Machine Learning Project

Pascal Grabbe

19 Juni 2018

Set working directory and load data:

```
setwd("D://R Coursera/Machine Learning")

fileURL1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
destfile="./traindata.csv"
download.file(fileURL1 , destfile, method="auto")
traindata <- read.csv("traindata.csv", sep = ",", header = TRUE)

fileURL2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
destfile="./testdata1.csv"
download.file(fileURL2 , destfile, method="auto")
testdata <- read.csv("testdata1.csv", sep = ",", header = TRUE)
```

Load needed libraries:

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

Split the training set in train and test set:

```
inTrain  <- createDataPartition(y = traindata$classe, p = 0.75, list = F)
train    <- traindata[inTrain,]
test     <- traindata[-inTrain,]
dim(train); dim(test);dim(testdata)

## [1] 14718   160
```

```
## [1] 4904  160

## [1]  20 160
```

Exclude variables with close to zero variance which are meaningless for out model and check if all three sets have the same column count:

```
NZV      <- nearZeroVar(train)
train    <- train[, -NZV]
test     <- test[, -NZV]
testdata <- testdata[, -NZV]
dim(train);dim(test);dim(testdata)

## [1] 14718   103

## [1] 4904  103

## [1]  20 103
```

Exclude variables with zero values and check again the variable number on all three datasets:

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

## [1] 14718    59

## [1] 4904   59

## [1] 20 59
```

Exclude the first six variables wich have also no meaning for our model, also check the dimensions again:

```
train      = train[, -c(1:6)]
test       = test [, -c(1:6)]
testdata   = testdata [, -c(1:6)]
dim(train);dim(test);dim(testdata)

## [1] 14718    53

## [1] 4904   53

## [1] 20 53
```

First model Random Forest

We do simple cross validation with three resampling iterations and apply our fitted model to the test set:

```
set.seed(365)
controlRF  <- trainControl(method="cv", number=3, verboseIter=FALSE)
```

```
modFitRF   <- train(classe ~ ., data=train, method="rf", trControl=controlRF)
sol        <- predict(modFitRF, newdata = test)
solconf    <- confusionMatrix(sol, test$classe)
solconf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    6    0    0    0
##          B    0  940    6    0    0
##          C    0    3  847   10    0
##          D    0    0    2  794    2
##          E    0    0    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9915, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9905   0.9906   0.9876   0.9978
## Specificity            0.9983   0.9985   0.9968   0.9990   1.0000
## Pos Pred Value         0.9957   0.9937   0.9849   0.9950   1.0000
## Neg Pred Value         1.0000   0.9977   0.9980   0.9976   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1917   0.1727   0.1619   0.1833
## Detection Prevalence   0.2857   0.1929   0.1754   0.1627   0.1833
## Balanced Accuracy      0.9991   0.9945   0.9937   0.9933   0.9989
```

With an accuracy of .9941 our model seems to be quite good, so we fit it to the test set:

```
predictRF  <- predict(modFitRF, testdata)
predictRF
```

```
##  [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
```

Second model Generalized Boosted Model

Just try another model to compare our result. We now use boosting with trees:

```
set.seed(555)
controlGBM <- trainControl(method="repeatedcv", number=3, verboseIter=TRUE)
```

```
PCFit      <- train(classe ~., data = train, method ="gbm",
trControl=controlGBM)

## + Fold1.Rep1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1279
##      2        1.5234             nan     0.1000    0.0847
##      3        1.4666             nan     0.1000    0.0649
##      4        1.4226             nan     0.1000    0.0502
##      5        1.3886             nan     0.1000    0.0463
##      6        1.3582             nan     0.1000    0.0434
##      7        1.3302             nan     0.1000    0.0377
##      8        1.3057             nan     0.1000    0.0373
##      9        1.2814             nan     0.1000    0.0319
##     10        1.2608             nan     0.1000    0.0308
##     20        1.1070             nan     0.1000    0.0160
##     40        0.9341             nan     0.1000    0.0120
##     60        0.8244             nan     0.1000    0.0063
##     80        0.7436             nan     0.1000    0.0061
##    100        0.6796             nan     0.1000    0.0040
##    120        0.6296             nan     0.1000    0.0026
##    140        0.5850             nan     0.1000    0.0018
##    150        0.5644             nan     0.1000    0.0029
##
## - Fold1.Rep1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10,
n.trees=150
## + Fold1.Rep1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1841
##      2        1.4880             nan     0.1000    0.1313
##      3        1.4021             nan     0.1000    0.1008
##      4        1.3362             nan     0.1000    0.0862
##      5        1.2820             nan     0.1000    0.0704
##      6        1.2378             nan     0.1000    0.0695
##      7        1.1943             nan     0.1000    0.0606
##      8        1.1566             nan     0.1000    0.0442
##      9        1.1271             nan     0.1000    0.0365
##     10        1.1024             nan     0.1000    0.0375
##     20        0.8966             nan     0.1000    0.0239
##     40        0.6820             nan     0.1000    0.0116
##     60        0.5574             nan     0.1000    0.0066
##     80        0.4677             nan     0.1000    0.0051
##    100        0.4002             nan     0.1000    0.0040
##    120        0.3458             nan     0.1000    0.0025
##    140        0.3049             nan     0.1000    0.0024
##    150        0.2866             nan     0.1000    0.0015
##
## - Fold1.Rep1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10,
n.trees=150
```

```
## + Fold1.Rep1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2370
##      2        1.4591             nan     0.1000    0.1636
##      3        1.3575             nan     0.1000    0.1240
##      4        1.2778             nan     0.1000    0.1092
##      5        1.2086             nan     0.1000    0.0830
##      6        1.1555             nan     0.1000    0.0689
##      7        1.1105             nan     0.1000    0.0677
##      8        1.0681             nan     0.1000    0.0684
##      9        1.0247             nan     0.1000    0.0608
##     10        0.9859             nan     0.1000    0.0508
##     20        0.7584             nan     0.1000    0.0225
##     40        0.5265             nan     0.1000    0.0104
##     60        0.4055             nan     0.1000    0.0076
##     80        0.3218             nan     0.1000    0.0019
##    100        0.2655             nan     0.1000    0.0024
##    120        0.2234             nan     0.1000    0.0044
##    140        0.1895             nan     0.1000    0.0017
##    150        0.1746             nan     0.1000    0.0015
##
## - Fold1.Rep1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10,
n.trees=150
## + Fold2.Rep1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1271
##      2        1.5228             nan     0.1000    0.0901
##      3        1.4647             nan     0.1000    0.0671
##      4        1.4209             nan     0.1000    0.0529
##      5        1.3856             nan     0.1000    0.0511
##      6        1.3522             nan     0.1000    0.0387
##      7        1.3267             nan     0.1000    0.0366
##      8        1.3017             nan     0.1000    0.0354
##      9        1.2789             nan     0.1000    0.0307
##     10        1.2582             nan     0.1000    0.0314
##     20        1.1055             nan     0.1000    0.0190
##     40        0.9317             nan     0.1000    0.0081
##     60        0.8234             nan     0.1000    0.0058
##     80        0.7413             nan     0.1000    0.0054
##    100        0.6765             nan     0.1000    0.0041
##    120        0.6247             nan     0.1000    0.0035
##    140        0.5806             nan     0.1000    0.0019
##    150        0.5616             nan     0.1000    0.0018
##
## - Fold2.Rep1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10,
n.trees=150
## + Fold2.Rep1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.6094             nan     0.1000    0.1865
##      2        1.4859             nan     0.1000    0.1235
##      3        1.4037             nan     0.1000    0.1039
##      4        1.3362             nan     0.1000    0.0829
##      5        1.2827             nan     0.1000    0.0745
##      6        1.2347             nan     0.1000    0.0613
##      7        1.1957             nan     0.1000    0.0553
##      8        1.1595             nan     0.1000    0.0536
##      9        1.1253             nan     0.1000    0.0509
##     10        1.0939             nan     0.1000    0.0439
##     20        0.8895             nan     0.1000    0.0213
##     40        0.6790             nan     0.1000    0.0131
##     60        0.5505             nan     0.1000    0.0079
##     80        0.4617             nan     0.1000    0.0051
##    100        0.3940             nan     0.1000    0.0035
##    120        0.3429             nan     0.1000    0.0034
##    140        0.2989             nan     0.1000    0.0025
##    150        0.2819             nan     0.1000    0.0023
##
## - Fold2.Rep1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10,
n.trees=150
## + Fold2.Rep1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2372
##      2        1.4593             nan     0.1000    0.1584
##      3        1.3593             nan     0.1000    0.1249
##      4        1.2802             nan     0.1000    0.1013
##      5        1.2144             nan     0.1000    0.0891
##      6        1.1570             nan     0.1000    0.0744
##      7        1.1085             nan     0.1000    0.0730
##      8        1.0612             nan     0.1000    0.0632
##      9        1.0202             nan     0.1000    0.0649
##     10        0.9806             nan     0.1000    0.0455
##     20        0.7562             nan     0.1000    0.0223
##     40        0.5279             nan     0.1000    0.0118
##     60        0.4061             nan     0.1000    0.0073
##     80        0.3213             nan     0.1000    0.0041
##    100        0.2599             nan     0.1000    0.0021
##    120        0.2187             nan     0.1000    0.0010
##    140        0.1863             nan     0.1000    0.0018
##    150        0.1735             nan     0.1000    0.0016
##
## - Fold2.Rep1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10,
n.trees=150
## + Fold3.Rep1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1269
##      2        1.5247             nan     0.1000    0.0901
##      3        1.4658             nan     0.1000    0.0655
```

```
##      4        1.4207             nan     0.1000    0.0530
##      5        1.3864             nan     0.1000    0.0526
##      6        1.3530             nan     0.1000    0.0403
##      7        1.3275             nan     0.1000    0.0412
##      8        1.3014             nan     0.1000    0.0323
##      9        1.2797             nan     0.1000    0.0311
##     10        1.2601             nan     0.1000    0.0283
##     20        1.1022             nan     0.1000    0.0176
##     40        0.9289             nan     0.1000    0.0091
##     60        0.8200             nan     0.1000    0.0044
##     80        0.7404             nan     0.1000    0.0043
##    100        0.6761             nan     0.1000    0.0024
##    120        0.6231             nan     0.1000    0.0033
##    140        0.5814             nan     0.1000    0.0013
##    150        0.5630             nan     0.1000    0.0021
##
## - Fold3.Rep1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10,
n.trees=150
## + Fold3.Rep1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1856
##      2        1.4886             nan     0.1000    0.1232
##      3        1.4061             nan     0.1000    0.1054
##      4        1.3373             nan     0.1000    0.0867
##      5        1.2822             nan     0.1000    0.0717
##      6        1.2356             nan     0.1000    0.0662
##      7        1.1940             nan     0.1000    0.0562
##      8        1.1569             nan     0.1000    0.0579
##      9        1.1221             nan     0.1000    0.0408
##     10        1.0949             nan     0.1000    0.0468
##     20        0.8938             nan     0.1000    0.0267
##     40        0.6766             nan     0.1000    0.0104
##     60        0.5526             nan     0.1000    0.0111
##     80        0.4636             nan     0.1000    0.0059
##    100        0.3981             nan     0.1000    0.0043
##    120        0.3453             nan     0.1000    0.0029
##    140        0.3020             nan     0.1000    0.0024
##    150        0.2829             nan     0.1000    0.0020
##
## - Fold3.Rep1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10,
n.trees=150
## + Fold3.Rep1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10,
n.trees=150
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2321
##      2        1.4623             nan     0.1000    0.1629
##      3        1.3568             nan     0.1000    0.1193
##      4        1.2792             nan     0.1000    0.0976
##      5        1.2164             nan     0.1000    0.0957
##      6        1.1552             nan     0.1000    0.0745
```

```
##      7        1.1078             nan     0.1000    0.0683
##      8        1.0650             nan     0.1000    0.0719
##      9        1.0211             nan     0.1000    0.0595
##     10        0.9846             nan     0.1000    0.0492
##     20        0.7582             nan     0.1000    0.0262
##     40        0.5290             nan     0.1000    0.0155
##     60        0.4046             nan     0.1000    0.0087
##     80        0.3232             nan     0.1000    0.0060
##    100        0.2631             nan     0.1000    0.0036
##    120        0.2168             nan     0.1000    0.0019
##    140        0.1847             nan     0.1000    0.0021
##    150        0.1699             nan     0.1000    0.0007
##
## - Fold3.Rep1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10,
n.trees=150
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1,
n.minobsinnode = 10 on full training set
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2332
##      2        1.4603             nan     0.1000    0.1617
##      3        1.3583             nan     0.1000    0.1248
##      4        1.2801             nan     0.1000    0.1006
##      5        1.2164             nan     0.1000    0.0861
##      6        1.1611             nan     0.1000    0.0722
##      7        1.1153             nan     0.1000    0.0807
##      8        1.0657             nan     0.1000    0.0575
##      9        1.0301             nan     0.1000    0.0551
##     10        0.9964             nan     0.1000    0.0539
##     20        0.7620             nan     0.1000    0.0238
##     40        0.5391             nan     0.1000    0.0092
##     60        0.4063             nan     0.1000    0.0101
##     80        0.3247             nan     0.1000    0.0051
##    100        0.2687             nan     0.1000    0.0024
##    120        0.2249             nan     0.1000    0.0021
##    140        0.1921             nan     0.1000    0.0029
##    150        0.1772             nan     0.1000    0.0006
```

```
sol2       <- predict(PCFit, newdata = test)
sol2conf   <- confusionMatrix(sol2, test$classe)
sol2conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1377   25    0    0    2
##          B   10  898   33    2   11
##          C    7   21  810   22   15
##          D    0    4   12  775   13
##          E    1    1    0    5  860
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9625
##                  95% CI : (0.9568, 0.9676)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9525
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9871   0.9463   0.9474   0.9639   0.9545
## Specificity            0.9923   0.9858   0.9839   0.9929   0.9983
## Pos Pred Value         0.9808   0.9413   0.9257   0.9639   0.9919
## Neg Pred Value         0.9949   0.9871   0.9888   0.9929   0.9898
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2808   0.1831   0.1652   0.1580   0.1754
## Detection Prevalence   0.2863   0.1945   0.1784   0.1639   0.1768
## Balanced Accuracy      0.9897   0.9660   0.9657   0.9784   0.9764
```

With an accuracy of 0.9631 boosting with trees is slightly worse than random forest, although it produces the same predictions:

```
predictGBM <- predict(PCFit, testdata)
predictGBM
```

```
##  [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
```