Ensemble Classifier and Binary Classification

David Dai
Fall 2018 Cogs 118A Student
University of California, San Diego
San Diego, CA 92093
d4dai@ucsd.edu

Abstract

In the past decade, numerous classification algorithms have been developed. One category of the classification algorithms that stand out among all algorithms is the ensemble classification. For this project, I am going to use random forest, Adaboot, gradient boosting, bagging with k-nearest neighbor and bagging with decision tree to test on five different datasets: ADULT, ABALONE, BANK, IRIS, and OCCUPANCY. The goal is to compute the overall best ensemble classification method among the five algorithms and show that gradient boosting has an equally good performance as random forest.

1 Introduction

Inspired by the research done by Rich Caruana and Alexandru Niculescu-Mizil, "*An Empirical Comparison of Supervised Learning Algorithms[1],*" I decided to conduct a study regarding ensemble classification learning algorithm. Ensemble classification algorithms are using multiple learning algorithms to train the classification method with a better predictive accuracy. Three of the most typical classifiers among the ensemble classification algorithms are bagging, boosting and random forest. Many people argue that random forest classifier would perform the best among all the classification algorithms. In my case, however, one classifier from the boosting classification method may perform equally or even better than the random forest classifier. In the following, I am going to demonstrate the result of different ensemble classifier by running on 5 different datasets with 3 different partitions and a 3-fold cross validation regarding the tuning parameters. As the training set gets larger, the predictive accuracy becomes higher. And the data will show a similar excellent result from both random forest classifier and gradient boosting classifier.

2 Method

2.1 Learning Algorithms

I attempt to compute the ideal parameter(s) for each classification algorithms from the ensemble library. With the computed best parameter(s), I then will compute the testing and training accuracy and error with the classification algorithm. This section will provide the parameters that are used to train each classification algorithm.

Random Forests Classifier: I picked the Breiman-Cutler implementations because this provides better results than other random forests implementations. The main idea is fitting a number of decision tree classifiers on sub-samples of the dataset. Each fitting calculation, the method will manipulate the weights on each feature and improve the predictive accuracy. I manipulate n_esitmators and decide the number of trees will be built. The number of trees could be built for the set: 1, 25, 50, 100 or 20.

Adaboot Classifier: For boosting, I actually trained both Adabooting and gradient boosting. In this case, Adaboot is fitting a classifier on the whole dataset and fit the same dataset to the classifier over and over to adjust the weights and improve the predictive accuracy. I manipulate n_estimators and decide the maximum number of estimators at the end of boosting. The maximum number of estimators could be: 10, 60, 120, 200, or 300.

Gradient Boosting Classifier: This is the second case of boosting. Because I am doing binary classification, there will be one regression tree to fit all the negative gradients of the binomial deviance loss function. [2] I manipulate n_estimators with 10, 60, 120, 200, or 300.

Bagging Classifier with KNN: I also computed two bagging classifiers. This one I selected the black-box estimator to be K-nearest Neighbors algorithm (KNN). I picked the default KNN with 5 queries and uniform distance. In this case of bagging classifier, the classifier will generate a number of subsets randomly and form each classification of their own with KNN. Then, the end will be a voting session from each subset to decide the classification. Here, I manipulated max_samples, which controls the number of samples to draw from the dataset. I also manipulated max_features, which controls the number of features to draw from the dataset. I picked max_samples to be 0.15, 0.3, 0.45, 0.6, or 0.75 and I picked max_features to be 025, 0.4, 0.5, 0.6, or 0.75.

Bagging Classifier with DT: In the second case of bagging classifier, I picked the black-box estimator to be Decision Tree (DT). I picked the default DT and I manipulated the max_samples parameter and max_features parameter. I picked max_samples to be 0.15, 0.3, 0.45, 0.6, or 0.75 and I picked max_features to be 025, 0.4, 0.5, 0.6, or 0.75.

2.2 Performance Metric and Comparison

For this design, I will value the performance of each classifier with its testing accuracy and testing error percentage. Both values will be in the range of 0 and1. Testing error percentage will be the complement of the testing accuracy. If the classification is trained properly, the testing accuracy should be in above 0.5 and the testing error percentage should be below 0.5.

Since I am deciding the better performance of the classifier based on its testing accuracy, I will use the same ideology to compare across classifiers. The average testing accuracy across all trials will be the data used for comparison. If one classifier's average testing accuracy is higher than the others, it means that this classifier has a better performance than the others. In the other case, if one classifier's average testing error percentage across all trials is lower than the other classifiers, this classifier performs the best among the five classifiers. This classifier performance comparison method, however, does not consider the amount of processing power used by the computer or the amount of time to generate a result. With the increasing modern computers' processing power, however, this disadvantage could be ignored.

2.3 Datasets

I compare the algorithms on three different binary classification datasets. Abalone Dataset (ABALONE), Adult Dataset (ADULT), Bank Marketing Dataset (BANK), IRIS Dataset (NURSERY), and Occupancy Detection Dataset (OCCUPANCY), are from the UCI Machine Learning Repository.

ABALONE: The goal of classifying this dataset is to predict an abalone's ring size based on the physical measurements. This multivariate dataset has 4177 number of instances with 8 attributes across categorical data, integer data and real number data. In my case, however, predicting abalone's ring number is not a binary classification. Thus, I change the original goal of the dataset to predict an abalone's ring size is bigger than 11.5 or smaller than 11.5. If the abalone is bigger than 11.5, it's 1 and if it's smaller than it's 0. I also use one hot encoding to convert the categorical data into binary.

ADULT: The goal of classifying this dataset is to predict a person's income is whether exceeds $50K per year or below $50K per year. This multivariate dataset has 48842 number of instances and each instance has 14 number of attributes. The attribute characteristics are categorical and integer. For the five classification algorithms listed above, I will convert all of them into binary with one hot encoding algorithm.

BANK: The goal of the dataset is to predict if the client will subscribe a term deposit or not. This multivariate dataset has 45211 number of instances with 17 attributes. There are categorical data and integer data. The prediction column is a categorical data with only two possibilities of either yes or no. Thus, I convert all the categorical data into binary with one hot encoding and remove the last column to avoid redundant data. As a result, if the client is willing to subscribe, the corresponding index at the last column would be 0

and otherwise it would be 1.

IRIS: The original goal is to decide what kind of iris flower based on the length and width. To make the goal into a binary classification problem, I change the goal of the dataset to predict whether the flower is iris-versicolor or not. The dataset has 150 number of instances with 4 categorical classifications. I convert all the categorical data into binary with one hot encoding and remove the redundant column. If it is versicolor, it would be 1 and otherwise, it would be 0.

OCCUPANCY: The goal of the dataset is to test if the room is occupied or not based on the data given. The dataset has 20560 number of instances with 7 real number attributes. However, I remove the column about time entry. I do not think time would be a factor of deciding whether the room is occupied or not. As for the other real number attributes, I leave them as the original way they are.

3 Experiment

3.1 Design

The experiment has three repeating trials for each dataset. In each trial, the dataset would first be shuffled. Then, the dataset would be separated in three different ways. The first partition would be 80% of the whole dataset being the training and validation data and the rest of the 20% would be the testing data. The second partition would be 50% being the training and validation data and the other 50% being the testing data. Lastly, the third partition would be 20% of the whole dataset being the training and validation data and the remaining 80% being the testing data. For each partition, I am going train each ensemble classifier with the training and validation data. During the training process, I am going to select a list of potential input for the designated parameter and I will use cross validation to tune the parameters with the list. The goal is to increase the predictive accuracy with the 3-fold cross The selected input from the list that has the highest accuracy rate would be the ideal parameter. I first train the random tree by having different numbers of trees will be built. Then, I train the Adaboot classifier and the gradient boosting

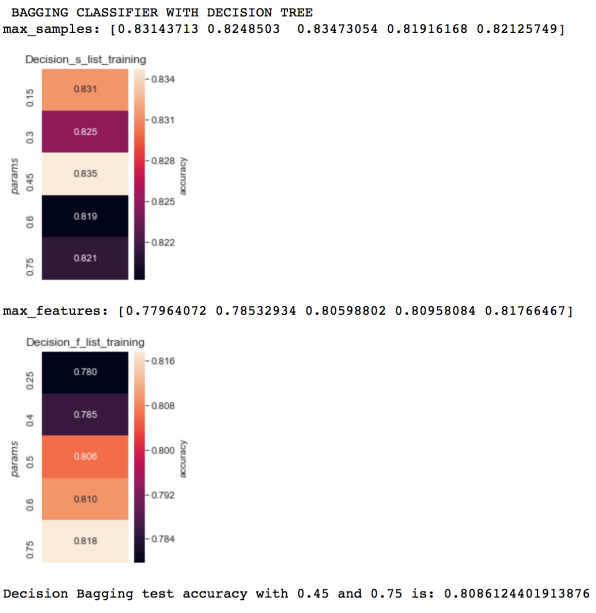

Figure 1. Abalone 80/20 Decision Bagging Training

classifier by manipulating its maximum number of estimators at the end of the boosting. Next up, I train both of the bagging classifiers by tuning the number of samples to draw from the X dataset with the default number of features. As you can see on the right, *Figure 1* calculates the highest accuracy by changing max_samples and then the second step is to calculate the highest accuracy by changing max_features. After finding the ideal number of samples, I then will tune the number of features to draw from the X dataset. After tuning each ensemble classifier, I then will use it to predict the values from the testing data and calculate the testing accuracy. After the three repeating trails are done, I will then calculate each ensemble classifier's testing accuracy according to the data partition.

3.1 Result

	Random Forest	Adaboot	Gradient Boost	K-neighbor Bagging	Decision Tree Bagging
ABALONE 80/20	0.828548644338118	0.767145135566188	0.832535885167464	0.836921850079744	0.821371610845295
ABALONE 50/50	0.823754789272030	0.757024265644955	0.828544061302682	0.80826976372924	0.808748403575989
ABALONE 20/80	0.816621769929163	0.766836276563903	0.816721540456949	0.502254813927965	0.800658485483388
ADULT 80/20	0.849736424586724	0.831516454270945	0.866369824453656	0.82839449065151	0.844362556937407
ADULT 50/50	0.852588907315275	0.815142395020780	0.870871977560756	0.804966934872960	0.848043731957496
ADULT 20/90	0.847594917271296	0.805290030327459	0.866200877832802	0.823678452147875	0.837984823473709
BANK 80/20	0.905709757086512	0.875078329462936	0.907773968815658	0.884993917947583	0.897858380331011
BANK 50/50	0.905467574980093	0.876153823468695	0.907487687634551	0.886047951871184	0.897136453448936
BANK 20/80	0.902421041960426	0.870496834305620	0.904337968978959	0.885002810878560	0.894522934004257
IRIS 80/20	0.922222222222222	0.933333333333333	0.933333333333333	0.922222222222222	0.933333333333333
IRIS 50/50	0.964444444444444	0.946666666666666	0.942222222222222	0.955555555555555	0.928888888888888
IRIS 20/80	0.913888888888889	0.916666666666666	0.938888888888888	0.875	0.913888888888888
OCCUPANCY 80/20	0.994020160601401	0.991286519733470	0.991799077396207	0.992140782504698	0.988552878865539
OCCUPANCY 50/50	0.994257588187038	0.991796554552912	0.993710691823899	0.991796554552912	0.993710691823899
OCCUPANCY 20/80	0.993719559087413	0.990387080235837	0.993164145945484	0.96650431532871	0.992950525506280

Table 2. Summation of all partition of all dataset training accuracy

4 Conclusion

Since I am measuring the performance of the ensemble classifier by the accuracy of the testing dataset. Thus, I decide to create a function to calculate the overall accuracy of the classifier. The overall accuracy would equal to sum of all the accuracy number from each dataset and then divided by the number of datasets times three because each dataset has tree partition. Applying this

equation to the chart above, we could achieve the following chart.

	Random Forest	Adaboot	Gradient Boosting	KNN Bagging	DT Bagging
Calculation Testing Accuracy	0.9009997793447364	0.8756546910546911	0.906264143454234	0.8642500277513813	0.8934675058242878

Table 2. Average testing accuracy

	Random Forest	Adaboot	Gradient Boosting	KNN Bagging	DT Bagging
Calculation Testing Error	0.0990002206552636	0.1243453089453089	0.093735856545766	0.1357499722486187	0.1065324941757122

Table 3. Average testing error rate

The above table about average testing error rate is computed by the average testing accuracy. The error rate is the complement of the accuracy percentage. As described in the performance metric section: if one classifier's average testing accuracy is higher than the others, it means that this classifier has a better performance than the others. In the other case, if one classifier's average testing error percentage across all trials is lower than the other classifiers, this classifier performs the best among the five classifiers. As both Table 2 and Table 3 is showing, two of the better performance among the five classifiers are the random forest classifier and the gradient boosting classifier.

References

[1] Caruana, Rich, and Alexandru Niculescu-Mizil. "An Empirical Comparison of Supervised Learning Algorithms." *Proceeding of the 23rd International Conference on Machine Learning* – ICML '06, 2006.

[2] "3.2.4.3.5. Sklearn.ensemble.GradientBoostingClassifier¶." *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoosting Classifier.html.