PreMachLearn_031 - Course Project

Miguel Sena e Silva

Sunday, August 29, 2015

Executive Summary

This project uses data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways (classified in the “classe” variable: A - Correctly; B-E - Incorrectly). More information on the data source is available from this website (see the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here. With this data (19.622 observations of 160 variables) we must develop a machine learning algorithm that “learns” how to classify new data (not yet classified). This exercise involves using our prediction model to predict 20 different test cases, available here.

This document describes:

- how I built my model,
- how I used cross validation,
- what the expected out of sample error I obtained,
- and all the choices I made to get it.

Introduction

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, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). This info is given in the “classe” variable, in the training set.

The goal of this project is to build a prediction algorithm that identifies these categories in a independent data set.

Getting the data

```
##Load training data

training <- read.table("./projData/training.csv", sep = ",", header = TRUE)

dim(training)

## [1] 19622   160

##Load testing data

testing <- read.table("./projData/testing.csv", sep = ",", header = TRUE)

dim(testing)

## [1]  20 160
```

Understanding and preparing the Data for the exercise

I initially took a peek at the test data set, to get a feeling of what it is about. The strategy I pursued involved finding the relevant covariates in the testing dataset (no point in considering null variance covariates in the classification algorithm…), as there are lots of variables that don't have any information. For this, I use the *nearZeroVar* formula available in R package caret.

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

nsv_ts <- nearZeroVar(testing, saveMetrics=TRUE)

#nsv_ts
```

The following code picks the relevant covariates for the prediction algorithm.

```
#define relevant covariates

covariates <- rownames(nsv_ts[nsv_ts$zeroVar == 0,])

## covariates

## summary(covariates)
```

In an effort to avoid *over fitting*, I also disregarded some variables not directly related to the Human Activity Recognition exercise, such

as**user_name**, **raw_timestamp_part_1**, **raw_timestamp_part_2**, **cvtd_timestamp**, or **num_window**. I also disregarded counters **X** and**problem_id**.

```
#removes spurious variables from covariates
covariates <- covariates[7:58]
covariates
```

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

The next code chunk applies this set of variables to filter the relevant covariates in the training data set.

```
# add predictive variable to list of variables
covariates[length(covariates)+1] <- "classe"
trainWork <- training[,covariates]
```

Looking out for missing values…there aren't any, so nothing to worry here.

```
NAnumber <- sapply(trainWork, function(x) sum(is.na(x)))
sum(NAnumber)
```

```
## [1] 0
```

In the next step, I plotted a histogram of all variables, to infer about the skewness and normality of covariates (the command line for this is given below, although inactive). I then standardized all covariates (with the *preProcess* formula, also from the caret package), aiming to better fit the data to the purpose at hands.

```
## for(i in 1:dim(trainWork)[2]){hist(as.numeric(trainWork[,i]), main = covariates[i])}
preObj  <- preProcess(trainWork[,-53], method = c("center","scale"))
trainSt <- predict(preObj, trainWork[,-53])
trainSt$classe <- trainWork$classe
## summary(trainSt)
```

Training the model

My first approach to training the model was to perform a default Random Forest algorithm with the *caret* package, Keeping faithful to ***Veloso et al*** approach, who initially justified this methodology due to *"the characteristic noise in the sensor data"*. For that, I divided the training set in 60/40.

```
set.seed(12345)
inTrain = createDataPartition(y = trainSt$classe, p = 0.6, list = FALSE)
trainCDP = trainSt[ inTrain,]
testCDP = trainSt[-inTrain,]
set.seed(123)
modFit1 <- train(classe ~ ., method = "rf", data = trainCDP)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
saveRDS(modFit1, file="modFitRandom1.rds"); ##myVariableName = readRDS("myFile.rds")
```

With this model, I got the following out of sample errors' statistics:

```
pred1 <- predict(modFit1, newdata = testCDP)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
confM1 <- confusionMatrix(pred1,testCDP$classe)
confM1
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228   11    0    0    0
##          B    4 1501    7    0    3
##          C    0    6 1356   18    3
##          D    0    0    5 1266    4
##          E    0    0    0    2 1432
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9897, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9898
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9888   0.9912   0.9844   0.9931
## Specificity            0.9980   0.9978   0.9958   0.9986   0.9997
## Pos Pred Value         0.9951   0.9908   0.9805   0.9929   0.9986
## Neg Pred Value         0.9993   0.9973   0.9981   0.9970   0.9984
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1913   0.1728   0.1614   0.1825
## Detection Prevalence   0.2854   0.1931   0.1763   0.1625   0.1828
## Balanced Accuracy      0.9981   0.9933   0.9935   0.9915   0.9964
```

I got an overall “out of sample” accuracy of 0.9919704, which is indeed a very high figure.

Still, I tried to improve this model, by means of a 10 k-fold repeated cross validation. As you can see below, I was able to improve the overall accuracy of the predicting algorithm, but just marginally (in line with what Jeff talked about in his lectures).

```
fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  ## repeated ten times
  repeats = 10)

modFit2 <- train(classe ~ ., data = trainCDP,
                 method = "rf",
                 trControl = fitControl,
                 verbose = FALSE)

saveRDS(modFit2, file="modFitRandom2.rds")
pred2 <- predict(modFit2, newdata = testCDP)
confM2 <- confusionMatrix(pred2,testCDP$classe)
confM2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   11    0    0    0
##          B    3 1502    7    0    2
##          C    0    5 1358   18    3
##          D    0    0    3 1266    5
##          E    0    0    0    2 1432
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9903, 0.9943)
```

```
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9895   0.9927   0.9844   0.9931
## Specificity            0.9980   0.9981   0.9960   0.9988   0.9997
## Pos Pred Value         0.9951   0.9921   0.9812   0.9937   0.9986
## Neg Pred Value         0.9995   0.9975   0.9985   0.9970   0.9984
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1914   0.1731   0.1614   0.1825
## Detection Prevalence   0.2855   0.1930   0.1764   0.1624   0.1828
## Balanced Accuracy      0.9983   0.9938   0.9943   0.9916   0.9964
```

This last model is used to answer the *Course Project Submission* part. To do so, one must prepare the Test data set so that it can be processed by our algorithm (using the same transformations as those used to train the model). The following table assigns probabilities of each observation belonging to a “classe”" category, according to the classification algorithm and, for each case, puts forward the classe with higher probability.

```
testWork <- testing[,covariates[1:(length(covariates)-1)]]
testSt  <- predict(preObj, testWork)


##run prediction on test data
predSubmit <- predict(modFit2, newdata = testSt, type = "prob")
results <- cbind(predSubmit, classe = LETTERS[apply(predSubmit,1,
function(x) which (x == max(x)))])
results
```

```
##        A     B     C     D     E classe
## 1  0.054 0.766 0.140 0.020 0.020      B
## 2  0.976 0.016 0.004 0.002 0.002      A
## 3  0.110 0.784 0.048 0.008 0.050      B
```

```
## 4   0.924 0.004 0.032 0.038 0.002      A
## 5   0.980 0.008 0.010 0.000 0.002      A
## 6   0.010 0.016 0.048 0.012 0.914      E
## 7   0.016 0.002 0.042 0.918 0.022      D
## 8   0.038 0.792 0.040 0.098 0.032      B
## 9   0.998 0.002 0.000 0.000 0.000      A
## 10 0.990 0.002 0.000 0.008 0.000      A
## 11 0.024 0.790 0.116 0.034 0.036      B
## 12 0.002 0.034 0.904 0.018 0.042      C
## 13 0.002 0.984 0.002 0.000 0.012      B
## 14 1.000 0.000 0.000 0.000 0.000      A
## 15 0.004 0.018 0.014 0.008 0.956      E
## 16 0.008 0.030 0.000 0.006 0.956      E
## 17 0.948 0.000 0.000 0.000 0.052      A
## 18 0.020 0.872 0.018 0.082 0.008      B
## 19 0.150 0.828 0.004 0.018 0.000      B
## 20 0.000 1.000 0.000 0.000 0.000      B
```

Finally, the last code chunk produces the files submitted to Coursera's **Course Project: Submission**.

```
answers <- as.character(results$classe)

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=F
ALSE)
     }
}
 pml_write_files
```