Predict exercise manner by using machine learning skills

wang

November 22, 2015

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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

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

The test data are 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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Goal

The goal of this project is to predict the manner in which they did the exercise. One key step is to clean the data. I used several criteria to remove data with "NAs" and some insignificant data. A validation data set is also required to find which algorithm is better.

Summary

It turns out that the randomForest algorithm leads to the most accurate predicting results with accuracy about 0.998. Thus, I use the randomForest algorithm to predict the exercise manner for the test data.

Load required packages

```
library(caret); library(rpart); library(rattle); library(randomForest);library(rpart.plot);library(RCol
## Loading required package: lattice
## Loading required package: ggplot2
## Rattle: A free graphical interface for data mining with R.
## Version 4.0.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```r
set.seed(0917) # for reproducible computation
```

Load the data

```r
if(!file.exists("pml-training.csv")){
url.train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
dest.train <- "pml-training.csv"
download.file(url.train, dest.train)
}

data.train <- read.csv("./pml-training.csv")
dim(data.train)
```

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

```r
if(!file.exists("pml-testing.csv")){
url.test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
dest.test <- "pml-testing.csv"
download.file(url.test, dest.test)
}
data.test <- read.csv("./pml-testing.csv")
dim(data.test)
```

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

Clean the data

1, Clean colomns with "NAs"

```r
data.train <- data.train[, colSums(is.na(data.train)) == 0]
data.test <- data.test[, colSums(is.na(data.test)) == 0]
dim(data.train); dim(data.test)
```

```
## [1] 19622    93
```

```
## [1] 20 60
```

2, Remove NearZeroVariance variables

```r
nzv <- nearZeroVar(data.train, saveMetrics=TRUE)
data.train <- data.train[,nzv$nzv==FALSE]

nzv<- nearZeroVar(data.test,saveMetrics=TRUE)
data.test <- data.test[,nzv$nzv==FALSE]
dim(data.train); dim(data.test)
```

```
## [1] 19622    59
```

```
## [1] 20 59
```

3, remove quantities related to “X” and “timestamp”

```r
trainRemove <- grepl("^X|timestamp", names(data.train))
data.train <- data.train[, !trainRemove]

testRemove <- grepl("^X|timestamp", names(data.test))
data.test <- data.test[, !testRemove]
```

```r
name.train <- colnames(data.train)
name.test <- colnames(data.test)

for(i in 1:(length(name.train)-1)){
    if(name.train[i] != name.train[i]) stop("Need more steps to clean the data")
}#Make sure the data is clean enough for use

str(data.train)
```

```
## 'data.frame':    19622 obs. of  55 variables:
##  $ user_name           : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The data is well cleaned up with 54 predictors for both training data and test data. For training data, there are 19622 records while for testing data, there is 20 records.

Split the data into training and validation sub-datasets

```
inTrain <- createDataPartition(y=data.train$classe, p=0.8, list=FALSE)
Training <- data.train[inTrain, ]
Validation <- data.train[-inTrain, ]
dim(Training); dim(Validation)
```

```
## [1] 15699    55
```

```
## [1] 3923   55
```

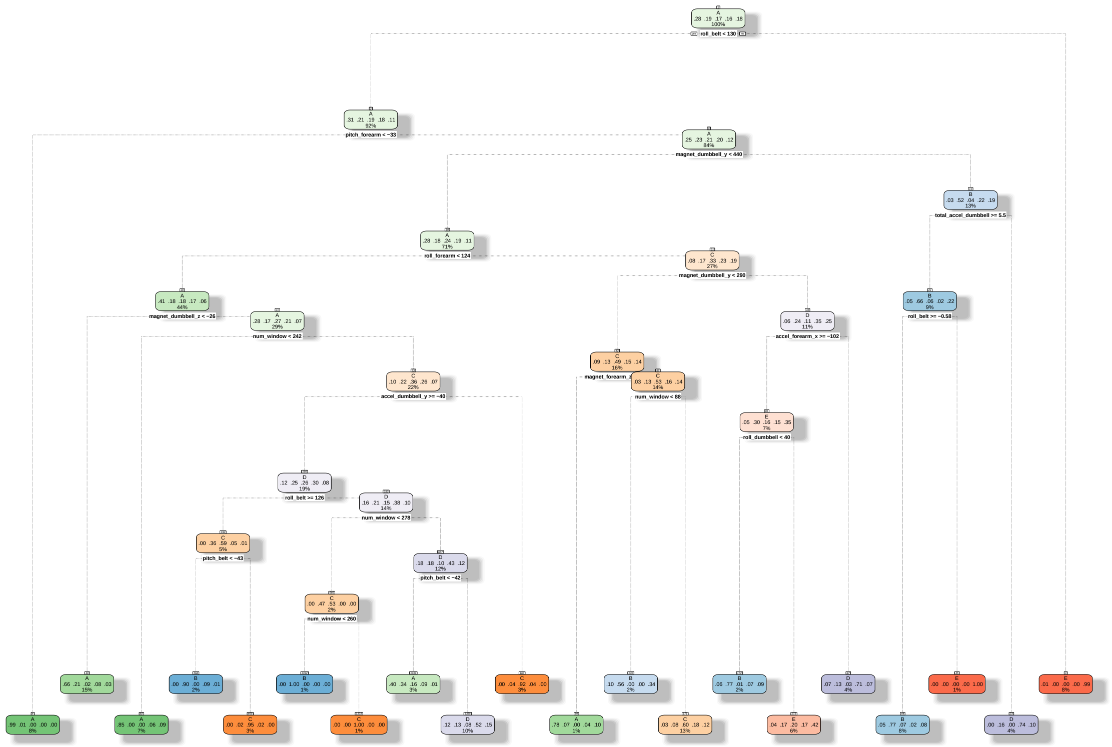
We use 80 percent of data for training and 20 percent for validation.

Prediction by using ML: Decision Tree

```
modfit1 <- rpart(classe ~ ., data=Training, method="class")

fancyRpartPlot(modfit1)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2015−Nov−22 03:31:06 Qingze

```
predict.Tree <- predict(modfit1, Validation,type = "class")
ConFus.Tree <- confusionMatrix(predict.Tree,Validation$classe)
ConFus.Tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1012  174   40   67   55
##          B   23  390   18   10   72
##          C   12   53  538   85   52
##          D   53   91   45  446   93
##          E   16   51   43   35  449
##
## Overall Statistics
##
```

```
##                Accuracy : 0.7227
##                  95% CI : (0.7084, 0.7366)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6471
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9068  0.51383   0.7865   0.6936   0.6227
## Specificity            0.8803  0.96113   0.9376   0.9140   0.9547
## Pos Pred Value         0.7507  0.76023   0.7270   0.6126   0.7559
## Neg Pred Value         0.9596  0.89179   0.9541   0.9383   0.9183
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2580  0.09941   0.1371   0.1137   0.1145
## Detection Prevalence   0.3436  0.13077   0.1886   0.1856   0.1514
## Balanced Accuracy      0.8936  0.73748   0.8621   0.8038   0.7887
```

The accuracy is 0.723.

Prediction by using ML: randomForest

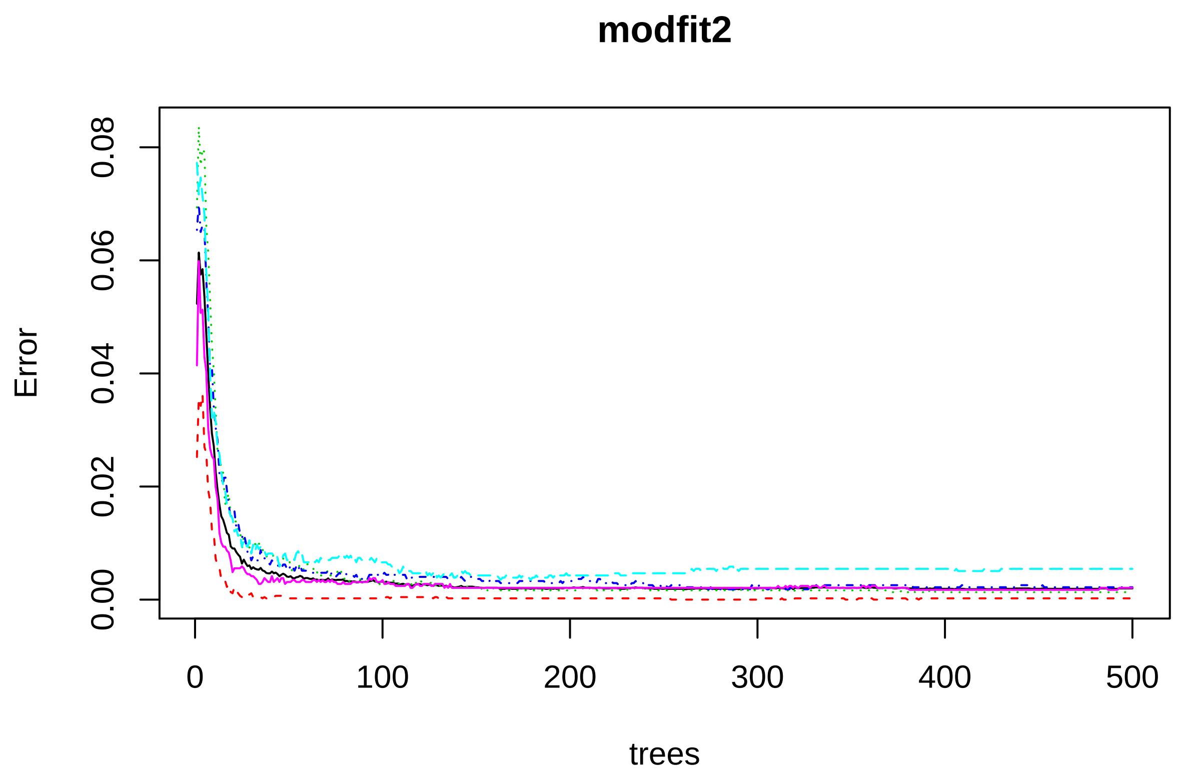
```
modfit2 <- randomForest(classe ~ ., data=Training)

predict.ranFor <- predict(modfit2, Validation, type = "class")
ConFus.ranFor <- confusionMatrix(predict.ranFor,Validation$classe)
ConFus.ranFor
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    1    0    0
##          C    0    0  683    1    0
##          D    0    0    0  642    6
##          E    0    0    0    0  715
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.996, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9985   0.9984   0.9917
## Specificity            1.0000   0.9997   0.9997   0.9982   1.0000
## Pos Pred Value         1.0000   0.9987   0.9985   0.9907   1.0000
## Neg Pred Value         1.0000   1.0000   0.9997   0.9997   0.9981
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1935   0.1741   0.1637   0.1823
## Detection Prevalence   0.2845   0.1937   0.1744   0.1652   0.1823
## Balanced Accuracy      1.0000   0.9998   0.9991   0.9983   0.9958
```

```r
plot(modfit2)
```



The accuracy is, amazingly, 0.998. Thus, the randomForest algorithm leads to a better prediction result on the validation data set. Therefore, we use the randomForest algorithm to predict exercise manner in the test data set.

Predicting on the test data by using randomForest algorithm

```r
predict.Test <- predict(modfit2, data.test, type = "class")
predict.Test
```

```
##  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
## Levels: A B C D E
```