Project Report - Practical Machine Learning Course

Mario Melchiori

November 21, 2015

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, your goal will be to 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. More information is available from the website here: (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of the project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We will also use your prediction model to predict 20 different test cases.

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. They have been very generous in allowing their data to be used for us.

Load data

```
pml.training <- read.csv("~/datasciencecoursera/pml-training.csv",na.strings = c("NA","#DIV/0!"))
pml.testing <- read.csv("~/datasciencecoursera/pml-testing.csv",na.strings = c("NA", "#DIV/0!"))
```

Cleaning data

We choose the features that contains the labels arm or belt or dumbbell or forearm and delete the features with 19000 rows or more NA values.

```
response <- pml.training[ ,160]
pml.training$classe <- response
pml.training <- pml.training[ ,grep("arm|belt|dumbbell|forearm", names(pml.training))]
pml.training <- pml.training[ ,colSums(is.na(pml.training)) < 19000]
pml.training$classe <- response
```

First, we inspect no NA values in the dataset. Then, we split the dataset into two. We randomly subsample 60% of the data for training purposes, while the 40% remainder will be used only for testing, evaluation and accuracy measurement.

```
table(complete.cases(pml.training))
```

```
##
##  TRUE
## 19622
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.2
```

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

```
inTrain <- createDataPartition(y=pml.training$classe,
                               p=0.6, list=FALSE)
training <- pml.training[inTrain,]
testing <- pml.training[-inTrain,]
rm(list = "pml.training")
```

Model Selection

We will test the Random Forest's mtry parameter and set the mtree parameter equal to 400:

```
library(randomForest)
```

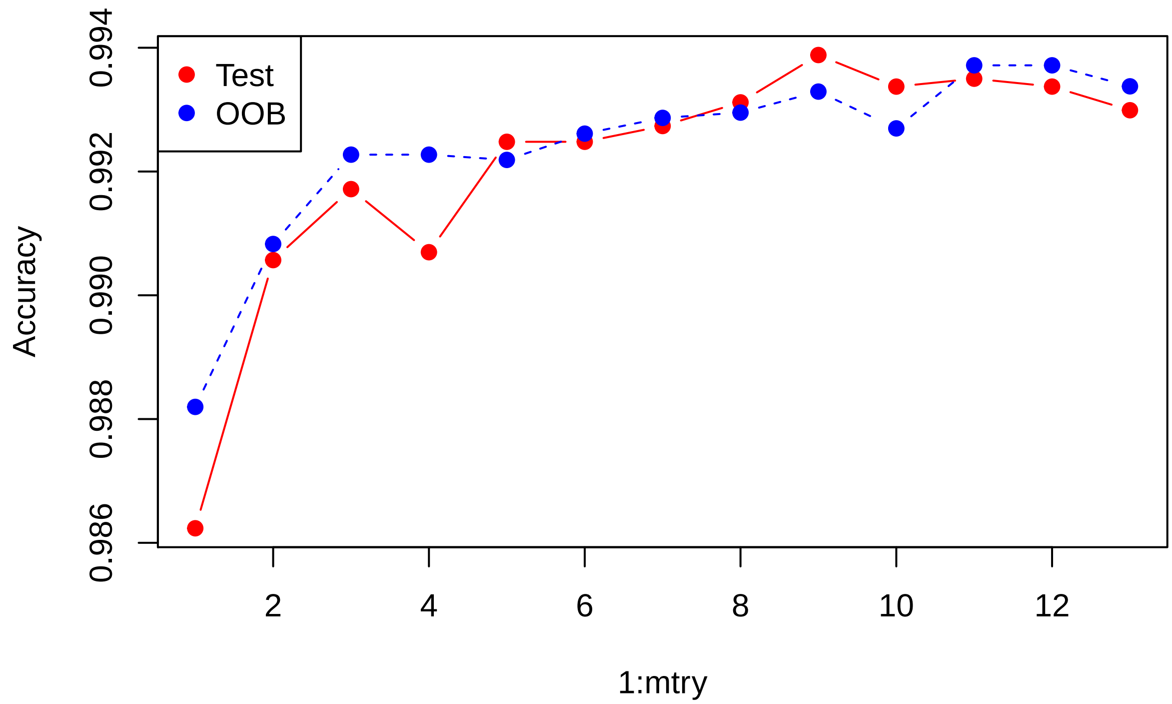
```
## Warning: package 'randomForest' was built under R version 3.2.2
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(22115)
oob.err=double(13)
test.err=double(13)
for(mtry in 1:13){
  fit=randomForest(classe ~ ., data=training,mtry=mtry,ntree=400)
  oob.err[mtry]=fit$err.rate[400]
  pred=predict(fit,testing)
  test.err[mtry]=postResample(pred, testing$classe) [1]
  cat(mtry," ")
}
```

```
## 1  2  3  4  5  6  7  8  9  10  11  12  13
```

```r
matplot(1:mtry,cbind(test.err,1-oob.err),pch=19,col=c("red","blue"),type="b",ylab="Accuracy")
legend("topleft",legend=c("Test","OOB"),pch=19,col=c("red","blue"))
```




```r
which(test.err==max(test.err), arr.ind=TRUE)
```

```
## [1] 9
```

```r
which(1-oob.err==max(1-oob.err), arr.ind=TRUE)
```

```
## [1] 11 12
```

We choose mtry parameter equal to 10 and mtree equal to 400.

Variables selection

To select the features we use cross validation:

```r
cvfeatures <- rfcv(trainx = training[,-53], trainy = training[,53],
                   scale = FALSE, step=-10, cv.fold=5)
cbind(cvfeatures $error.cv)
```

```
##           [,1]
## 52 0.008576766
## 42 0.009765625
## 32 0.009510870
## 22 0.011803668
## 12 0.011718750
## 2  0.349099864
```

By using 52 features we obtain the lower croos validation error. Below wee see the importance each variable.

```r
rf=randomForest(classe ~ ., data=training, mtree = 400, mtry = 10)
rfImportance <- data.frame(variable = names(rf$importance[,1]), importance = rf$importance[,1])
rfImportance <- rfImportance[ order(-rfImportance[,2]),]
rownames(rfImportance ) <- NULL
rfImportance
```

```
##                variable importance
## 1             roll_belt  836.57569
## 2              yaw_belt  565.56310
## 3         pitch_forearm  543.06300
## 4     magnet_dumbbell_z  487.96201
## 5            pitch_belt  452.61912
## 6     magnet_dumbbell_y  440.43108
## 7          roll_forearm  384.96473
## 8     magnet_dumbbell_x  283.26775
## 9         roll_dumbbell  247.12945
## 10     accel_dumbbell_y  243.80690
## 11         accel_belt_z  242.76583
## 12        magnet_belt_z  233.34387
## 13        magnet_belt_y  225.18065
## 14      accel_forearm_x  192.99654
## 15             roll_arm  189.89152
## 16     accel_dumbbell_z  185.88323
## 17         gyros_belt_z  182.24007
## 18     magnet_forearm_z  171.21914
## 19 total_accel_dumbbell  157.14708
## 20     accel_dumbbell_x  152.95133
## 21         yaw_dumbbell  149.70641
## 22        magnet_belt_x  148.29618
## 23      accel_forearm_z  147.43516
## 24     gyros_dumbbell_y  139.98194
## 25         magnet_arm_x  136.29281
## 26              yaw_arm  136.27794
## 27         magnet_arm_y  131.92958
## 28     magnet_forearm_y  123.75434
## 29          accel_arm_x  116.68057
## 30     magnet_forearm_x  115.61849
## 31     total_accel_belt  115.56206
## 32         magnet_arm_z  102.29711
## 33          yaw_forearm   96.38120
## 34       pitch_dumbbell   95.72588
## 35            pitch_arm   92.00704
## 36          accel_arm_y   87.68141
```

```
## 37     gyros_dumbbell_x   76.11465
## 38          gyros_arm_y   75.29043
## 39      accel_forearm_y   74.13994
## 40          gyros_arm_x   71.22225
## 41          accel_arm_z   69.48630
## 42      gyros_forearm_y   66.63572
## 43         accel_belt_y   62.72269
## 44         gyros_belt_y   62.28471
## 45  total_accel_forearm   60.45609
## 46         accel_belt_x   59.99409
## 47      total_accel_arm   54.21392
## 48         gyros_belt_x   52.99795
## 49     gyros_dumbbell_z   49.14100
## 50      gyros_forearm_z   46.75226
## 51      gyros_forearm_x   40.72914
## 52          gyros_arm_z   33.14598
```

Model Validation

We proceed to examine how well the model fits on the data testing set.

```
predictionTesting <- predict(rf, newdata = testing)
confusionMatrix(predictionTesting, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    7    0    0    0
##          B    1 1496    7    0    0
##          C    0   15 1360   12    1
##          D    0    0    1 1270    4
##          E    1    0    0    4 1437
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9912, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9855   0.9942   0.9876   0.9965
## Specificity            0.9988   0.9987   0.9957   0.9992   0.9992
## Pos Pred Value         0.9969   0.9947   0.9798   0.9961   0.9965
## Neg Pred Value         0.9996   0.9965   0.9988   0.9976   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
```

```
## Detection Rate         0.2842   0.1907   0.1733   0.1619   0.1832
## Detection Prevalence   0.2851   0.1917   0.1769   0.1625   0.1838
## Balanced Accuracy      0.9989   0.9921   0.9949   0.9934   0.9979
```

Model Testing

Finaly, we use the model to write the results on the test set, as instructed:

```
predict_testing <- predict(rf,pml.testing)
predictionTesting = predict(rf ,newdata = pml.testing)
answers <- as.character(predict(rf, pml.testing))
answers
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```