Data624 - Homework9

Amit Kapoor

4/26/2021

Contents


```
library(mlbench)
library(randomForest)
library(caret)
library(party)
library(gbm)
library(Cubist)
library(rpart)
```

Exercise 8.1

Recreate the simulated data from Exercise 7.2

```
set.seed(317)
simulated <- mlbench.friedman1(200, sd = 1)
simulated <- cbind(simulated$x, simulated$y)
simulated <- as.data.frame(simulated)
colnames(simulated)[ncol(simulated)] = "y"
```

(a)

Fit a random forest model to all of the predictors, then estimate the variable importance scores:

```
model1 <- randomForest(y ~ ., data = simulated, importance = TRUE, ntree = 1000)
rfImp1 <- varImp(model1, scale = FALSE)
```

Did the random forest model significantly use the uninformative predictors (V6 – V10)?

```
rfImp1
```

```
##          Overall
## V1   6.017023817
## V2   7.197326850
## V3   1.969544508
## V4   9.875194405
## V5   2.087946997
## V6   0.060239433
## V7  -0.051620797
## V8  -0.024680384
## V9  -0.003664196
## V10 -0.082197496
```

Based on the above results, the random forest model doesnt significantly use the uninformative predictors (V6 – V10).

(b)

Now add an additional predictor that is highly correlated with one of the informative predictors. For example:

```
set.seed(317)
simulated$duplicate1 = simulated$V1 + rnorm(200) * .1
cor(simulated$duplicate1, simulated$V1)
```

```
## [1] 0.9400187
```

Fit another random forest model to these data. Did the importance score for V1 change? What happens when you add another predictor that is also highly correlated with V1?

```
model2 <- randomForest(y ~ ., data = simulated, importance = TRUE, ntree = 1000)
rfImp2 <- varImp(model2, scale = FALSE)
```

```
rfImp2
```

```
##                Overall
## V1          4.92452320
## V2          6.75435224
## V3          1.76154183
## V4          9.29279187
## V5          2.19868128
## V6          0.16614783
## V7         -0.04778316
## V8          0.01070291
## V9         -0.01098350
## V10        -0.15881709
## duplicate1  2.59912860
```

We see here after adding another predictor that is highly correlated with V1, its importance got reduced. The importance now is splitted between V1 and duplicate1 after adding the highly correlated duplicate1.

(c)

Use the `cforest` function in the `party` package to fit a random forest model using conditional inference trees. The `party` package function `varimp` can calculate predictor importance. The conditional argument of that function toggles between the traditional importance measure and the modified version described in Strobl et al. (2007). Does this importance show the same pattern as the traditional random forest model?

```
model3 <- cforest(y ~ ., data = simulated)
# conditional = TRUE
ctrue <- varimp(model3, conditional = TRUE)
# conditional = FALSE
cfalse <- varimp(model3, conditional = FALSE)
```

```
cbind(model2=rfImp2, cforest_con=ctrue,cforest_uncon=cfalse )
```

```
##                Overall cforest_con cforest_uncon
## V1          4.92452320  2.06175787  4.4446682522
## V2          6.75435224  5.06766860  6.8801452007
## V3          1.76154183  0.15834716  0.2782253511
## V4          9.29279187  6.67572503 10.2896859310
## V5          2.19868128  1.20148004  1.7910397555
## V6          0.16614783  0.09955285  0.1839316106
## V7         -0.04778316 -0.01022655  0.0151452375
## V8          0.01070291  0.01982557 -0.0004442511
## V9         -0.01098350  0.06097283  0.0166033367
## V10        -0.15881709 -0.06822672 -0.0523436249
## duplicate1  2.59912860  0.91810824  2.0665125520
```

We can see here that importance shows the different pattern as compared to traditional random forest model. V4, remains the most important variable in all the 3 cases. Also we can see that uninformative predictors remains low in all 3 cases.

(d)

Repeat this process with different tree models, such as boosted trees and Cubist. Does the same pattern occur?

```
set.seed(317)

# boosting regression trees via stochastic gradient boosting machines

gbmGrid <- expand.grid(interaction.depth = seq(1, 7, by = 2),
                       n.trees = seq(100, 1000, by = 50),
                       shrinkage = 0.1,
                       n.minobsinnode = 5)

gbm_model <- train(y ~ ., data = simulated,
                   method = "gbm",
                   tuneGrid = gbmGrid,
                   verbose = FALSE)

gbm_imp <- varImp(gbm_model)
gbm_imp
```

```
## gbm variable importance
##
##             Overall
```

```
## V4         100.0000
## V2          80.1464
## V1          58.5491
## V3          48.3656
## V5          39.4555
## duplicate1  19.3401
## V6           6.1123
## V8           1.9197
## V7           0.4404
## V10          0.1300
## V9           0.0000
```

```
set.seed(317)

# cubist
cubist_model <- cubist(x = simulated[, names(simulated)[names(simulated) != 'y']],
                       y = simulated[,c('y')])
cubist_imp <- varImp(cubist_model)
cubist_imp
```

```
##            Overall
## V2            91.5
## V1            69.0
## V4            69.0
## V3            33.0
## V5            50.0
## V8             2.5
## V6             0.0
## V7             0.0
## V9             0.0
## V10            0.0
## duplicate1     0.0
```

Comparing the results with cforest, the uninformative predictors (V6-V10) still appear as lowest in ranking. V4 still appear as highest for boosted trees but cubist shows V2 as highest in ranking.

Exercise 8.2

Use a simulation to show tree bias with different granularities.

We will do the simulation here with 4 variables having different granularities. The response variable we will choose, would be a function of random selection and some noise.

```
set.seed(317)

df <- data.frame(x1 = sample(0:10000/10000, 250, replace = TRUE),
                 x2 = sample(0:100/100, 250, replace = TRUE),
                 x3 = sample(0:1000/1000, 250, replace = TRUE),
                 x4 = sample(0:10/10, 250, replace = TRUE))

df$y <- df$x1 + df$x4 + rnorm(250)

str(df)
```

```
## 'data.frame':    250 obs. of  5 variables:
##  $ x1: num  0.461 0.404 0.522 0.896 0.51 ...
```

```
##  $ x2: num  0.49 0.22 0.99 0.31 0.5 0.87 0.32 0.36 0.22 0.94 ...
##  $ x3: num  0.113 0.353 0.711 0.232 0.825 0.906 0.558 0.448 0.603 0.784 ...
##  $ x4: num  0.1 0.6 0.9 0.2 0.6 0.8 0.3 1 0.6 0.7 ...
##  $ y : num  0.824 0.444 2.531 1.344 2.635 ...
```

```
# rpart
rp_model <- rpart(y~., data=df)
varImp(rp_model)
```

```
##      Overall
## x1 1.1982394
## x2 1.3662639
## x3 0.9649745
## x4 0.8960316
```

We can see the tree mostly uses x1 to split and x4 the least. Though x2 and x3 are not used to generate target but they are also used by tree to split. With this simulation, it is evident here that there is a selection bias in the tree model where favored predictors have more distinct values.

Exercise 8.3

In stochastic gradient boosting, the bagging fraction and learning rate will govern the construction of the trees as they are guided by the gradient. Although the optimal values of these parameters should be obtained through the tuning process, it is helpful to understand how the magnitudes of these parameters affect the magnitudes of variable importance. Figure 8.24 provides the variable importance plots for boosting using two extreme values for the bagging fraction (0.1 and 0.9) and the learning rate (0.1 and 0.9) for the solubility data. The left-hand plot has both parameters set to 0.1, and the right-hand plot has both set to 0.9:

(a)

Why does the model on the right focus its importance on just the first few predictors, whereas the model on the left spreads importance across more predictors?

Seeing the graphs, the one on the right has higher bagging fraction and higher learning rate that means it used larger chunk of data and increases the correlation in every iteration. Thus only less number of variables are considered important. Not the plot on the left has lower bagging fraction and lower learning rate that means it uses small chunk of data for model training and less dependent in each iteration. Thus the model on the left spreads importance across more predictors.

(b)

Which model do you think would be more predictive of other samples?

Bagging fraction and learning rate are considered important params to control overfitting. Based on above explanation, the model with smaller bagging fraction and learning rate will lead to better generalization over the test/new data. Given that, the model with smaller learning rate and bagging fraction would be more predictive over other samples.

(c)

How would increasing interaction depth affect the slope of predictor importance for either model in Fig. 8.24?

Increasing the interaction depth would include more predictors and result to spread out importance. This would result the slope of predictor importance become flatten.

Exercise 8.7

Refer to Exercises 6.3 and 7.5 which describe a chemical manufacturing process. Use the same data imputation, data splitting, and pre-processing steps as before and train several tree-based models:

(a)

Which tree-based regression model gives the optimal resampling and test set performance?

(b)

Which predictors are most important in the optimal tree-based regression model? Do either the biological or process variables dominate the list? How do the top 10 important predictors compare to the top 10 predictors from the optimal linear and nonlinear models?

(c)

Plot the optimal single tree with the distribution of yield in the terminal nodes. Does this view of the data provide additional knowledge about the biological or process predictors and their relationship with yield?