course project machine learning

sandro

15/9/2017

Executive Summary

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.

The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and to predict the manner in which they did the exercise.

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

Data

Download data

The data for this project come from this source: http://web.archive.org/web/20170809020213/http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf (http://web.archive.org/web/20170809020213/http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf)

```
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
ing.csv", destfile = "pml.training.csv", method = "curl")

download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testi
ng.csv", destfile = "pml.testing.csv", method = "curl")
```

```
pml.training<-read.csv("pml.training.csv", na.strings=c("NA","#DIV/0!",""))
pml.testing<-read.csv("pml.testing.csv", na.strings=c("NA","#DIV/0!",""))
```

Exploring data

This is a dataset of 160 varibales and the last variable is the actual class.
Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

```
#exploring dimensons of the two dataset
dim(pml.training)
```

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

```
dim(pml.testing)
```

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

```
#four sensor sources of measures:
belt = grep(pattern = "_belt", names(pml.training))
length(belt)
```

```
## [1] 38
```

```
range(belt)
```

```
## [1]  8 45
```

```
arm=grep(pattern = "_arm", names(pml.training))
length(arm)
```

```
## [1] 38
```

```
range(arm)
```

```
## [1] 46 83
```

```
dumbbell=grep(pattern = "_dumbbell", names(pml.training))
length(dumbbell)
```

```
## [1] 38
```

```
range(dumbbell)
```

```
## [1]  84 121
```

```
forearm=grep(pattern = "_forearm", names(pml.training))
length(forearm)
```

```
## [1] 38
```

```
range(forearm)
```

```
## [1] 122 159
```

Cleaning data

Training data

```
#selecting the columns containing sensor measures or classe variables
var<-grep(pattern = "_belt|_arm|_dumbbell|_forearm|classe", names(pml.training))

#variables not containing sensor measures
names(pml.training[,-var])
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

```
#eliminating columns that not contains sensor measures or the variable response "c
lasse"
new.pml.training<-pml.training[, var]
dim(new.pml.training)
```

```
## [1] 19622   153
```

```
#there are some variables with 100% NA values and only 53 variables have no NA
NAvector<-colSums(is.na(new.pml.training))/19622
range(NAvector)
```

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

```
v<-which(NAvector==0)
length(v)
```

```
## [1] 53
```

```
data.training<-new.pml.training[,v]
table(complete.cases(data.training))
```

```
##
##  TRUE
## 19622
```

Testing data

```
#selecting the columns containing sensor measures or problem_id variables
var.test<-grep(pattern = "_belt|_arm|_dumbbell|_forearm|problem_id", names(pml.tes
ting))


#eliminating columns that not contains sensor measures or the variable response "p
roble_id"
new.pml.testing<-pml.testing[, var.test]

#select the same variables of training data
data.testing<-new.pml.testing[,v]

#data.testing has 20 predictors that are the same of the set data.training
dim(data.testing)
```

```
## [1] 20 53
```

```
#column names are the same in both sets (except the last column)
trainingNames <- colnames(data.training)
testingNames <- colnames(data.testing)
all.equal(trainingNames[1:length(trainingNames)-1], testingNames[1:length(testingN
ames)-1])
```

```
## [1] TRUE
```

Splitting data.training

```
set.seed(1)
library(caret)
```

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

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

```
attach(data.training)
inTrain<-createDataPartition(y=classe, p=0.70, list = FALSE)

training<-data.training[inTrain,]
testing<-data.training[-inTrain,]
```

Elaborating data and prediction

Training three models

```
library(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
```

```
library(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
```

```
#bagging: we use 52 predictors, 500 n.trees
bag.model<-randomForest(classe~., data=training, mtry=52, importance=TRUE)
bag.model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, mtry = 52,      importance
= TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 52
##
##         OOB estimate of  error rate: 1.4%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3885   11    7    1    2 0.005376344
## B   35 2598   18    4    3 0.022573363
## C    7   17 2352   19    1 0.018363940
## D    1    4   35 2208    4 0.019538188
## E    0    6    8    9 2502 0.009108911
```

```
#randomForest: we use m=7 about sqrt(52) predictors, 500 n.trees
rf.model<-randomForest(classe~., data=training, importance=TRUE)
rf.model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.53%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    5    1    0    1 0.001792115
## B   12 2642    4    0    0 0.006019564
## C    0   13 2379    4    0 0.007095159
## D    0    0   24 2226    2 0.011545293
## E    0    0    1    6 2518 0.002772277
```

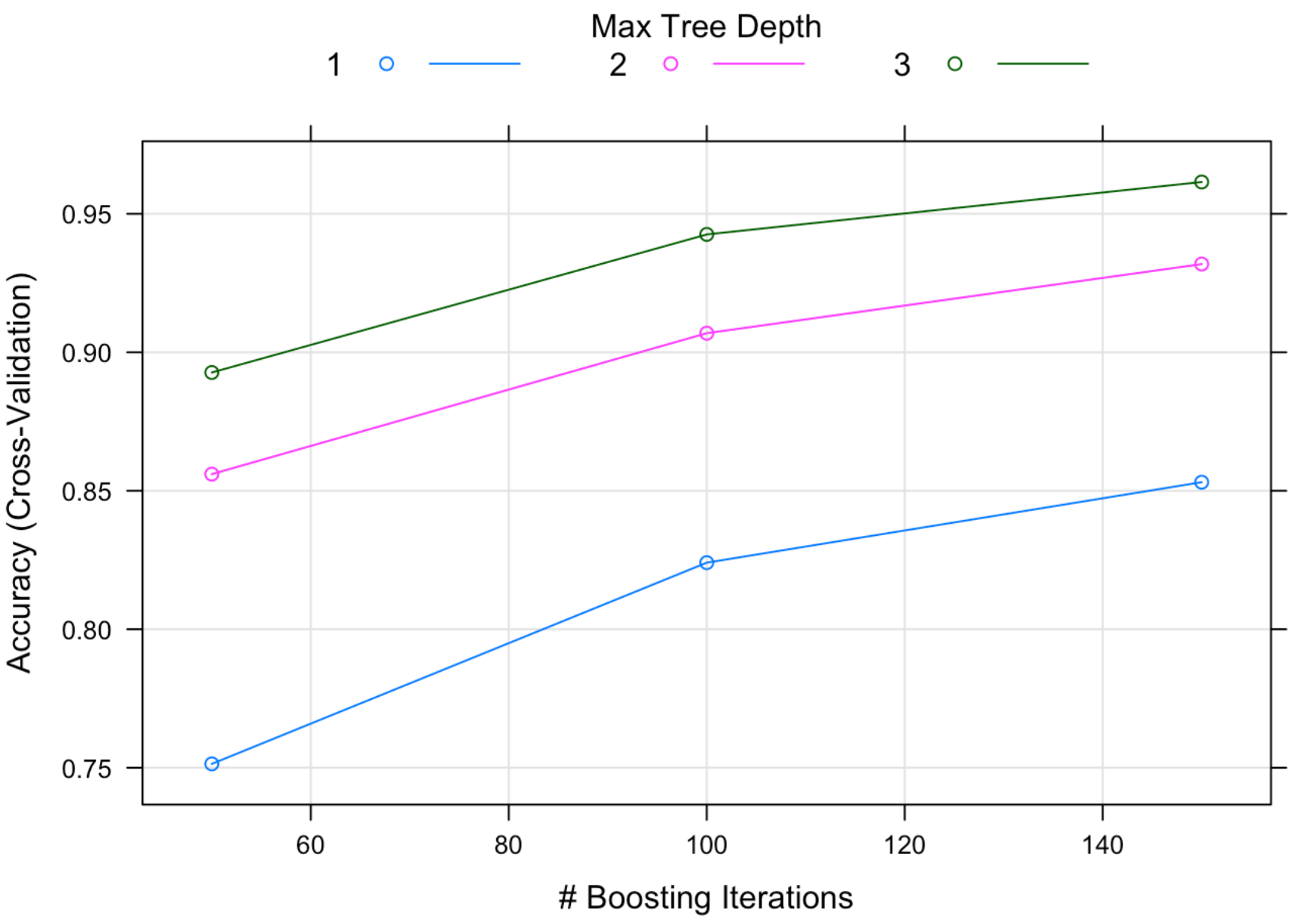
```
#boosting: we use cross validation with k=10 folds, 150 n.trees, interaction.depth
=3, shrinkage=0.1
boost.model<-train(classe~., method="gbm", data=training, verbose=F, trControl = t
rainControl(method = "cv", number = 10))
```

```
## Loading required package: plyr
```

```
boost.model
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12363, 12364, 12362, 12364, 12363, 12362, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7514022  0.6847606
##   1                  100      0.8240515  0.7772809
##   1                  150      0.8530973  0.8140217
##   2                   50      0.8560107  0.8175676
##   2                  100      0.9068945  0.8821711
##   2                  150      0.9318630  0.9137811
##   3                   50      0.8926986  0.8641283
##   3                  100      0.9425655  0.9273215
##   3                  150      0.9614928  0.9512805
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
plot(boost.model)
```



Prediction of the three models

```
#bagging
yhat.bag<-predict(bag.model, newdata=testing)
confusionMatrix(yhat.bag, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1658    8    2    1    0
##          B   10 1125   15    2    2
##          C    4    4 1005   16    2
##          D    0    1    2  942    2
##          E    2    1    2    3 1076
##
## Overall Statistics
##
##                Accuracy : 0.9866
##                  95% CI : (0.9833, 0.9894)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.983
##  Mcnemar's Test P-Value : 0.01504
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9904   0.9877   0.9795   0.9772   0.9945
## Specificity            0.9974   0.9939   0.9946   0.9990   0.9983
## Pos Pred Value         0.9934   0.9749   0.9748   0.9947   0.9926
## Neg Pred Value         0.9962   0.9970   0.9957   0.9955   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2817   0.1912   0.1708   0.1601   0.1828
## Detection Prevalence   0.2836   0.1961   0.1752   0.1609   0.1842
## Balanced Accuracy      0.9939   0.9908   0.9871   0.9881   0.9964
```

```
#randomForest
yhat.rf<-predict(rf.model, newdata=testing)
confusionMatrix(yhat.rf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    1 1137   10    0    0
##          C    0    0 1014    9    0
##          D    0    0    2  954    2
##          E    1    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9931, 0.9968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9982   0.9883   0.9896   0.9982
## Specificity            0.9995   0.9977   0.9981   0.9992   0.9996
## Pos Pred Value         0.9988   0.9904   0.9912   0.9958   0.9982
## Neg Pred Value         0.9995   0.9996   0.9975   0.9980   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1932   0.1723   0.1621   0.1835
## Detection Prevalence   0.2845   0.1951   0.1738   0.1628   0.1839
## Balanced Accuracy      0.9992   0.9980   0.9932   0.9944   0.9989
```

```
#boosting
yhat.boost<-predict(boost.model, testing)
confusionMatrix(yhat.boost, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1640   30    0    1    2
##          B   19 1080   37    1   16
##          C    9   29  977   41   11
##          D    1    0   10  915    8
##          E    5    0    2    6 1045
##
## Overall Statistics
##
##                Accuracy : 0.9613
##                  95% CI : (0.956, 0.966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.951
##  Mcnemar's Test P-Value : 1.979e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9797   0.9482   0.9522   0.9492   0.9658
## Specificity            0.9922   0.9846   0.9815   0.9961   0.9973
## Pos Pred Value         0.9803   0.9367   0.9157   0.9797   0.9877
## Neg Pred Value         0.9919   0.9875   0.9898   0.9901   0.9923
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2787   0.1835   0.1660   0.1555   0.1776
## Detection Prevalence   0.2843   0.1959   0.1813   0.1587   0.1798
## Balanced Accuracy      0.9859   0.9664   0.9669   0.9727   0.9815
```

Conclusions

Analysis of the three models

All the three models performe well, in fact all of them have high prediction accuracy on validation set. The best one, however, is the random forest model with a OOB estimate of error rate equal to 0.53%, while, when the model is applied to the validation dataset, both the accuracy and k-value overcome the value of 99%. Consequently we decided to apply the random forest model to the analysis of the 20 observations testing set.

Prediction on 20 testing samples

```
#randomForest
yhat.rf2<-predict(rf.model, newdata=data.testing)
yhat.rf2
```

```
##  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
```

References

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Website for the Groupware@LES (mailto:Groupware@LES) Human Activity Recognition project.

An Introduction to Statistical Learning, G. James, D. Witten, T. Hastie, R. Tibshirani. Ed. Springer Verlag (2013). ISBN: 978-1-4614-7138-7.

The Elements of Statistical Learning (2nd. Edition, 10th printing), T. Hastie, R. Tibshirani, J. Friedman. Ed. Springer Verlag (2009). ISBN: 978-0-3878-4857-0.