Practical Machine Learning Course Project

Brian Altman

January 28, 2016

Background

With devices such as Jawbone Up, Nike FuelBand, and Fitbit it is 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:

- A: according to the specification
- B: elbows to the front
- C: lift dumbbell only halfway
- D: lower dumbbell only halfway
- E: hips to the fron

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

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).

```r
library(data.table)
library(caret)
```

Libraries and packages

```
## Warning: package 'caret' was built under R version 3.2.3

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

```r
#install.packages("randomForest")
#install.packages("e1071")
knitr::opts_chunk$set(cache=TRUE)
```

```r
pml_training <- fread("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
pml_testing  <- fread("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
pml_training <- as.data.frame(pml_training)
```

Load data sets

Data Preparation Remove any columns that have missing values

```
MissingDataColumns <- sapply(pml_training, function (x) any(is.na(x) | x == " " | x == ""))
pml_training_without_missing <- pml_training[,!MissingDataColumns]
```

Remove columns unrelated to sensors of interest

```
TrainingSensorColumns <- grep(pattern = "_belt|_arm|_dumbbell|_forearm", names(pml_training_without_mis
pml_training_cleaned <- pml_training_without_missing[,c(TrainingSensorColumns,60)]
pml_training_cleaned$classe <- as.factor(pml_training_cleaned$classe)
str(pml_training_cleaned)
```

```
## '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 ...
```

Remove values close to zero

```
pml_training_cleaned_pp <-preProcess(pml_training_cleaned,method=c('center', 'scale'))
pml_training_cleaned_pp_pr <- predict(pml_training_cleaned_pp, pml_training_cleaned)
pml_training_cleaned_pp_pr$classe <- pml_training_cleaned$classe
NearZeroValues <- nearZeroVar(pml_training_cleaned_pp_pr,saveMetrics=TRUE)
pml_training_cleaned_pp_pr_z <- pml_training_cleaned_pp_pr[,NearZeroValues$nzv==FALSE]
```

```


pml_training_partition <- createDataPartition(pml_training_cleaned_pp_pr_z$classe, p=0.75)
pml_training_cleaned_Train      <- pml_training_cleaned[ pml_training_partition  [[1]],]
pml_training_cleaned_Validaton  <- pml_training_cleaned[-pml_training_partition  [[1]],]
dim(pml_training_cleaned_Train)
```

Partition the training cleaned data set into training and validation

```
## [1] 14718    53
```

```
dim(pml_training_cleaned_Validaton)
```

```
## [1] 4904   53
```

```


#Sys.time()  #used to monitor progress
#TC1 <- trainControl(method='cv', allowParallel=TRUE, number=5,verboseIter = TRUE)  #used to monitor pr
TC1 <- trainControl(method='cv', allowParallel=TRUE, number=5)
TrainingModel <- train(classe ~., method="rf", data=pml_training_cleaned_Train,trControl=TC1)
```

Using Random forest, train the model using the training partition

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

```
## Warning: package 'randomForest' was built under R version 3.2.3
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
TrainingModel
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11773, 11774, 11775, 11776, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9918467  0.9896860  0.001421237  0.001797916
##   27    0.9908275  0.9883964  0.000721108  0.000912632
##   52    0.9851882  0.9812614  0.001839471  0.002329465
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
#Sys.time() #used to monitor progress
```

```


TrainingPrediction <- predict(TrainingModel, pml_training_cleaned_Train)
confusionMatrix(TrainingPrediction, pml_training_cleaned_Train$classe)
```

Determine accuracy of model

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                 Accuracy : 1
##                   95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
trainingPred <- predict(TrainingModel, pml_training_cleaned_Validaton)
confusionMatrix(trainingPred, pml_training_cleaned_Validaton$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    2    0    0    0
##          B    0  946    4    0    0
##          C    0    1  849    8    4
##          D    0    0    2  796    5
##          E    0    0    0    0  892
## 
## Overall Statistics
## 
##                Accuracy : 0.9947
##                  95% CI : (0.9922, 0.9965)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9930   0.9900   0.9900
## Specificity            0.9994   0.9990   0.9968   0.9983   1.0000
## Pos Pred Value         0.9986   0.9958   0.9849   0.9913   1.0000
## Neg Pred Value         1.0000   0.9992   0.9985   0.9980   0.9978
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1731   0.1623   0.1819
## Detection Prevalence   0.2849   0.1937   0.1758   0.1637   0.1819
## Balanced Accuracy      0.9997   0.9979   0.9949   0.9942   0.9950
```

```
pml_testing_prediction <- predict(TrainingModel, pml_testing)
pml_testing_prediction
```

Based on pml_testing data, create predictions for class answers

```
##  [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
```