Pattern Recognition Letters 29 (2008) 1524–1536

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

RotBoost: A technique for combining Rotation Forest and AdaBoost

Chun-Xia Zhang [*], Jiang-She Zhang

School of Science, Xi'an Jiaotong University, Xi'an Shaanxi 710049, People's Republic of China

ARTICLE INFO

Article history:
Received 28 May 2007
Received in revised form 5 January 2008
Available online 18 March 2008

Communicated by N.S.V. Rao

Keywords:
Ensemble method
Base learning algorithm
AdaBoost
Rotation Forest
Bagging
MultiBoost

ABSTRACT

This paper presents a novel ensemble classifier generation technique RotBoost, which is constructed by combining Rotation Forest and AdaBoost. The experiments conducted with 36 real-world data sets available from the UCI repository, among which a classification tree is adopted as the base learning algorithm, demonstrate that RotBoost can generate ensemble classifiers with significantly lower prediction error than either Rotation Forest or AdaBoost more often than the reverse. Meanwhile, RotBoost is found to perform much better than Bagging and MultiBoost. Through employing the bias and variance decompositions of error to gain more insight of the considered classification methods, RotBoost is seen to simultaneously reduce the bias and variance terms of a single tree and the decrement achieved by it is much greater than that done by the other ensemble methods, which leads RotBoost to perform best among the considered classification procedures. Furthermore, RotBoost has a potential advantage over AdaBoost of suiting parallel execution.

1. Introduction

In recent years, the ensemble classifier generation techniques (see for example, Breiman, 1996, 2001; Freund and Schapire, 1996, 1997; Leblanc and Tibshirani, 1996; Latinne et al., 2002; Skurichina and Duin, 2002; Rodríguez et al., 2006; Banfield et al., 2007) are rapidly growing and enjoying a lot of attention from pattern recognition and machine learning communities due to their potential to greatly increase prediction accuracy of a learning system. These techniques generally work by means of firstly generating an ensemble of base classifiers via applying a given base learning algorithm to different permutated training sets, and then the outputs from each ensemble member are combined in a suitable way to create the prediction of the ensemble classifier. The combination is often performed by voting for the most popular class. Examples of these techniques include Bagging (Breiman, 1996), AdaBoost (Freund and Schapire, 1996, 1997), Arcing (Breiman, 1998), Random Subspace (Ho, 1998), Random Forest (Breiman, 2001) and Rotation Forest (Rodríguez et al., 2006).

Among these ensemble learning approaches, AdaBoost (Freund and Schapire, 1996, 1997) has become a very popular one for its simplicity and adaptability and up to now there have been many variants of it (Optiz and Maclin, 1999; Dietterich, 2000; Friedman et al., 2000; Webb, 2000; Meir and Rätsch, 2003; Jin and Zhang, 2007). AdaBoost constructs an ensemble of subsidiary classifiers by applying a given base learning algorithm to successive derived training sets that are formed by either resampling from the original training set (Freund and Schapire, 1996; Breiman, 1998) or reweighting the original training set (Freund and Schapire, 1997) according to a set of weights maintained over the training set. Initially, the weights assigned to each training instance are set to be equal and in subsequent iterations, these weights are adjusted so that the weight of the instances misclassified by the previously trained classifiers is increased whereas that of the correctly classified ones is decreased. Thus, AdaBoost attempts to produce new classifiers that are able to better predict the "hard" instances for the previous ensemble members.

Based on Principal Component Analysis (PCA), Rodríguez et al. (2006) recently proposed a new ensemble classifier generation technique Rotation Forest, and demonstrated that it performs much better than several other ensemble methods on some benchmark classification data sets from the UCI repository (Blake and Merz, 1998). Its main idea is to simultaneously encourage diversity and individual accuracy within an ensemble classifier. Specifically, diversity is promoted by using PCA to do feature extraction for each base classifier and accuracy is sought by keeping all principal components and also using the whole data set to train each base classifier.

In view of the fact that both AdaBoost and Rotation Forest are successful ensemble classifier generation techniques and they apply a given base learning algorithm to the permutated training sets to construct their ensemble members with the only difference lying in the ways to perturb the original training set, it is plausible that a combination of the two may achieve even lower prediction

* Corresponding author. Fax: +86 029 82668559.
E-mail addresses: cxzhang@mail.xjtu.edu.cn (C.-X. Zhang), jszhang@mail.xjtu.edu.cn (J.-S. Zhang).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.03.006

error than either of them. This paper proposes a novel ensemble generation method RotBoost, which is constructed by integrating the ideas of Rotation Forest and AdaBoost. The experimental studies conducted with 36 benchmark real-world classification data sets available from the UCI repository, among which a classification tree (Breiman et al., 1984) is adopted as the base learning algorithm, show that RotBoost can create ensemble classifiers with significantly lower prediction error than either Rotation Forest or AdaBoost more often than the reverse. At the same time, RotBoost is also found to perform much better than Bagging and MultiBoost (Webb, 2000) on the utilized benchmark data sets. Furthermore, two methods for decomposing error into bias and variance terms are employed to gain more insight of the error reduction achieved by RotBoost as well as several other ensemble methods. The experimental results indicate that RotBoost simultaneously reduces the bias and variance terms of a single tree and the decrement achieved by it is much more great than that done by the other ensemble methods, which leads RotBoost to perform best among the considered classification procedures.

The remainder of this paper is organized as follows. Section 2 firstly reviews the ensemble classifier generation approaches AdaBoost and Rotation Forest briefly, and then describes the proposed algorithm RotBoost in detail. The experimental studies are presented in Section 3 and Section 4 offers the conclusions as well as the future work of this paper.

2. RotBoost: a combination of Rotation Forest and AdaBoost

Consider a training set $\mathscr{L} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ consisting of N independent instances, in which each case $(\boldsymbol{x}_i, y_i)$ is described by an input attribute vector $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathbb{R}^p$ and a class label y_i which takes value from the label space $\Phi = \{1, 2, \ldots, J\}$. In a classification task, the goal is to use the information only from $\mathscr{L}$ to construct classifiers that have good generalization capability, namely, perform well on the previously unseen data which are not used for learning the classifiers.

For simplicity of the notations, let X be an $N \times p$ matrix composed of the values of p input attributes for each training instance and Y be an N-dimensional column vector containing the outputs of each training instance in $\mathscr{L}$. Put in another way, $\mathscr{L}$ can be expressed as concatenating X and Y horizontally, that is, $\mathscr{L} = [X\ Y]$. Denote by $C_1, C_2, \ldots, C_T$ the base classifiers included into an ensemble classifier, say, C^*. And let $F = (X_1, X_2, \ldots, X_p)^{\mathrm{T}}$ be the attribute set composed of p input attributes. Before embarking on proposing the RotBoost algorithm, we briefly review the ensemble methods AdaBoost and Rotation Forest as follows.

AdaBoost (Freund and Schapire, 1996, 1997) is a sequential algorithm in which each new classifier is built by taking into account the performance of the previously generated classifiers. Fig. 1 displays the pseudocodes of the AdaBoost algorithm. In this ensemble method, a set of weights $D_t(i)$ $(i = 1, 2, \ldots, N)$ are maintained over the original training set $\mathscr{L}$ and initially they are set to be equal (namely, all training instances have the same importance). In subsequent iterations, these weights are adjusted so that the weight of the instances misclassified by the previously trained classifiers is increased whereas that of the correctly classified ones is decreased. In this way, these "hard" instances can be better predicted by the subsequently trained classifiers. The training set $\mathscr{L}_t$ used for learning each classifier C_t can be obtained by either resampling from the original training set $\mathscr{L}$ (Freund and Schapire, 1996; Breiman, 1998) or reweighting the original training set $\mathscr{L}$ (Freund and Schapire, 1997) according to the updated probability distribution D_t maintained over $\mathscr{L}$. In the current research, the former is chosen due to its simple implementation. Furthermore, each base classifier C_t is assigned to a weight in the training phase and the final decision of the ensemble classifier is obtained by weighted voting of the outputs from each ensemble member.

Note that in the above algorithm, the iteration is carried out until T classifiers are generated or until a classifier achieves an error zero or an accuracy below 0.5, in which cases the weight updating rule fails and the algorithm stops. In order to prevent early stopping when a classifier C_t has $\varepsilon_t > 0.5$, one common variant of AdaBoost is to reset the weight $D_t(i) = 1/N$ $(i = 1, 2, \ldots, N)$ to continue the boosting process (Bauer and Kohavi, 1999). If $\varepsilon_t = 0$, the classifier C_t is incorporated into the ensemble with $\varepsilon_t = 10^{-10}$ instead of $\varepsilon_t = 0$ which will assign an infinite weight to the vote of the classifier C_t for constructing the ensemble classifier C^* (Webb, 2000). These modifications allow the boosting process to always generate the specified number of base classifiers and, in general, increase the prediction accuracy of AdaBoost (Bauer and Kohavi, 1999; Webb, 2000). In the following discussions and experiments, the aforementioned modifications are included into AdaBoost in order to make the comparisons with other ensemble methods that always produce the desired number of base classifiers (such as Bagging and Rotation Forest) much fairer.

Rotation Forest is another newly proposed successful ensemble classifier generation technique (Rodríguez et al., 2006), in which the training set for each base classifier is formed by applying PCA to rotate the original attribute axes. Specifically, to create the training data for a base classifier, the attribute set F is randomly split into K subsets (K is a parameter of the algorithm) and PCA is applied to each subset. All principal components are retained in order to preserve the variability information in the data. Thus, K axis rotations take place to form the new attributes for a base classifier. The main idea of Rotation Forest is to simultaneously encourage diversity and individual accuracy within the ensemble: diversity is promoted through doing feature extraction for each base classifier and accuracy is sought by keeping all principal components and also using the whole data set to train each base classifier. The detailed steps of Rotation Forest are described in Fig. 2. It is worthwhile to point out that in step (b) of the algorithm listed in Fig. 2, the aim to select the sample size of $X'_{t,k}$ to be smaller than that of $X_{t,k}$ is twofold: one is to avoid obtaining identical coefficients of the principal components if the same attribute subset is chosen for different base classifiers and the other is to increase the diversity among the generated ensemble members.

As indicated by many researchers (Hansen and Salamon, 1990; Optiz and Shavlik, 1996; Dietterich, 1997; Rodríguez et al., 2006), for an ensemble classifier to achieve much better generalization capability than its constituent members, it is critical that the ensemble classifier consists of highly accurate members which at the same time disagree as much as possible. It has also been noted by Krogh and Vedelsby (1995) that the prediction accuracy of an ensemble classifier can be further improved on condition that the diversity of the ensemble members is increased whereas their individual errors are not affected. Using multiple approaches to generate ensemble members should increase diversity in the ensemble membership and if this can be done without greatly increasing the error in the individual predictions, it can be expected to decrease error of the resulting ensemble classifier. In view of this fact, we propose in this paper a novel ensemble classifier construction method RotBoost through employing both Rotation Forest and AdaBoost to generate base classifiers. Fig. 3 gives the pseudocodes of this algorithm.

When employing the above proposed RotBoost algorithm to solve a classification task, some parameters included in it should be specified in advance. As with the most ensemble methods, the values of the parameters S and T that, respectively, specify the numbers of iterations done for Rotation Forest and AdaBoost can be subjectively determined by the user and the value of K (or M) can be chosen to be a moderate value according to the size of the

Input

- $\mathscr{L}$: a training set, $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ in which $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip}) \in \mathbb{R}^p$ and $y_i \in \Phi = \{1, 2, \cdots, J\}$.
- $\mathscr{W}$: a base learning algorithm.
- T: number of iterations.
- $\mathbf{x}$: a data point to be classified.

Training Phase

Initialize the weight distribution over $\mathscr{L}$ as $D_1(i) = 1/N \ (i = 1, 2, \cdots, N)$.

For $t = 1, \cdots, T$

1. According to the distribution D_t, draw N training instances at random from $\mathscr{L}$ with replacement to compose a new set $\mathscr{L}_t = \{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^N$.
2. Provide $\mathscr{L}_t$ as the input of $\mathscr{W}$ to train a classifier C_t, and then compute the error of C_t as
$$\varepsilon_t = \mathrm{Pr}_{i \sim D_t}(C_t(\mathbf{x}_i) \neq y_i) = \sum_{i=1}^N I(C_t(\mathbf{x}_i) \neq y_i) D_t(i),$$
where $I(\cdot)$ is the indicator function which takes value 1 or 0 depending on whether the ith training instance is misclassified by C_t or not.
3. If $\varepsilon_t > 0.5$ or $\varepsilon_t = 0$, then set $T = t - 1$ and abort loop.
4. Let $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.
5. Update the weight distribution D_t over $\mathscr{L}$ as
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } C_t(\mathbf{x}_i) = y_i \\ e^{\alpha_t}, & \text{if } C_t(\mathbf{x}_i) \neq y_i \end{cases}$$
where Z_t is a normalization factor being chosen so that D_{t+1} is a probability distribution over $\mathscr{L}$.

Endfor

Output

- the class label for $\mathbf{x}$ predicted by the ensemble classifier C^* as
$$C^*(\mathbf{x}) = \underset{y \in \Phi}{\mathrm{argmax}} \sum_{t=1}^T \alpha_t I(C_t(\mathbf{x}) = y).$$

Fig. 1. The AdaBoost algorithm.

attribute set *F*. Since the good performance of an ensemble method largely depends on the instability of the used base learning algorithm (Breiman, 1996, 1998; Dietterich, 2000), $\mathscr{W}$ can be therefore generally selected to be either a classification tree (Breiman et al., 1984) or a neural network (Hansen and Salamon, 1990) which is instable in the sense that small permutations in its training data or in its construction can lead to large changes in the constructed predictor (Breiman, 1996, 1998).

3. Experimental studies

In this section, the performance of RotBoost is examined and compared with that of several other approaches for constructing ensemble classifiers using 36 benchmark real-world classification data sets publicly available from the UCI repository (Blake and Merz, 1998). In the following discussions, the considered classifier combination methods mainly contain RotBoost, Bagging (Breiman, 1996), AdaBoost (Freund and Schapire, 1997), MultiBoost (Webb, 2000) and Rotation Forest (Rodríguez et al., 2006). Considering that the main ideas of RotBoost are similar to those of MultiBoost which is a combination of Bagging and AdaBoost (Webb, 2000), here we also took MultiBoost into account to check whether RotBoost can outperform it. At the same time, the results computed with the base learning algorithm were also considered for a complete comparison. In order to simplify the notations, Rotation Forest was abbreviated as RotForest when reporting the experimental results in the following descriptions.

```
Input
  • $\mathscr{L}$: a training set, $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} = [X\ Y]$ where $X$ is an $N \times p$ matrix containing
    the input attribute values and $Y$ is an $N$-dimensional column vector containing the
    class labels.
  • $K$: number of attribute subsets (or $M$: number of input attributes contained in each
    subset).
  • $\mathscr{W}$: a base learning algorithm.
  • $T$: number of iterations.
  • $\mathbf{x}$: a data point to be classified.
Training Phase
For $t = 1, 2, \cdots, T$
  • Calculate the rotation matrix $R_t^a$ for the $t$th classifier $C_t$.
    1. Randomly split the attribute set $F$ into $K$ subsets $F_{t,k}$ $(k = 1, 2, \cdots, K)$.
    2. For $k = 1, 2, \cdots, K$
       (a) Select the columns of $X$ that correspond to the attributes in $F_{t,k}$ to compose a
           submatrix $X_{t,k}$.
       (b) Draw a bootstrap sample $X'_{t,k}$ (with sample size generally smaller than that of
           $X_{t,k}$) from $X_{t,k}$.
       (c) Apply PCA to $X'_{t,k}$ to obtain a matrix $D_{t,k}$ whose $i$th column consists of the
           coefficients of the $i$th principal component.
    3. EndFor
    4. Arrange the matrices $D_{t,k}$ $(k = 1, 2, \cdots, K)$ into a block diagonal matrix $R_t$.
    5. Construct the rotation matrix $R_t^a$ by rearranging the rows of $R_t$ so that they cor-
       respond to the original attributes in $F$.
  • Provide $[XR_t^a\ Y]$ as the input of $\mathscr{W}$ to build a classifier $C_t$.
EndFor
Output
  • the class label for $\mathbf{x}$ predicted by the ensemble classifier $C^*$ as
        $C^*(\mathbf{x}) = \underset{y \in \Phi}{\operatorname{argmax}} \sum_{t=1}^{T} I(C_t(\mathbf{x}R_t^a) = y)$,
    where $I(\cdot)$ is an indicator function.
```

Fig. 2. The Rotation Forest algorithm.

Table 1 summarizes the main characteristics of the data sets utilized in our empirical study. This selection includes data sets with different characteristics and from a variety of fields. It should be pointed out that the "House" data set was originally a regression task. Following the technique utilized by Lim et al. (2000), the classes were created from the attribute median value of owner-occupied homes as follows: class = 1 if log(median value) $\leqslant$ 9.84, class = 2 if 9.84 < log(median value) $\leqslant$ 10.075, class = 3 otherwise.

In Table 1, the first three columns, respectively, give the name, sample size and number of classes of each data set. The other columns summarize the information of input attributes, among which the numbers of numerical, categorical and total attributes contained in each data set are all given in detail. The figures given in the third row of Table 1 indicate the number of possible categories that a categorical attribute has. For each data set, the numbers listed in the columns from 5 to 13 denote how many the corresponding categorical attributes are contained in this set. For example, "Cmc" contains seven categorical input attributes, in which three attributes are binary-valued and four attributes are four-valued. Furthermore, some locations in a row are left blank when the data set does not include the corresponding input attributes in it.

When preprocessing the data, the instances that contain missing values are deleted from the data sets and the categorical attri-

```
Input
  • $\mathscr{L}$: a training set, $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} = [X\ Y]$ where $X$ is an $N \times p$ matrix containing
    the input attribute values and $Y$ is an $N$-dimensional column vector containing the
    class labels.
  • $K$: number of attribute subsets (or $M$: number of input attributes contained in each
    subset).
  • $\mathscr{W}$: a base learning algorithm.
  • $S$: number of iterations for Rotation Forest.
  • $T$: number of iterations for AdaBoost.
  • $\mathbf{x}$: a data point to be classified.
Training Phase
For $s = 1, 2, \cdots, S$
  1. Use the steps similar to those in Rotation Forest to compute the rotation matrix, say,
     $R_s^a$ and let $\mathscr{L}^a = [XR_s^a\ Y]$ be the training set for classifier $C_s$.
  2. Initialize the weight distribution over $\mathscr{L}^a$ as $D_1(i) = 1/N\ (i = 1, 2, \cdots, N)$.
  3. For $t = 1, \cdots, T$
     (a) According to the distribution $D_t$, perform $N$ extractions randomly from $\mathscr{L}^a$ with
         replacement to compose a new set $\mathscr{L}_t^a$.
     (b) Apply $\mathscr{W}$ to $\mathscr{L}_t^a$ to train a classifier $C_t^a$, and then compute the error of $C_t^a$ as
             $\varepsilon_t = \Pr_{i \sim D_t}(C_t^a(\mathbf{x}_i) \neq y_i) = \sum_{i=1}^{N} I(C_t^a(\mathbf{x}_i) \neq y_i) D_t(i).$
     (c) If $\varepsilon_t > 0.5$, then set $D_t(i) = 1/N\ (i = 1, 2, \cdots, N)$ and goto step (a); if $\varepsilon_t = 0$, then
         set $\varepsilon_t = 10^{-10}$ to continue the following iterations.
     (d) Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
     (e) Update the distribution $D_t$ over $\mathscr{L}^a$ as
             $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } C_t^a(\mathbf{x}_i) = y_i \\ e^{\alpha_t}, & \text{if } C_t^a(\mathbf{x}_i) \neq y_i \end{cases}$
         where $Z_t$ is a normalization factor being chosen so that $D_{t+1}$ is a probability dis-
         tribution over $\mathscr{L}^a$.
  4. Endfor
  5. Let $C_s(\mathbf{x}) = \underset{y \in \Phi}{\operatorname{argmax}} \sum_{t=1}^{T} \alpha_t I(C_t^a(\mathbf{x}) = y)$.
Endfor
Output
  • the class label for $\mathbf{x}$ predicted by the final ensemble classifier $C^*$ as
        $C^*(\mathbf{x}) = \underset{y \in \Phi}{\operatorname{argmax}} \sum_{s=1}^{S} I(C_s(\mathbf{x}) = y).$
```

Fig. 3. The RotBoost algorithm.

butes are converted into numerical ones considering that PCA is defined for numerical attributes. Each categorical attribute was replaced by *c* binary ones encoded numerically as 0 and 1, where *c* is the number of possible categories of the categorical attribute. This encoding mechanism unifies all the inputs so that PCA can be carried out on any attribute subset.

Table 1
Summary of the used data sets

Data set	Size	Classes	Number of input attributes										
			Numerical	Categorical									Total
				2	3	4	5	8	9	12	13	14	
Abalone	4177	3	7		1								8
Australian	690	2	6	4	2				1			1	14
Balance	625	3	4										4
Bcs	286	2	1	3	1		1		1	1	1		9
Bcw	683	2	9										9
Car	1748	4	4		3	3							6
Cleveland	297	5	6	3	3	1							13
Cmc	1473	3	2	3		4							9
Creditr	653	2	6	4	2	1			1			1	15
Dermatology	366	6	34										34
Ecoli	336	8	7										7
German	1000	2	24										24
Glass	214	7	10										10
Haberman	306	2	3										3
Heart	270	2	6	3	3	1							13
Hepatitis	80	2	6	13									19
House	506	3	12	1									13
Ionosphere	351	2	34										34
Iris	150	3	4										4
Liver	345	2	6										6
Lymphography	148	4		9	4	3		2					18
Pendigits	10992	10	16										16
Pima	768	2	8										8
Segmentation	2310	7	19										19
Sonar	208	2	60										60
Soybean	683	19	35										35
Spect	267	2		22									4
Spectf	269	2	44										44
Vehicle	846	4	18										18
Votes	435	2		16									16
Vowelc	990	11	10	1									11
Wdbc	569	2	30										30
Wine	178	3	13										13
Wpbc	194	2	32										32
Yeast	1484	10	8										8
Zoo	101	7	1	15									16

Because there have no separate training and testing data for these data sets except for "Pendigits", the cross-validation method was employed to compute the error rate of each classification procedure. Since there is no expert on what the "right" number of folds is for cross-validation and Webb (2000) utilized 10 runs of threefold cross-validation method to estimate the prediction errors of his considered methods and obtained satisfactory results, we also adopted the threefold cross-validation procedure to achieve our task. Considering that 10 runs of trial may be too small to get reliable results, we conducted 20 runs of the threefold cross-validation method for each combination of data set and classification procedure, and then the error rate averaged over 20 trials was utilized to evaluate the performance of that classification method. As for the "Pendigits" data set, we firstly combined its training and testing data together, and then executed the above process so that the results obtained with it are comparable to those calculated with the other data sets.

The experimental settings were as follows. In all the ensemble methods, a classification tree (Breiman et al., 1984) was always adopted as the base learning algorithm because it is sensitive to the changes in its training data and can still be very accurate. The following experiments were all conducted in Matlab software with version 7.1. The classification tree algorithm was realized by the "Treefit" algorithm contained in Matlab. The parameters included in this algorithm, such as the number of training instances that impure nodes to be split must have, were all set to be the default values. The implementations of the considered ensemble classifier generation methods were all realized in Matlab by writing programs according to their pseudocodes which are described in the corresponding papers. For the techniques Bagging, AdaBoost, MultiBoost and RotForest, 100 trees were trained to constitute the corresponding ensemble classifiers. With respect to RotBoost, the number of iterations done for Rotation Forest and AdaBoost were both taken to be 10 so that an ensemble classifier created by it also consists of 100 trees. As for the parameter M (namely, the number of attributes contained in each attribute subset) included in RotForest and RotBoost, the value of it was taken to be 2 for the data sets with size of the attribute set F smaller than or equal to 10 and to be 3 for the data sets with size of F greater than 10.

3.1. Comparison of prediction error

Table 2 reports the means of prediction error (expressed in %) for each classification method on the considered data sets, where the values following "±" are their respective standard deviations. In order to see whether RotBoost is significantly better or worse than other methods from the statistical viewpoint, a one-tailed paired t-test was performed with significance level $\alpha = 0.05$ and the results for which a significant difference with RotBoost was found are marked with a bullet or an open circle next to them. A bullet next to a result indicates that RotBoost is significantly better than the corresponding method. An open circle next to a result denotes that RotBoost performs significantly worse than the corresponding method. In the triplet labeled "Win–Tie–Loss" in the last row of Table 2, the first value is the number of data sets on

Table 2
Means and standard deviations of prediction error (expressed in %) for each algorithm on the benchmark data sets

Data set	RotBoost	SingleTree	Bagging	AdaBoost	MultiBoost	RotForest
Abalone	35.63 ± 0.48	41.79 ± 0.89•	36.26 ± 0.44•	37.10 ± 0.59•	36.23 ± 0.42•	34.98 ± 0.27 ∘
Australian	13.46 ± 0.79	16.97 ± 1.12•	13.09 ± 0.54 ∘	13.66 ± 0.75	13.22 ± 0.82	13.66 ± 0.78
Balance	11.26 ± 1.37	21.72 ± 1.19•	15.03 ± 0.93•	21.62 ± 0.75•	18.03 ± 0.83•	9.62 ± 0.72 ∘
Bcs	29.19 ± 1.60	33.78 ± 1.65•	29.06 ± 1.37	32.22 ± 2.17•	30.80 ± 1.24•	30.44 ± 2.97•
Bcw	3.21 ± 0.37	5.53 ± 0.71•	3.45 ± 0.34•	3.15 ± 0.35	3.13 ± 0.30	2.89 ± 0.33 ∘
Car	3.94 ± 0.37	5.35 ± 0.51•	3.91 ± 0.27	4.01 ± 0.27	3.91 ± 0.41	3.63 ± 0.29 ∘
Cleveland	43.80 ± 1.28	47.96 ± 2.08•	44.09 ± 2.05	45.55 ± 1.97•	44.48 ± 1.73	44.38 ± 1.47
Cmc	47.68 ± 0.98	49.23 ± 1.13•	46.14 ± 1.11∘	48.52 ± 0.84•	47.20 ± 0.91	46.97 ± 0.78 ∘
Creditr	12.66 ± 0.75	16.78 ± 1.21•	13.38 ± 1.01•	12.70 ± 0.72	13.25 ± 0.77•	13.18 ± 0.66•
Dermatology	2.80 ± 0.61	5.37 ± 0.81•	3.60 ± 0.78•	3.14 ± 0.80	3.02 ± 0.79	2.62 ± 0.68
Ecoli	15.37 ± 1.02	19.82 ± 1.56•	17.23 ± 1.36•	15.70 ± 1.16	14.99 ± 1.16	16.37 ± 1.33•
German	24.56 ± 0.85	35.49 ± 1.37•	28.24 ± 0.70•	25.38 ± 0.89•	24.87 ± 0.72	28.47 ± 0.83•
Glass	24.46 ± 0.87	24.49 ± 0.84	24.53 ± 0.92	24.56 ± 0.88	24.37 ± 0.83	24.39 ± 0.56
Haberman	31.37 ± 2.13	32.93 ± 3.27•	29.75 ± 1.76∘	33.76 ± 2.09•	32.06 ± 2.01	31.32 ± 2.24
Heart	18.67 ± 1.45	31.26 ± 2.72•	23.28 ± 1.67•	21.91 ± 2.16•	20.80 ± 1.44•	21.09 ± 1.31•
Hepatitis	32.75 ± 4.81	41.25 ± 4.42•	33.62 ± 2.63	36.62 ± 5.10•	35.69 ± 4.62•	33.81 ± 4.51
House	21.07 ± 1.36	27.15 ± 1.46•	21.90 ± 1.26•	21.56 ± 1.16	21.18 ± 1.04	21.33 ± 1.00
Ionosphere	5.57 ± 0.76	12.37 ± 1.18•	8.46 ± 0.74•	6.30 ± 0.77•	6.27 ± 0.80•	5.53 ± 0.72
Iris	4.53 ± 0.91	5.63 ± 1.32•	4.70 ± 1.23	5.33 ± 1.08•	5.80 ± 1.15•	4.37 ± 1.00
Liver	30.35 ± 1.81	35.85 ± 2.31•	29.80 ± 2.13	30.59 ± 1.48	29.04 ± 2.07∘	29.68 ± 1.99
Lymphography	26.05 ± 13.44	32.91 ± 11.07•	33.92 ± 10.93•	29.16 ± 13.66	23.92 ± 13.19	34.32 ± 11.30 •
Pendigits	0.62 ± 0.06	4.75 ± 0.23•	1.99 ± 0.08•	0.65 ± 0.06•	0.71 ± 0.05•	0.65 ± 0.04•
Pima	24.62 ± 1.19	29.40 ± 1.23•	24.04 ± 0.77	25.74 ± 0.90•	24.79 ± 0.76	25.31 ± 1.14
Segmentation	1.83 ± 0.19	4.87 ± 0.34•	3.25 ± 0.24•	1.66 ± 0.25∘	1.79 ± 0.19	2.03 ± 0.25•
Sonar	17.23 ± 2.16	30.10 ± 2.36•	23.08 ± 1.82•	17.98 ± 2.14	18.63 ± 2.23•	17.26 ± 2.24
Soybean	6.50 ± 0.78	11.36 ± 1.31•	8.38 ± 0.81•	6.59 ± 0.79	6.54 ± 0.65	6.99 ± 0.65•
Spect	17.98 ± 1.17	20.24 ± 2.17•	17.08 ± 1.50∘	18.45 ± 0.91	18.67 ± 0.90•	17.71 ± 1.18
Spectf	19.66 ± 1.57	25.82 ± 2.44•	19.54 ± 1.33	19.42 ± 1.38	18.72 ± 2.03	19.03 ± 1.49
Vehicle	21.92 ± 1.09	30.68 ± 1.25•	25.79 ± 0.97•	23.68 ± 1.06•	23.65 ± 0.89•	21.82 ± 1.00
Votes	4.74 ± 0.49	5.41 ± 0.66•	4.57 ± 0.64	5.08 ± 0.66•	4.81 ± 0.72	4.22 ± 0.72∘
Vowelc	5.33 ± 0.71	27.41 ± 1.40•	13.62 ± 0.82•	6.25 ± 0.76•	7.28 ± 1.27•	6.48 ± 0.83•
Wdbc	3.22 ± 0.37	7.57 ± 0.98•	4.84 ± 0.48•	3.53 ± 0.56•	3.44 ± 0.52	4.93 ± 0.64•
Wine	3.06 ± 0.67	9.38 ± 1.77•	4.55 ± 1.77•	3.31 ± 0.81	3.23 ± 1.01	5.11 ± 1.77•
Wpbc	25.49 ± 1.47	35.34 ± 2.98•	25.93 ± 2.00	26.37 ± 1.69•	25.82 ± 1.56	27.91 ± 2.49•
Yeast	39.78 ± 1.05	47.68 ± 1.22•	40.25 ± 0.82	41.27 ± 0.92•	40.04 ± 0.68	40.62 ± 0.93•
Zoo	5.49 ± 1.83	12.37 ± 3.68•	10.74 ± 3.00•	6.44 ± 2.73	7.28 ± 2.91•	11.43 ± 3.07•
Win–Tie–Loss		35–1–0	20–12–4	19–16–1	14–21–1	15–15–6

"•" indicates that RotBoost is significantly better and "∘" denotes that RotBoost is significantly worse at the significance level $\alpha = 0.05$.

which RotBoost performs significantly better than the corresponding algorithm; the second one is the number of data sets on which the difference between the performance of RotBoost and that of the corresponding algorithm is not significant; and the third one denotes the number of data sets on which RotBoost behaves significantly worse than the compared algorithm.

As can be seen from Table 2, RotBoost performs significantly better than SingleTree on all but the "Glass" data set where RotBoost has lower error than SingleTree but the difference between them is not significant. When compared with Bagging, the statistically significant difference is favorable in 20 sets, unfavorable in 4 sets and not significant in 12 sets for RotBoost. Meanwhile, RotBoost is seen to outperform AdaBoost, MultiBoost and RotForest in most cases although the advantage of RotBoost is not significant in some cases.

Furthermore, it is well known that no algorithm can hold a general advantage in terms of generalization capability over another one across all possible classification tasks. However, the relative advantage of an algorithm is possible across a set of real-world tasks, or at least a specific subset thereof (Webb, 2000). Considering that the 36 data sets taken from the UCI repository include different characteristics and come from various fields, these sets may be representative of those in the UCI repository and hence may be at the very least representative of those commonly used in machine learning research. Therefore, it should be possible to use statistical analysis to draw some conclusions relating to the expected general relative performance of the previously compared classification algorithms over at least this restricted class of learning tasks. Here we employed some statistics utilized by Webb (2000) to compare the relative performance of the considered algorithms.

Table 3 provides the comparative summaries of the mean errors computed previously for each algorithm across all the used data sets and it has the following format. In the following descriptions,

Table 3
Comparison of error for each algorithm across all the used data sets

Algorithm	SingleTree	Bagging	AdaBoost	MultiBoost	RotForest	RotBoost
Mean over all sets (%)	23.50	19.20	18.86	18.27	18.46	17.66
SingleTree						
$\dot{r}$		0.764	0.684	0.674	0.679	0.636
s		34–0–2	34–0–2	35–0–1	35–0–1	36–0–0
q		<0.001	<0.001	<0.001	<0.001	<0.001
Bagging						
$\dot{r}$			0.895	0.882	0.888	0.833
s			19–0–17	24–1–11	21–0–15	26–0–10
q			0.868	0.041	0.405	0.011
AdaBoost						
$\dot{r}$				0.986	0.993	0.931
s				28–0–8	23–2–11	33–0–3
q				0.001	0.058	<0.001
MultiBoost						
$\dot{r}$					1.007	0.944
s					19–0–17	26–0–10
q					0.868	0.011
RotForest						
$\dot{r}$						0.938
s						21–0–15
q						0.405

row indicates the mean error on a data set for the algorithm labeled in a row and *col* stands for the mean error on a data set for the algorithm labeled in a column. Rows labeled as $\dot{r}$ present the geometric mean of error ratio *col*/*row*. Rows labeled as *s* give the "win–tie–loss" statistic, where the three values, respectively, refers to the numbers of data sets for which $col < row$, $col = row$ and $col > row$. Rows labeled as *q* report the test *p*-values of a two-tailed sign test based on the win–tie–loss record. If the value of *q* is smaller than the given significance level, the difference between the two compared algorithms is significant and otherwise it is not significant. If the Bonferroni correction is taken into consideration to control type I error for multiple tests, the difference between two algorithms should be deemed as significant only if the value of *q* is smaller than $\alpha/36$.

The results in Table 3 show that RotBoost obtains the lowest mean error among the compared algorithms and all of the comparative metrics rank these algorithms in the same order from best to worst as RotBoost, MultiBoost, RotForest, AdaBoost, Bagging and SingleTree, except that the win–tie–loss record favors RotForest over MultiBoost. Consider the relative performance of RotBoost in particular, it achieves a lower geometric mean ratio of error over that of SingleTree than any other ensemble method. The geometric means of error ratios computed for RotBoost over each other algorithm all favor RotBoost. Compared with every other algorithm, RotBoost always achieves lower error for more data sets than the other algorithms and the advantage of RotBoost over all the other algorithms but RotForest is statistically significant when the significance level α is set to be 0.05. If taking into account the Bonferroni correction criterion, RotBoost is observed to be only significantly better than SingleTree and AdaBoost.

In Tables 2 and 3, RotBoost is seen to frequently beat either AdaBoost or RotForest. However, one may naturally ask whether it obtains a midpoint of the two or it usually outperforms the better of the two. In fact, the error of RotBoost is lower than the minimum of the errors of AdaBoost and RotForest for 20 data sets and higher for 16 ones.

3.2. Bias/variance analysis of error performance

In order to investigate the reasons for the better performance of an ensemble classifier than its constituent members, many researchers (for example, see Breiman, 1998; Bauer and Kohavi, 1999; Optiz and Maclin, 1999; Webb, 2000; Zhou et al., 2002; James, 2003; Rodríguez et al., 2005) have studied this problem by decomposing error into bias and variance terms. The decomposition of a learning machine's error into bias and variance terms originates from analyzing learning models with numeric outputs in regression problems (Geman et al., 1992). Given a fixed target and training set size, the conventional formulation of the decomposition breaks the expected error into the sum of three non-negative quantities:

- *Intrinsic "target noise"* (σ^2)*:* This quantity is a lower bound on the expected error of any learning algorithm. It is the expected error of the Bayes optimal classifier.
- *Squared "bias"* (Bias^2)*:* This quantity measures how closely the learning algorithm's average guess (over all possible training sets of the given size) matches the target.
- *"Variance"* (Var)*:* This quantity measures how much the learning algorithm's guess fluctuates from the target for the different training sets of the given size.

For classification problems, the above decomposition is inappropriate because class labels are not numeric. Although the class labels utilized in the previous discussions are denoted by the integers from 1 to *J*, they are only used to discriminate the classes and have not any real meaning. Since there is not a proper manner to translate the decomposition of error in regression tasks to contexts where the value to be predicted is categorical, a number of ways to decompose error into bias and variance terms in the field of classification prediction tasks have been proposed (Kong and Dietterich, 1995; Kohavi and Wolpert, 1996; Friedman, 1997; Breiman, 1998; James, 2003). Each of these definitions is able to provide some valuable insight into different aspects of a learning machine's performance (Webb, 2000; Nadeau and Bengio, 2003; Webb and Conilione, 2006). Through analyzing the performance of a classification procedure in terms of bias and variance, AdaBoost is found to reduce both the bias and variance terms of error while Bagging is found to reduce the variance term only (Bauer and Kohavi, 1999; Breiman, 1998). To the best of our knowledge, Rodríguez et al. (2005) utilized the method proposed by Webb and Conilione (2006) to compute the bias and variance defined by Kohavi and Wolpert (1996) to further investigate the performance of Rotation Forest and found that Rotation Forest reduces both bias and variance while the reduction of bias achieved by it is smaller than that done by AdaBoost. Webb (2000) also adopted several ways of decomposing error into bias and variance to examine and compare the performance of MultiBoost with that of several other ensemble methods and he drew a conclusion that MultiBoost is more effective at reducing bias than Bagging and is more effective at reducing variance than AdaBoost. In order to gain more insight into the performance of the proposed method RotBoost, two related bias/variance definitions (Kohavi and Wolpert, 1996; Bauer and Kohavi, 1999) are used in the current research and they are, respectively, denoted by $\text{Bias}^2_{\text{BK}}/\text{Var}_{\text{BK}}$ and $\text{Bias}^2_{\text{KW}}/\text{Var}_{\text{KW}}$ in the following discussions.

Theoretically, the prediction error of a classifier should also be decomposed into three terms (that is, irreducible error, squared bias and variance), however, it is usually difficult to estimate irreducible error in real-world learning tasks for which the true underlying class distribution is unknown and there are generally too few instances at any given point in the instance space to reliably estimate the class distribution at that point. In the commonly used methods to compute bias and variance terms, the irreducible error is generally aggregated into both bias and variance or only the bias term due to the fact that irreducible error is invariant across learning algorithms for a single learning task and hence not a significant factor in comparative evaluations.

If denote by Y_{H} the random variable representing the evaluated label of an instance and Y_{F} the random variable standing for the true label of an instance, $\text{Bias}^2_{\text{BK}}/\text{Var}_{\text{BK}}$ and $\text{Bias}^2_{\text{KW}}/\text{Var}_{\text{KW}}$ defined for a testing instance $(\boldsymbol{x}, y)$ can be expressed as

$$\begin{cases} \text{Bias}^2_{\text{BK}}(\boldsymbol{x}) = \Pr(Y_{\text{H}}^{\mathscr{L}} \neq y \ \& \ Y_{\text{H}}^{\mathscr{L}} = y^c|\boldsymbol{x}), \\ \text{Var}_{\text{BK}}(\boldsymbol{x}) = \Pr(Y_{\text{H}}^{\mathscr{L}} \neq y \ \& \ Y_{\text{H}}^{\mathscr{L}} \neq y^c|\boldsymbol{x}), \end{cases}$$

and

$$\begin{cases} \text{Bias}^2_{\text{KW}}(\boldsymbol{x}) = \frac{1}{2}\sum\limits_{y' \in \Phi} \left[\Pr(Y_{\text{F}} = y'|\boldsymbol{x}) - \Pr(Y_{\text{H}}^{\mathscr{L}} = y'|\boldsymbol{x})\right]^2, \\ \text{Var}_{\text{KW}}(\boldsymbol{x}) = \frac{1}{2}\left\{1 - \sum\limits_{y' \in \Phi} [\Pr(Y_{\text{H}}^{\mathscr{L}} = y'|\boldsymbol{x})]^2\right\}, \end{cases}$$

where $y^c = \arg\max_{y^* \in \Phi} \Pr(Y_{\text{H}}^{\mathscr{L}} = y^*)$ is the central tendency of instance $\boldsymbol{x}$ predicted by the learning machines that are trained on different training sets drawn from the same distribution. Here, the superscript $\mathscr{L}$ is used in $Y_{\text{H}}^{\mathscr{L}}$ to denote that the evaluated class label is predicted by the machine trained on the set $\mathscr{L}$. The term $\Pr(\cdot)$ in the above formulae can be computed as the frequency that the event included in the parentheses occurs in the trials which are conducted with different training sets of the given size.

To compute the above statistics, the distribution that the training data come from should be known in advance. However, the training distribution is generally unknown and the knowledge we have in a standard machine learning experimental situation is only a small sample. In consequence, these terms should be estimated instead. However, there have not come to a consensus on how to achieve this task. Among the estimation techniques, some researchers (Bauer and Kohavi, 1999; Zhou et al., 2002) used a two-stage process to evaluate the values of $\mathrm{Bias}^2_{\mathrm{KW}}(\boldsymbol{x})$ and $\mathrm{Var}_{KW}(\boldsymbol{x})$. Webb (2000) proposed to conduct 10 runs of the three-fold cross-validation method to estimate the above-mentioned bias and variance terms. Valentini and Dietterich (2004) presented an alternative procedure based on the out-of-bag technique to compute bias and variance from data. Furthermore, Webb and Conilione (2006) also suggested an approach based on cross-validation to estimate a learning machine's error as well as the bias and variance decomposition of it (Kohavi and Wolpert, 1996).

In our experimental situations, the method similar to that used by Webb (2000), that is, multiple trials of the threefold cross-validation procedure, was utilized to estimate the bias and variance terms defined above. However, 20 trials instead of 10 trials which were adopted by Webb (2000) were conducted in the following experiments in view of the fact that 10 trials may be not enough to obtain reliable results and if much more trials are run, the computational cost may be too expensive for some large data sets while the improvement of the obtained estimates resulting from the additional computation is negligible. The main steps of the estimation process are described as follows: the available data, say, $\mathscr{D}$, are firstly divided into threefolds f_1, f_2 and f_3 at random and this process is repeated 20 times to produce 60-folds $f_1^1, f_2^1, f_3^1, f_1^2, f_2^2, f_3^2, \ldots, f_1^{20}, f_2^{20}, f_3^{20}$. Then each of the fold from a triplet is used as a test set once for a classifier learned from the other two-

Table 4
Comparison of contribution of $\mathrm{Bias}^2_{\mathrm{BK}}$ to error

Algorithm	SingleTree	Bagging	AdaBoost	MultiBoost	RotForest	RotBoost
Mean over all sets (%)	13.45	14.47	13.77	13.64	13.64	13.04
SingleTree						
$\dot{r}$		1.017	0.877	0.880	0.883	0.812
s		11–0–25	18–0–18	19–0–17	18–2–16	21–0–15
q		0.029	>0.999	0.868	0.864	0.405
Bagging						
$\dot{r}$			0.862	0.865	0.868	0.798
s			26–0–10	26–0–10	28–0–8	29–0–7
q			0.011	0.011	0.001	<0.001
AdaBoost						
$\dot{r}$				1.003	1.006	0.926
s				16–2–18	20–0–16	27–1–8
q				0.864	0.618	0.002
MultiBoost						
$\dot{r}$					1.003	0.923
s					20–0–16	26–1–9
q					0.618	0.006
RotForest						
$\dot{r}$						0.920
s						24–0–12
q						0.065

Table 5
Comparison of contribution of $\mathrm{Var}_{\mathrm{BK}}$ to error

Algorithm	SingleTree	Bagging	AdaBoost	MultiBoost	RotForest	RotBoost
Mean over all sets (%)	10.05	4.73	5.09	4.63	4.82	4.63
SingleTree						
$\dot{r}$		0.430	0.425	0.398	0.398	0.394
s		35–0–1	34–0–2	35–0–1	35–0–1	35–0–1
q		<0.001	<0.001	<0.001	<0.001	<0.001
Bagging						
$\dot{r}$			0.990	0.926	0.926	0.917
s			16–1–19	18–0–18	19–0–17	19–1–16
q			0.736	>0.999	0.868	0.736
AdaBoost						
$\dot{r}$				0.936	0.936	0.926
s				27–2–7	23–0–13	25–0–11
q				0.001	0.133	0.029
MultiBoost						
$\dot{r}$					1.000	0.990
s					17–0–19	14–1–21
q					0.868	0.311
RotForest						
$\dot{r}$						0.990
s						18–1–17
q						>0.999

Table 6
Comparison of contribution of $\mathrm{Bias}^2_{\mathrm{KW}}$ to error

Algorithm	SingleTree	Bagging	AdaBoost	MultiBoost	RotForest	RotBoost
Mean over all sets (%)	13.72	14.22	13.39	13.36	13.46	12.73
SingleTree						
$\dot{r}$		0.967	0.830	0.839	0.847	0.775
s		14–0–22	18–1–17	21–0–15	20–0–16	25–0–11
q		0.243	>0.999	0.405	0.618	0.029
Bagging						
$\dot{r}$			0.858	0.867	0.876	0.802
s			27–0–9	27–0–9	27–0–9	30–0–6
q			0.004	0.004	0.004	<0.001
AdaBoost						
$\dot{r}$				1.010	1.021	0.934
s				12–1–23	16–1–19	29–1–6
q				0.896	0.736	<0.001
MultiBoost						
$\dot{r}$					1.010	0.924
s					19–0–17	24–0–12
q					0.868	0.065
RotForest						
$\dot{r}$						0.915
s						27–1–8
q						0.002

Table 7
Comparison of contribution of $\mathrm{Var}_{\mathrm{KW}}$ to error

Algorithm	SingleTree	Bagging	AdaBoost	MultiBoost	RotForest	RotBoost
Mean over all sets (%)	9.78	4.97	5.48	4.90	5.01	4.93
SingleTree						
$\dot{r}$		0.482	0.481	0.443	0.437	0.437
s		36–0–0	36–0–0	35–0–1	35–0–1	35–0–1
q		<0.001	<0.001	<0.001	<0.001	<0.001
Bagging						
$\dot{r}$			0.998	0.920	0.908	0.906
s			14–0–22	17–0–19	18–0–18	19–0–17
q			0.243	0.868	>0.999	0.868
AdaBoost						
$\dot{r}$				0.922	0.910	0.908
s				31–1–4	25–1–10	27–0–9
q				<0.001	0.017	0.004
MultiBoost						
$\dot{r}$					0.986	0.985
s					18–1–17	12–1–23
q					>0.999	0.090
RotForest						
$\dot{r}$						0.999
s						17–0–19
q						0.868

Table 8
Comparison of prediction error for RotBoost with different combinations of S and T

$S \times T$	RB50 × 2	RB25 × 4	RB10 × 10	RB4 × 25	RB2 × 50
Mean over all sets (%)	17.48	17.48	17.66	17.97	18.58
RB50 × 2					
$\dot{r}$		0.989	0.998	1.018	1.083
s		17–0–19	12–0–24	9–0–27	5–2–29
p		0.868	0.065	0.004	<0.001
RB25 × 4					
$\dot{r}$			1.009	1.029	1.095
s			15–0–21	8–0–28	4–0–32
p			0.405	0.001	<0.001
RB10 × 10					
$\dot{r}$				1.020	1.085
s				10–1–25	2–0–34
p				0.017	<0.001
RB4 × 25					
$\dot{r}$					1.064
s					4–0–32
p					<0.001

folds in the triplet. For convenience of the notations, we will use the abbreviations $T_1^i = f_2^i \bigcup f_3^i$, $T_2^i = f_1^i \bigcup f_3^i$ and $T_3^i = f_1^i \bigcup f_2^i$. Using this approach, the central tendency y^c for a data point $\boldsymbol{x}$ is estimated by

$$y^c = \arg\max_{y \in \Phi} \left(\sum_{i=1}^{20} \sum_{j=1}^{3} I(\boldsymbol{x} \in f_j^i \ \& \ Y_H^{T_j^i} = y) \right),$$

where $I(\cdot)$ denotes an indicator function.

Once the cross-validation trials have been completed, the relevant measures can be estimated directly from the observed distribution of results. The use of cross-validation in this way has the advantage that every instance in the available data $\mathscr{D}$ is used the same number of times, both for training and for testing. The aim to repeat the trial for multiple times is to more accurately estimate the average performance of an algorithm from the training sets of the specified size that are drawn from the available data $\mathscr{D}$.

According to the above procedure, the previously defined bias and variance decompositions $\text{Bias}^2_{\text{BK}}/\text{Var}_{\text{BK}}$ and $\text{Bias}^2_{\text{KW}}/\text{Var}_{\text{KW}}$ of the errors that correspond to each classification method were estimated for each instance in every data set listed in Table 1, and then their values were averaged on that data set. Detailed decompositions of mean error into $\text{Bias}^2_{\text{BK}}$ and Var_{BK} as well as $\text{Bias}^2_{\text{KW}}$ and Var_{KW} are provided in the Appendix and the following several tables give the summary of these results. These tables have the same format as that of Table 3. Tables 4 and 5, respectively, summarize the relative performance of the algorithms with respect to mean $\text{Bias}^2_{\text{BK}}$ and Var_{BK} contributions to mean error over all the considered data sets.

As can be seen from Table 4, the order of these algorithms ranked in terms of mean $\text{Bias}^2_{\text{BK}}$ value from best to worst is RotBoost, SingleTree, RotForest, MultiBoost, AdaBoost and Bagging. However, the geometric mean of $\text{Bias}^2_{\text{BK}}$ ratios and the win–tie–loss statistic show that RotForest, MultiBoost and AdaBoost all achieve lower $\text{Bias}^2_{\text{BK}}$ value than SingleTree. Even though RotForest and MultiBoost have the same $\text{Bias}^2_{\text{BK}}$ values, the geometric mean of $\text{Bias}^2_{\text{BK}}$ ratios favors MultiBoost over RotForest whereas the win–tie–loss record favors RotForest over MultiBoost. Consider

Table 9
$\text{Bias}^2_{\text{BK}}$ and Var_{BK} decomposition of error for each algorithm on each data set

Data set	$\text{Bias}^2_{\text{BK}}$						Var_{BK}					
	Single	Bag	AdaB	MulB	RotF	RotB	Single	Bag	AdaB	MulB	RotF	RotB
Abalone	23.66	29.22	28.11	28.44	28.73	27.03	18.13	7.04	8.99	7.79	6.25	8.61
Australian	10.45	11.00	10.88	10.52	11.20	10.75	6.52	2.09	2.78	2.70	2.46	2.72
Balance	10.30	7.66	13.26	9.85	5.35	5.80	11.42	7.37	8.37	8.18	4.27	5.46
Bcs	23.11	23.44	23.16	23.67	22.48	21.68	10.66	5.61	9.06	7.13	7.95	7.52
Bcw	3.05	3.01	2.50	2.60	2.49	2.54	2.49	0.44	0.66	0.53	0.40	0.67
Car	2.75	3.01	3.31	3.21	2.71	3.27	2.60	0.91	0.70	0.70	0.92	0.67
Cleveland	23.72	31.09	31.58	32.05	30.91	31.14	24.24	13.00	13.97	12.42	13.47	12.66
Cmc	29.69	35.62	35.66	35.50	33.83	35.13	19.53	10.52	12.86	11.71	13.14	12.55
Creditr	10.15	11.14	9.84	11.16	10.70	10.35	6.64	2.24	2.86	2.10	2.48	2.31
Dermatology	3.45	2.46	1.80	1.98	1.68	1.80	1.91	1.15	1.34	1.03	0.95	1.01
Ecoli	11.74	12.63	11.61	10.97	11.43	11.77	8.08	4.60	4.09	4.02	4.94	3.60
German	21.48	22.68	18.69	19.21	24.07	19.12	14.01	5.56	6.69	5.65	4.40	5.44
Glass	23.34	23.48	23.15	23.18	23.39	23.11	1.14	1.05	1.40	1.19	1.00	1.36
Haberman	22.47	24.53	26.93	26.05	24.17	26.44	10.46	5.23	6.83	6.01	7.16	4.93
Heart	17.81	17.39	16.63	15.70	17.72	14.24	13.44	5.89	5.28	5.09	3.37	4.43
Hepatitis	29.44	25.19	28.44	28.44	27.00	23.50	11.81	8.44	8.19	7.25	6.81	9.25
House	15.50	16.84	16.97	16.83	17.77	16.34	11.65	5.06	4.59	4.35	3.57	4.72
Ionosphere	7.58	6.89	5.24	5.24	4.69	4.00	4.79	1.57	1.05	1.03	0.84	1.57
Iris	3.50	3.53	4.37	4.70	3.50	3.73	2.13	1.17	0.97	1.10	0.87	0.80
Liver	20.25	22.90	23.30	21.17	21.13	22.36	15.61	6.90	7.29	7.87	8.55	7.99
Lymphography	13.92	14.46	7.47	7.77	10.91	9.09	18.99	19.46	21.69	16.15	23.41	16.96
Pendigits	1.55	1.30	0.46	0.52	0.50	0.41	3.20	0.69	0.19	0.19	0.15	0.21
Pima	17.49	20.28	20.17	20.25	19.97	19.21	11.91	3.76	5.57	4.53	5.34	5.40
Segmentation	2.10	2.13	1.10	1.14	1.28	1.13	2.77	1.11	0.56	0.65	0.75	0.70
Sonar	14.42	16.75	11.92	11.73	10.91	10.12	15.67	6.32	6.06	6.90	6.35	7.12
Soybean	5.79	5.39	4.53	4.52	4.59	4.48	5.56	2.99	2.06	2.02	2.40	2.02
Spect	13.11	14.53	15.45	16.03	15.41	15.36	7.13	2.55	3.00	2.64	2.30	2.62
Spectf	15.71	15.97	15.86	14.80	15.74	15.52	10.11	3.57	3.57	3.92	3.29	4.14
Vehicle	16.70	19.52	17.48	17.65	16.86	15.60	13.98	6.26	6.21	5.99	4.96	6.32
Votes	4.02	4.14	4.06	4.03	3.39	3.91	1.38	0.43	1.02	0.77	0.83	0.83
Vowelc	8.11	8.05	2.32	3.67	2.74	1.81	19.30	5.57	3.93	3.62	3.74	3.52
Wdbc	3.30	3.37	2.34	2.19	3.30	2.27	4.27	1.47	1.20	1.26	1.63	0.96
Wine	3.40	1.97	1.88	2.05	2.75	2.05	5.98	2.58	1.43	1.18	2.36	1.01
Wpbc	18.66	21.57	20.95	21.24	21.62	22.60	16.68	4.36	5.41	4.59	6.29	2.89
Yeast	25.44	31.14	30.35	29.95	30.21	29.40	22.24	9.11	10.92	10.09	10.41	10.38
Zoo	6.88	6.68	4.11	2.97	6.04	2.28	5.50	4.06	2.33	4.31	5.40	3.22

the relative performance of RotBoost in terms of $\mathrm{Bias}^2_{\mathrm{BK}}$, all the metrics demonstrate that it outperforms the other classification methods. At the significance level 0.05, RotBoost achieves lower $\mathrm{Bias}^2_{\mathrm{BK}}$ value for significantly more data sets than Bagging, AdaBoost and MultiBoost, while only the advantage over Bagging and AdaBoost is significant if the Bonferroni correction is taken into consideration.

From Table 5, we can observe that the considered algorithms are rated from best to worst in terms of mean $\mathrm{Var}_{\mathrm{BK}}$ as RotBoost, MultiBoost, Bagging, RotForest, AdaBoost and SingleTree where MultiBoost and RotBoost achieve the same values of mean $\mathrm{Var}_{\mathrm{BK}}$. However, the geometric mean of $\mathrm{Var}_{\mathrm{BK}}$ ratios favors RotBoost over MultiBoost whereas the win–tie–loss metric reverses the order of them. At the same time, we find that in terms of the win–tie–loss statistic of $\mathrm{Var}_{\mathrm{BK}}$ value, RotBoost performs significantly better than SingleTree and AdaBoost at the significance level 0.05. Based on the results reported in Tables 4 and 5, RotBoost is seen to simultaneously reduce the bias and variance terms of a single tree and the decrement achieved by it is much greater than that done by the other ensemble methods, which leads it to behave best among the compared classification methods. In fact, RotForest also decreases both the bias and variance terms, however, the reduction achieved by it is smaller than that done by RotBoost.

Tables 6 and 7, respectively, summarize the relative performance of the considered algorithms with respect to mean $\mathrm{Bias}^2_{\mathrm{KW}}$ and $\mathrm{Var}_{\mathrm{KW}}$ over the used data sets. In terms of mean $\mathrm{bias}^2_{\mathrm{KW}}$ value, these algorithms are ordered from best to worst as RotBoost, MultiBoost, AdaBoost, RotForest, SingleTree and Bagging. Nevertheless, both the geometric mean of $\mathrm{Bias}^2_{\mathrm{KW}}$ ratios and the win–tie–loss statistic favor AdaBoost over MultiBoost. Meanwhile, RotBoost is seen to have smaller $\mathrm{Bias}^2_{\mathrm{KW}}$ on much more data sets than all the other algorithms with the advantage over only MultiBoost is not significant when the significance level is set to be 0.05. Furthermore, if we take into account all the used metrics with respect to Bias^2 (that is, $\mathrm{Bias}^2_{\mathrm{BK}}$ or $\mathrm{Bias}^2_{\mathrm{KW}}$) to assess the performance of the compared classification algorithms, the order of them from best to worst will be RotBoost, AdaBoost, MultiBoost, RotForest, SingleTree and Bagging, which is basically consistent with the conclusion drawn from the results reported in Table 4.

As can be seen from Table 7, RotBoost has lower mean $\mathrm{Var}_{\mathrm{KW}}$ value than all the other algorithms except for MultiBoost. When comparing RotBoost with MultiBoost, the geometric mean of $\mathrm{Var}_{\mathrm{KW}}$ ratios shows that the former behaves slightly better than the latter while the win–tie–loss measure supports the reverse case. At the significance level 0.05, RotBoost is observed to significantly outperform SingleTree and AdaBoost based on the win–tie–loss metric of $\mathrm{Var}_{\mathrm{KW}}$ value. By comparing these conclusions with those obtained from Table 5, we can find that they are essentially the same, that is, RotBoost reduces much more variance term of the error of a single tree than all the other considered ensemble methods except for MultiBoost.

Furthermore, one may find that there is slight difference between the estimates of the bias and variance terms produced by the definition of Bauer and Kohavi (1999) and those calculated according to the definition of Kohavi and Wolpert (1996). One of the reasons may be that in a real-world learning task, the irreducible error is aggregated into both bias and variance terms in the error decomposition method proposed by Bauer and Kohavi

Table 10
$\mathrm{Bias}^2_{\mathrm{KW}}$ and $\mathrm{Var}_{\mathrm{KW}}$ decomposition of error for each algorithm on each data set

Data set	$\mathrm{Bias}^2_{\mathrm{KW}}$						$\mathrm{Var}_{\mathrm{KW}}$					
	Single	Bag	AdaB	MulB	RotF	RotB	Single	Bag	AdaB	MulB	RotF	RotB
Abalone	23.94	28.11	27.14	27.21	27.92	26.02	17.85	8.15	9.96	9.02	7.06	9.61
Australian	9.89	10.52	10.28	10.21	10.65	10.38	7.08	2.57	3.38	3.00	3.00	3.08
Balance	12.10	9.38	13.85	11.08	6.18	6.10	9.62	5.65	7.78	6.95	3.44	5.15
Bcs	21.70	22.64	22.21	22.37	22.08	21.03	12.07	6.42	10.01	8.44	8.36	8.17
Bcw	3.00	2.69	2.41	2.45	2.41	2.48	2.53	0.76	0.75	0.68	0.48	0.73
Car	2.96	2.97	2.96	2.89	2.64	2.92	2.39	0.94	1.05	1.02	0.99	1.02
Cleveland	28.43	32.49	32.68	33.38	32.62	32.62	19.53	11.60	12.87	11.10	11.76	11.19
Cmc	30.08	34.62	34.81	34.83	33.86	34.75	19.15	11.52	13.71	12.37	13.11	12.94
Creditr	9.94	10.61	9.27	10.43	10.36	9.77	6.85	2.77	3.43	2.82	2.83	2.89
Dermatology	3.41	2.50	1.95	1.96	1.77	1.83	1.96	1.10	1.19	1.06	0.86	0.98
Ecoli	12.55	12.74	11.44	11.11	11.55	11.38	7.28	4.49	4.26	3.87	4.82	3.99
German	21.66	21.94	18.02	18.41	23.36	18.02	14.33	6.31	7.36	6.46	5.10	6.54
Glass	23.21	23.36	23.35	23.05	23.34	23.06	1.28	1.17	1.21	1.32	1.05	1.40
Haberman	21.70	23.35	24.78	24.67	22.34	24.31	11.23	6.40	8.98	7.39	8.98	7.07
Heart	17.24	16.64	15.99	15.03	16.90	13.43	14.02	6.64	5.92	5.77	4.02	5.24
Hepatitis	26.44	25.48	25.97	26.35	23.79	22.95	14.81	8.14	10.66	9.34	10.01	9.80
House	15.48	16.57	16.05	16.05	16.81	15.34	11.68	5.53	5.51	5.12	4.52	5.72
Ionosphere	7.27	6.41	5.08	5.17	4.26	3.89	5.10	2.06	1.21	1.10	1.27	1.68
Iris	3.85	3.57	3.94	4.53	3.31	3.77	1.78	1.13	1.40	1.27	1.05	0.76
Liver	19.77	21.04	21.30	19.83	20.28	20.52	16.09	8.76	9.29	9.21	9.40	9.83
Lymphography	15.69	16.93	12.20	10.61	15.26	11.73	17.22	16.99	16.96	13.31	19.06	14.31
Pendigits	1.99	1.30	0.46	0.52	0.51	0.42	2.76	0.69	0.19	0.19	0.15	0.20
Pima	17.11	19.21	19.16	19.23	18.92	18.36	12.28	4.83	6.58	5.55	6.39	6.26
Segmentation	2.41	2.15	1.10	1.14	1.37	1.19	2.47	1.10	0.56	0.65	0.67	0.64
Sonar	14.64	15.56	11.12	11.96	10.77	9.84	15.45	7.52	6.86	6.67	6.49	7.40
Soybean	6.20	5.40	4.38	4.48	4.58	4.28	5.15	2.98	2.21	2.07	2.41	2.23
Spect	14.13	14.26	14.55	15.13	14.65	14.34	6.11	2.82	3.90	3.54	3.06	3.63
Spectf	14.28	15.16	14.76	14.24	14.90	14.42	11.54	4.38	4.66	4.47	4.14	5.24
Vehicle	16.36	18.17	16.12	16.29	15.69	14.45	14.32	7.61	7.56	7.35	6.13	7.47
Votes	3.70	3.74	3.90	3.83	3.42	3.65	1.71	0.82	1.19	0.98	0.80	1.09
Vowelc	11.20	7.59	2.85	3.57	3.02	2.12	16.21	6.03	3.40	3.71	3.46	3.21
Wdbc	3.67	3.36	2.35	2.26	3.08	2.30	3.90	1.48	1.19	1.18	1.85	0.93
Wine	4.17	2.41	2.00	2.11	2.93	2.14	5.21	2.14	1.32	1.12	2.18	0.92
Wpbc	19.19	21.23	19.90	20.53	21.70	21.69	16.14	4.69	6.47	5.30	6.22	3.80
Yeast	28.14	31.33	29.96	30.37	30.50	29.83	19.54	8.92	11.32	9.67	10.12	9.95
Zoo	7.05	6.60	3.60	3.76	6.66	2.93	5.33	4.15	2.84	3.51	4.78	2.57

Table 11
Means and standard deviations of test error computed by RotBoost with different combinations of the values for S and T

Data set	RB50 × 2	RB25 × 4	RB10 × 10	RB4 × 25	RB2 × 50
Abalone	35.51 ± 0.43	35.47 ± 0.43	35.63 ± 0.48	36.55 ± 0.54	37.79 ± 0.47
Australian	13.44 ± 0.68	13.33 ± 0.76	13.46 ± 0.79	14.11 ± 0.81	14.19 ± 0.77
Balance	10.10 ± 0.97	10.30 ± 0.75	11.26 ± 1.37	12.64 ± 1.83	15.01 ± 1.63
Bcs	28.50 ± 1.71	28.88 ± 2.19	29.19 ± 1.60	29.70 ± 1.49	29.90 ± 2.05
Bcw	3.03 ± 0.37	2.97 ± 0.34	3.21 ± 0.37	3.04 ± 0.36	3.45 ± 0.48
Car	3.69 ± 0.28	4.00 ± 0.33	3.94 ± 0.37	4.09 ± 0.37	4.77 ± 0.38
Cleveland	43.25 ± 1.53	43.40 ± 1.38	43.80 ± 1.28	42.66 ± 1.48	43.21 ± 1.46
Cmc	46.59 ± 0.87	46.80 ± 0.59	47.68 ± 0.98	48.25 ± 0.97	48.33 ± 1.02
Creditr	12.62 ± 0.62	12.86 ± 0.61	12.66 ± 0.75	13.02 ± 0.80	13.74 ± 0.55
Dermatology	2.14 ± 0.51	2.71 ± 0.54	2.80 ± 0.61	2.78 ± 0.71	3.32 ± 0.80
Ecoli	14.90 ± 1.41	14.36 ± 0.99	15.37 ± 1.02	15.12 ± 1.27	15.95 ± 1.34
German	24.79 ± 0.78	24.49 ± 0.56	24.56 ± 0.85	25.27 ± 1.09	26.03 ± 0.80
Glass	24.25 ± 0.98	24.32 ± 0.75	24.46 ± 0.87	24.46 ± 0.70	24.67 ± 0.75
Haberman	30.56 ± 2.03	30.61 ± 1.44	31.37 ± 2.13	31.96 ± 2.19	31.45 ± 2.17
Heart	18.26 ± 1.67	18.78 ± 1.32	18.67 ± 1.45	18.98 ± 1.05	20.31 ± 1.87
Hepatitis	33.38 ± 4.55	32.87 ± 4.85	32.75 ± 4.81	33.88 ± 4.17	33.38 ± 4.85
House	21.39 ± 1.21	21.01 ± 1.27	21.07 ± 1.36	21.70 ± 1.13	22.76 ± 0.81
Ionosphere	5.78 ± 0.71	5.68 ± 0.74	5.57 ± 0.76	5.90 ± 0.65	6.50 ± 0.90
Iris	4.13 ± 0.74	4.17 ± 0.65	4.53 ± 0.91	4.63 ± 0.90	5.07 ± 1.21
Liver	29.19 ± 2.11	28.76 ± 1.95	30.35 ± 1.81	31.82 ± 2.14	33.39 ± 1.56
Lymphography	23.75 ± 11.83	25.20 ± 12.14	26.05 ± 13.44	25.07 ± 12.12	26.45 ± 12.82
Pendigits	0.75 ± 0.05	0.65 ± 0.05	0.62 ± 0.06	0.64 ± 0.05	0.75 ± 0.06
Pima	24.29 ± 1.13	24.38 ± 1.07	24.62 ± 1.19	25.68 ± 0.96	26.31 ± 0.77
Segmentation	2.04 ± 0.19	1.88 ± 0.21	1.83 ± 0.19	1.89 ± 0.20	2.09 ± 0.24
Sonar	19.71 ± 1.98	18.49 ± 1.59	17.23 ± 2.16	17.62 ± 2.28	18.00 ± 2.43
Soybean	6.36 ± 0.53	6.55 ± 0.62	6.50 ± 0.78	6.16 ± 0.49	6.98 ± 0.58
Spect	17.94 ± 1.59	17.73 ± 1.25	17.98 ± 1.17	19.06 ± 1.42	20.86 ± 2.05
Spectf	18.74 ± 1.20	19.05 ± 1.36	19.66 ± 1.57	19.81 ± 1.94	20.00 ± 1.42
Vehicle	21.97 ± 0.72	22.09 ± 1.12	21.92 ± 1.09	21.83 ± 1.08	22.90 ± 1.12
Votes	4.27 ± 0.43	4.70 ± 0.52	4.74 ± 0.49	4.76 ± 0.66	5.11 ± 0.64
Vowelc	6.32 ± 0.76	5.47 ± 0.61	5.33 ± 0.71	5.58 ± 0.80	6.75 ± 0.79
Wdbc	3.66 ± 0.45	3.43 ± 0.55	3.22 ± 0.37	3.14 ± 0.56	3.29 ± 0.56
Wine	4.07 ± 1.37	2.84 ± 0.88	3.06 ± 0.67	2.87 ± 0.85	3.43 ± 1.25
Wpbc	24.74 ± 0.96	25.67 ± 1.55	25.49 ± 1.47	25.05 ± 1.57	24.33 ± 1.33
Yeast	39.36 ± 0.68	39.38 ± 0.72	39.78 ± 1.05	40.40 ± 0.66	41.70 ± 0.93
Zoo	5.99 ± 3.09	5.69 ± 2.27	5.49 ± 1.83	6.78 ± 3.83	6.53 ± 2.56

(1999) whereas that is included only into the bias term in the error decomposition method developed by Kohavi and Wolpert (1996).

3.3. *Optimal combination of S and T*

In the previous experiments conducted to compare the performance of RotBoost with that of several other classification algorithms, the number of iterations done for RotForest and AdaBoost (that is, the values of the parameters S and T) within RotBoost were both taken to be 10. In view of lacking theoretical grounds for emphasizing the effect of either RotForest or AdaBoost, it was decided that the iterations done for them should be set to be equal so as to balance the two. It is interesting, however, to consider other possible trade-offs between these two factors. The larger the value of S is, the more RotForest is performed. The larger the value of T is, the greater the influence of AdaBoost is emphasized. In order to investigate the effect of these two factors, some more combinations of the values for S and T (namely, RB50 × 2, RB25 × 4, RB10 × 10, RB4 × 25, RB2 × 50) are chosen to conduct experiments in this subsection. Table 8 presents the comparison of prediction error for each considered combination and the detailed results are postponed to the Appendix.

It is observed from Table 8 that, RB50 × 2 and RB25 × 4 achieve the same mean errors across all the data sets and the results of the geometric mean of error ratios as well as those of the win–tie–loss metric show that neither one of RB50 × 2 and RB25 × 4 can beat the other one. At the same time, RB50 × 2 and RB25 × 4 are seen to achieve much better performance than the other several combinations. Furthermore, we can find that RB2 × 50 perform much worse than the other combinations in terms of all the used statistics and the difference between it with each of the other case is significant no matter whether the Bonferroni correction is taken into account or not. Meanwhile, RB4 × 25 is also observed to be significantly worse than RB50 × 2, RB25 × 4 and RB10 × 10 at the significance level 0.05 except that the advantage of RB10 × 10 over it is not statistically significant if the Bonferroni correction is taken into consideration. As far as RB10 × 10 is concerned, the results reported in Table 8 indicate that it behaves a little worse than RB50 × 2 and RB25 × 4 but this inferiority is not significant from the statistical viewpoint. Therefore, we can draw a conclusion that RB50 × 2 and RB25 × 4 enjoy a general advantage over the other cases with respect to prediction error and either of them can be a preferential choice when using RotBoost to solve a practical classification task.

4. Conclusions and future work

This paper presents a novel ensemble classifier generation method, RotBoost, which is a combination of Rotation Forest and AdaBoost. When constructing ensemble classifiers using the classification tree algorithm proposed by Breiman et al. (1984) as the base learning algorithm, RotBoost is demonstrated to produce ensemble classifiers with significantly lower error than either Rotation Forest or AdaBoost more often than the reverse across the used 36 UCI data sets. At the same time, RotBoost is also found to perform much better than Bagging and MultiBoost. The improvement of prediction accuracy achieved by RotBoost is obtained with negligible increase in computational costs. In fact, RotBoost offers a potential computational advantage over AdaBoost in that it is amenable to parallel execution. Each sub-ensemble classifier formed by AdaBoost can be learned independently of the other ones. When employing the bias and variance decompositions of error (Bauer and Kohavi, 1999; Kohavi and Wolpert, 1996) to analyze the performance of each considered classification procedure, RotBoost is found to simultaneously reduce the bias and variance terms of a single classification tree and the decrement achieved by RotBoost is much more substantial than that done by the other compared ensemble methods (that is, Bagging, AdaBoost, RotForest and MultiBoost), which leads it to perform best among the considered classification approaches. Furthermore, the experiments conducted to study the effect of different numbers of iterations done for Rotation Forest and AdaBoost to the performance of RotBoost show that Rotation Forest can be conducted more times than AdaBoost in order to obtain a more accurate ensemble classifier.

Nevertheless, there are still some interesting problems deserved to be investigated further, which include but are not limited to the following items.

- Whether can RotBoost be improved further by introducing more modalities of perturbation as proposed by Zhou and Yu (2005) which is effective in maintaining high diversity as well as high accuracy?
- Evaluation of the performance of RotBoost by adopting other algorithms such as a neural network as the base learning algorithm.
- How will the proposed method RotBoost perform if it is extended to solve regression problems?
- How to automatically select the optimal combination of the parameters S and T?

Acknowledgements

This work was supported in part by the National Natural Science Foundations of China (No. 10531030 and 60675013) and National Basic Research Program of China (973 Program, No. 2007CB311002). The authors would like to express their devout thanks to the associate editor and two anonymous reviewers for their valuable comments and suggestions which lead to a substantial improvement of the paper.

Appendix

Table 9 presents the values of mean $\text{Bias}^2_{\text{BK}}$ and Var_{BK} computed for each algorithm on every used data set and Table 10 reports the corresponding results of $\text{Bias}^2_{\text{KW}}$ and Var_{KW}. Table 11 summarizes the mean errors computed with various combinations of the values for parameters S and T included into RotBoost. One reason to list tables of values rather than graphical representations of the relative performance here is to allow more precise comparisons of the considered classification methods. It is worthwhile to point out that these values are expressed in percentage and in some situations the reported values of bias and variance ($\text{Bias}^2_{\text{BK}}$ and Var_{BK}, $\text{Bias}^2_{\text{KW}}$ and Var_{KW}) may not sum precisely to those of error listed in Table 2 due to rounding of the values.

References

Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P., 2007. A comparison of decision tree ensemble creation techniques. IEEE Trans. Pattern Anal. Machine Intell. 29 (1), 173–180.

Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learn. 36 (1–2), 105–139.

Blake, C.L., Merz, C.J., 1998. UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Breiman, L., 1996. Bagging predictors. Machine Learn. 24 (2), 123–140.

Breiman, L., 1998. Arcing classifiers. Ann. Statist. 26 (3), 801–849.

Breiman, L., 2001. Random forests. Machine Learn. 45 (1), 5–32.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Chapman & Hall, New Work.

Dietterich, T.G., 1997. Machine-learning research: Four current directions. AI. Mag. 18 (4), 97–136.

Dietterich, T.G., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. Machine Learn. 40 (2), 139–157.

Freund, Y., Schapire, R.E., 1996. Experiments with a new boosting algorithm. In: Proc. 13th Internat. Conf. on Machine Learn.. Morgan Kaufmann, Bari, Italy, pp. 148–156.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. System Sci. 55 (1), 119–139.

Friedman, J.H., 1997. On bias, variance, 0/1-loss, and the curse-of-dimensionality. Data Min. Knowl. Disc. 1 (1), 55–77.

Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: A statistical view of boosting. Ann. Statist. 28 (2), 337–407.

Geman, S., Bienenstock, E., Doursat, R., 1992. Neural networks and the bias/variance dilemma. Neural Comput. 4 (1), 1–58.

Hansen, L., Salamon, P., 1990. Neural networks ensembles. IEEE Trans. Pattern Anal. Machine Intell. 12 (10), 993–1001.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Machine Intell. 20 (8), 832–844.

James, G.M., 2003. Variance and bias for general loss functions. Machine Learn. 51 (2), 115–135.

Jin, R., Zhang, J., 2007. Multi-class learning by smoothed boosting. Machine Learn. 67 (3), 207–227.

Kohavi, R., Wolpert, D., 1996. Bias plus variance decomposition for zero-one loss functions. In: Proc. 13th Internat. Conf. on Machine Learn.. Morgan Kaufmann, Bari, Italy, pp. 275–283.

Kong, E.B., Dietterich, T.G., 1995. Error-correcting output coding corrects bias and variance. In: Proc. 12th Internat. Conf. on Machine Learn.. Morgan Kaufmann, San Francisco, pp. 313–321.

Krogh, A., Vedelsby, J., 1995. Neural network ensembles, cross validation, and active learning. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (Eds.), Adv. Neural Inform. Proc. Sys, vol. 7. MIT Press, Cambridge MA, pp. 231–238.

Latinne, P., Debeir, O., Decaestecker, C., 2002. Combining different methods and number of weak decision trees. Pattern Anal. Appl. 5 (2), 201–209.

Leblanc, M., Tibshirani, R., 1996. Combining estimates in regression and classification. J. Amer. Statist. Assoc. 91 (436), 1641–1650.

Lim, T.S., Loh, W.Y., Shin, Y.S., 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learn. 40 (3), 203–229.

Meir, R., Rätsch, G., 2003. An introduction to boosting and leveraging. In: Advanced Lectures on Machine Learning. Lect. Notes Comput. Sci., vol. 2600. Springer-Verlag, Berlin, pp. 118–183.

Nadeau, C., Bengio, Y., 2003. Inference for the generalization error. Machine Learn. 52 (3), 239–281.

Optiz, D., Maclin, R., 1999. Popular ensemble methods: An empirical study. J. Artif. Intell. Res. 11, 169–198.

Optiz, D.W., Shavlik, J.W., 1996. Generating accurate and diverse members of a neural-network ensemble. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.M. (Eds.), Adv. Neural Inform. Proc. Sys, vol. 8. MIT Press, Cambridge MA, pp. 535–541.

Rodríguez, J.J., Alonso, C.J., Prieto, O.J., 2005. Bias and variance of rotation-based ensembles. In: Cabestany, J., Prieto, A., Sandoval, D.F. (Eds.), IWANN, Lect. Notes Comput. Sci., vol. 3512. Springer-Verlag, Berlin, pp. 779–786.

Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J., 2006. Rotation forest: A new classifier ensemble method. IEEE Trans. Pattern Anal. Machine Intell. 28 (10), 1619–1630.

Skurichina, M., Duin, R.P.W., 2002. Bagging, boosting and the random subspace method for linear classifiers. Pattern Anal. Appl. 5 (2), 121–135.

Valentini, G., Dietterich, T.G., 2004. Bias–variance analysis of support vector machines for the development of SVM-based ensemble methods. J. Machine Learn. Res. 5, 725–775.

Webb, G.I., 2000. MultiBoosting: A technique for combining boosting and wagging. Machine Learn. 40 (2), 159–196.

Webb, G.I., Conilione, P., 2006. Estimating bias and variance from data. <http://www.csse.monash.edu.au/~webb/Files/WebbConilione06.pdf>.

Zhou, Z.H., Wu, J.X., Tang, W., 2002. Ensembling neural networks: Many could be better than all. Artif. Intell. 137 (1–2), 239–263.

Zhou, Z.H., Yu, Y., 2005. Ensembling local learners through multimodal perturbation. IEEE Trans. Syst. Man Cybernet. B 35 (4), 725–735.