Improving Bagging Predictors

Hyunjoong Kim[1], Dongjun Chung[2]

Abstract

Ensemble method has been known as one of the most powerful classification tools that can improve prediction accuracy. Ensemble method also has been understood as "perturb and combine" strategy. Many studies have tried to develop ensemble methods by improving perturbation. In this paper, we propose two new ensemble methods that improve combining, based on the idea of pattern matching. In the experiment with simulation data and with real dataset, the proposed ensemble methods performed better than bagging. The proposed ensemble methods give the most accurate prediction when the pruned tree was used as the base learner.

KEY WORDS: Pattern matching; Ensemble; Bagging; Classification tree

1. Introduction

Since its first introduction to the world, ensemble method has been proved that it improves the prediction accuracy dramatically. Breiman (1998) referred ensemble as the "perturb and combine" strategy. Ensemble method generates multiple versions of predictions by perturbing the training dataset, then combines these multiple versions of predictions into a single predictor. Bagging (Breiman, 1996) and boosting (AdaBoost.M1; Freund and Schapire, 1996) are most successful ones among many ensemble methods. Bagging perturbs the dataset by using bootstrap sampling, then combines the predictions from these perturbed dataset with unweighted majority voting. Boosting increases weight on the misclassified observations through iterations. The observations are again classified using the modified weight. Then it combines these predictions with weighted majority voting by giving more weight on the iterations with more accuracy. Researchers have developed ensemble methods mostly by improving perturbation. In this paper, we propose two ensemble methods that can give more accurate predictions than bagging by improving aggregation strategy.

2. Proposed Ensemble Methods

2.1. Pattern Matching

1) Assistant Professor, Department of Applied Statistics, Yonsei University, 134 Sinchon-dong, Seodaemun-gu, Seoul 120-749, Korea. e-mail: hkim@yonsei.ac.kr
2) M.A. Candidate, Department of Applied Statistics, Yonsei University, 134 Sinchon-dong, Seodaemun-gu, Seoul 120-749, Korea. e-mail: soonceagain@yonsei.ac.kr

Mojirsheibani (1999) proposed a new combining method using pattern matching. Let $\{1,2,\ldots,K\}$ be the set of classes, $\{C_1, C_2, \ldots, C_M\}$ the set of classifiers, X_i and Y_i the vector and the class membership of i^{th} observation, $C_m(X_i)$ the prediction of X_i made by the classifier C_m and X_0 a future observation. Then, the method by Mojirsheibani (1999) assigns X_0 to the group k^*, if

$$k* = \arg\max_{1 \le k \le K} \sum_{i=1}^{n} \prod_{j=1}^{M} I\{C_{n,j}(X_i) = C_{n,j}(X_0)\} I(Y_i = k).$$

Note that the classifier C_i, $i = 1,2,\ldots,M$, are not the result of perturbation, but independent classifiers. This method can be expressed as an algorithm in the following way:

Algorithm: Pattern Matching

1. Predict the training dataset, X_i, using the classifiers $C_1, C_2, \ldots, C_M$.
2. Predict a new observation, X_0, using the classifiers $C_1, C_2, \ldots, C_M$.
3. Find the same pattern as X_0 of step 2 from the pattern of X_i of step 1.
4. Count the number of each group among the observations with the patterns of step 3.
5. Assign the new observation, X_0, to the group that is counted most in step 4.

Mojirsheibani (1999) proved that pattern matching asymptotically followed the accuracy of the most accurate classifiers under consideration. For example, when the dataset satisfies the normality assumption, linear discriminant analysis (LDA) gives most accurate prediction. In contrast, when the normality assumption is severely violated, k-nearest-neighbor predicts more accurately than LDA. If LDA and k-nearest-neighbor are combined with pattern matching, the combined prediction has the accuracy of the better classifier among LDA and k-nearest-neighbor.

In pattern matching, the most important parameter is the number of classifiers, M, because M controls the sensitivity of combining. As M becomes larger, there exists more types of patterns and the predictions must be made with more complex pattern structure. However, if M is too large compared to the size of the sample, there are many empty cells in the pattern structure. Obviously, the observation in empty cell in the pattern matrix cannot be predicted. Therefore, Mojirsheibani (1999) suggested to choose M = log(n), where n is the size of the training dataset.

2.2. Proposed Methods

Pattern matching can be applied to bagging or boosting predictions. In this way, the

combined prediction is expected to obtain the accuracy of the best bagging or boosting prediction. Only bagging is considered for the reason discussed in the next section. Let B is the number of bootstrap samples in bagging. If we try to use pattern matching on B prediction results, it might be unsuccessful because the number of classifiers B is much larger than the suggested log(n). To reduce the number of classifiers for pattern matching, we propose two schemes. First, we conduct several baggings on subsets of bootstrap samples. The number of subsets is maintained roughly as log(n). Then the pattern matching is performed on the bagged classifiers. Second, we can conduct pattern matching on subsets of bootstrap samples. The size of samples in each subset is roughly maintained as log(n). Then the bagging is performed on the pattern matched classifications. The algorithms of the proposed methods are given below. After all, the proposed methods would need similar number of bootstrap samples as bagging.

Algorithm 1. bagged pattern matching

1. Bagging with B = Round(B/log(n). Make log(n) bagging predictions.
2. Pattern matching with log(n) predictions obtained in step 1.

Algorithm 2. pattern matched bagging

1. Pattern matching with log(n) bootstrap samples. Make Round(B/log(n)) pattern matched predictions.
2. Usual unweighted majority voting with Round(B/log(n)) predictions obtained in step 1.

2.3. Related Issues

During the research, it was found that the proposed methods improve the accuracy of bagging predictions, but not that of boosting predictions. Figure 1 shows the test error rate of each prediction that is not voted. The boosting predictions before voting have much larger test error rates than the bagging prediction at each iteration, even though boosting

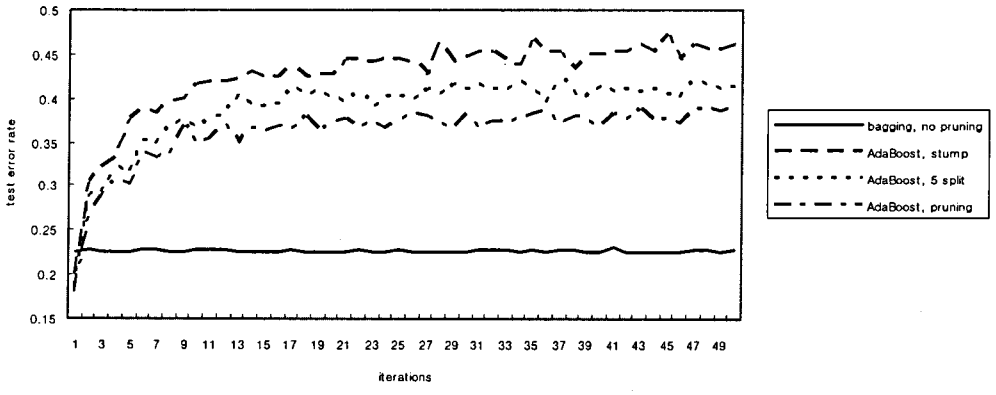

Figure 1. Test error rate of each prediction in bagging and boosting, before voting

gives similar test error rate as bagging after voting. Since the pattern matching require accurate classifiers in the algorithm, pattern matching with boosting cannot improve the original boosting. Therefore only the bagging is considered in the proposed methods.

In this paper, classification tree was used as the base learner. When using classification trees for ensemble, it is important to choose appropriate size of trees. It is well known that unpruned trees are most appropriate for bagging. In the proposed methods, however, the pruned tree is most appropriate because the proposed methods also require accurate classifiers in the algorithm.

3. Experiment

To compare the performance of the proposed methods with other classification methods, we experimented with simulation data and the real data. The accuracy of the proposed methods is compared with CART (Breiman et al., 1984), bagging with unpruned tree, logistic regression and boosting. The number of bootstrap samples in bagging and the iterations of boosting were fixed as 50. Figure 2 shows the histogram of the test error rate of all the methods in all the dataset. The results in the Table 2-4 show the mean and standard deviation of the test error rate. The lowest error rate in each dataset is written in bold case. CART was the base learner for all kinds of ensemble. For implementation, R 2.1.1 was used. CART was implemented using the function *rpart()* in the library *rpart*, logistic regression using the function *glm()* in the library *stats* and bagging using the function *bagging()* in the library *ipred*. Boosting was programmed based on the algorithm of AdaBoost.M1 in Hastie et al. (2001).

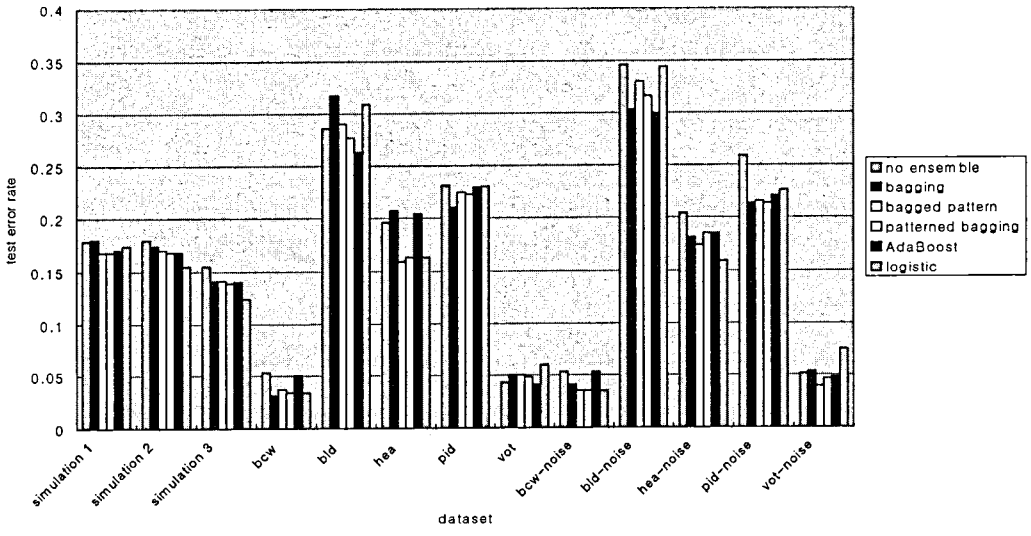

Figure 2. Histogram of the test error rate of simulation and real dataset

3.1. Simulation

For simulation study, we generated three simulation data. Only binary classification problem is considered. The simulation was iterated 100 times and 500 samples were generated for each class at each iteration. The training dataset and the test dataset were generated independently from the same distribution. Three simulation data are described below, where X and Y mean the vector and the class membership of each observation. Table 2 indicates that the proposed methods improve the bagging method.

Simulation data 1

$$Y=0: X\sim N((1,3,2)', I_3), \qquad Y=1: X\sim N((2,5,1)', diag(1,4,2))$$

Simulation data 2

$$Logit(p)=\sin(3x_1+5x_2)-x_3-2x_4+4x_5+\varepsilon, \quad \varepsilon\sim N(0,5)$$
$$(x_1,x_2,x_3,x_4,x_5)'\sim N_5((0,0,0,0,0)', diag(1,4,7,2,5))$$
$$Y=\begin{cases}1 & , \ p\geq 0.5\\ 0 & , \ p<0.5\end{cases}$$

Simulation data 3

Same as in Simulation data 2, but with the different covariance matrix,

$$\begin{pmatrix}5&1&0&0&0\\1&3&2&1&0\\0&2&7&2&3\\0&1&2&5&0\\0&0&3&0&9\end{pmatrix}$$

3.2. Real Dataset

As in the experiment with simulation data, only binary classification problem is considered. Table 3 gives a brief description of the real dataset and the noise variables added to these dataset. The experiment on the real dataset was implemented with the same setting as in Lim et al. (2000). The real dataset below can be found in the repository of machine learning databases of University of California, Urvine.[3] The accuracy was measured using 10-fold cross-validation. Table 4 indicates that both proposed methods improve the bagging method mostly except only three set. Figure 2 also demonstrates that the proposed methods are quite compatible with other classification methods.

Table 2. Results of the simulation data

dataset	no ensemble	bagging	bagged pattern	patterned bagging
simulation 1	0.179 (0.013)	0.180 (0.012)	**0.168 (0.011)**	**0.168 (0.012)**
simulation 2	0.180 (0.014)	0.174 (0.012)	0.170 (0.013)	**0.168 (0.013)**
simulation 3	0.155 (0.014)	0.141 (0.011)	0.141 (0.012)	**0.139 (0.012)**

3) http://www.ics.uci.edu/~mlearn/MLRepository.html

4. Conclusion

The experiments with simulation data and the real dataset indicate that the proposed ensemble methods improve the bagging method. The proposed methods performed best prediction when the pruned tree was used as the base learner. Among the proposed methods, "pattern matched bagging" performs slightly better than "bagged pattern matching."

References

Breiman, L. (1996), Bagging predictors, Machine Learning, Vol. 24, No. 2, pp. 123-140

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984), Classification and Regression Trees (CART), Chapman & Hall/CRC, New-York

Freund, Y. and Schapire, R. (1996), Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, pp. 148-156

Hastie, T., Tibshirani, R. and Friedman, J.H. (2001), The Element of Statistical Learning - Data Mining, Inference, and Prediction, Springer-Verlag, New-York

Lim, T.S., Loh, W.Y. and Shih Y.S. (2000), A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine Learning, Vol. 40, No. 3, pp. 203-229

Mojirsheibani, M. (1999), Combining classifiers via discretization, Journal of the American Statistical Association, Vol. 94, No. 446, pp. 600-609

Table 3. Description of the real dataset and adding noise variables

dataset	original variables		noise variables	
	numerical	categorical	numerical	categorical
Breast cancer Wisconsin (bcw)	9		9 U(1,10)	
Bupa liver disorders (bld)	6		9 N(0,1)	
Statlog heart disease (hea)	7	6	7 N(0,1)	
Pima Indians diabetes (pid)	7		8 N(0,1)	
Congressional voting records (vot)		16		14 U(0,3)

Table 4. Results of the real dataset and the real dataset with the noise variables added

dataset	no ensemble	bagging	bagged pattern	patterned bagging
bcw	0.053 (0.020)	**0.031 (0.028)**	0.037 (0.022)	0.034 (0.022)
bld	0.286 (0.066)	0.317 (0.068)	0.290 (0.060)	**0.277 (0.067)**
hea	0.196 (0.092)	0.207 (0.084)	**0.159(0.070)**	0.163 (0.082)
pid	0.231 (0.050)	**0.210 (0.055)**	0.224 (0.053)	0.222 (0.058)
vot	0.043 (0.025)	0.050 (0.041)	0.050 (0.030)	**0.048 (0.027)**
bcw-noise	0.053 (0.020)	0.041 (0.029)	**0.035(0.023)**	**0.035(0.020)**
bld-noise	0.346 (0.070)	**0.303 (0.085)**	0.330 (0.067)	0.316 (0.088)
hea-noise	0.204 (0.088)	0.181 (0.071)	**0.174 (0.080)**	0.185 (0.070)
pid-noise	0.259 (0.048)	**0.213(0.044)**	0.215 (0.065)	**0.213(0.051)**
vot-noise	0.051 (0.032)	0.053 (0.034)	**0.039(0.031)**	0.046 (0.028)