Human Activity Recognition

Amr Abdelhamed

10/20/2020

view on on rpub: Human Activity Recognition

Executive Summary

In this report,using **Groupware@LES Human Activity Recognition Project** data will build a machine learning modal to classify human activity with **99 %** accuracy using **Random Forest** and **Gradient Boosting** .

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, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell 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:**Groupware@LES Human Activity Recognition Project**.

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(RGtk2)
library(rattle)
library(randomForest)
library(gbm)
library(doParallel)  #use Parallel processing
set.seed(611)
cl <- makeCluster(detectCores() - 1)
```

Data Descriptions

- The **training data** for this project are available here: **pml-training.cs**
- The **test data** are available here: **pml-testing.csv**
- The data for this project come from this source: **Groupware@LES Human Activity Recognition Project**.

by: **Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6. Cited by 2 (Google Scholar)**

Dowenload and loading the data

```
if( !(file.exists('pml-training.csv')) ){
  train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(train_url, 'pml-training.csv')}

if( !(file.exists('pml-testing.csv')) ) {
  test_url  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(test_url, 'pml-testing.csv')}

trainData <- read.csv('pml-training.csv')
testData <- read.csv('pml-testing.csv')

dim(trainData)
```

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

Data cleaning and preprocessing

Identify and **Remove** the columns that has **very small variability** as is not relevant to include them in the model.

```
non_zero_var <- (nearZeroVar(trainData,names = TRUE ,allowParallel = T))

print(paste('There are',length(non_zero_var) ,"cloumnes has near zero variance." ))
```

```
## [1] "There are 60 cloumnes has near zero variance."
```

```
trainData <- trainData[,!(colnames(trainData) %in% non_zero_var)]
testData <- testData[,!(colnames(testData) %in% non_zero_var)]
dim(trainData)
```

```
## [1] 19622   100
```

60 columns removed now the data contain **100 columns**.

Identify and **Remove** the the columns that has more then **80 percent** of the rows **NA values**.

```
NAcol <-(sapply(trainData, function(x) mean(is.na(x))) ) > .80
print(paste('There',sum(NAcol) ,"cloumnes has more then 80 percent of there rows NA values." ))
```

```
## [1] "There 41 cloumnes has more then 80 percent of there rows NA values."
```

```r
trainData <- trainData[,!(NAcol)]
testData <- testData[,!(NAcol)]
dim(trainData)
```

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

41 columns removed now the data contain **59 columns**.

Identify the the columns that has **any `NA` values** to handle the missing values.

```r
colnames(trainData[,(sapply(trainData, function(x) {sum(is.na(x)) }) > 0)])
```

```
## character(0)
```

There is no **any missing values** in the data.

Form documentation of the data there some columns for `user_name` and `time` when the data collected so will remove the those columns.

```r
dnames <- c('X',"user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp")
trainData <- trainData[,!(colnames(trainData) %in% dnames)]
testData <- testData[,!(colnames(testData) %in% dnames)]
trainData$classe <-factor(trainData$classe)
testData$problem_id <-factor(testData$problem_id)
```

After the data **cleaning** will **Split** the data to **train** and **validation** sets.

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

dim(training)
```

```
## [1] 15699    54
```

```r
dim(validation)
```

```
## [1] 3923   54
```

Model fitting

Decision tree fitting using 10 fold cross valuation:

`registerDoParallel(cl)` to start Parallel processing using `CPU` cores cluster created by library **doParallel**.

```r
registerDoParallel(cl)
fitControl <- trainControl(method = "cv",
number = 10,
allowParallel = TRUE,
verbose=FALSE)
ptm <- proc.time()
DT_Model <- train(classe~. ,data=training,method = 'rpart',trControl= fitControl)
DT_time<- proc.time() - ptm
DT_time
```

```
##    user  system elapsed
##    3.14    0.03    6.06
```

```r
DT_Model$finalModel
```

```
## n= 15699
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 15699 11235 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 14374  9923 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1261     7 A (0.99 0.0056 0 0 0) *
##      5) pitch_forearm>=-33.95 13113  9916 A (0.24 0.23 0.21 0.2 0.12)
##       10) num_window>=45.5 12535  9338 A (0.26 0.24 0.22 0.2 0.088)
##         20) magnet_dumbbell_y< 439.5 10673  7542 A (0.29 0.19 0.25 0.19 0.083)
##           40) num_window< 241.5 2557  1042 A (0.59 0.14 0.12 0.12 0.029) *
##           41) num_window>=241.5 8116  5776 C (0.2 0.2 0.29 0.21 0.1)
##             82) magnet_dumbbell_z< -27.5 1808   641 A (0.65 0.2 0.062 0.071 0.017) *
##             83) magnet_dumbbell_z>=-27.5 6308  4081 C (0.071 0.2 0.35 0.25 0.12) *
##         21) magnet_dumbbell_y>=439.5 1862   820 B (0.035 0.56 0.047 0.24 0.12) *
##       11) num_window< 45.5 578   111 E (0 0 0 0.19 0.81) *
##    3) roll_belt>=130.5 1325    13 E (0.0098 0 0 0 0.99) *
```

```r
DT_Model_prediction<- predict(DT_Model, validation)
DT_Model_cm<-confusionMatrix(DT_Model_prediction,validation$classe)
DT_Model_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 990 204  98 103  31
##          B  15 244  20 119  58
##          C 110 311 566 386 207
##          D   0   0   0   0   0
##          E   1   0   0  35 425
##
## Overall Statistics
##
##                Accuracy : 0.5672
##                  95% CI : (0.5515, 0.5827)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4467
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8871   0.3215   0.8275   0.0000   0.5895
## Specificity            0.8447   0.9330   0.6869   1.0000   0.9888
## Pos Pred Value         0.6942   0.5351   0.3582      NaN   0.9219
## Neg Pred Value         0.9495   0.8515   0.9496   0.8361   0.9145
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2524   0.0622   0.1443   0.0000   0.1083
## Detection Prevalence   0.3635   0.1162   0.4028   0.0000   0.1175
## Balanced Accuracy      0.8659   0.6272   0.7572   0.5000   0.7891
```

The **Decision Tree Accuracy** not satisfying but **56% Accuracy** for one tree not bad score so try to fit random forest.

Random Forest fitting using 10 fold cross valuation:

```
fitControl <- trainControl(method = "cv",
number = 10,
allowParallel = TRUE,
verbose=FALSE)
ptm <- proc.time()
RF_Model <- train(classe~. ,data=training,method = 'rf',ntree= 80,trControl= fitControl)
RF_time <-proc.time() - ptm
RF_time
```

```
##    user  system elapsed
##    7.14    0.11   62.95
```

```
RF_Model_prediction<- predict(RF_Model, validation)
RF_Model_cm<-confusionMatrix(RF_Model_prediction, validation$classe)
RF_Model_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    1    0    0    0
##          B    1  758    0    0    0
##          C    0    0  684    2    0
##          D    0    0    0  641    3
##          E    1    0    0    0  718
##
## 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            0.9982   0.9987   1.0000   0.9969   0.9958
## Specificity            0.9996   0.9997   0.9994   0.9991   0.9997
## Pos Pred Value         0.9991   0.9987   0.9971   0.9953   0.9986
## Neg Pred Value         0.9993   0.9997   1.0000   0.9994   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1932   0.1744   0.1634   0.1830
## Detection Prevalence   0.2842   0.1935   0.1749   0.1642   0.1833
## Balanced Accuracy      0.9989   0.9992   0.9997   0.9980   0.9978
```

Random Forest has higher **training time** and also **superior accuracy** then the Decision Tree **99.8%** which is close to **human Accuracy**.

Gradient Boosting fitting using 10 fold cross valuation:

```
fitControl <- trainControl(method = "cv",
number = 10,
allowParallel = FALSE,
verbose=FALSE)
ptm <- proc.time()
GB_Model<- train(classe~., data=training, method ='gbm',trControl= fitControl,verbose=FALSE)
GB_time <-proc.time() - ptm

stopCluster(cl)
```

```
GB_Model_prediction<- predict(GB_Model, validation)
GB_Model_cm<-confusionMatrix(GB_Model_prediction,factor(validation$classe))
GB_Model_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1112    5    0    0    0
##          B    4  747    4    0    1
##          C    0    7  675    7    3
##          D    0    0    5  636   11
##          E    0    0    0    0  706
##
```

```
## Overall Statistics
##
##                Accuracy : 0.988
##                  95% CI : (0.9841, 0.9912)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9848
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9842   0.9868   0.9891   0.9792
## Specificity            0.9982   0.9972   0.9948   0.9951   1.0000
## Pos Pred Value         0.9955   0.9881   0.9754   0.9755   1.0000
## Neg Pred Value         0.9986   0.9962   0.9972   0.9979   0.9953
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2835   0.1904   0.1721   0.1621   0.1800
## Detection Prevalence   0.2847   0.1927   0.1764   0.1662   0.1800
## Balanced Accuracy      0.9973   0.9907   0.9908   0.9921   0.9896
```

The accuracy of **Gradient Boosting** is **99%**,approximately equals **the Random Forest**.

Compere Random Forest and Gradient Boosting scores:

Subtract overall **Gradient Boosting** score from overall **Random Forest** score.

```
round(RF_Model_cm$overall -GB_Model_cm$overall,4)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##         0.0099         0.0126         0.0119         0.0079         0.0000
## AccuracyPValue  McnemarPValue
##         0.0000            NaN
```

```
paste('random forest time ',round(RF_time[3]/60,3),'minute' )
```

```
## [1] "random forest time  1.049 minute"
```

```
paste('Gradient Boosting time ',round(GB_time[3]/60,3), 'minute')
```

```
## [1] "Gradient Boosting time  7.545 minute"
```

The **difference** between two overall score is very small but the **Random Forest has smaller training time**.

Expected out of sample error

```r
paste('Out of sample error is equall',round(1- RF_Model_cm$overall['Accuracy'],digits =5))
```

```
## [1] "Out of sample error is equall 0.00204"
```

The expected **out-of-sample error** is estimated at **`0.002`, or `0.2%`**. The expected **out-of-sample error** is calculated as **`1 - accuracy`** for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be **miss-classified**.

Conclusion

we conclude that, Random Forest is more accurate than Gradient Boosting Model and faster also.

Prediction by Random Forest Model on testing data.

```r
testData$problem_id
```

```
##  [1] 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

```r
predict(RF_Model, testData)
```

```
##  [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
```