Practical Machine Learning Project

Tuan Bui

1/15/2018

Overview:

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:HAR (see the section on the Weight Lifting Exercise Dataset).

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

```
training<-read.csv("./pml-training.csv")
testing<-read.csv("./pml-testing.csv")
```

Data Preprocessing

First of all, we give the data a first look: is there any missing values?

```
unique(sapply(training,function(i) i%>%is.na%>%sum))
```

```
## [1]     0 19216
```

So there are only two types of columns: no missing values and contains 19216 missing values.

```
table(sapply(training,function(i) i%>%is.na%>%sum))
```

```
##
##     0 19216
##    93    67
```

There are 93 columns with no missing values and 67 columns with 19216 missing values. Is there any coincidence here since those columns all have the same number of missing values? We will look at the `new_window` colums to see what happened

```
table(training$new_window)
```

```
##
##    no   yes
## 19216   406
```

We can roughly see that the number of `no` in `new_window` is equal to the observed number of missing values. And indeed by closer looking at the data, we confirm that all missing values belong to the category `new_window=="no"`.

Now have a look at the testing data with variable `new_window`

```
testing$new_window
```

```
##  [1] no no no no no no no no no no no no no no no no no no no no
## Levels: no
```

They are all `no`. Technically we need to divide the data into two parts and build regression models for each part respectively. But in this project, we will only look at the subset of the data with `new_window="no"`. With this subseted data, we can remove all the colums with missing values without losing any information.

```
new_train<-subset(training,new_window=="no")
```

However, the missing values in the data can be the values `#DIV/0!` or `""`. We remove all the colums with missing values

```
new_train[new_train==""]<-NA;
new_train[new_train=="#DIV/0!"]<-NA;
cleanedTrain <- new_train[, colSums(is.na(new_train)) == 0]
cleanedTest <- testing[, colSums(is.na(testing)) == 0]
dim(cleanedTrain)
```

```
## [1] 19216    60
```

```
dim(cleanedTest)
```

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

After cleaning the missing data, the training data and the testing data contains 60 variables. However, the variables containing the information of users, timestamp and windows don't contribute to the regression then we remove all of these variables.

```
cleanedTrain<-cleanedTrain[,-c(1:7)]
cleanedTest<-cleanedTest[,-c(1:7)]
```

Finally, the data contains 53 variables.

```
inTrain <- createDataPartition(cleanedTrain$classe, p = 0.7, list = FALSE)
train <- cleanedTrain[inTrain, ]
valid <- cleanedTrain[-inTrain, ]

x_predictor<-cleanedTest[,-53]
```

Building models

Classification Trees

```
fit_rpart <- train(classe ~ ., data = train, method = "rpart")
print(fit_rpart, digits = 4)
```

```
## CART
##
## 13453 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
```

```
## Summary of sample sizes: 13453, 13453, 13453, 13453, 13453, 13453, ...
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03824  0.5019    0.35402
##   0.05965  0.3959    0.17367
##   0.11369  0.3456    0.09164
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03824.
```

The accuracy of the algorithm

```r
predict_rpart <- predict(fit_rpart, valid)
confusionMatrix(valid$classe, predict_rpart)$overall[1]
```

```
##  Accuracy
## 0.4992192
```

The accuracy of the classification tree algorithm is too poor. We will investigate the accuracy of Random Forest

Random Forest

```r
fit_rf<-train(classe ~ ., data = train, method = "rf",trControl=trainControl(method = "cv", number = 5))
print(fit_rpart, digits = 4)
```

```
## CART
##
## 13453 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13453, 13453, 13453, 13453, 13453, 13453, ...
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03824  0.5019    0.35402
##   0.05965  0.3959    0.17367
##   0.11369  0.3456    0.09164
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03824.
```

The accuracy of the algorithm

```r
predict_rf <- predict(fit_rf, valid)
confusionMatrix(valid$classe, predict_rf)$overall[1]
```

```
##  Accuracy
## 0.9928856
```

The accuracy of Random Forest is 99.24% which is too high and is the signal of overfitting.

Generalised Boosting Model

Construct the model

```
gbm_fit<-train(classe ~ ., data = train, method = "gbm",trControl=trainControl(method = "repeatedcv", nu
```

```
## Loading required package: gbm

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

## Loading required package: plyr

## ------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1278
##      2        1.5229             nan     0.1000    0.0859
##      3        1.4643             nan     0.1000    0.0671
##      4        1.4204             nan     0.1000    0.0535
##      5        1.3848             nan     0.1000    0.0494
##      6        1.3522             nan     0.1000    0.0409
##      7        1.3258             nan     0.1000    0.0392
##      8        1.3005             nan     0.1000    0.0361
##      9        1.2777             nan     0.1000    0.0304
##     10        1.2574             nan     0.1000    0.0337
##     20        1.0980             nan     0.1000    0.0196
##     40        0.9253             nan     0.1000    0.0087
##     60        0.8193             nan     0.1000    0.0065
##     80        0.7390             nan     0.1000    0.0042
##    100        0.6767             nan     0.1000    0.0032
##    120        0.6239             nan     0.1000    0.0035
##    140        0.5796             nan     0.1000    0.0021
##    150        0.5602             nan     0.1000    0.0019
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1907
##      2        1.4887             nan     0.1000    0.1291
##      3        1.4047             nan     0.1000    0.1052
##      4        1.3374             nan     0.1000    0.0832
##      5        1.2845             nan     0.1000    0.0820
##      6        1.2336             nan     0.1000    0.0533
##      7        1.1972             nan     0.1000    0.0613
##      8        1.1591             nan     0.1000    0.0527
##      9        1.1261             nan     0.1000    0.0500
##     10        1.0947             nan     0.1000    0.0409
##     20        0.8915             nan     0.1000    0.0259
##     40        0.6792             nan     0.1000    0.0131
##     60        0.5493             nan     0.1000    0.0062
##     80        0.4624             nan     0.1000    0.0041
##    100        0.3922             nan     0.1000    0.0042
##    120        0.3402             nan     0.1000    0.0034
##    140        0.2990             nan     0.1000    0.0022
##    150        0.2811             nan     0.1000    0.0024
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2399
##      2        1.4584             nan     0.1000    0.1572
##      3        1.3581             nan     0.1000    0.1314
##      4        1.2757             nan     0.1000    0.1066
##      5        1.2086             nan     0.1000    0.0887
##      6        1.1504             nan     0.1000    0.0780
##      7        1.0986             nan     0.1000    0.0642
##      8        1.0574             nan     0.1000    0.0587
##      9        1.0197             nan     0.1000    0.0621
##     10        0.9825             nan     0.1000    0.0596
##     20        0.7447             nan     0.1000    0.0240
##     40        0.5218             nan     0.1000    0.0066
##     60        0.3976             nan     0.1000    0.0063
##     80        0.3190             nan     0.1000    0.0050
##    100        0.2620             nan     0.1000    0.0044
##    120        0.2176             nan     0.1000    0.0027
##    140        0.1846             nan     0.1000    0.0017
##    150        0.1701             nan     0.1000    0.0014
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1331
##      2        1.5214             nan     0.1000    0.0885
##      3        1.4641             nan     0.1000    0.0686
##      4        1.4204             nan     0.1000    0.0530
##      5        1.3855             nan     0.1000    0.0513
##      6        1.3524             nan     0.1000    0.0393
##      7        1.3261             nan     0.1000    0.0400
##      8        1.3012             nan     0.1000    0.0338
##      9        1.2795             nan     0.1000    0.0316
##     10        1.2592             nan     0.1000    0.0369
##     20        1.0984             nan     0.1000    0.0180
##     40        0.9262             nan     0.1000    0.0102
##     60        0.8152             nan     0.1000    0.0078
```

```
##     80        0.7360             nan     0.1000    0.0038
##    100        0.6752             nan     0.1000    0.0042
##    120        0.6212             nan     0.1000    0.0037
##    140        0.5782             nan     0.1000    0.0031
##    150        0.5588             nan     0.1000    0.0021
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1921
##      2        1.4865             nan     0.1000    0.1328
##      3        1.4001             nan     0.1000    0.1061
##      4        1.3333             nan     0.1000    0.0850
##      5        1.2777             nan     0.1000    0.0753
##      6        1.2289             nan     0.1000    0.0635
##      7        1.1877             nan     0.1000    0.0628
##      8        1.1490             nan     0.1000    0.0477
##      9        1.1167             nan     0.1000    0.0471
##     10        1.0867             nan     0.1000    0.0461
##     20        0.8874             nan     0.1000    0.0242
##     40        0.6733             nan     0.1000    0.0118
##     60        0.5515             nan     0.1000    0.0071
##     80        0.4623             nan     0.1000    0.0054
##    100        0.3951             nan     0.1000    0.0022
##    120        0.3453             nan     0.1000    0.0024
##    140        0.3046             nan     0.1000    0.0022
##    150        0.2862             nan     0.1000    0.0029
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2417
##      2        1.4595             nan     0.1000    0.1646
##      3        1.3551             nan     0.1000    0.1271
##      4        1.2743             nan     0.1000    0.1164
##      5        1.2009             nan     0.1000    0.0849
##      6        1.1470             nan     0.1000    0.0736
##      7        1.0993             nan     0.1000    0.0775
##      8        1.0517             nan     0.1000    0.0544
##      9        1.0170             nan     0.1000    0.0639
##     10        0.9773             nan     0.1000    0.0528
##     20        0.7521             nan     0.1000    0.0205
##     40        0.5273             nan     0.1000    0.0100
##     60        0.4022             nan     0.1000    0.0084
##     80        0.3201             nan     0.1000    0.0039
##    100        0.2611             nan     0.1000    0.0037
##    120        0.2177             nan     0.1000    0.0016
##    140        0.1871             nan     0.1000    0.0024
##    150        0.1718             nan     0.1000    0.0011
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1268
##      2        1.5227             nan     0.1000    0.0884
##      3        1.4646             nan     0.1000    0.0691
##      4        1.4201             nan     0.1000    0.0532
##      5        1.3840             nan     0.1000    0.0477
##      6        1.3525             nan     0.1000    0.0447
##      7        1.3239             nan     0.1000    0.0357
```

```
## 8 1.3009 nan 0.1000 0.0351
## 9 1.2786 nan 0.1000 0.0310
## 10 1.2577 nan 0.1000 0.0290
## 20 1.1041 nan 0.1000 0.0180
## 40 0.9314 nan 0.1000 0.0089
## 60 0.8203 nan 0.1000 0.0053
## 80 0.7403 nan 0.1000 0.0056
## 100 0.6810 nan 0.1000 0.0061
## 120 0.6285 nan 0.1000 0.0037
## 140 0.5834 nan 0.1000 0.0027
## 150 0.5647 nan 0.1000 0.0023
##
## Iter TrainDeviance ValidDeviance StepSize Improve
## 1 1.6094 nan 0.1000 0.1897
## 2 1.4886 nan 0.1000 0.1312
## 3 1.4042 nan 0.1000 0.1036
## 4 1.3377 nan 0.1000 0.0840
## 5 1.2844 nan 0.1000 0.0723
## 6 1.2395 nan 0.1000 0.0672
## 7 1.1963 nan 0.1000 0.0532
## 8 1.1606 nan 0.1000 0.0551
## 9 1.1254 nan 0.1000 0.0466
## 10 1.0959 nan 0.1000 0.0536
## 20 0.8924 nan 0.1000 0.0210
## 40 0.6813 nan 0.1000 0.0114
## 60 0.5495 nan 0.1000 0.0098
## 80 0.4600 nan 0.1000 0.0062
## 100 0.3945 nan 0.1000 0.0032
## 120 0.3452 nan 0.1000 0.0019
## 140 0.3044 nan 0.1000 0.0023
## 150 0.2881 nan 0.1000 0.0024
##
## Iter TrainDeviance ValidDeviance StepSize Improve
## 1 1.6094 nan 0.1000 0.2389
## 2 1.4572 nan 0.1000 0.1610
## 3 1.3551 nan 0.1000 0.1201
## 4 1.2774 nan 0.1000 0.1074
## 5 1.2106 nan 0.1000 0.0873
## 6 1.1551 nan 0.1000 0.0756
## 7 1.1068 nan 0.1000 0.0620
## 8 1.0671 nan 0.1000 0.0598
## 9 1.0288 nan 0.1000 0.0468
## 10 0.9981 nan 0.1000 0.0611
## 20 0.7572 nan 0.1000 0.0286
## 40 0.5283 nan 0.1000 0.0098
## 60 0.4042 nan 0.1000 0.0059
## 80 0.3217 nan 0.1000 0.0041
## 100 0.2645 nan 0.1000 0.0033
## 120 0.2217 nan 0.1000 0.0017
## 140 0.1879 nan 0.1000 0.0014
## 150 0.1726 nan 0.1000 0.0016
##
## Iter TrainDeviance ValidDeviance StepSize Improve
## 1 1.6094 nan 0.1000 0.1270
```

```
##      2        1.5217             nan     0.1000    0.0874
##      3        1.4625             nan     0.1000    0.0701
##      4        1.4176             nan     0.1000    0.0537
##      5        1.3827             nan     0.1000    0.0435
##      6        1.3531             nan     0.1000    0.0432
##      7        1.3248             nan     0.1000    0.0415
##      8        1.2977             nan     0.1000    0.0382
##      9        1.2742             nan     0.1000    0.0313
##     10        1.2533             nan     0.1000    0.0304
##     20        1.0922             nan     0.1000    0.0166
##     40        0.9240             nan     0.1000    0.0106
##     60        0.8150             nan     0.1000    0.0054
##     80        0.7339             nan     0.1000    0.0066
##    100        0.6716             nan     0.1000    0.0041
##    120        0.6207             nan     0.1000    0.0035
##    140        0.5795             nan     0.1000    0.0024
##    150        0.5592             nan     0.1000    0.0028
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1877
##      2        1.4889             nan     0.1000    0.1305
##      3        1.4032             nan     0.1000    0.1075
##      4        1.3342             nan     0.1000    0.0795
##      5        1.2814             nan     0.1000    0.0746
##      6        1.2339             nan     0.1000    0.0658
##      7        1.1926             nan     0.1000    0.0689
##      8        1.1504             nan     0.1000    0.0499
##      9        1.1175             nan     0.1000    0.0442
##     10        1.0900             nan     0.1000    0.0465
##     20        0.8807             nan     0.1000    0.0207
##     40        0.6772             nan     0.1000    0.0139
##     60        0.5447             nan     0.1000    0.0067
##     80        0.4581             nan     0.1000    0.0045
##    100        0.3933             nan     0.1000    0.0037
##    120        0.3405             nan     0.1000    0.0030
##    140        0.2990             nan     0.1000    0.0014
##    150        0.2824             nan     0.1000    0.0012
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2437
##      2        1.4579             nan     0.1000    0.1658
##      3        1.3514             nan     0.1000    0.1298
##      4        1.2704             nan     0.1000    0.1099
##      5        1.2001             nan     0.1000    0.0878
##      6        1.1452             nan     0.1000    0.0672
##      7        1.1019             nan     0.1000    0.0677
##      8        1.0586             nan     0.1000    0.0680
##      9        1.0153             nan     0.1000    0.0587
##     10        0.9780             nan     0.1000    0.0499
##     20        0.7531             nan     0.1000    0.0255
##     40        0.5188             nan     0.1000    0.0131
##     60        0.3993             nan     0.1000    0.0097
##     80        0.3183             nan     0.1000    0.0056
##    100        0.2595             nan     0.1000    0.0015
```

```
##    120        0.2204             nan     0.1000    0.0018
##    140        0.1879             nan     0.1000    0.0017
##    150        0.1737             nan     0.1000    0.0016
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1289
##      2        1.5245             nan     0.1000    0.0866
##      3        1.4676             nan     0.1000    0.0657
##      4        1.4234             nan     0.1000    0.0528
##      5        1.3883             nan     0.1000    0.0523
##      6        1.3540             nan     0.1000    0.0365
##      7        1.3298             nan     0.1000    0.0384
##      8        1.3044             nan     0.1000    0.0386
##      9        1.2796             nan     0.1000    0.0285
##     10        1.2606             nan     0.1000    0.0322
##     20        1.1066             nan     0.1000    0.0173
##     40        0.9326             nan     0.1000    0.0085
##     60        0.8266             nan     0.1000    0.0067
##     80        0.7446             nan     0.1000    0.0044
##    100        0.6827             nan     0.1000    0.0038
##    120        0.6329             nan     0.1000    0.0039
##    140        0.5882             nan     0.1000    0.0022
##    150        0.5673             nan     0.1000    0.0023
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1800
##      2        1.4898             nan     0.1000    0.1300
##      3        1.4057             nan     0.1000    0.1020
##      4        1.3406             nan     0.1000    0.0846
##      5        1.2863             nan     0.1000    0.0737
##      6        1.2392             nan     0.1000    0.0709
##      7        1.1955             nan     0.1000    0.0667
##      8        1.1541             nan     0.1000    0.0510
##      9        1.1217             nan     0.1000    0.0412
##     10        1.0943             nan     0.1000    0.0443
##     20        0.8964             nan     0.1000    0.0193
##     40        0.6804             nan     0.1000    0.0144
##     60        0.5510             nan     0.1000    0.0059
##     80        0.4639             nan     0.1000    0.0069
##    100        0.3979             nan     0.1000    0.0052
##    120        0.3486             nan     0.1000    0.0034
##    140        0.3055             nan     0.1000    0.0026
##    150        0.2882             nan     0.1000    0.0010
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2330
##      2        1.4641             nan     0.1000    0.1676
##      3        1.3603             nan     0.1000    0.1308
##      4        1.2796             nan     0.1000    0.0993
##      5        1.2169             nan     0.1000    0.0862
##      6        1.1615             nan     0.1000    0.0880
##      7        1.1078             nan     0.1000    0.0725
##      8        1.0629             nan     0.1000    0.0605
##      9        1.0243             nan     0.1000    0.0605
```

```
##     10        0.9862             nan     0.1000    0.0470
##     20        0.7544             nan     0.1000    0.0295
##     40        0.5283             nan     0.1000    0.0096
##     60        0.4074             nan     0.1000    0.0058
##     80        0.3248             nan     0.1000    0.0057
##    100        0.2626             nan     0.1000    0.0028
##    120        0.2207             nan     0.1000    0.0027
##    140        0.1865             nan     0.1000    0.0015
##    150        0.1729             nan     0.1000    0.0014
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2416
##      2        1.4591             nan     0.1000    0.1622
##      3        1.3542             nan     0.1000    0.1310
##      4        1.2720             nan     0.1000    0.1032
##      5        1.2077             nan     0.1000    0.0896
##      6        1.1522             nan     0.1000    0.0718
##      7        1.1074             nan     0.1000    0.0620
##      8        1.0674             nan     0.1000    0.0505
##      9        1.0337             nan     0.1000    0.0712
##     10        0.9900             nan     0.1000    0.0506
##     20        0.7598             nan     0.1000    0.0231
##     40        0.5309             nan     0.1000    0.0100
##     60        0.4057             nan     0.1000    0.0077
##     80        0.3232             nan     0.1000    0.0036
##    100        0.2655             nan     0.1000    0.0030
##    120        0.2233             nan     0.1000    0.0017
##    140        0.1886             nan     0.1000    0.0018
##    150        0.1753             nan     0.1000    0.0020
```

The accuracy of the algorithm

```
predict_gbm <- predict(gbm_fit, valid)
confusionMatrix(valid$classe, predict_gbm)$overall[1]
```

```
##  Accuracy
## 0.9562728
```

So we have built three models one with very poor accuracy and one with a sign of overfitting. We decide to use that last one Generalised Boosting Model for the prediction

```
pred<-predict(gbm_fit,x_predictor)
pred
```

```
##  [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
```