Machine Learning Assignment 1

Dean Findlay

Friday, November 21, 2014

Is it possible to classify how well an exercise is completed by six participants by analysing data captured from accelerometers on their belt, forearm, arm and dumbell?

Introduction

This report is the written submission to be peer evaluated for the machine learning module as part of the data science specialisation offered by John Hopkins University through Coursera.
The report will follow the structure of loading in the data and performing some exploratory data analysis and preparing the predictor variables, creating the model then assessing the model by predicting against a validation data. Justification on actions taken will be given throughout.

Loading the data

The first step towards fitting the prediction model is to obtain the training data from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv). If the working directory is set to the folder that the downloaded file is in then it can be read using the read.csv command.

In order to run many of the machine learning functions then the desired packages must be loaded into memory.

```
require(caret)
require(randomForest)
```

The seed should also be set (to 1234 in this case) in order for the remaining steps that use random varialbes to be reproducible.

Exploratory data analysis and preparing the predictors

The result of running str() on the training set (Appendix A) show that there are many variables that contain NA or no data.

This is because there is summary data for training sessions which can be found when the ‘new_window’ variable is ‘yes’. This data will not be used in the final model so it will need to be stripped out first by removing the rows where ‘new_window’ = ‘yes’

```
training <- training[training$new_window=="no",]
training[,"new_window"] <- NULL
```

then removing the columns where the data is primarily ‘NA’ or blank

```
list <- as.vector(apply(training, 2, function(x){sum(is.na(x))}))
subset <- list==0
training<- training[,subset]

list <- as.vector(apply(training, 2, function(x){sum(x=="")}))
subset <- list==0
training<- training[,subset]
```

The timestamp data, ‘X’ column and num_window also need removing as they will not be used in the prediction model.

```
training[,"X"] <- NULL
training[,"raw_timestamp_part_1"] <- NULL
training[,"raw_timestamp_part_2"] <- NULL
training[,"cvtd_timestamp"] <- NULL
training[,"num_window"] <- NULL
```

The names can be kept because the test data that is to be used for submission has them, though if the algorithm was to be used with other people not in the initial study they should be left out as they would not provide relevant data.

The training dataset will be subdived into a train and validation set (75/25 split) for the purpose of assessing the models predictve power.
Note, the random forests algortihm uses boosting as a form of cross validation but for the purpose of this study a validation data set will also be used to ensure that cross validation is undertaken.

```
inTrain <- createDataPartition(training$classe, p=.75, list=F)

train <- training[inTrain,]
validation <- training[-inTrain,]
```

Creating the predictive model

To create the predictive model the ‘Random Forest’ algorithm will be used which acts like a decision tree in that it creates branches where each set of decisions leads to a classification, but also uses a technique called bagging which uses bootstrapping to create many trees which are then aggregated into one fianl tree which is used for classification.

The model is fit using the a call to the random forest function with the number of trees to be created set at 2000.

```
modelFit <- randomForest(as.factor(classe)~., data=train, ntree=2000)
```

The summary of the final model can be seen below

```
##
## Call:
##  randomForest(formula = as.factor(classe) ~ ., data = train, ntree = 2000)
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.49%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4101    3    0    0    0    0.000731
## B   11 2772    6    0    0    0.006095
## C    0   15 2497    2    0    0.006762
## D    0    0   23 2335    3    0.011012
## E    0    0    1    6 2639    0.002646
```

This gives an estimated Out Of Bag (OOB) error rate of 0.49% which translates to roughly 95% accuracy.

Testing the predictive model

As well as the accuracy given from the fitted model the out of sample error can be estimated by running the fitted model against the validation data in a confusion matrix and obtaining its accuracy; this is shown below.

predict on validation set

```
confusionMatrix(validation$classe, predict(modelFit,validation))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1367    0    0    0    0
##          B    7  922    0    0    0
##          C    0    3  835    0    0
##          D    0    0   13  772    1
##          E    0    0    0    4  878
##
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.996)
##     No Information Rate : 0.286
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.995    0.997    0.985    0.995    0.999
## Specificity             1.000    0.998    0.999    0.997    0.999
## Pos Pred Value          1.000    0.992    0.996    0.982    0.995
## Neg Pred Value          0.998    0.999    0.997    0.999    1.000
## Prevalence              0.286    0.193    0.177    0.162    0.183
## Detection Rate          0.285    0.192    0.174    0.161    0.183
## Detection Prevalence    0.285    0.193    0.175    0.164    0.184
## Balanced Accuracy       0.997    0.997    0.992    0.996    0.999
```

The output of the confusion matrix shows that this is a very good fit with an estimated out of sample accuracy of 99.4% with 95% confidance intervals only differing by 0.2% percent in either direction.

Appendix

Appendix A

Results from running str() on the training data

```
## '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 132308
4232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 48432
3 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     : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt    : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt      : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt     : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1   : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt      : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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           : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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           : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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     : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1
1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1
1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1
1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1
1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```