# Predicting exercise quality with random forest

This is a course project for the practical machine learning class from JHU on Coursera. The data for this project is published in a conference proceeding: Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. The data is kindly made available by the authors at http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Emma Yu

Jan 2015

## Introduction

Many effort has been put into recognizing the type of exciecising, but people don't usually evaluate the correctness and effectiveness of the exercise. The goal of this project is to use controled data to train the algorithm to learn to evaluate exercise quality. Data is gathered 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. 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 Processing

First, let's download the training and testing data set from:

```
download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',
              "pml-training.csv", method = "curl")
download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',
              "pml-testing.csv", method = "curl")
```

In my experiments, I found using 10% of the training dataset provides 88% prediction accuracy. Since increasing the amount of training data will increase the running time dramatically, and the 88% accuracy is already acceptable. I decided to use only 10 percent of the training data set to build our model for prediction.

```
# Subset 10% of the data
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")

library(caret)
```

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

```
insmall <- createDataPartition(y = training$classe,
                               p = 0.90, list = FALSE)
smdata <- training[-insmall,]

# First only extract accelerator data
names <- names(smdata)
acc <- grep('acc', names)
acc <- c(acc, grep('classe', names))
smacc <- smdata[acc]
#head(smacc)

namesacc <- names(smacc)
varacc <- grep('var', namesacc)
smacc <- smacc[-varacc]
#head(smacc)
```

## Building the model

After several experiment, I found random forest can achieve decent performace with a relative small amount of data. The algorithm takes 367.261s in total to process.

```
system.time(modFit <- train(classe ~., data = smacc, method ='rf', prox = TRUE))
```

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

```
##    user  system elapsed 
## 387.773   4.930 393.443
```

```
modFit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 16.38%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 504  13  15  23   3  0.09677419
## B  37 292  31   9  10  0.22955145
## C  23  20 284  13   2  0.16959064
## D  20   7  29 258   7  0.19626168
## E   6  25  15  13 301  0.16388889
```

## Model testing

We testing the algorithm with test data ramdomply selected from the original dataset, and clean the data as the training set.

```
test <- createDataPartition(y = training$classe,
                            p = 0.9, list = FALSE)
testdata <- training[-test,]

# clean just as the training dataset
names <- names(testdata)
acc <- grep('acc', names)
acc <- c(acc, grep('classe', names))
smtest <- testdata[,acc]

namesacc <- names(smtest)
vartest <- grep('var', namesacc)
smtest <- smtest[-vartest]
#head(smtest)

pred <- predict(modFit, smtest)
smtest$predRight <- pred == smtest$classe
table(pred, smtest$classe)
```

```
##
## pred   A   B   C   D   E
##    A 515  34  14  18   6
##    B   8 295  11   3  23
##    C  18  35 309  24  14
##    D  15   9   5 270   7
##    E   2   6   3   6 310
```

The estimation of the out-of-sample accurracy is:

```
sum(smtest$predRight)/1960.
```

```
## [1] 0.8668367
```

## Model prediction

```
# clean just as the training dataset
names <- names(testing)
acc <- grep('acc', names)
acc <- c(acc, grep('classe', names))
smtesting <- testing[,acc]

namesacc <- names(smtesting)
vartest <- grep('var', namesacc)
smtestting <- smtesting[-vartest]
#head(smtest)

answers <- predict(modFit, smtesting)
```

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

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(answers)
```