Machine Learning Project

Alvaro Fierro

21 de noviembre de 2015

Background

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).

According to this information, the variable «classe» has five possible values: A (correct exercise), B-E (common mistakes).

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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Loading the data

```
trainData <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

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

trainFile <- "C:/Users/Usuario/Google Drive/R programming/8. Practical Machine Learning/Project 1/pml-t

testFile <- "C:/Users/Usuario/Google Drive/R programming/8. Practical Machine Learning/Project 1/pml-te

if(!file.exists(trainFile)) {
        download.file(trainData, destfile = trainFile)
}

if(!file.exists(testFile)) {
        download.file(testData, destfile = testFile)
}

trainn <- read.csv(trainFile)
test <- read.csv(testFile)
```

Loading the libraries

```
library(plyr)
library(caret) # Machine learning library
```

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

```
library(ggplot2)
library(randomForest) # classification and regression
```

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

```
library(rpart) # regression partitioning and trees
library(rpart.plot) #pretty printing trees
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Versión 4.0.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

Preliminary data processing

Train and test data have different columns.

```
classe <- names(train)[colnames(train)!=colnames(test)]
problem_id <- names(test)[colnames(train)!=colnames(test)]
```

We are going to remove unnecesary and steady columns and detect columns with missing values

```
# Remove first five columns

trainn <- trainn[,6:dim(trainn)[2]]
test <- test[,6:dim(test)[2]]

# Select not steady columns
not_steady <- !nearZeroVar(trainn, saveMetrics = T)$nzv
trainn <- trainn[,not_steady]
test <- test[,not_steady]

# Find columns > 70% na values
nmissing <- function(x) sum(is.na(x))
```

```
few_na<-!colwise(nmissing)(trainn)> 0.7*dim(trainn)[1]
trainn <- trainn[,few_na]
test <- test[,few_na]
```

Let us take a look at the subset of variables

```
names(trainn)
```

```
##  [1] "num_window"           "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

Select training and cross validation sets.

```
set.seed(100)
inTrain <- createDataPartition(y=trainn$classe,p=0.75,list=FALSE)

training <- trainn[inTrain,]
validation <- trainn[-inTrain,]
```

Data Modelling

We are going to test the several models and compare the goodness of fit

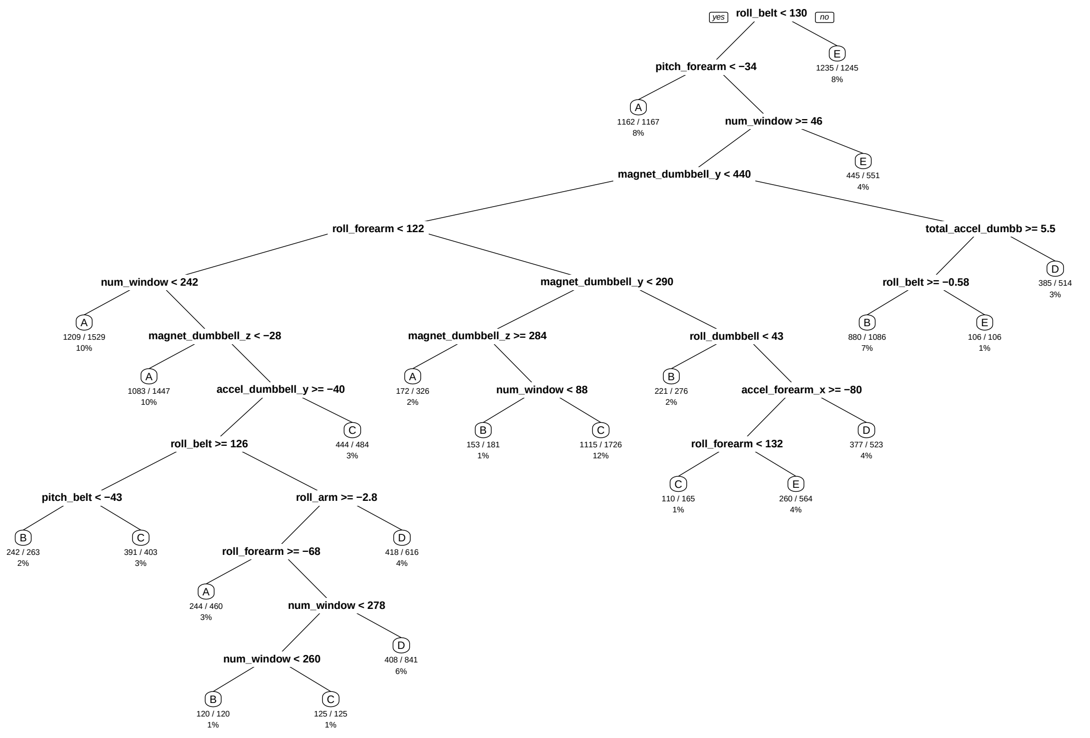
Decision Trees

```
model_dt <- rpart(classe ~ ., data=training, method="class")
predict_dt <- predict(model_dt, validation, type="class")
conf_dt<-confusionMatrix(predict_dt,validation$classe)
conf_dt
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1289  223   28   60   47
##          B   40  521   31   22   21
##          C   12   71  700  115   58
##          D   41   99   76  534  105
##          E   13   35   20   73  670
##
## Overall Statistics
##
##                Accuracy : 0.7573
##                  95% CI : (0.7451, 0.7693)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6915
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9240   0.5490   0.8187   0.6642   0.7436
## Specificity            0.8980   0.9712   0.9368   0.9217   0.9648
## Pos Pred Value         0.7826   0.8205   0.7322   0.6246   0.8261
## Neg Pred Value         0.9675   0.8997   0.9607   0.9333   0.9436
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2628   0.1062   0.1427   0.1089   0.1366
## Detection Prevalence   0.3358   0.1295   0.1949   0.1743   0.1654
## Balanced Accuracy      0.9110   0.7601   0.8777   0.7929   0.8542
```

```
rpart.plot(model_dt, extra=102, under = TRUE, faclen=0)
```



The accuracy of the decision tree is 0.7573409.

Random Forests

```
set.seed(100)
model_rf <- randomForest(classe ~ ., data=training, type="class")
predict_rf <- predict(model_rf, validation)
conf_rf<-confusionMatrix(predict_rf,validation$classe)
conf_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  947    1    0    0
##          C    0    1  854    5    0
##          D    0    0    0  799    3
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9978
##                  95% CI : (0.996, 0.9989)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9972
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9988   0.9938   0.9967
## Specificity            0.9997   0.9997   0.9985   0.9993   1.0000
## Pos Pred Value         0.9993   0.9989   0.9930   0.9963   1.0000
## Neg Pred Value         1.0000   0.9995   0.9998   0.9988   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1741   0.1629   0.1831
## Detection Prevalence   0.2847   0.1933   0.1754   0.1635   0.1831
## Balanced Accuracy      0.9999   0.9988   0.9987   0.9965   0.9983
```

The accuracy of the random forest is 0.9977569.

The comparison between the two methods shows that random forests performs better.

Test data prediction

```r
predict_rt_test <- predict(model_rf, test)
predict_rt_test
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Coursera assignment

```r
predictions <- as.vector(predict_rt_test)

pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE,
            col.names = FALSE)
    }
}

pml_write_files(predictions)
```