Weight Lifting Exercise Prediction Assignment

Raja Karipineni

November 21, 2015

1. Introduction and Overview

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. For this project, our goal is 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 and data is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We may use any of the other variables to predict with. This report describes how we built this model, how we used cross validation, what we think the expected out of sample error is, and why we made the choices we did. We will also use this prediction model to predict 20 different test cases.

2. Data Set and Credit to Data Authors

The training data for this project is available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data is 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. It was part of paper published by following people:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises (http://groupware.les.inf.puc-rio.br/work.jsf?p1=11201). Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises#ixzz3sC65PSxO

Load the required libraries first.

```
library(caret)
```

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

```
library(ggplot2)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

Downloaded data files before hand from above dataset URLs to local project directory.

```r
# Sample download commands for programatically downloading:
# trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
# download.file(url=train_url, destfile="pml-training.csv")
training <- read.csv('pml-training.csv', na.strings=c("NA","#DIV/0!",""))
testing  <- read.csv('pml-testing.csv' , na.strings=c("NA","#DIV/0!",""))
names(training) # output suppressed in this code chunk with results='hide'
summary(training$classe) # classe is the outcome
```

Split training data set into newTrain and newTest(needed for cross validation strategy) data sets. Keep the original testing data set untouched.

```r
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
newTrain <- training[inTrain,]
newTest  <- training[-inTrain,]
dim(newTrain)
```

```
## [1] 11776   160
```

```r
dim(newTest)
```

```
## [1] 7846  160
```

3. Cleanup of Data

We will remove columns with 70% or more of NA values. In addition remove near zero values and non-predictor columns like timestamp (first seven columns).

```r
len <- length(newTrain)
newTrainSubset1 <- newTrain

for (i in 1:len)
{
   if ( sum(is.na(newTrain[ , i])) / nrow(newTrain) >= .70) #identify columns with 70% or more NAs
   {

      for (j in 1:length(newTrainSubset1))
      {
        if ( length(grep(names(newTrain[i]), names(newTrainSubset1)[j])) == 1)  # see column names are
        {
           newTrainSubset1 <- newTrainSubset1[ , -j]   # remove this jth column
        }
      }
   }
}

dim(newTrainSubset1)
```

```
## [1] 11776    60
```

```r
#Remove first seven columns which includes timestamps ...etc.
newTrainClean <- newTrainSubset1[, -c(1:7)]
dim(newTrainClean)  # it should be seven less columns now
```

```
## [1] 11776    53
```

```r
nearZeroValues <- nearZeroVar(newTrainClean, saveMetrics = TRUE)
nearZeroValues # since all are false, no further cleanup is required.
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.109259    8.67866848   FALSE FALSE
## pitch_belt            1.082645   13.62092391   FALSE FALSE
## yaw_belt              1.024691   14.49558424   FALSE FALSE
## total_accel_belt      1.053944    0.22927989   FALSE FALSE
## gyros_belt_x          1.000000    1.04449728   FALSE FALSE
## gyros_belt_y          1.182595    0.55197011   FALSE FALSE
## gyros_belt_z          1.098299    1.37567935   FALSE FALSE
## accel_belt_x          1.028807    1.32472826   FALSE FALSE
## accel_belt_y          1.074310    1.12941576   FALSE FALSE
## accel_belt_z          1.115830    2.36922554   FALSE FALSE
## magnet_belt_x         1.116279    2.49660326   FALSE FALSE
## magnet_belt_y         1.065163    2.42866848   FALSE FALSE
## magnet_belt_z         1.032491    3.58355978   FALSE FALSE
## roll_arm             47.511628   19.37839674   FALSE FALSE
## pitch_arm            88.869565   22.35054348   FALSE FALSE
## yaw_arm              32.951613   21.44191576   FALSE FALSE
## total_accel_arm       1.018450    0.56046196   FALSE FALSE
## gyros_arm_x           1.039216    5.32438859   FALSE FALSE
## gyros_arm_y           1.591216    3.09952446   FALSE FALSE
## gyros_arm_z           1.116129    1.90217391   FALSE FALSE
## accel_arm_x           1.000000    6.40285326   FALSE FALSE
## accel_arm_y           1.113636    4.39028533   FALSE FALSE
## accel_arm_z           1.177215    6.44531250   FALSE FALSE
## magnet_arm_x          1.018182   11.03091033   FALSE FALSE
## magnet_arm_y          1.060000    7.22656250   FALSE FALSE
## magnet_arm_z          1.043478   10.53838315   FALSE FALSE
## roll_dumbbell         1.075949   87.83967391   FALSE FALSE
## pitch_dumbbell        1.976471   85.67425272   FALSE FALSE
## yaw_dumbbell          1.075949   87.14334239   FALSE FALSE
## total_accel_dumbbell  1.114035    0.35665761   FALSE FALSE
## gyros_dumbbell_x      1.029810    1.92764946   FALSE FALSE
## gyros_dumbbell_y      1.273504    2.19938859   FALSE FALSE
## gyros_dumbbell_z      1.051771    1.66440217   FALSE FALSE
## accel_dumbbell_x      1.000000    3.41372283   FALSE FALSE
## accel_dumbbell_y      1.000000    3.79585598   FALSE FALSE
## accel_dumbbell_z      1.039735    3.35427989   FALSE FALSE
## magnet_dumbbell_x     1.037383    8.81453804   FALSE FALSE
## magnet_dumbbell_y     1.221154    6.86141304   FALSE FALSE
## magnet_dumbbell_z     1.136752    5.57065217   FALSE FALSE
## roll_forearm         12.303665   14.97961957   FALSE FALSE
## pitch_forearm        55.904762   21.06827446   FALSE FALSE
```

```
## yaw_forearm          16.193103   14.29177989   FALSE FALSE
## total_accel_forearm   1.130376    0.56895380   FALSE FALSE
## gyros_forearm_x       1.024768    2.37771739   FALSE FALSE
## gyros_forearm_y       1.012658    6.06317935   FALSE FALSE
## gyros_forearm_z       1.218638    2.40319293   FALSE FALSE
## accel_forearm_x       1.076923    6.52173913   FALSE FALSE
## accel_forearm_y       1.000000    8.18614130   FALSE FALSE
## accel_forearm_z       1.052083    4.65353261   FALSE FALSE
## magnet_forearm_x      1.000000   12.04144022   FALSE FALSE
## magnet_forearm_y      1.236364   15.37024457   FALSE FALSE
## magnet_forearm_z      1.081081   13.30672554   FALSE FALSE
## classe                1.469065    0.04245924   FALSE FALSE
```

4. Random Forest Tree Model

Since Random Forest Tree model is widely used, we are going to use the same model for predictive algorithm. Any efficiency/performance considerations are out of scope for this project.

```
set.seed(1456)
modelFit1 <- randomForest(classe ~ ., data = newTrainClean)
# fancyRpartPlot(modelFit1)  # could not install rattle pkg due to several unsolved errors/dependencies
print(modelFit1)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = newTrainClean)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.73%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3343    4    0    0    1 0.001493429
## B   15 2258    6    0    0 0.009214568
## C    0   17 2035    2    0 0.009250243
## D    0    0   30 1898    2 0.016580311
## E    0    1    1    7 2156 0.004157044
```

Generate Predictions

```
predictions <- predict(modelFit1, newTest, type="class")
```

Let us use ConfusionMatrix on newTest data set (we set aside from training set) to do cross validate using the predictions we just created.

```
confusionMatrix(predictions, newTest$classe)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   15    0    0    0
##          B    1 1501    7    0    0
##          C    0    2 1361   12    0
##          D    0    0    0 1273    6
##          E    0    0    0    1 1436
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9925, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9888   0.9949   0.9899   0.9958
## Specificity            0.9973   0.9987   0.9978   0.9991   0.9998
## Pos Pred Value         0.9933   0.9947   0.9898   0.9953   0.9993
## Neg Pred Value         0.9998   0.9973   0.9989   0.9980   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1913   0.1735   0.1622   0.1830
## Detection Prevalence   0.2863   0.1923   0.1752   0.1630   0.1832
## Balanced Accuracy      0.9984   0.9938   0.9964   0.9945   0.9978
```

As you see above accuracy is close to 99%.

5. Error Rate

Now let us calculate Out Of Sample (OOS) error rate. We expect this error rate to be less than 1% (for brevity) and let us cross-validation data set. Let us take missClass() function given in quiz3 (ques 4) and modify to use it here.

```
missClass = function(values, predictions) {
        sum(predictions != values) / length(values)
        }
oosErr <- missClass(newTest$classe, predictions) # out of sample error
oosErr
```

```
## [1] 0.005607953
```

As calculated (see output) above the out of sample error rate is less than one percent: 0.69% (less than one percent).

6. Generating Final Predictions for Submission to Course web site

Apply our model on given testing data set of 20 observations. Each prediction is written to separate output file.

```
predictionsFinal <- predict(modelFit1, testing, type="class")

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(predictionsFinal)
```

Look for 20 files generated in project directory and submit to Coursera web site for grading.

Results:

```
# Levels: A B C D E (corresponds to one correct way [A] and four incorrect ways of weight lifting).
# Test case#s and predicted values
# 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
```