Practical Machine Learning Course Project

Human Activity Recognition (HAR)

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 dumbell 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 objective

The goal of the project is to predict the manner in which they did the exercise.We are going to model the ability of giving correct feedback in three aspects that pertain to qualitative activity recognition: 1) the problem of specifying correct execution, 2) the automatic and robust detection of execution mistakes, and 3) how to provide feedback on the quality of execution to the user.

Background

The dataset we are going to analyse, the Weight Lifting Exercise Dataset, has 5 classes, one correct and four incorrect executed exercises.

```
Class A: 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
```

Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Preparing and Downloading Data

```
if (!file.exists("training.csv")) {
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", dest="training.csv
}

if (!file.exists("testing.csv")) {
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", dest="testing.csv")
}

training = read.csv("training.csv",header = TRUE,na.strings=c("NA","#DIV/0!", ""))
testing = read.csv("testing.csv",header = TRUE,na.strings=c("NA","#DIV/0!", ""))

str(training)
```

```
'data.frame':   19622 obs. of  160 variables:
 $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323
 $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484434
 $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
 $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
 $ 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 ...
 $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
 $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
 $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
 $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
 $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
 $ min_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
 $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
 $ amplitude_yaw_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
 $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
 $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
 $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
 $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
 $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
 $ 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 ...
 $ var_accel_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
 $ avg_roll_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
 $ stddev_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ var_roll_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
 $ avg_pitch_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
 $ stddev_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ var_pitch_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
 $ avg_yaw_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ stddev_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
 $ var_yaw_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
 $ 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 ...
 $ kurtosis_roll_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ kurtosis_picth_arm    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ kurtosis_yaw_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ skewness_roll_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ skewness_pitch_arm    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ skewness_yaw_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ max_roll_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
 $ max_picth_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
 $ max_yaw_arm           : int  NA NA NA NA NA NA NA NA NA NA ...
 $ min_roll_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
 $ min_pitch_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
 $ min_yaw_arm           : int  NA NA NA NA NA NA NA NA NA NA ...
 $ amplitude_roll_arm    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ amplitude_pitch_arm   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ amplitude_yaw_arm     : int  NA NA NA NA NA NA NA NA NA NA ...
 $ 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 ...
 $ kurtosis_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
 $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
 $ kurtosis_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
 $ skewness_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
 $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
 $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
 $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
 $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
 $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
 $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
 $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
  [list output truncated]
```

```r
table(training$classe)
```

```
   A    B    C    D    E 
5580 3797 3422 3216 3607 
```

```r
dim(training)
```

```
[1] 19622   160
```

Data Cleaning and Preparation

Step one is to explore the data to look for obvious data errors / data noise. Looking at the initial dataset we notice it has dimentions of 19622 observations by 160 variables. Some features seems to have a lot of NA's and looking at the top 1% (T1), top 5% (T5) and bottom 10% (B10) there seems to be a pattern. The dimensions drop from T1 to T5 but stays the same from T5 to B10 suggesting these exercises were started but stopped shortly after (like a false start) and they all take place in the top 5% of time hence we filter out these exercises by disregarding observations with at least 95% NA's which brings us down from 160 to 93 variables.

```r
training99 <- training[, colSums(is.na(training)) < nrow(training) * 0.99]
training95 <- training[, colSums(is.na(training)) < nrow(training) * 0.95]
training10 <- training[, colSums(is.na(training)) < nrow(training) * 0.1]
dim(training99)
```

```
[1] 19622   154
```

```r
dim(training95)
```

```
[1] 19622    60
```

```r
dim(training10)
```

```
[1] 19622    60
```

Since the exercises are about measuring correctly and incorrectly executed movements we are going to remove variables with little or zero variance:

```r
# install.packages("caret", repos = 'http://cran.rstudio.com')
library(caret)
NZV_Filter <- nearZeroVar(training95,saveMetrics = TRUE)
trainingVar <- training95[,NZV_Filter$nzv == FALSE]
dim(trainingVar)
```

```
[1] 19622    59
```

This brings the dataset down with 34 variable to 59 vairable. Finally we are going to remove variables not directly related to the classification detection.

```r
trainingFinal <- trainingVar[,-c(1:6)]
dim(trainingFinal)
```

```
[1] 19622    53
```

```r
library(ggplot2)
ggplot(trainingFinal,aes(classe)) +
  geom_histogram(binwidth = 1,colour = "blue", fill = "darkgrey") +
  xlab("Classes") +
  ylab ("Frequency (events)") +
  ggtitle("The Fives Classes to predict")
```

Which gives us the dataset we are going to use for the modelling.

Building the Prediction Model:

```r
# install.packages("rpart", repos = 'http://cran.rstudio.com')
library(rpart)  # Recursive partitioning for classification trees
# install.packages("rpart.plot", repos = 'http://cran.rstudio.com')
library(rpart.plot)
# install.packages("caTools", repos = 'http://cran.rstudio.com')
library(caTools)
# install.packages("rattle", repos = 'http://cran.rstudio.com')
library(rattle)
# install.packages("randomForest", repos = 'http://cran.rstudio.com')
library(randomForest)
```

We split the training data with 70% used for training our model and the remainding 30% left for cross validation of the model. For replication purposes we set a random seed at the beginning.

```r
#random seed
set.seed(123)
trainIndex <- createDataPartition(y = trainingFinal$classe, p=0.7,list=FALSE);
trainingPartition <- trainingFinal[trainIndex,];
testingPartition <- trainingFinal[-trainIndex,];
```

Model Predictions and Cross Validation

We are going to test with three different models: A Classification Tree Model, Linear Discriminant Analysis Model and Random Forest Model. We are going to use Accuracy as the deciding parameter. We are training with the trainingPartition data and doing cross validation with the testingPartition data.

Classification Tree Model

```r
ClassTreeModel <- rpart(classe ~ ., data=trainingPartition, method="class")
predict_CTM <- predict(ClassTreeModel, testingPartition, type = "class")
confusionMatrix(testingPartition$classe, predict_CTM)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1424   26   33  167   24
         B  184  667  107  106   75
         C   52   54  701  193   26
         D   56   48   53  735   72
         E   25   70   52  157  778

Overall Statistics

               Accuracy : 0.7315
                 95% CI : (0.72, 0.7428)
    No Information Rate : 0.2958
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6606
 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.8179   0.7711   0.7410   0.5412   0.7979
Specificity            0.9397   0.9060   0.9342   0.9494   0.9381
Pos Pred Value         0.8507   0.5856   0.6832   0.7624   0.7190
Neg Pred Value         0.9247   0.9583   0.9496   0.8734   0.9590
Prevalence             0.2958   0.1470   0.1607   0.2308   0.1657
Detection Rate         0.2420   0.1133   0.1191   0.1249   0.1322
Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
Balanced Accuracy      0.8788   0.8385   0.8376   0.7453   0.8680
```

```r
fancyRpartPlot(ClassTreeModel)
```

The classification Tree Model gave an overall accuracy of 73.15%.

Linear Discriminant Analysis Model

```
LinearDiscriminantAnalysisModel <- train(classe ~ ., data=trainingPartition, method="lda")
predict_LDAM <- predict(LinearDiscriminantAnalysisModel, testingPartition, type = "raw")
confusionMatrix(testingPartition$classe, predict_LDAM)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1374   33  134  128    5
         B  185  721  144   47   42
         C  115  111  678  106   16
         D   49   51  124  694   46
         E   49  185  105  101  642

Overall Statistics

               Accuracy : 0.6982
                 95% CI : (0.6863, 0.7099)
    No Information Rate : 0.3011
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6178
 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.7754   0.6549   0.5722   0.6450   0.8549
Specificity            0.9271   0.9126   0.9260   0.9439   0.9143
Pos Pred Value         0.8208   0.6330   0.6608   0.7199   0.5933
Neg Pred Value         0.9055   0.9199   0.8957   0.9224   0.9773
Prevalence             0.3011   0.1871   0.2014   0.1828   0.1276
Detection Rate         0.2335   0.1225   0.1152   0.1179   0.1091
Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
Balanced Accuracy      0.8512   0.7837   0.7491   0.7944   0.8846
```

The Linear Discriminant Analysis Model gave an overall accuracy of 69.82%.

Random Forest Model

Using the Random Forest Model we get an accuracy of 99.47% which is quite impressive.

```
RandomForestModel <- randomForest(classe ~ ., data = trainingPartition, type="class")
predict_RFM <- predict(RandomForestModel,newdata=testingPartition)
confusionMatrix(testingPartition$classe,predict_RFM)
```

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction    A    B    C    D    E
         A 1673    1    0    0    0
         B    5 1134    0    0    0
         C    0   11 1015    0    0
         D    0    0   13  950    1
         E    0    0    0    0 1082

Overall Statistics

               Accuracy : 0.9947
                 95% CI : (0.9925, 0.9964)
    No Information Rate : 0.2851
    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            0.9970   0.9895   0.9874   1.0000   0.9991
Specificity            0.9998   0.9989   0.9977   0.9972   1.0000
Pos Pred Value         0.9994   0.9956   0.9893   0.9855   1.0000
Neg Pred Value         0.9988   0.9975   0.9973   1.0000   0.9998
Prevalence             0.2851   0.1947   0.1747   0.1614   0.1840
Detection Rate         0.2843   0.1927   0.1725   0.1614   0.1839
Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
Balanced Accuracy      0.9984   0.9942   0.9925   0.9986   0.9995
```

and the random Forest Model is therefore the one we are going to use for predicting the 20 test cases

Apply your machine learning algorithm to the 20 test cases

```
TestCases <- predict(RandomForestModel, newdata=testing, type="class")
print(TestCases)
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 
 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
```

```
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 = FALSE)
    }
}

pml_write_files(TestCases)
```

Acknowledgement:

Datasource: http://groupware.les.inf.puc-rio.br/har. has kindly provided the data for this analysis.