

A Weight-Adjusted Voting Algorithm for Ensemble of Classifiers

Hyunjoong Kim^{a,*}, Hyeuk Kim^b, Hojin Moon^c, Hongshik Ahn^b

^a*Department of Applied Statistics, Yonsei University, Seoul 120-749, South Korea*

^b*Department of Applied Math and Statistics, Stony Brook University, Stony Brook, NY 11794-3600*

^c*Department of Mathematics and Statistics, California State University, Long Beach, CA 90840-1001*

Abstract

We present a new weighted voting classification ensemble method, called WAVE, that uses two weight vectors: a weight vector of classifiers and a weight vector of instances. The instance weight vector assigns higher weights to observations that are hard to classify. The weight vector of classifiers puts larger weights on classifiers that perform better on hard-to-classify instances. One weight vector is designed to be calculated in conjunction with the other through an iterative procedure. That is, the instances of higher weights play more important role in determining the weights of classifiers, and vice versa. We proved that the iterated weight vectors converge to the optimal weights which can be directly calculated from the performance matrix of classifiers in an ensemble. The final prediction of the ensemble is obtained by the voting using the optimal weight vector of classifiers. To compare the performance between a simple majority voting and the proposed weighted voting, we applied both of the voting methods to bootstrap aggregation and investigated the performance on 28 data sets. The result shows that the proposed weighted voting performs

*Corresponding author, tel: +82 2 2123 2545, fax: +82-2-2123-8638

Email address: hkim@yonsei.ac.kr (Hyunjoong Kim)

significantly better than the simple majority voting in general.

Keywords: Ensemble, Voting, Aggregation, Classification, Cross validation, Bagging, Boosting, Random Forest

1. Introduction

Classification is a predictive modeling whose target variable is categorical. In ensemble method for classification, many classifiers are combined to make a final prediction (Dietterich, 2000). Ensemble methods show better performances than a single classifier in general (Hansen and Salamon, 1990). The final decision is usually made by voting after combining the predictions from set of classifiers.

The use of ensemble of classifiers has gained wide acceptance in machine learning and statistics community thanks to significant improvement in accuracy (Breiman, 1996; Freund and Schapire, 1996; Bauer and Kohavi, 1999). Two popular ensemble methods, Boosting (Schapire, 1990; Freund and Schapire, 1996) and Bagging (Breiman, 1996), have received heavy attention. These methods use resampled or reweighted training sets from the original data. Then a learning algorithm is repeatedly applied for each of resampled or reweighted training set. Boosting was developed as a method for improving the performance of any weak learning algorithm, which requires only to be slightly better than random guess. Boosting changes adaptively the distribution of the training set based on the performance of previously created classifiers. To combine the classifiers in ensemble, Boosting takes a weighted majority vote of their predictions. The Bagging algorithm uses bootstrap samples to build the classifiers in ensemble. Each bootstrap sample is formed by randomly sampling, with replacement, the same number of instances as the original data. The final classification produced by the ensemble of these classifiers is obtained by simple majority voting. Later, Breiman (2001) developed Random Forest

23 by combining tree predictors such that each level of trees depends on the values
24 of features sampled independently. Some other ensemble methods (Breiman, 1998;
25 Ho, 1998; Webb, 2000; Hothorn and Lausen, 2003, 2005; Chandra and Yao, 2006;
26 Maslov and Gertner, 2006; Rodriguez et al., 2006; Zhang and Zhang, 2008; Masnadi-
27 Shirazi and Vasconcelos, 2011) have shown that the ensemble can outperform single
28 predictors in many cases.

29 Classification ensemble methods can be categorized by the ways they are built
30 (Kuncheva, 2005; Rokach, 2009). Specifically, Kuncheva (2005) defined four dimen-
31 sions for characterizing ensemble methods: combination level, classifier level, feature
32 level, and data level. In this article, we will focus on the combination level that deals
33 with the ways the classifier decisions are combined.

34 Simple majority voting is a decision rule that selects one of many alternatives,
35 based on the predicted classes with the most votes (Lam and Suen, 1997). It is
36 the decision rule used most often in ensemble methods. Weighted majority voting
37 can be made if the decision of each classifier is multiplied by a weight to reflect
38 the individual confidence of these decisions (Rahman et al., 2002). Simple majority
39 voting is a special case of weighted majority voting, assigning an equal weight of
40 $1/k$ to each classifier where k is the number of classifiers in an ensemble.

41 We propose a new weighted voting classification ensemble method. When an
42 ensemble of classifiers is constructed, the proposed method assigns unique voting
43 weights to each classifier in the ensemble. Specifically, using an iterative process, a
44 weight vector for the classifiers and another weight vector for the instances will be
45 obtained in the learning phase of model formation. We prove convergence of these
46 vectors in Section 2. After the final iteration, hard-to-classify instances get higher
47 weights and, subsequently, better performing classifiers on the hard-to-classify in-
48 stances are assigned larger weights. The final prediction of the ensemble is obtained

49 by the voting using the weight vectors of classifiers. In this paper, since boot-
50 strapping is a common choice for perturbation, we adopt the bootstrap aggregation
51 scheme to create ensemble of classifiers. We will refer the new ensemble method
52 as WAVE (Weight-Adjusted Voting for Ensembles of classifiers) for brevity. WAVE
53 uses the proposed weight adjusted voting algorithm under bootstrap aggregation
54 scheme. Note that the ensemble method using simple majority voting under boot-
55 strap aggregation is essentially Bagging.

56 Hard-to-classify instances are defined as those near class boundaries. For ex-
57 ample, support vectors in support vector machine (SVM) are the hard-to-classify
58 instances. They are tough ones to classify correctly because they are close to several
59 nearby classes. Outliers or mislabeled instances are also hard-to-classify. We assume
60 in this article that outliers or mislabeled instances are few in real world data, thus
61 ignorable.

62 To measure the effectiveness of the WAVE's voting method over the simple
63 majority voting, we will compare two methods using identical set of classifiers.
64 This will eliminate the source of discrepancy originated from unequal classifiers.
65 Therefore the difference in accuracy can solely be attributed to the difference of
66 voting methods.

67 We investigate their performances on 28 real and simulated data sets. To evalu-
68 ate the general classification performance of WAVE, further comparison with Boost-
69 ing and Random Forest (Breiman, 2001) are also included in the experiment. Deci-
70 sion tree is used as the base learning algorithm in the comparison. The comparison
71 results are in Section 4.

72 The classification error can be explained using a bias-variance decomposition.
73 Two classifiers with equal accuracy may have very different breakdowns with respect
74 to error. In Section 4, the bias-variance decomposition of each method is also

75 explored.

76 **2. Method Development**

77 In this section we introduce the WAVE ensemble method. This work was initi-
78 ated from an academic setting to evaluate students' performance on solving problems
79 of various levels of difficulty. The idea is to give more weights (scores) to students
80 who can solve harder questions. In classification of patients for individualized treat-
81 ment, for example, the classifier that can correctly predict a patient's prognosis
82 which is more complex and difficult gets more weight than other classifiers in the
83 learning phase.

84 *2.1. Iterative Weight Adjust Algorithm*

85 Suppose that there are two classifiers. In simple majority voting, all classifiers
86 have equal weights. Hence, if the two classifiers make different predictions for an
87 instance, the final decision becomes arbitrary due to the tied votes. Now suppose
88 the first one tends to classify more hard-to-classify instances correctly, whereas the
89 second classifier does easier instances correctly. If the two classifiers make different
90 predictions on an unseen instance, it is reasonable to give more weight to the first
91 classifier which classifies more difficult instances correctly.

92 It is necessary to properly assess hard-to-classify instances. In an academic
93 setting, harder questions may be defined as those fewer students get correct answers.
94 Similarly in an ensemble, hard-to-classify instances can be thought as those that
95 fewer classifiers make correct predictions. Based on this idea, we design two weight
96 vectors: a weight vector of classifiers and a weight vector of instances. The weights
97 for instances are proportional to the degree of difficulty of each instance. They are
98 taken into account in assigning weights to classifiers. Thus the weights for classifiers
99 and the weights for instances influence each other. Through the cross relationship

100 between the performance of classifiers and the difficulty of instances, we can find
 101 the optimal weights for classifiers and instances. These weights are found by an
 102 iterative procedure and determined only by the matrix about the instances and
 103 the classifiers. We do not need to assume prior knowledge of the behavior of the
 104 individual classifiers.

105 Suppose we have n instances and k classifiers in an ensemble. We let \mathbf{X} be
 106 an $n \times k$ ‘performance matrix’ indicating whether the classification is right (1) or
 107 wrong (0), and its transpose \mathbf{X}' . Let \mathbf{J}_{ij} be an $i \times j$ matrix consisting of 1’s for
 108 any dimension i and j . We also define $\mathbf{1}_n$ and $\mathbf{1}_k$ as $n \times 1$ and $k \times 1$ vectors of 1’s,
 109 respectively. Finally, \mathbf{I}_k denotes a $k \times k$ identity matrix. The algorithm to get two
 110 weight vectors is described as follows:

111 **Algorithm 1.** *Iterative weight adjust algorithm:*

112 1. *Set an initial instance weight vector*

$$\mathbf{Q}_0 = \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k}.$$

113 \mathbf{Q}_0 takes higher instance weights for the rows of \mathbf{X} with fewer 1’s (i.e., hard-
 114 to-classify instances). The denominator is a normalizing factor for unit norm.

115 2. For $m = 1, 2, \dots$, repeat (a) and (b).

116 (a) *Calculate a classifier weight vector*

$$\mathbf{P}_m = \frac{\mathbf{X}'\mathbf{Q}_{m-1}}{\mathbf{1}'_k\mathbf{X}'\mathbf{Q}_{m-1}}.$$

117 \mathbf{P}_m assigns higher weights on more accurate classifiers (i.e., columns
 118 of \mathbf{X} with more 1’s) after the instance weight \mathbf{Q}_{m-1} incorporated. The
 119 denominator is a normalizing factor for unit norm.

120 (b) *Update the instance weight vector*

$$\mathbf{Q}_m = \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{P}_m}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{P}_m}.$$

121 \mathbf{Q}_m assigns higher weights on hard-to-classify instances after incorporat-
122 ing the classifier weight vector \mathbf{P}_m . The denominator is a normalizing
123 factor for unit norm.

124 3. Stop the step #2 when the weight vectors \mathbf{P}_m and \mathbf{Q}_m become stable. Denote
125 \mathbf{P}^* and \mathbf{Q}^* be the final weight vectors.

126 We provide a simple example illustrating the above algorithm. We suppose
127 that there are three classifiers and four instances in an ensemble. We define $\mathbf{X} =$
128 $(\mathbf{x}_1|\mathbf{x}_2|\mathbf{x}_3)$ and let $\mathbf{x}_1 = (1, 1, 1, 0)'$, $\mathbf{x}_2 = (0, 1, 1, 0)'$ and $\mathbf{x}_3 = (1, 1, 0, 1)'$, where
129 \mathbf{x}_i indicates a performance vector by the i th classifier. Here, 1 represents a cor-
130 rect decision from a classifier and 0 represents an incorrect decision. We obtain the
131 normalized weight vector on classifier decisions $\mathbf{P}^* = (0.311, 0.097, 0.592)'$ and the
132 normalized weight vector on instances $\mathbf{Q}^* = (0.311, 0.000, 0.140, 0.548)'$ by Algo-
133 rithm 1.

134 The classifier weight \mathbf{P}^* can be explained as follows. The accuracies of the
135 classifiers are (0.75, 0.5, 0.75). Although the first and the third classifiers have the
136 same error rate, more weight is given to the third classifier (0.592) than the first
137 classifier (0.311) because the former classified the most difficult instance correctly.
138 It is the only classifier which had the correct decision on the fourth instance. The
139 weight for the fourth instance is highest (0.548) in \mathbf{Q}^* . The least classifier weight
140 is given to the second classifier (0.097) because it misclassified higher weighted
141 instances and is the most inaccurate among the three. Regarding \mathbf{Q}^* , zero weight
142 is given to the second instance because all the classifiers made correct decisions.
143 The highest weight (0.548) is given to the fourth instance because it is the hardest
144 to classify. Only the third classifier predicted it correctly. Although the first and
145 the third instance get the same accuracy (2/3), we note that the weights are quite
146 different. This is due to the effect of \mathbf{P}^* . The first instance is misclassified only

147 by the second classifier which has the least weight. On the other hand, the third
 148 instance is misclassified by the most important classifier. When the instances are
 149 equally difficult, an instance on which the higher weighted classifier works better
 150 must get a higher value in \mathbf{Q}^* . Therefore, the first instance received a higher weight
 151 than the third instance.

152 2.2. Convergence

153 We now provide the proof on the convergence of two weight vectors introduced
 154 in Algorithm 1.

155 **Theorem 1.** *With an initial condition $\mathbf{Q}_0 = (\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k/\mathbf{1}'_n(\mathbf{J}_{nk} -$*
 156 *$\mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k$, the weight vectors converge as*

$$\mathbf{P}^* = \lim_{m \rightarrow \infty} \mathbf{P}_m = \lim_{m \rightarrow \infty} \frac{\mathbf{X}'\mathbf{Q}_{m-1}}{\mathbf{1}'_k\mathbf{X}'\mathbf{Q}_{m-1}} = \frac{(\sum_{i=1}^r \mathbf{u}_i\mathbf{u}'_i)\mathbf{1}_k}{\mathbf{1}'_k(\sum_{i=1}^r \mathbf{u}_i\mathbf{u}'_i)\mathbf{1}_k}$$

157 and

$$\begin{aligned} \mathbf{Q}^* = \lim_{m \rightarrow \infty} \mathbf{Q}_m &= \lim_{m \rightarrow \infty} \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{P}_m}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{P}_m} \\ &= \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)(\sum_{i=1}^r \mathbf{u}_i\mathbf{u}'_i)\mathbf{1}_k}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)(\sum_{i=1}^r \mathbf{u}_i\mathbf{u}'_i)\mathbf{1}_k}, \end{aligned}$$

158 where r is the number of dominating eigenvalues of $\mathbf{X}'(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)$ such
 159 that $\lambda_1 = \lambda_2 = \dots = \lambda_r > \lambda_{r+1}$, $1 \leq r \leq k$, and \mathbf{u}_i is the eigenvector corresponding
 160 to the eigenvalue λ_i , $i = 1, \dots, r$.

161 **Proof:** With $\mathbf{Q}_0 = (\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k/\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k$, \mathbf{P}_1 and
 162 \mathbf{Q}_1 are updated as

$$\mathbf{P}_1 = \frac{\mathbf{X}'\mathbf{Q}_0}{\mathbf{1}'_k\mathbf{X}'\mathbf{Q}_0} = \frac{\mathbf{X}'(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k}{\mathbf{1}'_k\mathbf{X}'(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{1}_k} = \frac{\mathbf{T}\mathbf{1}_k}{\mathbf{1}'_k\mathbf{T}\mathbf{1}_k}$$

163 and

$$\mathbf{Q}_1 = \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{P}_1}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{P}_1} = \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{T}\mathbf{1}_k}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{T}\mathbf{1}_k},$$

164 where $\mathbf{T} = \mathbf{X}'(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)$. At $m = 2$,

$$\mathbf{P}_2 = \frac{\mathbf{X}'\mathbf{Q}_1}{\mathbf{1}'_k\mathbf{X}'\mathbf{Q}_1} = \frac{\mathbf{T}^2\mathbf{1}_k}{\mathbf{1}'_k\mathbf{T}^2\mathbf{1}_k} \quad \text{and} \quad \mathbf{Q}_2 = \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{T}^2\mathbf{1}_k}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{T}^2\mathbf{1}_k}.$$

165 At the m -th iteration,

$$\mathbf{P}_m = \frac{\mathbf{T}^m\mathbf{1}_k}{\mathbf{1}'_k\mathbf{T}^m\mathbf{1}_k} \quad \text{and} \quad \mathbf{Q}_m = \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{T}^m\mathbf{1}_k}{\mathbf{1}'_n(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)\mathbf{T}^m\mathbf{1}_k}.$$

166 We note that \mathbf{T} is a positive semi-definite matrix, thus all of its eigenvalues are
 167 nonnegative, and there is at least one positive eigenvalue. Note that \mathbf{T} can be
 168 decomposed as $\mathbf{T} = \mathbf{U}\mathbf{D}\mathbf{U}'$, where $\mathbf{U}\mathbf{U}' = \mathbf{I}$. Here, \mathbf{U} is a $k \times k$ matrix whose
 169 column vectors are the eigenvectors of \mathbf{T} , and \mathbf{I} is a $k \times k$ identity matrix. \mathbf{D} is a
 170 diagonal matrix whose diagonal elements are the eigenvalues of \mathbf{T} . The eigenvalues
 171 of \mathbf{T} are $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k \geq 0$, where $\lambda_1 > 0$. Therefore, there are no dominant
 172 eigenvalues with opposite sign. Since $\mathbf{T} = \mathbf{U}\mathbf{D}\mathbf{U}'$, $\mathbf{T}^m = \mathbf{U}\mathbf{D}^m\mathbf{U}' = \sum_{i=1}^k \lambda_i^m \mathbf{u}_i \mathbf{u}'_i$,
 173 where \mathbf{u}_i is the eigenvector corresponding to λ_i . Suppose $\lambda_1 = \lambda_2 = \dots = \lambda_r > \lambda_{r+1}$,
 174 $1 \leq r < k$. Then

$$\mathbf{T}^m = \sum_{i=1}^r \lambda_1^m \mathbf{u}_i \mathbf{u}'_i + \sum_{j=r+1}^k \lambda_j^m \mathbf{u}_j \mathbf{u}'_j = \lambda_1^m \left[\sum_{i=1}^r \mathbf{u}_i \mathbf{u}'_i + \sum_{j=r+1}^k (\lambda_j^m / \lambda_1^m) \mathbf{u}_j \mathbf{u}'_j \right].$$

175 Since $\lambda_1 > \lambda_j$ for $j = r + 1, \dots, k$ and $\lambda_1 > 0$, $\lim_{m \rightarrow \infty} (\lambda_j^m / \lambda_1^m) = 0$ and λ_1^m is
 176 cancelled out in the numerator and the denominator of \mathbf{P}_m and \mathbf{Q}_m . Hence

$$\lim_{m \rightarrow \infty} \mathbf{P}_m = \mathbf{P}^* = \frac{(\sum_{i=1}^r \mathbf{u}_i \mathbf{u}'_i) \mathbf{1}_k}{\mathbf{1}'_k (\sum_{i=1}^r \mathbf{u}_i \mathbf{u}'_i) \mathbf{1}_k}$$

177 and

$$\lim_{m \rightarrow \infty} \mathbf{Q}_m = \mathbf{Q}^* = \frac{(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k) (\sum_{i=1}^r \mathbf{u}_i \mathbf{u}'_i) \mathbf{1}_k}{\mathbf{1}'_n (\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k) (\sum_{i=1}^r \mathbf{u}_i \mathbf{u}'_i) \mathbf{1}_k}.$$

178 This concludes the proof.

179 *2.3. WAVE Ensemble Method*

180 We observe from Theorem 1 that \mathbf{P}^* and \mathbf{Q}^* , the converged forms of \mathbf{P}_m and \mathbf{Q}_m ,
181 respectively, have closed solutions. In consequence, \mathbf{P}^* can be calculated directly
182 from $\mathbf{X}'(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)$. Therefore, we can find the classifier weight vector
183 \mathbf{P}^* without using Algorithm 1. Consequently, the weight vector \mathbf{Q}^* does not need
184 to be calculated. Promoted by the result of Theorem 1, we present a new ensemble
185 method, called WAVE, in Figure 1. As mentioned previously, the proposed WAVE
186 ensemble method uses bootstrap aggregation scheme and the new weight-adjusted
187 voting algorithm. We note that WAVE’s voting algorithm can be applied in any
188 other aggregation schemes because the classifier weight vector is found after the
189 ensemble formation is completed. However, this article concentrates on the new
190 voting algorithm with bootstrap aggregation scheme only.

191 **3. Illustration**

192 In this section, we illustrate how WAVE voting can improve the simple majority
193 voting using an example. Let us consider a two-class chessboard data shown in
194 Figure 2. We try to solve the classification of the black and white squares in the
195 chessboard. Two input variables are uniformly distributed over $(0, 4)$ and represent
196 the horizontal and vertical axes, individually. The class memberships of instances
197 are determined according to the chessboard regions.

198 To better observe the improvement by WAVE over Bagging, we intentionally
199 employ a weak learning scheme: a few learned classifiers in the ensemble with small-
200 size training data. We set the number of classifiers in the ensemble (ensemble size) to
201 be only 4. We generated 200 random instances as a training set. The base classifier
202 employed is the CART (Breiman et al., 1984) decision tree algorithm. We grow
203 unpruned trees as each classifier. The test set of 16,000 instances is independently

- Input:

- L : training set composed of n instances
- L_y : class membership of L
- k : number of classifiers in an ensemble

- Output:

- $C^*(\cdot)$: Weighted combination of outputs of classifiers, that is

$$C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{b=1}^k \mathbf{P}_b^* \times I(C_b(\mathbf{x}) = y)$$

1. for $b = 1$ to k {
 2. $L_{bt} = \text{BootstrapSample}(L)$
 3. $C_b = \text{TrainClassifier}(L_{bt})$
 4. $X_b = I\{C_b(L) = L_y\}$, $n \times 1$ vector of 0's (wrong) and 1's (correct)
 5. Add C_b to pool C
 6. }
 7. $\mathbf{X} = [X_1, \dots, X_k] = n \times k$ matrix consisting of 0's (wrong) and 1's (correct)
 8. $\mathbf{T} = \mathbf{X}'(\mathbf{J}_{nk} - \mathbf{X})(\mathbf{J}_{kk} - \mathbf{I}_k)$
 9. $\lambda_i =$ eigenvalues of \mathbf{T} , $i = 1, \dots, k$
 10. $\mathbf{u}_i =$ eigenvector corresponding to λ_i , $i = 1, \dots, k$
 11. $r =$ number of dominating eigenvalues such that $\lambda_1 = \lambda_2 = \dots = \lambda_r > \lambda_{r+1}$,
 $1 \leq r \leq k$
 12. $\mathbf{P}^* = \frac{(\sum_{i=1}^r \mathbf{u}_i \mathbf{u}_i') \mathbf{1}_k}{\mathbf{1}_k' (\sum_{i=1}^r \mathbf{u}_i \mathbf{u}_i') \mathbf{1}_k} = [\mathbf{P}_1^*, \dots, \mathbf{P}_k^*]'$
 13. return C and \mathbf{P}^*
-

Figure 1: The pseudocode for the WAVE ensemble method.

Figure 2: Chessboard data with two classes.

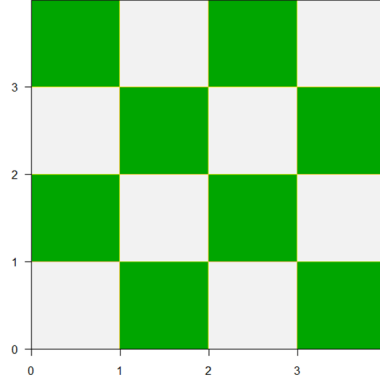
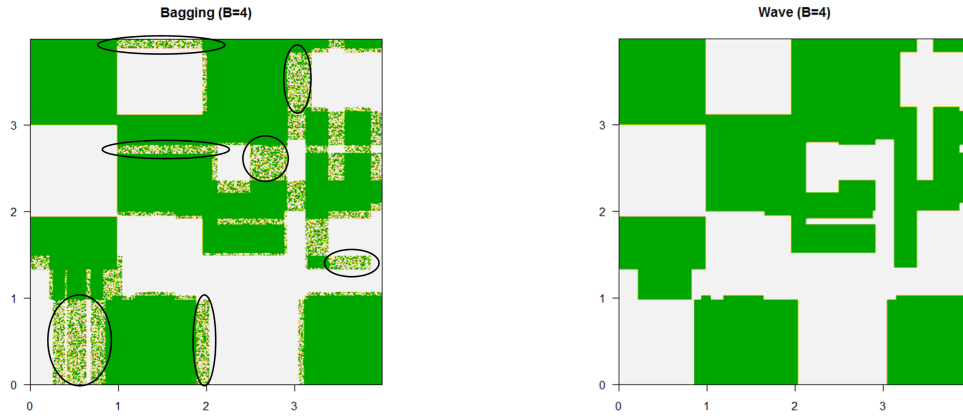


Figure 3: Bagging and WAVE classification predictions on chessboard data. The elliptical marks on the left panel refer to some major areas that Bagging has less accurate results but WAVE has successfully improved.



204 generated from the same setting.

205 Figure 3 shows the classification predictions by Bagging and WAVE. Due to the
206 weak learning scheme, the prediction accuracies are relatively poor considering the
207 simple structure of the data. The accuracies of Bagging and WAVE are 0.80 and
208 0.83, respectively.

Table 1: Comparison of WAVE and Bagging when mislabeled observations exist.

Mislabeled %	Mean accuracy		Paired t-test	
	WAVE	Bagging	S.E.	P-value
0%	0.796	0.767	0.005	0.00004
5%	0.742	0.720	0.005	0.0002
10%	0.710	0.692	0.004	0.0006
20%	0.643	0.631	0.004	0.007

209 It is observed that some prediction areas by Bagging show less accurate results.
 210 Some of such areas are marked with ellipsoid. The instances in these areas are harder
 211 to classify correctly than those in other areas. Since some trees in the ensemble that
 212 have better classification on these areas get higher weights, WAVE can successfully
 213 improve the classification on such instances.

214 Mislabeled instances, although not common and assumed to be ignorable in this
 215 paper, may affect the performance of WAVE adversely because they are also hard-
 216 to-classify instances. We carry out an experiment using the chessboard data to
 217 see the impact of mislabeled instances. The experiment is repeated 20 times with
 218 the same setting of Figure 3. The paired t-test is also performed to compare two
 219 methods pairwise for each repeat. Up to 20% of mislabeled instances in training
 220 data are included because it is impractical to imagine more than 20% of mislabeled
 221 instances in real world. It is observed in Table 1 that the significance of WAVE over
 222 Bagging reduces as the mislabeled instances increases. However, WAVE does not
 223 fail by the mislabeled instances. We interpret that, although mislabeled instances
 224 affect the performance of WAVE adversely, 'good' instances near class boundaries
 225 (e.g. support vectors) still play an important role maintaining the performance of
 226 WAVE.

227 4. Experimental Studies

228 We describe an empirical study comparing WAVE and Bagging. Other popular
229 ensemble methods such as Boosting (Freund and Schapire, 1996) and Random Forest
230 (Breiman, 2001) methods are also included in the comparison to show the relative
231 performance of WAVE, regardless of ensemble mechanism. The accuracy of a single
232 tree is provided as a baseline performance. We also provide a bias and variance
233 decomposition of errors to show how different methods influence these two terms.

234 4.1. Experimental Design

235 We collect 28 real or artificial datasets as summarized in Table 2. Most of the
236 data come from UCI Data Repository (Asuncion and Newman, 2007). To obtain a
237 better measure of predictive accuracy, we compare several ensemble methods using
238 10-fold cross-validation. The cross-validation accuracy is the average of the ten
239 estimates. The 10-fold cross-validation is repeated 100 times with different seeds
240 each time to provide more stable estimates. The final accuracy estimate for a dataset
241 is the average of 100 cross-validation accuracies.

242 A decision tree using CART (Breiman et al., 1984) algorithm is employed as
243 the base learning algorithm. Decision tree classifiers are generated multiple times
244 by each ensemble method’s own mechanism. The generated classifiers are then
245 combined to form an ensemble.

246 We performed all the experiments using R statistical package. The CART algo-
247 rithm was implemented in the library ‘rpart’ of R package. We implement Bagging
248 and WAVE under the R environment. We also implement Boosting using the multi-
249 class boosting algorithm called SAMME (Zhu et al., 2009). We use ‘randomForest’
250 function available in R for implementing Random Forest.

251 Tree size of a decision tree in an ensemble can be an interesting issue in the

252 comparative study. When a single tree algorithm is compared, the pruning option
253 should be used for optimal performance because pruning procedure would prevent
254 over-fitting of training data which occurs frequently by unpruned trees. On the
255 other hand, Bauer and Kohavi (1999) reported that unpruned trees give more accu-
256 rate classifications in Bagging because they reduce the bias more effectively. Since
257 WAVE also uses bootstrap aggregation like Bagging, it would be reasonable to grow
258 unpruned trees for WAVE. By the similar reason, unpruned trees would be adequate
259 for Random Forest. For ‘randomForest’ function in R, the unpruned tree is indeed
260 the default option.

261 On the other hand, it is known that Boosting performs better with smaller trees.
262 Hastie et al. (2001) claimed that using trees with between four and eight terminal
263 nodes works well in most cases, and that performance is fairly insensitive to the
264 choice from this range. Friedman et al. (2000) gave an example in which stumps
265 are superior to larger trees. As a general strategy, one might choose stumps if
266 it is suspected that effects are additive, and choose larger trees if one anticipates
267 interactions for which adjustments should be made. Hastie et al. (2001) mentioned
268 that a reasonable default option for Boosting is to limit the tree growing with the
269 maximum tree depth being the number of classes, where depth 0 denotes the root
270 node. We used the same default option for the experiment.

271 *4.2. Experimental Results*

272 *4.2.1. Significance and Relative Improvement*

273 For fair comparisons, all the methods are run on the same 10-folded dataset,
274 thus the cross-validation accuracy of each method is comparable. Since we repeat
275 the cross-validation 100 times, there are 100 paired accuracies to be compared.

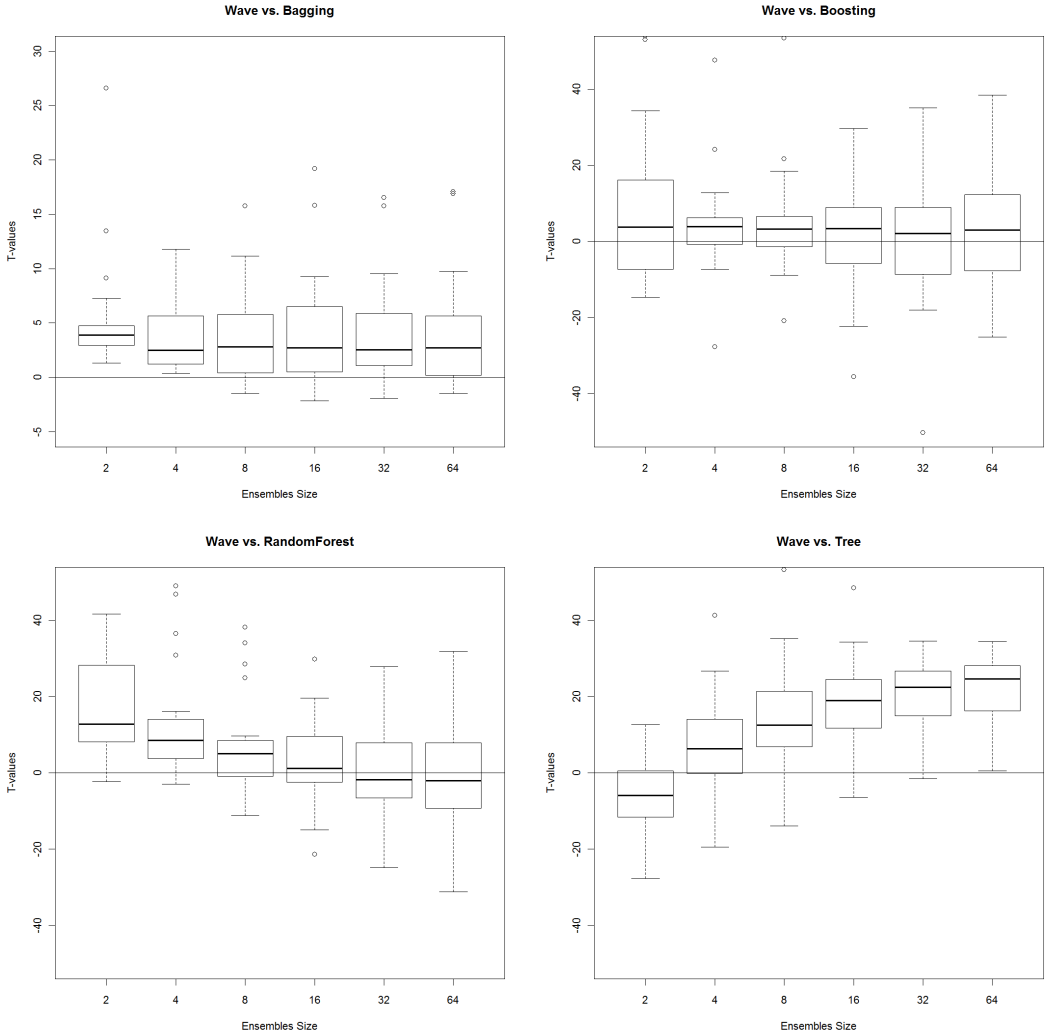
276 We use paired t-test to compare the accuracy of the methods. A paired t-statistic
277 will be calculated from each dataset. A large t-statistic would mean consistently

278 better accuracy for one method over the other. We plot the t-statistics from 28
279 datasets in Figure 4. The t-statistics when comparing WAVE with other methods
280 pairwise are presented here. We tried various numbers of classifiers in an ensemble to
281 see if the relationship between the t-statistics and the ensemble size exists. Between
282 WAVE and Bagging, boxplots are in general similar over various sizes of ensemble.
283 Because the t-statistics are large and mostly positive, it suggests that WAVE indeed
284 outperforms Bagging consistently. In other words, the WAVE voting method is
285 significantly more effective than simple majority voting.

286 Although not strong, WAVE shows slightly better results than Boosting across
287 the various ensemble sizes. There is no clear trend between the t-statistics and
288 the ensemble size. When comparing WAVE and Random Forest, while WAVE is
289 better for small ensembles, it appears that Random Forest gets its slight margin as
290 an ensemble gets bigger. However, the statistical significance is weak because the
291 variability of t-statistics also increases.

292 It is obvious that WAVE outperforms a single decision tree as the ensemble size
293 increases. It is also notable that a single pruned tree can be better than WAVE when
294 the ensemble size is only two ($k = 2$). In fact, this is true for all ensemble methods
295 because WAVE shows more accurate results than other ensemble methods for $k = 2$.
296 This is due to the fact that the learned classifiers in an ensemble are weaker than
297 the base classifier built on the whole data. For the example of Bagging and WAVE,
298 bootstrapping is used to resample the training data at each stage of ensemble. It is
299 well known that an instance in the training data has probability of 63.2% of being
300 selected at least once by bootstrap. This means that each bootstrap sample contains
301 only about 63.2% unique instances from the training data. Therefore, with only two
302 ensembles ($k = 2$), the instances participated in the two classifiers are far less than
303 the original training data. As a result, the two-classifier-ensemble becomes worse

Figure 4: Paired t-statistics for comparing WAVE vs. other methods. Large positive values mean significantly better accuracy for WAVE over other methods.



304 than a single pruned tree.

305 We also compare the relative improvement of accuracy between WAVE and other
306 methods. The value of relative improvement is calculated at each of 10-fold cross-
307 validation, and is defined as follows.

$$\text{Relative improvement of A over B} = \frac{\text{Error rate of B} - \text{Error rate of A}}{\text{Error rate of B}}.$$

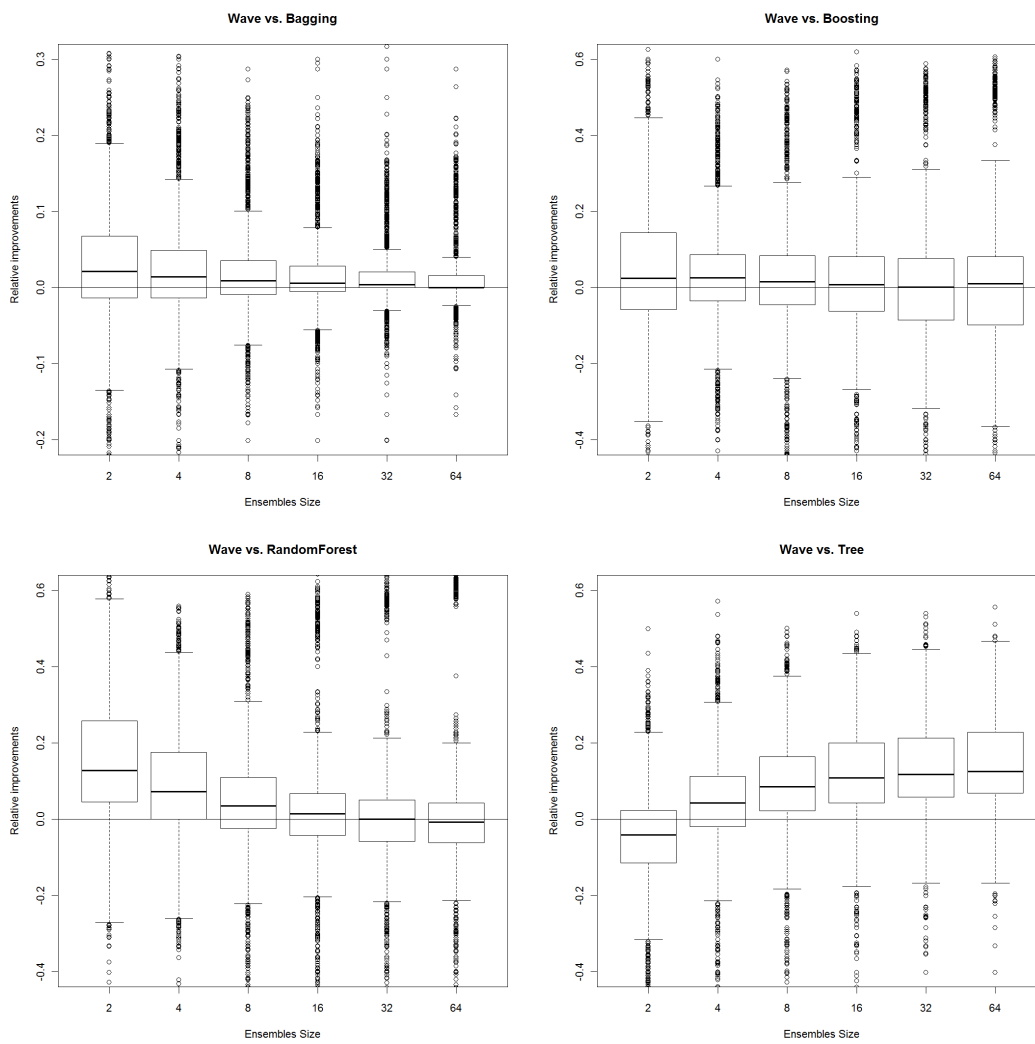
308 Since there are 100 repeats of 10-fold cross-validations for each of 28 dataset, we
309 calculated 2,800 relative improvements overall. We show the relative improvements
310 of accuracy in Figure 5.

311 When WAVE is compared with Bagging, the relative improvement is getting
312 smaller as the ensemble size increases, but at the same time, the variance also gets
313 smaller. This implies that, although the relative improvement gets smaller, the
314 differences become more consistent as the ensemble size increases. Therefore, the
315 statistical significance may still hold. This result coincides with the t-statistics in
316 Figure 4.

317 WAVE and Boosting are quite comparable in the relative improvement although
318 WAVE has a slight margin over Boosting. WAVE shows improved accuracy for
319 small ensembles over Random Forest. However, it seems Random Forest begins to
320 catch up for larger ensembles.

321 As seen earlier, WAVE outperforms a single pruned tree as the ensemble size
322 grows. Also it shows a single pruned tree can be better than any other ensemble
323 method when the ensemble size is only two. We observe that mean relative im-
324 provement of WAVE over a single pruned tree is about 15% when ensemble size is
325 64.

Figure 5: Relative improvements of WAVE over other methods.



326 *4.2.2. Accuracy Comparison*

327 We present the mean of 100 cross-validation (10-fold each) accuracies for 28
328 datasets. The results of ensembles with 64 classifiers are shown in Table 3. For
329 each dataset, the first and the second best results are highlighted in boldface. Fur-
330 thermore, paired t-tests and Wilcoxon signed rank tests between a single pruned
331 tree and each ensemble method are performed pairwise. Corresponding statistics
332 are given at the bottom of the table.

333 WAVE and Random Forest give similar number of highlights and worsts. WAVE
334 shows more highlights than Boosting and Bagging. It is noted that Boosting has
335 more worst cases than other ensemble methods. It seems Boosting shows improved
336 result in general, but it gets worse for some types of dataset. This was previously
337 reported in literature such as in Bauer and Kohavi (1999).

338 Bauer and Kohavi (1999) also observed that Bagging performs better than a
339 single tree uniformly on many datasets they studied. Since WAVE extends the
340 Bagging’s simple majority voting, the consistency of better performance is even
341 stronger. This probably is the reason why WAVE is the best among ensemble
342 methods in terms of t and Wilcoxon statistics.

343 The chessboard data called ‘Int’ and introduced in Section 3 reveals poor ac-
344 curacy for Boosting and Random Forest. Specifically, Random Forest shows very
345 poor result. This one dataset may substantially affect the t-statistics. Thus, we
346 also listed the statistics excluding ‘Int’ data in the parenthesis for reference. In
347 summary, when both the mean and the variance are considered, WAVE shows the
348 most significant improvement over a single pruned tree.

349 *4.2.3. Bias and Variance Decomposition*

350 The bias and variance decomposition (Geman et al., 1992) is a useful tool for
351 analyzing classification algorithms. It was first studied for regression problems with

352 quadratic loss function. For classification, the quadratic loss function is inappro-
353 priate because class labels are categorical. Several proposals for decomposing clas-
354 sification error into bias and variance have been suggested, including Kong and
355 Dietterich (1995), Kohavi and Wolpert (1996), and Breiman (1998). We choose to
356 use the decomposition by Kohavi and Wolpert (1996).

357 To properly estimate the bias and the variance, test data need to be fixed to
358 calculate variations iteration to iteration. On the other hand, different training
359 data must be sampled from the same population in each iteration. For this reason,
360 it is natural to consider artificial data because it is easier to sample training data
361 repeatedly than using real dataset.

362 Three scenarios of artificial data used in the experiment are described in Table 4.
363 The observation size of both training and test data are set to 500, respectively. The
364 ensemble size is given as 50. In one iteration, all the ensemble methods are performed
365 on a training data, then the error rate is calculated for the fixed test data. The whole
366 process is repeated 1000 times to estimate the bias and the variance of predictions.

367 Bauer and Kohavi (1999) reported that the Bagging method reduces the variance
368 dramatically when compared to a single decision tree. Because the instability of a
369 tree encourages a wide variety of trees in the ensemble, it can be interpreted that the
370 averaging over variety of trees reduces the variance. They also found that the error
371 reduction of Boosting is due to both bias and variance reduction. We interpret the
372 adjustable case weights serve to reduce the bias, while the averaging of the results
373 reduces the variance. Since WAVE finds voting weights (\mathbf{P}^*) considering higher
374 weights on hard-to-classify instances, it is expected that it reduces bias compared to
375 Bagging. The variance would be the same as that of bagging because the averaging
376 is still involved. Table 5 presents the bias and variance decomposition for three
377 scenarios.

378 From data A and B, we can see WAVE attained its good performance by reducing
379 the variance compared to a single pruned trees. Its advantage over Bagging can be
380 attributed to the reduction of bias. Random Forest also shows similar behavior
381 with Bagging and WAVE in the sense that it is a variance reducer. For data C
382 (chessboard), WAVE demonstrates superb result by reducing both bias and variance.
383 The poor result by Boosting and Random Forest is due to high bias and variance.

384 5. Concluding Remarks

385 As a weighted voting method, Boosting also assigns more weights on hard-to-
386 classify instances. The instance weight vector \mathbf{Q}^* keeps updated as the classifiers
387 are added in the ensemble. The hard-to-classify instances in Boosting are defined as
388 those misclassified by the previous classifier during the ensemble formation. Thus,
389 the instance weight vector \mathbf{Q}^* changes dramatically as the classifier gives very dif-
390 ferent predictions at the next stage. The classifier weight vector \mathbf{P}^* is proportional
391 to the weighted accuracy of the corresponding classifier using the instance weight
392 \mathbf{Q}^* . In Boosting, the two weight vectors are simultaneously found as the ensemble
393 grows.

394 In WAVE, on the other hand, the two weight vectors are sought after the en-
395 semble formation is completed. The hard-to-classify instances are defined as those
396 frequently misclassified by classifiers in the ensemble. The weight vector \mathbf{Q}^* is de-
397 signed to be sought in conjunction with \mathbf{P}^* , and vice versa. This implies that the
398 classifiers with higher weights in \mathbf{P}^* have more influences in choosing the weight
399 vector \mathbf{Q}^* . Similarly, the instances with higher weights in \mathbf{Q}^* play more important
400 role in determining the weight vector \mathbf{P}^* . We showed that the two weight vectors
401 can be found directly from the performance matrix \mathbf{X} as defined in Section 2.1.
402 The final prediction of the ensemble is obtained by the voting using the final weight

403 vector of classifiers.

404 For the performance comparison between WAVE and other ensemble methods,
405 we implemented an experiment using 28 real or simulation data sets. We demon-
406 strated that WAVE consistently outperforms Bagging that uses simple majority
407 voting scheme. Although not significant, we observed a tendency that WAVE may
408 perform better than Boosting. WAVE showed compatible performance with Ran-
409 dom Forest. When we compared ensemble methods and pruned trees, WAVE showed
410 the most consistent and significant improvement over the pruned trees.

411 Instances near boundaries of classes are in general hard to classify. Simple ma-
412 jority voting has a low prediction accuracy on these instances because the voting
413 result is usually a tough call. Since WAVE gives higher weight on the classifiers
414 that can perform better on these instances, it improves the simple majority voting
415 scheme significantly. This leads to the reduction of bias in classification. Via bias-
416 variance decomposition, we verified that WAVE reduces the classification bias while
417 maintaining low variance.

418 The WAVE voting algorithm is combined with bootstrap aggregation scheme
419 in this article. However, it can be used in conjunction with any other aggregation
420 schemes. We will explore this direction further in the future.

421 **Acknowledgements**

422 The authors gratefully acknowledge the many helpful suggestions of anonymous
423 reviewers. The authors are also grateful to Professor Chong Jin Park of California
424 State University at San Diego for the contribution of the weighted voting scheme.
425 This work was partly supported by Basic Science Research program through the
426 National Research Foundation of Korea(NRF) funded by the Ministry of Education,
427 Science, and Technology(2009-0072019).

428 **References**

- 429 Asuncion, A., Newman, D. J., 2007. *UCI Machine Learning Repository*. University
430 of California, Irvine, School of Information and Computer Science, [http://www.](http://www.ics.uci.edu/~mllearn/MLRepository.html)
431 [ics.uci.edu/~mllearn/MLRepository.html](http://www.ics.uci.edu/~mllearn/MLRepository.html).
- 432 Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algo-
433 rithms: bagging, boosting, and variants. *Machine Learning* **36**, pp. 105–139.
- 434 Breiman, L., 1996. Bagging predictors. *Machine Learning* **24**, pp. 123–140.
- 435 Breiman, L., 1998. Arcing classifiers. *The Annals of Statistics* **26**, pp. 801–824.
- 436 Breiman, L., 2001. Random forests. *Machine Learning* **45**, pp. 5–32.
- 437 Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., 1984. *Classification and*
438 *Regression Trees*. Chapman & Hall, New York.
- 439 Chandra, A., Yao, X., 2006. Evolving hybrid ensembles of learning machines for
440 better generalisation. *Neurocomputing* **69**, pp. 686–700.
- 441 Dietterich, T. G., 2000. *Ensemble methods in machine learning*. Springer, Berlin.
- 442 Freund, Y., Schapire, R., 1996. Experiments with a new boosting algorithm. In: *Pro-*
443 *ceedings of the Thirteenth International Conference on Machine Learning*. Morgan
444 Kaufmann, pp. 148–156.
- 445 Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: a statis-
446 tical view of boosting. *The annals of statistics* **28**, pp. 337–407.
- 447 Geman, S., Bienenstock, E., Doursat, R., 1992. Neural networks and the
448 bias/variance dilemma. *Neural Computation* **4**, pp. 1–48.

- 449 Hansen, L. K., Salamon, P., 1990. Neural network ensembles. *IEEE Transactions*
450 *on Pattern Analysis and machine Intelligence* **12**, pp. 993–1001.
- 451 Hastie, T., Tibshirani, R., Friedman, J., 2001. *The Elements of Statistical Learning:*
452 *Data Mining, Inference, and Prediction*. Springer–Verlag, New York.
- 453 Heinz, G., Peterson, L. J., Johnson, R. W., Kerk, C. J., 2003. Exploring relationships
454 in body dimensions. *Journal of Statistics Education* **11**, [http://www.amstat.](http://www.amstat.org/~publications/jse/v11n2/datasets.heinz.html)
455 [org/~publications/jse/v11n2/datasets.heinz.html](http://www.amstat.org/~publications/jse/v11n2/datasets.heinz.html).
- 456 Ho, T. K., 1998. The random subspace method for constructing decision forests.
457 *IEEE Transactions of Pattern Analysis and Machine Intelligence* **20**, pp. 832–
458 844.
- 459 Hothorn, T., Lausen, B., 2003. Double-bagging: combining classifiers by bootstrap
460 aggregation. *Pattern Recognition* **36**, pp. 1303 – 1309.
- 461 Hothorn, T., Lausen, B., 2005. Bundling classifiers by bagging trees. *Computational*
462 *Statistics & Data Analysis* **49**, pp. 1068 – 1078.
- 463 Kim, H., Loh, W.-Y., 2001. Classification trees with unbiased multiway splits. *Jour-*
464 *nal of the American Statistical Association* **96**, pp. 589–604.
- 465 Kim, H., Loh, W.-Y., 2003. Classification trees with bivariate linear discriminant
466 node models. *Journal of Computational and Graphical Statistics* **12**, pp. 512–530.
- 467 Kohavi, R., Wolpert, D. H., 1996. Bias plus variance decomposition for zero–one loss
468 functions. In: *Proceedings of the Thirteenth International Conference on Machine*
469 *Learning*. Morgan Kaufmann, pp. 275–283.

- 470 Kong, E. B., Dietterich, T. G., 1995. Error-correcting output coding corrects bias
471 and variance. In: *Proceedings of the Twelfthth International Conference on Ma-*
472 *chine Learning*. Morgan Kaufmann, pp. 313–321.
- 473 Kuncheva, L., 2005. Diversity in multiple classifier systems. *Information Fusion* **6**,
474 pp. 3–4.
- 475 Lam, L., Suen, C. Y., 1997. Application of majority voting to pattern recognition:
476 An analysis of its behavior and performance. *IEEE Transactions on Pattern Anal-*
477 *ysis and Machine Intelligence* **27**, pp. 553–568.
- 478 Loh, W.-Y., 2009. Improving the precision of classification trees. *The Annals of*
479 *Applied Statistics* **3**, 1710 – 1737.
- 480 Maslov, I. V., Gertner, I., 2006. Multi-sensor fusion: An evolutionary algorithm
481 approach. *Information Fusion* **7**, 304–330.
- 482 Masnadi-Shirazi, H., Vasconcelos, N., 2011. Cost-sensitive boosting. *IEEE Transac-*
483 *tions on Pattern Analysis and Machine Intelligence* **33**, 294–309.
- 484 Rahman, A. F. R., Alam, H., Fairhurst, M. C., 2002. Multiple classifier combination
485 for character recognition: Revisiting the majority voting system and its variations.
486 *Document Analysis systems V, Lecture Notes in Computer Science* **2423**, pp. 167–
487 178.
- 488 Rodriguez, J. J., Kuncheva, L. I., Alonso, C. J., 2006. Rotation forest: A new
489 classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine*
490 *Intelligence* **28**, pp. 1619–1630.
- 491 Rokach, L., 2009. Taxonomy for characterizing ensemble methods in classification

- 492 tasks: a review and annotated bibliography. *Computational Statistics and Data*
493 *Analysis* **53**, pp. 4046–4072.
- 494 Schapire, R. E., 1990. The strength of weak learnability. *Machine Learning* **5**, pp.
495 197–227.
- 496 Statlib, 2010. Datasets archive. Carnegie Mellon University, Department of Statis-
497 tics, <http://lib.stat.cmu.edu>.
- 498 Terhune, J. M., 1994. Geographical variation of harp seal underwater vocalisations.
499 *Canadian Journal of Zoology* **72**, pp. 892–897.
- 500 Webb, G. I., 2000. Multiboosting: A technique for combining boosting and wagging.
501 *Machine Learning* **40**, pp. 159–196.
- 502 Zhang, C. X., Zhang, J. S., 2008. A local boosting algorithm for solving classification
503 problems. *Computational Statistics and Data Analysis* **52**, pp. 1928–1941.
- 504 Zhu, J., Zou, H., Rosset, S., Hastie, T., 2009. Multi-class adaboost. *Statistics and*
505 *Its Interface* **2**, pp. 349–360.

Table 2: Dataset description.

Data	Description	# instances	# classes	# variables	Source
Bcw	Breast Cancer Wisconsin	699	2	10	UCI
Bld	Liver Disorders	345	2	6	UCI
Bod	Body Dimension	507	2	24	Heinz et al. (2003)
Bos	Boston Housing	506	3	14	UCI
Cmc	Contraceptive Method Choice	1473	3	9	UCI
Col	Horse Colic	368	3	27	UCI
Cre	Credit Approval	690	2	15	UCI
Cyl	Cylinder Bands	540	2	35	UCI
Der	Dermatology	366	6	33	UCI
Dia	Diabetes	532	2	7	Loh (2009)
Fis	Fish	159	7	7	Kim and Loh (2003)
Ger	Germa Credit	1000	2	20	UCI
Gla	Glass	214	6	9	UCI
Hea	Statlog (Heart)	270	2	13	UCI
Int	Chessboard	1000	2	10	Section 3
Ion	Ionosphere	351	2	34	UCI
Iri	Iris	150	3	4	UCI
Lak	Lakes	259	6	16	Loh (2009)
Led	LED Display Domain	6000	10	7	UCI
Pid	Pima Indians Diabetes	768	2	8	UCI
Pov	Poverty	97	6	6	Kim and Loh (2001)
Sea	Vocalisations of Harp Seals	3000	3	7	Terhune (1994)
Spe	SPECTF Heart	267	2	44	UCI
Usn	Usnews	1302	3	27	Statlib (2010)
Veh	Statlog (Vehicle Silhouettes)	946	4	18	UCI
Vol	Volcano	1521	6	6	Loh (2009)
Vot	Congressional Voting Records	435	2	16	UCI
Vow	Vowel Recognition	990	11	10	UCI

Table 3: Accuracy of 28 dataset. The value given is the mean accuracy (in percentage) of 100 repeated cross-validations (10-fold each). The ensemble size is 64. The first and the second best results are highlighted in bold type. Paired t-test and Wilcoxon signed rank test statistics are given at the bottom lines with the same statistics excluding ‘Int’ data given in the parenthesis.

Data	WAVE	Bagging	Boosting	Random Forest	Single Tree
Bcw	96.24	96.17	96.16	97.08	94.50
Bld	72.15	72.13	72.31	72.51	68.15
Bod	93.50	93.37	97.82	94.20	91.43
Bos	77.67	77.59	78.22	78.33	74.10
Cmc	56.13	56.12	54.65	55.03	55.04
Col	70.89	70.78	68.27	71.23	66.06
Cre	86.26	86.25	85.09	86.35	85.21
Cyl	74.88	74.71	75.11	73.72	64.61
Der	95.35	94.97	96.92	97.55	93.36
Dia	76.16	76.18	75.62	76.14	74.78
Fis	82.47	82.46	82.32	81.07	80.31
Ger	75.62	75.57	75.06	74.67	72.52
Gla	75.70	75.68	77.93	76.79	70.17
Hea	82.71	82.76	78.97	83.30	77.66
Int	82.26	79.62	63.06	53.40	80.01
Ion	91.08	90.94	92.37	92.65	87.28
Iri	95.53	95.35	95.17	95.61	95.50
Lak	42.43	42.37	40.25	42.29	36.73
Led	71.69	71.42	70.60	72.52	68.19
Pid	78.01	78.05	76.60	77.89	74.90
Pov	67.30	67.39	61.45	62.93	65.57
Sea	59.38	59.23	64.59	60.73	58.09
Spe	80.47	80.54	79.77	80.76	73.81
Usn	75.05	75.00	71.69	73.58	73.17
Veh	71.46	71.39	74.64	71.83	68.18
Vol	88.38	88.12	88.67	87.44	87.17
Vot	95.43	95.44	95.51	95.70	94.29
Vow	70.78	70.62	94.21	72.73	60.14
# highlights	16	10	12	17	1
# worst	0	0	5	1	22
t-statistics	6.87 (6.73)	6.32 (6.57)	2.25 (3.10)	1.96 (5.69)	
Wilcoxon-statistics	406 (378)	403 (377)	349 (348)	365.5 (365.5)	

Table 4: Three artificial dataset for bias and variance decomposition.

Data	Class label	Predictor variables
A	$Y=\{0, 1\}$	$X_i \sim N(0, 1.5)$ when $Y = 0$, for $i = 1, 2$ $X_i \sim N(0, 5)$ and $\sum_{i=1}^2 X_i > 5$ when $Y = 1$, for $i = 1, 2$
B	$Y=\{0, 1\}$	$X_i \sim \text{uniform}(-i, +i)$, $i = 1, \dots, 10$, $p = \frac{\exp[\cos(x_1+x_2+x_3)+\sin(x_4+x_5+x_6)]}{1+\exp[\cos(x_1+x_2+x_3)+\sin(x_4+x_5+x_6)]}$ $Y \sim \text{binomial}(1, p)$
C	$Y=\{0, 1\}$	chessboard as in Section 3

Table 5: Bias and variance decomposition.

Methods	Data A		Data B		Data C	
	Bias ²	Variance	Bias ²	Variance	Bias ²	Variance
Single Tree	0.096	0.053	0.222	0.239	0.196	0.242
Bagging	0.104	0.032	0.221	0.206	0.193	0.234
Wave	0.103	0.032	0.219	0.206	0.184	0.231
Boosting	0.085	0.056	0.210	0.220	0.222	0.237
Random Forest	0.107	0.028	0.226	0.209	0.232	0.246