Analysis of wearable fitness devices data using machine learning models: (1)random forest model and (2) rpart model

Andy Tai

16 September 2018

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://web.archive.org/web/20161224072740/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 (or activity) in which they did the exercise. These exercises (labelled as “A, B, C, D, E”) are captured in the “classe” variable in the training set, and our intent is to predict the type of exercises in the testing dataset.

Initialisation

clear memory space

```
rm(list=ls()) #free up memory so as to download the data set
# .rs.restartR()
```

check memory limit and allocate more memory to R

```
memory.limit() #[1] 8090
```

```
## [1] 8090
```

```
memory.limit(10*10^10)
```

```
## [1] 1e+11
```

set working directory

```
setwd("C:\\Users\\Andy's Home PC\\Documents\\Coursera Courses\\Data Science\\Practical Machine
 Learning\\Week 4\\Peer Graded Assignment")
```

download training data and read the downloaded file

```
if(!file.exists("./data")){dir.create("./data")} #this will create a folder "data", if it does
n't exists
fileUrl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(fileUrl, destfile="./data/pml-training.csv", method="curl") #this download the fi
le
training <- read.csv("./data/pml-training.csv", sep=",", header=TRUE, na.strings=c("NA", "#DIV/
0!", ""))
```

download testing data and read the downloaded file

```
if(!file.exists("./data")){dir.create("./data")} #this will create a folder "data", if it does
n't exists
fileUrl2 = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(fileUrl2, destfile="./data/pml-testing.csv", method="curl") #this download the fi
le
testing <- read.csv("./data/pml-testing.csv", sep=",", header=TRUE, na.strings=c("NA", "#DIV/
0!", ""))
```

Load necessary packages and set seed

Load the essential packages

```
suppressMessages(library(caret))
suppressMessages(library(randomForest))
suppressMessages(library(rpart))
suppressMessages(library(rpart.plot))
suppressMessages(library(RColorBrewer))
suppressMessages(library(rattle))
suppressMessages(library(ggplot2))
```

set seed to ensure reproducibility

```
set.seed(650542)
```

Exploring the Data

Find size of training dataset & testing dataset

```
dim(training)#[1] 19622   160
```

```
## [1] 19622   160
```

```
dim(testing)#[1]  20 160
```

```
## [1]  20 160
```

We further split our training dataset into 2 smaller data sets to use for training and testing while minimising model over-fitting. Let's use 80:20 for our training:testing sub-data sets accordingly.

```
inTrain <- createDataPartition(training$classe, p=0.8, list=FALSE)
subTraining <- training[inTrain, ]
```

```
subTesting <- training[-inTrain, ]
dim(subTraining)#[1] 15699   160
```

```
## [1] 15699   160
```

```
dim(subTesting)#[1] 3923  160
```

```
## [1] 3923  160
```

Find details of the “classe” variable

```
str(training$classe) #factor variable with 5 levels
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 ...
```

```
table(subTraining$classe)#A:4464; B:3038; C:2738; D:2573; E:2886
```

```
##
##    A    B    C    D    E
## 4464 3038 2738 2573 2886
```

In our subtraining dataset, there are 15699 observation and 160 variables. One of these variables, “classe” column is a factor variable comprising of 5 levels.

Data Pre-Processing

(1)Remove missing values

Let’s check for any missing values in training dataset, and store them in a list

```
na_count<- sapply(1:dim(training)[2], function(x) sum(is.na(training[,x])))
na_list <- which(na_count>0)
```

Let’s remove missing values in subTraining, subTesting, and testing datasets.

```
subTraining<-subTraining[,-na_list]
subTesting<-subTesting[,-na_list]
testing<-testing[,-na_list]
```

(2)Remove irrelevant columns

Let’s print the first 7 columns of the training set

```
head(training[, c(1:7)])
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
## 2 2  carlitos           1323084231               808298 05/12/2011 11:23
## 3 3  carlitos           1323084231               820366 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
```

```
## 5 5  carlitos           1323084232               196328 05/12/2011 11:23
## 6 6  carlitos           1323084232               304277 05/12/2011 11:23
##   new_window num_window
## 1         no         11
## 2         no         11
## 3         no         11
## 4         no         12
## 5         no         12
## 6         no         12
```

Note that these first 7 columns are not relevant, let's remove them in all our datasets (i.e. subTraining, subTesting, and testing).

```
subTraining<-subTraining[,-c(1:7)]
subTesting<-subTesting[,-c(1:7)]
testing<-testing[,-c(1:7)]
```

Processing our data

(1)Configure parallel processing

This step is important as running processes like random forest can take many hours.

```
library(parallel)
library(doParallel)
```

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

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

```
cluster <- makeCluster(detectCores()-1) #convention to leave 1 core for OS
registerDoParallel(cluster)
```

(2)Configure trainControl object

The key parameters for trainControl object are: (1) the resampling method "method"; (2) "number" that specifies the quantity of folds for K-fold cross-validation, and (3) "allowParallel" that tells caret to use the cluster registered in earlier step.

Here, we are using k-fold cross validation ("cv") method, with 5 folds, and allowing for parallel processing.

```
fitControl <- trainControl(method="cv", number=5, allowParallel=TRUE, verboseIter = TRUE)
```

(3)Training our pre-processed data in subTraining dataset, and validating it on subTesting dataset to ascertain its accuracy

(3a)Model 1 ("Random Forest") - using rf

```
#train model using subTraining dataset
model_RandomForest <- train(classe~., data=subTraining, method="rf", trControl=fitControl, tune
Grid=data.frame(mtry=7))
```

```
## Aggregating results
## Fitting final model on full training set
```

```
  #mtry: Number of variables randomly sampled as candidates at each split.

#predict trained model on subTesting dataset
predict_RandomForest <- predict(model_RandomForest, subTesting[,-53])#less off "classe" variabl
e in the 53th column

#calculate confusion matrix
accuracy_RandomForest <- confusionMatrix(predict_RandomForest, subTesting$classe)
print(accuracy_RandomForest)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    2    0    0    0
##          B    1  755    1    0    0
##          C    0    2  682    6    0
##          D    0    0    1  637    1
##          E    0    0    0    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.994, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9947   0.9971   0.9907   0.9986
## Specificity            0.9993   0.9994   0.9975   0.9994   1.0000
## Pos Pred Value         0.9982   0.9974   0.9884   0.9969   1.0000
## Neg Pred Value         0.9996   0.9987   0.9994   0.9982   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1925   0.1738   0.1624   0.1835
## Detection Prevalence   0.2847   0.1930   0.1759   0.1629   0.1835
## Balanced Accuracy      0.9992   0.9970   0.9973   0.9950   0.9993
```

The accuracy from our random forest model is more than 99%.

(3b)Model 2 (“Recursive Partitioning And Regression Trees (rpart)”) - using rpart

```
#train model using subTraining dataset
model_rpart <- train(classe~., data=subTraining, method="rpart", trControl=fitControl)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cp = 0.0357 on full training set
```

```
  #rpart no need to specify mtry

#predict trained model on subTesting dataset
predict_rpart <- predict(model_rpart, subTesting[,-53])#less off "classe" variable in the 53th
 column

#calculate confusion matrix
accuracy_rpart <- confusionMatrix(predict_rpart, subTesting$classe)
print(accuracy_rpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1015  308  314  289  104
##          B   19  257   30  116   98
##          C   79  194  340  238  197
##          D    0    0    0    0    0
##          E    3    0    0    0  322
##
## Overall Statistics
##
##                Accuracy : 0.493
##                  95% CI : (0.4772, 0.5088)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3375
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9095  0.33860  0.49708   0.0000  0.44660
## Specificity            0.6384  0.91688  0.78141   1.0000  0.99906
## Pos Pred Value         0.5000  0.49423  0.32443      NaN  0.99077
## Neg Pred Value         0.9466  0.85248  0.88035   0.8361  0.88911
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2587  0.06551  0.08667   0.0000  0.08208
## Detection Prevalence   0.5175  0.13255  0.26714   0.0000  0.08284
## Balanced Accuracy      0.7740  0.62774  0.63925   0.5000  0.72283
```

Accuracy from the decision tree (using “rpart”) is only about 50%.

Conclusion

Since our random forest model has greater accuracy, we shall apply the model on the testing dataset to predict the activity class.

Predictions on testing dataset

```
  #predict trained model on subTesting dataset
predict_RandomForest_validation <- predict(model_RandomForest, testing)
predict_RandomForest_validation
```

```
##  [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
```

Hence, the predicted activity classes for the 20 rows in the testing dataset are: B A B A A E D B A A B C B A E E A B B B