Course Project: PML Human Activity Recognition

SN Jalis

January 31, 2016

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. 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).

Human Activity Recognition (HAR) is a key research area that is gaining increasing attention, especially for the development of context-aware systems. There are many potential applications for HAR, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises. 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.

Six participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

This report will describe how the data captured are used to identify the parameters involved in predicting the movement involved based on the above classification, and then to predict the movement for 20 test cases.

The training data were divided into two groups, a training data and a validation data (to be used to validate the data), to derived the prediction model by using the training data, to validate the model where an expected out-of-sample error rate of less than 0.5%, or 99.5% accuracy, would be acceptable before it is used to perform the prediction on the 20 test cases - that must have 100% accuracy (to obtain 20 points awarded).

The training model developed using Random Forest was able to achieve over 99.99% accuracy, or less than 0.03% out-of-sample error, and was able to predict the 20 test cases with 100% accuracy.

Getting and cleaning the data

Getting the data from the given URL

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

Cleaning the data

```r
# Remove columns with Near Zero Values
subTrain <-training[, names(training)[!(nzv(training, saveMetrics = T)[, 4])]]

# Remove columns with NA or is empty
subTrain <-subTrain[, names(subTrain)[sapply(subTrain, function (x)
        ! (any(is.na(x) | x == "")))]]


# Remove V1 which seems to be a serial number, and
# cvtd_timestamp that is unlikely to influence the prediction
subTrain <- subTrain[,-1]
subTrain <- subTrain[, c(1:3, 5:58)]
```

Create the prediction model (using random forest)

Using the training data, separate out a set to be used for validation

```
# Divide the training data into a training set and a validation set
inTrain <- createDataPartition(subTrain$classe, p = 0.6, list = FALSE)
subTraining <- subTrain[inTrain,]
subValidation <- subTrain[-inTrain,]
```

Using the first set of data, to create the prediction model (using random forest).

Because of the CPU resources required, first, setup to run it in Parallel, using all the CPU cores available.as advised by Course Forum. Also, even with it running on multiple cores, it will take a long time to create the prediction model. As such, load the model from a previous good run by checking if the model file exists.

```
## Loading objects:
##   fit
```

Measure accuracy and sample error of the prediction model

Using the training subset and create a prediction. Then measure it's accuracy

```
predTrain <- predict(fit, subTraining)
```

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

```
confusionMatrix(predTrain, subTraining$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    1    0    0    0
##          B    0 2275    1    0    0
##          C    0    3 2053    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 0.9996
##                  95% CI : (0.999, 0.9999)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9995   1.0000   1.0000
## Specificity            0.9999   0.9999   0.9997   1.0000   1.0000
## Pos Pred Value         0.9997   0.9996   0.9985   1.0000   1.0000
## Neg Pred Value         1.0000   0.9996   0.9999   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1743   0.1639   0.1838
## Detection Prevalence   0.2844   0.1933   0.1746   0.1639   0.1838
## Balanced Accuracy      0.9999   0.9991   0.9996   1.0000   1.0000
```

From the model, the following are the list of important predictors in the model.

```r
varImp(fit)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 60)
##
##                      Overall
## raw_timestamp_part_1 100.000
## num_window            51.707
## roll_belt             46.707
## pitch_forearm         27.486
## magnet_dumbbell_z     21.299
## magnet_dumbbell_y     17.873
## pitch_belt            16.383
## yaw_belt              16.245
## roll_forearm          15.442
## accel_dumbbell_y       7.936
## roll_dumbbell          7.924
## accel_forearm_x        6.671
## total_accel_dumbbell   6.049
## magnet_dumbbell_x      5.960
## accel_belt_z           5.795
## magnet_belt_y          5.412
## magnet_belt_z          4.713
## accel_dumbbell_z       4.529
## yaw_dumbbell           3.492
## gyros_belt_z           3.348
```

```
fit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 31
##
##         OOB estimate of  error rate: 0.13%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    2 2276    1    0    0 0.0013163668
## C    0    3 2049    2    0 0.0024342746
## D    0    0    3 1926    1 0.0020725389
## E    0    0    0    2 2163 0.0009237875
```

Applying the prediction model

Apply the prediction model to the testing data. The predicted classification are (and were 100% accurate):

```
predTesting <- predict(fit, testing)
predTesting
```

```
##  [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
```

And generate the files for submission with the given R code from the assignment.

```
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(predTesting)
```

Conclusion

The model predicted the 20 test cases with 100% accuracy. All 20 points were awarded after submitting the 20 test files.