Machine Learning Assignment

Asier Goikoetxea

20 de julio de 2016

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

First of all, we are going to start by loading the required packages and the reading the data. Since training data is also quite big, we are going to split it again to create a validation dataset that will be very usefull for cross-validation analyzing the accuracy and other variables of the different Machine Learning Methods before we create our final model and use it in the Test dataset.

```
#Load packages and read the data
library(caret)
```

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

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

```
library(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
```

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

Exploratory Analysis and Preprocessing

Looking at the dimmension of the training set, we can see it has 15699 observations and 160 variables. Our goal is to create the best model to predict the "classe" variable.

```r
#overall structure of the data and somre plots
 dim(train_data)
```

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

```r
#variable names
 names(train_data)
```

```
##   [1] "X"                        "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

However, first we will preprocess de data to improve the performance of our analysis. We start by removing the columns that contain NA values and also removing some unnecessary variables like: X, user name, timestamp and new_window.

```r
# remove columns with NA values
slim_training <- train_data[,!sapply(train_data,function(x) any(is.na(x)))]

# remove useless variables
del_log <- grepl("X|user_name|timestamp|new_window", colnames(slim_training))
slim_training <- slim_training[, !del_log]
```

Next we will perform a Near Zero Variance analysis to the data to identify the variables that don't add information because they have almost no variance and remove this variables.

```r
# remove zero variance variables
nzv <- nearZeroVar(slim_training, saveMetrics = TRUE)
slim_training <- slim_training[, !nzv$nzv]
dim(slim_training)
```

```
## [1] 19622    54
```

With this preprocessing methods we have reduced the variables to 54.

Model Fitting

First we are going to create a data partition to have a training and Validation set to perform cross validation analysis before we use our model with the test dataset.

```
#Create data partition to have a validation dataset
set.seed(2233)
inTrain <- createDataPartition(slim_training$classe, p=0.7, list = FALSE)
training <- slim_training[inTrain, ]
validation <- slim_training[-inTrain, ]
```

We will fit a random forest model and analyze the model using the validation set and the confussionMatrix function. We will use the randomForest function directly instead of caret package to improve speed.

```
#create a few training models: random forests, boosting, lda...

forest_model <- randomForest(classe~., data=training, importance=TRUE, ntree=100)
forest_pred <- predict(forest_model, validation)
confusionMatrix(forest_pred, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1135    2    0    0
##          C    0    0 1024    6    0
##          D    0    0    0  958    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9956, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9981   0.9938   0.9963
## Specificity            0.9991   0.9996   0.9988   0.9992   1.0000
## Pos Pred Value         0.9976   0.9982   0.9942   0.9958   1.0000
## Neg Pred Value         1.0000   0.9992   0.9996   0.9988   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1740   0.1628   0.1832
## Detection Prevalence   0.2851   0.1932   0.1750   0.1635   0.1832
## Balanced Accuracy      0.9995   0.9980   0.9984   0.9965   0.9982
```

With the random forest model, we get a 0.9971 accuracy in the validation dataset. The 95% confidence intervals are 0.9954 and 0.9983

We will also fit a General Boosted Method "GBM" and repeat the process to compare the two models.

```r
#create a few training models: random forests, boosting, lda...

gbm_model <- train(classe~., data= training, method="gbm", verbose=FALSE)
```

```
## Loading required package: gbm

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1

## Loading required package: plyr
```

```r
#create predictions using validation dataset
gbm_pred <- predict(gbm_model, validation)
```

```r
confusionMatrix(gbm_pred, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667   14    0    0    0
##          B    7 1117    6    5    5
##          C    0    8 1017   18    3
##          D    0    0    2  939   15
##          E    0    0    1    2 1059
##
## Overall Statistics
##
##                Accuracy : 0.9854
##                  95% CI : (0.982, 0.9883)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9815
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9807   0.9912   0.9741   0.9787
## Specificity            0.9967   0.9952   0.9940   0.9965   0.9994
## Pos Pred Value         0.9917   0.9798   0.9723   0.9822   0.9972
## Neg Pred Value         0.9983   0.9954   0.9981   0.9949   0.9952
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2833   0.1898   0.1728   0.1596   0.1799
## Detection Prevalence   0.2856   0.1937   0.1777   0.1624   0.1805
## Balanced Accuracy      0.9962   0.9879   0.9926   0.9853   0.9891
```

Using GBM we get a 0.9861 accuracy in the validation dataset. The 95% confidence intervals are 0.9827 and 0.9889 So comparing the two models, the random forest method has the best results.

Conclusions

To end the assignment, we will test our model with the test dataset to see the results.

```r
final_predictions <- predict(forest_model, testing)
final_predictions
```

```
##  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
```