**1.Introduction**

CART(Classification and Regression Tree) is a kind of decision tree,  proposed by  Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone in 1984. It can be used for both classification and regression.Since it is based on decision tree, so before moving forward to CART, let’s take a look on decision tree.

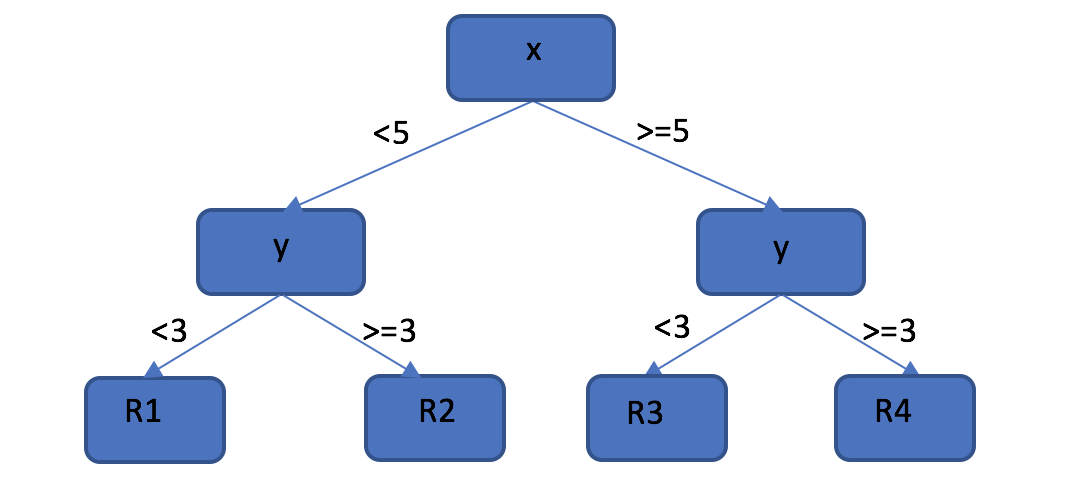


figure-1 Decision Tree

Figure-1 is a very simple example of decision tree, it divide two-dimensional coordinate system into 4 parts by making decision on x and y. For example, if our input is (x,y)=(6,6), it will go through x>=5 and y>=3 and located at R4. In the decision tree theory, we call {x,y} are attributes of the input data, the ‘yes’ or ‘no’ answers to these attributes will help us to classify those given input data into some different results, we call the result for a given data is its type.

In decision tree, we can have any number of branches for a node, however, CART is a binary tree. The internal node features of binary tree are yes and no, the left branch is a branch with a value of yes, and the right branch is a branch with a value of no.Such a tree uses dichotomies to process each feature recursively, dividing the feature space into finite units, and determining the predicted probability distributions on these units.

**2.Random Forest**

One famous algorithm in CART is random forest,  it add an addition layer of randomness to bagging, so what is bagging(bootstrap aggregating)? Here is a piece of pseudocode to show how it works:

*Input: original data set S*

*sample(n): sample n training data from S with replacement*

*model(data): use given data to get a model.*

*model\_list = []*

*for i= 1,2,...,N:*

*d = sample(n)*

*m = model(d)*

*model\_list.add(m)*

After these operation, we will get N models, then we can use these models to process our new data. By this method, we can improve our estimation at some extent. In random forest,  we will also sample n samples from original data set with replacement, but now the number of original data is n, too. Generally speaking, the effect of random forest is better than bagging.

Random forest is composed by many decision tree, for each tree, we build it as follow steps(N:the number of input samples for a decision tree, M: the dimension of attributes):

1)From original data set S, sampling N times with replacement, and we can get a training data set S\*, then we use it as the input the root node, and start our training.

2)We decide a number m smaller than M, normally, it should satisfy m<<M, for each node(include root node), we randomly choose m attributes as candidates, then we choose best one of them as split point. This step will shorten the time of the algorithm remarkably.

3)Repeat step-2 until we build a tree with all attributes.

Using this method to build t decision trees,  we can get a random forest. Now we can use it to deal with some classification and regression work. For classification, we would get the vote from all decision trees, and choose the biggest one as our result, for regression, we would use the mean value as our criterion. And for each tree, we don’t input all the original samples into it, so we can use those samples have not been used to test it(generally we would have ⅓ of total samples that can be used as test data). Putting them into the tree, we can get a result for each data, the ratio of failed result is called OOB error, it is computed as:

First,we need to install randomForest package,the newest release is 4.6-12,and it was developed by Andy Liaw and Matthew Wiener.

*> install.packages("randomForest")*

Then, let's see a classification example

*> library(randomForest)*

*> data(iris)*

*> set.seed(13)*

*> iris.rf <- randomForest(Species ~ ., data = iris, mtry = 2, do.trace = 100)*

*ntree      OOB      1      2      3*

*100:   6.67%  0.00%  8.00% 12.00%*

*200:   5.33%  0.00%  8.00%  8.00%*

*300:   5.33%  0.00%  8.00%  8.00%*

*400:   4.67%  0.00%  8.00%  6.00%*

*500:   4.67%  0.00%  8.00%  6.00%*

*> print(iris.rf)*

*Call:*

*randomForest(formula = Species ~ ., data = iris, mtry = 2, do.trace = 100)*

*Type of random forest: classification*

*Number of trees: 500*

*No. of variables tried at each split: 2*

*OOB estimate of  error rate: 4.67%*

*Confusion matrix:*

*setosa versicolor virginica class.error*

*setosa         50          0         0        0.00*

*versicolor      0         46         4        0.08*

*virginica       0          3        47        0.06*

In this case, our original data frame is iris, it has 150 rows and 5 columns,4 rows are some attributes of flowers, and one row is the tag(species). It has three kinds of species: setosa, versicolor and virginica, 50 rows of data for each of them.

RandomForest uses classification model, “Species ~ .” means column “Species” in iris is the tag in our case. ‘mtry=2’ and some default values like ‘ntree=500, replace=TRUE,sampsize=if (replace) nrow(x) else ceiling(.632\*nrows), nodesize = if (!is.null(y) && !is.factory(y) 5 else 1)’ means we will construct 500 trees, for each tree, each node has only one attribute, and at each split we randomly choose two attributes as candidates. And since we set “do.trace=100”, so every 100 tree, it will print an OOB error.

At last, we can get an OOB error and a confusion matrix, we have discussed how to compute it, so now we focus on confusion matrix. After building 500 trees, we input all 150 rows of data into these trees. For each row, every tree will give a classification result for it, we choose the biggest voter as its final species. From our output, for example, four versicolor is classified as virginica, so we can get

for each species, we can get a class.error like this , then we can get a confusion matrix.

Next, let’s see a regression example:

*> data(airquality)*

*> set.seed(18)*

*> ozone.rf <- randomForest(Ozone ~ ., data=airquality, mtry = 3, ntree = 400, na.action = na.omit)*

*> print(ozone.rf)*

*Call:*

*randomForest(formula = Ozone ~ ., data = airquality, mtry = 3, ntree = 400, na.action = na.omit)*

*Type of random forest: regression*

*Number of trees: 400*

*No. of variables tried at each split: 3*

*Mean of squared residuals: 297.8237*

*% Var explained: 72.86*

In this case , the data frame ‘airquality’, we build 400 trees, in each tree and at each split, we randomly choose 3 attributes as candidates. The “mean of squared residuals” is

where is the average of the OOB predictions for the ith observation. The “percent variance explained” is

where is the variance of , computed with n as divisor (rather than n−1).

<https://cran.r-project.org/web/packages/randomForest/randomForest.pdf>

<https://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm>