Weight Lifting Exercises - “How Well” Recognition

Jane Chen

April 12, 2017

```
training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")

install.packages("caret", repo = "https://cran.r-project.org"); library(caret)
```

```
##
## The downloaded binary packages are in
##  /var/folders/zm/j0sqzj8n1zld752bhqfflylc0000gn/T//RtmpJw0GY5/downloaded_packages
```

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

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

```
install.packages("randomForest", repo = "https://cran.r-project.org"); library(randomForest)
```

```
##
## The downloaded binary packages are in
##  /var/folders/zm/j0sqzj8n1zld752bhqfflylc0000gn/T//RtmpJw0GY5/downloaded_packages
```

```
## 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
```

Introduction

This is a Write-Up report for Coursera Practical Machine Learning (from Data Science specialization) Peer-Reviewed Assessment. The aim of this project is to analysis the training dataset and create a prediction model in how well the user has performed the Unilateral Dumbbell Biceps Curl by using the sensors.

Each class (`classe` in the datasets) indicates different classes of “how-well” the participants have performed the bicep curls. - Class A: exactly according to the specification - Class B: throwing the elbows to the front - Class C: lifting the dumbbell only halfway - Class D: lowering the dumbbell only halfway - Class E: throwing the hips to the front

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4eRMLOkyO

Method

```
# Point A: remove the index, username, timestamps and windows
training <- training[, -c(1:7)]
# Point B: remove columns where most observations don't have the result for (NA)
factorCol <- logical(ncol(training))
for (i in 1:(ncol(training)-1)) {
        factorCol[i] <- (class(training[,i ]) == "factor")
}
numTrain <- training
for (i in (1:ncol(training))[factorCol]) {
        numTrain[, i] <- as.numeric(as.character(training[, i]))
}
wanted <- logical(ncol(numTrain))
for (i in 1:ncol(numTrain)) {
        wanted[i] <- (sum(is.na(numTrain[, i])) == 0)
}
newTrain <- numTrain[, wanted]
str(newTrain)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Point C: create train, test and validation sets
set.seed(12345)
inTrain <- createDataPartition(y = newTrain$classe, p = 0.6, list = FALSE)
subTrain <- newTrain[inTrain, ]
subTest <- newTrain[-inTrain, ]
```

The first few columns, including index, uswername, time staps and window, were removed as they were unrelated to the prediction(see point A from the R code above). Variables with mostly `NA`s were also removed as the second step (see point B). The remaining training dataset was then separated into `subTrain` and `subTest` to train the model and predict out of sample error rate (see point C).

Model Selection and Cross Validation

In this analysis, two prediction modelling were used, random forest (RF) and linear discriminant analysis (LDA). RF was chosen to represent the machine learning algorithm for classification, and LDA was chosen to represent the machine learning algorithm for linear combination and pattern recognition. With 5-fold cross validation, the out of sample error rate for RF are much smaller than that of LDA (accuracy are approximately 0.99 and 0.7 respectively, see code and result in appendix). Hence, the model selected here is RF.

The out of sample error is approximately 0.0057354 for the random forest model.

Result

```r
finalMdl <- randomForest(classe ~ ., data = newTrain, method = "class")
prediction <- predict(finalMdl, testing)
```

The predictions for the testing dataset are B, A, B, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B.

Appendix

5-Fold Cross Validation for Random Forest Prediction

```r
mdl_rf1 <- randomForest(classe ~ ., data = fold1_train, method = "class")
pred_rf1 <- predict(mdl_rf1, fold1_test)
confusionMatrix(pred_rf1, fold1_test$classe)$overall[1]
```

```
##  Accuracy
## 0.9872666
```

```
mdl_rf2 <- randomForest(classe ~ ., data = fold2_train, method = "class")
pred_rf2 <- predict(mdl_rf2, fold2_test)
confusionMatrix(pred_rf2, fold2_test$classe)$overall[1]
```

```
##  Accuracy
## 0.9898089
```

```
mdl_rf3 <- randomForest(classe ~ ., data = fold3_train, method = "class")
pred_rf3 <- predict(mdl_rf3, fold3_test)
confusionMatrix(pred_rf3, fold3_test$classe)$overall[1]
```

```
##  Accuracy
## 0.9927813
```

```
mdl_rf4 <- randomForest(classe ~ ., data = fold4_train, method = "class")
pred_rf4 <- predict(mdl_rf4, fold4_test)
confusionMatrix(pred_rf4, fold4_test$classe)$overall[1]
```

```
##  Accuracy
## 0.9919286
```

```
mdl_rf5 <- randomForest(classe ~ ., data = fold5_train, method = "class")
pred_rf5 <- predict(mdl_rf5, fold5_test)
confusionMatrix(pred_rf5, fold5_test$classe)$overall[1]
```

```
##  Accuracy
## 0.9872666
```

5-Fold Cross Validation for Linear Discriminant Analysis Prediction

```
mdl_lda1 <- train(classe ~ ., data = fold1_train, method = "lda")
pred_lda1 <- predict(mdl_lda1, fold1_test)
confusionMatrix(pred_lda1, fold1_test$classe)$overall[1]
```

```
##  Accuracy
## 0.7054329
```

```
mdl_lda2 <- train(classe ~ ., data = fold2_train, method = "lda")
pred_lda2 <- predict(mdl_lda2, fold2_test)
confusionMatrix(pred_lda2, fold2_test$classe)$overall[1]
```

```
##  Accuracy
## 0.7176221
```

```
mdl_lda3 <- train(classe ~ ., data = fold3_train, method = "lda")
pred_lda3 <- predict(mdl_lda3, fold3_test)
confusionMatrix(pred_lda3, fold3_test$classe)$overall[1]
```

```
##  Accuracy
## 0.7044586
```

```
mdl_lda4 <- train(classe ~ ., data = fold4_train, method = "lda")
pred_lda4 <- predict(mdl_lda4, fold4_test)
confusionMatrix(pred_lda4, fold4_test$classe)$overall[1]
```

```
##  Accuracy
## 0.7047579
```

```
mdl_lda5 <- train(classe ~ ., data = fold5_train, method = "lda")
pred_lda5 <- predict(mdl_lda5, fold5_test)
confusionMatrix(pred_lda5, fold5_test$classe)$overall[1]
```

```
##  Accuracy
## 0.6719015
```