Prediction Assignment Report

Man Zhu

2016/11/25

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. And the goal of the project is to predict the manner in which they did the exercise

Load R packages

```
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
```

Read the data

I have downloaded the data from the given URL and stored the data in the default location. Then, we can read the two csv files.

```
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

```
dim(training)
```

```
## [1] 19622   160
```

```
dim(testing)
```

```
## [1]  20 160
```

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables.

Clean the data

Before we bulid the forcasting model, the raw data should be cleaned. We will remove the observations with missing values as well as some meaningless variables.

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
classe <- training$classe # save the classe data before be removed
trainremove <- grepl("^X|timestamp|window", names(training))
testremove <- grepl("^X|timestamp|window", names(testing))
training <- training[, !trainremove]
testing <- testing[, !testremove]
traincleaned <- training[, sapply(training, is.numeric)]
testcleaned <- testing[, sapply(testing, is.numeric)]
traincleaned$classe <- classe # add classe to the traincleaned dataframe
testcleaned <- testcleaned[, -ncol(testcleaned)] # remove the problem_id in testdata
```

Slice the data

Then, we will divide the traincleaned into a trainData (70%) and a testData (30%) and use the testData to conduct cross validation in the near future.

```
set.seed(8110)
inTrain <- createDataPartition(traincleaned$classe, p = 0.70, list = F)
trainData <- traincleaned[inTrain, ]
testData <- traincleaned[-inTrain, ]
```

Bulid the model

We bulid a forcasting model using random forest algorithm because it is unexcelled in accuracy, runs efficiently on large data bases and gives estimates of what variables are important in the classification. In the validation step, 5-fold cross validation is used when applying the rf algorithm.

```
modRf <- train(classe ~ ., data = trainData, method = "rf",
               trControl = trainControl(method = "cv", 5), ntree = 100)
```

Show the model:

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10991, 10990, 10988
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9892990  0.9864624
##   27    0.9896625  0.9869223
##   52    0.9824560  0.9778026
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Then, we see the prediction performance

```
predictRf <- predict(modRf, testData)
CM <- confusionMatrix(testData$classe, predictRf)
accuracy <- postResample(predictRf, testData$classe)
Esample_error <- - as.numeric(accuracy[1]) + 1
```

Let's see the outcome of Confusion Matrix:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    0    2    0    1
##          B    4 1134    1    0    0
##          C    0    3 1023    0    0
##          D    0    1   11  952    0
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9965   0.9865   0.9969   0.9991
## Specificity            0.9993   0.9989   0.9994   0.9976   0.9994
## Pos Pred Value         0.9982   0.9956   0.9971   0.9876   0.9972
## Neg Pred Value         0.9991   0.9992   0.9971   0.9994   0.9998
## Prevalence             0.2846   0.1934   0.1762   0.1623   0.1835
## Detection Rate         0.2839   0.1927   0.1738   0.1618   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9977   0.9929   0.9972   0.9992
```

The accuracy of the model:

```
##  Accuracy     Kappa
## 0.9955820 0.9944116
```

The expected out of sample error:

```
## [1] 0.004418012
```

Thus, we can conclude that the accuracy of the model is about 99.56% and the expected out of sample error is about 0.44%.

Predicting for test data set

Now, we apply the model to the original testing data set (have been cleaned)

```
pred <- predict(modRf, testcleaned)
```

Show the prediction result for the test data set:

```
##  [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
```

One figure: Visualization for the Decision Tree

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel)
```