Data621 - Blog3

Amit Kapoor

4/3/2021

Regression Trees

Tree-based models consist of one or more nested conditional statements for the predictors that partition the data. Within these partitions, a model is used to predict the outcome. In the tree models terminology, there are two splits of the data into three terminal nodes or leaves of the tree. To get a prediction for new data, we would follow the if-then statements defined by the tree using values of that sample's predictors until we come to a terminal node. The model formula in the terminal node would be used to get the prediction.

To demonstrate here various tree based models here, We will use the package `mlbench` that contains a function called `mlbench.friedman1` that simulates the non linear data.

Single Trees

Regression trees partition a data set into smaller groups and then fit a simple model for each subgroup. Basic regression trees partition the data into smaller groups that are more homogenous against the response. To achieve outcome consistency, regression trees determine the predictor to split on and value of the split, the depth or complexity of the tree and the prediction equation in the terminal nodes.

`caret` package implements the `rpart` method with cp as the tuning parameter. `caret` by default prunes tree based models. `cp` is the parameter used by `rpart` to determine when to prune.

```
set.seed(317)

singletree.model <- train(x=trainingData$x,
                          y=trainingData$y,
                          method = "rpart",
                          tuneLength = 5,
                          trControl = trainControl(method = "cv"))

singletree.model
```

```
## CART
##
## 200 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
## Resampling results across tuning parameters:
##
##   cp          RMSE      Rsquared    MAE
##   0.07534530  3.934818  0.37817606  3.180454
##   0.07900591  3.959539  0.37205821  3.197096
```

```
##   0.10653465  4.026705  0.33163441  3.239278
##   0.12310943  4.241378  0.26076200  3.458335
##   0.19745805  4.796923  0.09492617  3.965858
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.0753453.
```

Model Trees

One limitation of simple regression trees is that each terminal node uses the average of the training set outcomes in that node for prediction. As a consequence, these models may not do a good job predicting samples whose true outcomes are extremely high or low. The model tree approach differs from regression trees as the splitting criterion is different, the terminal nodes predict using a linear model and prediction is often a combination of the predictions from different models along the same path through the tree.

To tune model trees model, the **train** function in the **caret** package has method = "M5" that evaluates model trees and the rule-based versions of the model along with smoothing and pruning.

```
set.seed(317)
mdltree.model <- train(x=trainingData$x,
                       y=trainingData$y,
                       method = "M5",
                       trControl = trainControl(method = "cv"),
                       control = Weka_control(M = 2))

mdltree.model
```

```
## Model Tree
##
## 200 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
## Resampling results across tuning parameters:
##
##   pruned  smoothed  rules  RMSE      Rsquared   MAE
##   Yes     Yes       Yes    2.928634  0.6555064  2.341575
##   Yes     Yes       No     3.257857  0.5666599  2.661570
##   Yes     No        Yes    2.814728  0.6744466  2.229229
##   Yes     No        No     3.088767  0.6026558  2.443961
##   No      Yes       Yes    3.294271  0.5511199  2.648493
##   No      Yes       No     3.226561  0.5699273  2.613535
##   No      No        Yes    4.336560  0.3551470  3.597369
##   No      No        No     3.573643  0.5040270  2.940323
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were pruned = Yes, smoothed = No and
##  rules = Yes.
```

Bagged Trees

```
bagCtrl <- bagControl(fit = ctreeBag$fit,
                      predict = ctreeBag$pred,
```

```
                      aggregate = ctreeBag$aggregate)

bagg.model <- train(x=trainingData$x,
                    y=trainingData$y,
                    method="treebag",
                    tuneLength = 2,
                    trControl = trainControl(method = "cv"),
                    bagCtrl=bagCtrl)

bagg.model
```

```
## Bagged CART
##
## 200 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   2.850772  0.6859794  2.251662
```

Random Forests

```
set.seed(317)

randfrst.model <- train(x=trainingData$x,
                        y=trainingData$y,
                        method = "rf",
                        tuneLength = 2,
                        trControl = trainControl(method = "cv"))

randfrst.model
```

```
## Random Forest
##
## 200 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##    2    2.891769  0.8425261  2.363624
##   10    2.477719  0.7778500  1.987306
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 10.
```

Boosting

```r
set.seed(317)

# boosting regression trees via stochastic gradient boosting machines
gbmGrid <- expand.grid(interaction.depth = seq(1, 2, by = 1),
                       n.trees = seq(100, 200, by = 50),
                       shrinkage = 0.1,
                       n.minobsinnode = 5)

gbm.model <- train(x=trainingData$x,
                   y=trainingData$y,
                   method = "gbm",
                   tuneGrid = gbmGrid,
                   trControl = trainControl(method = "cv"),
                   verbose = FALSE)

gbm.model
```

```
## Stochastic Gradient Boosting
##
## 200 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                  100      2.187126  0.8221941  1.802816
##   1                  150      2.023134  0.8335915  1.664078
##   1                  200      1.969022  0.8380564  1.609419
##   2                  100      1.976804  0.8440204  1.594093
##   2                  150      1.874660  0.8579755  1.523923
##   2                  200      1.846831  0.8610865  1.495919
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 5
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 200, interaction.depth =
##  2, shrinkage = 0.1 and n.minobsinnode = 5.
```

Cubist

```r
set.seed(317)

cubist.model <- train(x=trainingData$x,
                      y=trainingData$y,
                      method = "cubist",
                      tuneLength = 5,
                      trControl = trainControl(method = "cv"))
```

```r
cubist.model
```

```
## Cubist
##
## 200 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 180, 180, 180, 180, 180, 180, ...
## Resampling results across tuning parameters:
##
##   committees  neighbors  RMSE      Rsquared   MAE
##    1          0          2.862530  0.6562333  2.107156
##    1          5          2.612391  0.7113876  1.890289
##    1          9          2.592513  0.7158471  1.882125
##   10          0          1.763904  0.8675705  1.329510
##   10          5          1.767245  0.8637853  1.354904
##   10          9          1.718827  0.8735992  1.327531
##   20          0          1.593833  0.8915076  1.209124
##   20          5          1.624610  0.8858174  1.253957
##   20          9          1.587134  0.8930052  1.221502
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were committees = 20 and neighbors = 9.
```

References

Applied Predictive Modeling by Max Kuhn and Kjell Johnson