The Random Forest Package

Nadia Aquil

12/3/2017

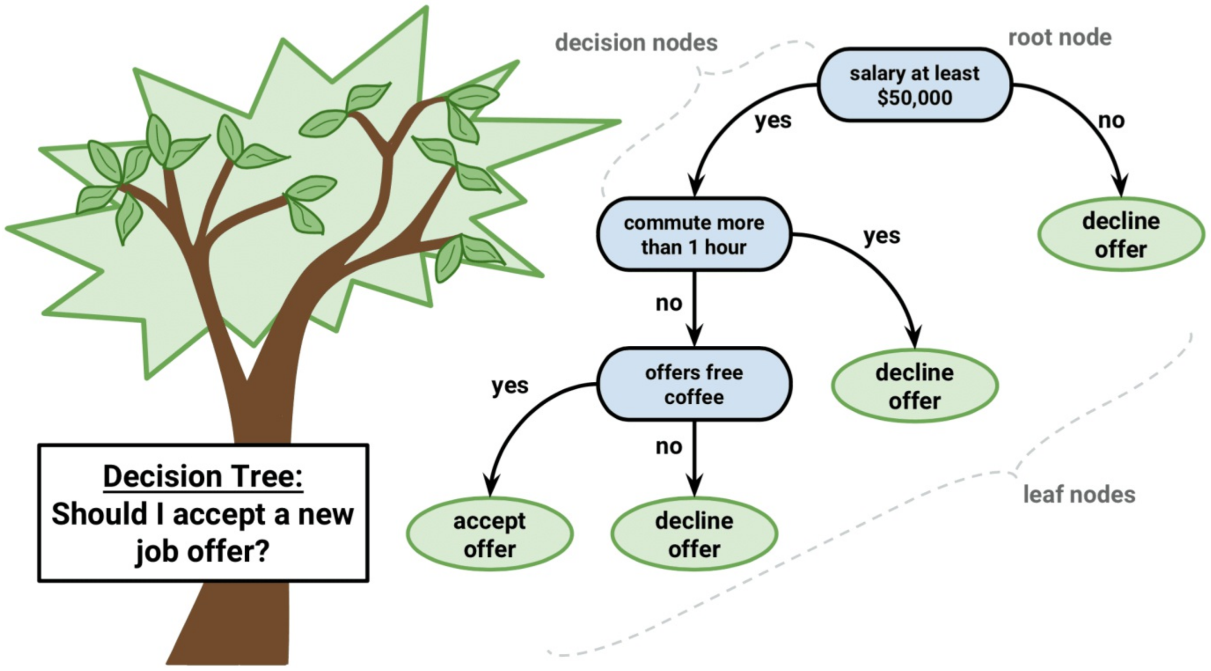
Introduction

Random forests are a machine learning method for classification, regression, and other tasks. They work by building several decision trees during training, and then averaging the results together. This is known as "ensemble learning". The idea is that creating several models and then averaging the result will end up with less variance than creating one model. One main advantage of random forests is that the classifier will not overfit the model, because it takes the average of many trees instead of just one.

The commonly used parameters of randomForest include: x: the random forest formula data: the input data frame ntree: the number of decisions to grow replace: indicates whether to take sample with or without replacement sampsize: sample size to be drawn from input data

Background Information

In order to understand random forests, it is first necessary to understand how a decision tree works, because a random forest is made up of several decision trees. The pseudocode of the decision tree algorithm is as follows: 1. Place the best attribute of the data at the root of the tree. 2. Split the training set into subsets, which are made so that each subset has data swith the same value for an attribute. 3. Repeat 1 and 2 on each subset until you find leaf nodes in all the brances of the tree.



Data Training Example

Let's look at an example. The data set we will use is one that lists each passenger on the Titanic alone with some biographical data (age, sex) and whether they survived the shipwreck.

```
# get randomforest if it's not there already
if (!require("randomForest")) {
    install.packages("randomForest")
    library(randomForest)
}
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
# it is a good idea to set the random seed so that the results are reproducible every time
# otherwise you will get different classifications for each run
set.seed(415)

#download dataset, read into a dataframe
data <- read.table("http://math.ucdenver.edu/RTutorial/titanic.txt", h=T, sep="\t")
#make survived into a yes/no variable. this step is called "classification"
data$Survived <- as.factor(ifelse(data$Survived==1, "yes", "no"))

#split data into a training and test set
result <- runif(nrow(data)) <= .75
data.train <- data[result,]
data.test <- data[-result,]

#train a random forest
forest <- randomForest(Survived ~ PClass + Age + Sex,
             data=data.train, importance=TRUE, na.action=na.omit)

#the importance variable indicates how important each variable is in the random forest
imp <- importance(forest)

# this line sets the "survived" variable as the most important, followed by the passenger's
# class, age, and sex
o <- order(imp[,3], decreasing=T)
imp[o,]
```

```
##               no      yes MeanDecreaseAccuracy MeanDecreaseGini
## Sex    48.97897 51.48329             52.41936         67.91830
## PClass 20.47996 19.65379             23.03907         19.96737
## Age    19.09494 12.60653             22.36990         18.79398
```

```
# confusion matrix [[True Neg, False Pos], [False Neg, True Pos]]
table(data.test$Survived, predict(forest, data.test), dnn=list("actual", "predicted"))
```

```
##       predicted
## actual  no yes
##    no  387  56
##    yes  82 230
```

the outcome after running this is a “confusion matrix” as follows:

```
#              no      yes MeanDecreaseAccuracy MeanDecreaseGini
#Sex    48.97897 51.48329             52.41936         67.91830
#PClass 20.47996 19.65379             23.03907         19.96737
#Age    19.09494 12.60653             22.36990         18.79398
#      predicted
#actual  no yes
#   no  387  56
#   yes  82 230
```

What does this data mean? the classifier made a total of 755 predictions (1 for each passenger). Based on the training data that included various information about each passenger, the model then builds several decision trees that simulate the same list of passengers. Each tree predicts based on the variables that were given an “importance” score. Out of these cases, the model predicted “yes” 286 times (56 + 230) and “no” 467 times (387 + 82). In reality, 312 passengers survived (82 + 230) and 433 did not (387 + 56). Thus the random forest classifier got pretty close. Depending on the random seed used, the numbers might vary a little bit, but in general they form a good approximation. After being trained on the data set, this random forest can now make predictions for each passenger, based on their age, sex and class, to determine whether they are likely to survive or not.

Uses of Random Forests

Random forests are used for many machine learning applications. They are used by many recommendation engines for indentifying how likely it is that a customer will want to buy a certain product, based on similar customer profiles. They are useful in identifying trends in the stock market and for diagnosing diseases by analyzing medical records.

Conclusion

Random forests are ultimately a powerful tool that are useful in a wide variety of machine learning settings. They can be used to make predictions based on available data. The R package randomForest provides an intuitive platform for building random forests and training them on whatever data one wants to analyze.

References

http://blog.yhat.com/posts/10-R-packages-I-wish-I-knew-about-earlier.html

https://en.wikipedia.org/wiki/Random_forest#Algorithm

https://cran.r-project.org/web/packages/randomForest/index.html

http://trevorstephens.com/kaggle-titanic-tutorial/r-part-5-random-forests/

http://www.listendata.com/2014/11/random-forest-with-r.html http://blog.datadive.net/random-forest-interpretation-with-scikit-learn/

http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/