

B. Perturbing Learning Parameters

Injecting randomness into the base learner could be helpful in generating accurate but diverse component learners, which has been applied by Kolen and Pollack [21] to neural networks through setting different initial weights for different component networks, by Kwok and Carter [24] and Dietterich [12] to C4.5 decision trees through introducing some randomness

to the selection of tests for splitting tree nodes, and by Ali and Pazzani [2] to rule inducer FOIL through randomly selecting some good candidate rule conditions.

Nearest neighbor classifier has no separate training phase, but when a new instance is given to be classified, the classifier will identify several neighboring training examples for this new instance and then usually use the majority voting scheme to produce the classification. The Euclidean distance is generally used to measure the distance between different instances described by continuous attributes. Actually, the Euclidean distance is a special case of Minkowsky distance, with order 2. That is, in the formal definition of Minkowsky distance shown in Eq. 1, p is set to 2 for obtaining Euclidean distance. Here $\mathbf{x}_1$ and $\mathbf{x}_2$ are two instances described by d-dimensional continuous attribute vectors.

$$Minkowsky_p(\mathbf{x}_1, \mathbf{x}_2) = \left(\sum_{n=1}^{d} |\mathbf{x}_{1,n} - \mathbf{x}_{2,n}|^p \right)^{1/p} \quad (1)$$

Through setting different values to p, different distance metrics can be obtained. In general, the smaller the value of p, the more robust the resulting distance metric to data variations; while the bigger the value of p, the more sensitive the resulting distance metric to variations. That means these resulting distance metrics may help identify different vicinities of a given instance. Such a fact can be exploited in ensemble construction.

However, Minkowsky distance can hardly deal with categorical attributes. Fortunately, VDM (Value Difference Metric) [27] can be a good complement. Let $N_{a,u}$ denote the number of training examples holding value u on categorical attribute a, $N_{a,u,c}$ denote the number of training examples belonging to the c-th class and holding value u on a, and n denote the number of classes. The distance between two values u and v on a can be computed by a simplified version of VDM shown in Eq. 2.

$$VDM_p(u, v) = \sum_{c=1}^{n} \left| \frac{N_{a,u,c}}{N_{a,u}} - \frac{N_{a,v,c}}{N_{a,v}} \right|^p \quad (2)$$

A new distance metric can be developed through combining the Minkowsky distance and VDM in the way as Eq. 3 shows, where the first j attributes are categorical while the remaining $(d - j)$ ones are continuous attributes normalized to $[0, 1]$. Here the distance is called *Minkovdm* distance. It is evident that such a distance metric can deal with both continuous and categorical attributes.

$$Minkovdm_p(\mathbf{x}_1, \mathbf{x}_2) = (\sum_{h=1}^{j} VDM_p(\mathbf{x}_{1,h}, \mathbf{x}_{2,h}) + \sum_{h=j+1}^{d} |\mathbf{x}_{1,h} - \mathbf{x}_{2,h}|^p)^{1/p} \quad (3)$$

The FASBIR algorithm perturbs the learning parameter of k-NN classifier through randomly choosing a p value from a distance order set $\{p_1, p_2, \cdots, p_l\}$ to instantiate Eq. 3 for each component k-NN classifier. Empirical study reveals that when coupling with bootstrap sampling, such a simple scheme is helpful in increasing the diversity of relatively accurate component k-NN classifiers.

TABLE I
PSEUDO-CODE DESCRIBING THE FASBIR ALGORITHM

FASBIR($\mathbf{x}$, D, Y, k, t, Ω, f, s)

Input: $\mathbf{x}$: Instance to label
D: Data set
Y: Label set $\{y_1, y_2, \cdots, y_n\}$
k: Number of neighbors to query
t: Number of component learners to be constructed
Ω: Distance order set $\{p_1, p_2, \cdots, p_l\}$
f: Threshold for attribute filtering
s: Parameter determining the size of attribute subsets

%% the part that can be performed off-line:
$D^f \leftarrow AttrFilter(D, f)$; %% perform attribute filtering under %% the control of threshold f
for $i \in \{1..t\}$ **do**
  $D_i^f \leftarrow BootstrapSample(D^f)$;
  $D_i^{f*} \leftarrow Subspace(D_i^f, s)$;
    %% D_i^{f*} is described with a randomly chosen subset of %% attributes whose size is controlled by s
  $p_i \leftarrow RandomSelect(\Omega)$
end of for
%% the part to be performed on-line:
for $i \in \{1..n\}$ **do** $count_i \leftarrow 0$ %% n is the number of classes
for $i \in \{1..t\}$ **do**
  $\mathcal{Z} \leftarrow Neighbor(\mathbf{x}, k, D_i^{f*}, Minkovdm_{p_i})$;
    %% k nearest neighbors of $\mathbf{x}$ are identified in D_i^{f*}
    %% according to the distance metric $Minkovdm_{p_i}$
  $i^* \leftarrow \arg\max_i \sum_{\substack{y_i \in Y,\, z \in \mathcal{Z} \\ z.label = y_i}} 1$;
    %% y_{i^*} is the label held by majority examples in $\mathcal{Z}$
  $count_{i^*} \leftarrow count_{i^*} + 1$
    %% y_{i^*} receives the vote of a component learner
end of for

Output: $\mathbf{x}.label \leftarrow y_{\arg\max_i (count_i)}$
    %% $\mathbf{x}.label$ is determined via majority voting

C. The Algorithm

The FASBIR algorithm works as follows. At first, it performs attribute filtering on the original training set D so that some irrelevant attributes are removed. Let D^f denote the resulting data set which is described with the remaining attributes. Then, FASBIR generates t bootstrap samples from D^f, each of which is described with a randomly chosen subset of attributes whose size is controlled by the parameter s. After that, a p value of the Minkovdm distance is randomly chosen for this sample from a distance order set, i.e. Ω. When a new instance is given to be labeled, its neighboring training examples are identified on each of these samples, using the filtered attribute subspace and the p value associated with that sample. Note that replicated neighbors may be identified because a bootstrap sample may contain replicated examples. On each sample, the identified neighbors vote for the label of the new instance, while the result is regarded as the prediction made by the component k-NN classifier of the sample. Finally, the predictions of the component k-NN

classifiers are combined via majority voting. The pseudo-code of the FASBIR algorithm is shown in Table I. Comparing with Bagging, FASBIR has three more parameters to set, i.e. Ω, f and s.

Note that the running time cost of FASBIR is quite small because the processes of attribute filtering, attribute subspace, bootstrap sampling, and the configuration of the distance metrics can all be performed off-line. Only the processes for identifying the nearest neighboring training examples and voting the component predictions are required to be executed on-line. Therefore, the running time cost of FASBIR is comparable to that of the established efficient algorithms, i.e. Bagging and Random Subspace.

It is worth noting that Alkoot and Kittler [3] have also tried to adapt Bagging to nearest neighbor classifiers. The main purpose of their work is to minimize the *veto effect* [1] in combining the soft outputs of k-NN classifiers by product rule, therefore they proposed a new combination scheme using the Bayesian prior to marginalizing the k-NN estimates, and obtained good effect on very small data sets (with up to 80 training examples in the experiments reported in [3]). In contrast to Alkoot and Kittler's work, the FASBIR algorithm attempts to combine the hard output of k-NN classifiers via majority voting, where there is no veto effect. Moreover, FASBIR works with a new method for generating accurate but diverse component k-NN classifiers with the help of multimodal perturbation, while it does not work with any new combination schemes. Furthermore, the empirical study reported in the next section shows that FASBIR can be effective on big data sets besides small ones.

III. Empirical Study

Twenty data sets from the UCI Machine Learning Repository [5] are used in the empirical study, where the original instances with missing values have been removed since standard k-NN classifier can hardly deal with missing values while the purpose of this paper is not to design any technique to improve such ability of k-NN. Information on these data sets is tabulated in Table II. Note that the data sets are separated into two groups, i.e. big and small, according to Eq. 4, which indicates the size of the data set relative to its dimensionality and number of classes.

$$COEF = \frac{size}{\textit{number of attributes} \times \textit{number of classes}} \quad (4)$$

Since FASBIR utilizes multimodal perturbation, several degenerated variants can be easily derived. First, if only the perturbation on the input attributes is employed, then the *FAS* (Filtered Attribute Subspace) algorithm is obtained. That is, the FAS algorithm performs attribute filtering on the original training set, then randomly chooses a subset of remaining attributes for each of the component k-NN classifiers. Second, if only the perturbation on the learning parameters is employed, then the *InRand* (Injected Randomness) algorithm is obtained. That is, the InRand algorithm randomly chooses a p value from the distance order set to instantiate the Minkovdm distance for each of the component k-NN classifiers. Third, if both the perturbation on the training data and the perturbation on the learning parameter are employed, then the *BagInRand* (Bagging with Injected Randomness) algorithm is obtained, which has been presented in a previous work [30]. Moreover, if the perturbations on the input attributes, learning parameters, and training data are all used, but in perturbing the input attributes only the attribute subspace is used, then the *RanBIR* (Random Subspace with BagInRand) algorithm is obtained. That is, RanBIR is almost identical to FASBIR except that it does not utilize attribute filtering. Note that Bagging and Random Subspace (abbreviated as RanSub in this section) can also be viewed as degenerated variants of FASBIR, where the former uses only the perturbation on the training data while the latter uses only attribute subspace to perturb the input attributes. These algorithms are summarized in Table III and will be empirically compared in this section.

TABLE II
Experimental data sets

Data set	Size	Attribute		Class	$COEF$
		Categorical	Continuous		
soybean	562	0	35	19	0.85
autos	159	15	10	7	0.91
sonar	208	60	0	2	1.73
lymph	148	3	15	4	2.06
glass	214	9	0	7	3.40
anneal	898	6	32	6	3.94
heart-c	296	6	7	5	4.55
ionosphere	351	34	0	2	5.16
vowel	990	10	3	11	6.92
vote	232	0	16	2	7.25
heart-s	270	13	0	2	10.38
vehicle	846	18	0	4	11.75
iris	150	4	0	3	12.50
breast-c	277	0	9	2	15.39
segment	2,310	19	0	7	17.37
credit-a	653	6	9	2	21.77
credit-g	1,000	7	13	2	25.00
breast-w	683	9	0	2	37.94
diabetes	768	8	0	2	48.00
balance	625	4	0	3	52.08

On each data set, for each compared ensemble algorithm, five kinds of k-nearest neighbor classifiers are tested, where the value of k is set to 1, 3, 5, 7 and 9, respectively. Moreover, five ensemble sizes are tried, including 20, 40, 60, 80 and 100. Here the subspace rate used by Random Subspace and RanBIR is set to 0.33, which means about 1/3 original attributes are used by each component k-NN classifier; the Ω used by InRand, BagInRand, RanBIR and FASBIR is set to $\{1, 2, 3\}$; the f used by FAS and FASBIR is set to $0.33 \times AveGain$, where $AveGain$ denotes the average information gain of all the original attributes; the s used by FAS and FASBIR is

[1] *Veto effect* is the phenomenon that if estimation errors drive one of the class *a posteriori* probability estimates to zero, the output of the product fusion will also be zero, even if other classifiers provide a lot of support for the class.

TABLE III

FASBIR AND SOME DEGENERATED VARIANTS

Data set	Perturb training data with	Perturb input attributes with		Perturb learning parameters with
	bootstrap sampling	attribute filtering	attribute subspace	distance configuration
Bagging	YES	NO	NO	NO
FAS	NO	YES	YES	NO
InRand	NO	NO	NO	YES
BagInRand	YES	NO	NO	YES
RanSub	NO	NO	YES	NO
RanBIR	YES	NO	YES	YES
FASBIR	YES	YES	YES	YES

TABLE IV

COMPARISON ON GENERALIZATION ERROR WITH $k = 1$ AND ENSEMBLE SIZE SET TO 100

Data set	Single	Bagging	FAS	InRand	BagInRand	RanSub	RanBIR	FASBIR
soybean	.0782±.0057	.0839±.0048	.0717±.0036	.0692±.0041	.0759±.0058	**.0655±.0032**	.0677±.0026	.0716±.0037
autos	.2090±.0121	.2021±.0122	.1534±.0164	.1935±.0116	.1963±.0138	.1689±.0129	.1670±.0191	**.1506±.0134**
sonar	.1426±.0149	.1441±.0078	.1387±.0121	.1474±.0104	.1418±.0092	**.1050±.0138**	**.1089±.0155**	.1325±.0132
lymph	.1455±.0141	**.1283±.0114**	**.1269±.0151**	**.1364±.0145**	**.1291±.0104**	**.1305±.0096**	.1490±.0113	.1382±.0134
glass	.3025±.0167	.3034±.0097	**.1993±.0166**	.3034±.0128	.2902±.0115	.2151±.0108	.2243±.0134	**.1960±.0137**
anneal	.0093±.0024	.0096±.0021	.0097±.0022	.0088±.0008	.0091±.0014	.0209±.0025	.0212±.0026	**.0073±.0012**
heart-c	.2391±.0143	.2250±.0092	.1827±.0110	.2099±.0122	.1905±.0095	.1793±.0127	.1804±.0087	**.1672±.0081**
ionosphere	.1312±.0057	.1324±.0055	.0795±.0041	.1252±.0067	.1200±.0090	**.0710±.0067**	.0800±.0058	.0826±.0055
vowel	**.0076±.0020**	.0081±.0021	.0108±.0019	.0084±.0022	.0090±.0019	.0127±.0032	.0152±.0022	.0140±.0010
vote	.0471±.0056	.0414±.0066	.0449±.0063	.0462±.0058	.0423±.0058	.0540±.0087	.0548±.0056	**.0410±.0038**
heart-s	.2411±.0100	.2400±.0105	.1833±.0129	.2411±.0105	.2330±.0101	.1852±.0078	**.1759±.0059**	**.1715±.0106**
vehicle	.3040±.0057	.3013±.0069	.2813±.0085	.3028±.0064	.2934±.0095	.2796±.0085	.2771±.0077	**.2743±.0050**
iris	.0440±.0064	**.0407±.0021**	.0707±.0164	.0447±.0032	.0440±.0034	.0667±.0133	.0447±.0045	.0467±.0070
breast-c	.3057±.0248	.2764±.0099	.2672±.0167	.3065±.0225	.2728±.0102	.2568±.0085	**.2528±.0080**	.2556±.0065
segment	.0282±.0016	.0286±.0017	.0281±.0020	.0270±.0021	.0266±.0017	**.0235±.0017**	.0237±.0017	.0285±.0021
credit-a	.1785±.0103	.1544±.0065	.1445±.0058	.1763±.0102	.1441±.0045	.1405±.0077	**.1348±.0060**	**.1359±.0062**
credit-g	.2885±.0102	.2649±.0066	.2514±.0068	.2881±.0076	.2729±.0067	.2563±.0054	.2624±.0048	**.2481±.0057**
breast-w	.0419±.0029	.0410±.0024	.0279±.0021	.0426±.0027	.0362±.0034	**.0249±.0022**	**.0251±.0019**	.0260±.0020
diabetes	.2951±.0072	.2861±.0081	.2659±.0089	.2960±.0099	.2870±.0087	.2750±.0082	.2644±.0087	**.2537±.0089**
balance	.2626±.0071	.2278±.0089	.3337±.0142	.2632±.0083	.2109±.0110	.3290±.0112	**.1382±.0069**	**.1371±.0075**

set to 0.5 so that the number of attributes utilized by each component k-NN classifier is roughly same to that used in Random Subspace.

A. Comparison on Generalization Error

Ten times 10-fold cross validation is performed on each of the data sets listed in Table II. In detail, each data set is partitioned into ten subsets with similar sizes and distributions. Then, the union of nine subsets is used as the training set while the remaining subset is used as the test set, which is repeated for ten times such that every subset has been used as the test set once. The average test result is regarded as the result of the 10-fold cross validation. The whole above process is repeated for 10 times with random partitions of the ten subsets, and the average results as well as the standard deviations of these different partitions are recorded.

Table IV presents the generalization error of single k-NN classifier and that of the compared ensemble algorithms, where the values following "±" are standard deviations. Here the k value is fixed to 1 and ensemble sizes of all the ensemble algorithms are fixed to 100, because the 1-NN classifier is often regarded as a baseline [11] while 100 component learners are often used in literatures on k-NN ensembles [4] [20], and the experimental results on other k values and ensemble sizes will be presented later in this subsection. Pairwise two-tailed t-tests with 0.01 significance level are performed. In detail, on each data set, the algorithm with the smallest error, say $algo$, is picked out at first. Then, the performance of other algorithms are compared with that of $algo$ with t-test, so that the algorithms are grouped into two subsets, that is, algorithms significantly worse than $algo$ or algorithms comparable to $algo$. Finally, if neither of these two subsets is empty, then the table entries corresponding to the algorithms belonging to the second subset are boldfaced, while the boldfaced performance is called the *significantly best performance*.

Table IV shows that FASBIR achieves the significantly best performance on 55% ($11/20$) data sets, which is evidently the best algorithm. The performance of RanSub and RanBIR are comparable, apparently better than the remaining algorithms except FASBIR. Both FASBIR and RanSub are effective on data sets with big $COEF$ as well as on data sets with small $COEF$, while RanBIR appears better on data sets with big

TABLE V
COMPARISON ON ERROR REDUCTION RATES WITH $k = 1$ AND ENSEMBLE SIZE SET TO 100

Dataset	Bagging	FAS	InRand	BagInRand	RanSub	RanBIR	FASBIR
soybean	-0.07	0.08	0.12	0.03	**0.16**	0.13	0.08
autos	0.03	0.27	0.07	0.06	0.19	0.20	**0.28**
sonar	-0.01	0.03	-0.03	0.01	**0.26**	**0.24**	0.07
lymph	**0.12**	**0.13**	**0.06**	**0.11**	**0.10**	-0.02	0.05
glass	0.00	**0.34**	0.00	0.04	0.29	0.26	**0.35**
anneal	-0.03	-0.04	0.05	0.02	-1.25	-1.28	**0.22**
heart-c	0.06	0.24	0.12	0.20	0.25	0.25	**0.30**
ionosphere	-0.01	0.39	0.05	0.09	**0.46**	0.39	0.37
vowel	-0.07	-0.42	-0.11	-0.18	-0.67	-1.00	-0.84
vote	0.12	0.05	0.02	0.10	-0.15	-0.16	**0.13**
heart-s	0.00	0.24	0.00	0.03	0.23	**0.27**	**0.29**
vehicle	0.01	0.07	0.00	0.03	0.08	0.09	**0.10**
iris	**0.07**	-0.61	-0.02	0.00	-0.52	-0.02	-0.06
breast-c	0.10	0.13	0.00	0.11	0.16	**0.17**	0.16
segment	-0.01	0.00	0.04	0.06	**0.17**	0.16	-0.01
credit-a	0.14	0.19	0.01	0.19	0.21	**0.24**	**0.24**
credit-g	0.08	0.13	0.00	0.05	0.11	0.09	**0.14**
breast-w	0.02	0.33	-0.02	0.14	**0.41**	**0.40**	0.38
diabetes	0.03	0.10	0.00	0.03	0.07	0.10	**0.14**
balance	0.13	-0.27	0.00	0.20	-0.25	**0.47**	**0.48**
average	0.04	0.07	0.02	0.07	0.02	0.05	0.14

$COEF$. It is noteworthy that although neither of Bagging, FAS and InRand is better than RanSub, their combination, i.e. FASBIR, is better than RanSub. This observation confirms that multimodal perturbation is effective in ensemble construction, even when no single-modal perturbation works well. Moreover, the observation that FASBIR is better than RanBIR also tells that the introduction of attribute filtering is beneficial.

Table V compares the *error reduction rates* of the ensemble algorithms shown in Table IV, where the error reduction rate of an ensemble algorithm $algo$ is defined as Eq. 5. The positive values indicate the decrease of errors while the negative values indicate the increase of errors, where the error of single k-NN classifier is regarded as the baseline. The "average" row of Table V confirms that FASBIR is the best ensemble algorithm, whose error reduction rate is as twice as that of the second largest ones.

$$rate_{algo} = 1 - \frac{error_{algo}}{error_{Single}} \tag{5}$$

Moreover, to compare the robustness of these algorithms, that is, how well the particular algorithm $algo$ performs in different situations, a criterion is defined similar to the one used in [29]. In detail, the relative performance of algorithm $algo$ on a particular data set is expressed by dividing its error $error_{algo}$ by the biggest error among all the compared algorithms, as shown in Eq. 6.

$$r_{algo} = \frac{error_{algo}}{\max\limits_{\alpha} error_{\alpha}} \tag{6}$$

The worst algorithm $algo^*$ on that data set has $r_{algo^*} = 1$, and all the other algorithms have $r_{algo} \leq 1$. The smaller the value of r_{algo}, the better the performance of the algorithm. Thus the sum of r_{algo} over all data sets provides a good indication of the robustness of the algorithm $algo$. The smaller the value of the sum, the better the robustness of the algorithm.

The distribution of r_{algo} of the compared algorithms on the experimental data sets is shown in Fig. 1. For each algorithm, the twenty values of r_{algo} are stacked for the ease of comparison. Here the ensemble size is fixed to 100. Five robustness figures corresponding to $k = 1, 3, \ldots, 9$, respectively, are shown.

Fig. 1 discloses that on all the k values, FASBIR achieves the smallest sum of r_{algo}, which means FASBIR is the best algorithm on all the k values. Moreover, Fig. 1 shows that FAS and BagInRand are worse than RanBIR, while very competitive with RanSub. In fact, except on $k = 1$, FAS and BagInRand are almost always better than RanSub. Considering Fig. 1 together with Tables IV and V, it can be found that FASBIR is not only effective on the biggest number of data sets, but also with the biggest error reduction effect and the best robustness; RanBIR is effective on the second biggest number of data sets and with the second best robustness, but its error reduction effect is not so big; while although RanSub is effective on the second biggest number of data sets, its error reduction effect and robustness are not so well.

Since five sizes of ensembles are tested in the empirical study, the influence of the ensemble size on the generalization ability can be analyzed. Fig. 2 presents the *error area charts* of the compared ensemble algorithms on the 20 data sets with increasing ensemble size. Here the k value is fixed to 1.

Fig. 2 shows that the increase of the ensemble size is apparently beneficial to the performance of FAS and FASBIR, while not apparently to Bagging, InRand, BagInRand, RanSub

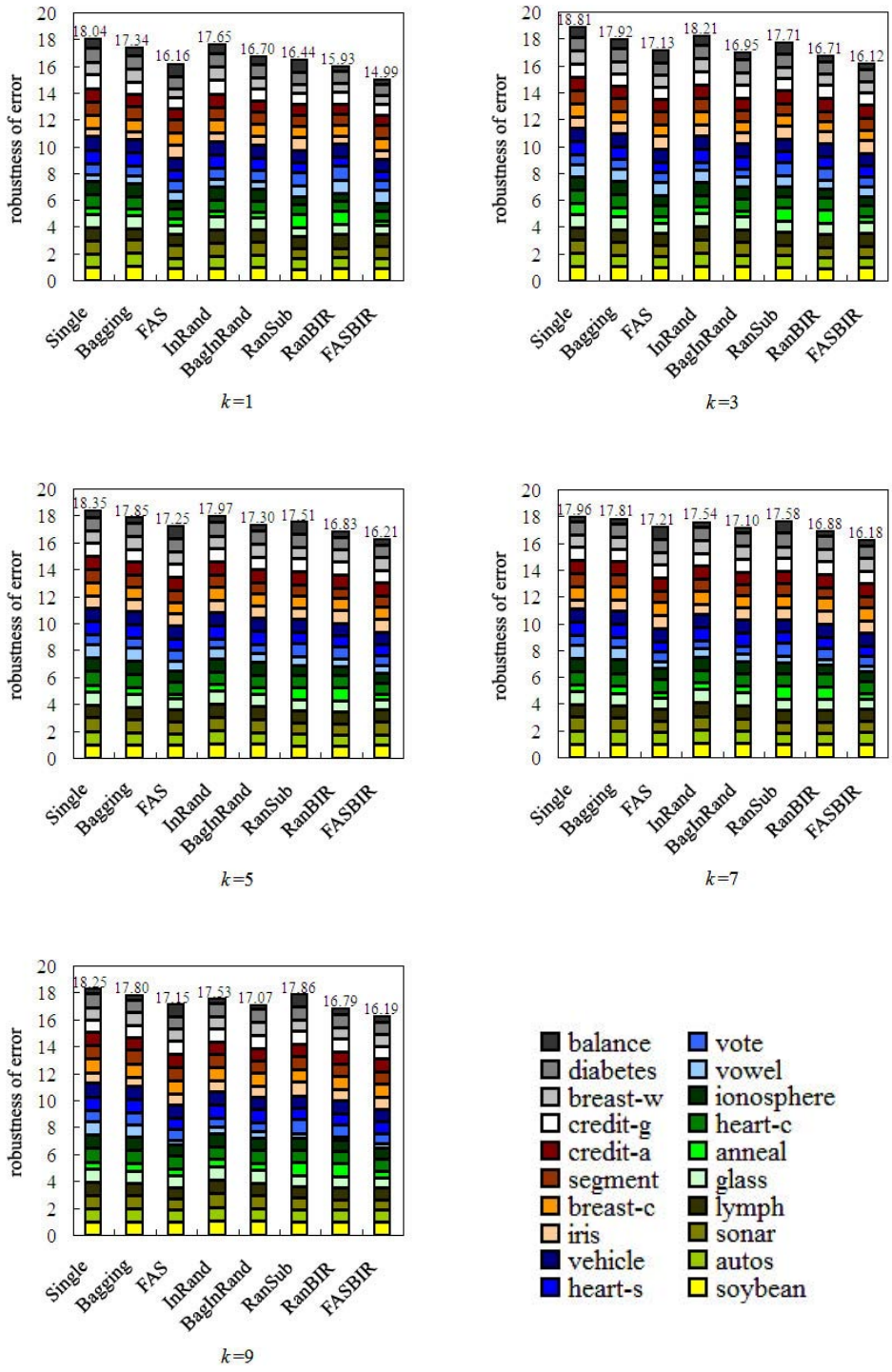
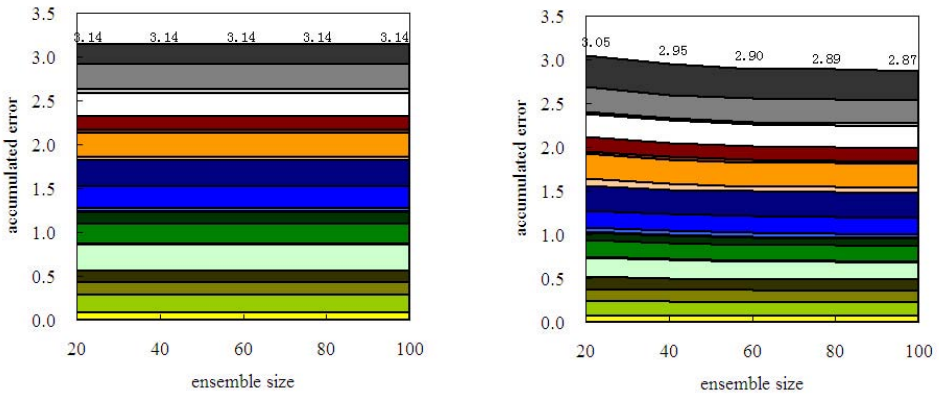


Fig. 1. Robustness of the compared algorithms on 20 UCI data sets with $k = 1, 3, \ldots, 9$ and ensemble size set to 100

and RanBIR. It is evident that this property of FASBIR, i.e. apparently benefitting from the increase of ensemble size, is inherited from FAS and originated from the utilizing of attribute filtering.

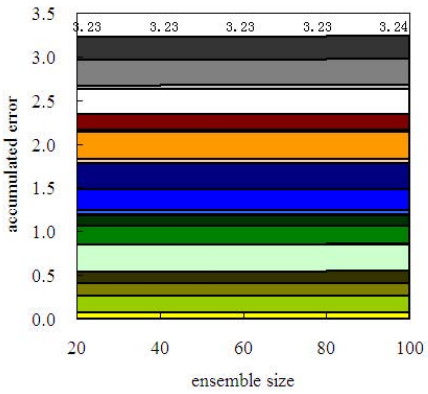
Among all the twenty experimental data sets, under a certain k value and a certain ensemble size (sz), if an algorithm $algo$ achieves the significantly best performance on v number of data sets, then it is called that the *winning frequency* of $algo$ is $\frac{v}{20}\%$. Table VI presents the 1st- and 2nd-highest winning frequencies under different combinations of the k values and ensemble sizes, and the algorithms achieving them.

Table VI shows that FASBIR achieves the highest winning frequency on 17 among all these 25 configurations, while on six of the remaining eight configurations FASBIR achieves the second-highest winning frequency. As for other algorithms, RanBIR, BagInRand, RanSub, and Bagging achieve the high-

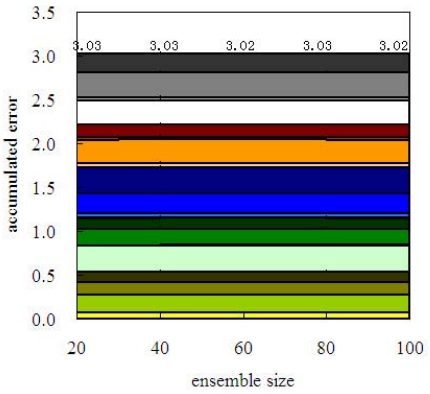


a. Bagging

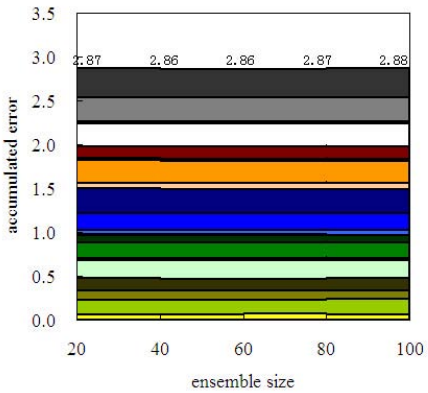
b. FAS



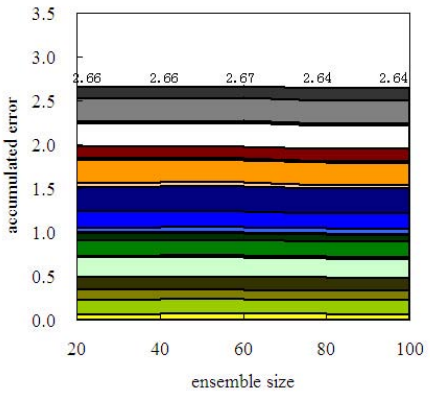
c. InRand



d. BagInRand



e. RanSub



f. RanBIR

Fig. 2. Error area charts of the ensemble algorithms on 20 UCI data sets with $k = 1$ and ensemble size varying from 20 to 100 (to be continued)

est winning frequency on some configurations, concretely, on 11, 3, 1, and 1, respectively. These observations tell that FASBIR is the best among the compared algorithms.

The FASBIR algorithm is also compared with the *ID* (Input Decimation) algorithm [28], which perturbs the input attributes by presenting different attribute subsets to each component learners. In detail, the ID algorithm chooses different subsets of attributes based on the correlations between individual attributes and the class labels, and trains classifiers on each of these subsets. For an n-class problem, ID trains n classifiers each corresponding to a class, therefore, the ensemble is consists of n classifiers. It was reported that the ID algorithm could perform better than Random Subspace [28].

Since the ID algorithm can only work with data sets with at least three classes [28], comparison between FASBIR and ID are held on eleven out of the twenty data sets shown in Table II. Moreover, since the ensemble size of ID is determined by the number of classes of the data set, in the comparison

g. FASBIR

Fig. 2. Error area charts of the ensemble algorithms on 20 UCI data sets with $k = 1$ and ensemble size varying from 20 to 100 (continued)

TABLE VI
THE 1ST- AND 2ND-HIGHEST WINNING FREQUENCIES AND THE CORRESPONDING ALGORITHMS

sz	$k = 1$	$k = 3$	$k = 5$	$k = 7$	$k = 9$
20	55%(RanBIR) 35%(RanSub)	35%(FASBIR,BagInRand) 30%(FAS)	35%(Bagging,BagInRand) 30%(FASBIR,FAS)	40%(FASBIR) 30%(BagInRand)	40%(FASBIR,BagInRand) 30%(FAS)
40	45%(FASBIR,RanBIR) 40%(RanSub)	40%(FASBIR,RanBIR) 20%(Bagging)	40%(FASBIR) 35%(FAS)	45%(FASBIR) 35%(RanBIR)	40%(FASBIR) 30%(FAS,RanBIR)
60	45%(FASBIR) 35%(RanBIR)	50%(RanBIR) 40%(FASBIR)	40%(FASBIR,RanBIR) 30%(FAS)	35%(FASBIR) 30%(RanSub)	30%(FASBIR,RanBIR) 20%(Bagging,FAS, BagInRand,RanSub)
80	55%(FASBIR) 45%(RanBIR)	35%(RanBIR) 30%(FASBIR)	55%(RanBIR) 35%(FASBIR)	50%(RanBIR) 30%(FASBIR)	45%(FASBIR) 30%(RanBIR)
100	55%(FASBIR) 30%(RanSub,RanBIR)	40%(FASBIR,RanBIR) 25%(FAS,RanSub)	55%(FASBIR) 30%(RanBIR)	40%(RanSub) 35%(RanBIR)	55%(RanBIR) 30%(FASBIR)

the ensemble size of FASBIR is set to 20, much smaller than the size used before. Due to these differences, the comparison between FASBIR and ID is reported in Table VII, separated from the previous comparisons reported in Tables IV and V and Figs. 1 and 2. Note that two configurations of the *attribute selection rate* used by ID are tested, that is, 0.2 and 0.8. Pairwise two-tailed t-tests with 0.01 significance level are performed, and on each data set the table entry of the algorithm which achieves the significantly best performance is boldfaced.

Table VII reveals that FASBIR is significantly better than ID (0.2) on all data sets and significantly better than ID (0.8) on six data sets. This indicates that FASBIR is better than the ID algorithm.

TABLE VII
COMPARISON WITH THE ID ALGORITHM

Dataset	ID (0.2)	ID (0.8)	FASBIR
soybean	.0769±.0040	.0778±.0066	**.0711±.0047**
autos	.3017±.0089	.2142±.0133	**.1680±.0148**
lymph	.2734±.0222	**.1450±.0164**	.1550±.0172
glass	.3621±.0121	.3100±.0099	**.2086±.0136**
anneal	.0262±.0034	**.0094±.0024**	.0105±.0026
heart-c	.2385±.0128	.2385±.0128	**.1786±.0179**
vowel	.1192±.0096	**.0077±.0020**	.0191±.0031
vehicle	.4414±.0078	.3095±.0069	**.2822±.0089**
iris	.0780±.0144	**.0447 ±.0032**	**.0480 ±.0082**
segment	.0801±.0033	**.0267±.0015**	.0310±.0022
balance	.4977±.0295	.2609±.0067	**.2088±.0132**

B. Error-Ambiguity Decomposition

In order to explore why FASBIR works well, the error-ambiguity decomposition [22] mentioned before is performed. The errors of the ensembles presented in Table IV are regarded as E, while the average errors of the component learners constituting the ensembles are regarded as $\bar{E}$. Then, $\bar{A}$ can be obtained from $\bar{E} - E$ since $E = \bar{E} - \bar{A}$ [22], which discloses the improvement of the ensemble over the individual average. After that, the robustness of $\bar{E}$ and $\bar{A}$ can be computed in a similar way as Eq. 6, which is shown in Fig. 3 [2], where the legend of the figure is the same as that has been used in Figs. 1 and 2.

Fig. 3 shows that comparing with Bagging, both the $\bar{E}$ and $\bar{A}$ of FASBIR are bigger, which indicates that although

[2]Note that the robustness values appearing in different figures cannot be compared directly because they are with different baselines.

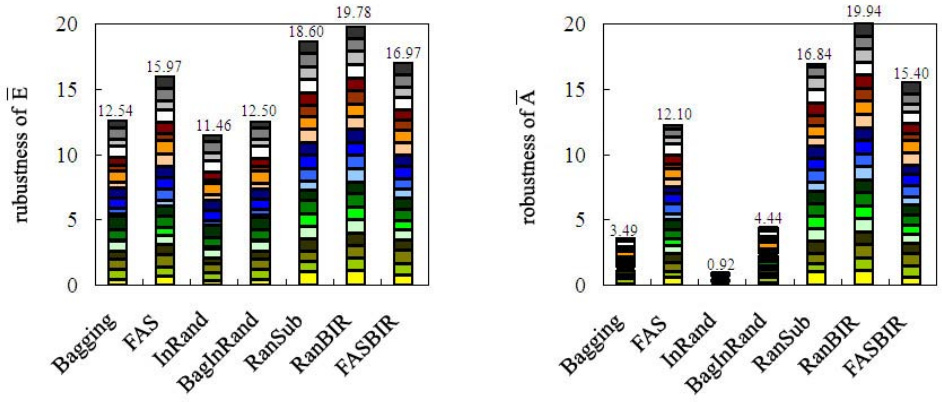


Fig. 3. Robustness of $\bar{E}$ and $\bar{A}$ with $k = 1$ and ensemble size set to 100

the component learners of FASBIR is not so accurate as that of Bagging, the ambiguity among the component learners of FASBIR is bigger than that of Bagging. Fig. 3 also shows that comparing with RanSub and RanBIR, both the $\bar{E}$ and $\bar{A}$ of FASBIR are smaller, which indicates that the ambiguity among the component learners of FASBIR is not so big as that of RanSub and RanBIR, but the accuracy of the component learners of FASBIR is bigger than that of RanSub and RanBIR.

Moreover, Fig. 3 shows that although the $\bar{A}$ of InRand is smaller than that of Bagging, the $\bar{A}$ of BagInRand which combines InRand with Bagging can be bigger than that of Bagging while reserving roughly same $\bar{E}$. This confirms the usefulness of multimodal perturbation. Furthermore, Fig. 3 shows that Bagging and InRand work with component k-NN classifiers with relatively small $\bar{E}$ and $\bar{A}$, while FAS works with component k-NN classifiers with relatively big $\bar{E}$ and $\bar{A}$. This suggests that different modalities of perturbation might contribute in different ways to ensemble construction.

IV. Conclusion

In this paper, a new ensemble learning algorithm FASBIR is proposed, which is designed for building ensembles of nearest neighbor classifiers. This algorithm works through integrating the perturbations on the training data, input attributes and learning parameters together. A large empirical study shows that although this algorithm is simple, it can effectively improve the accuracy of nearest neighbor classifiers.

FASBIR outperforms Bagging in generating more diverse component k-NN classifiers, while outperforms Random Subspace in generating more accurate component k-NN classifiers. It seems that different modalities of perturbation might contribute to ensemble construction, at least, that of FASBIR, in different ways. However, at present it is not clear how these different modalities of perturbation interacts with each other. This is an interesting issue to be explored in the future, which might shed light on the design of more powerful ensemble learning algorithms.

Moreover, as mentioned before, besides the perturbation on the training data, input attributes and learning parameters, the perturbation on the output targets may also be helpful. It is curious whether such perturbation can be incorporated into FASBIR and bring further advantages. Furthermore, although this paper shows that multimodal perturbation is effective in building ensembles of nearest neighbor classifiers, it is not clear whether it is also effective on other kinds of stable base learners such as naive Bayes classifier, or even effective on unstable base learners such as decision trees and neural networks. Also, exploring the performance of multimodal perturbation with larger ensembles is also an interesting issue. These have been left to be investigated in the future.

Acknowledgement

The comments and suggestions from the anonymous reviewers greatly improved this paper.

References

[1] D.W. Aha, “Lazy learning: special issue editorial,” Artificial Intelligence Review, vol.11, no.1–5, pp.7–10, 1997.
[2] K.M. Ali and M.J. Pazzani, “Error reduction through learning multiple descriptions,” Machine Learning, vol.24, no.3, pp.173–202, 1996.
[3] F.M. Alkoot and J. Kittler, “Moderating k-NN classifiers,” Pattern Analysis & Applications, vol.5, no.3, pp.326–332, 2002.
[4] S.D. Bay, “Combining nearest neighbor classifiers through multiple feature subsets,” in Proceedings of the 15th International Conference on Machine Learning, Madison, WI, pp.37–45, 1998.
[5] C. Blake, E. Keogh, and C.J. Merz, “UCI repository of machine learning databases” [http://www.ics.uci.edu/∼mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, CA, 1998.
[6] L. Breiman, “Bagging predictors,” Machine Learning, vol.24, no.2, pp.123–140, 1996.
[7] L. Breiman, “Bias, variance, and arcing classifiers,” Technical Report 460, Statistics Department, University of California, Berkeley, CA, 1996.
[8] L. Breiman, “Randomizing outputs to increase prediction accuracy,” Machine Learning, vol.40, no.3, pp.229–242, 2000.
[9] L. Breiman, “Random forest,” Machine Learning, vol.45, no.1, pp.5–32, 2001.
[10] K.J. Cherkauer, “Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks,” in Proceedings of the AAAI’96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, pp.15–21, 1996.
[11] B.V. Dasarathy, Nearest Neighbor Norms: NN Pattern Classification Techniques, Los Alamitos, CA: IEEE Computer Society Press, 1991.
[12] T.G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization,” Machine Learning, vol.40, no.2, pp.139–157, 2000.
[13] T.G. Dietterich, “Ensemble learning,” in The Handbook of Brain Theory and Neural Networks, 2nd edition, M.A. Arbib, Ed. Cambridge, MA: MIT Press, 2002.

[14] T.G. Dietterich and G. Bakiri, "Solving multi-class learning problems via error-correcting output codes," Journal of Artificial Intelligence Research, vol.2, pp.263–286, 1995.
[15] B. Efron and R. Tibshirani, An Introduction to the Bootstrap, New York: Chapman & Hall, 1993.
[16] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in Proceedings of the 2nd European Conference on Computational Learning Theory, Barcelona, Spain, pp.23–37, 1995.
[17] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," The Annals of Statistics, vol.28, no.2, pp.337–407, 2000.
[18] L.O. Hall, K.W. Bowyer, R.E. Banfield, D. Bhadoria, and S. Eschrich, "Comparing pure parallel ensemble creation techniques against bagging," in Proceedings of the 3rd IEEE International Conference on Data Mining, Melbourne, FL, pp.533–536, 2003.
[19] T.K. Ho, "The random subspace method for constructing decision forests," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.20, no.8, pp.832–844, 1998.
[20] T.K. Ho, "Nearest neighbors in random subspaces," in Lecture Notes in Computer Science 1451, A. Amin, D. Dori, P. Pudil, and H. Freeman, Eds. Berlin: Springer, pp.640–648, 1998.
[21] J.F. Kolen and J.B. Pollack, "Back propagation is sensitive to initial conditions," in Advances in Neural Information Processing Systems 3, R.P. Lippmann, J.E. Moody, and D.S. Touretzky, Eds. San Francisco, CA: Morgan Kaufmann, pp.860–867, 1991.
[22] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in Advances in Neural Information Processing Systems 7, G. Tesauro, D.S. Touretzky, and T.K. Leen, Eds. Cambridge, MA: MIT Press, pp.231–238, 1995.
[23] L.I. Kuncheva and C.J. Whitaker, "Measures of diversity in classifier ensembles," Machine Learning, vol.51, no.2, pp.181–207, 2003.
[24] S.W. Kwok and C. Carter, "Multiple decision trees," in Proceedings of the 4th International Conference on Uncertainty in Artificial Intelligence, New York, NY, pp.327–338, 1988.
[25] L. Lam, "Classifier combinations: implementations and theoretical issues," in Lecture Notes in Computer Science 1857, J. Kittler and F. Roli, Eds. Berlin: Springer, pp.78–86, 2000.
[26] P. Latinne, O. Debeir, and C. Decaestecker, "Different ways of weakening decision trees and their impact on classification accuracy of DT combination," in Lecture Notes in Computer Science 1857, J. Kittler and F. Roli, Eds. Berlin: Springer, pp.200–209, 2000.
[27] C. Stanfill and D. Waltz, "Toward memory-based reasoning," Communications of the ACM, vol.29, no.12, pp.1213–1228, 1986.
[28] K. Tumer and N.C. Oza, "Input decimated ensembles," Pattern Analysis & Applications, vol.6, no.1, pp.65–77, 2003.
[29] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas, "Non-linear dimensionality reduction techniques for classification and visualization," in Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, pp.645–651, 2002.
[30] Z.-H. Zhou and Y. Yu, "Adapt bagging to nearest neighbor classifiers," Journal of Computer Science & Technology, vol.20, no.1, pp.48–54, 2005.

Zhi-Hua Zhou (S'00-M'01) received the BSc, MSc and PhD degrees in computer science from Nanjing University, China, in 1996, 1998 and 2000, respectively, all with the highest honor. He joined the Department of Computer Science & Technology of Nanjing University as a lecturer in 2001, and is a professor and head of the LAMDA group at present. His research interests are in artificial intelligence, machine learning, data mining, pattern recognition, information retrieval, neural computing, and evolutionary computing. In these areas he has published over 40 technical papers in refereed international journals or conference proceedings. He has won the Microsoft Fellowship Award (1999), the National Excellent Doctoral Dissertation Award of China (2003), and the Award of National Outstanding Youth Foundation of China (2004). He is an associate editor of *Knowledge and Information Systems* (Springer), and on the editorial boards of *Artificial Intelligence in Medicine* (Elsevier) and *International Journal of Data Warehousing and Mining* (Idea Group). He served as the organising chair of the 7th Chinese Workshop on Machine Learning (2000), program co-chair of the 9th Chinese Conference on Machine Learning (2004), and program committee member for numerous international conferences. He is a senior member of China Computer Federation (CCF) and the vice chair of CCF Artificial Intelligence & Pattern Recognition Society, a councilor of Chinese Association of Artificial Intelligence (CAAI), the vice chair and chief secretary of CAAI Machine Learning Society, and a member of IEEE and IEEE Computer Society.

Yang Yu received the BSc degree in computer science from Nanjing University, China, in 2004. He has won some awards such as China Computer World Scholarship (2004) and Peoples' Scholarship for outstanding undergraduates. Currently he is a MSc student at the Department of Computer Science & Technology of Nanjing University, and a member of the LAMDA group, supervised by Prof. Zhi-Hua Zhou. His research interests are in machine learning and evolutionary computing.