Human Activity Recognition Analysis

Rich Robinson

25 July 2015

Introduction

The aim of this project was to take data gathered from 6 subjects performing bicep curls with a dumbbell correctly and incorrectly (in four different manners), giving a total of five possible outcomes, and build a machine learning model that could predict from the sensor data if the subject was performing the exercise correctly or incorrectly (in a certain way). This is extending the world of computer recognition of human actions to now a qualitive feedback about how well the action was performed.

Exploring the Data

The training data provided contained both raw and summary data gathered from the sensors as well as the known outcome of how well the activity was being performed. The test data for use later in predicting the outcome of 20 activities is exactly the same except the outcome variable has been removed.

Some exploratory analysis of the data shows that from the original 160 variables, a large proportion of them have ahigh number of missing values. In order to simplify things for now these variables were removed from the data set, leaving a total of 60 variables. In addition to these the first 7 columns of username, window information and timestamps are not so useful to the anaylsis and can also be removed, leaving a total of 53 variables.

These 53 variables are:

```
names(my_train)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"            
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"        
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"        
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"       
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"           
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"         
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"         
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"        
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"       
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"    
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"    
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"   
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"         
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"     
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"     
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"    
## [52] "magnet_forearm_z"     "classe"
```

As you can see, these are mostly raw telemtary data gathered by the sensors and not summary variables derived from the raw data.

The provided trainging data was split into a training set (60%) and validation set (40%) to provide a basic level of cross validation to assess the out-of-sample error rate before executing the model on the provided test data.

Modelling the Data

Random Forests

This problem is clearly a classification problem with 5 possible outcomes. This naturally suggests a learning model bases on classification trees would be most suitable. Earlier attempts using **Recursive Partitioning and Regression Trees** (`rpart`) didn’t prove very accurate, so **Random Forests** was a choice for improved accuracy but at a dramatic cost to computation time. The `doParallel` package was also used to help improve the execution but taking advantage of multiple processor cores.

```
  # read in the saved model from earlier
  rf_mod_smp <- readRDS("rf_mod_smp.rds")

  ## firstly we need to manipulate the pml_test data so that it is in the sam
e format as the training data
  cols <- names(my_train)
  my_test <- pml_test[, names(pml_test) %in% cols]

  test_ans2 <- predict(rf_mod_smp, newdata = my_test)
```

The machine I use to complete these assignments on in a Laptop which is about 4 years old. It’s a 32bit machine with 3 Gb of RAM. This became a limitation when calculating my model not only in long execution times but also I kept getting strange errors.

- “Unable to allocate memory for vector of size 1 GB”
- “There were missing values in resampled performance measures.”

I was unable to resolve these in time so I resorted to building my model on a sub-sample of 1000 records from my training set. This gave me a model with an **in-sample error rate** of **9.1%**.

```
pred_rf <- predict(rf_mod_smp, newdata = my_valid)

con_mat2 <- confusionMatrix(pred_rf, my_valid$classe)
con_mat2$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2085  102   31   29    2
##          B   62 1294   85   38   37
##          C   34   87 1195   99   14
##          D   37   24   54 1110   28
##          E   14   11    3   10 1361
```

The confusion matrix above shows how well the model performed on the validation data set. considering that I was only able to train my model on 1000 records, the model achieved and accuracy of **89.8%** which is not too bad. Therefore the **out-of-sample error rate** of **10.2%**.

Predicting New Values

The assignment asked us to predict the outcome of 20 new records which were not part of our training data. My predictions for these can be seen below.

```
##  [1] B A B A A E D D A A B B B A E E A B B B
## Levels: A B C D E
```

Possible Improvements

This analysis is by no means complete. To improve this given more time, I would like to have access to a more powerful computer on which to generate my models. Also I would like to implement a more analytical approach to predictor selection, looking at correlations between predictors and maybe use some pre-processing methods such as principle components anaylsis to combine certain variables and therefore reduce the number of predictors.