Canonical Forest

A Dissertation Presented

by

Yu-Chuan Chen

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Statistics)

Stony Brook University

January 2014

Stony Brook University

The Graduate School

Yu-Chuan Chen

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Dr. Hongshik Ahn – Dissertation Advisor
Professor, Department of Applied Mathematics and Statistics

Dr. Wei Zhu – Chairperson of Defense
Deputy Chair, Professor, Department of Applied Mathematics and Statistics

Dr. Song Wu – Member
Assistant Professor, Department of Applied Mathematics and Statistics

Dr. Yiyi Zhou – Outside Member
Assistant Professor, Department of Economics

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Canonical Forest

by

Yu-Chuan Chen

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Statistics)

Stony Brook University

2014

In this dissertation, we propose a new classification ensemble method named Canonical Forest. This new ensemble method uses canonical linear discriminant analysis (CLDA) and bootstrap resampling method to create more accurate and diverse classifiers in an ensemble. Although CLDA is commonly used for dimension reduction, we note here CLDA serves as a linear transformation tool rather than a dimension reduction tool. Since CLDA will find the transformed space that separates the classes farther in distribution, classifiers built on this space will be more accurate than those on the original space. To further diversify the classifiers in an ensemble, CLDA is applied only on a partial mutually exclusive feature space for each bootstrap sample. To compare the performance of

Canonical Forest and other widely used ensemble methods including Bagging, Adaboost, Samme, Random Forest, and Rotation Forest, we tested them on 29 real or artificial data sets. In addition to the classification accuracy, we also investigated the diversity and the bias and variance decomposition of each ensemble method. Because Canonical Forest cannot be applied to high-dimensional data directly, we propose another version of Canonical Forest called High-Dimensional Canonical Forest (HDCF) that is specifically designed for the high-dimensional data. By implementing the algorithm of Random Subspace into Canonical Forest, we can naturally apply Canonical Forest to high-dimensional data without performing feature selection or feature reduction first. We compared the performance of HDCF with some current popular high-dimensional classification algorithms including SVM, CERP, and Random Forest using gene imprinting, estrogen and leukemia data sets.

Table of Contents

List of Figures

List of Tables

Chapter 1

Background

Classification analysis is a procedure that builds a model or a rule using the features (predictor variables) in old data where each observation is associated with only one known class label and then uses this model or this rule to assign a class label to each observation in new data where observations are not associated with class labels. The features in the data could be either continuous or discrete. The learning algorithm in classification analysis is also known as supervised learning in data mining because it requires that known class labels are pre-assigned to all observations in training data set. A training data set is the data set which is used to build the model during classification analysis. The algorithm implemented in a classification analysis is called a classifier. Although several measures can be used to evaluate the performance of a classifier, classification accuracy is the most commonly used one. In general, a good classifier should yield a low misclassification rate. When evaluating the performance of a classification method, cross validation is a common method used in order to obtain a proper estimate. In k-fold cross validation, the data set is randomly divided into roughly equal-sized k mutually exclusive subsets. For each run, one subset serves as test data and the remaining $k-1$ subsets serve as training data. This process is repeated k times with each subset serving as test data only once and is included in training data $k-1$ times. Those k estimates from k repeated process are then combined to obtain the performance of a classifier. In the case where k is equal to n, which is the number of instances, it is called leave-one-out cross validation (LOOCV). In each run of LOOCV, only one instance serves as test data and the remaining $n-1$ instances serve as training data. LOOCV is often used in the situation where the sample size is small.

Classification analysis is widely used in many fields such as credit scoring, pattern recognition, DNA sequencing, medical diagnosis and also many other fields of science. For example, a bank could divide the existing customers into two possible groups: customers with “good credit” and customers with “bad credit” based on their credit history. These existing customers then serve as training data and the characteristics of these existing customers such as age, gender, occupation, income, etc. serve as features in classification analysis to generate a model or a rule to help the bank make a decision on the financial requests from new customers.

One of the earliest works on classification analysis is known as Linear Discriminant Analysis (LDA) that was first introduced in Fisher (1936). In this work, Fisher developed a linear discriminant model to classify the Iris flowers into three different species (Iris setosa, Iris virginica and Iris versicolor) based on their characteristics (length and width of the sepals and petals). After that, many different algorithms of classification analysis have been developed including logistic regression (Berkson, 1944), k-nearest neighbor (k-NN: Cover and Hart, 1967), neural networks (Rosenblatt, 1958), decision trees, naïve Bayes (Duda and Hart, 1973), support vector machines (SVM: Vapnik, 1995), and nearest shrunken centroid (Tibshirani et al., 2002).

LDA is a linear classifier that classifies instances by finding a reduced set of linear combination of original features that can better separate the distinct groups than using the original features. In the situation where the groups are not linearly separable, quadratic discriminant analysis (QDA) is often used. QDA is a more generalization of LDA, which allows the classifier to separate groups by quadratic boundaries.

Logistic regression is one kind of regression analysis that is used when the response variable is dichotomous (i.e. Y=0 or Y=1). It predicts the probability of the outcome of an event by fitting the logit model on the response variable in regression analysis. When the outcomes of a response variable are more than two, it can be extended to multinomial logistic regression by fitting the multinomial logit model if the response variable is nominal, or it can be extended to ordinal logistic regression by fitting the ordinal logit model if the response variable is ordinal. It was Berkson who first introduced the logit model in 1944. A brief description of the origins of logistic regression can be found in Cramer (2002).

k-NN is a very simple classification algorithm that classifies an instance only based on its k closest or nearest neighbors. That is, an instance is assigned to the class that is the majority class among its k nearest neighbors. Among many different distance measures, Euclidean distance is the most common distance measure used in a k-NN classifier. A small value of k often results in high variance and therefore is unstable. However, a large value of k often leads to high bias and therefore becomes less accurate. A proper value of k may be obtained by adopting cross-validation methods.

A neural network is composed of many interconnected neurons or nodes and is organized in layers. A neuron is usually connected by a set of neurons that are associated with different weights from one layer to another layer. Although the idea of neural networks was first introduced by McCulloch and Pitts (1943), it was Rosenblatt (1958) who first introduced the perceptron and brought the neural networks to the classification world.

Naïve Bayes is a probabilistic classifier that predicts the class label by estimating the posterior probability of each class conditioning on the given features. The instance is then assigned to the class with the highest posterior probability. A naïve Bayes classifier is based on the Bayes theorem and it requires the assumption that the features are independent given a certain class.

SVM is a binary classifier that finds an optimal hyperplane that can linearly separate two classes with a maximum margin. A maximum margin means that the distance from the nearest instance to this hyperplane is maximized on both sides. If such a linearly separating hyperplane cannot be found in the original input space, we can project the original input space into a higher dimensional space called kernel space where we can find an optimal linearly separating hyperplane. To classify on multi-class instances, SVM needs to reduce the multi-class classification to multiple two-class classifications.

Nearest shrunken centroid is a classification algorithm modified based on nearest centroid. The nearest centroid classifies an instance by assigning the class of its closet centroid in squared distance. The nearest shrunken centroid also adopts this algorithm but makes a modification to it by shrinking the class centroids toward the overall centroids using a threshold. By shrinking the class centroids, nearest shrunken centroid may reduce the effects of noisy variables.

The algorithm of nearest shrunken centroid was developed by Tibshirani et al. (2002).

Among all the various classification algorithms in classification analysis, decision trees are one of the current popular algorithms in classification analysis because they are simple to understand and interpret and they also perform well. There are many different versions of decision trees. Automatic Interaction Detection (AID) was the first decision tree method that was introduced by Morgan and Sonquist (1963). After that, many other different algorithms of decision trees have been proposed. Some popular algorithms of decision trees include CHAID (Chi-squared automatic interaction detection: Kass, 1980), CART (Classification and regression trees: Breiman et al., 1984), ID3 (Iterative Dichotomiser 3: Quinlan, 1986), C4.5 (Quinlan, 1993), and QUEST (Quick, unbiased and efficient statistical trees: Loh and Shih, 1997).

In general, the algorithm of a decision tree begins with a binary split from the root node where the value of one of the input features is used for the split. After a split is made, the same process is applied to each new node until the tree stops splitting and this repeated process is called recursive partitioning. There are various splitting rules developed for decision trees to find the split. Among them, information gain and gini index are the two most commonly used splitting rules in decision trees. The nodes with no further splits are called terminal nodes. Each observation is classified into only one terminal node that follows a unique path. That is, the observations in one terminal node should be mutually exclusive from the observations in other terminal nodes. Sometimes when the tree grows too large, the overfitting problem may occur. If a learning algorithm performs very well in classifying training data but gives poor results in predicting new data, it is said to overfit training data. For example, the misclassification rate on the training data is nearly zero but the misclassification rate on the new data is very large. When a tree is overfitting, some actions must be taken to downsize the tree to a proper size. These actions are called pruning. Minimal cost-complexity (Breiman et al., 1984) is one commonly used pruning method to avoid overfitting training data. First a tree is fully grown without any restrictions and then this tree is pruned back by deleting a pair of terminal nodes one at a time to minimize the cost-complexity risk. Cross validation method is often used for the trees to be pruned appropriately.

Recently in classification analysis the concept of combining multiple classifiers is widely used to further improve the performance of individual classifiers. The way of combining multiple classifiers to produce a new classifier is called a classification ensemble method. The combination of the predictions from multiple base classifiers is usually through majority voting or weighted voting to produce the final prediction. For example, suppose we have an ensemble classifier composed of three base classifiers classifying three instances with only two classes (i.e. Y = 0 or Y = 1). The predictions from these three classifiers are (0, 1, 1), (0, 0, 1), and (1, 1, 1), respectively. Then the final prediction from this ensemble classifier would be (0, 1, 1) if a majority voting was used. If a weighted voting was used instead, then the weight for each classifier must be taken into account and the result might be different. In fact, majority voting can be thought as a special case of weighted voting when the weight assigned to each base classifier is equal.

In classification ensemble methodologies, it is known that a more powerful classifier can be generated by combining many weak classifiers to improve the accuracy (Ji and Ma 1997; Hastie et al., 2001), where a weak classifier means that it performs slightly better than random guess. That is, the classification accuracy of a weak classifier is only required to be slightly greater than 0.5. For any ensemble classifier, let p be the classification accuracy of each base classifier and B be the number of base classifiers in this ensemble classifier. Here we let $B = 2k + 1$, where k is a nonnegative integer. If majority voting is used to combine base classifiers, then the classification accuracy of this ensemble classifier would be

$$A_B = \sum_{i=k+1}^{B} \binom{B}{i} p^i (1-p)^{B-i}$$

under the assumption that the decisions of the base classifiers are independent. Lam and Suen (1997) showed that A_B is strictly increasing if p is greater than 0.5 and strictly decreasing if p is smaller than 0.5. Ahn et al. (2007) also pointed out that the classification accuracy of an ensemble classifier will converge to 1 if p is greater than 0.5. Therefore, in order to be beneficial from using ensemble methodology, the base classifiers must perform better than random guess. If the base classifiers perform worse than random guess, then combining these classifiers will only lead to a poorer result.

The number of base classifiers in an ensemble classifier is called the ensemble size. In general, the steps to create an ensemble classifier are:

1. Generate multiple samples from the original training data set by using a resampling method
2. Build multiple classifiers from each of those resampled samples
3. A classification ensemble is built by combining these classifiers through majority voting or weighted voting

Usually, we would like the base classifiers in an ensemble classifier to be diverse in order to improve the performance. Diversity means that the base classifiers have errors on different parts. Usually when we form a committee to make a decision on something, we would prefer to include people who have different opinions. It will be meaningless to form a committee if the committee members are always in agreement. Likewise, the accuracy of a classification ensemble can only be further improved by having diverse base classifiers. For example, suppose we have two ensemble classifiers A and B, and each of them is composed of three base classifiers and four instances. The performance from the three base classifiers in A are (1, 0, 1, 0), (0, 1, 1, 0), and (1, 1, 0, 0), respectively while the performance from the three base classifiers in B are (1, 1, 0, 0), (1, 1, 0, 0), and (1, 1, 0, 0), respectively. Here 0 indicates that the decision is incorrect and 1 indicates that the decision is correct. Now if we take a majority voting on combining base classifiers for A and B, the performance from A and B are (1, 1, 1, 0) and (1, 1, 0, 0), respectively. It is not too hard to see that performance of ensemble A has been improved by having diverse classifiers and ensemble B has not gained anything from combining classifiers since the base classifiers in B are always in agreement. Since diversity plays an important role in the performance of an ensemble classifier, measures of diversity become important issues in ensemble classification analysis. Some good discussions on measures of diversity in an ensemble classifier can be found in Kuncheva and Whitaker (2003), Kuncheva (2004), and Brown and Kuncheva (2010). Since it requires diverse base classifiers for an ensemble classifier to have a better performance, decision trees are often used as base classifiers in ensemble methods due to the fact that they are sensitive to minor changes but still quite accurate.

Tukey (1977) seemed to be the first one who started using the ensemble methodology by combining two linear regression models (Bühlmann and Yu, 2003; Rokach, 2010). After that, many classification ensemble methods have been developed to improve the classification accuracy. Early work of ensemble algorithms on classification analysis including Boosting (Schapire, 1990; Freund and Schapire, 1996), Bagging (Breiman, 1996), and Random Subspaces (Ho, 1998) are now widely used due to their high classification accuracy.

Boosting changes the distribution of the training dataset of current classifier adaptively based on the previous classifiers' performance and then combines these classifiers through a weighted voting where the weight is obtained during training phase. Bagging generates different training data for each classifier using a bootstrap resampling method in order to increase the diversity of classifiers while still maintaining the accuracy of classifiers. The final class prediction of Bagging is obtained by taking a majority voting on these classifiers. Random Subspaces train each classifier using a randomly selected feature subset from the original feature space so that the diversity of classifiers can be further increased. Like Bagging, Random Subspaces combine the classifiers using a majority voting. Random Forest, a variant of Bagging, is developed by Breiman (2001) to further improve the diversity of Bagging and therefore improve the performance. Based on the algorithm of Bagging, Random Forest diversifies the classifiers more by using only a random feature subspace at each node when growing a classification tree. Basically, both Random Subspace and Random Forest utilize the concept of feature subspace to further increase the diversity of an ensemble classifier.

Recently two ensemble algorithms, Rotation Forest (Rodríguez et al., 2006) and CERP (Classification by Ensembles from Random Partitions: Ahn et al., 2007) have also received attention due to good performance. Rotation Forest applies Principal Component Analysis (PCA) to each bootstrap sample to further diversify the classifiers and then combines these classifiers through a majority voting. CERP partitions the input feature space into several mutually exclusive feature subspaces and each classifier is trained under each feature subspace. By randomly partitioning the feature space, CERP generates more diverse but less correlated classifiers. The classifiers in CERP are combined with a majority voting or an average voting.

In this dissertation, we propose a new classification ensemble method named Canonical Forest. By applying CLDA (canonical linear discriminant analysis) to each bootstrap sample, Canonical Forest generates accurate and diverse classifiers. In Canonical Forest, CLDA is served as a linear transformation tool instead of a dimension reduction tool. That is, we still keep all the features while applying CLDA. Since CLDA will find the transformed space that separates the distribution farther among classes, classifiers built on this space will be more accurate than those on the original space. To further diversify the classifiers, CLDA is applied to each partial mutually exclusive feature space instead of directly being applied to the whole feature space for each bootstrap sample.

We also propose another version of Canonical Forest called HDCF (High Dimensional Canonical Forest) that is specifically designed for the high dimensional data since Canonical Forest cannot be applied to high dimensional data directly. By implementing the algorithm of Random Subspace into Canonical Forest, we can naturally apply Canonical Forest to high dimensional data without performing feature selection or feature reduction first. Basically, HDCF is very similar to Canonical Forest except that it only uses a random subset of features from the original feature space when constructing each base classifier.

To investigate the performance of Canonical Forest, we compare it with some current popular ensemble methods including Boosting, Bagging, Random Forest, and Rotation Forest. Twenty nine real or artificial data sets are used to evaluate the performance of each ensemble method. Besides investigating the classification accuracy, we also investigate the bias and variance decomposition.

We also compare the performance of HDCF with some current popular high dimensional classification algorithms including SVM, CERP, and Random Forest using gene imprinting, estrogen and leukemia data sets. A brief introduction to these three data sets is included in Chapter 4.

Chapter 2

Review of existing ensemble methods

In this chapter, we will briefly introduce some current popular ensemble methods that were used to compare the performance of Canonical Forest and HDCF in this study.

2.1 Adaboost

Adaboost (Freund and Schapire 1996; 1997) is the most widely used boosting method and it is available as a package named Adaboost.M1 in R. Adaboost builds classifiers one at a time by changing the weight distribution of current classifier based on previous classifier's classification and then combines these classifiers through weighted voting. At the beginning, the weights are equally assigned to each instance in training set. Then in next iteration, the weights of incorrectly classified instances are increased and the weights of correctly classified instances are reduced. Therefore, the classifier is forced to focus more on those hard-to-classify instances during next iteration. The same process is repeated until the last iteration is done. After all the classifiers are built, a weighted voting method is used to combine all these classifiers. The weights of classifiers are obtained during the training phase. A decision stump (Iba and Langley, 1992) is often served as the base classifier in Adaboost. For a decision tree that only splits once is called a decision stump. That is, a decision stump utilizes only one feature to make a classification.

LogitBoost is a variant version of Adaboost that was developed by Friedman et al. (2000). The main difference between LogitBoost and Adaboost is that LogitBoost uses the logistic loss function while Adaboost uses the exponential loss function. L_2Boost (Friedman, 2001; Bühlmann and Yu, 2003) is another well-known variant version of Adaboost by adopting the L_2-loss function. Another major difference between L_2Boost and Adaboost is that L_2Boost does not assign more weights to the misclassified instances like Adaboost does. SparseL_2Boost (Bühlmann and Yu, 2006) is a modification of L_2Boost that is aimed to increase the sparsity and therefore shows an advantage when dealing with high-dimensional data with a lot of noise features. Recently, a new boosting algorithm that is specifically designed for the asymmetric mislabeled data named Asymmetric η-Boost was proposed by Hayashi (2012). Asymmetric η-Boost is a generalization algorithm of Adaboost that adopts the asymmetric η-loss function making the boosting algorithm more resistant to asymmetric mislabeled data.

When applied to two-class classification problems, AdaBoost has been proven to be successful in producing accurate classifiers. However, when classifying instances with more than two classes, Adaboost needs to reduce the multi-class classification to multiple two-class classification. To overcome this problem, a multi-class Adaboost algorithm, Samme (Stagewise Additive Modeling using a Multi-class Exponential loss function), has been proposed by Zhu et al. (2009). Samme is a new boosting algorithm that naturally extends Adaboost to a multi-class classification without reducing it to multiple two-class classification by slightly changing the weights on the misclassified instances. Samme has been proven to be quite successful when dealing with multi-class classification problems.

2.2 Bagging

Bagging (Bootstrap aggregating: Breiman, 1996) is one of the most well-known ensemble methods. It utilizes a resampling method called bootstrap at the training phase when building a classifier in order to build more diverse classifiers. Here bootstrap resampling method means that it randomly draws instances from data with replacement. For each base classifier, a bootstrap sample is drawn from the original training data set and then serves as the new training data set. Since each bootstrap sample is randomly drawn from the whole training data set with replacement, the distribution of each bootstrap sample is similar to the distribution of the original training data set. Therefore, these base classifiers trained with bootstrap samples are still maintaining the accuracies but now become more diverse. A simple majority voting is then used to combine these base classifiers to form the final classification. Although bagging classifiers usually can yield a better performance than a single classifier, it is not always the case. Breiman (1996) indicated that Bagging algorithm is most useful when the base classifiers are unstable. Here an unstable classifier means that the classifier tends to have a significantly different result while there is only a slight change on the training data. In Breiman's study (1996), decision trees are chosen to be the base classifiers in Bagging algorithm because decision trees are accurate but tend to have different results even there is only a small difference between training data.

Wagging (Weight aggregating: Bauer and Kohavi, 1999), a variant of Bagging, is an ensemble classifier that diversifies the base classifiers by adding random weights to the training instances instead of resampling from the training data. For each classifier, all training instances are assigned with equal weights and then Gaussian noise with mean zero and a specified standard deviation is added to each weight to change the weights of these training instances. In this way, the classifiers in Wagging are diversified since the weights of instances in training data vary from one classifier to another. Webb (2000) proposed a new boosting method, named Multiboost, by combining the algorithm of Adaboost and Wagging. When decision trees are served as the base classifiers, Multiboost has a lower error rate than any of Adaboost, Bagging or Wagging in Webb's study. It should be noted that continuous Poisson distribution, instead of Gaussian noise, is used to assign the weights of training instances in Multiboost.

Panov and Džeroski (2007) proposed to combine the algorithm of Bagging and Random Subspaces to create a more accurate and diverse ensemble classifier. Random Subspaces (Ho, 1998) is an algorithm that uses only a random subset of input features when training each base classifier to increase the diversity. The details about Random Subspace are provided in Chapter 3. By training each classifier with a bootstrap sample consisted of only a subspace of input features, this new algorithm proposed by Panov and Džeroski (2007) is more diverse than Bagging and Random Subspaces and hence has a better performance.

Double-Bagging (Hothorn and Lausen, 2003) is a CLDA based ensemble method that combines the algorithms of CLDA and Bagging. For each base classifier, Double-Bagging first takes a bootstrap sample from training data and then applies CLDA to the out of bag (OOB) sample to obtain the coefficient matrix. The details about OOB sample are introduced in Section 2.3. The canonical features can be computed using this coefficient matrix in the bootstrap sample. The base classifier is then built using both the original features and the canonical features. Finally, a simple majority voting is used to combine all these base classifiers.

2.3 Random Forest

Random Forest (Breiman 2001), a variant version of Bagging, is a decision tree based ensemble method. To make base classifiers more diverse, the algorithm of Random Subspace is implemented into the algorithm of Bagging at each node when building a tree in Random Forest. Like Bagging, Random Forest also uses a bootstrap sampling method at the training phase when building a tree. When applying bootstrap sampling method, there are about one-third of the instances not contained in the bootstrap sample and this “left-out” sample is called out-of-bag (OOB) sample. One of the advantages of Random Forest is that it does not require using cross-validation to evaluate its performance. The performance of Random Forest can be evaluated simply using the OOB sample. Another important feature that Random Forest has is feature importance ranking. To rank the importance of features in Random Forest, the values of each feature are randomly permuted and OOB accuracy is calculated for each feature after permutation. The importance score of a feature is then determined by the mean of the difference between the original OOB accuracy and the permuted OOB accuracy from all the trees. This measure of importance score is also called the measure of mean decrease in accuracy. Features with higher importance score are more important than features with lower importance score.

To be more diverse than Bagging, Random Forest uses only a random subset of the features instead of all features at each node when growing a tree. Denote p as the number of all the features. Only m features are selected at random from the p features and a split is made using only these m features at each node. Bagging can be thought as a special case of Random Forest when $m = p$. Different number of features used at each node when growing a tree can lead to different results. Square root of p is the default feature subset size of a node in the R Random Forest package named *randomForest*. Ahn et al. (2007) observed that this default yields consistently good results in many data sets. Because Random Forest only uses partial features at each node when growing a tree, it can be directly applied to high-dimensional data without performing feature selection first. It should be noted that when growing a tree, the tree needs to be grown to the largest size without any pruning method implemented in the algorithm of Random Forest. Due to the increase of diversity, Random Forest performs better than Bagging and therefore is often used in classification analysis.

2.4 Rotation Forest

Rotation Forest is a new classification ensemble algorithm developed by Rodríguez et al. (2006). To fit a classifier in an ensemble, it first randomly splits the training data of p features into K subsets. That is, each feature subset will contain roughly $m = p/K$ features. Next, for each feature subset, a class is randomly removed and then a sample containing 75% of the original sample is drawn from this data set using bootstrap sampling method. Then principal component analysis (PCA) is applied to this bootstrap sample using the features in the subset. PCA is a frequently used tool in data reduction. By applying PCA, one can reduce the input features into a smaller set of uncorrelated components where each component is a linear combination of the original features.

For a given input data set, we first find its covariance matrix and then calculate its eigenvalues and corresponding eigenvectors. The first component is then the eigenvector with the largest eigenvalue and accounts for most of the variation in input data. The second component is the eigenvector with the second largest eigenvalue and accounts for most of the remaining variation in input data. It should be noted that the second component is orthogonal to the first component. All other succeeding components follow this same rule and are orthogonal to all previous components. Usually, the first few components should account for most of the variation in original data. Therefore, we can just use these components instead of all the input variables without losing too much information. More about data reduction will be discussed in Chapter 3. When applying PCA in Rotation Forest, all the components will be kept to preserve all the information. After applying PCA, a rotation matrix is built using the coefficients obtained from PCA and then multiplied to the original training data set (an $N \times p$ matrix) to obtain a new training data set. After all the base classifiers are built, a majority voting is used to combine these base classifiers. Rotation Forest is quite successful due to its high accuracy and diversity.

RotBoost (Zhang and Zhang, 2008) is a recently proposed ensemble method that combines the algorithms of Rotation Forest and Adaboost to improve the performance. RotBoost first uses the algorithm of Rotation Forest to obtain the rotation matrix and then the new training data set is obtained using this rotation matrix. After the new training data set is formed, the algorithm of Adaboost is then used to build a classifier. Like Rotation Forest, Rotboost also utilizes a majority voting to form the final classification. By simultaneously reducing the bias and the variance, the performance of RotBoost showed superiority among the considered ensemble methods in Zhang and Zhang’s study.

2.5 CERP

CERP was recently developed by Ahn et al. (2007) and it is a high-dimensional data oriented classification ensemble algorithm. First the input feature space is randomly partitioned into several mutually exclusive feature subspaces with roughly equal sizes. Then under each subspace, a tree is grown with only the features in this subspace. By doing this, CERP avoids the problem of high dimensionality and therefore does not need a feature selection or dimension reduction before classifying. Ten-fold cross validation is used during the training phase to determine the number of subspaces which is also the ensemble size of CERP. Finally, a final classification is obtained by combining all these classifiers simply through a majority voting or average voting depending on its base classifiers. One of the two different base classifiers was used in Ahn et al.'s study: CART and logistic regression trees. CART-based CERP takes a majority voting to form the final classification and logistic regression tree-based CERP takes an average voting to form the final classification. Because all the feature subspaces are formed mutually exclusive, the correlations among the classifiers are substantially reduced in CERP. Hansen and Salamon (1990), Ho (1998) and Kuncheva et al. (2003) noted that the performance of an ensemble is further improved with less correlated base classifiers (Ahn et al., 2007). By partitioning the whole feature space into several mutually exclusive feature subspaces, CERP becomes an accurate and efficient ensemble classifier. Ahn et al. (2007) also proposed to generate multiple ensembles by re-partitioning the feature space and then combine these ensembles to further improve the accuracy of CERP. Eleven ensemble classifiers were used in their study.

LORENS (Logistic regression ensembles: Lim et al., 2010) is an ensemble method based on the algorithm of CERP that employs logistic regression as the base classifiers. Since logistic regression is a binary classifier, LORENS can be only applied to two-class classification problems. To overcome this problem, Lee et al. (2013) proposed an extended version of LORENS, named mLORENS (Multinomial logistic regression ensembles) that allows LORENS to be applied to multi-class problems by using multinomial logistic regression as its base classifiers. Both LORENS and mLORENS use the average voting method to combine the base classifiers.

Chapter 3

Method

3.1 Canonical Linear Discriminant Analysis

Dimension reduction technique is often used to reduce the number of input features, specifically in high-dimensional data. When the number of features of the input data is too large, typically much larger than the number of instances, it is called high-dimensional data. In classification analysis, it is well known that classification methods can perform poorly on high-dimensional data. This is mainly because high-dimensional data usually consist of too many redundant features which are also called noise features. When data contain too many noise features, models built from these data may consist of too much useless information and result in poor performance during classification phase. Therefore, many classification methods need to reduce the original input feature space into a smaller feature set when classifying high-dimensional data. The way of reducing original feature space into a smaller feature set is called dimension reduction. If the dimension reduction is successful, then this new feature set should still keep most of the information from the original input data. We then can train models using the new feature set instead of all the features. For example, if the input data are p-dimensional, it can be reduced to q-dimensional ($q < p$) without losing too much information by applying dimension reduction technique. PCA and CLDA (Canonical Linear Discriminant Analysis) are the two most widely used dimension reduction techniques.

The general idea of CLDA is to find a linear combination that maximizes the between-class variance relative to the within-class variance (Hastie et al. 2001). That is, it finds a linear transformation of features which separates the class distribution as far as possible. In this sense, classifier using the features extracted by CLDA performs better compared to the one using original features. Therefore, CLDA can always make a good separation among classes and hence is a better feature reduction method than PCA in classification analysis since PCA does not utilize the class information when performing feature reduction. The algorithm of CLDA modified from Hastie et al. (2001) is given in Figure 1.

Although CLDA is known as a dimension reduction tool, we do not use it as a dimension reduction technique in this experiment. Instead, we keep all the components when applying CLDA to preserve all the information and to further diversify the classifiers. Therefore, here CLDA only serves as a linear transformation tool to make the input data separate as far as possible among classes.

Given:

- X: the objects in the training data set (an N x p matrix)
- C: the number of classes
- p: the number of variables
- S_i: the covariance of class *i*

Procedure:

1. Compute the class centroid matrix $M_{C\times p}$, where the *i*-*j* entry is the mean of class *i* for variable *j*.
2. Compute the common covariance matrix W:
$$W = \sum_{i=1}^{C}(n_i - 1)S_i$$
3. Compute $M^* = MW^{-1/2}$ by using eigen-decomposition of *W*.
4. Obtain the between covariance matrix B^* by computing the covariance matrix of M^*.
5. Do the eigenvalue-decomposition of B^* such that $B^* = VDV^T$.
6. The columns v_i of *V* define the coordinates of the optimal subspaces.
7. Convert X to the coordinates in the new subspace:
$Z_i = v_i^T W^{-1/2} X$
8. Z_i is the i^{th} canonical coordinate.

Figure 1: Algorithm of CLDA.

3.2 Canonical Forest

Rotation Forest is a PCA based ensemble method proposed by Rodríguez et al. in 2006. It has been proven to be a quite successful ensemble method due to its high classification accuracy comparing to other current widely used ensemble methods. However, like we discussed in previous section, CLDA is a more suitable feature extraction technique than PCA when it comes to classification analysis due to the fact that PCA does not utilize class information while CLDA does. Therefore, we propose a new classification ensemble method called Canonical Forest by replacing PCA with CLDA in the algorithm of Rotation Forest to further improve the classification accuracy. Basically, the main difference between Canonical Forest and Rotation Forest is on the method for feature extraction. CLDA is used in Canonical Forest while PCA is used in Rotation Forest.

Let $x = [x_1, \dots, x_p]^T$ be an instance represented by p features and let X be the training data composed of n instances in a form of an $n \times p$ matrix. Let Y be the class labels $(1, \cdots, C)$ of the training data where $Y = [y_1, \dots, y_n]^T$. The classifiers in an ensemble are denoted by $L_1, \dots, L_B$ and the feature set is denoted by F. Like Rotation Forest, all the classifiers can be trained in parallel. To set up the training data for classifier L_i, we use the following steps.

1. Randomly split F into K subsets. The subsets are made disjoint to increase the diversity of an ensemble. For simplicity, assume that p is divisible by K. Then each feature subset contains $m = p/K$ features. If p is not divisible by K, we let $m = [p/K] + 1$ and the last subset is set to contain the remaining features.
2. Denote $F_{i,j}$ as the j-th subset of features of the training data for classifier L_i. For each of such subsets, draw a bootstrap sample with 75 percent of the original sample size.
3. Run CLDA on $F_{i,j}$ and obtain a coefficient matrix $A_{i,j} = [A_{i,j}^{(1)}, A_{i,j}^{(2)}, \dots, A_{i,j}^{(m_j)}]$ of size $m \times m$. Note that each of $A_{i,j}^{(1)}, A_{i,j}^{(2)}, \dots, A_{i,j}^{(m_j)}$ has size of m because all the canonical components are kept without dimension reduction.
4. Arrange the obtained coefficient matrix $A_{i,j}$ into a $p \times p$ block diagonal matrix R_i, where

$$R_i = \begin{bmatrix} A_{i,1}^{(1)}, A_{i,1}^{(2)}, \dots, A_{i,1}^{(m_1)} & [0] & \cdots \\ [0] & \ddots & \vdots \\ \vdots & \cdots & A_{i,K}^{(1)}, A_{i,K}^{(2)}, \dots, A_{i,K}^{(m_K)} \end{bmatrix}$$

5. Construct the rotation matrix R_i^a (a $p \times p$ matrix) by rearranging the rows of R_i so that they correspond to the original features in F.
6. The new training data for classifier L_i is (XR_i^a, Y).

Figure 2 shows the pseudocode for the entire algorithm of Canonical Forest. Here decision trees are used as base classifiers. Since the centroids of C classes in p-dimensional input space span at most $C - 1$ dimensional subspace, it is common to extract $C - 1$ features in CLDA for dimension reduction in general. However, we use CLDA as a linear transformation tool rather than a dimension reduction tool for this study. Therefore, we let the number of extracted features to be the number of the original features. This results in m extracted features in each subset of features even if m is greater than $C - 1$. Although these extra features may not contribute much to the discriminatory power, they will encourage the classifiers to be more diverse and it can yield higher accuracy of the ensemble.

```
Input
Given
    • $X$: training data composed of n instances (an $n \times p$ matrix)
    • $Y$: the labels of the training data (an $n \times 1$ vector)
    • $B$: number of classifiers in an ensemble
    • $K$: number of subsets
    • $w = (1, \cdots, C)$: set of class labels

Training Phase
For $i = 1, \dots, B$
    1. Randomly split $\boldsymbol{F}$ (the feature set) into $K$ subsets: $F_{i,j}$ (for $j = 1, \dots, K$)
    2. For $j = 1, \dots, K$
            ▪ Let $X_{i,j}$ be the data matrix that corresponds to the features in $F_{i,j}$
            ▪ Draw a bootstrap sample $X'_{i,j}$ (with sample size 75% of the number
              of instances in $X_{i,j}$) from $X_{i,j}$
            ▪ Apply CLDA to $X'_{i,j}$ to obtain a coefficient matrix $A_{i,j}$
    3. Arrange $A_{i,j}$ ($j = 1, \cdots, K$) into a block diagonal matrix $R_i$
    4. Construct the rotation matrix $R_i^a$ by rearranging the rows of $R_i$ so that they
       correspond to the original order of features in $\boldsymbol{F}$
    5. Use $(XR_i^a, Y)$ as the training data to build a classifier $L_i$

Test Phase
        • For a given instance $x$, the predicted class label from classifier $L$ is:
              $L(x) = argmax_{y \in w} \sum_{i=1}^{B} I(L_i\,(xR_i^a) = y)$
```

Figure 2: Pseudocode of Canonical Forest.

3.3 Canonical Forest with the Weight-Adjusted Voting Algorithm

Majority voting is a simple combining algorithm often used in ensemble methods when combining base classifiers. Although it is simple and efficient, sometimes it can be biased since it doesn’t take the performance of classifiers into account but assigns equal weights to each classifier instead. This could be a problem when some classifiers perform better on hard-to-classify instances. Here hard-to-classify instances are defined as those instances that are correctly classified by only a few classifiers. Therefore, the results might be misled if classifiers that perform better on hard-to-classify instances are assigned the same weight as those that perform worse on hard-to-classify instances. To solve this problem, Kim et al. (2011) proposed a new weighted voting algorithm named WAVE (Weight-Adjusted Voting for Ensemble of Classifiers). Unlike other weighted voting algorithms, WAVE can be applied after all the classifiers are built. That is, the classifiers can still be trained in parallel when applying WAVE.

The general idea of WAVE is to assign more weights to the classifiers that can classify hard-to-classify instances better. To assign weights to each classifier properly, WAVE uses both the weight vector of classifiers and the weight vector of instances. The instances that are correctly classified by fewer classifiers receive more weight and the classifiers that perform better on those hard-to-classify instances also receive more weight. Therefore, these two weight vectors are influenced by each other and are updated iteratively through a repeated process. The algorithms of obtaining weight vectors for both instances and classifiers are shown in Figure 3. Once the weight matrix is obtained, the classifiers are then combined using this weight matrix in an ensemble algorithm. To make the calculations easier, Kim et al. (2011) proved the convergence of both weight vectors of instances and classifiers and obtained the final weight vectors of instances and classifiers directly without going through an iterative process. The algorithm for obtaining the final weight vectors directly is shown in Figure 4. Because the weight vector of classifiers is obtained after the ensemble is formed, we can easily apply it to Canonical Forest to improve the performance.

```
Given
  • $n$: number of instances in data
  • $B$: number of classifiers in an ensemble
  • $W$: an $n \times B$ performance matrix indicating whether the prediction is
        correct (1) or wrong (0)
  • $J_{ij}$: an $i \times j$ matrix with all the entries equal to 1
  • $1_n$: an $n \times 1$ vector of 1's
  • $I_B$: a $B \times B$ identity matrix

Procedure:
  1. Calculate $Q_0$, the initial weight vector of instances:
         $Q_0 = \frac{(J_{nB}-W)(J_{BB}-I_B)1_B}{1'_n(J_{nB}-W)(J_{BB}-I_B)1_B}$
  2. For $m = 1,2,\ldots$, calculate $P_m$ and $Q_m$ by repeating the following process
     until both $P_m$ and $Q_m$ become stable, where $P_m$ is the weight vector of
     classifiers.
       • $P_m = \frac{W'Q_{m-1}}{1'_k W' Q_{m-1}}$
       • $Q_m = \frac{(J_{nB}-W)(J_{BB}-I_B)1_B}{1'_n(J_{nB}-W)(J_{BB}-I_B)1_B}$
  3. After $P_m$ and $Q_m$ become stable, denote $P^*$ and $Q^*$ as the final weight
     vectors for classifiers and instances, respectively.
```

Figure 3: Pseudocode of obtaining the weight vectors through an iterative process.

```
Given
    • $n$: number of instances in data
    • $B$: number of classifiers in an ensemble
    • $W$: an $n \times B$ performance matrix indicating whether the prediction is
          correct (1) or wrong (0)
    • $J_{ij}$: an $i \times j$ matrix with all the entries equal to 1
    • $1_n$: an $n \times 1$ vector of 1's
    • $I_B$: a $B \times B$ identity matrix

Procedure:
    1. Let $T = W'(J_{nB} - W)(J_{BB} - I_B)$
    2. Find $\lambda_i$, eigenvalues of $T,\ i = 1, \dots, B$
    3. Find $\mu_i$, eigenvector corresponding to $\lambda_i,\ i = 1, \dots, B$
    4. Let $r$ be the number of dominating eigenvalues such that $\lambda_1 = \lambda_2 =$
       $\cdots = \lambda_r > \lambda_{r+1}, 1 \leq r \leq B$
    5. Compute $P^* = \frac{(\sum_{i=1}^{r} \mu_i \mu_i')1_B}{1_B'(\sum_{i=1}^{r} \mu_i \mu_i')1_B} = [P_1^*, \cdots, P_B^*]'$, where $P^*$ is the weight
       matrix of classifiers
    6. Compute $Q^* = \frac{(J_{nB}-W)(J_{BB}-I_B)(\sum_{i=1}^{r} \mu_i \mu_i')1_B}{1_n'(J_{nB}-W)(J_{BB}-I_B)(\sum_{i=1}^{r} \mu_i \mu_i')1_B} = [Q_1^*, \cdots, Q_n^*]'$, where $Q^*$
       is the weight matrix of instances
```

Figure 4: Pseudocode of directly obtaining the final weight vectors.

3.4 High-Dimensional Canonical Forest (HDCF)

Although Canonical Forest is an ensemble algorithm based on CLDA which is a commonly used dimension reduction tool for dealing with high-dimensional data, it cannot be directly applied to high-dimensional data since CLDA only serves as a linear transformation tool rather than a dimension reduction tool in Canonical Forest. Therefore, like most of the other classification methods, Canonical Forest needs to perform a feature selection or feature reduction before classifying on high-dimensional data. In order for Canonical Forest to be directly applied to high-dimensional data, we propose a different version of Canonical Forest called High-Dimensional Canonical Forest (HDCF) by implementing the algorithm of Random Subspace (Ho 1998) into Canonical Forest.

Random Subspace is an ensemble algorithm that utilizes the random feature subspace to increase the diversity of classifiers. Unlike Bagging diversifies the classifiers by repeatedly resampling from the training data, Random Subspace diversifies the classifiers by randomly selecting feature subsets from the original feature space. That is, Random Subspace only uses a random subset of features instead of all features to construct a classifier. Since all the classifiers are generated independently, the classifiers can be trained in parallel in Random Subspace. However, it should be noted that generating independently does not mean that the classifiers are independent. After each classifier is constructed, an average voting method is used to combine these classifiers. By using only some instead of all features, Random Subspace not only increases the diversity but also easily overcomes the problem caused by high-dimensional data and hence does not require a feature selection or feature reduction before classification. Therefore, Random Subspace is most useful when classifying on high-dimensional data but may be a bad choice when classifying on data with only a few features. The entire algorithm of Random Subspace is shown in Figure 5.

Here we naturally extend Canonical Forest to be applied on high-dimensional data by implementing the idea of random feature subspace into the algorithm of Canonical Forest. That is, only some randomly selected features from the whole features are used when growing each tree in Canonical Forest. This new classification ensemble algorithm is called High-Dimensional Canonical Forest (HDCF) since it is specifically designed for high-dimensional data. The only difference between HDCF and Canonical Forest is that HDCF uses only a subset of whole feature set while Canonical Forest uses the whole feature set when growing a tree. Figure 6 shows the pseudocode for the entire algorithm of HDCF. Like Canonical Forest, decision trees are used as base classifiers in HDCF.

```
**Input**
Given
    • $X$: training data composed of n instances (an $n \times p$ matrix)
    • $Y$: the labels of the training data (an $n \times 1$ vector)
    • $B$: number of classifiers in an ensemble
    • $q$: number of features selected from the feature set
    • $w = (1, \cdots, C)$: set of class labels

**Training Phase**
For $i = 1, \ldots, B$
    1. Randomly select a feature subset $f_i$ of size q from $\boldsymbol{F}$ (the feature set)
    2. Let $X_i$ be the data matrix that corresponds to the features in $f_i$
    3. Use $(X_i, Y)$ as the training data to build a classifier $L_i$

**Test Phase**
        • For a given instance $x$, let $p_i(w|x_i)$ be the probability that $x$ belongs to
          class $w$ from classifier $L_i$. The average probability that $x$ belongs to class $w$
          from $B$ classifiers is:
              $p(w|x) = \frac{1}{B}\sum_{i=1}^{B} p_i(w|x_i)$
        • Assign $x$ to class $w$ where $p(w|x)$ is the maximum
```

Figure 5: Pseudocode of Random Subspace.

```
Input
Given
    • $X$: training data composed of n instances (an $n \times p$ matrix)
    • $Y$: the labels of the training data (an $n \times 1$ vector)
    • $B$: number of classifiers in an ensemble
    • $K$: number of subsets
    • $q$: number of features selected from the feature set
    • $w = (1, \cdots, C)$: set of class labels

Training Phase
For $i = 1, \dots, B$
    1. Randomly select a feature subset $f_i$ of size q from $\boldsymbol{F}$ (the feature set)
    2. Randomly split $f_i$ into K subsets: $f_{i,j}$ (for j = 1, . . ., $K$) and Let $X_i$ be the data
       matrix that corresponds to the features in $f_i$
    3. For $j = 1, \dots, K$
            ▪ Let $X_{i,j}$ be the data matrix that corresponds to the features in $f_{i,j}$
            ▪ Draw a bootstrap sample $X'_{i,j}$ (with sample size 75% of the number
              of instances in $X_{i,j}$) from $X_{i,j}$
            ▪ Apply CLDA to $X'_{i,j}$ to obtain a coefficient matrix $A_{i,j}$
    4. Arrange $A_{i,j}$ $(j = 1, \cdots, K)$ into a block diagonal matrix $R_i$
    5. Construct the rotation matrix $R_i^a$ by rearranging the rows of $R_i$ so that they
       correspond to the original order of features in $f_i$
    6. Use $(X_i R_i^a, Y)$ as the training data to build a classifier $L_i$

Test Phase
        • For a given instance x, the predicted class label from classifier $L$ is:
                $L(x) = argmax_{y \in w} \sum_{i=1}^{B} I(L_i\,(x_i R_i^a) = y)$
```

Figure 6: Pseudocode of High-Dimensional Canonical Forest.

Chapter 4

Experiments

4.1 Performance Comparison

In this dissertation, we conducted an experiment using 29 real or artificial data sets to compare the performance of Canonical Forest with the performance of other widely used ensemble methods including Bagging, Adaboost, Samme, Random Forest, and Rotation Forest. We also provided weighted voting version of Canonical Forest using WAVE to further improve the performance of Canonical Forest. Decision trees were used as the base classifiers for all the ensemble methods. We used unpruned decision trees as base classifiers except for Adaboost and Samme. For Adaboost and Samme, we set the maximum depth of each single tree to be the number of classes, which is the default setting in the R package called adaboost.M1. A decision tree program named rpart available in the R package is used for the experiment. Rpart is based on the CART (Classification and Regression Trees) algorithm (Breiman et al., 1984). In Random Forest, we set the number of features chosen at each node equal to the square root of the number of all features, which is the default setting in the R Random Forest package named randomForest. In Canonical Forest and Rotation Forest, the number of features in each subset is set to be $m = 3$. If p is not divisible by 3, then the last subset was completed with the remaining features (1 or 2 features). It should be noted that the performance of all the ensemble methods were compared under the fixed same ensemble size for fair comparisons.

The data are summarized in Table 1. Most of the data come from UCI Data Repository (Asuncion and Newman, 2007) and package mlbench (Leisch and Dimitriadou, 2010) of the R library. Since PCA and CLDA cannot be applied to discrete features, all the discrete features have been removed from each data set.

Table 1: Data used in the comparison of Canonical Forest with other ensemble methods.

Data Set	Observations	Continuous Features	Class	Source
aba	4177	7	2	UCI
aus	690	11	2	UCI
bld	345	6	2	UCI
bod	507	24	2	Heinz et al. (2003)
bos	506	12	3	UCI
cir	1000	10	2	R library
dia	768	8	2	Loh (2010)
ech	131	5	2	UCI
fis	159	6	7	Kim and Loh (2003)
hea	270	10	2	UCI
int	1000	9	2	Kim et al. (2011)
ion	351	32	2	UCI
iri	150	4	3	UCI
lak	259	13	6	Loh (2010)
led	6000	7	10	UCI
mam	961	5	2	R library
pid	532	7	2	UCI
pks	195	22	2	R library
pov	97	6	6	Kim and Loh (2001)
rng	1000	10	2	R library
sea	3000	7	3	Terhune (1994)
snr	208	60	2	R library
spe	267	44	2	UCI
trn	1000	10	2	R library
twn	1000	10	2	R library
usn	1302	26	3	Statlib (2010)
veh	846	18	4	UCI
vol	1521	4	6	Loh (2010)
vow	990	10	11	UCI

Figure 7 shows boxplots comparing Canonical Forest with the other ensemble methods for ensemble sizes 2, 4, 8, 16, 32, and 64. Twenty repetitions of 3-fold cross-validation were performed for each of the data sets. Here the comparison was done by using paired t-test. The x-axis indicates the ensemble size, B, and the y-axis indicates the t-test statistic. A positive t-test statistic shows that Canonical Forest has a better performance; a negative t-test statistic shows that Canonical Forest has a poorer performance. From Figure 7, we can see that the t-test statistic increases as ensemble size increases for Decision Trees and Bagging; t-test statistic decreases as ensemble size increases for Random Forest; no obvious trends are found for Adaboost, Samme, and Rotation Forest.

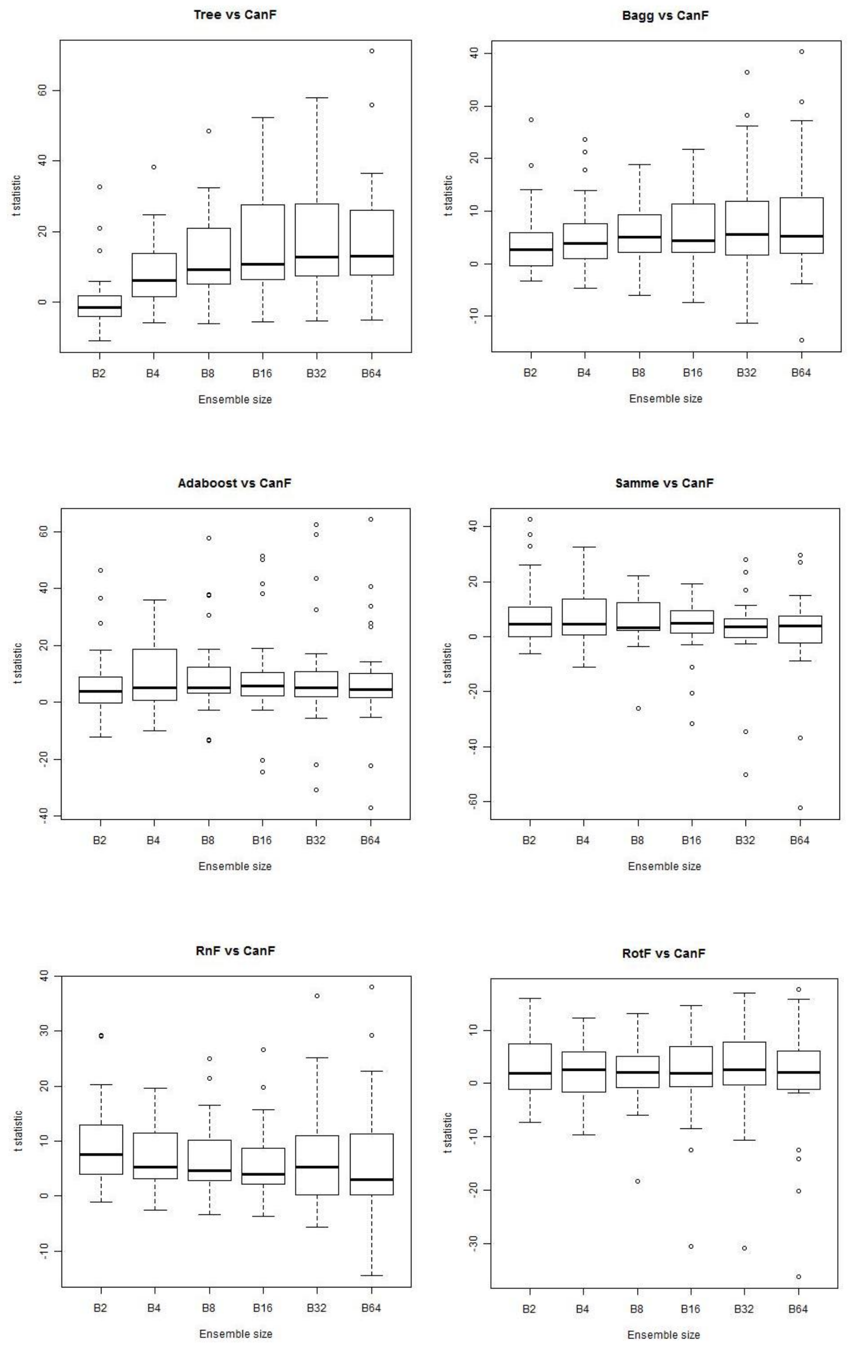


Figure 7: Boxplots of other ensemble methods compared to Canonical Forest using paired t-test. CanF stands for Canonical Forest; RnF stands for Random Forest; RotF stands for Rotation Forest.

Figure 8 shows the relative improvement of the other ensemble methods compared to Canonical Forest for ensemble sizes 2, 4, 8, 16, 32, and 64 using boxplots. The relative improvement is defined as

$$RI = \frac{error\ rate\ of\ A - error\ rate\ of\ B}{error\ rate\ of\ A}$$

Here B is Canonical Forest and A is the ensemble method to be compared. From Figure 8, we can see that the relative improvement increases as the ensemble size increases for Decision Tree and Bagging; it decreases as the ensemble size increases for Random Forest; there are no obvious trends for Adaboost, Samme, and Rotation Forest.

The result using relative improvement is quite consistent with the result using paired t-test. In general, we see that Canonical Forest is better than any other ensemble methods at each ensemble size, especially when the ensemble size is small.

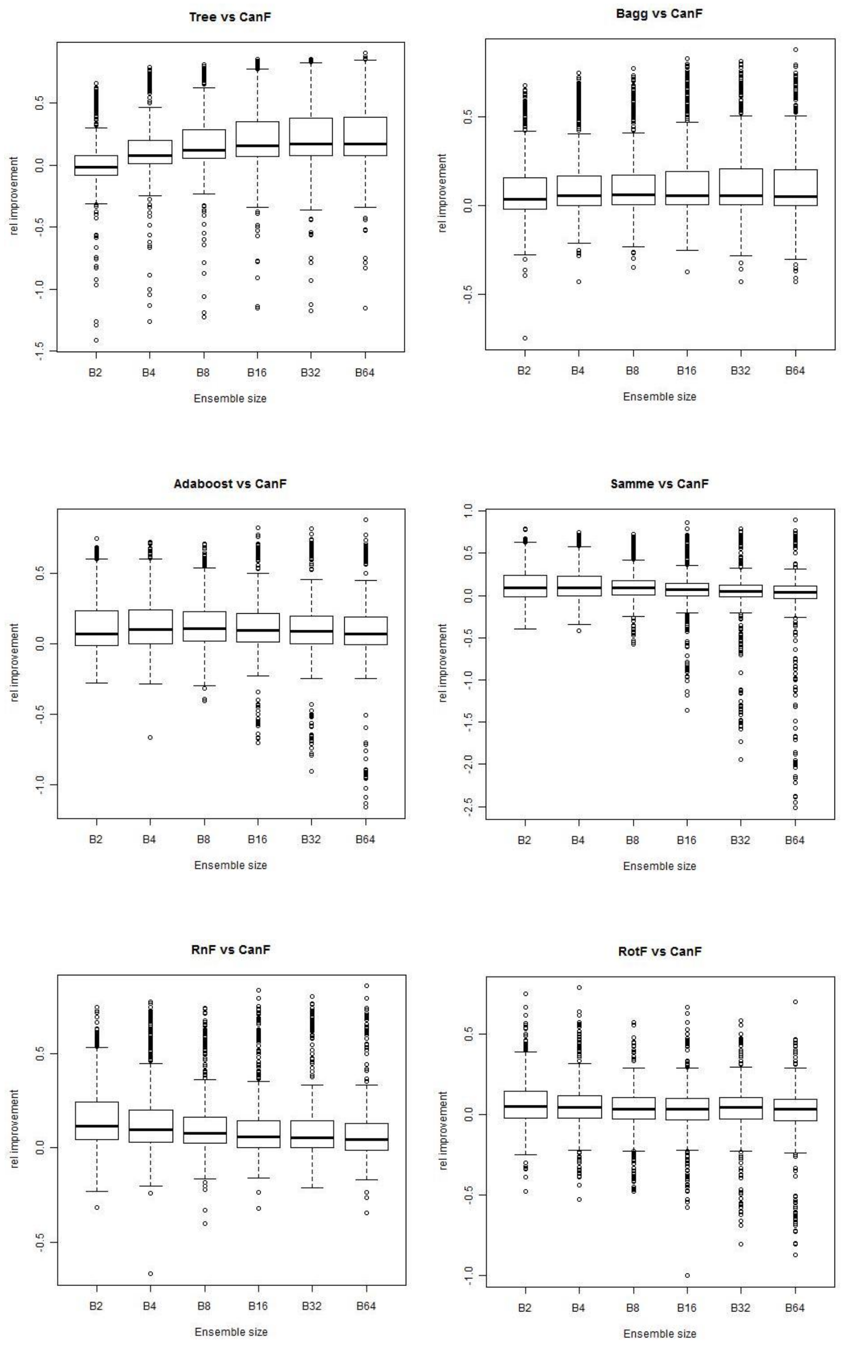


Figure 8: Relative Improvement of the other ensemble methods compared to Canonical Forest represented using box-plot. CanF stands for Canonical Forest; RnF stands for Random Forest; RotF stands for Rotation Forest.

To compare the accuracy among different ensemble methods, we fixed the ensemble size at $B = 64$ and 500. For each data set and ensemble method, twenty 3-fold cross validations were performed. The average accuracies are shown in Table 2 and Table 3. We also show the accuracy of single tree as a reference.

The results are marked with a plus or minus sign next to them if there is a significant difference between the respective ensemble method and Canonical Forest. If Canonical Forest is significantly more accurate than the compared method (column) for the particular data set (row), then we put a plus sign next to a result. If Canonical Forest is significantly less accurate, then we put a minus sign next to a result. The second + or - sign in Table 3 indicates that Canonical Forest with WAVE is used to compare with other methods. The comparison was done by using paired t-test at two-sided significant level $\alpha = 0.05$.

Table 2: Classification accuracy of tree and ensemble methods comparing with Cononical Forest with ensemble size $B = 64$.

Data Set	Tree	Bagging	Adaboost	Samme	Random Forest	Rotation Forest	Canonical Forest
aba	0.7628+	0.7755+	0.7334+	0.7711+	0.7697+	0.7820	0.7814
aus	0.8546+	0.8648	0.8542+	0.8411+	0.8624	0.8490+	0.8628
bld	0.6516+	0.7007+	0.6854+	0.7039+	0.7148+	0.7041+	0.7268
bod	0.9036+	0.9282+	0.9735+	0.9748+	0.9402+	0.9729+	0.9776
bos	0.7400+	0.7710	0.7715+	0.7760	0.7787	0.7615+	0.7763
cir	0.7054+	0.7874 -	0.6674+	0.8852 -	0.7852 -	0.7796	0.7780
dia	0.7446+	0.7555+	0.7520+	0.7477+	0.7569+	0.7542+	0.7617
ech	0.6347+	0.6931	0.6702+	0.6668+	0.6878	0.6550+	0.6920
fis	0.8025+	0.8252+	0.8264+	0.8179+	0.8145+	0.9145+	0.9428
hea	0.7430+	0.7787+	0.7806+	0.7561+	0.7891+	0.7694+	0.7980
int	0.6666 -	0.6586 -	0.5234+	0.5924 -	0.5294+	0.7347 -	0.5664
ion	0.8736+	0.9056+	0.9199+	0.9217+	0.9238+	0.9430 -	0.9349
iri	0.9457+	0.9457+	0.9440+	0.9443+	0.9487+	0.9523	0.9580
lak	0.3651+	0.4041	0.4027	0.4023+	0.4344 -	0.4027	0.4147
led	0.6830+	0.7076	0.7331 -	0.6967+	0.7234 -	0.7218 -	0.7084
mam	0.8268	0.8314 -	0.8318 -	0.8262	0.8329 -	0.8213+	0.8270
pid	0.7508+	0.7766	0.7731+	0.7574+	0.7756	0.7695+	0.7808
pks	0.8456+	0.8800+	0.8651+	0.9128 -	0.8862+	0.8959	0.8964
pov	0.6227+	0.6139+	0.6448	0.6320+	0.6000+	0.6129+	0.6603
rng	0.8186+	0.8688+	0.7280+	0.9150 -	0.8870+	0.9044	0.9024
sea	0.5872+	0.6068+	0.5432+	0.6465 -	0.6082+	0.6052+	0.6338
snr	0.7154+	0.7834+	0.8216	0.8392	0.8075+	0.8363	0.8272
spe	0.7620+	0.8052	0.8060	0.7903+	0.8096	0.8150	0.8103
trn	0.7280+	0.8267+	0.8399	0.8424 -	0.8454 -	0.8572 -	0.8380
twn	0.8212+	0.9342+	0.9608+	0.9588+	0.9558+	0.9684	0.9696
usn	0.6748+	0.7099+	0.7270 -	0.7047+	0.7155	0.7126	0.7166
veh	0.6757+	0.7080+	0.7342+	0.7483+	0.7181+	0.7369+	0.7645
vol	0.5176+	0.5383+	0.5382+	0.4996+	0.5320+	0.5130+	0.5517
vow	0.5582+	0.6518+	0.8653 -	0.9149 -	0.7198+	0.7805 -	0.7472
(Win/Tie/Loss)	(1/1/27)	(3/7/19)	(4/5/20)	(7/3/19)	(5/6/18)	(5/10/14)	

+ Canonical Forest is significantly better, - Canonical Forest is significantly worse, level of significance 0.05

Table 3: Classification accuracy of tree and ensemble methods comparing with Cononical Forest with ensemble size $B = 500$.

Data Set	Tree	Bagging	Adaboost	Samme	Random Forest	Rotation Forest	Canonical Forest	Canonical Forest with WAVE
aba	0.7628+ +	0.7764+ +	0.7336+ +	0.7796+ +	0.7709+ +	0.7825	0.7817	0.7819
aus	0.8546+ +	0.8667 - -	0.8544+ +	0.8354+ +	0.8651	0.8476+ +	0.8631	0.8641
bld	0.6516+ +	0.7096+ +	0.6968+ +	0.6852+ +	0.7170+ +	0.7070+ +	0.7317	0.7313
bod	0.9036+ +	0.9305+ +	0.9772	0.9792	0.9404+ +	0.9727+ +	0.9775	0.9785
bos	0.7400+ +	0.7731	0.7762 - -	0.7862 - -	0.7828 - -	0.7633+ +	0.7710	0.7718
cir	0.7054+ +	0.7883 - -	0.6590+ +	0.9132 - -	0.7872 - -	0.7733	0.7744	0.7758
dia	0.7446+ +	0.7579+ +	0.7537+ +	0.7304+ +	0.7601 +	0.7558+ +	0.7628	0.7636
ech	0.6347+ +	0.6912	0.6718+ +	0.6546+ +	0.6912	0.6618+ +	0.6927	0.6905
fis	0.8028+ +	0.8233+ +	0.8248+ +	0.8189+ +	0.8173+ +	0.9182+ +	0.9456	0.9497
hea [1]	0.7050+ +	0.7339+ +	0.7554	0.6939+ +	0.7507	0.7228+ +	0.7474	0.7481
int	0.6664 - -	0.6764 - -	0.5302+ +	0.6514 - -	0.5310+ +	0.7335 - -	0.5618	0.5697
ion	0.8736+ +	0.9091+ +	0.9222+ +	0.9245+ +	0.9266+ +	0.9442 - -	0.9360	0.9365
iri	0.9457+ +	0.9443+ +	0.9470+ +	0.9433+ +	0.9477+ +	0.9540	0.9597	0.9593
lak	0.3647+ +	0.4085	0.4066	0.4075	0.4365 - -	0.4058	0.4158	0.4145
led	0.6830+ +	0.7078 +	0.7343 - -	0.5829+ +	0.7296 - -	0.7223 - -	0.7090	0.7121
mam	0.8268	0.8320 - -	0.8334 - -	0.8157+ +	0.8339 - -	0.8230+ +	0.8283	0.8277
pid	0.7513+ +	0.7780+ +	0.7744+ +	0.7448+ +	0.7787	0.7706+ +	0.7820	0.7817
pks	0.8467+ +	0.8810+ +	0.8715+ +	0.9233 - -	0.8844+ +	0.9003	0.8938	0.8936
pov	0.6211+ +	0.6129+ +	0.6505 +	0.5897+ +	0.6062+ +	0.6077+ +	0.6624	0.6665
rng	0.8184+ +	0.8719+ +	0.7320+ +	0.9210 - -	0.8878+ +	0.9060	0.9059	0.9069
sea	0.5872+ +	0.6083+ +	0.5446+ +	0.6737 - -	0.6112+ +	0.6077+ +	0.6370	0.6370
snr	0.7154+ +	0.7863+ +	0.8351	0.8560 - -	0.8111+ +	0.8387	0.8322	0.8325
spe	0.7599+ +	0.8062+ +	0.8054+ +	0.7919+ +	0.8101	0.8133	0.8118	0.8116
trn	0.7280+ +	0.8268+ +	0.8452	0.8472	0.8476	0.8612 - -	0.8444	0.8453
twn	0.8212+ +	0.9389+ +	0.9644+ +	0.9620+ +	0.9618+ +	0.9700	0.9694	0.9695
usn	0.6748+ +	0.7134+ +	0.7284 - -	0.7111+ +	0.7192	0.7144+ +	0.7193	0.7198
veh	0.6757+ +	0.7100+ +	0.7370+ +	0.7606 +	0.7220+ +	0.7398+ +	0.7655	0.7697
vol	0.5176+ +	0.5384+ +	0.5381+ +	0.5082+ +	0.5327+ +	0.5100+ +	0.5517	0.5537
vow	0.5582+ +	0.6553+ +	0.8853 - -	0.9376 - -	0.7374+ +	0.7870 - -	0.7589	0.7641
(Win/Tie/Loss)[2]	(1/1/27)	(4/4/21)	(5/6/18)	(8/4/17)	(5/8/16)	(5/9/15)		
(Win/Tie/Loss)[3]	(1/1/27)	(4/3/22)	(5/5/19)	(8/3/18)	(5/7/17)	(5/9/15)		

+ Canonical Forest is significantly better, - Canonical Forest is significantly worse (the second one stands for Canonical Forest with WAVE)

1 Some features of data set hea were removed due to computational problems

2 The summary result of Canonical Forest

3 The summary result of Canonical Forest with WAVE

Table 4 and Table 5 show the summaries of the comparisons given in Tables 2 and 3, respectively. The entry a_{ij} shows the frequency that the method in column (j) is more accurate than the method in row (i). The number in parentheses shows the frequency that these differences are statistically significant. In Table 4, for example, the value in row 2, column 7 is 24 (19). This means that Canonical Forest was more accurate than Bagging in 24 of the 29 comparisons and less accurate in 5 comparisons. The number in the parentheses indicates that Canonical Forest has been significantly better than Bagging in 19 data sets. In 3 of the remaining 5 cases, Bagging was significantly better than Canonical Forest (the entry in row 7, column 2 is 5 (3)).

Table 4: Summary of comparisons among different methods with ensemble size $B = 64$.

	Tree	Bagging	Adaboost	Samme	Random Forest	Rotation Forest	Canonical Forest
Tree	-	26 (26)	22 (22)	24 (21)	27 (25)	25 (24)	28 (27)
Bagging	2 (0)	-	15 (10)	14 (11)	20 (14)	17 (15)	24 (19)
Adaboost	7 (5)	14 (9)	-	15 (10)	19 (14)	16 (14)	24 (20)
Samme	5 (3)	15 (13)	14 (9)	-	16 (11)	17 (13)	21 (19)
Random Forest	2 (2)	9 (4)	10 (10)	13 (11)	-	15 (12)	23 (18)
Rotation Forest	4 (1)	12 (7)	13 (9)	12 (8)	14 (10)	-	19 (14)
Canonical Forest	1 (1)	5 (3)	5 (4)	8 (7)	6 (5)	10 (5)	-

The entry a_{ij} shows the frequency that the method in column (j) is more accurate than the method in row (i). The number in parentheses shows the frequency that these differences are statistically significant.

Table 5: Summary of comparisons among different methods with ensemble size $B = 500$.

	Tree	Bagging	Adaboost	Samme	Random Forest	Rotation Forest	Canonical Forest
Tree	-	27 (26)	23 (22)	19 (19)	27 (26)	25 (25)	28 (27)
Bagging	2 (0)	-	15 (12)	13 (13)	21 (16)	16 (15)	24 (21)
Adaboost	6 (5)	14 (10)	-	14 (11)	18 (11)	16 (13)	21 (18)
Samme	10 (6)	16 (11)	15 (14)	-	16 (12)	19 (15)	19 (17)
Random Forest	2 (1)	7 (3)	11 (10)	13 (10)	-	15 (13)	21 (16)
Rotation Forest	4 (1)	13 (8)	13 (10)	10 (9)	14 (14)	-	18 (15)
Canonical Forest	1 (1)	5 (4)	8 (5)	10 (8)	8 (5)	11 (5)	-

The entry a_{ij} shows the frequency that the method in column (j) is more accurate than the method in row (i). The number in parentheses shows the frequency that these differences are statistically significant.

Tables 6 and 7 present the ranking of the methods based on the frequency that each method was significantly more accurate and significantly less accurate than other method. In Table 6, for example, the number of wins for Canonical Forest is 117. It is obtained by the sum of the numbers in parentheses in the column of Canonical Forest in Table 4. Similarly, the number of losses is obtained by the sum of the numbers in parentheses in the row of Canonical Forest which is 25. Therefore, the dominance rank of Canonical Forest is $117 - 25 = 92$ at ensemble size $B = 64$.

Tables 4 through 7 clearly show that Canonical Forest outperformed other widely used ensemble methods: Bagging, Adaboost, Samme, Random Forest, and Rotation Forest. The dominance rank of Canonical Forest (92 and 86) is substantially larger than the second best at both ensemble sizes.

Table 6: Rank of the methods using the significantly different results from the results in Table 4.

Method	Dominance rank (Wins-Losses)	Wins	Losses
Canonical Forest	92	117 (82%)	25 (18%)
Rotation Forest	34	83 (63%)	49 (37%)
Random Forest	22	79 (58%)	57 (42%)
Samme	0	68 (50%)	68 (50%)
Bagging	-7	62 (47%)	69 (53%)
Adaboost	-8	64 (47%)	72 (53%)
Tree	-133	12 (8%)	145 (92%)

Table 7: Rank of the methods using the significantly different results from the results in Table 5.

Method	Dominance rank (Wins-Losses)	Wins	Losses
Canonical Forest	86	114 (80%)	28 (20%)
Random Forest	31	84 (61%)	53 (39%)
Rotation Forest	29	86 (60%)	57 (40%)
Adaboost	5	73 (52%)	68 (48%)
Samme	-5	70 (48%)	75 (52%)
Bagging	-15	62 (45%)	77 (55%)
Tree	-131	14 (9%)	145 (91%)

To confirm the superiority of Canonical Forest is not just by chance, we performed the exact Binomial test on the classification accuracy of Tables 2 and 3. The one-sided test was adopted to obtain the p-values for the alternative hypothesis that Canonical Forest performs better than another method. That is, the null hypothesis was set to be $p = 0.5$ and the alternative hypothesis was set to be $p > 0.5$, where p is the probability that Canonical Forest performs better than another method for a given data set. It should be noted that the problem of multiple comparisons may arise when we are trying to test all hypotheses simultaneously. When testing several hypotheses at the same time, although the type I error rate for each test is controlled at $\alpha = 0.05$, the overall type I error rate becomes $1 - 0.95^m$, where m is the number of hypotheses tested simultaneously. Since here we would like to test 6 hypotheses at the same time, the overall type I error rate increases to 0.265 when the type I error rate for each test is controlled at $\alpha = 0.05$. That is, the probability of having at least one type I error in these 6 comparisons is equal to 0.265 which is quite high. Therefore, here we used Holm-Bonferroni (Holm, 1979) correction to adjust the significance level for a multiple test to maintain the overall type I error rate of $\alpha = 0.05$. The entire algorithm of Holm-Bonferroni correction is shown is Figure 9. By using Holm-Bonferroni correction, the probability of having at least one type I error in these 6 comparisons is now less than or equal to $\alpha = 0.05$.

Table 8 shows the p-values of the exact Binomial test. The six adjusted α, from the smallest to the largest, would be 0.008, 0.01, 0.013, 0.017, 0.025, and 0.05. The p-values arranged in increasing order are compared with the Holm-Bonferroni adjusted α values. The Holm-Bonferroni multiple comparison is performed sequentially beginning with the smallest p-value. As a result, for the ensemble size of $B = 64$, Canonical Forest is significantly more accurate than the other methods. Except Samme, for the ensemble size of $B = 500$, Canonical Forest is also significantly more accurate than the other methods. The p-value for the comparison with Samme, however, is quite close to the adjusted significance level of $\alpha = 0.05$. It should be noted that after applying WAVE to Canonical Forest, its performance became significantly more accurate than that of all the other methods.

We did not include Canonical Forest with WAVE for the ensemble size of $B = 64$ because Canonical Forest performed very well with a small ensemble size. The gap in performance between Canonical Forest and the other methods decreases as the ensemble size increases. As we have expected, the classification accuracy of Canonical Forest has been slightly improved after implementing WAVE for the ensemble size of $B = 500$.

Given:

- m: number of null hypotheses
- p_i: the corresponding p-value for null hypothesis H_i
- α: type I error rate

Procedure:

9. Order the p-values such that $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}$
10. Let $H_{(i)}$ be the corresponding null hypothesis of $p_{(i)}$
11. Compare $p_{(i)}$ to $\frac{\alpha}{m-(i-1)}$ starting from $i = 1$ until find the smallest index r such that $p_{(r)} > \frac{\alpha}{m-(r-1)}$ where $1 \leq r \leq m$
12. Do not reject any null hypotheses when $r = 1$; reject all the null hypotheses when $r = m$
13. For $1 < r < m$, reject null hypotheses $H_{(1)}, H_{(2)}, \cdots, H_{(r-1)}$ and do not reject hull hypotheses $H_{(r)}, H_{(r+1)}, \cdots, H_{(m)}$

Figure 9: Algorithm of Holm-Bonferroni correction.

Table 8: P-values of exact Binomial test for comparing Canonical Forest with other methods.

Adjusted α	0.008	0.01	0.0125	0.017	0.025	0.05
Method[1]	Tree	Bagging	Adaboost	Samme	Random Forest	Rotation Forest
p-value[1]	1×10^{-7}•	0.0004•	0.0008•	0.0053•	0.0145•	0.0318•
Method[2]	Tree	Bagging	Adaboost	Random Forest	Rotation Forest	Samme
p-value[2]	1×10^{-7}•	0.0004•	0.0053•	0.0133•	0.0207•	0.0539
Method[3]	Tree	Bagging	Adaboost	Random Forest	Rotation Forest	Samme
p-value[3]	1×10^{-7}•	0.0003•	0.0033•	0.0085•	0.0207•	0.0378•

• The result is significant by using Holm-Bonferroni Correction at $\alpha = 0.05$
1 p-value was obtained comparing with Canonical Forest at ensemble size 64
2 p-value was obtained comparing with Canonical Forest at ensemble size 500
3 p-value was obtained comparing with Canonical Forest with WAVE at ensemble size 500

4.2 Diversity and κ-error Diagram

In addition to the accuracy, we investigated the diversity of each ensemble method since the diversity plays an important role in the performance of an ensemble method. In general, ensemble methods tend to perform better if the base classifiers are more diverse. Here kappa statistic κ (Cohen, 1960) was used to investigate the diversity of base classifiers in ensemble methods. The kappa statistic κ is a measure of agreement commonly used to measure the degree of agreement between two categorical variables. In classification analysis, the agreement between two classifiers L_i and L_j can also be measured by kappa statistic κ (Margineantu and Dietterich, 1997; Dietterice, 2000; Kuncheva and Whitaker, 2003; Rodríguez et al., 2006). To calculate κ between two classifiers, we first have to compute p_1, the relative observed agreement between two classifiers which is an estimate of the probability of reaching agreement between two classifiers, and p_2, the probability of reaching agreement between two classifiers by chance, given the observed counts (Margineantu and Dietterich 1997). Then kappa statistic κ can be calculated by $\kappa = \frac{p_1 - p_2}{1 - p_2}$. The entire algorithm of computing κ between two classifiers is shown in Figure 10. An ensemble of B classifiers will have $B(B-1)/2$ pairs of classifiers (L_i, L_j). Small values of κ indicate high disagreement (high diversity) between two classifiers and large values of κ indicate high agreement (low diversity) between two classifiers. When $\kappa = 1$, these two classifiers can be considered identical. Since high diversity among classifiers in an ensemble would produce small values of κ, we prefer κ to be as small as possible.

Given:

- M: an $C \times C$ matrix where each entry M_{ij} shows the number of instances that the first classifier assigns to class i while the second classifier assigns to class j.
- n: number of instances in test data
- C: the number of classes

Procedure:

1. Calculate p_1, the relative observed agreement between two classifiers:

$$p_1 = \frac{\sum_{i=1}^{C} M_{ii}}{n}$$

2. Calculate p_2, the probability of reaching agreement between two classifiers by chance, given the observed counts:

$$p_2 = \sum_{i=1}^{C} \left(\sum_{j=1}^{C} \frac{M_{ij}}{n}\right) (\sum_{j=1}^{C} \frac{M_{ji}}{n})$$

3. Compute kappa statistic κ, the measure of agreement between two classifiers:

$$\kappa = \frac{p_1 - p_2}{1 - p_2}$$

Figure 10: Algorithm of computing kappa statistic κ.

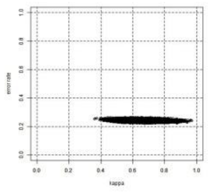
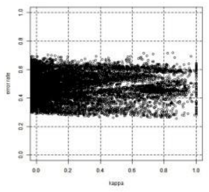
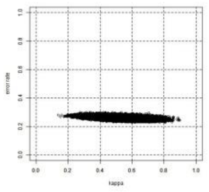
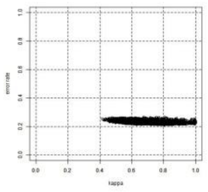
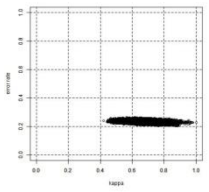
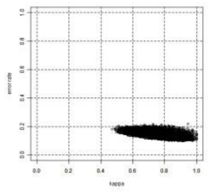
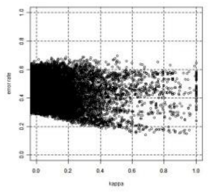
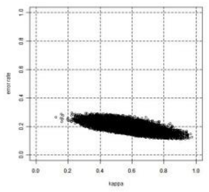
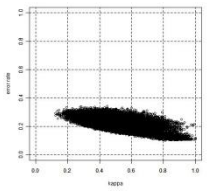
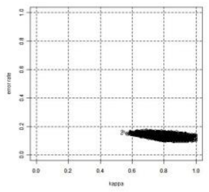
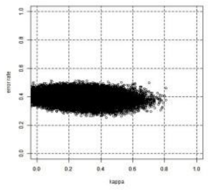
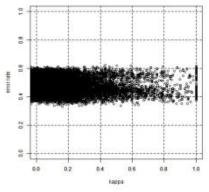
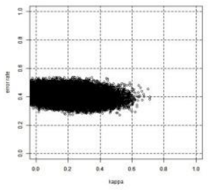
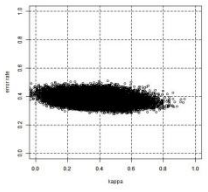
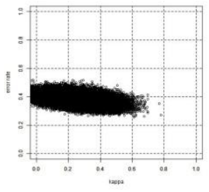
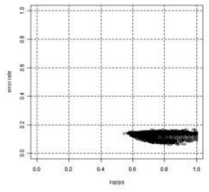
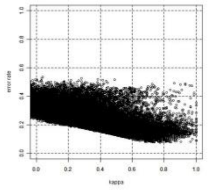
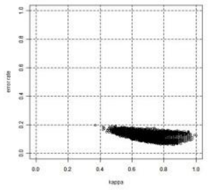
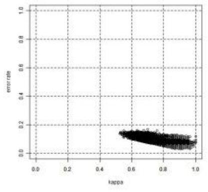
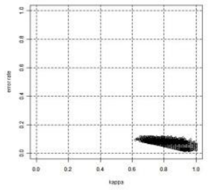
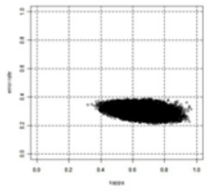
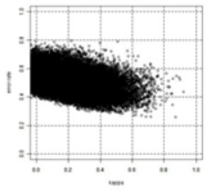
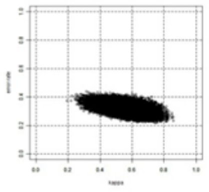
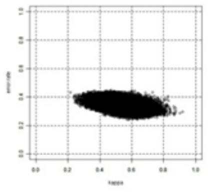
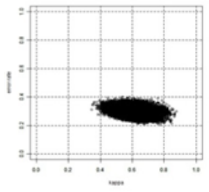
To clearly see the pattern of base classifiers in an ensemble classifier, Margineantu and Dietterich (1997) suggested using κ-error diagram to visualize the diversity and accuracy of base classifiers in an ensemble classifier. Since κ-error diagram is an often used tool to investigate the diversity of ensemble methods (Dietterich, 2000; Kuncheva and Whitaker, 2003; Rodríguez et al., 2006), here we also chose to use κ-error diagram to explore the behavior of base classifiers on diversity and accuracy in each ensemble classifier. Figure 11 shows the κ-error diagrams for all the 29 data sets. For each κ-error diagram, the x-axis indicates the kappa statistic κ and the y-axis indicates the averaged error of L_i *and* L_j, denoted as $E_{i,j} = \frac{E_i + E_j}{2}$, where E_i and E_j are the error rates of L_i *and* L_j respectively. Since small values of κ indicate that the classifiers are more diverse and small values of $E_{i,j}$ indicate high accuracy, the ideal location of dots is in the lower left corner. For each ensemble method, five hundred base classifiers were fitted using training data, then the result of the test data were used to calculate κ and the error rates for each given data set.

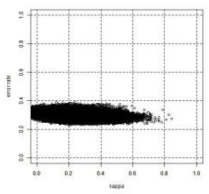
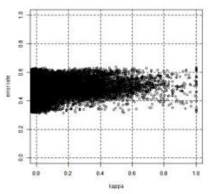
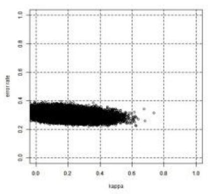
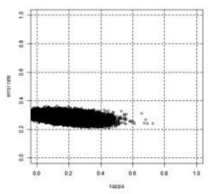
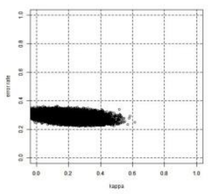
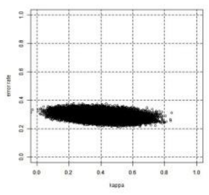
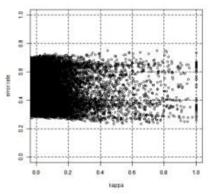
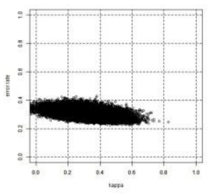
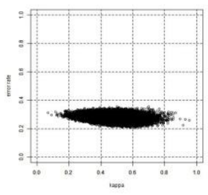
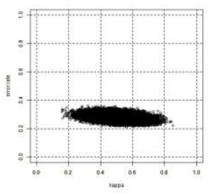
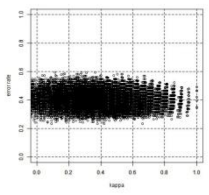
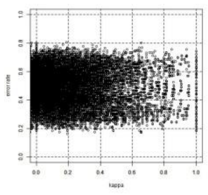
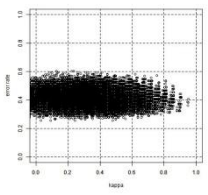
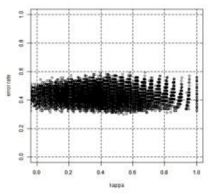
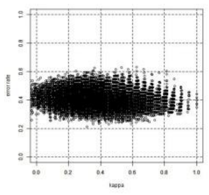
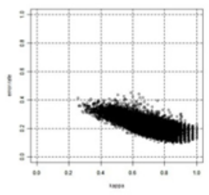
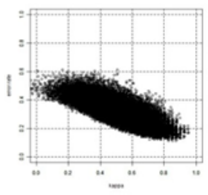
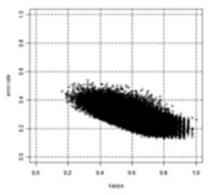
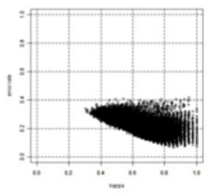
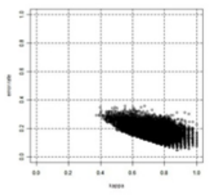
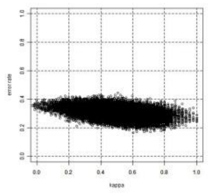
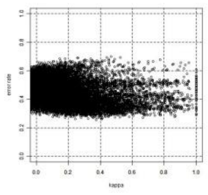
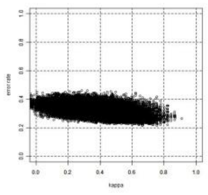
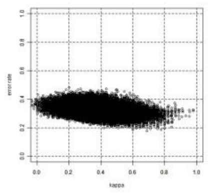
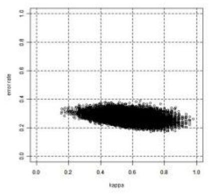
By using the κ-error diagram, we can clearly see the relative position of the ensemble methods for each data set. To help read Figure 11, take data set aus as an example. Samme has the largest diversity but also has the highest error rate. The reason why Samme is very different from the other ensemble methods is because it is a weighted ensemble method. We can see that Canonical Forest has the lowest error rate (the position of Canonical Forest is lower on the y-axis) but slightly less diverse (the position of Canonical Forest is farther on the x-axis) than the other four ensemble methods.

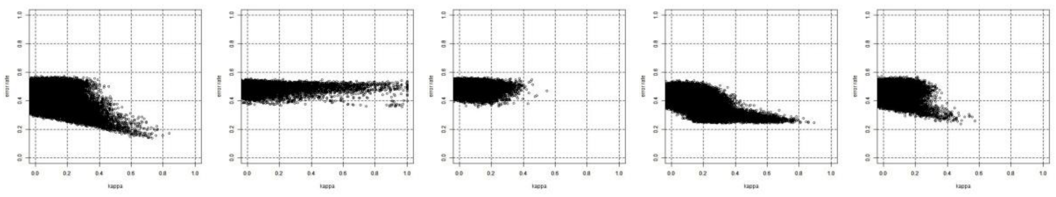
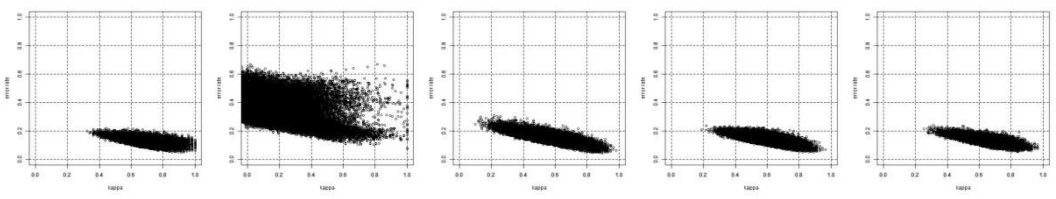
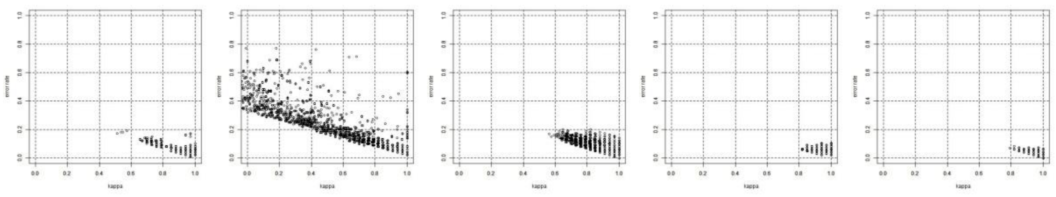
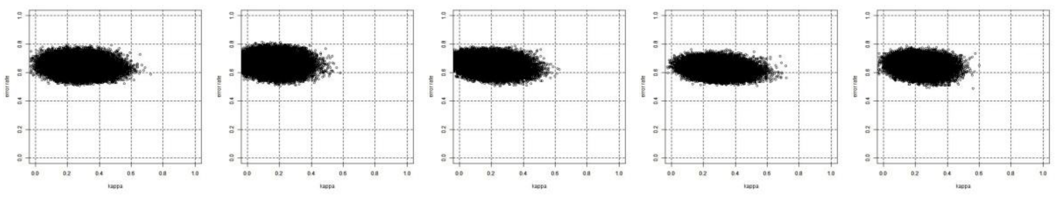
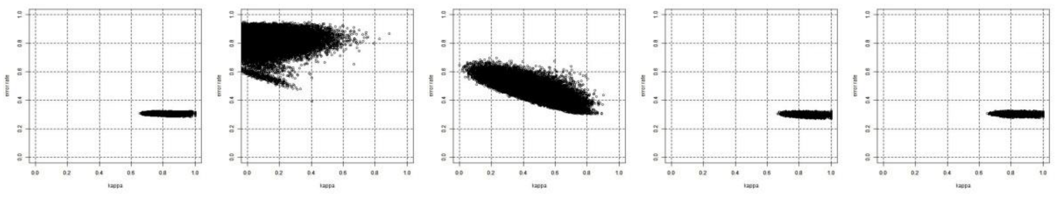
In general, the position of Canonical Forest is always lower on the y-axis which means it has higher accuracy and is quite similar to Bagging and Rotation Forest on the x-axis which means it has comparable diversity compared to Bagging and Rotation Forest.

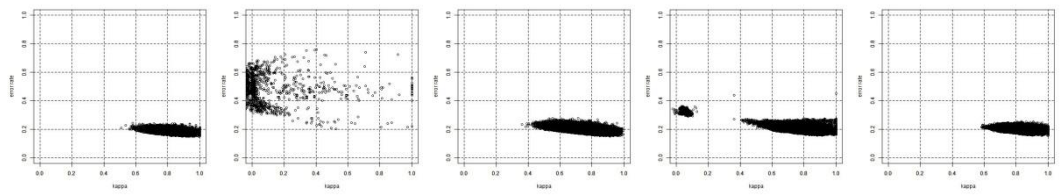
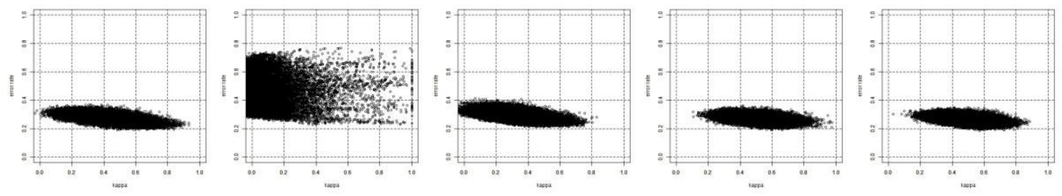
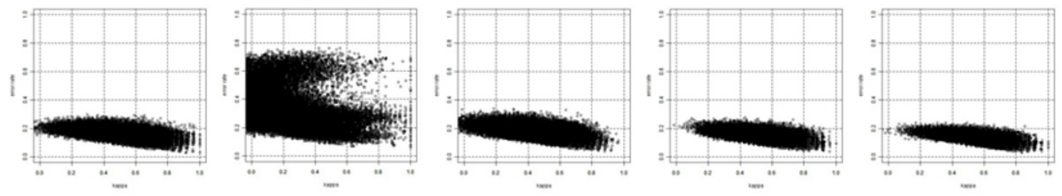
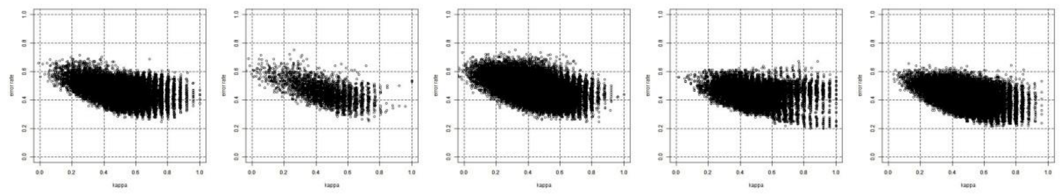
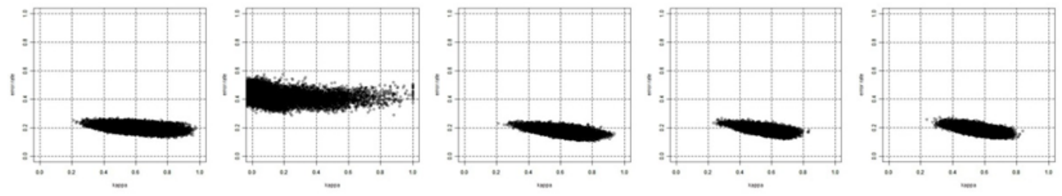
To see the relative positions of each ensemble method clearly, we put all the ensemble methods together in just one plot for each data set by constructing the contour plot of κ-error diagram. Figure 12 shows contour plots of κ-error diagrams for all the 29 data sets. Here we used contour plots so that it is easier to distinguish all of them and hence make the differences look clearer. The x-axis indicates κ and the y-axis indicates the error rate of classification. Taking data set 'fis' for example, Canonical Forest is on the lower right corner due to its lower error rate and less diversity than the other ensemble methods; Samme is on the upper left corner due to its higher error rate and higher diversity.

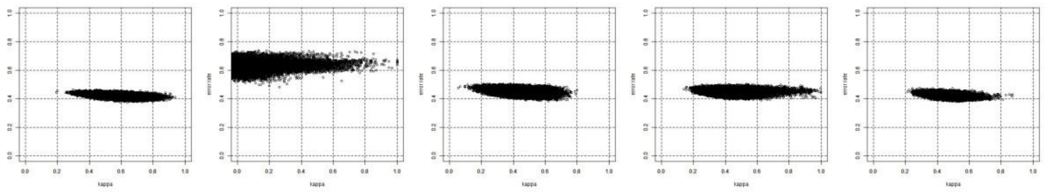
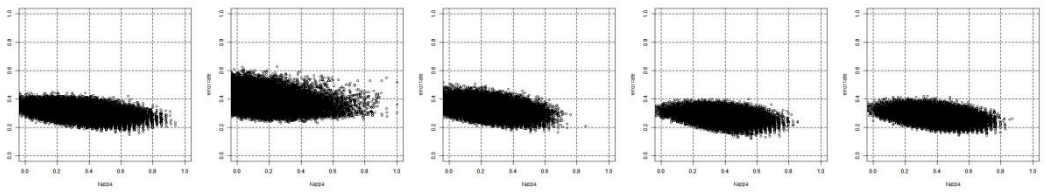
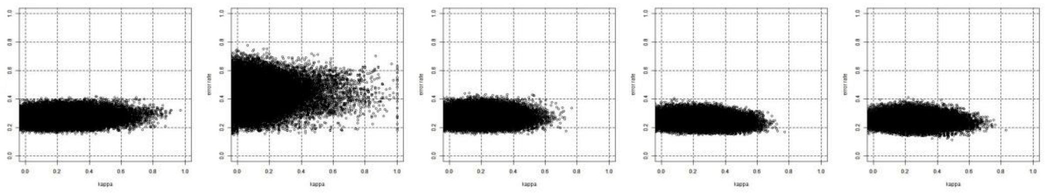
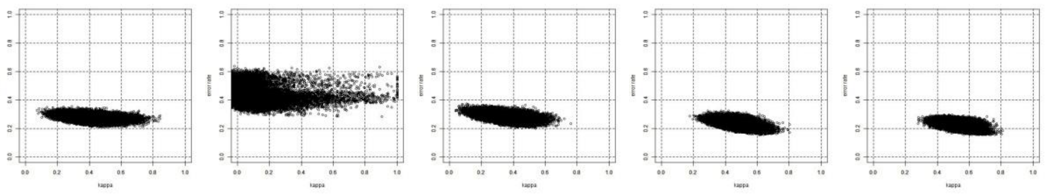
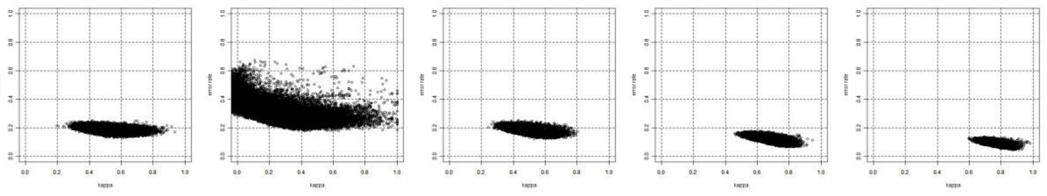
In general, Samme is not shown in the most of these plots because of the very high error rate and diversity. Canonical Forest appears to outperform the other ensemble methods in terms of accuracy while the diversity is quite similar to that of Bagging and Rotation Forest and less diverse than Samme and Random Forest. This suggests that if we can increase the diversity of Canonical Forest, it could be a more powerful ensemble method.

	Bagging	Samme	Random Forest	Rotation Forest	Canonical Forest
aba					
aus					
bld					
bod					
bos					

cir	    
dia	    
ech	    
fis	    
hea	    

int	
ion	
iri	
lak	
led	

mam	
pid	
pks	
pov	
rng	

sea	
snr	
spe	
trn	
twn	

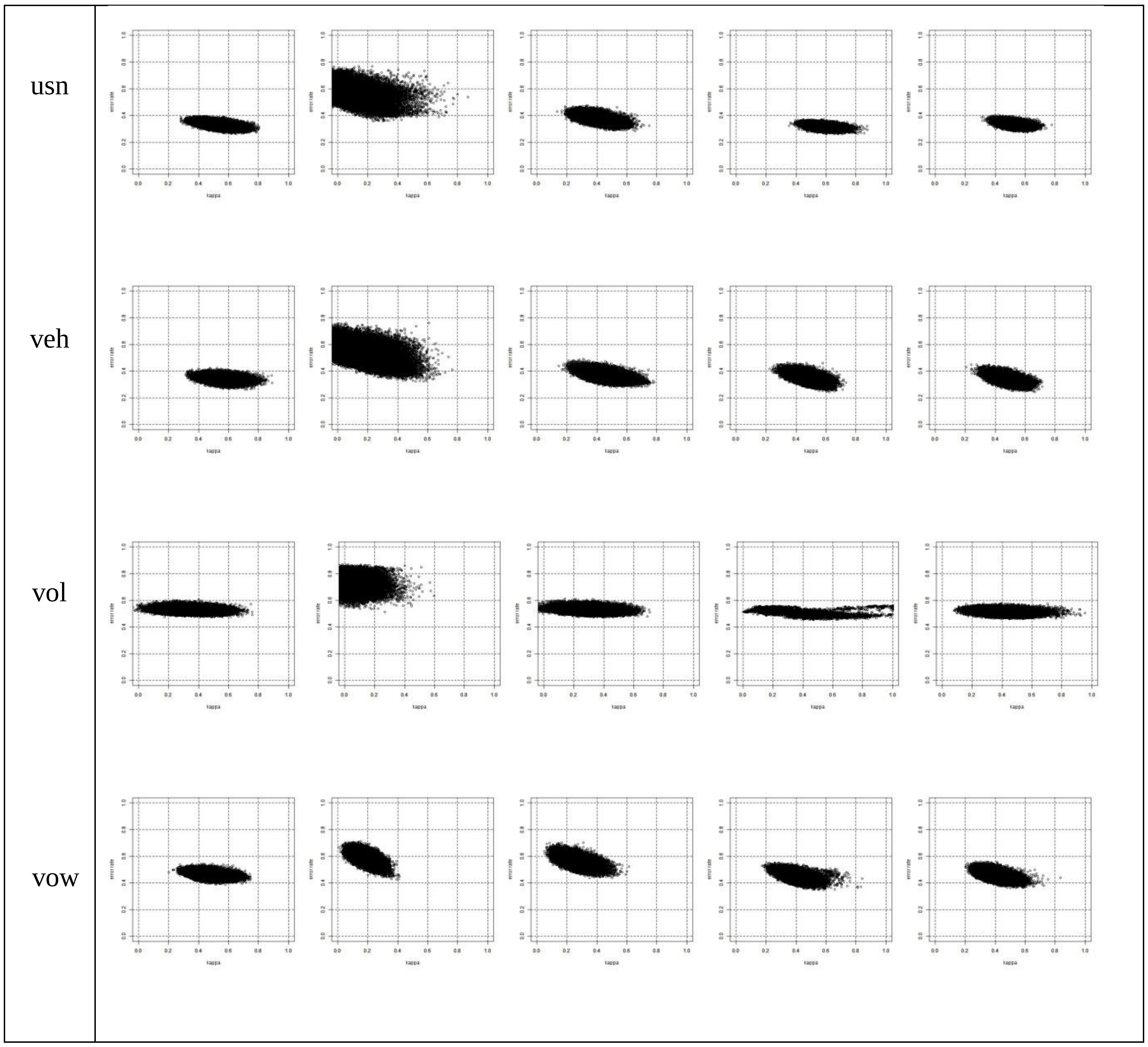
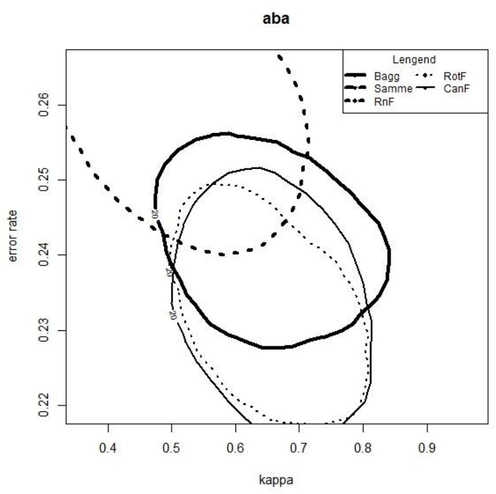
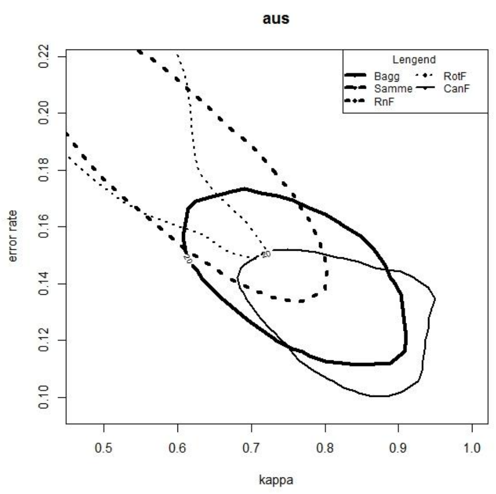


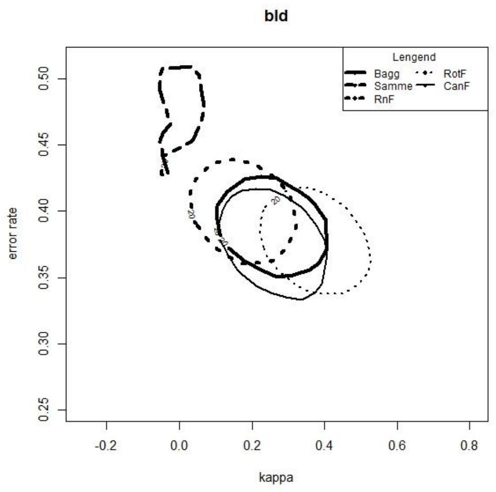
Figure 11: Kappa error diagram. x-axis = κ, y-axis = $E_{i,j}$ (average error of the pair of classifiers).



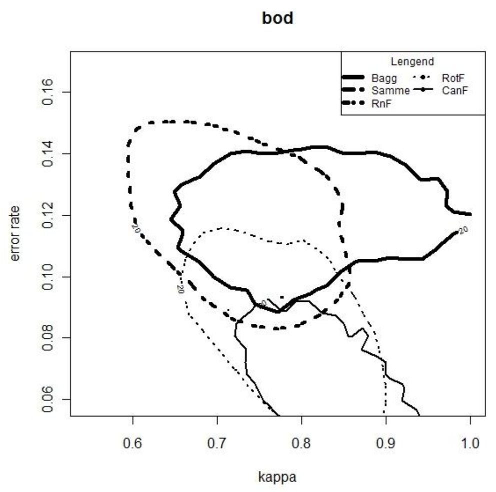
aba
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20



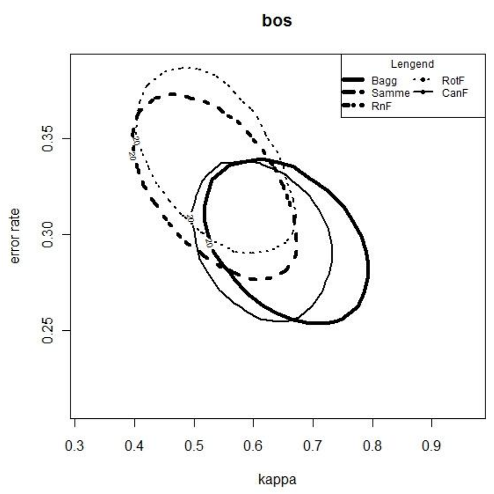
aus
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

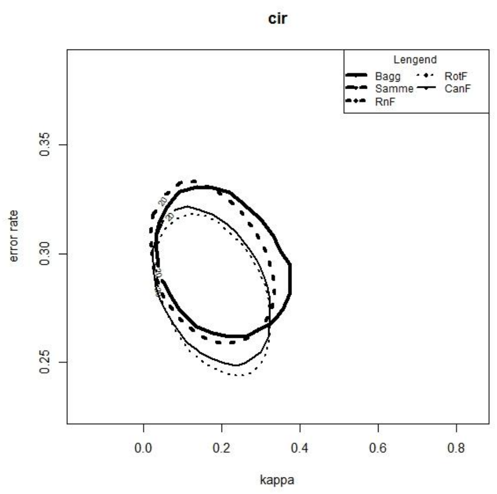


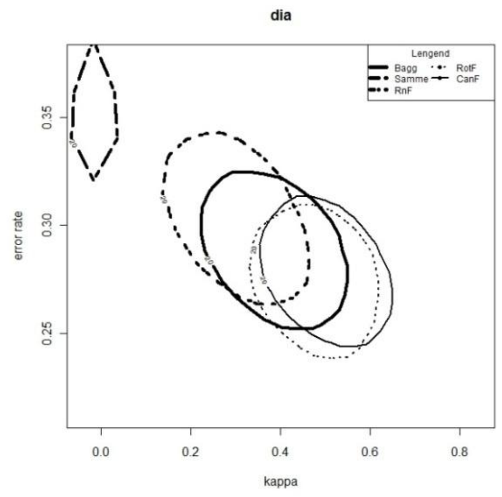
bld
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

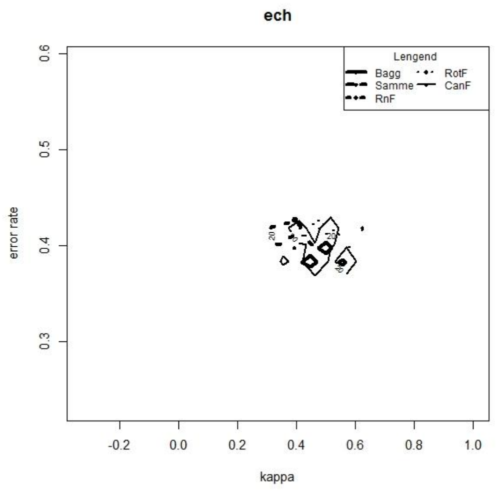


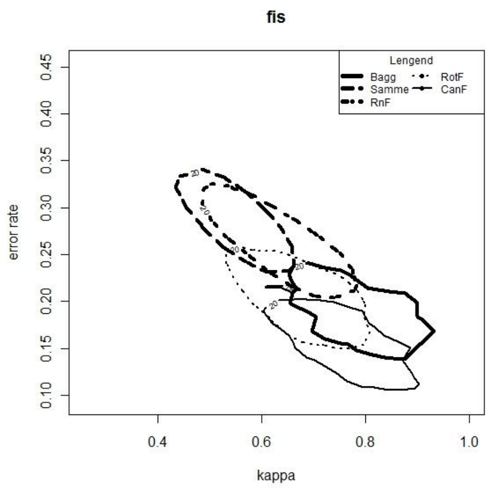
bod
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

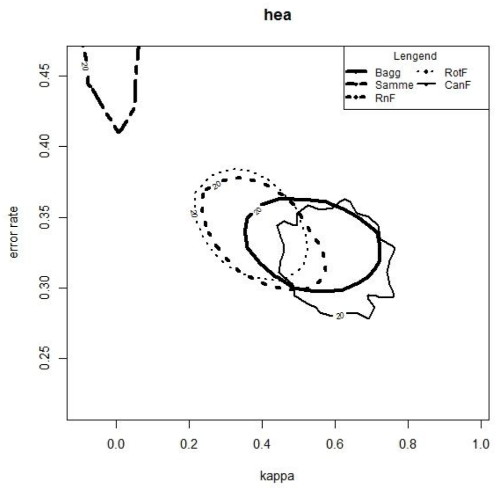

bos
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
0.25
0.30
0.35
kappa
0.3
0.4
0.5
0.6
0.7
0.8
0.9

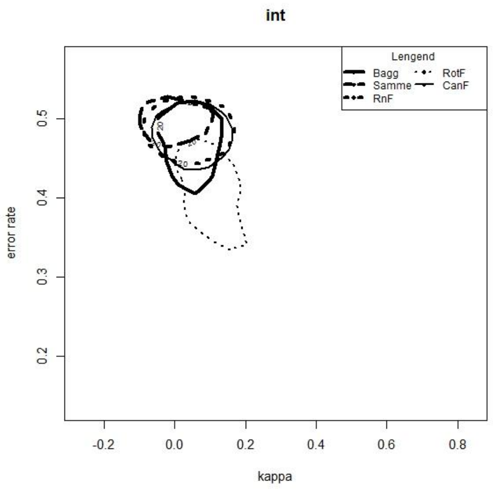

cir
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
0.25
0.30
0.35
kappa
0.0
0.2
0.4
0.6
0.8

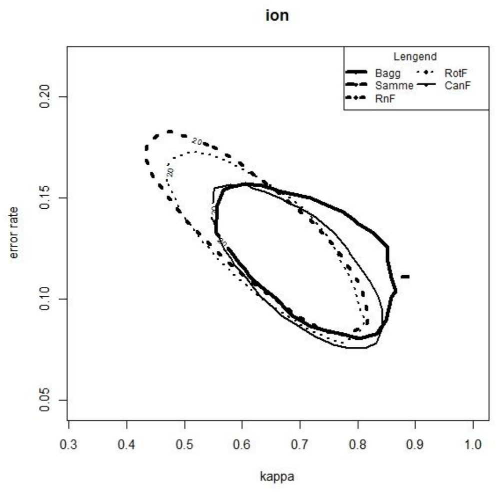

dia
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
0.25
0.30
0.35
kappa
0.0
0.2
0.4
0.6
0.8

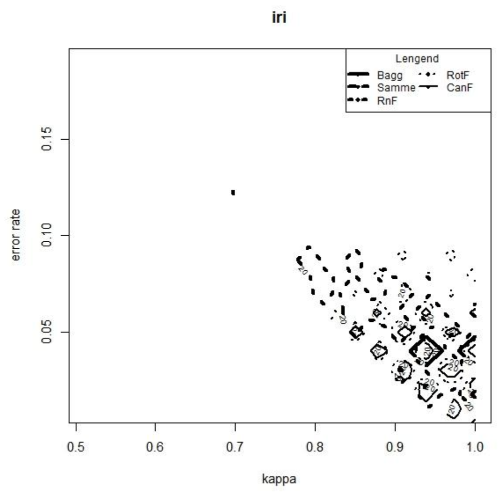

ech
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
0.3
0.4
0.5
0.6
kappa
-0.2
0.0
0.2
0.4
0.6
0.8
1.0

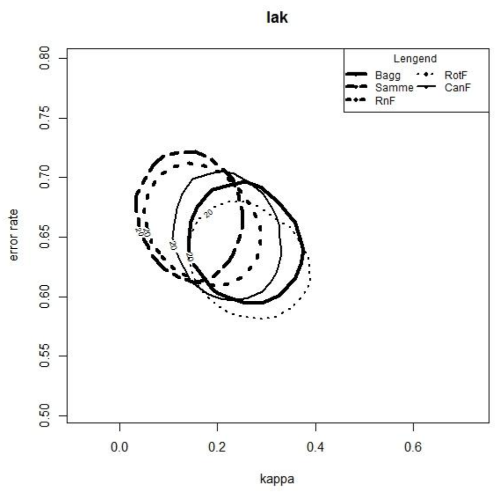

fis
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

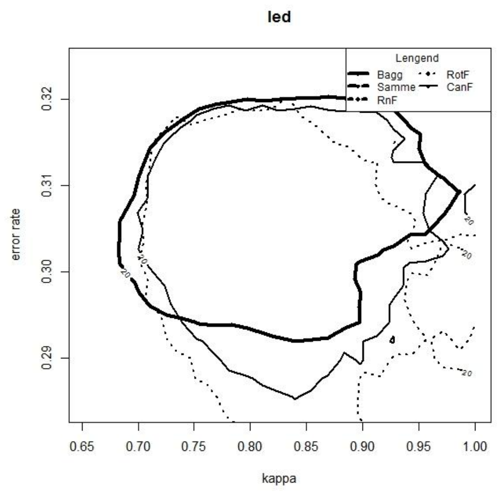

hea
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

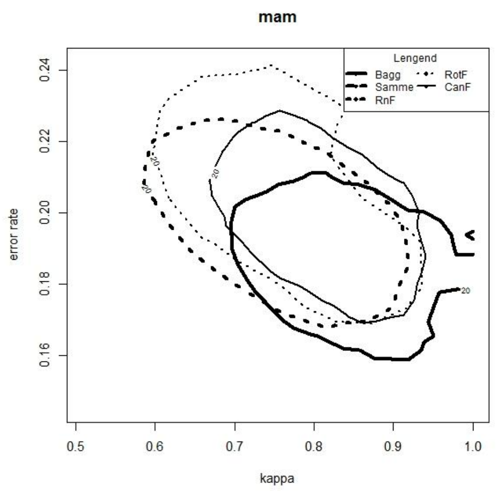

int
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

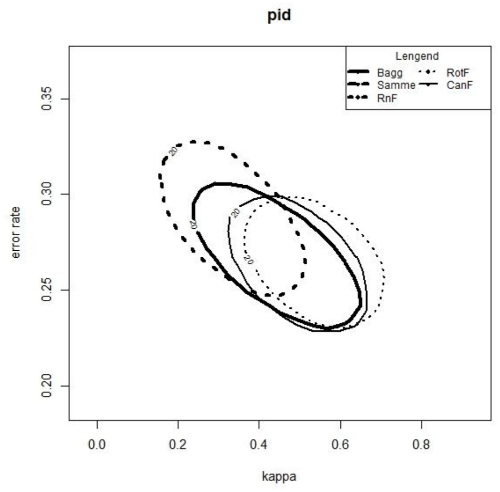

ion
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

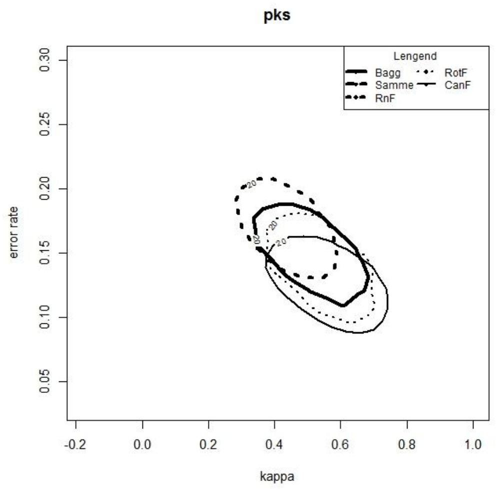

iri
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

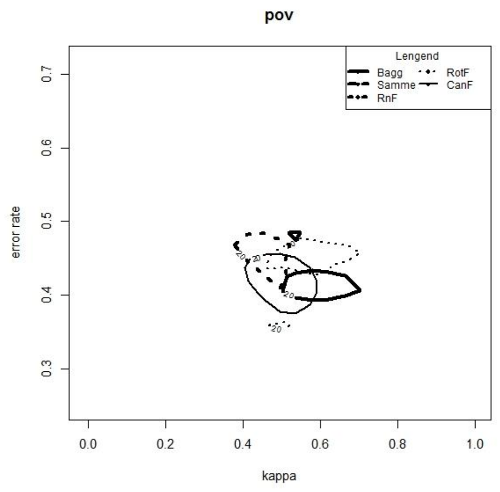

lak
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

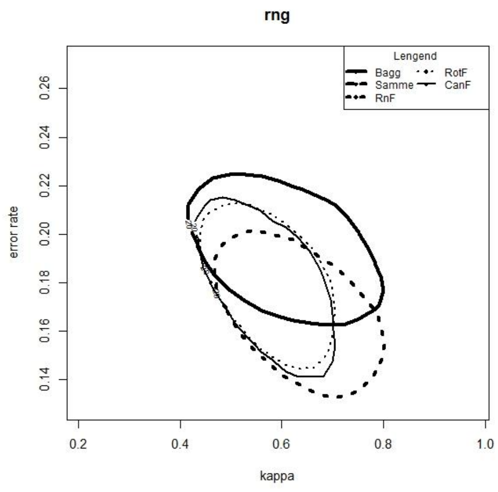

led
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

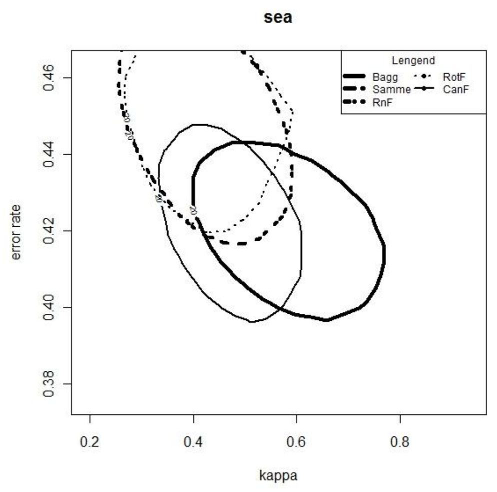

mam
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

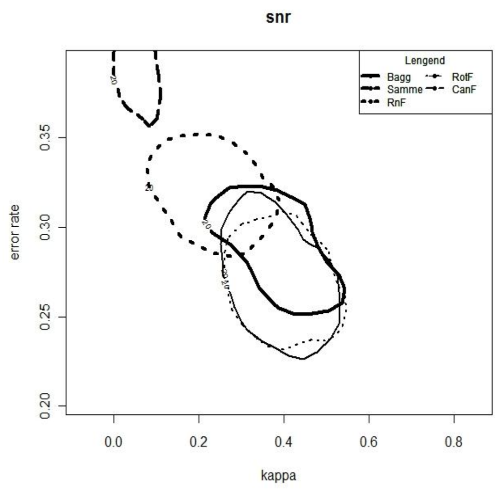

pid
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

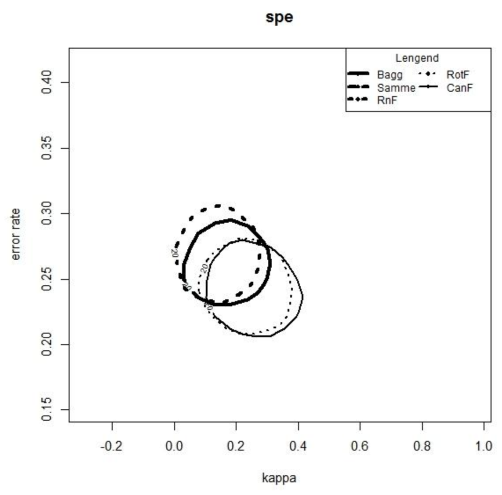

pks
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

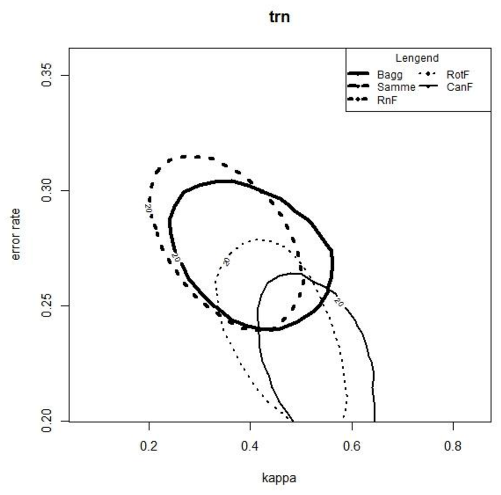

pov
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

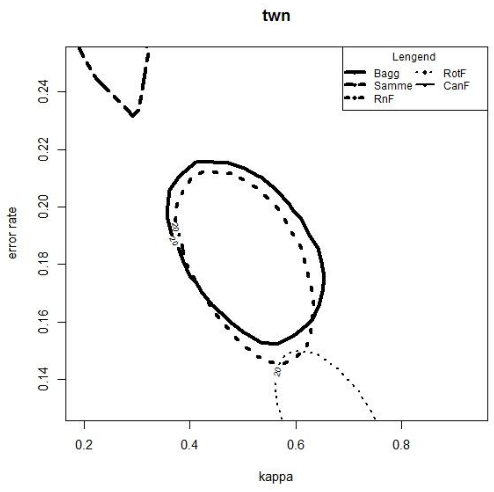

rng
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

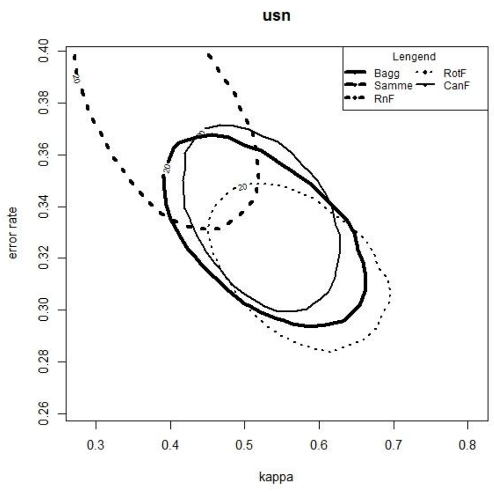

sea
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
0.38
0.40
0.42
0.44
0.46
0.2
0.4
0.6
0.8

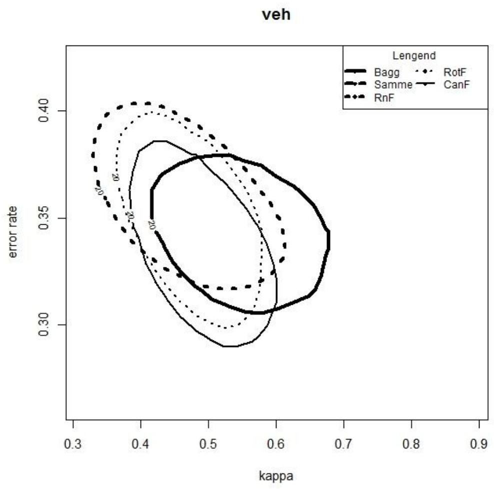

snr
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
0.20
0.25
0.30
0.35
0.0
0.2
0.4
0.6
0.8

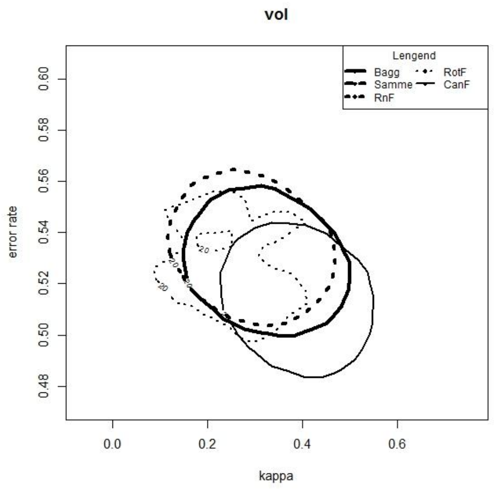

spe
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
0.15
0.20
0.25
0.30
0.35
0.40
-0.2
0.0
0.2
0.4
0.6
0.8
1.0


trn
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
0.20
0.25
0.30
0.35
0.2
0.4
0.6
0.8


twn
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20


usn
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20


veh
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20


vol
Lengend
Bagg
Samme
RnF
RotF
CanF
error rate
kappa
20

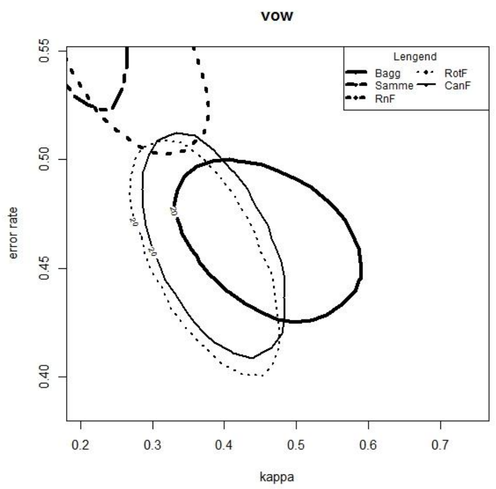


Figure 12: Contour plot of κ-error diagrams for the five ensemble methods on the same plot for all the 29 data sets.

4.3 Bias and Variance Decomposition

The bias and variance decomposition of the error (Geman et al., 1992) is a popular and useful approach. The bias measures the distance between the classifier and the target function and the variance measures variation among the predictions from different classifiers. Several authors including Kong and Dietterich (1995), Kohavi and Wolpert (1996), and Breiman (1998) have proposed different methods for the bias and variance decomposition. In this dissertation we used the decomposition proposed by Kohavi and Wolpert (1996). The same data sets as described in Table 1 (except data set 'aus' due to computational problem) were used to analyze the bias and variance decomposition. Here we used 3-fold cross-validation for each data. First we randomly split each data set into three parts: subset 1, subset 2 and subset 3. Then we used subset 1 as a test set and used subsets 2 and 3 for generating a training set. Next we randomly selected 3/4 from subsets 2 and 3 to be training set and repeated this process 20 times. To estimate the bias and variance properly, the unbiased estimators were used. After we analyzed the bias and variance decomposition using subset 1, the same logic is applied to subset 2 and subset 3. An ensemble of size 500 is used for each ensemble method.

Table 9 shows the comparison of bias^2. To help read Win – Loss in Table 9, we take Samme as an example. The Win – Loss is 21 – 7 for Samme when compared to Canonical Forest. This means that bias of Samme is smaller than the bias of Canonical Forest for 21 data sets and larger than the bias of Canonical Forest for 7 data sets. Samme appears to be the best among all the ensemble methods considered here in reducing bias while Canonical Forest is in the middle. Table 10 compares the variance. It shows that Canonical Forest dominates all the other methods in reducing the variance. In Tables 9 and 10, the p-values were obtained by performing Wilcoxon signed-rank test. We found that the high accuracy of Canonical Forest shown in the previous section was mainly due to the variance reduction.

Table 9: Comparison of contribution of Bias^2 to error.

Data Set	Tree	Bagging	Adaboost	Samme	Random Forest	Rotation Forest	Canonical Forest
aba	0.1856	0.1990	0.2272	0.1828	0.2039	0.1998	0.2047
bld	0.2146	0.2480	0.2574	0.2231	0.2255	0.2722	0.2247
bod	0.0632	0.0636	0.0236	0.0226	0.0530	0.0376	0.0188
bos	0.1766	0.1892	0.1838	0.1672	0.1841	0.2021	0.1984
cir	0.1713	0.1747	0.2374	0.0534	0.1771	0.2143	0.2115
dia	0.1769	0.2045	0.2188	0.2039	0.2085	0.2142	0.2100
ech	0.2021	0.2302	0.2154	0.2203	0.2373	0.2753	0.2358
fis	0.1304	0.1310	0.1304	0.1163	0.1398	0.0563	0.0387
hea	0.2059	0.2245	0.2223	0.2494	0.2187	0.2440	0.1988
int	0.2087	0.2943	0.3703	0.2538	0.3573	0.2255	0.3419
ion	0.0869	0.0668	0.0634	0.0672	0.0669	0.0517	0.0573
iri	0.0376	0.0386	0.0417	0.0441	0.0405	0.0389	0.0257
lak	0.4080	0.4586	0.4741	0.4623	0.4436	0.4452	0.4705
led	0.2550	0.2572	0.2406	0.2865	0.2406	0.2467	0.2522
mam	0.1615	0.1651	0.1547	0.1620	0.1646	0.1675	0.1651
pid	0.1635	0.1921	0.1995	0.1953	0.1925	0.2029	0.2014
pks	0.0925	0.1091	0.1103	0.0542	0.0930	0.0784	0.0928
pov	0.2337	0.2703	0.2569	0.2420	0.2889	0.2810	0.2639
rng	0.1138	0.0982	0.2079	0.0520	0.0862	0.0794	0.0824
sea	0.3297	0.3593	0.3842	0.2413	0.3475	0.3558	0.3383
snr	0.1574	0.1775	0.1312	0.1152	0.1527	0.1270	0.1419
spe	0.1456	0.1588	0.1567	0.1528	0.1562	0.1473	0.1483
trn	0.1377	0.1251	0.1233	0.1131	0.1163	0.1186	0.1380
twn	0.0868	0.0464	0.0271	0.0278	0.0343	0.0275	0.0305
usn	0.2155	0.2443	0.2339	0.2244	0.2457	0.2547	0.2529
veh	0.2270	0.2391	0.2176	0.1747	0.2212	0.2130	0.2054
vol	0.3586	0.3968	0.4017	0.3438	0.4013	0.3777	0.3943
vow	0.2916	0.2909	0.0822	0.0475	0.1796	0.1736	0.1971
Win - Loss	17 - 11	10 - 18	11 - 17	21 - 7	11 - 17	13 - 15	
p-value	0.7674	0.0282	0.0628	0.9936	0.1751	0.4070	

Table 10: Comparison of contribution of Variance to error.

Data Set	Tree	Bagging	Adaboost	Samme	Random Forest	Rotation Forest	Canonical Forest
aba	0.0604	0.0266	0.0393	0.0412	0.0239	0.0180	0.0143
bld	0.1579	0.0736	0.0688	0.1019	0.0809	0.0549	0.0580
bod	0.0453	0.0159	0.0093	0.0089	0.0086	0.0064	0.0058
bos	0.1003	0.0438	0.0433	0.0579	0.0356	0.0436	0.0373
cir	0.1251	0.0444	0.1073	0.0446	0.0352	0.0240	0.0257
dia	0.0867	0.0349	0.0248	0.0776	0.0302	0.0301	0.0272
ech	0.1535	0.0670	0.0936	0.1026	0.0600	0.0665	0.0593
fis	0.0720	0.0404	0.0587	0.1108	0.0397	0.0249	0.0142
hea	0.1017	0.0482	0.0380	0.0725	0.0411	0.0484	0.0504
int	0.2144	0.1303	0.1225	0.1394	0.1297	0.1049	0.1223
ion	0.0493	0.0225	0.0209	0.0201	0.0075	0.0067	0.0123
iri	0.0161	0.0110	0.0113	0.0221	0.0098	0.0100	0.0079
lak	0.2413	0.1418	0.1492	0.1524	0.1412	0.1634	0.1423
led	0.0605	0.0377	0.0261	0.1438	0.0330	0.0329	0.0407
mam	0.0157	0.0093	0.0155	0.0356	0.0119	0.0181	0.0154
pid	0.0893	0.0357	0.0275	0.0750	0.0321	0.0277	0.0296
pks	0.0722	0.0330	0.0326	0.0286	0.0257	0.0344	0.0258
pov	0.1652	0.1172	0.1071	0.1775	0.0952	0.1168	0.0977
rng	0.0841	0.0375	0.0589	0.0294	0.0241	0.0196	0.0202
sea	0.0968	0.0457	0.0744	0.1060	0.0429	0.0532	0.0394
snr	0.1403	0.0579	0.0596	0.0622	0.0486	0.0455	0.0399
spe	0.1078	0.0302	0.0302	0.0518	0.0217	0.0328	0.0335
trn	0.1328	0.0492	0.0315	0.0469	0.0319	0.0181	0.0173
twn	0.1035	0.0296	0.0117	0.0147	0.0113	0.0038	0.0042
usn	0.1247	0.0485	0.0401	0.0730	0.0376	0.0473	0.0336
veh	0.1317	0.0694	0.0640	0.0830	0.0586	0.0533	0.0469
vol	0.1427	0.0748	0.0652	0.1534	0.0700	0.1227	0.0646
vow	0.1876	0.0929	0.0546	0.0384	0.0908	0.0730	0.0841
Win - Loss	0 - 28	5 – 23	6 - 22	1 - 27	9 - 19	11 - 17	
p-value	3.73×10^{-09}	1.00×10^{-06}	0.0012	2.00×10^{-06}	0.0116	0.0314	

4.4 High-Dimensional Canonical Forest (HDCF)

We also conducted an experiment using data sets on gene imprinting, estrogen and leukemia to compare HDCF with Random Forest, CERP, and SVM which are popular and successful classification methods on high dimensional data sets. The brief descriptions for these three data sets excerpted from Ahn et al. (2007) are provided below.

- **Identification of Imprinted Genes (Greally, 2002):** The data contain 131 samples and 1446 features. Among these 131 samples, 43 are imprinted and 88 are non-imprinted. The available data set made by John Greally can be found at http://greallylab.aecom.yu.edu/~greally/imprinting_data.txt.
- **Classification of Chemicals for Estrogen Activity (Blair et al., 2000):** The data contain 232 samples and 312 features. Among these 232 samples, 131 chemicals show estrogen receptor binding activity and 101 are inactive in a competitive estrogen receptor binding assay (Blair et al., 2000). The available data set can be found at http://www.ams.sunysb.edu/~hahn/research/CERP/estrogen.txt.
- **Classification of Leukemia Subtypes (Dudoit et al., 2002):** The data contain 72 samples and 3571 features. Among these 72 samples, 47 are

classified as acute myeoloid leukemia and 25 are classified as acute lymphoblastic leukemia. The available data set can be found at http://www.ams.sunysb.edu/~hahn/research/CERP/leukemia.txt.

We evaluated the performance by comparing their classification accuracy, area under the curve (AUC), and also the balance between sensitivity and specificity. Decision trees were used as the base classifiers for all three ensemble methods: Random Forest, CERP, and HDCF. For both Random Forest and Canonical Forest, the ensemble sizes were fixed at $B = 500$. In HDCF, the number of features in each subset is set to be $m = 3$. If p is not divisible by 3, then the last subset was completed with the remaining features (1 or 2 features). Besides, the number of features selected from the feature set is set to be $q = 3p^{1/2}$ in HDCF. For each data set, twenty repetitions of 10-fold cross-validation were performed.

The results of classification accuracy are marked with a bullet next to them if HDCF is significantly better than the compared method. Since HDCF is not significantly worse than any methods for any data sets, we don't have to use a symbol to show it. The comparison was done by using paired t-test at two-sided significant level of $\alpha = 0.05$.

Table 11 shows the classification accuracy along with sensitivity and specificity on gene imprinting data set. For gene imprinting data set, HDCF has the highest classification accuracy among all the methods and is significantly better than all other three methods. Although HDCF shows some imbalance between sensitivity and specificity, that's because the data itself is quite unbalanced (the proportion of the majority and minority groups in responses in the data is only 0.33). Besides, HDCF is still the most balanced one compared to other methods and the sensitivity of HDCF (0.6628) is much larger than that of SVM (0.4767).

For the estrogen data set (see Table 12), although HDCF is only significantly more accurate than SVM, it has the second highest classification accuracy and is only slightly less than CERP that has the highest classification accuracy. Because the data are quite balanced (the proportion of the responses in the data is 0.56), all the methods show good balance between sensitivity and specificity. It is worth mentioning that the balance between sensitivity and specificity in HDCF was slightly better than all the other methods.

For leukemia data set (see Table 13), basically all the methods have pretty high classification accuracies. Therefore, it's not surprised to find that they all have good balance between sensitivity and specificity due to such high classification accuracy. Nevertheless, HDCF still has the highest classification accuracy and a better balance among all these methods.

In general, it seems that Canonical Forest gives consistently better performance in terms of classification accuracy and balance between sensitivity and specificity among all these methods for these three data sets.

Table 11: Accuracy of classification methods for the imprinting data.

Method	ACC	p-value[1]	AUC	Sensitivity	Specificity
CERP	0.8653● (0.0137)	1.35×10^{-07}	0.7974 (0.0197)	0.6000 (0.0389)	0.9949 (0.0078)
SVM	0.7893● (0.0151)	1.18×10^{-15}	0.7094 (0.0210)	0.4767 (0.0437)	0.9420 (0.0156)
RF	0.8748● (0.0125)	2.05×10^{-05}	0.8167 (0.0189)	0.6477 (0.0408)	0.9858 (0.0110)
HDCF	0.8866 (0.0087)	-	0.8294 (0.0130)	0.6628 (0.0267)	0.9960 (0.0056)

● HDCF is significantly better than the compared method

[1] p-value was obtained by comparing the accuracy of HDCF with the accuracy of the compared method using two-sided paired t-test.

Table 12: Accuracy of classification methods for the estrogen data.

Method	ACC	p-value[1]	AUC	Sensitivity	Specificity
CERP	0.8494 (0.0109)	0.1508	0.8424 (0.0115)	0.8962 (0.0100)	0.7886 (0.0180)
SVM	0.8289● (0.0098)	9.83×10^{-05}	0.8197 (0.0096)	0.8908 (0.0127)	0.7485 (0.0104)
RF	0.8394 (0.0125)	0.1396	0.8341 (0.0137)	0.8752 (0.0090)	0.7931 (0.0242)
HDCF	0.8448 (0.0146)	-	0.8392 (0.0144)	0.8824 (0.0186)	0.7960 (0.0162)

● HDCF is significantly better than the compared method

[1] p-value was obtained by comparing the accuracy of HDCF with the accuracy of the compared method using two-sided paired t-test.

Table 13: Accuracy of classification methods for the leukemia data.

Method	ACC	p-value[1]	AUC	Sensitivity	Specificity
CERP	0.9785 (0.0084)	0.1864	0.9690 (0.0121)	0.9380 (0.0242)	1 (0)
SVM	0.9771 (0.0093)	0.0724	0.9670 (0.0134)	0.9340 (0.0268)	1 (0)
RF	0.9819 (0.0079)	0.6663	0.9745 (0.0101)	0.9500 (0.0178)	0.9990 (0.0048)
HDCF	0.9826 (0.0089)	-	0.9759 (0.0111)	0.9540 (0.0196)	0.9979 (0.0065)

[1] p-value was obtained by comparing the accuracy of HDCF with the accuracy of the compared method using two-sided paired t-test.

4.5 Simulation Study

We also conducted a simulation study to evaluate the performance of HDCF in comparison with Random Forest, CERP, and SVM using high dimensional simulated data. Here we adopted the simulation design that was generated by Lee et al. (2013). We generated two data sets with 120 instances and 500 independent features. One served as training set and another one served as test set. The number of instances assigned to each class is the same which is 40 for each class. To fairly evaluate the performance of each classification method, we generated one hundred pairs of these training and test sets and then took the average of the performance from these 100 testing sets. To generate 500 features in each data set, we first generated fifty features using three different normal distributions, and then generated the remaining 450 features from one normal distribution to serve as noise. Figure 13 shows the simulation design. Three data sets, A, B, and C were generated from the standard deviation σ equal to 1, 2, and 3, respectively. For each of 100 pairs of training and test sets in each data set, the classifier was built using the training set and then evaluated using the test set. The accuracy averaged from the 100 test sets for each classification method from each data set is provided in Tables 14 through 16.

As expected, we can see that the classification accuracy decreases as the standard deviation increases. When standard deviation equals 1, all the methods perform very well. The performance of HDCF is better than all the other three methods and is significantly better than CERP and SVM. As the standard deviation increases to 2 and 3, the performances of these four methods drop dramatically. Nevertheless, HDCF still has the highest classification accuracy among all these methods and also performs significantly better than all the other methods. It is clear that HDCF outperforms all the other three methods in our simulation study especially when the standard deviation becomes larger. It is interesting that SVM has higher classification accuracy than Random Forest when the standard deviation goes up to 3 while it is less accurate than Random Forest when the standard deviations equal 1 and 2.

Table 14: Accuracy of classification methods for data set A.

Method	ACC	p-value[1]	AUC	Sensitivity	Specificity
CERP	0.9503● (0.0223)	1.96×10^{-25}	0.9628 (0.0337)	Class 1 0.9768 (0.0262) Class 2 0.8760 (0.0656) Class 3 0.9985 (0.0060)	Class 1 0.9474 (0.0304) Class 2 0.9876 (0.0134) Class 3 0.9905 (0.0102)
SVM	0.9703● (0.0153)	5.12×10^{-08}	0.9778 (0.0205)	Class 1 0.9633 (0.0327) Class 2 0.9515 (0.0335) Class 3 0.9963 (0.0114)	Class 1 0.9779 (0.0155) Class 2 0.9798 (0.0166) Class 3 0.9979 (0.0053)
RF	0.9792 (0.0130)	0.0976	0.9844 (0.0163)	Class 1 0.9785 (0.0238) Class 2 0.9598 (0.0310) Class 3 0.9993 (0.0043)	Class 1 0.9820 (0.0148) Class 2 0.9889 (0.0119) Class 3 0.9979 (0.0050)
HDCF	0.9813 (0.0132)	-	0.9860 (0.0160)	Class 1 0.9845 (0.0218) Class 2 0.9595 (0.0305) Class 3 1 (0)	Class 1 0.9813 (0.0149) Class 2 0.9923 (0.0109) Class 3 0.9985 (0.0041)

● HDCF is significantly better than the compared method

[1] p-value was obtained by comparing the accuracy of HDCF with the accuracy of the compared method using two-sided paired t-test.

Table 15: Accuracy of classification methods for data set B.

Method	ACC	p-value[1]	AUC	Sensitivity	Specificity
CERP	0.7237● (0.0446)	2.10×10^{-29}	0.7928 (0.1002)	Class 1 0.7735 (0.0859) Class 2 0.5033 (0.0869) Class 3 0.8943 (0.0562)	Class 1 0.8309 (0.0436) Class 2 0.8529 (0.0500) Class 3 0.9018 (0.0398)
SVM	0.7656● (0.0370)	3.80×10^{-14}	0.8242 (0.0776)	Class 1 0.7508 (0.0808) Class 2 0.6735 (0.0888) Class 3 0.8725 (0.0519)	Class 1 0.8858 (0.0404) Class 2 0.8150 (0.0499) Class 3 0.9476 (0.0266)
RF	0.7753● (0.0397)	1.73×10^{-09}	0.8314 (0.0880)	Class 1 0.7925 (0.0672) Class 2 0.6068 (0.0944) Class 3 0.9265 (0.0500)	Class 1 0.8588 (0.0445) Class 2 0.8684 (0.0419) Class 3 0.9358 (0.0302)
HDCF	0.7983 (0.0373)	-	0.8487 (0.0891)	Class 1 0.8560 (0.0617) Class 2 0.5878 (0.0872) Class 3 0.9510 (0.0374)	Class 1 0.8454 (0.0421) Class 2 0.9090 (0.0354) Class 3 0.9430 (0.0271)

● HDCF is significantly better than the compared method

[1] p-value was obtained by comparing the accuracy of HDCF with the accuracy of the compared method using two-sided paired t-test.

Table 16: Accuracy of classification methods for data set C.

Method	ACC	p-value[1]	AUC	Sensitivity	Specificity
CERP	0.5502● (0.0468)	1.23×10^{-29}	0.6626 (0.0882)	Class 1 0.5828 (0.0892) Class 2 0.3973 (0.0957) Class 3 0.6705 (0.0809)	Class 1 0.7766 (0.0527) Class 2 0.7306 (0.0540) Class 3 0.8180 (0.0482)
SVM	0.6108● (0.0440)	3.36×10^{-06}	0.7081 (0.0895)	Class 1 0.6185 (0.0921) Class 2 0.5045 (0.0875) Class 3 0.7093 (0.0848)	Class 1 0.8220 (0.0544) Class 2 0.7099 (0.0640) Class 3 0.8843 (0.0397)
RF	0.5878● (0.0465)	8.39×10^{-16}	0.6908 (0.0890)	Class 1 0.6020 (0.0934) Class 2 0.4560 (0.0969) Class 3 0.7053 (0.0779)	Class 1 0.7981 (0.0518) Class 2 0.7320 (0.0585) Class 3 0.8515 (0.0429)
HDCF	0.6301 (0.0459)	-	0.7226 (0.1019)	Class 1 0.7005 (0.0817) Class 2 0.4148 (0.0878) Class 3 0.7750 (0.0791)	Class 1 0.7684 (0.0498) Class 2 0.7956 (0.0484) Class 3 0.8811 (0.0424)

● HDCF is significantly better than the compared method

[1] p-value was obtained by comparing the accuracy of HDCF with the accuracy of the compared method using two-sided paired t-test.

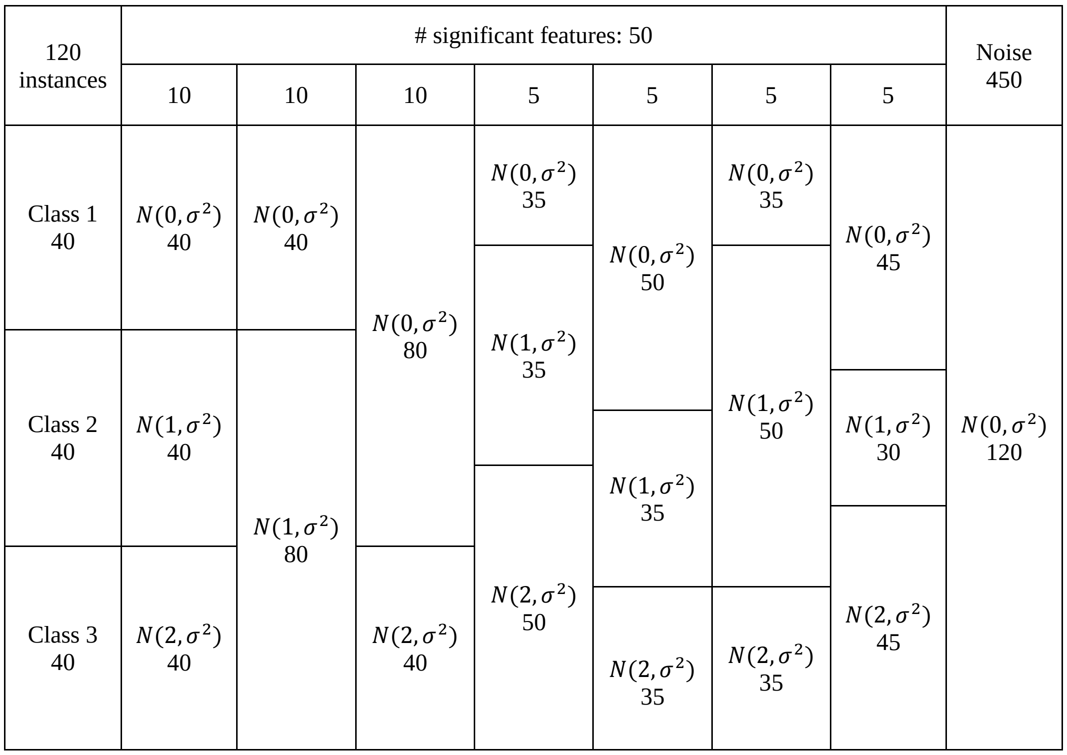


Figure 13: Simulation design.

Chapter 5

Conclusion and Discussion

We introduced a new ensemble method called Canonical Forest. It uses CLDA to perform a linear transformation on the original input data so that the transformed input data can be more distinct among classes. To enhance the diversity, the features are split into subsets, and then CLDA is applied on each subset separately. It should be noted that disjoint subsets would yield enhanced diversity because there is no overlap between subsets. However, the number of subsets is not necessarily related to the diversity of classifiers in Canonical Forest. A classifier is built using all the canonical components when applying CLDA. Here CLDA serves as a linear transformation tool rather than a dimension reduction tool since we keep all the features while applying CLDA. The reason why we chose CLDA is because it is a supervised learning method. When linearly transforming the data, CLDA utilizes the class information and makes the classes more separable after transformation. Therefore, this makes CLDA a better linear transformation tool than other dimension reduction methods when applied in classification analysis. Canonical Forest is formed by combining these classifiers with a majority vote.

Although both Double-Bagging and Canonical Forest are CLDA-based ensemble methods, they are different due to the fact that CLDA is applied to an OOB sample in Double-Bagging while CLDA is applied to a bootstrap sample in Canonical Forest. Besides, CLDA is applied to the whole feature space in Double-Bagging while CLDA is applied to each mutually exclusive feature subset in Canonical Forest.

Canonical Forest performed better in terms of accuracy than the other widely used ensemble methods especially when the ensemble size is small based on our experiment. Exact Binomial test showed the superiority of Canonical Forest over other ensemble methods. Besides, the performance of Canonical Forest is further improved by implementing WAVE. It slightly increased the frequency of significantly outperforming other ensemble methods. Through the investigation of bias and variance decomposition, we found that the reduction of variance played a major role in improving the accuracy of Canonical Forest. The gap in performance between Canonical Forest and the other methods decreased a little as the ensemble size increased. By investigating the κ-error diagram, we found that this is because Canonical Forest is slightly less diverse than the other ensemble methods. Nevertheless, Canonical Forest still showed a better performance in terms of the classification accuracy when it was compared to the other methods with the ensemble size of 500 which is nearly optimal for the other methods.

We also proposed another version of Canonical Forest, which is suitable for high-dimensional data, named HDCF (High Dimensional Canonical Forest) by implementing the algorithm of Random Subspace into Canonical Forest. Basically, the only difference between Canonical Forest and HDCF is the number of features used in training phase when growing a tree. Unlike Canonical Forest, HDCF uses only a feature subset instead of all the features when training the classifier. By doing so, HDCF can be applied to high-dimensional data without any feature pre-selection.

In our experiments, HDCF had the highest classification accuracy in gene imprinting and leukemia data sets and second highest classification accuracy in estrogen data set when comparing with the performance of three widely used high-dimensional classification methods: CERP, Random Forest and SVM. Besides the classification accuracy, we also investigated the balance between sensitivity and specificity for all these four classification methods. The performance of HDCF on the balance between sensitivity and specificity was quite comparable to the other three classification methods. We also evaluated the performance of HDCF by using the simulated data and HDCF gave consistently better classification accuracies than the other three methods in all three simulated data. From the experiments and the simulation study, we have shown that HDCF is quite a comparable classification method for classifying high-dimensional data.

Since both Canonical Forest and HDCF are CLDA-based ensemble methods, the performance of Canonical Forest and HDCF may depend on the performance of CLDA. Therefore, Canonical Forest and HDCF will perform better in the situations where CLDA performs better. In general, CLDA works better when the discriminatory information is in the mean instead of the variance of the data. Besides, CLDA also works better in balanced data (i.e., the number of instances for each class is roughly equal) than in unbalanced data because it needs representative instances for each class to make a better separation among classes.

Chapter 6

Future Study

In this dissertation, the discrete features were removed from all the data in the experiment because CLDA and PCA cannot be applied to discrete features. Although it is acceptable to apply CLDA or PCA to ordinal discrete features by treating them as continuous features, they still cannot be applied to nominal discrete features. We will continue working on this issue in the future study.

It also should be noted that the number of features in each subset was set to be $m = 3$ for both Rotation Forest and Canonical Forest because this was the parameter setting used in the experiment in Rotation Forest (Rodríguez et al., 2006). We used the same parameter setting to make a consistent comparison. We will further investigate the choice of m on the performance of Canonical Forest in future study like Kuncheva and Rodríguez (2007) did.

To overcome the issue that Canonical Forest works better in balanced data than unbalanced data, we will adopt weighted bootstrap sampling method instead of regular bootstrap sampling method when constructing each classifier in Canonical Forest. An experiment on evaluating the performance of Canonical Forest with weighted bootstrap sampling method will be conducted in our future study.

References

[1] Ahn, H., Moon, H., Fazzari, M. J., Lim, N., Chen, J. J. and Kodell, R. L., 2007. Classification by Ensembles from Random Partitions of High-Dimensional Data. *Computational Statistics and Data Analysis*, **51** 6166-6179.

[2] Asuncion, A., Newman, D. J., 2007. UCI *Machine Learning Repository*. University of California, Irvine, School of Information and Computer Science, http://www.ics.uci.edu/ ~mlearn/MLRepository.html.

[3] Bauer, E. and Kohavi, R., 1999. An Empirical Comparison of Voting Classification Algorithms: Bagging, boosting and variants. *Machine Learning*, **36 (1/2)** 105-139.

[4] Berkson, J., 1944. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, **39 (227)** 357-365.

[5] Blair, R., Fang, H., Branham, W. S., Hass, B., Dial, S. L., Moland, C. L., Tong, W., Shi, L., Perkins, R. and Sheehan, D. M., 2000. Estrogen Receptor Relative Binding Affinities of 188 Natural and Xenochemicals: Structural Diversity of Ligands. *Toxicological Sciences*, **54** 138-153.

[6] Breiman, L., 1996. Bagging predictors. *Machine Learning*, **24** 123-140.

[7] Breiman, L., 1998. Arcing classifiers. *The Annals of Statistics*, **26** 801-849.

[8] Breiman, L., 2001. Random Forest. *Machine Learning*, **45** 5-32.

[9] Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J., 1984. *Classification and Regression Trees*. Wadsworth, Belmont, CA.

[10] Brown, G., and Kuncheva, L.I., 2010. ”Good” and ”bad” diversity in majority vote ensembles. *9th Int. Workshop on Multiple Classifier Systems (MCS 2010)*, 124-133.

[11] Bühlmann, P. and Yu., B., 2003. Boosting with the L2 loss: regression and classification. Journal of the American Statistical Association, **98** 324-339.

[12] Bühlmann, P. and Yu, B., 2006. Sparse boosting. *Journal of Machine Learning Research*, **7** 1001-1024.

[13] Cohen, J., 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20(1)** 37-46.

[14] Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, **13(1)** 21-27.

[15] Cramer, J.S., 2002. The Origins of Logistic Regression. *Tinbergen Institute Working Paper* No. 2002-119/4, Tinbergen Institute.

[16] Dietterich, T. G., 2000. Ensemble Methods in Machine Learning. *Multiple Classifier Systems*, 1-15.

[17] Duda, R., and Hart, P., 1973. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York.

[18] Dudoit, S., Fridlyand, J., and Speed, T. P., 2002. Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. *Journal of the American Statistical Association*, **97** 77-87.

[19] Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7(2)** 179-188.

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

[21] Freund, Y. and Schapire, R., 1997. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, **55** 119-139.

[22] Friedman, J., 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, **29** 1189-1232.

[23] Friedman, J., Hastie, T. and Tibshirani, R., 2000. Additive logistic regression: a statistical view of boosting. *Annals of Statistics,* **28(2)** 337-407.

[24] Geman, S., Bienenstock, E., and Doursat, R., 1992. Neural networks and the bias/variance dilemma. *Neural Computation*, **4** 1-48.

[25] Greally, J. M., 2002. Short Interspersed Transposable Elements (SINEs) Are Excluded from Imprinted Regions in the Human Genome. *Proceedings of National Academy of Science,* **99** 327-332.

[26] Hansen, L. and Salamon, P., 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12** 993–1001.

[27] Hastie, T., Tibshirani, R. and Friedman, J., 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Verlag, New York.

[28] Hayashi, K., 2012. A boosting method with asymmetric mislabeling probabilities which depend on covariates. *Computational Statistics,* **27** 203-218.

[29] Heinz, G., Peterson, L. J., Johnson, R. W. and Kerk, C. J., 2003. Exploring relationships in body dimensions. *Journal of Statistics Education*, **11.** http://www.amstat.org/~publicati ons/jse/v11n2/datasets.heinz.html.

[30] Ho, T. K., 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20** 832-844.

[31] Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics,* **6** 65-70.

[32] Hothorn, T. and Lausen, B., 2003. Double-Bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, **36** 1303-1309.

[33] Iba, W. and Langley, P., 1992. Introduction of one-level decision trees. In *Machine Learning: Proceedings of the Ninth International Conference,* 233-240.

[34] Jeetha, R., 2009. An Integrated Study on Decision Tree Induction Algorithms in Data Mining. Articlebase.com. http://www.articlesbase.com/computers-articles/an-integrated-study-on-decision-tree-induction-algorithms-in-data-mining-1287150.html

[35] Ji, C. and Ma, S., 1997. Combinations of Weak Classifiers. *IEEE Transactions on Neural Networks*, **8(1)** 32-42.

[36] Kass, C.V., 1980. An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Journal of Applied Statistics*, **29(2)** 119-127.

[37] Kim, H. and Loh, W.-Y., 2001. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, **96** 589-604.

[38] Kim, H. and Loh, W.-Y., 2003. Classification trees with bivariate linear discriminant node models. *Journal of Computational and Graphical Statistics*, **12** 512-530.

[39] Kim, H., Kim, H., Moon, H. and Ahn, H., 2011. Iterative Weight-Adjusted Voting Algorithm for Ensemble of Classifiers. *Journal of the Korean Statistical Society*, **40** 437-449.

[40] Kohavi, R., Wolpert, D. H., 1996. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 275-283.

[41] Kong, E. B., and Dietterich, T. G., 1995. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning.* Morgan Kaufmann, 313-321.

[42] Kuncheva, L.I., 2004. Diversity in Multiple Classifier Systems (editorial). *Information Fusion*, **6(1)** 3-4.

[43] Kuncheva, L.I. and Rodríguez, J. J., 2007. An experimental study on rotation forest ensembles. *Multiple Classifier Systems*, 459-468.

[44] Kuncheva, L.I. and Whitaker, C.J., 2003. Measures of Diversity in Classifier Ensembles. *Machine Learning*, **51** 181-207.

[45] Kuncheva, L.I., Whitaker, C.J., Shipp, C. and Duin, R., 2003. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications*, **6(1)** 22-31.

[46] Lam L., and Suen, C. Y., 1997. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transaction on Systems, Man and Cybernetics*, **27** 553-568.

[47] Lee, K., Ahn, H., Moon, H., Kodell, R. L. and Chen, J. J., 2013. Multinomial Logistic Regression Ensembles. *Journal of Biopharmaceutical Statistics,* **23** 681-694.

[48] Leisch, F. and Dimitriadou, E., 2010. mlbench: Machine Learning Benchmark Problems. R package version 2.0-0.

[49] Lim, N., Ahn, H., Moon, H. and Chen, J.J., 2010. Classification of High-dimensional Data with Ensemble of Logistic Regression Models. *Journal of Biopharmaceutical Statistics*, **20** 160-171.

[50] Loh, W.-Y., 2010. Improving the precision of classification trees. *Annals of Applied Statistics*, **4** 1710-1737.

[51] Loh, W.-Y., and Shih, Y.S., 1997. Split Selection Methods for Classification Trees. *Statistica Sinica*, **7** 815-840.

[52] Margineantu, D. D., and Dietterich, T. G., 1997. Pruning Adaptive Boosting. In *Machine Learning: Proceedings of the Fourteenth International Conference*, 211-218.

[53] McCulloch, W., Pitts, W., 1943. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, **5 (4)** 115-133.

[54] Morgan, J.N., and Sonquist, J.A., 1963. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, **58** 415-434.

[55] Panov, P. and Džeroski, S., 2007. Combining bagging and random subspaces to create better ensembles. *Advances in Intelligent Data Analysis VII*. Springer Berlin, 118-129.

[56] Quinlan, J.R., 1993. *C4.5: Programs for Machine Learning*. *Morgan Kaufmann Publishers*.

[57] Rodríguez, J. J., Kuncheva, L. I. and Alonso, C. J., 2006. Rotation Forest: A New Classifier Ensemble Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28(10)** 1619-1630.

[58] Rokach, L., 2010. *Pattern Classification Using Ensemble Methods: Series in Machine Perception and Artificial Intelligence*. World Scientific Publishing Company.

[59] Rosenblatt, F., 1958. The Perceptron: A Probalistic Model For Information Storage And Organization In The Brain. *Psychological Review,* **65 (6)** 386-408.

[60] Schapire, R. E., 1990. The strength of weak learnability. *Machine Learning*, **5** 197-227.

[61] Statlib, 2010. Datasets archive. Carnegie Mellon University, Department of Statistics, http://lib.stat.cmu.edu.

[62] Terhune, J. M., 1994. Geographical variation of harp seal underwater vocalisations. *Canadian Journal of Zoology*, **72** 892-897.

[63] Tibshirani, R., Hastie, T., Narasimhan, B., Chu, G., 2002. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc Natl Acad Sci U S A,* **99** 6567-6572.

[64] Tukey, J.W., 1977. *Exploratory data analysis*. Addison-Wesley, New York.

[65] Vapnik, V., 1995. *The Nature of Statistical Learning Theory*. Springer Verlag, New York.

[66] Webb, G.I., 2000. MultiBoosting: A Technique for Combining Boosting and Wagging. *Machine Learning*, **40(2)** 159-196.

[67] Zaiane, O. R., 1999. Introduction to Data Mining. University of Alberta, Department of Computing Science, http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput690/index.html.

[68] Zhang, C.-X. and Zhang, J.-S., 2008. RotBoost: a technique for combining rotation forest and AdaBoost. *Pattern Recognition Letters*, **29 (10)** 1524-1536.

[69] Zhu, J., Rosset, S., Zou, H., and Hastie, T., 2009. Multi-class Adaboost. *Statistics* and Its Interface, **2** 349-360.