

# Combining Decision Trees using Systematic Patterns

Hyunjoong Kim

Department of Statistics  
University of Tennessee  
Knoxville, TN 37996  
hjkim@utk.edu

## Abstract

Tree ensemble or combining methods that use re-sampling technique have been highlighted recently in Statistical classification and Data mining. In this paper, we propose a new ensemble method in decision trees that utilizes systematic patterns of classification. The new method improved the prediction accuracy of a single decision tree algorithm. It is also observed that this method performs reasonably well with fewer number of re-samples compared to the popular Bagging or Boosting methods. An experiment with real dataset is carried out to see the performance of the new method.

*Key words and phrases:* Bagging, Boosting, Decision trees, Ensemble of trees, Stacking, Voting

## 1 INTRODUCTION

Let  $\mathbf{y} = (y_1, \dots, y_N)^T$  denote the class variable observed on  $N$  instances. The observed data matrix is  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$ , where  $\mathbf{x}_i = (x_1, \dots, x_p)_i$  denote the observed vector for  $i$ th instance. Let  $\mathcal{X}$  be a  $p$ -dimensional measurement space containing all possible values of  $x_1, \dots, x_p$ . Let  $\mathcal{X}_s$  be the disjoint partitions of  $\mathcal{X}$  such that  $\mathcal{X} = \bigcup_s \mathcal{X}_s$ . A decision tree learning algorithm finds  $\mathcal{X}_s$  by recursively partitioning the data space into sub-regions within which the distribution of classes is more homogeneous. It provides a tree-structure according to the split taking the form of  $\{\text{Is } x_i \leq c?\}$  for ordered variables or of  $\{\text{Is } x_j \in C?\}$  for nominal variables, where  $c$  is an arbitrary point in the range of  $x_i$  and  $C$  is in the range of all subsets of categories on  $x_j$ . By recursively choosing splits that maximize the homogeneity of the classes of the node, one can grow a sequence of trees. Tree growing is continued until the stopping criteria are satisfied. A decision tree is

then selected by pruning back the tree based on certain criteria. Predictions are determined by the distribution of classes in each terminal node.

Decision tree algorithms such as C4.5 (Quinlan, 1993) or CART (Breiman, Friedman, Olshen and Stone, 1984) have attracted many researchers because they provide intuitive interpretations. In recent years, there has been growing interest in learning algorithms that achieve high accuracy. In order to increase the prediction accuracy of decision trees, ensemble or combining methods have been highlighted in both Computer Science and Statistics. Many researchers have investigated the technique of combining predictions of individually trained classifiers when classifying future instances. In fact, the idea of combining predictions of different models began to receive attentions since Wolpert (1992) proposed stacking algorithm in the context of regression. Breiman (1996b) developed stacked regression algorithm by adopting backward elimination and ridge regression. In the context of classification, methods such as Bagging (Breiman, 1996a) and Boosting (Freund and Schapire, 1996) have been shown to be very successful in improving the accuracy of the classification algorithm; see Bauer and Kohavi (1999), Opitz (1999), and Dietterich (2000) for comparison results. These methods rely on re-sampling techniques to obtain different learning samples for each of the decision trees. Many ensemble (or combining) methods are further developed. Some important contributions include Schapire and Singer (1999), Breiman (2000), and Web (2000).

The goal of this paper is to introduce a new ensemble method in decision trees. We propose a method that utilizes systematic patterns of classification results. It is shown that this method improves the prediction accuracy of a single decision tree algorithm. We think that the proposed method requires fewer number of re-samples to keep the comparable accuracy with the popular Bagging or Boosting methods. Although the number of re-samples via bootstrap is generally considered unimportant as computational resources continue to expand, we believe it is still important to derive an efficient ensemble method as the size of dataset increases rapidly.

The rest of the paper is structured as follows. A review for existing methods is given in Section 2. The new method is described in Section 3 and experiment results with many dataset are presented in Section 4. Section 5 concludes the paper with closing remarks.

## 2 METHODOLOGY REVIEW

Typical approach of ensemble methods in classification is first to generate many different learning samples via re-sampling technique. Let  $\mathcal{L}$  be the training data. Then  $b$  re-samples denoted by  $\mathcal{L}_1, \dots, \mathcal{L}_b$  are generated from  $\mathcal{L}$ . We then apply the classification algorithm and construct classifiers on these samples. If an algorithm

$\mathcal{A}$  is used,  $b$  classifiers denoted by  $\mathcal{T}_1, \dots, \mathcal{T}_b$  are constructed based on  $\mathcal{L}_1, \dots, \mathcal{L}_b$ , respectively. Note that  $\mathcal{T}_1, \dots, \mathcal{T}_b$  are constructed by the same algorithm  $\mathcal{A}$ . Finally, we summarize the performance of the classification algorithm by combining the prediction results of each classifier.

An ensemble method consists of three important components. The first component is the selection of re-sampling technique. The number of participating classification algorithms is the second, and the combining method to get overall prediction is the third. Bagging uses bootstrap technique for the first and majority voting for the third. It uses only one classification algorithm for the second component because its main purpose is to increase the accuracy of the specific classification algorithm.

Mojirsheibani (1999) proposed a method to combine several classification algorithms without re-sampling. The main focus of this method is to combine different algorithms rather than to combine prediction results of one algorithm. As an illustration, several classification algorithms  $\mathcal{A}_1, \dots, \mathcal{A}_b$  are used to generate classifiers  $\mathcal{T}_1, \dots, \mathcal{T}_b$ , respectively. The prediction results based on  $\mathcal{T}_1, \dots, \mathcal{T}_b$  are used to predict the learning or evaluation data. Note here that this method does not re-sample the learning dataset  $\mathcal{L}$ . This method used the idea called “discretization” to combine the predictions. Since this method requires several classification algorithms to be applied on the same learning sample, the method is not appropriate for combining predictions of the specific algorithm under our interest. Mojirsheibani (1999)’s algorithm does not have the first component while several classification algorithms are required for the second. “Discretization” of the predictions are utilized as the third component to get overall prediction of each instance. We call this method the M-method.

For the stacking algorithm, Wolpert (1992) and Breiman (1996b) used the leave-one-out data which they called level-one data. That is, the stacking algorithm uses leave-one-out technique for the first component. It also requires several regression methods for the second as in M-method, then combining predictions is achieved by ridge regression based on the prediction results of all participating regression methods. We note that the stacking method is designed for the regression problem.

Table 1 summarizes the differences of the existing and the new ensemble methods. The new method is discussed in Section 3.

### 3 THE NEW COMBINING METHOD

In the new method, we use either cross-validation or bootstrap for the first component. Since our main objective is to combine the predictions of the particular classification algorithm, we will consider only one classification method as the

Table 1: Differences of ensemble methods.

Method	re-sampling technique	# learning algorithms	combining method
Bagging	bootstrap	1	majority voting
M-method	no re-sample	several	discretization
Stacking	leave-one-out	several	ridge regression
New method	cross-validation or bootstrap	1	discretization

second component. Finally, we adopt the “discretization” technique for the third component.

Our method utilizes systematic patterns of predictions from the classifiers in each learning sample. For an easy illustration of the new method, we first provide the simplest implementation of the method as follows.

- (a). Let  $\mathcal{L}$  be the learning sample and  $\mathcal{L}_1$  and  $\mathcal{L}_2$  be two random re-samples of  $\mathcal{L}$ .
- (b). Construct a decision tree on  $\mathcal{L}_1$  and  $\mathcal{L}_2$  and call them  $\mathcal{T}_1$  and  $\mathcal{T}_2$ , respectively.  $\mathcal{T}_1$  and  $\mathcal{T}_2$  are generated by the same algorithm, thus not independent.
- (c). Use  $\mathcal{L}$  to pass down the trees  $\mathcal{T}_1$  and  $\mathcal{T}_2$  to get the predicted classes for each instance in  $\mathcal{L}$ .
- (d). For binary class (say 0 and 1) data, there are four possible classification patterns from  $\mathcal{T}_1$  and  $\mathcal{T}_2$  such as (0, 0), (0, 1), (1, 0), (1, 1), where the first element in each pair denotes the predicted class by  $\mathcal{T}_1$  and the second by  $\mathcal{T}_2$ .
- (e). Among observations in the learning sample with predicted class (0, 0), we count the number of instances with actual class 0 and with actual class 1. Repeat this for all patterns. As an example, we provide an arbitrary pattern table in Table 2.
- (f). If the majority of samples with predictions (0, 0) have actual class 1 as in the example, we find a systematic pattern that  $\mathcal{T}_1$  and  $\mathcal{T}_2$  misclassify the instance. In this case, we decide to predict any instance with predicted class (0, 0) to be class 1 because we want to correct a systematic pattern of misclassification.
- (g). Find the systematic classification patterns for all possible combinations as described in (f).
- (h). Predict a future case based on the classification pattern found in (g).

Table 2: An example pattern table.

predicted class	actual class	count
(0, 0)	0	4
	1	30
(0, 1)	0	9
	1	3
(1, 0)	0	9
	1	8
(1, 1)	0	2
	1	35
total		100

For simplicity, in the above, we considered only two re-samples of  $\mathcal{L}$  and two classes. The new method can be used on several re-sampled learning samples as long as there are enough data to explore all the combinations of predictions and classes.

Our generalized algorithm is summarized in Algorithm 1.

**Algorithm 1** *The new combining method using systematic patterns of classification.*

1. Generate  $b$  re-samples from the learning data  $\mathcal{L}$  to have  $\mathcal{L}_1, \dots, \mathcal{L}_b$ .
2. The algorithm under our interest, say a decision tree, is applied to  $\mathcal{L}_1, \dots, \mathcal{L}_b$  and construct  $b$  classifiers  $\mathcal{T}_1, \dots, \mathcal{T}_b$ .
3. Use  $\mathcal{T}_1, \dots, \mathcal{T}_b$  to get the predictions on  $\mathcal{L}$ . Note that the predictions are not acquired for  $\mathcal{L}_1, \dots, \mathcal{L}_b$ . Also note that Bagging algorithm does not require this step.
4. Let  $\hat{f}_m(\mathbf{x}_i)$  be the prediction made by  $\mathcal{T}_m, m = 1, \dots, b$  for  $i$ th instance in  $\mathcal{L}$ . Then the prediction matrix for learning data can be constructed as follows.

$$\begin{bmatrix} \hat{f}_1(\mathbf{x}_1), & \dots, & \hat{f}_b(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \hat{f}_1(\mathbf{x}_N), & \dots, & \hat{f}_b(\mathbf{x}_N) \end{bmatrix}$$

5. Let  $\mathcal{P}$  be an index set such that

$$\mathcal{P} = \{\omega; \omega = (k_1, \dots, k_b), \text{ where } k_m \in (1, \dots, K), m = 1, \dots, b\}.$$

Table 3: A generalized pattern table.

$\omega$	actual class	$V(\omega, \cdot)$
$(1, \dots, 1)$	1	$V((1, \dots, 1), 1)$
	$\vdots$	$\vdots$
	$K$	$V((1, \dots, 1), K)$
$\vdots$	$\vdots$	$\vdots$
	1	$V(k_1, \dots, k_b), 1)$
	$\vdots$	$\vdots$
$(k_1, \dots, k_b)$	$K$	$V(k_1, \dots, k_b), K)$
	$\vdots$	$\vdots$
	$\vdots$	$\vdots$
$(K, \dots, K)$	1	$V((K, \dots, K), 1)$
	$\vdots$	$\vdots$
	$K$	$V((K, \dots, K), K)$

Define a pattern matching score function

$$V(\omega^*, k) = \sum_{i=1}^N I[(\hat{f}_1(\mathbf{x}_i), \dots, \hat{f}_b(\mathbf{x}_i)) = \omega^*] \times I(\mathbf{y}_i = k), \text{ where } \omega^* \in \mathcal{P}.$$

- Find the systematic patterns by constructing the pattern table based on the predictions of  $\mathcal{L}$ . The pattern table lists the elements of  $\mathcal{P}$  and its corresponding pattern matching score  $V(\cdot, \cdot)$ . The pattern table is shown in Table 3. The number of rows in the pattern table is  $K^{b+1}$  where  $K$  is the number of classes. We note that in practice there are many empty rows in the pattern table.
- To get the predictions on a future instance or a validation data, use Algorithm 2.

**Algorithm 2** Predict future cases.

- Let  $\mathbf{x}_o$  be a future observation or an evaluation dataset point.
- Put  $\mathbf{x}_o$  to the classifiers  $\mathcal{T}_1, \dots, \mathcal{T}_b$  and get the predictions  $\hat{f}_1(\mathbf{x}_o), \dots, \hat{f}_b(\mathbf{x}_o)$ .
- Identify  $\omega^*$  such that  $\omega^* = (\hat{f}_1(\mathbf{x}_o), \dots, \hat{f}_b(\mathbf{x}_o))$ .
- $\mathbf{x}_o$  is assigned to class  $k^*$  if  $k^* = \arg \max_k \{V(\omega^*, k)\}$

When there is a tie in the pattern matching score function, we use Bagging algorithm to avoid ties. We can use  $v$ -fold cross-validation or bootstrap for generating re-samples. While the accuracy of the new method is comparable, we think

that the proposed method requires smaller number of re-samples than the usual Bagging method which typically uses 50 re-sampled training data.

Although our method can be applied to any classification algorithm, we consider the decision trees in the discussion. The current implementation of the new method is based on 4-fold cross-validation of the learning sample. That is,  $\mathcal{L}$  is randomly partitioned into  $\mathcal{L}_1$ ,  $\mathcal{L}_2$ ,  $\mathcal{L}_3$ , and  $\mathcal{L}_4$ . Let  $\mathcal{L}^{(1)} = \{\mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4\}$ ,  $\mathcal{L}^{(2)} = \{\mathcal{L}_1, \mathcal{L}_3, \mathcal{L}_4\}$ , and so forth. Then we apply a decision tree algorithm on  $\mathcal{L}^{(1)}$ ,  $\mathcal{L}^{(2)}$ ,  $\mathcal{L}^{(3)}$ , and  $\mathcal{L}^{(4)}$  to get  $\mathcal{T}_1$ ,  $\mathcal{T}_2$ ,  $\mathcal{T}_3$ , and  $\mathcal{T}_4$ , respectively. Next, we use  $\mathcal{L}$  to pass down the trees  $\mathcal{T}_1$ ,  $\mathcal{T}_2$ ,  $\mathcal{T}_3$ , and  $\mathcal{T}_4$  to get the predictions of each instance in  $\mathcal{L}$ . By pattern matching the classification results of these four trees, we can predict a future case. The comparison of this implementation with the existing methods using real data is given in Section 4.

## 4 REAL DATA EXPERIMENT

In this section, we compare our proposed method with existing combining methods. We made comparisons using 11 dataset in terms of misclassification error rates. Table 4 lists the brief data descriptions. The estimated error rates are based on 10-fold cross-validation. For the comparison purpose, the 4-fold cross-validation scheme of the current implementation is performed within each 10-fold cross-validated learning sample. Our proposed combining method is implemented using C4.5 (Quinlan, 1993). Non-Bagging C4.5 and Bagging C4.5 are considered for comparison. Salford Systems CART (Steinberg and Colla, 1997) with Bagging option and Arcing option are also compared as well as the non-Bagging option.

Table 5 lists the comparison results. We can observe that the proposed method based on 4-fold cross-validation of learning sample gains some accuracy over the non-bagged decision trees. However, the gain by the current implementation is not as large as the existing combining methods over the non-bagged decision trees. For comparable accuracy with the existing methods, larger number of re-samples may be required for the new method.

## 5 CONCLUDING REMARKS

We introduced a new combining method. The new method utilizes systematic patterns of predictions from the decision trees in each learning sample. It is shown that this method improved the prediction accuracy of the baseline decision tree classifier, but not as much as the Bagging method did. We believe this is due to the small number of re-samples we adopted in the current implementation of the method. We used 4 re-samples for the experiment while the Bagging method used 50 re-samples. If we increase the number of re-samples for the new method,

Table 4: Dataset descriptions. Column  $N$  denotes the number of cases,  $J$  the number of classes,  $m$  the number of numerical variables, and  $c$  the number of categorical variables.

Code	Description	Source	$N$	$J$	$m$	$c$
BCW	Breast Cancer Wisconsin	UCI	683	2	9	0
DNA	StatLog DNA	UCI	2000	3	0	60
HEA	StatLog heart disease	UCI	270	2	7	6
LED	LED display	UCI	2000	10	0	7
PID	PIMA Indian disorders	UCI	532	2	7	0
SAT	StatLog satellite image	UCI	4435	6	36	0
SEG	Image segmentation	UCI	2310	7	19	0
THY	Thyroid disease	UCI	3772	3	6	15
VEH	StatLog vehicle silhouette	UCI	846	4	18	0
VOT	Congressional voting records	UCI	435	2	0	16
WAV	Waveform	UCI	600	3	21	0

UCI: University of California, Irvine, Repository of Machine Learning Databases

Table 5: Mean ten-fold cross-validation error rates and the number of trees generated in each method. We used the default options of the softwares.

Method	Mean error rate	# trees generated
Pattern matching using C4.5	0.139	4
Non-Bagging C4.5	0.146	1
Non-Bagging CART	0.150	1
Bagging C4.5	0.126	50
Bagging CART	0.126	50
Arcing CART	0.128	50

a comparable accuracy with the popular Bagging or Boosting methods would be achieved. We are not sure at this point how many re-samples are sufficient for the comparable accuracy. We expect it is still significantly smaller than 50. We observe the new method can be extended to several directions.

First, the new method can be applied to the level-one data scheme of the stacking algorithm. In this case, small number of re-samples are generated within each level-one data to give several predictions for each instance. After repeating this process for all training data, we use the pattern matching idea. Second, we can also utilize stacking algorithm to get the set of  $y'$ s and predictions using the tree sequence considered in pruning. For each sequence of trees, we will get the predictions for all instance. Then the pattern matching idea can be used there. Third, Bagging algorithm is used up to the point where we have set of  $y'$ s and predictions, then derive a method to get coefficients of predictions. We plan to pursue these issues in the future.

## References

- Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Machine Learning* **36**: 105–139.
- Breiman, L. (1996a). Bagging predictors, *Machine Learning* **24**: 123–140.
- Breiman, L. (1996b). Stacked regressions, *Machine Learning* **24**: 49–64.
- Breiman, L. (2000). Randomizing outputs to increase prediction accuracy, *Machine Learning* **40**: 229–242.
- Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984). *Classification and Regression Trees*, Chapman & Hall, New York.
- Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, *Machine Learning* **40**: 139–157.
- Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm, *Proceedings of the Thirteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 148–156.
- Mojirsheibani, M. (1999). Combining classifiers via discretization, *Journal of the American Statistical Association* **94**: 600–609.
- Opitz, D. (1999). Popular ensemble methods: An empirical study, *Journal of Artificial Intelligence Research* **11**: 169–198.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo.

Schapire, R. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions, *Machine Learning* **37**: 297–336.

Steinberg, D. and Colla, P. (1997). *CART—Classification and Regression Trees: A Supplementary Manual for Windows*, Salford Systems Inc., San Diego.

Web, G. (2000). Multiboosting: A technique for combining boosting and wagging, *Machine Learning* **40**: 159–196.

Wolpert, D. (1992). Stacked generalization, *Neural Networks* **5**: 241–259.