Practical Machine Learning Project

YVH

16 November 2015

Intro

Using data from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) we try to build a model to predict the manner in which people did an excercise: 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)

Read in the data: test and training

```
  url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  temp <- tempfile()
  download.file(url,temp,method="curl")
  TrainData <- read.csv(temp)
  unlink(temp)

  url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  temp <- tempfile()
  download.file(url,temp,method="curl")
  TestData <- read.csv(temp)
  unlink(temp)
```

Preparation

There are a lot of variables with hardly any data conataining lots of NA’s. Let’s remove those, together with the time stamps etc. I assume there is no time dependance.

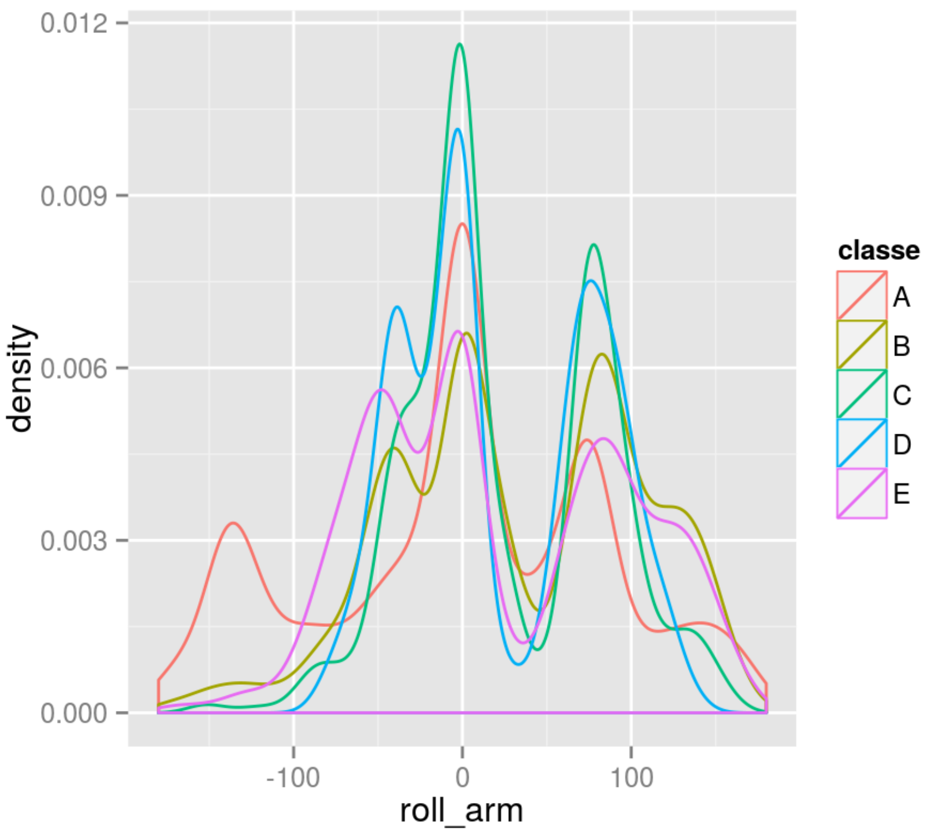
```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
varNoNA <- names(TrainData[apply(TrainData, 2, function(x) !any(is.na(x)))])
TrainDataPrep <- select(TrainData,one_of(varNoNA))
varNoEmptyNoNA <- names(TrainDataPrep[apply(TrainDataPrep, 2, function(x) !an
y(x==""))])
TrainDataPrep <- select(TrainData,one_of(varNoEmptyNoNA))
notmeaningfull <- c("X","user_name","raw_timestamp_part_1","raw_timestamp_par
t_2","cvtd_timestamp","new_window","num_window")
TrainDataPrep <- select(TrainDataPrep,-one_of(notmeaningfull))
```

There are a lot of variables. I will show one plot of a random picked variable:

```
    library(ggplot2)
    qplot(roll_arm,colour=classe, data=TrainDataPrep,geom='density')
```



By accident I found a predictor that can be used to determine if the classe is equal to A or not.

Model construction and training

There are many variables. Instead of having a look at all of them I opt for a brute force method: gradient boosting machine or GBM; a boosted tree model with all the remaining variable as predictors. We use repeated 10 fold cross validation to tune our parameters and estimate our out-of-sample error/accuracy. In order to speed thing up I use multiple CPU cores.

```
library(doMC)
library('caret')
registerDoMC(cores = 4)

set.seed(825)
fitControl <- trainControl(## 10-fold CV
                           method = "repeatedcv",
                           number = 10,
                           ## repeated three times
                           repeats = 10)



gbmFit1 <- train(classe ~ ., data = TrainDataPrep,
               method = "gbm",
               trControl = fitControl, verbose = FALSE)
```

Result

```
gbmFit1
```

```
## Stochastic Gradient Boosting
##
## 19622 samples
##    52 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 17660, 17659, 17659, 17660, 17660, 17660, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD
##   1                   50      0.7520999  0.6857081  0.009470252
##   1                  100      0.8218633  0.7745043  0.007657279
##   1                  150      0.8543268  0.8156570  0.007513900
##   2                   50      0.8561108  0.8176764  0.006993230
##   2                  100      0.9074863  0.8829296  0.006148780
##   2                  150      0.9326827  0.9148172  0.005466310
##   3                   50      0.8970751  0.8696996  0.007035011
##   3                  100      0.9432373  0.9281771  0.005045446
##   3                  150      0.9631843  0.9534240  0.004217350
##   Kappa SD
##   0.011974570
##   0.009681536
##   0.009495588
##   0.008860755
##   0.007780047
##   0.006918682
##   0.008904903
##   0.006382422
##   0.005337498
##
## 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.
```

The final model has a misclasification out-of-sample error (equals 1 minus accuracy) of 0.0368157 with a standard deviation of 0.0042173; this was estimated using 10 fold cross validation (repeated 10 times).