Practical Machine Learning Course Project

Zhengguo Chu

31 December, 2016

Synopsis

This study use the data from Groupware website. http://groupware.les.inf.puc-rio.br/har The goal is to build a model to predict if participants perform barbell movement correctly or not. In this study, we chose random forest method and the result showed very good accuracy.

Data Processing

a). - Reading the Data:

The first step is to read the data from csv. There are lots of empty cells, as well as special values like "NA" and "#DIV/0!", therefore we did some special handling when importing the data.

```
pml_training <- read.csv(file = "pml-training.csv",
                         header = TRUE,
                         sep = ",",
                         na.strings = c("","" ", "NA", "#DIV/0!", NA_character_)
)


pml_testing <- read.csv(file = "pml-testing.csv",
                        header = TRUE,
                        sep = ",",
                        na.strings = c("","" ", "NA", "#DIV/0!", NA_character_)
)
```

b). - Clean the data

From the raw data we noticed that there were many columns having lots of NA or empty values, therefore we cleaned the data and focused only to those columns with majority non-empty or non-NA values. We used 19000 as the threshold. We also noticed that the first 7 columns were just timestamps or sequence numbers, which were not useful in this particular case, thus we will filter those columns as well. The data were reduced to 53 columns.

```
NA_Count <- sapply(pml_training, function(x) sum(length(which(is.na(x))))>19000)
NA_Count_T <- t(NA_Count)
col <- colnames(NA_Count_T)[apply(NA_Count_T, 1, function(u) u == FALSE)]
col <- col[8:60]
training_clean <- subset(pml_training, select = col)
dim(training_clean)
```

```
## [1] 19622    53
```

Build the Model

We used caret package to build the model.

Cross Validation: We split the original pml training data down to two parts: training and testing sets with p = 0.75.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.4
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
inTrain <- createDataPartition(training_clean$classe, p = 0.75, list = FALSE)
training <- training_clean[inTrain, ]
testing <- training_clean[-inTrain, ]
```

Model Selection: This project is to predict "Classe", which is a categorical factor. And we need a very high accuracy according to the required model accuracy for Course project and we want the out of sample error to be less than 0.01. Random forrest usually have very high accuracy and is natually suitable for factor variable. Therefore we chose random forest as our method. However, the average running time is very slow.

To help run the model, we used parellel library as discussed in the course forum. Improving Performance of Random Forest in caret::train(). We also used a beep function to remind us when it finished and monitor the time it used.

```
library(beepr)
set.seed(1)
fitControl <- trainControl(method = "cv",
                           number = 10,
                           allowParallel = TRUE
)

# Make Cluster
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
cluster <- makeCluster(detectCores()- 1)
registerDoParallel(cluster)

# Begin the timer
ptm <- proc.time()

# Fit the model
modFit <- train(classe ~ .,
                data = training,
                method = "rf",
                trainControl = fitControl
                )
```

```
## Loading required package: randomForest

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# End the timer
usedtime <- proc.time() - ptm

# Stop Cluster
stopCluster(cluster)
registerDoSEQ()

# Issue Beep
beep()
```

The total run time for building the random forest model is:

```
usedtime
```

```
##     user   system  elapsed
##   53.891    1.210 1119.013
```

The final model is as below:

```
modFit
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9893964  0.9865846
##   27    0.9895907  0.9868307
##   52    0.9806604  0.9755316
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Test the Model

At this point we can use the put-aside testing data to validate the model.

```
pred <- predict(modFit, newdata = testing)
table(pred, testing$classe)
```

```
##
## pred    A    B    C    D    E
##    A 1395    4    0    0    0
##    B    0  941    4    0    0
##    C    0    4  848    5    0
##    D    0    0    3  798    0
##    E    0    0    0    1  901
```

```
confusionMatrix(testing$classe, pred)$overall['Accuracy']
```

```
##  Accuracy
## 0.9957178
```

```
confusionMatrix(testing$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    4  941    4    0    0
##          C    0    4  848    3    0
##          D    0    0    5  798    1
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9957
##                  95% CI : (0.9935, 0.9973)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9958   0.9895   0.9963   0.9989
## Specificity            1.0000   0.9980   0.9983   0.9985   1.0000
## Pos Pred Value         1.0000   0.9916   0.9918   0.9925   1.0000
## Neg Pred Value         0.9989   0.9990   0.9978   0.9993   0.9998
## Prevalence             0.2853   0.1927   0.1748   0.1633   0.1839
## Detection Rate         0.2845   0.1919   0.1729   0.1627   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9986   0.9969   0.9939   0.9974   0.9994
```

The overall accuracy is about 0.9957, the out of sample error is 0.0043, which is less than 0.01

Apply this model to predict the new data

Then we can use this model to predict the original pml-testing data. We need to do the same filtering for the this testing data too.

```
pml_testing$classe <- NA
final_testing <- subset(pml_testing, select = col)
final_pred <- predict(modFit, final_testing)
final_pred
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

Conclusion

This model correctly predicted all the 20 testing cases.