Exercise Manner Prediction

Blas López

24 d'abril 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).

Extracted from Course Project Assignment

Code book

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

There are a high number of columns in these data sets, some are irrelevant for this study (as "username", several kind of timestamps,...) and others will be used as predictor below.

The singular variable, with a complete mean in this study, is `classe` which is a factor cotaining the following values according to the manner how execercises are done:

- `A` exactly according to the specification (Class A)
- `B` throwing the elbows to the front (Class B)
- `C` lifting the dumbbell only halfway (Class C)
- `D` lowering the dumbbell only halfway (Class D)
- `E` throwing the hips to the front (Class E)

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3XOq13lDi

Extracted from Course Project Assignment

Requeriments

The following libraries are required in order to run the project :

```r
library(RCurl) # For loading data from a http connection
# For classification and regression and model fitting
library(caret)
library(randomForest)
library(gbm)
library(plyr) # required by gbm
```

Reproducibility

A pseudo-random number is setted as 129 (`myseed`) in order that the reproduction of this scripts produces the same results.

```
set.seed(myseed)
```

Data Processing

Loading the data

The data is loaded using the provided URL in the `training` variable and replacing several known null string values as `NA`.

The dataset for the final part of the project is, also, loaded in the `testing` variable and the same columns, as in the `training` data set, will be removed.

```
# The training set load
trainConn <- getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
training <- read.csv(text = trainConn, na.strings=c("NA","#DIV/0!",""))
rm(trainConn)

# The dataset to testing and predicting the final models
testConn <- getURL("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
testing <- read.csv(text = testConn, na.strings=c("NA","#DIV/0!",""))
rm(testConn)
```

After loading, the `training` data set contains 160 variables and 19622 rows.

*The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. from the **Qualitative Activity Recognition of Weight Lifting Exercises**[1].*

Read more: http://groupware.les.inf.puc-rio.br/har#literature#ixzz3XOQDGre7

Cleaning the data

Looking at the seven first columns in dataset it seems to be no needed for this study and, then, are removed.

```
names(training[,c(1:7)])
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

```
# Irrelevant columns removed
training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]
```

The `NA` values are clean after loading and, the zero covariate predictors, calculated using ***nearZeroVar*** , are removed from the dataset function (if needed)

[1] Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

```r
# Removing NA columns (getting only with no NA values)
keepcols <- colSums(is.na(training)) == 0
training <- training [ , keepcols]
testing <- testing [ , keepcols]

# determines near zero covariate predictors
nzv <- nearZeroVar(training)
# ... and remove it from training set (if any exists)
if ( length(nzv) > 0 ) {
   training <- training[,-nzv]
   testing <- testing[,-nzv]
}
```

After cleaning the `training` data set contains 53 variables and 19622 rows.

Note. These cleaning operations have been done without evaluating the impact of each one, but as methodological steps

Partitioning

The `training` data set is partitioned into two subsets (70% and 30% from the original one) in order to perform a cross-validation process.

```r
inTrain <- createDataPartition( y = training$classe, p=.7, list=FALSE)
sTraining <- training[inTrain,]
sTesting <- training[-inTrain,]
```

Data Set	Rows	Columns
training subset	13737	53
testing subset	5885	53

Modeling

Two models will be fitted using **Random Forest** and **Gradient Boosted Machine** and then these models will be compared in accuracy using the prediction done with each model.

```r
# Random Forest Model
rfModel <- randomForest(classe ~ . ,data = sTraining, method = "class")

# Gradient Boosted Machine

tc <- trainControl("repeatedcv", number=10, repeats=4,
                   classProbs=TRUE, savePred=T,
                   allowParallel=TRUE)
gbmModel <- train(classe ~ . , data = sTraining, method = "gbm", trControl=tc, verbose = FALSE)
```

Model comparisson

```r
# Confusion Matrix from each model
# now with random forest model
# ... predict using the testing subsample data set
rfPrediction <- predict(rfModel, sTesting)
# ... and generates the confusion matrix
rfConfusion <- confusionMatrix(sTesting$classe,rfPrediction)

# ... and now with boosted
gbmPrediction <- predict(gbmModel, sTesting)
gbmConfusion <- confusionMatrix(sTesting$classe,gbmPrediction)
```

Model	Accuracy	Kappa
Random Forest	0.9949023	0.9935514
Gradient Boosted Machine	0.9607477	0.9607477

Model selection

As shown in the previous table the accuracy for Random Forest (0.9949023) is better than Gradient Boosted Machine. The first one will be the selected model.

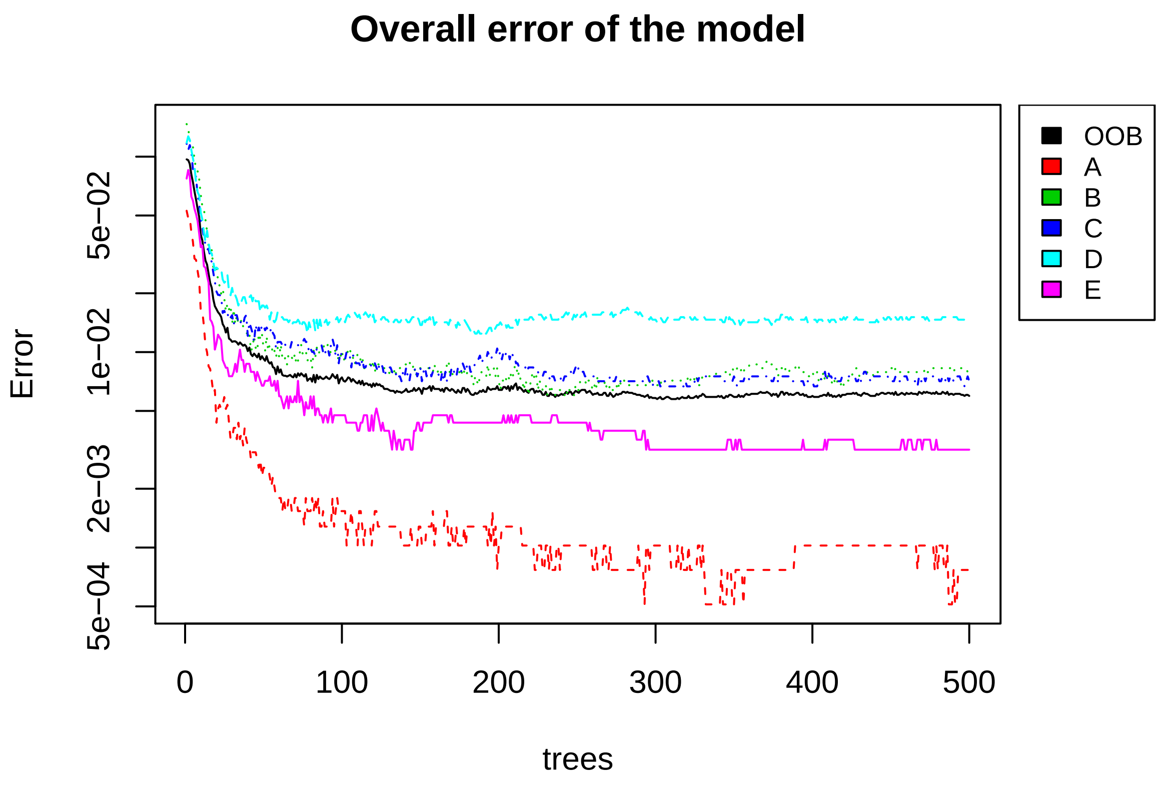
The selected model is **Random Forest**

The expected out-of-sample error is `1 - accuracy` (that means 0.0050977) in the final model fitted by the training dataset.

The next plot is about the overall error of the model for each `classe` plus the *Out-of-bag* (OOB).

```r
# two plots one for the plot and another for legend
layout(matrix(c(1,2),nrow=1),  width=c(4,1))
# No margin on the right side
par( mar=c(5,4,4,0))
plot(rfModel, log="y", main = "Overall error of the model" )

#No margin on the left side
par(mar=c(5,0,4,2))
plot(c(0,1),type="n", axes=F, xlab="", ylab="")
legend("top", colnames(rfModel$err.rate),col=1:6,cex=0.8,fill=1:6)
```



Finally, the confusion matrix of the selected is :

```
##      A    B    C    D    E  class.error
## A 3903    3    0    0    0 0.0007680492
## B   12 2636   10    0    0 0.0082768999
## C    0   16 2379    1    0 0.0070951586
## D    0    0   29 2220    3 0.0142095915
## E    0    0    1    7 2517 0.0031683168
```

See Annex 1 for detailed confusion matrix

Submission

When `training` dataset was loaded, a new dataset (`testing`) was loaded too, in order to be used with the selected model and predict new values.

Now, this second dataset, is used as input to predict the values with the selected model and to generate the files to submit for the final part of the Course Project.

```
# function for generation of submission file
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)
  }
}
```

```r
# make the prediction
answers <- predict(rfModel, testing)
pml_write_files(answers)
```

The final answers are :

```
##  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
```

Annex 1 Confusion Matrix

Random Forest Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    5 1129    5    0    0
##          C    0    5 1021    0    0
##          D    0    0    9  951    4
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9947   0.9865   0.9989   0.9963
## Specificity            0.9998   0.9979   0.9990   0.9974   0.9998
## Pos Pred Value         0.9994   0.9912   0.9951   0.9865   0.9991
## Neg Pred Value         0.9988   0.9987   0.9971   0.9998   0.9992
## Prevalence             0.2851   0.1929   0.1759   0.1618   0.1844
## Detection Rate         0.2843   0.1918   0.1735   0.1616   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9963   0.9927   0.9982   0.9981
```

Gradient Boosted Machine Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1645   22    4    2    1
##          B   31 1082   22    1    3
##          C    0   36  978   11    1
##          D    0    1   42  916    5
##          E    4   17   15   13 1033
##
## Overall Statistics
##
##                Accuracy : 0.9607
##                  95% CI : (0.9555, 0.9656)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9503
##  Mcnemar's Test P-Value : 1.694e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9792   0.9344   0.9218   0.9714   0.9904
## Specificity            0.9931   0.9879   0.9900   0.9903   0.9899
## Pos Pred Value         0.9827   0.9500   0.9532   0.9502   0.9547
## Neg Pred Value         0.9917   0.9840   0.9829   0.9945   0.9979
## Prevalence             0.2855   0.1968   0.1803   0.1602   0.1772
## Detection Rate         0.2795   0.1839   0.1662   0.1556   0.1755
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9861   0.9612   0.9559   0.9808   0.9901
```

Annex 2. Importance of predictors

The next table shows the 20 first predictors by importance for the selected model.

```
i <- importance(rfModel)
df <- data.frame(dimnames(i)[[1]],i,row.names=NULL)
names(df) <- c("Predictor","Overall")
head(df[with(df,order(-df$Overall)),],20)
```

```
##              Predictor  Overall
## 1            roll_belt 855.8542
## 3             yaw_belt 645.6858
## 41       pitch_forearm 545.5458
## 39    magnet_dumbbell_z 536.3517
## 2           pitch_belt 488.2546
## 38    magnet_dumbbell_y 452.4319
## 40        roll_forearm 417.1868
## 37    magnet_dumbbell_x 365.0276
## 27       roll_dumbbell 297.9266
## 35     accel_dumbbell_y 290.1357
## 12        magnet_belt_y 287.1905
## 10         accel_belt_z 282.8754
## 13        magnet_belt_z 282.2240
## 47      accel_forearm_x 230.5697
## 36     accel_dumbbell_z 223.9844
## 14             roll_arm 222.5898
## 7           gyros_belt_z 204.8664
## 52     magnet_forearm_z 198.9247
## 30 total_accel_dumbbell 188.7698
## 49      accel_forearm_z 183.7449
```

References