Prediction Of Wieght Lifting Results

Francois Laforgia

24 Jan 2016

Summary

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, data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants is used. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The aim of the project is to predict which way the exercise has been done basing on a machine learning algorithm.

Data Source

The source of the data comes from http://groupware.les.inf.puc-rio.br/har.
The data set is provided in 2 spurces, the training data set used to build the algorithm is available on https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv and the data set used to do the final prediction used for the project is on https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv.

Data Exploratory

Data Loading

After loading the data, the first was to split the training set in two parts, a training and a testing part. Doing that allows me to keep untouched the testing set for the project and to reduce the risk to have a bad prediction because of overfitting.

```
library(dplyr)
library(caret)
library(randomForest)

setwd("/Users/flaforgia/Documents/PML/Project")
HAR.training <- read.csv("pml-training.csv", na.strings=c("NA", "#DIV/0!"))

options(max.print=999999)
set.seed(3233)

train <- createDataPartition(y=HAR.training$classe, p=0.6, list=FALSE)
training <- HAR.training[train,]
testing <- HAR.training[-train,]
```

Data Cleaning

I decided to remove all the variables that have a zero variance or that are close to variance of zero. These nzv variables will not have an impact on the model fitting because they have few, if none, variation and thus it

is like adding a constant variable to the computation. Furthermore this could cause an overfitting of the predictors. To remove these predictora, I used the command nzv provided with the package caret.
After the nzv variables were removed I also removed the NA variables. Those variables can cause a bias in thre model. But to keep an uncertainity that could avoid again an overfitting, I only removed the predictors when the NA is above a threshold of 90% (10599 NA).
And finally I removed the seven first columns which are not specific to the data mesured.

At the end it appears that the predictores removed are the ones calculated like the standard deviation or the mean of the Euler's angles (cf. the pdf document available on http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf).

```
training.nzv <- nzv(training)
filtered.training <- training[,-training.nzv]
training.set <- data.frame(apply(filtered.training,2,function(x){sum(is.na(x))}))
training.set <- cbind(rownames(training.set) , training.set)
colnames(training.set) <- c("sensor", "sum.NA")

names.factor <- c()
for (i in 1:nrow(training.set)) {
       if (training.set[i,2] <= 10599) {
              names.factor <- c(names.factor, as.character(training.set[i,1]))
        }
}

training.final <- filtered.training[,names.factor]
training.final <- training.final[,-(1:7)]
colnames(training.final)
```

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

Model Fitting

Before we run the train method, I executed a preprocessing step with Principal Component Analysis to confirm that the remaining predictors are or not correlated. I used a threshold of 99% to keep the best accuracy possible. Once the PCA was done, I created a new train variable that will be used for the train function.

```
preprocess.HAR.pca <- preProcess(training.final[,-52], method="pca", thresh=0.99)
train.HAR <- predict(preprocess.HAR.pca, training.final[,-52])
preprocess.HAR.pca
```

```
## Created from 11776 samples and 51 variables
##
## Pre-processing:
##   - centered (51)
##   - ignored (0)
##   - principal component signal extraction (51)
##   - scaled (51)
##
## PCA needed 36 components to capture 99 percent of the variance
```

The preprocessing step shows that only 36 predictors were used to reach 99% of accuracy.
Finally I fitted the model with a random forest which is the alogrithm that gives the better accuracy. I used the default options for this operation

```
fit.rf <- train(training.final$classe~., method="rf", data=train.HAR)
```

Model Verification

To verify the accuracy, I use it against the test dataset built earlier and I display the confusionMatrix. Before I can predict from the test set I applied the pca modification as I did fro the training set.

```
testing.rf <- predict(preprocess.HAR.pca, testing[,-160])
testing.final <- predict(fit.rf, testing.rf)
confusionMatrix(testing$classe, testing.final)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2214    9    6    3    0
##          B   34 1469   12    1    2
##          C    4   16 1334   14    0
##          D    2    3   57 1218    6
##          E    0    5    7    3 1427
##
## Overall Statistics
##
##                Accuracy : 0.9765
##                  95% CI : (0.973, 0.9798)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9703
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9823   0.9780   0.9421   0.9831   0.9944
## Specificity            0.9968   0.9923   0.9947   0.9897   0.9977
## Pos Pred Value         0.9919   0.9677   0.9751   0.9471   0.9896
## Neg Pred Value         0.9929   0.9948   0.9873   0.9968   0.9988
## Prevalence             0.2873   0.1914   0.1805   0.1579   0.1829
## Detection Rate         0.2822   0.1872   0.1700   0.1552   0.1819
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9895   0.9852   0.9684   0.9864   0.9960
```

Conclusion

Based on the result from the confusionMatrix, the model has an accuracy around 98% which is good. I will use this model to predict the classes on the validation set provided.