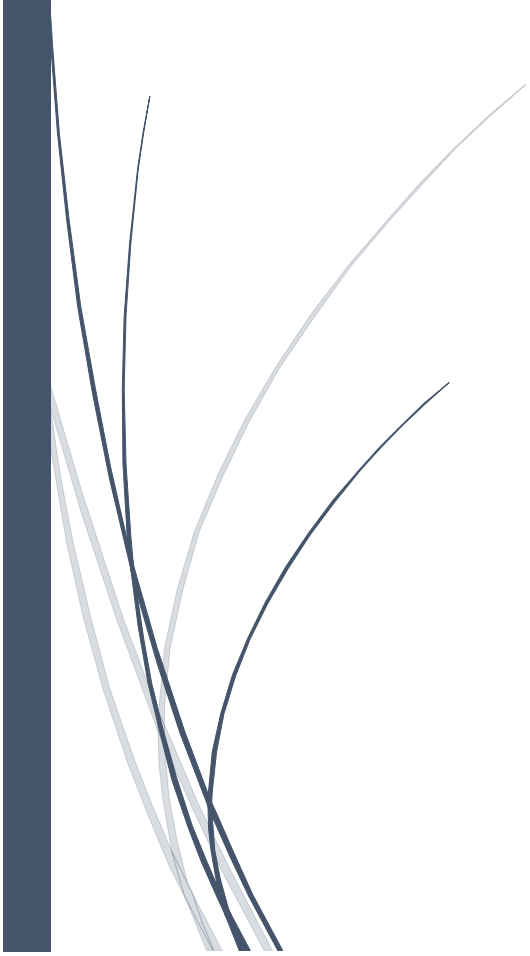
12/15/2017

Random Forest Classification Algorithm Implement in R

STAT545 FINAL PROJECT



Background

Random forest is an ensembled learning method for classification and regression. The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995[1][2]. The method combines Breiman's "bagging" idea and the random selection of features. The algorithm is preferred because it can be used for both classification and regression tasks. By growing enough trees in the forest, it can avoid overfitting problem. Furthermore, the classifier of Random Forest can handle missing values, and the Random Forest classifier can be modeled for categorical values. Random Forest has widely application in Banking, Medicine, Stock Market and E-commerce.

Objective

The objective of the project is to implement random forest classification in R without using package. The algorithm is mainly composed of two parts: building the trees and prediction. Based on the algorithm, dataset with categorical and continuous variables will be tested and the error would be evaluated.

Methods

Decision Tree

The random forest starts with a standard machine learning technique called a "decision tree". Decision trees are individual learners that are combined. They are one of the most popular learning methods commonly used for data exploration.

In this example, the tree advises us, based upon weather conditions, whether to play ball. For example, if the outlook is sunny and the humidity is less than or equal to 70, then it's probably OK to play.

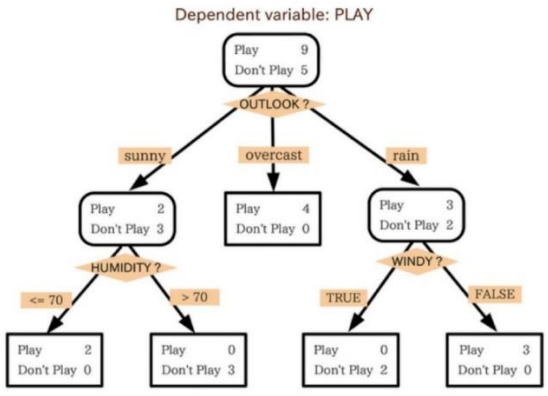


Figure 1 Illustration of Decision Tree

Random Forest (RF)

Using decision trees, we can build a random forest. One problem that might occur with one big (deep) single decision tree is that it can overfit. That is the decision tree can "memorize" the training set the way a person might memorize an Eye Chart.

The point of RF is to prevent overfitting. It does this by creating random subsets of the features and building smaller (shallow) trees using the subsets and then it combines the subtrees.
For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some criterion like information gain, gini index, etc. These criterions will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value is placed at the root.
A Gini score gives an idea of how good a split is by how mixed the classes are in the two groups created by the split. A perfect separation results in a Gini score of 0, whereas the worst case split that results in 50/50 classes in each group results in a Gini score of 1.0 (for a 2 class problem). It means an attribute with lower gini index should be preferred. If dataset T is split into n subsets with Ni, the gini indexi of the split data is defined as:

$$\text{Gini Index} = 1 - \sum_{i} p_i^2$$

$$\text{Gini}_{\text{split}}(T) = \sum_{i}^{n} \frac{N_i}{N} \text{Gini}(T_i)$$

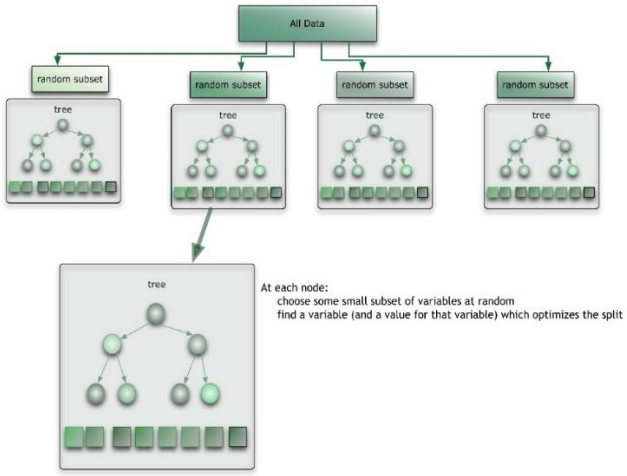


Figure 2 Illustration of Random Forest

When decision trees are built, prediction can be made by majority votes of the set of N trees.

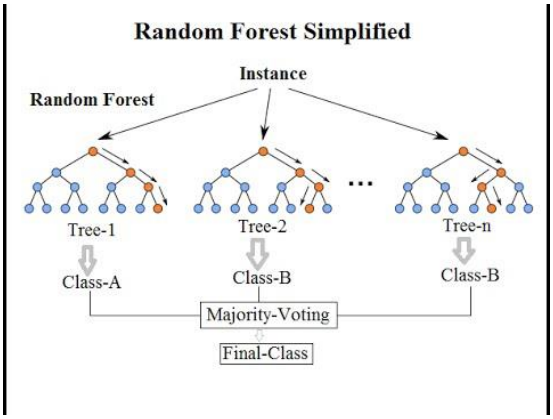


Figure 3 Majority Vote

OOB error

For each tree grown, approximate 1/3 of samples are not selected in bootstrap, which is called out of bootstrap or bag samples. Using OOB samples as input to the corresponding tree, predictions are made as if they were novel test samples. The OOB error is estimated as below. B is the number of trees.

$$\text{OOB Error} = \frac{1}{B}\sum_{i=1}^{B}\frac{N_{error[i]}}{N_i}$$

The advantages of random forest are:

1. It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.

2. It runs efficiently on large databases.

3. It can handle thousands of input variables without variable deletion.

4. It gives estimates of what variables are important in the classification.

5. It generates an internal unbiased estimate of the generalization error as the forest building progresses.

6. It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

Pseudo Code

1. Build forest

1.1 First, create two R class types: Tree and Node for easier handling of the tree-structure.

Tree will be used in a recursive structure that contains either another tree of a node at each branch. Node is used for the terminal location in the trees. Each branch that contains a node will signify that the algorithm has either:

- Every element in the subset belongs to the same class.
- There are no more attributes to be selected.
- There are no more examples in the subset.

For b=1 to B trees, for each tree, repeat

1.2 Draw a bootstrap sample n of size N (with replacement) from the training data.

1.3 Select m variables at random from the M variables. m should be much less than M.

1.4 For each node calculate the best split based on the Gini index.

- If the variable is categorical, split the dataset based on the number of categories;

- If the variable is continuous, calculate the gini index when each data point is treated as split point. Then select the split point with minimum gini index and cut the data into two subsets.
- If there are no attributes left to classify with, return the most common class left in the dataset as a best approximation.
- If the target attribute of the subset data is unique, create a terminal node with the unique target value.

1.5 Repeat 1.2, 1.3, 1.4 on each subset until you find leaf nodes in all the branches of the tree.

2. Output the ensemble of trees $\{T_i\}_{i=1}^{B}$.

3. Predict

3.1 Feed the test data to each tree and calculate the votes for each predicted target.

3.2 Consider the highest voted predicted target as the final prediction

Result and Evaluation

Test 1

To testify the accuracy of the algorithm, a dataset "mushroom" is applied[3] . The data contains 23 attributes and 8124 observations. Attribute 'class' is the target attribute. 'e' means edible and 'p' means poisonous. The aim is to predict whether a mushroom is poisonous or not.

Table 1 Attributes of Mushrooms

Target Attribute	**Class p: poisonous e: edible**
Feature Attribute	**cap-shape, cap-color, bruises, odor: gill-attachment, gill-spacing, gill-size, gill-color, stalk-shape, stalk-root, stalk-color-above-ring, stalk-color-below-ring, veil-type, veil-color, ring-number, ring-type, spore-print-color, population, habitat.**

With 8000 test data, 100 trees, 10 attributes and 66% samples, the confusion matrix and OOB Error are shown below.

```
          true
predicted  e  p
        e 68  0
        p  0 57
[1] "Out of bag error of mushroom: 0.0201750121536218"
```

The prediction is 100% correct and the OOB error is 2.02%, which is pretty good. The group also tried other parameters. Even the result of small number of trees turned out very accurate. Since random forest works well with large dataset. The high accuracy is not a surprise.

Test 2

To test whether the continuous variable works, the second dataset is iris. It includes 569 observations and 29 feature attributes. The attributes of breast cancer dataset is shown below.

With 10 trees, 3 attributes and 66% samples, the confusion matrix and OOB error are shown below.

Table 2 Attributes of Breast Cancer

Target Attribute	**Diagnosis** **M = malignant, B = benign**
Feature Attribute	**radius_mean, texture_mean, perimeter_mean area_mean _mean, compactness_mean, concavity_mean, concave.points_mean, symmetry_mean, fractal_dimension_mean, radius_se, texture_se, perimeter_se area_se, smoothness_se, compactness_se, concavity_se, concave.points_se, symmetry_se, fractal_dimension_se, radius_worst, texture_worst perimeter_worst smoothness_worst, compactness_worst, concavity_worst, concave.points_worst, symmetry_worst , fractal_dimension_worst**

```
          true
predicted  B  M
        B 52  1
        M  0 16
```

```
[1] "Out of bag error of iris: 0.398706476141195"
```

The OOB error is quite large since the dataset is relatively small. Random forests have been observed to overfit for some datasets with noisy classification tasks. This might be the reason for high OOB error.

Contribution and Collaboration

Two group members had been working closely. At the earlier stage, two members looked up the relevant information about random forest and communicated in time. When it comes to coding part, two members worked on the same function at the same time. If one were responsible to write the original code, the other checked the debugged that part. Particularly Wenjin Zhang looked the test dataset and summarized the random forest algorithm and test result. Tianqiao Liu summarized the pseudo code.

What I learned

- The basic idea of types of decision trees: ID3, CART etc.
- Thoroughly and good understanding of every detail of random forest.
- Two indexes to select best attributes to split dataset: information gain and gini index. Random forest requires gini index. And the group also write information gain(entropy) in R.
- Boosting and bootstrap.
- The method to deal with tree structure in R by specifying class type.
- The index to evaluate random forest: out-of-bag error.
- R code proficiency, coding and debugging.

- Familiar myself with R function, e.g. table, prop.table, apply, cut, setdiff, seq_along etc.
- Growing interest in other classification algrithoms.

Reference

[1]. *Ho, Tin Kam (1995). Random Decision Forests.Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.*

[2]. Ho, Tin Kam (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **20** (8): 832–844.

[3]. https://www.kaggle.com/uciml/mushroom-classification

[4]. https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/data