Peer assessment ML

Rene int Veld

Sunday, December 11, 2016

Introduction project

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 dumbell of 6 participants in an experiment, and then predict the manner in which they did their exercises. This is the “classe” variable in the training set. We will use any of the other variables to predict with. Underneath we will be describing how we built our model, how we used cross validation, and what is the expected out of sample error is, and why we made the choices we have made. Furthermore, we will use your prediction model to predict 20 different test cases.

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 (A,..,E). More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The HAR (Human Activity Recognition) Dataset contains 5 classes (sitting-down, standing-up, standing, walking, and sitting) collected on 8 hours of activities of 4 healthy subjects. We also established a baseline performance index. Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4RhWds5wi (http://groupware.les.inf.puc-rio.br/har#ixzz4RhWds5wi)

Loading and preprocessing the data

Data: The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (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 (http://groupware.les.inf.puc-rio.br/har). We would like to thank the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), as they have been very generous in allowing their data to be used for this kind of assignment.

We have downloaded the above mentioned 2 files and stored it in local environment for this peer assessment:

```
setwd("~/R/working directory course/predicting activity")

url.train <-
    "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url.test <-
    "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(url.train, "train.csv")
download.file(url.test, "test.csv")
```

Now we have to read the csv files into R-files for further processing.

```
act.train <- read.csv("train.csv")
act.test <- read.csv("test.csv")
```

We know have a training database of 19,662 observations and 160 variables and a testing database of only 20 observations. Note that the observations were made between 28 November and 5 December of 2011, for as well the training as the testing database: which means that we do not make a prediction of future observation (ie. now time series involved).

We noted that all field names are the same, except for the last field. In the training set the last field is called ´classe´, this is the description of how the exercise was performed. In the test set this field is replaced by problem-id, this is the number of the observation of which a prediction has to be made of the method. This can be verified as follows:

```
table(act.train$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
table(act.test$problem_id)
```

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

Now we have to predict the methods of the test set. We noted that many of the 160 columns do not have a valid result (eg. blank or NA or #DIV/0), especially in the test set, so they can be omitted from both sets. These are fields that represent statistical measures: -amplitude -avg -kurtosis -max -min -skewness -stddev -var All these fields starting with above terms have been deleted in the train and test set to create new sets on which we will make our prediction. Furthermore we have to eliminate all variables that are not fit to predict which start with: - cvt(d_timestamp, is already presented in the raw data) - new(_window, almost always the same value) And last but not least the record number X has to be removed (highly correlated with classe!).

```
x <- names(act.train)
fields.to.select <- subset(x, ! substr(x[],1,3) %in% c("amp","avg","kur","ma
x","min","ske","std","var","cvt","new","num","raw"))
train <- subset(act.train, select = fields.to.select)
train$X <- NULL
```

This way we only analyse the 53 relevant fields for our prediction. Note that there is only one factor variable: the classe A-E, which we have to predict.

Choosing a prediction method

We can now choose a prediction method to predict the methods in the test set. To be sure our analysis runs smoothly, we nee several libraries, such as the caret library.

```
library(caret)
```

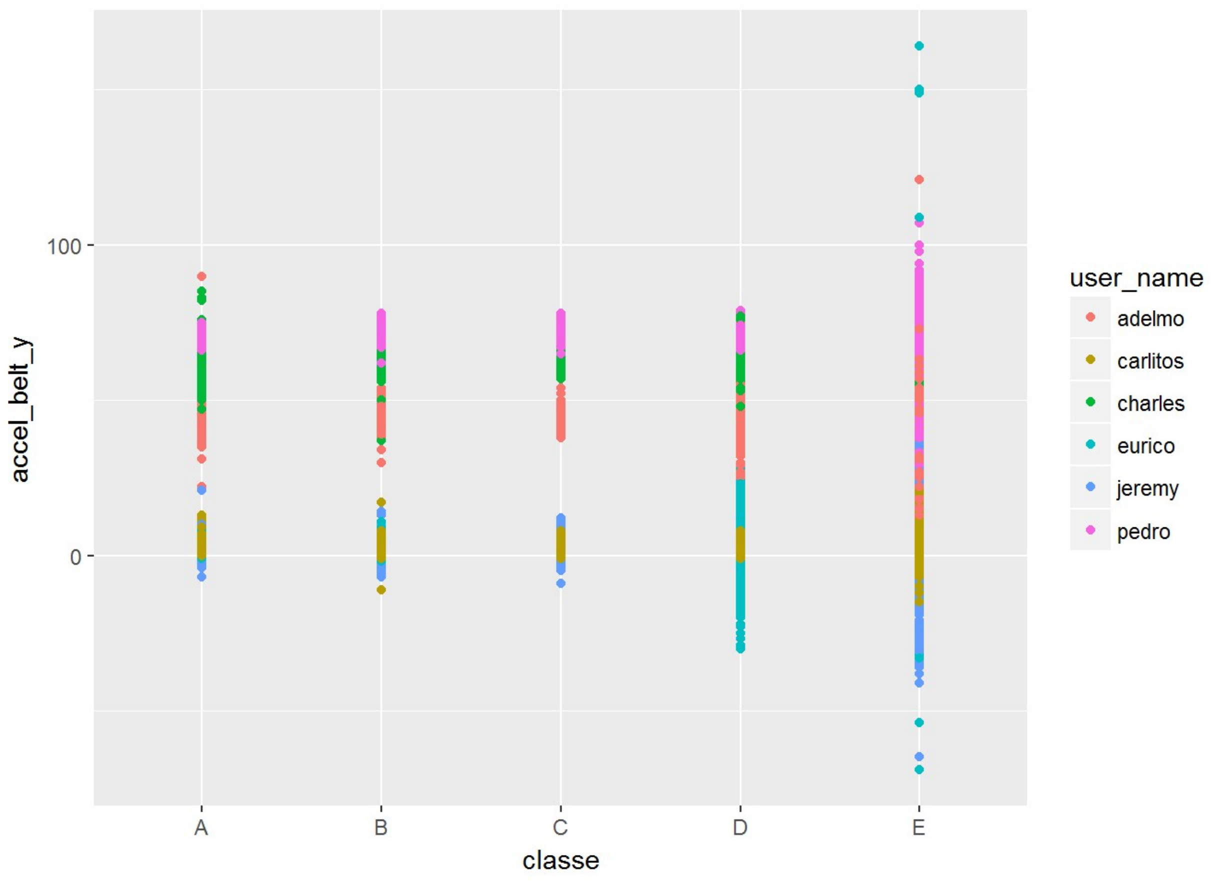
```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)
library(ggplot2)
library(MASS)
```

First we will make a picture of the data, based on 2 random variables.

```
qplot(classe, accel_belt_y, color=user_name, data=train)
```



The picture shows that there is no clear lineair indication of how to predict. The only clear thing we see is that classe E can be found in high but also in low values of the random chosen variable. And that this caused by Jeremy.

Now we will do some Cross Validation. For this we will split the train-set in a training and a testing set (the latter is not the test set on which we will make our final predictions). We will build the model on the training set and then evaluate the model based on the testing set.

```
set.seed(32366)

inTrain <- createDataPartition(y=train$classe,
                               p=0.9, list=FALSE)

training <- train[inTrain,]
testing <- train[-inTrain,]
```

```
t1 <- Sys.time()
modelFit <- train(classe~ .,method="rpart",data=training)
t2 <- Sys.time()
t2-t1
```

```
## Time difference of 19.62505 secs
```

```
modelFit
```

```
## CART
##
## 17662 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 17662, 17662, 17662, 17662, 17662, 17662, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03306962  0.5101916  0.35879060
##   0.05981013  0.4272733  0.22790821
##   0.11479430  0.3260207  0.06383764
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03306962.
```

```
pred <- predict(modelFit, newdata=testing)
confusionMatrix(pred, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 514 155 157 140  43
##          B   9 128  16  62  42
##          C  33  96 169 119 107
##          D   0   0   0   0   0
##          E   2   0   0   0 168
##
## Overall Statistics
##
##                Accuracy : 0.4995
##                  95% CI : (0.4771, 0.5219)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3462
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9211  0.33773  0.49415   0.0000  0.46667
## Specificity            0.6469  0.91841  0.78059   1.0000  0.99875
## Pos Pred Value         0.5094  0.49805  0.32252      NaN  0.98824
## Neg Pred Value         0.9537  0.85261  0.87953   0.8362  0.89274
## Prevalence             0.2847  0.19337  0.17449   0.1638  0.18367
## Detection Rate         0.2622  0.06531  0.08622   0.0000  0.08571
## Detection Prevalence   0.5148  0.13112  0.26735   0.0000  0.08673
## Balanced Accuracy      0.7840  0.62807  0.63737   0.5000  0.73271
```

This prediction is quite fast (18 secs) but not very accurate (52%). From the confusion Matrix you can see that only if the predicted value is E, this probably is true, but that also other proedictions occur when the real value is E.

This means if we would use the rpart method, we would predic only about 50% correct. So we need to improve.

We might use the same prediction, but now with pre-processing.

```
t1 <- Sys.time()
modelFit <- train(classe~ .,method="rpart", preProcess="pca", data=training)
t2 <- Sys.time()
t2-t1
```

```
## Time difference of 31.62546 secs
```

```
modelFit
```

```
## CART
##
## 17662 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: principal component signal extraction (57), centered
##  (57), scaled (57)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 17662, 17662, 17662, 17662, 17662, 17662, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03306962  0.3489992  0.1091431
##   0.05981013  0.3369433  0.0851220
##   0.11479430  0.2837673  0.0000000
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03306962.
```

The clearly did not help us further: it takes more time and the accuracy is worse. We now use the lda-model.

```
t1 <- Sys.time()
modelFit <- train(classe~ .,method="lda", data=training)
t2 <- Sys.time()
t2-t1
```

```
## Time difference of 11.7422 secs
```

```
modelFit
```

```
## Linear Discriminant Analysis
##
## 17662 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 17662, 17662, 17662, 17662, 17662, 17662, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.731603  0.6600504
##
##
```

```
pred <- predict(modelFit, newdata=testing)
confusionMatrix(pred,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 479  53  40  19  11
##          B  15 252  44   8  51
##          C  35  62 219  34  29
##          D  29   5  36 256  34
##          E   0   7   3   4 235
##
## Overall Statistics
##
##                Accuracy : 0.7352
##                  95% CI : (0.7151, 0.7546)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6643
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8584   0.6649   0.6404   0.7975   0.6528
## Specificity            0.9123   0.9254   0.9011   0.9365   0.9912
## Pos Pred Value         0.7957   0.6811   0.5778   0.7111   0.9438
## Neg Pred Value         0.9418   0.9201   0.9222   0.9594   0.9269
## Prevalence             0.2847   0.1934   0.1745   0.1638   0.1837
## Detection Rate         0.2444   0.1286   0.1117   0.1306   0.1199
## Detection Prevalence   0.3071   0.1888   0.1934   0.1837   0.1270
## Balanced Accuracy      0.8853   0.7951   0.7707   0.8670   0.8220
```

This is pretty good, the model takes only 12 seconds and the accuracy is 73%. Unfortunately this is again not good enough, for passing the test we need at least 80%. So now we will try random forest, which can take a long time, as appeared from some preliminary work.

```
t1 <- Sys.time()
modelFit <- train(classe~ .,method="rf", data=training)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
t2 <- Sys.time()
t2-t1
```

```
## Time difference of 1.886837 hours
```

```
modelFit
```

```
## Random Forest
##
## 17662 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 17662, 17662, 17662, 17662, 17662, 17662, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9909637  0.9885707
##   29    0.9920900  0.9899954
##   57    0.9841025  0.9798924
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 29.
```

```
pred <- predict(modelFit, newdata=testing)
confusionMatrix(pred,testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   A   B   C   D   E
##          A 558   2   0   0   0
##          B   0 375   1   0   1
##          C   0   2 340   2   1
##          D   0   0   1 319   2
##          E   0   0   0   0 356
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9939          
##                  95% CI : (0.9893, 0.9968)
##     No Information Rate : 0.2847          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9923          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9894   0.9942   0.9938   0.9889
## Specificity            0.9986   0.9987   0.9969   0.9982   1.0000
## Pos Pred Value         0.9964   0.9947   0.9855   0.9907   1.0000
## Neg Pred Value         1.0000   0.9975   0.9988   0.9988   0.9975
## Prevalence             0.2847   0.1934   0.1745   0.1638   0.1837
## Detection Rate         0.2847   0.1913   0.1735   0.1628   0.1816
## Detection Prevalence   0.2857   0.1923   0.1760   0.1643   0.1816
## Balanced Accuracy      0.9993   0.9941   0.9955   0.9960   0.9944
```

The random forest model without pre-processing scores 99,1% accuracy. It took also quite a time to process: 1.5 hours. NB. The glm-model does not work (too many errors). The gbm-model is very slow, might not work, i have stopped it. The mda-model gives many errors, with pre-processing it scores only 55% accuracy. Based on this we choose the rf-model without pre-processing.

Final prediction

We can do now use the prediction model to predict the final test set. A slightly better result could be obtained by running the rf-method again on the whole set, but because of the time it takes to re-run and the expected improvement (not too much), we did not perform this.

```
# modelFit2 <- train(classe ~ ., method="rf", data=train)
```

Now we have to clean the final test set just as the training set. With the rf-model we have answered the Quiz, and the result was as expected.

```
x <- names(act.test)
fields.to.select <- subset(x, ! substr(x[],1,3) %in% c("amp","avg","kur","ma
x","min","ske","std","var","cvt","new","num","raw"))
test <- subset(act.test, select = fields.to.select)
test$X <- NULL
pred <- predict(modelFit, newdata=test)
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
```

Conclusion

Because of the number of the variables and the spread in the results it is quite difficult to predict which prediction method works best. After we first had a reasonable result of 73% with the Linear Discriminant Analysis we could hugely improve with the random forest method, although this is around 1000x slower. So a clear pay off between speed and accuracy.