Practical Machine Learning

Zbyněk Zajíc

Friday, March 13, 2015

Background

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

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Data

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.1.3
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.1.3
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 3.1.3
```

```
set.seed(123)

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

```
## Warning: running command 'curl
## "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv" -o
## "pml-training.csv"' had status 127

## Warning in download.file(url = url_training, destfile = file_training,
## method = "curl"): download had nonzero exit status
```

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

```
## Warning: running command 'curl
## "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv" -o
## "pml-testing.csv"' had status 127

## Warning in download.file(url = url_testing, destfile = file_testing,
## method = "curl"): download had nonzero exit status
```

```
training<-read.csv("pml-training.csv",na.strings=c("NA",""), header=TRUE)
testing<-read.csv("pml-testing.csv",na.strings=c("NA",""), header=TRUE)
```

Preprocessing

Some of the columns are drasticaly fuyll of missing data, these columns will be deleted

```
colnames_train<-colnames(training)
nonNAs <- function(x) {
    as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))
}
colcnts <- nonNAs(training)
drops <- c()
for (cnt in 1:length(colcnts)) {
    if (colcnts[cnt] < nrow(training)) {
        drops <- c(drops, colnames_train[cnt])
    }
}


training <- training[,!(names(training) %in% drops)]
```

```
testing <- testing[,!(names(testing) %in% drops)]

#delete boring columns
training <- training[,8:length(colnames(training))]
testing <- testing[,8:length(colnames(testing))]
colnames_train<-colnames(training)
```

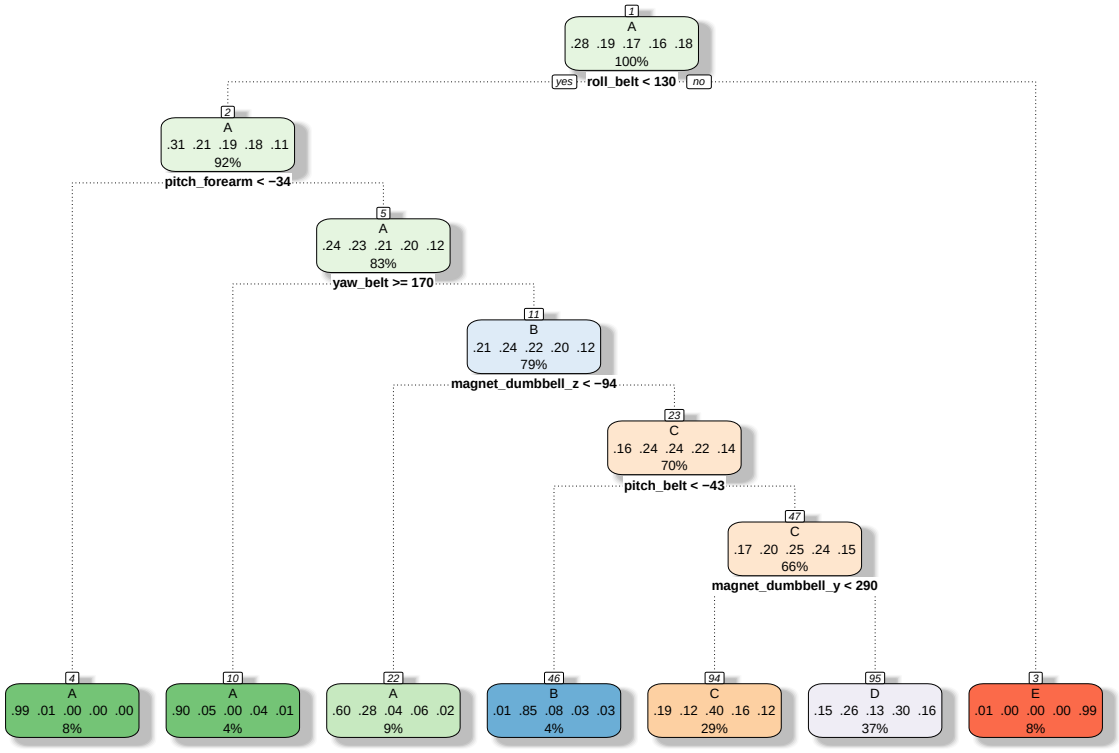
Split training data into tr and dev

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
tr <- training[inTrain, ];
dev <- training[-inTrain, ]
```

Modeling

Decision Tree

```
modFitDT <- train(classe ~ .,method='rpart', data=tr)
fancyRpartPlot(modFitDT$finalModel)
```



```r
#prediction from development data
pred <- predict(modFitDT, newdata=dev)
print(confusionMatrix(pred, dev$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1356  239   42   68   20
##          B    5  264   34    8    6
##          C  441  263  907  397  269
##          D  422  752  385  813  510
##          E    8    0    0    0  637
##
## Overall Statistics
##
##                Accuracy : 0.5069
##                  95% CI : (0.4958, 0.518)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3865
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6075  0.17391   0.6630   0.6322  0.44175
## Specificity            0.9343  0.99162   0.7885   0.6846  0.99875
## Pos Pred Value         0.7861  0.83281   0.3983   0.2821  0.98760
## Neg Pred Value         0.8569  0.83344   0.9172   0.9047  0.88821
## Prevalence             0.2845  0.19347   0.1744   0.1639  0.18379
## Detection Rate         0.1728  0.03365   0.1156   0.1036  0.08119
## Detection Prevalence   0.2199  0.04040   0.2902   0.3673  0.08221
## Balanced Accuracy      0.7709  0.58277   0.7258   0.6584  0.72025
```

The results on development data is wery poor (acc~50%)

Important variables

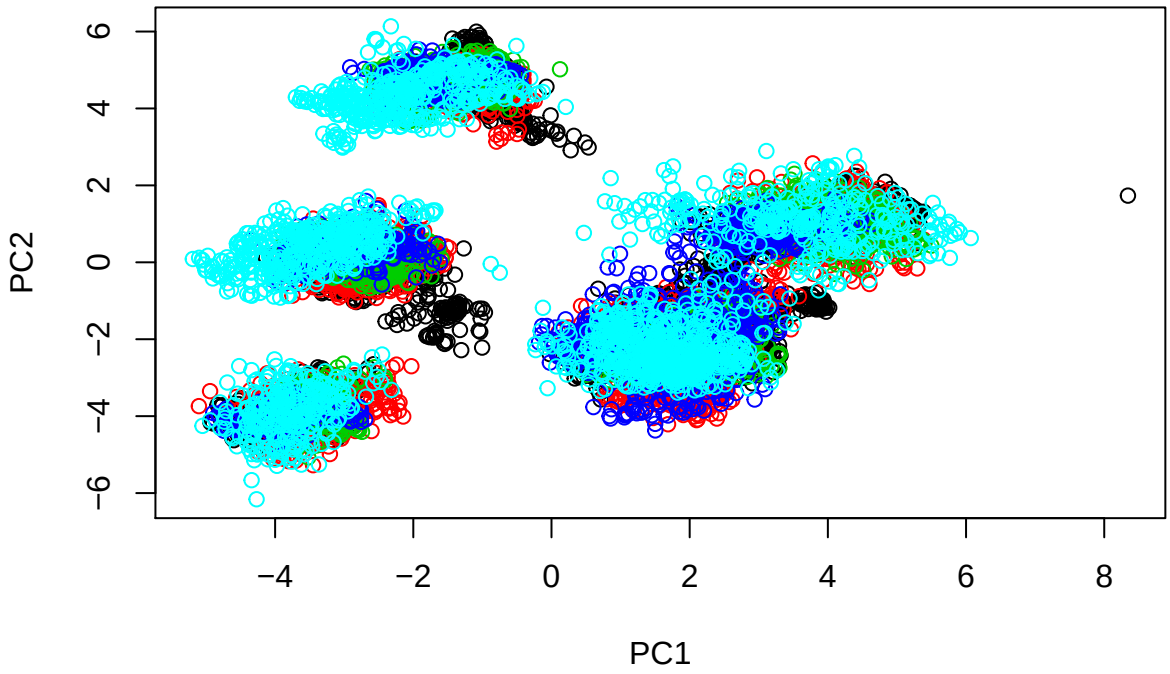
```r
varImp(modFitDT$finalModel)
```

```
##                       Overall
## accel_arm_x          337.8856
## accel_belt_z         551.5419
## accel_dumbbell_y     424.4081
## accel_forearm_x      217.1037
## magnet_arm_x         358.1205
## magnet_belt_y        511.9966
## magnet_dumbbell_x    410.8705
## magnet_dumbbell_y   1309.7523
## magnet_dumbbell_z    493.8489
## pitch_belt           487.7074
```

```
## pitch_forearm        1269.5509
## roll_belt            1325.2042
## roll_dumbbell         686.7328
## roll_forearm          634.9154
## total_accel_belt      457.9652
## yaw_belt              274.7506
## gyros_belt_x            0.0000
## gyros_belt_y            0.0000
## gyros_belt_z            0.0000
## accel_belt_x            0.0000
## accel_belt_y            0.0000
## magnet_belt_x           0.0000
## magnet_belt_z           0.0000
## roll_arm                0.0000
## pitch_arm               0.0000
## yaw_arm                 0.0000
## total_accel_arm         0.0000
## gyros_arm_x             0.0000
## gyros_arm_y             0.0000
## gyros_arm_z             0.0000
## accel_arm_y             0.0000
## accel_arm_z             0.0000
## magnet_arm_y            0.0000
## magnet_arm_z            0.0000
## pitch_dumbbell          0.0000
## yaw_dumbbell            0.0000
## total_accel_dumbbell    0.0000
## gyros_dumbbell_x        0.0000
## gyros_dumbbell_y        0.0000
## gyros_dumbbell_z        0.0000
## accel_dumbbell_x        0.0000
## accel_dumbbell_z        0.0000
## yaw_forearm             0.0000
## total_accel_forearm     0.0000
## gyros_forearm_x         0.0000
## gyros_forearm_y         0.0000
## gyros_forearm_z         0.0000
## accel_forearm_y         0.0000
## accel_forearm_z         0.0000
## magnet_forearm_x        0.0000
## magnet_forearm_y        0.0000
## magnet_forearm_z        0.0000
```

Only few are important and it is movement in tree axes (xyz) -> try to find only 2 importatnt pca features

```
PCAf<- preProcess(tr[,-length(tr)], method="pca", pcaComp = 2)
trPCA<-predict(PCAf,tr[,-length(tr)])
plot(trPCA,col=tr$classe)
```



in the figure there is no pattern, so try 10 pcaComp an train a Desision Tree on them

```
PCAf10<- preProcess(tr[,-length(tr)], method="pca", pcaComp = 10)
trPCA10<-predict(PCAf10,tr[,-length(tr)])
modFitDTPCA10 <- train(tr$classe ~ .,method='rpart', data=trPCA10)
devPCA10<-predict(PCAf10,dev[,-length(dev)])
pred <- predict(modFitDTPCA10, newdata=devPCA10)
print(confusionMatrix(pred, dev$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1758  752  928  614  658
##          B  203  339  139  190  206
##          C    0    0    0    0    0
##          D   86  236   56  372  169
##          E  185  191  245  110  409
##
## Overall Statistics
##
##                Accuracy : 0.3668
##                  95% CI : (0.3561, 0.3776)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.1633
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7876  0.22332   0.0000  0.28927  0.28363
## Specificity            0.4742  0.88338   1.0000  0.91662  0.88585
## Pos Pred Value         0.3732  0.31476      NaN  0.40479  0.35877
## Neg Pred Value         0.8489  0.82582   0.8256  0.86805  0.84596
## Prevalence             0.2845  0.19347   0.1744  0.16391  0.18379
## Detection Rate         0.2241  0.04321   0.0000  0.04741  0.05213
## Detection Prevalence   0.6003  0.13727   0.0000  0.11713  0.14530
## Balanced Accuracy      0.6309  0.55335   0.5000  0.60294  0.58474
```

Now the ACC~0.37, it is worst

Random Ferest

Time consuming computeation -> small trainning dataset

```
inTrain <- createDataPartition(y=tr$classe, p=0.2, list=FALSE)
tr_small <- tr[inTrain, ];
modFitRF_small<-train(classe~.,method='rf', data=tr_small, prox=TRUE)
```

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

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
#prediction from development data
pred <- predict(modFitRF_small, newdata=dev)
print(confusionMatrix(pred, dev$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2209   64    4    6    1
##          B    7 1397   66    6   12
##          C   12   47 1268   42   22
##          D    1    6   30 1231   27
##          E    3    4    0    1 1380
##
## Overall Statistics
##
##                Accuracy : 0.954
##                  95% CI : (0.9491, 0.9585)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9418
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9897   0.9203   0.9269   0.9572   0.9570
## Specificity            0.9866   0.9856   0.9810   0.9902   0.9988
## Pos Pred Value         0.9672   0.9388   0.9116   0.9506   0.9942
## Neg Pred Value         0.9959   0.9810   0.9845   0.9916   0.9904
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2815   0.1781   0.1616   0.1569   0.1759
## Detection Prevalence   0.2911   0.1897   0.1773   0.1651   0.1769
## Balanced Accuracy      0.9882   0.9530   0.9540   0.9737   0.9779
```

Now the ACC~0.90 on development data

Prediction testing data with Random Ferest

```
#prediction from development data
pred <- predict(modFitRF_small, newdata=testing)
print(pred)
```

```
##  [1] C A B A A E D D A A B C B A E E A B A B
## Levels: A B C D E
```