

Practical Machine Learning Peer Graded Assignment

Dhrubasattwata Roy Choudhury

12 June 2018

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 dumbbell 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).

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>

Load required packages for analysis

```
library(caret)
## Loading required package: lattice
## Loading required package: ggplot2
library(rpart)
library(ggplot2)
library(corrplot)
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
library(rattle)
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
set.seed(12345)
```

Load data

```
training_raw <- read.csv("pml-training.csv")[,-1]
```

```

testing <- read.csv("pml-testing.csv")[,-1]
# check dimension of the training and test dataset
dim(training_raw)
dim(testing)
## [1] 19622 159
## [1] 20 159

```

Clean data

```

# remove predictors that have many missing/NA values or non-unique values
NZV <- nearZeroVar(training_raw)
training <- training_raw[, -NZV]
testing <- testing[, -NZV]

# remove cases that have many missing/NA values
NaValues <- sapply(training, function(x) mean(is.na(x))) > 0.9
training <- training[, NaValues == "FALSE"]
testing <- testing[, NaValues == "FALSE"]

# remove id and time variables
training <- training[, -c(1:5)]
testing <- testing[, -c(1:5)]

# check dimension of the cleaned up dataset
dim(training)
## [1] 19622 53
dim(testing)
## [1] 20 53

# take a look at the training dataset
head(training)
## roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x gyros_belt_y
## 1 1.41 8.07 -94.4 3 0.00 0.00
## 2 1.41 8.07 -94.4 3 0.02 0.00
## 3 1.42 8.07 -94.4 3 0.00 0.00
## 4 1.48 8.05 -94.4 3 0.02 0.00
## 5 1.48 8.07 -94.4 3 0.02 0.02
## 6 1.45 8.06 -94.4 3 0.02 0.00
## gyros_belt_z accel_belt_x accel_belt_y accel_belt_z magnet_belt_x
## 1 -0.02 -21 4 22 -3
## 2 -0.02 -22 4 22 -7
## 3 -0.02 -20 5 23 -2
## 4 -0.03 -22 3 21 -6
## 5 -0.02 -21 2 24 -6
## 6 -0.02 -21 4 21 0
## magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm total_accel_arm
## 1 599 -313 -128 22.5 -161 34
## 2 608 -311 -128 22.5 -161 34
## 3 600 -305 -128 22.5 -161 34
## 4 604 -310 -128 22.1 -161 34
## 5 600 -302 -128 22.1 -161 34
## 6 603 -312 -128 22.0 -161 34

```

```

## gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y accel_arm_z
## 1 0.00 0.00 -0.02 -288 109 -123
## 2 0.02 -0.02 -0.02 -290 110 -125
## 3 0.02 -0.02 -0.02 -289 110 -126
## 4 0.02 -0.03 0.02 -289 111 -123
## 5 0.00 -0.03 0.00 -289 111 -123
## 6 0.02 -0.03 0.00 -289 111 -122
## magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell pitch_dumbbell
## 1 -368 337 516 13.05217 -70.49400
## 2 -369 337 513 13.13074 -70.63751
## 3 -368 344 513 12.85075 -70.27812
## 4 -372 344 512 13.43120 -70.39379
## 5 -374 337 506 13.37872 -70.42856
## 6 -369 342 513 13.38246 -70.81759
## yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y
## 1 -84.87394 37 0 -0.02
## 2 -84.71065 37 0 -0.02
## 3 -85.14078 37 0 -0.02
## 4 -84.87363 37 0 -0.02
## 5 -84.85306 37 0 -0.02
## 6 -84.46500 37 0 -0.02
## gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z
## 1 0.00 -234 47 -271
## 2 0.00 -233 47 -269
## 3 0.00 -232 46 -270
## 4 -0.02 -232 48 -269
## 5 0.00 -233 48 -270
## 6 0.00 -234 48 -269
## magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z roll_forearm
## 1 -559 293 -65 28.4
## 2 -555 296 -64 28.3
## 3 -561 298 -63 28.3
## 4 -552 303 -60 28.1
## 5 -554 292 -68 28.0
## 6 -558 294 -66 27.9
## pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 1 -63.9 -153 36 0.03
## 2 -63.9 -153 36 0.02
## 3 -63.9 -152 36 0.03
## 4 -63.9 -152 36 0.02
## 5 -63.9 -152 36 0.02
## 6 -63.9 -152 36 0.02
## gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 1 0.00 -0.02 192 203
## 2 0.00 -0.02 192 203
## 3 -0.02 0.00 196 204
## 4 -0.02 0.00 189 206
## 5 0.00 -0.02 189 206
## 6 -0.02 -0.03 193 203
## accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z
## 1 -215 -17 654 476
## 2 -216 -18 661 473

```

```
## 3      -213      -18      658      469
## 4      -214      -16      658      469
## 5      -214      -17      655      473
## 6      -215      -9       660      478
## classe
## 1      A
## 2      A
## 3      A
## 4      A
## 5      A
## 6      A
```

Prepare data partition, for later validation

```
inTrain <- createDataPartition(y= training$classe, p = 0.7, list = FALSE)
training <- training[inTrain, ]
crossvalidation <- training[!inTrain, ]
```

Now we can train our models given the preprocess with PCA

```
# decision trees
model_tree <- train(classe~., data = training, method = "rpart")
# print result of model prediction on original training and crossvalidation dataset
predict_training_tree <- predict(model_tree, training)
confusionmatrix_training_tree <- confusionMatrix(predict_training_tree, training$classe)

predict_crossvalidation_tree <- predict(model_tree, crossvalidation)
confusionmatrix_cv_tree <- confusionMatrix(predict_crossvalidation_tree, crossvalidation$classe)

print(confusionmatrix_cv_tree)
## Confusion Matrix and Statistics
##
##      Reference
## Prediction  A  B  C  D  E
##      A 1074 319 345 319 100
##      B   11 276  20 128  91
##      C   67 207 354 259 205
##      D    0  0  0  0  0
##      E    9  0  0  0 326
##
## Overall Statistics
##
##      Accuracy : 0.4939
##      95% CI : (0.4785, 0.5093)
##      No Information Rate : 0.2825
##      P-Value [Acc > NIR] : < 2.2e-16
##
##      Kappa : 0.3393
##      Mcnemar's Test P-Value : NA
```

```

##
## Statistics by Class:
##
##          Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9251 0.34414 0.49235 0.0000 0.45152
## Specificity      0.6328 0.92443 0.78237 1.0000 0.99734
## Pos Pred Value   0.4979 0.52471 0.32418   NaN 0.97313
## Neg Pred Value    0.9555 0.85324 0.87906 0.8282 0.89510
## Prevalence       0.2825 0.19513 0.17494 0.1718 0.17567
## Detection Rate    0.2613 0.06715 0.08613 0.0000 0.07932
## Detection Prevalence 0.5248 0.12798 0.26569 0.0000 0.08151
## Balanced Accuracy 0.7789 0.63428 0.63736 0.5000 0.72443
# random forest
model_rf <- train(classe~., data = training, method = "rf")
# print result of model prediction on original training and crossvalidation dataset
predict_training_rf <- predict(model_rf, training)
confusionmatrix_training_rf <- confusionMatrix(predict_training_rf, training$classe)

predict_crossvalidation_rf <- predict(model_rf, crossvalidation)
confusionmatrix_cv_rf <- confusionMatrix(predict_crossvalidation_rf, crossvalidation$classe)

print(confusionmatrix_cv_rf)
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  A  B  C  D  E
##      A 1161  0  0  0  0
##      B  0 802  0  0  0
##      C  0  0 719  0  0
##      D  0  0  0 706  0
##      E  0  0  0  0 722
##
## Overall Statistics
##
##          Accuracy : 1
##          95% CI : (0.9991, 1)
##      No Information Rate : 0.2825
##      P-Value [Acc > NIR] : < 2.2e-16
##
##          Kappa : 1
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: A Class: B Class: C Class: D Class: E
## Sensitivity      1.0000 1.0000 1.0000 1.0000 1.0000
## Specificity      1.0000 1.0000 1.0000 1.0000 1.0000
## Pos Pred Value    1.0000 1.0000 1.0000 1.0000 1.0000
## Neg Pred Value    1.0000 1.0000 1.0000 1.0000 1.0000
## Prevalence        0.2825 0.1951 0.1749 0.1718 0.1757
## Detection Rate    0.2825 0.1951 0.1749 0.1718 0.1757
## Detection Prevalence 0.2825 0.1951 0.1749 0.1718 0.1757

```

## Balanced Accuracy	1.0000	1.0000	1.0000	1.0000	1.0000
----------------------	--------	--------	--------	--------	--------

Conclusion

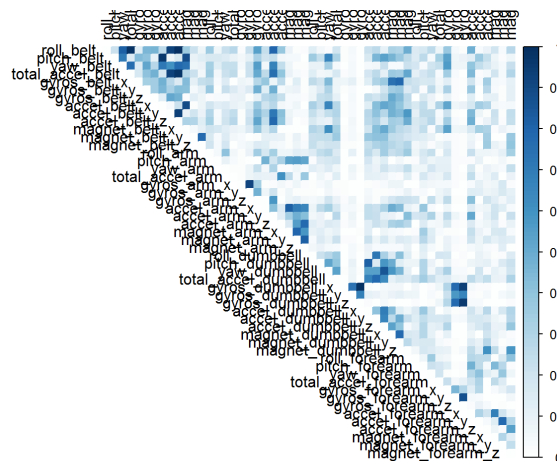
The confusionmatrix showed that the accuracy of the random forest models is better than the decision tree model. Therefore, we used this model to predict on the testing dataset.

Predict on testing dataset

```
predict_testing <- predict(model_rf, testing)
predict_testing
## [1] 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
```

Appendix

```
# explore the remaining predictors
# check the factor variables
predictor_factor <- which(apply(training, class) == "factor")
# explore correlation between predictors
predictor_cor <- abs(cor(training[, -predictor_factor]))
# turn lower tri to 0
predictor_cor[lower.tri(predictor_cor, diag = TRUE)] <- 0
# visualize result
corrplot(predictor_cor, method = "color", type = "upper", cl.lim = c(0, 1), tl.col = rgb(0, 0, 0))
```



```
# find highly correlated predictors
which(predictor_cor > 0.8, arr.ind = TRUE)
##           row col
## roll_belt    1  3
## roll_belt    1  4
## pitch_belt   2  8
## roll_belt    1  9
## total_accel_belt 4  9
```

```
## roll_belt      1 10
## total_accel_belt 4 10
## accel_belt_y   9 10
## pitch_belt     2 11
## accel_belt_x   8 11
## gyros_arm_x    18 19
## accel_arm_x    21 24
## magnet_arm_y   25 26
## gyros_dumbbell_x 31 33
## pitch_dumbbell 28 34
## yaw_dumbbell   29 36
## gyros_dumbbell_x 31 46
## gyros_dumbbell_z 33 46
## gyros_forearm_y 45 46
```

Therefore, there are highly correlated predictors, principal component analysis is necessary.