Hybrid weighted random forests for classifying very high-dimensional data

Baoxun Xu[1], Joshua Zhexue Huang[2], Graham Williams[2] and Yunming Ye[1]

[1]*Department of Computer Science, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, China*
[2]*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China*
Email: amusing002@gmail.com

Random forests are a popular classification method based on an ensemble of a single type of decision trees from subspaces of data. In the literature, there are many different types of decision tree algorithms, including C4.5, CART, and CHAID. Each type of decision tree algorithm may capture different information and structure. This paper proposes a hybrid weighted random forest algorithm, simultaneously using a feature weighting method and a hybrid forest method to classify very high dimensional data. The hybrid weighted random forest algorithm can effectively reduce subspace size and improve classification performance without increasing the error bound. We conduct a series of experiments on eight high dimensional datasets to compare our method with traditional random forest methods and other classification methods. The results show that our method consistently outperforms these traditional methods.

Keywords: Random Forests; Hybrid Weighted Random Forest; Classification; Decision tree;

1. INTRODUCTION

Random forests [1, 2] are a popular classification method which builds an ensemble of a single type of decision trees from different random subspaces of data. The decision trees are often either built using C4.5 [3] or CART [4], but only one type within a single random forest. In recent years, random forests have attracted increasing attention due to (1) its competitive performance compared with other classification methods, especially for high-dimensional data, (2) algorithmic intuitiveness and simplicity, and (3) its most important capability - "ensemble" using bagging [5] and stochastic discrimination [2].

Several methods have been proposed to grow random forests from subspaces of data [1, 2, 6, 7, 8, 9, 10]. In these methods, the most popular forest construction procedure was proposed by Breiman [1] to first use bagging to generate training data subsets for building individual trees. A subspace of features is then randomly selected at each node to grow branches of a decision tree. The trees are then combined as an ensemble into a forest. As an ensemble learner, the performance of a random forest is highly dependent on two factors: the performance of each tree and the diversity of the trees in the forests [11]. Breiman formulated the overall performance of a set of trees as the average strength and proved that the generalization error of a random forest is bounded by the ratio of the average correlation between trees divided by the square of the average strength of the trees.

For very high dimensional data, such as text data, there are usually a large portion of features that are uninformative to the classes. During this forest building process, informative features would have the large chance to be missed, if we randomly select a small subspace (Breiman suggested selecting $\lfloor log_2(M) + 1 \rfloor$ features in a subspace, where M is the number of independent features in the data) from high dimensional data [12]. As a result, weak trees are created from these subspaces, the average strength of those trees is reduced and the error bound of the random forest is enlarged. Therefore, when a large proportion of such "weak" trees are generated in a random forest, the forest has a large likelihood to make a wrong decision which mainly results from those "weak" trees' classification power.

To address this problem, we aim to optimize decision trees of a random forest by two strategies. One straightforward strategy is to enhance the classification performance of individual trees by a feature weighting method for subspace sampling [12, 13, 14]. In this method, feature weights are computed with respect to the correlations of features to the class feature and regarded as the probabilities of the feature to be selected in subspaces. This method obviously increases the classification performance of individual trees because the subspaces will be biased to contain more informative features. However, the chance of more correlated trees is also increased since the features with large weights are likely to be repeatedly selected.

The second strategy is more straightforward: use several different types of decision trees for each training data subset, to increase the diversity of the trees, and then select the optimal tree as the individual tree classifier in the random forest model. The work presented here extends the algorithm developed in [15]. Specifically, we build three different types of tree classifiers (C4.5, CART, and CHAID [16, 17]) for each training data subset. We then evaluate the performance of the three classifiers and select the best tree. In this way, we build a hybrid random forest which may include different types of decision trees in the ensemble. The added diversity of the decision trees can effectively improve the accuracy of each tree in the forest, and hence the classification performance of the ensemble. However, when we use this method to build the best random forest model for classifying high dimensional data, we can not be sure of what subspace size is best.

In this paper, we propose a hybrid weighted random forest algorithm by simultaneously using a new feature weighting method together with the hybrid random forest method to classify high dimensional data. In this new random forest algorithm, we calculate feature weights and use weighted sampling to randomly select features for subspaces at each node in building different types of trees classifiers (C4.5, CART, and CHAID) for each training data subset, and select the best tree as the individual tree in the final ensemble model.

Experiments were performed on 8 high dimensional text datasets with dimensions ranging from 2000 to 13195. We compared the performance of eight random forest methods and well-known classification methods: C4.5 random forest, CART random forest, CHAID random forest, hybrid random forest, C4.5 weighted random forest, CART weighted random forest, CHAID weighted random forest, hybrid weighted random forest, support vector machines [18], naive Bayes [19], and k-nearest neighbors [20]. The experimental results show that our hybrid weighted random forest achieves improved classification performance over the ten competitive methods.

The remainder of this paper is organized as follows. In Section 2, we introduce a framework for building a hybrid weighted random forest, and describe a new random forest algorithm. Section 3 summarizes four measures to evaluate random forest models. We present experimental results on 8 high dimensional text datasets in Section 4. Section 5 contains our conclusions.

2. HYBRID WEIGHTED RANDOM FORESTS

In this section, we first introduce a feature weighting method for subspace sampling. Then we present a general framework for building hybrid random forests. By integrating these two methods, we propose a novel hybrid weighted random forest algorithm.

TABLE 1. Contingency table of input feature A and class feature Y

	$Y = y_1$	$\dots$	$Y = y_j$	$\dots$	$Y = y_q$	Total
$A = a_1$	ϖ_{11}	$\dots$	ϖ_{1j}	$\dots$	ϖ_{1q}	$\varpi_{1\cdot}$
$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$A = a_i$	ϖ_{i1}	$\dots$	ϖ_{ij}	$\dots$	ϖ_{iq}	$\varpi_{i\cdot}$
$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$A = a_p$	ϖ_{p1}	$\dots$	ϖ_{pj}	$\dots$	ϖ_{pq}	$\varpi_{p\cdot}$
Total	$\varpi_{\cdot 1}$	$\dots$	$\varpi_{\cdot j}$	$\dots$	$\varpi_{\cdot q}$	ϖ

2.1. Notation

Let Y be the class (or target) feature with q distinct class labels y_j for $j = 1, \cdots, q$. For the purposes of our discussion we consider a single categorical feature A in dataset D with p distinct category values. We denote the distinct values by a_i for $i = 1, \cdots, p$. Numeric features can be discretized into p intervals with a supervised discretization method.

Assume D has *val* objects. The size of the subset of D satisfying the condition that $A = a_i$ and $Y = y_j$ is denoted by ϖ_{ij}. Considering all combinations of the categorical values of A and the labels of Y, we can obtain a *contingency table* [21] of A against Y as shown in Table 1. The far right column contains the *marginal totals for feature* A:

$$\varpi_{i\cdot} = \sum_{j=1}^{q} \varpi_{ij} \qquad \text{for } i = 1, \cdots, p \tag{1}$$

and the bottom row is the *marginal totals for class feature* Y:

$$\varpi_{\cdot j} = \sum_{i=1}^{p} \varpi_{ij} \qquad \text{for } j = 1, \cdots, q \tag{2}$$

The *grand total* (the total number of samples) is in the bottom right corner:

$$\varpi = \sum_{j=1}^{q} \sum_{i=1}^{p} \varpi_{ij} \tag{3}$$

Given a training dataset D and feature A we first compute the contingency table. The feature weights are then computed using the two methods to be discussed in the following subsection.

2.2. Feature Weighting Method

In this subsection, we give the details of the feature weighting method for subspace sampling in random forests. Consider an M-dimensional feature space $\{A_1, A_2, \dots, A_M\}$. We present how to compute the weights $\{w_1, w_2, \ldots, w_M\}$ for every feature in the space. These weights are then used in the improved algorithm to grow each decision tree in the random forest.

2.2.1. *Feature Weight Computation*

The weight of feature A represents the correlation between the values of feature A and the values of the class feature Y. A larger weight will indicate that the class labels of objects in the training dataset are more correlated with the values of feature A, indicating that A is more informative to the class of objects. Thus it is suggested that A has a stronger power in predicting the classes of new objects.

In the following, we propose to use the *chi-square statistic* to compute feature weights because this method can quantify the correspondence between two categorical variables.

Given the contingency table of an input feature A and the class feature Y of dataset D, the *chi-square statistic* of the two features is computed as:

$$corr(A,Y) = \sum_{i=1}^{p}\sum_{j=1}^{q} \frac{(\varpi_{ij} - t_{ij})^2}{t_{ij}} \tag{4}$$

where ϖ_{ij} is the observed frequency from the contingency table and t_{ij} is the expected frequency computed as

$$t_{ij} = \frac{\varpi_{i\cdot} \times \varpi_{\cdot j}}{\varpi} \tag{5}$$

The larger the measure $corr(A,Y)$, the more informative the feature A is in predicting class Y.

2.2.2. *Normalized Feature Weight*

In practice, feature weights are normalized for feature subspace sampling. We use $corr(A,Y)$ to measure the informativeness of these features and consider them as feature weights. However, to treat the weights as probabilities of features, we normalize the measures to ensure the sum of the normalized feature weights is equal to 1. Let $corr(A_i,Y)$ $(1 \le i \le M)$ be the set of M feature measures. We compute the normalized weights as

$$w_i = \frac{\sqrt{corr(A_i,Y)}}{\sum_{i=1}^{N}\sqrt{corr(A_i,Y)}} \tag{6}$$

Here, we use the square root to smooth the values of the measures. w_i can be considered as the probability that feature A_i is randomly sampled in a subspace. The more informative a feature is, the larger the weight and the higher the probability of the feature being selected.

2.3. Framework for Building a Hybrid Random Forest

As an ensemble learner, the performance of a random forest is highly dependent on two factors: the diversity among the trees and the accuracy of each tree [11]. Diversity is commonly obtained by using bagging and random subspace sampling. We introduce a further element of diversity by using different types of trees.

Considering an analogy with forestry, the different data subsets from bagging represent the "soil structures." Different decision tree algorithms represent "different tree species". Our approach has two key aspects: one is to use three types of decision tree algorithms to generate three different tree classifiers for each training data subset; the other is to evaluate the accuracy of each tree as the measure of tree importance. In this paper, we use the out-of-bag accuracy to assess the importance of a tree.

Following Breiman [1], we use bagging to generate a series of training data subsets from which we build trees. For each tree, the data subset used to grow the tree is called the "in-of-bag" (IOB) data and the remaining data subset is called the "out-of-bag" (OOB) data. Since OOB data is not used for building trees we can use this data to objectively evaluate each tree's accuracy and importance. The OOB accuracy gives an unbiased estimate of the true accuracy of a model.

Given n instances in a training dataset D and a tree classifier $h_k(IOB_k)$ built from the k'th training data subset IOB_k, we define the OOB accuracy of the tree $h_k(IOB_k)$, for $di \in D$, as:

$$OOBAcc_k = \frac{\sum_{i=1}^{n} I(h_k(d_i) = y_i; d_i \in OOB_k)}{\sum_{i=1}^{n} I(d_i \in OOB_k)} \tag{7}$$

where I(.) is an indicator function. The larger the $OOBAcc_k$, the better the classification quality of a tree.

We use the out-of-bag data subset OOB_i to calculate the out-of-bag accuracies of the three types of trees (C4.5, CART and CHAID) with evaluation values E_1, E_2 and E_3 respectively.

Fig. 1 illustrates the procedure for building a hybrid random forest model. Firstly, a series of IOB/OOB datasets are generated from the entire training dataset by bagging. Then, three types of tree classifiers (C4.5, CART and CHAID) are built using each IOB dataset. The corresponding OOB dataset is used to calculate the OOB accuracies of the three tree classifiers. Finally, we select the tree with the highest OOB accuracy as the final tree classifier, which is included in the hybrid random forest.

Building a hybrid random forest model in this way will increase the diversity among the trees. The classification performance of each individual tree classifier is also maximized.

2.4. Decision Tree Algorithms

The core of our approach is the diversity of decision tree algorithms in our random forest. Different decision tree algorithms grow structurally different trees from the same training data. Selecting a good decision tree algorithm to grow trees for a random forest is critical

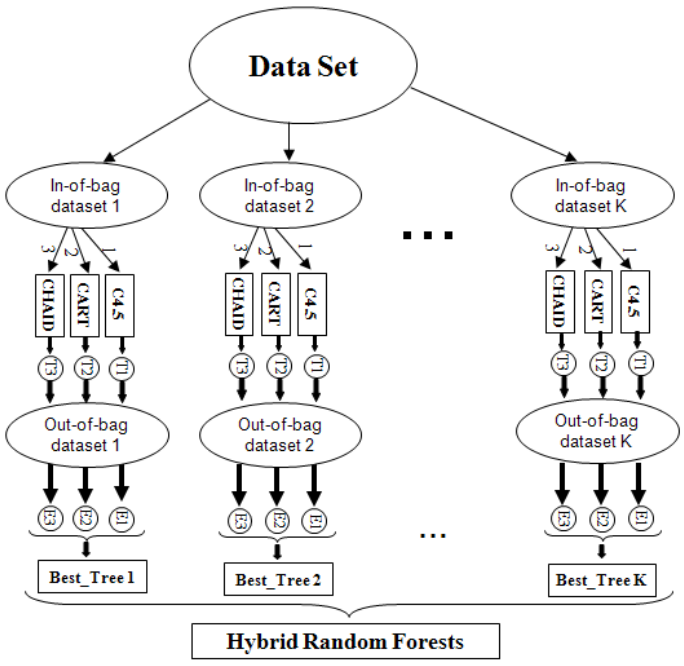

FIGURE 1. The Hybrid Random Forests framework.

for the performance of the random forest. Few studies have considered how different decision tree algorithms affect a random forest. We do so in this paper.

The common decision tree algorithms are as follows:

Classification Trees 4.5 (C4.5) is a supervised learning classification algorithm used to construct decision trees. Given a set of pre-classified objects, each described by a vector of attribute values, we construct a mapping from attribute values to classes. C4.5 uses a divide-and-conquer approach to grow decision trees. Beginning with the entire dataset, a tree is constructed by considering each predictor variable for dividing the dataset. The best predictor is chosen at each node using a impurity or diversity measure. The goal is to produce subsets of the data which are homogeneous with respect to the target variable. C4.5 selects the test that maximizes the information gain ratio (IGR) [3].

Classification and Regression Tree (CART) is a recursive partitioning method that can be used for both regression and classification. The main difference between C4.5 and CART is the test selection and evaluation process.

Chi-squared Automatic Interaction Detector (CHAID) method is based on the chi-square test of association. A CHAID decision tree is constructed by repeatedly splitting subsets of the space into two or more nodes. To determine the best split at any node, any allowable pair of categories of the predictor variables is merged until there is no statistically significant difference within the pair with respect to the target variable [16, 17].

From these decision tree algorithms, we can see that the difference lies in the way to split a node, such as the split functions and binary branches or multi-branches. In this work we use these different decision tree algorithms to build a hybrid random forest.

2.5. Hybrid Weighted Random Forest Algorithm

In this subsection we present a hybrid weighted random forest algorithm by simultaneously using the feature weights and a hybrid method to classify high dimensional data. The benefits of our algorithm has two aspects: Firstly, compared with hybrid forest method [15], we can use a small subspace size to create accurate random forest models. Secondly, compared with building a random forest using feature weighting [14], we can use several different types of decision trees for each training data subset to increase the diversities of trees. The added diversity of the decision trees can effectively improve the classification performance of the ensemble model. The detailed steps are introduced in **Algorithm 1**.

Input parameters to **Algorithm 1** include a training dataset D, the set of features A, the class feature Y, the number of trees in the random forest K and the size of subspaces m. The output is a random forest model $\mathscr{M}$. Lines 9–16 form the loop for building K decision trees. In the loop, Line 10 samples the training data D by sampling with replacement to generate an in-of-bag data subset IOB_i for building a decision tree. Line 11–14 build three types of tree classifiers (C4.5, CART, and CHAID). In this procedure, Line 12 calls the function $createTree_j()$ to build a tree classifier. Line 13 calculates the out-of-bag accuracy of the tree classifier. After this procedure, Line 15 selects the tree classifier with the maximum out-of-bag accuracy. K decision tree trees are thus generated to form a **hybrid weighted random forest** model $\mathscr{M}$.

Generically, function $createTree_j()$ first creates a new node. Then, it tests the stopping criteria to decide whether to return to the upper node or to split this node. If we choose to split this node, then the feature weighting method is used to randomly select m features as the subspace for node splitting. These features are used as candidates to generate the best split to partition the node. For each subset of the partition, $createTree_j()$ is called again to create a new node under the current node. If a leaf node is created, it returns to the parent node. This recursive process continues until a full tree is generated.

Algorithm 1 New Random Forest Algorithm

```
1: Input:
2: - $D$ : the training dataset,
3: - $A$ : the features space $\{A_1, A_2, ..., A_M\}$,
4: - $Y$ : the class features space $\{y_1, y_2, ..., y_q\}$,
5: - $K$ : the number of trees,
6: - $m$ : the size of subspaces.
7: Output: A random forest $\mathscr{M}$;
8: Method:
9: for $i = 1$ to $K$ do
10:    draw a bootstrap sample in-of-bag data subset $IOB_i$ and out-of-bag data subset $OOB_i$ from training dataset D;
11:    for $j = 1$ to 3 do
12:       $h_{i,j}(IOB_i) = createTree_j()$;
13:       use out-of-bag data subset $OOB_i$ to calculate the out-of-bag accuracy $OOBAcci, j$ of the tree classifier $h_{i,j}(IOB_i)$ by Equation(1);
14:    end for
15:    select $h_i(IOB_i)$ with the highest out-of-bag accuracy $OOBAcc_i$ as Optimal tree i;
16: end for
17: combine the $K$ tree classifiers $h_1(IOB_1), h_2(IOB_2), ..., h_K(IOB_K)$ into a random forest $\mathscr{M}$;
18:
19: Function createTree()
20: create a new node $\mathscr{N}$;
21: if stopping criteria is met then
22:    return $\mathscr{N}$ as a leaf node;
23: else
24:    for $j = 1$ to $M$ do
25:       compute the informativeness measure $corr(A_j, Y)$ by Equation (4);
26:    end for
27:    compute feature weights $\{w_1, w_2, ..., w_M\}$ by Equation (6);
28:    use the feature weighting method to randomly select $m$ features;
29:    use these $m$ features as candidates to generate the best split for the node to be partitioned;
30:    call createTree() for each split;
31: end if
32: return $\mathscr{N}$;
```

3. EVALUATION MEASURES

In this paper, we use five measures, i.e., strength, correlation, error bound c/s^2, test accuracy, and F1 metric, to evaluate our random forest models. Strength measures the collective performance of individual trees in a random forest and the correlation measures the diversity of the trees. The ratio of the correlation over the square of the strength c/s^2 indicates the generalization error bound of the random forest model. These three measures were introduced in [1]. The accuracy measures the performance of a random forest model on unseen test data. The F1 metric is a commonly used measure of classification performance.

3.1. Strength and Correlation Measures

We follow Breiman's method described in [1] to calculate the strength, correlation and the ratio c/s^2. Following Breiman's notation, we denote strength as s and correlation as $\bar{\rho}$. Let $h_k(IOB_k)$ be the kth tree classifier grown from the kth training data IOB_k sampled from D with replacement. Assume the random forest model contains K trees. The out-of-bag proportion of votes for $d_i \in D$ on class j is

$$Q(d_i, j) = \frac{\sum_{k=1}^{K} I(h_k(d_i) = j; d_i \notin IOB_k)}{\sum_{k=1}^{K} I(d_i \notin IOB_k)} \tag{8}$$

This is the number of trees in the random forest which are trained without d_i and classify d_i into class j, divided by the number of training datasets not containing d_i.

The strength s is computed as:

$$s = \frac{1}{n}\sum_{i=1}^{n}(Q(d_i, y_i) - max_{j \neq y_i} Q(d_i, j)) \tag{9}$$

where n is the number of objects in D and y_i indicates the true class of d_i.

The correlation $\bar{\rho}$ is computed as:

$$\bar{\rho} = \frac{\frac{1}{n}\sum_{i=1}^{n}(Q(d_i, y_i) - max_{j \neq y_i} Q(d_i, j))^2 - s^2}{(\frac{1}{K}\sum_{k=1}^{K}\sqrt{p_k + \bar{p}_k + (p_k - \bar{p}_k)^2})^2} \tag{10}$$

where

$$p_k = \frac{\sum_{i=1}^{n} I(h_k(d_i) = y_i; d_i \notin IOB_k)}{\sum_{i=1}^{n} I(d_i \notin IOB_k)} \tag{11}$$

and

$$\bar{p}_k = \frac{\sum_{i=1}^{n} I(h_k(d_i) = \hat{j}(d_i, Y); d_i \notin IOB_k)}{\sum_{i=1}^{n} I(d_i \notin IOB_k)} \tag{12}$$

where

$$\hat{j}(d_i, Y) = argmax_{j \neq y_i} Q(d, j) \tag{13}$$

is the class that obtains the maximal number of votes among all classes but the true class.

3.2. General Error Bound Measure c/s^2

Given the strength and correlation, the out-of-bag estimate of the c/s^2 measure can be computed.

An important theoretical result in Breiman's method is the upper bound of the generalization error of the random forest ensemble that is derived as

$$PE* \leq \bar{\rho}(1 - s^2)/s^2 \tag{14}$$

where $\bar{\rho}$ is the mean value of correlations between all pairs of individual classifiers and s is the strength of the set of individual classifiers that is estimated as the average accuracy of individual classifiers on D with out-of-bag evaluation. This inequality shows that the generalization error of a random forest is affected by the strength of individual classifiers and their mutual correlations. Therefore, Breiman defined the c/s^2 ratio to measure a random forest as

$$c/s^2 = \bar{\rho}/s^2 \tag{15}$$

The smaller the ratio, the better the performance of the random forest. As such, c/s^2 gives guidance for reducing the generalization error of random forests.

3.3. Test Accuracy

The test accuracy measures the classification performance of a random forest on the test data set. Let D_t be a test data and Y_t be the class labels. Given $d_i \in D_t$, the number of votes for d_i on class j is

$$N(d_i, j) = \sum_{k=1}^{K} I(h_k(d_i) = j) \tag{16}$$

The test accuracy is calculated as

$$Acc = \frac{1}{n}\sum_{i=1}^{n} I(N(d_i, y_i) - max_{j \neq y_i} N(d_i, j) > 0) \tag{17}$$

where n is the number of objects in D_t and y_i indicates the true class of d_i.

3.4. F1 Metric

To evaluate the performance of classification methods in dealing with an unbalanced class distribution, we use the F1 metric introduced by Yang and Liu [22]. This measure is equal to the harmonic mean of recall (α) and precision (β). The overall F1 score of the entire classification problem can be computed by a micro-average and a macro-average.

Micro-averaged F1 is computed globally over all classes, and emphasizes the performance of a classifier on common classes. Define α and β as follows:

$$\alpha = \frac{\sum_{i=1}^{q} TP_i}{\sum_{i=1}^{q}(TP_i + FP_i)}, \ \beta = \frac{\sum_{i=1}^{q} TP_i}{\sum_{i=1}^{q}(TP_i + FN_i)} \tag{18}$$

where q is the number of classes. TP_i (True Positives) is the number of objects correctly predicted as class i, FP_i (False Positives) is the number of objects that are predicted to belong to class i but do not. The micro-averaged F1 is computed as:

$$MicroF1 = \frac{2\alpha\beta}{\alpha + \beta} \tag{19}$$

Macro-averaged F1 is first computed locally over each class, and then the average over all classes is taken. It emphasizes the performance of a classifier on rare categories. Define α and β as follows:

$$\alpha_i = \frac{TP_i}{(TP_i + FP_i)}, \ \beta_i = \frac{TP_i}{(TP_i + FN_i)} \tag{20}$$

$F1$ for each category i and the macro-averaged F1 are computed as:

$$F1_i = \frac{2\alpha_i\beta_i}{\alpha_i + \beta_i}, \ MacroF1 = \frac{\sum_{i=1}^{q} F1_i}{q} \tag{21}$$

The larger the MicroF1 and MacroF1 values are, the higher the classification performance of the classifier.

TABLE 2. Summary statistic of 8 high-dimensional datasets

Name	Features	Instances	Classes	% Minority
Fbis	2000	2463	17	1.54
Re0	2886	1504	13	0.73
Re1	3758	1657	25	0.6
Tr41	7454	878	10	1.03
Wap	8460	1560	20	0.32
Tr31	10,128	927	7	0.22
La2s	12,432	3075	6	8.07
La1s	13,195	3204	6	8.52

4. EXPERIMENTS

In this section, we present two experiments that demonstrate the effectiveness of the new random forest algorithm for classifying high dimensional data. High dimensional datasets with various sizes and characteristics were used in the experiments. The first experiment is designed to show how our proposed method can reduce the generalization error bound c/s^2, and improve test accuracy when the size of the selected subspace is not too large. The second experiment is used to demonstrate the classification performance of our proposed method in comparison to other classification methods, i.e. SVM, NB and KNN.

4.1. Datasets

In the experiments, we used eight real-world high dimensional datasets. These datasets were selected due to their diversities in the number of features, the number of instances, and the number of classes. Their dimensionalities vary from 2000 to 13,195. Instances vary from 878 to 3204 and the minority class rate varies from 0.22% to 8.52%. In each dataset, we randomly select 70% of instances as the training dataset, and the remaining data as the test dataset. Detailed information of the eight datasets is listed in Table 2.

The **Fbis**, **Re0**, **Re1**, **Tr41**, **Wap**, **Tr31**, **La2s** and **La1s** datasets are classical text classification benchmark datasets which were carefully selected and preprocessed by Han and Karypis [23]. Dataset **Fbis** was compiled from the Foreign Broadcast Information Service TREC-5 [24]. The datasets **Re0** and **Re1** were selected from the Reuters-21578 text categorization test collection Distribution 1.0 [25]. The datasets **Tr41** and **Tr31** were derived from TREC-5 [24], TREC-6 [24], and TREC-7 [24]. Dataset **Wap** is from the WebACE project (WAP) [26]. The datasets **La2s** and **La1s** were selected from the Los Angeles Times for TREC-5 [24]. The classes of these datasets were generated from the relevance judgment provided in these collections.

4.2. Performance Comparisons between Random Forest Methods

The purpose of this experiment was to evaluate the effect of the hybrid weighted random forest method (**H_W_RF**) on strength, correlation, c/s^2, and test accuracy. The eight high dimensional datasets were analyzed and results were compared with seven other random forest methods, i.e., C4.5 random forest (**C4.5_RF**), CART random forest (**CART_RF**), CHAID random forest (**CHAID_RF**), hybrid random forest (**H_RF**), C4.5 weighted random forest (**C4.5_W_RF**), CART weighted random forest (**CART_W_RF**), CHAID weighted random forest (**CHAID_W_RF**). For each dataset, we ran each random forest algorithm against different sizes of the feature subspaces. Since the number of features in these datasets was very large, we started with a subspace of 10 features and increased the subspace by 5 more features each time. For a given subspace size, we built 100 trees for each random forest model. In order to obtain a stable result, we built 80 random forest models for each subspace size, each dataset and each algorithm, and computed the average values of the four measures of strength, correlation, c/s^2, and test accuracy as the final results for comparison. The performance of the eight random forest algorithms on the four measures for each of the 8 datasets is shown in Figs. 2, 3, 4, and 5.

Fig. 2 plots the strength for the eight methods against different subspace sizes on each of the 8 datasets. In the same subspace, the higher the strength, the better the result. From the curves, we can see that the new algorithm **(H_W_RF)** consistently performs better than the seven other random forest algorithms. The advantages are more obvious for small subspaces. The new algorithm quickly achieved higher strength as the subspace size increases. The seven other random forest algorithms require larger subspaces to achieve a higher strength. These results indicate that the hybrid weighted random forest algorithm enables random forest models to achieve a higher strength for small subspace sizes compared to the seven other random forest algorithms.

Fig. 3 plots the curves for the correlations for the eight random forest methods on the 8 datasets. For small subspace sizes, **H_RF, C4.5_RF, CART_RF, and CHAID_RF** produce higher correlations between the trees on all datasets. The correlation decreases as the subspace size increases. For the random forest models the lower the correlation between the trees then the better the final model. With our new random forest algorithm **(H_W_RF)** a low correlation level was achieved with very small subspaces in all 8 datasets. We also note that as the subspace size increased the correlation level increased as well. This is understandable because as the subspace size increases, the same informative features are more likely to be selected repeatedly in the subspaces, increasing the similarity of the decision trees. Therefore, the feature weighting method for subspace selection works well for small subspaces, at least from the point of view of the correlation measure.

Fig. 4 shows the error bound indicator c/s^2 for the eight methods on the 8 datasets. From these figures we can observe that as the subspace size increases, c/s^2 consistently reduces. The behaviour indicates that a subspace size larger than $\lfloor log_2(M)+1 \rfloor$ benefits all eight algorithms. However, the new algorithm **(H_W_RF)** achieved a lower level of c/s^2 at subspace size of $\lfloor log_2(M)+1 \rfloor$ than the seven other algorithms.

Fig. 5 plots the curves showing the accuracy of the eight random forest models on the test datasets from the 8 datasets. We can clearly see that the new random forest algorithm **(H_W_RF)** outperforms the seven other random forest algorithms in all eight data sets. It can be seen that the new method is more stable in classification performance than other methods. In all of these figures, it is observed that the highest test accuracy is often obtained with the default subspace size of $\lfloor log_2(M)+1 \rfloor$. This implies that in practice, large size subspaces are not necessary to grow high-quality trees for random forests.

4.3. Performance Comparisons with Other Classification Methods

We conducted a further experimental comparison against three other widely used text classification methods: support vector machines (SVM), Naive Bayes (NB), and k-nearest neighbor (KNN). The support vector machine used a linear Kernel with a regularization parameter of 0.03125, which was often used in text categorization. For Naive Bayes, we adopted the multi-variate Bernoulli event model that is frequently used in text classification [27]. For k-nearest neighbor (KNN), we set the number k of neighbors to 13. In the experiments, we used WEKA's implementation for these three text classification methods [28]. We used a single subspace size of features in all eight datasets to run the random forest algorithms. For **H_RF, C4.5_RF, CART_RF, and CHAID_RF**, we used a subspace size of 90 features in the first 6 datasets (i.e., Fbis, Re0, Re1, Tr41, Wap, and

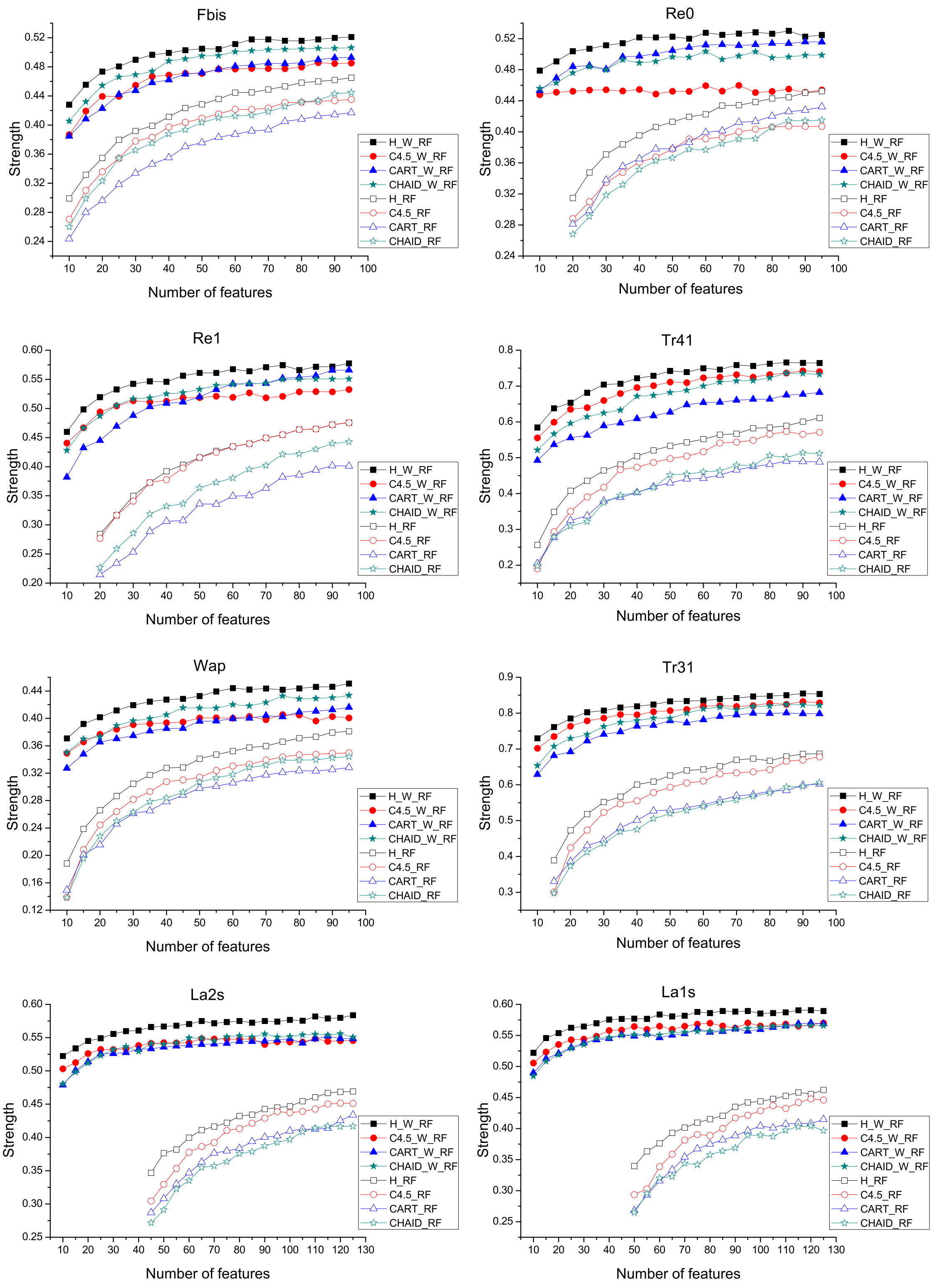

FIGURE 2. Strength changes against the number of features in the subspace on the 8 high dimensional datasets

Tr31) to run the random forest algorithms, and used a subspace size of 120 features in the last 2 datasets (La2s and La1s) to run these random forest algorithms. For **H_W_RF, C4.5_W_RF, CART_W_RF, and CHAID_W_RF**, we used Breiman's subspace size of $\lfloor log_2(M) + 1 \rfloor$ to run these random forest algorithms. This number of features provided a consistent result as shown in Fig. 5. In order to obtain stable results, we built 20 random forest models for each random forest algorithm and each dataset and present the average

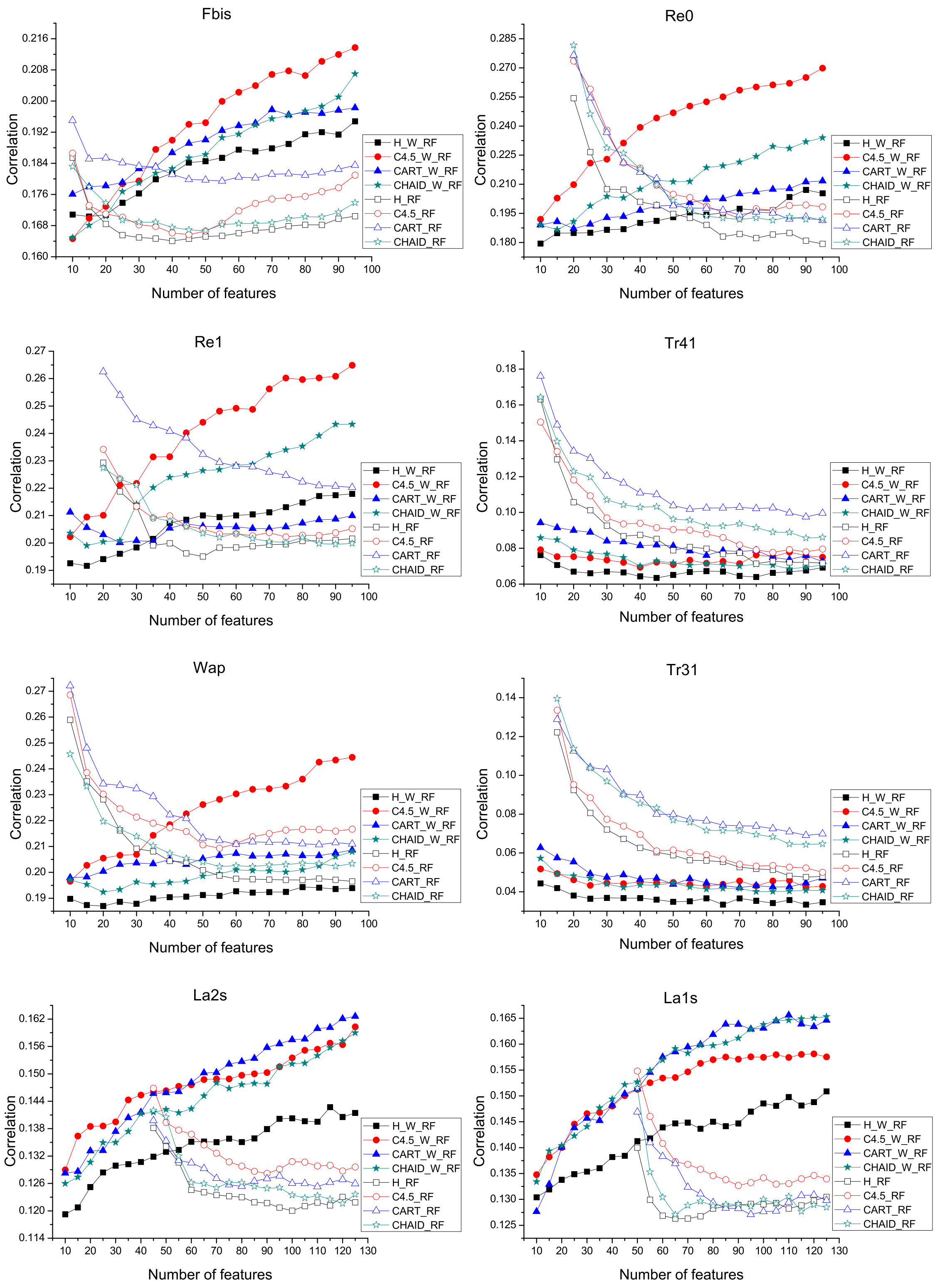

FIGURE 3. Correlation changes against the number of features in the subspace on the 8 high dimensional datasets

results, noting that the range of values are less than ±0.005 and the hybrid trees are always more accurate.

The comparison results of classification performance of eleven methods are shown in **Table 3**. The performance is estimated using test accuracy (**Acc**), Micro_F1 (**Mic**), and Macro_F1 (**Mac**). Boldface denotes best results between eleven classification methods. While the improvement is often quite small, there is always an improvement demonstrated. We observe that our proposed method (**H_W_RF**)

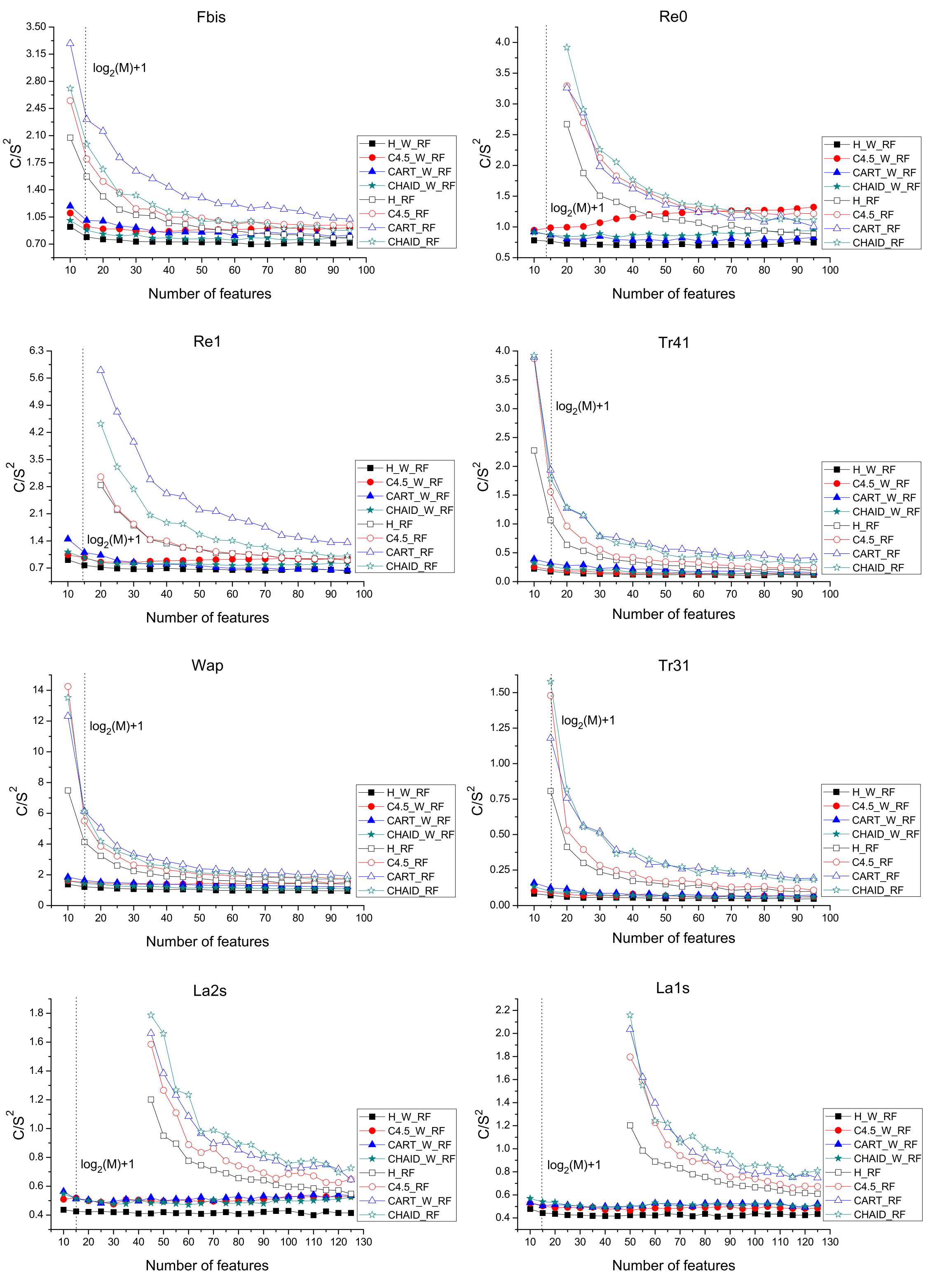

FIGURE 4. c/s^2 changes against the number of features in the subspace on the 8 high dimensional datasets

outperformed the other classification methods in all datasets.

5. CONCLUSIONS

In this paper, we presented a hybrid weighted random forest algorithm by simultaneously using a feature weighting method and a hybrid forest method to classify

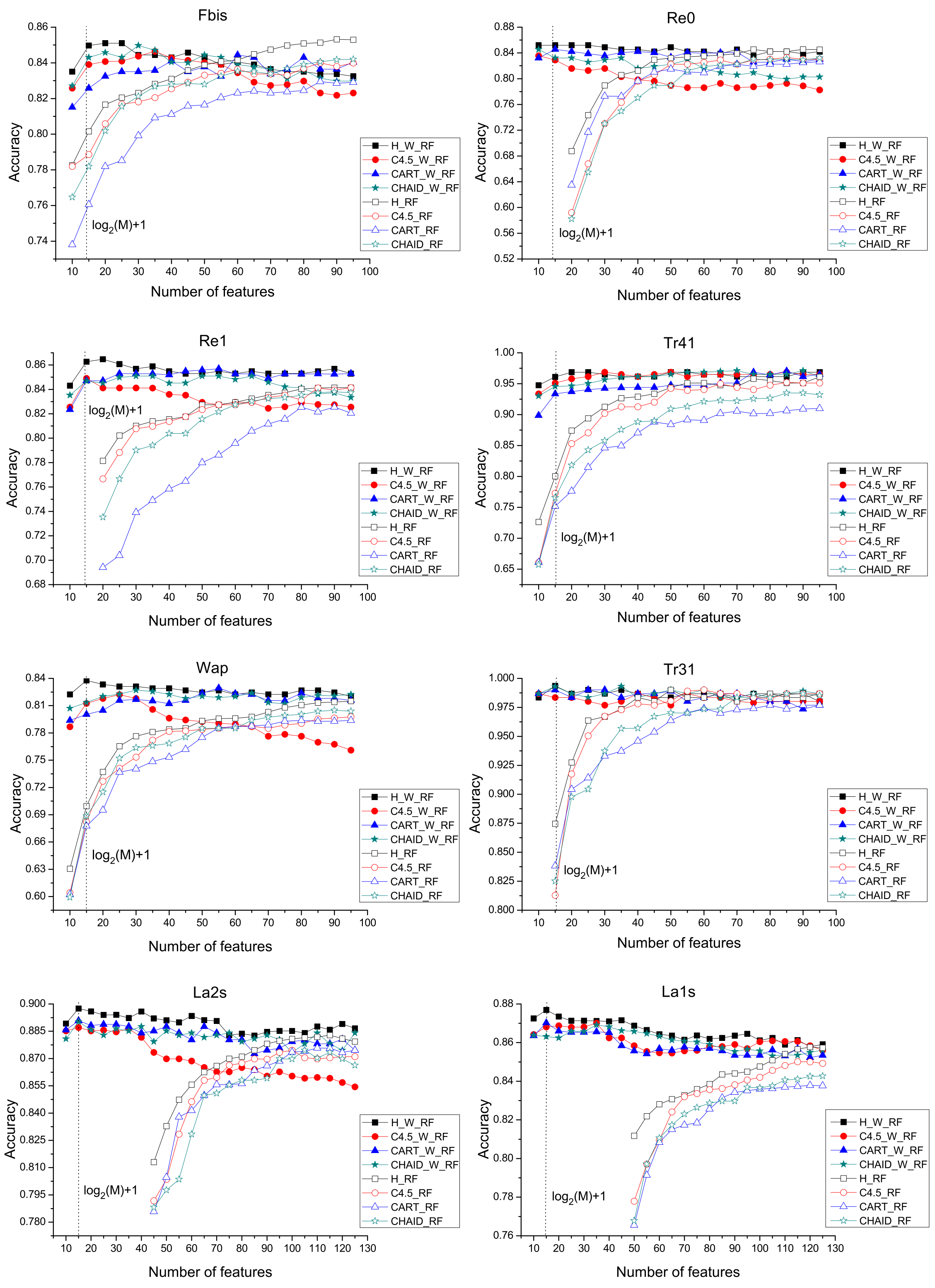

FIGURE 5. Test Accuracy changes against the number of features in the subspace on the 8 high dimensional datasets

high dimensional data. Our algorithm not only retains a small subspace size (Breiman's formula $\lfloor log_2(M)+1 \rfloor$ for determining the subspace size) to create accurate random forest models, but also effectively reduces the upper bound of the generalization error and improves classification performance. From the results of experiments on various high dimensional datasets, the random forest generated by our new method is superior to other classification methods. We can use the default $\lfloor log_2(M)+1 \rfloor$ subspace size and generally guarantee

TABLE 3. The comparison of results (best accuracy, Micro_F1, and Macro_F1 results) of the eleven methods on the 8 datasets

Dataset	Fbis			Re0			Re1			Tr41		
Measures	Acc	Mic	Mac	Acc	Mic	Mac	Acc	Mic	Mac	Acc	Mic	Mac
SVM	0.834	0.799	0.76	0.804	0.795	0.756	0.829	0.826	0.706	0.95	0.915	0.87
KNN	0.78	0.752	0.722	0.779	0.752	0.752	0.788	0.668	0.638	0.915	0.813	0.765
NB	0.776	0.74	0.706	0.784	0.741	0.619	0.816	0.732	0.58	0.935	0.856	0.782
H_RF	0.853	0.816	0.816	0.845	0.82	0.82	0.841	0.832	0.8	**0.953**	**0.926**	**0.895**
C4.5_RF	0.836	0.806	0.806	0.836	0.802	0.802	0.825	0.811	0.781	0.948	0.92	0.89
CART_RF	0.829	0.797	0.787	0.826	0.798	0.798	0.825	0.808	0.783	0.917	0.891	0.88
CHAID_RF	0.842	0.805	0.805	0.832	0.8	0.8	0.838	0.815	0.795	0.926	0.903	0.88
H_W_RF	**0.856**	**0.825**	**0.82**	**0.855**	**0.825**	**0.822**	**0.848**	**0.836**	**0.81**	**0.953**	**0.926**	**0.895**
C4.5_W_RF	0.841	0.809	0.815	0.845	0.815	0.812	0.838	0.826	0.795	0.95	0.922	0.892
CART_W_RF	0.835	0.805	0.81	0.839	0.81	0.805	0.835	0.818	0.79	0.935	0.91	0.88
CHAID_W_RF	0.839	0.815	0.812	0.842	0.812	0.815	0.84	0.83	0.8	0.942	0.915	0.88
Dataset	Wap			Tr31			la2s			la1s		
Measures	Acc	Mic	Mac	Acc	Mic	Mac	Acc	Mic	Mac	Acc	Mic	Mac
SVM	0.81	0.772	0.663	0.955	0.907	0.87	0.89	0.832	0.807	**0.875**	0.82	0.803
KNN	0.752	0.622	0.622	0.905	0.82	0.762	0.841	0.805	0.786	0.827	0.798	0.761
NB	0.797	0.742	0.559	0.925	0.832	0.81	**0.896**	0.815	0.79	0.87	0.802	0.775
H_RF	**0.815**	**0.805**	**0.735**	**0.965**	**0.925**	**0.88**	0.89	0.84	0.82	0.862	0.825	0.805
C4.5_RF	0.797	0.795	0.732	0.962	0.902	0.87	0.878	0.83	0.81	0.855	0.82	0.798
CART_RF	0.793	0.793	0.73	0.958	0.892	0.86	0.882	0.832	0.81	0.84	0.815	0.792
CHAID_RF	0.805	0.805	0.732	0.96	0.9	0.852	0.88	0.83	0.803	0.845	0.816	0.795
H_W_RF	**0.815**	**0.805**	**0.735**	**0.965**	**0.925**	**0.88**	**0.896**	**0.848**	**0.825**	**0.875**	**0.836**	**0.82**
C4.5_W_RF	0.805	0.795	0.732	0.962	0.911	0.87	0.886	0.835	0.816	0.866	0.825	0.81
CART_W_RF	0.8	0.792	0.73	0.96	0.902	0.865	0.887	0.835	0.812	0.87	0.825	0.81
CHAID_W_RF	0.811	0.795	0.73	0.96	0.905	0.855	0.887	0.833	0.81	0.865	0.825	0.805

to always produce the best models, on a variety of measures, by using the hybrid weighted random forest algorithm.

ACKNOWLEDGEMENTS

This research is supported in part by NSFC under Grant NO.61073195, and Shenzhen New Industry Development Fund under Grant NO.CXB201005250021A

REFERENCES

[1] Breiman, L. (2001) Random forests. *Machine learning*, **45**, 5–32.

[2] Ho, T. (1998) Random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 832–844.

[3] Quinlan, J. (1993) *C4.5: Programs for machine learning*. Morgan Kaufmann.

[4] Breiman, L. (1984) *Classification and regression trees*. Chapman & Hall/CRC.

[5] Breiman, L. (1996) Bagging predictors. *Machine learning*, **24**, 123–140.

[6] Ho, T. (1995) Random decision forests. *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 278–282. IEEE.

[7] Dietterich, T. (2000) An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, **40**, 139–157.

[8] Banfield, R., Hall, L., Bowyer, K., and Kegelmeyer, W. (2007) A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**, 173–180.

[9] Robnik-Šikonja, M. (2004) Improving random forests. *Proceedings of the 15th European Conference on Machine Learning*, pp. 359–370. Springer.

[10] Ho, T. (1998) C4.5 decision forests. *Proceedings of the Fourteenth International Conference on Pattern Recognition*, pp. 545–549. IEEE.

[11] Dietterrich, T. (1997) Machine learning research: four current direction. *Artificial Intelligence Magzine*, **18**, 97–136.

[12] Amaratunga, D., Cabrera, J., and Lee, Y. (2008) Enriched random forests. *Bioinformatics*, **24**, 2010–2014.

[13] Ye, Y., Li, H., Deng, X., and Huang, J. (2008) Feature weighting random forest for detection of hidden web search interfaces. *The Journal of Computational Linguistics and Chinese Language Processing*, **13**, 387–404.

[14] Xu, B., Huang, J., Williams, G., Wang, Q., and Ye, Y. (2012) Classifying very high-dimensional data with random forests built from small subspaces. *International Journal of Data Warehousing and Mining*, **8**, 45–62.

[15] Xu, B., Huang, J., Williams, G., Li, J., and Ye, Y. (2012) Hybrid random forests: Advantages of mixed trees in classifying text data. *Proceedings of the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer.

[16] Biggs, D., De Ville, B., and Suen, E. (1991) A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, **18**, 49–62.

[17] Ture, M., Kurt, I., Turhan Kurum, A., and Ozdamar, K. (2005) Comparing classification techniques for predicting essential hypertension. *Expert Systems with Applications*, **29**, 583–588.

[18] Begum, N., M.A., F., and Ren, F. (2009) Automatic text summarization using support vector machine. *International Journal of Innovative Computing, Information and Control*, **5**, 1987–1996.

[19] Chen, J., Huang, H., Tian, S., and Qu, Y. (2009) Feature selection for text classification with naive bayes. *Expert Systems with Applications*, **36**, 5432–5435.

[20] Tan, S. (2005) Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, **28**, 667–671.

[21] Pearson, K. (1904) *On the Theory of Contingency and Its Relation to Association and Normal Correlation.* Cambridge University Press.

[22] Yang, Y. and Liu, X. (1999) A re-examination of text categorization methods. *Proceedings of the 22th International Conference on Research and Development in Information Retrieval*, pp. 42–49. ACM.

[23] Han, E. and Karypis, G. (2000) Centroid-based document classification: Analysis and experimental results. *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 424–431. Springer.

[24] TREC. (2011) Text retrieval conference, http://trec.nist.gov.

[25] Lewis, D. (1999) Reuters-21578 text categorization test collection distribution 1.0, http://www.research.att.com/ lewis.

[26] Han, E., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., and Moore, J. (1998) Webace: A web agent for document categorization and exploration. *Proceedings of the 2nd International Conference on Autonomous Agents*, pp. 408–415. ACM.

[27] McCallum, A. and Nigam, K. (1998) A comparison of event models for naive bayes text classification. *AAAI-98 workshop on learning for text categorization*, pp. 41–48.

[28] Witten, I., Frank, E., and Hall, M. (2011) *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.