# personal motion recognition

*Jie Wu*

*February 26, 2019*

## Introduction

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. 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.

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. The goal of this project is to use data from accelerometers on the (1) belt (2) forearm (3) arm and (4) dumbell of the participants to identify the 5 correct and incorrect ways of barbell lifting.

## Data Source

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The dataset used in this project is a courtesy of "Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements".

## Date Processing

First we remove the columns with missing values. It turns out that the columns are such that either they contain no missing value or the vast majority of entries are missing values (19262 out of 19622).

```
suppressMessages(library(caret))
training <- read.csv('pml-training.csv')
## remove columns with NA
varcol <- sapply(training, function(x) sum(is.na(x))) == 0
training <- training[,varcol]
```

Now we extract the data produced from the four sensors on (1) belt (2) arm (3) dumbbell (4) forearm.

```
## four sensors data
beltcolindx <- grep('belt',names(training))
armcolindx <- grep('_arm',names(training))
dumbbellcolindx <- grep('dumbbell',names(training))
forearmcolindx <- grep('forearm',names(training))
colindx <- c(beltcolindx,armcolindx,dumbbellcolindx,
             forearmcolindx,dim(training)[2])
```

Finally we remove columns with near zero variance. It turns out that all the factor columns are such columns. This removal leaves the remaining columns with either numeric or integer class type.

```
nzv <- nearZeroVar(training)
colindx <- setdiff(colindx,nzv)
train <- training[,colindx]
```

## Prediction Analysis

We first separate the data into training and validation set.

```
inValid <- createDataPartition(y=train$classe,p=0.7,list=FALSE)
builddata <- train[inValid,]
validdata <- train[-inValid,]
```

### Prediction using decision tree

```
set.seed(1)
library(rpart)
system.time(fittree <- rpart(classe~.,data=builddata,method='class'))
```

```
##    user  system elapsed
##    1.91    0.00    1.91
```

```
predtree <- predict(fittree,validdata,type='class')
cfmtree <- confusionMatrix(predtree, validdata$classe)
cfmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1500  191   24   50   18
##          B   56  638   56   55   84
##          C   46  134  860  156  131
##          D   46   83   62  614   69
##          E   26   93   24   89  780
##
## Overall Statistics
##
##                Accuracy : 0.7463
##                  95% CI : (0.735, 0.7574)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6786
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8961   0.5601   0.8382   0.6369   0.7209
## Specificity            0.9328   0.9471   0.9039   0.9472   0.9517
## Pos Pred Value         0.8413   0.7177   0.6481   0.7025   0.7708
## Neg Pred Value         0.9576   0.8997   0.9636   0.9302   0.9380
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2549   0.1084   0.1461   0.1043   0.1325
```

```
## Detection Prevalence   0.3030   0.1511   0.2255   0.1485   0.1720
## Balanced Accuracy      0.9144   0.7536   0.8710   0.7920   0.8363
```

```
sprintf("Confusion matrix accuracy of decision tree is %.4f",cfmtree$overall[1])
```

```
## [1] "Confusion matrix accuracy of decision tree is 0.7463"
```

### Prediction using linear discriminant analysis

```
system.time(fitlda <- train(classe~.,data=builddata,method='lda'))
```

```
##    user  system elapsed
##    9.96    1.21   11.17
```

```
predlda <- predict(fitlda,validdata)
cfmlda <- confusionMatrix(predlda, validdata$classe)
cfmlda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1363  185   94   64   34
##          B   38  702  106   42  189
##          C  155  155  682  120   92
##          D  110   37  116  692  103
##          E    8   60   28   46  664
##
## Overall Statistics
##
##                Accuracy : 0.6972
##                  95% CI : (0.6853, 0.7089)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6168
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8142   0.6163   0.6647   0.7178   0.6137
## Specificity            0.9105   0.9210   0.8926   0.9256   0.9704
## Pos Pred Value         0.7833   0.6518   0.5664   0.6541   0.8238
## Neg Pred Value         0.9250   0.9091   0.9265   0.9437   0.9177
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2316   0.1193   0.1159   0.1176   0.1128
## Detection Prevalence   0.2957   0.1830   0.2046   0.1798   0.1370
## Balanced Accuracy      0.8623   0.7687   0.7786   0.8217   0.7921
```

```
sprintf("Confusion matrix accuracy of linear discriminant is %.4f",cfmlda$overall[1])
```

```
## [1] "Confusion matrix accuracy of linear discriminant is 0.6972"
```

### Prediction using random forest

```
ctrlrf <- trainControl(method='cv',number=3,verboseIter=FALSE)
system.time(fitrf <-train(classe~.,data=builddata,method='rf',trControl=ctrlrf))
```

```
##    user  system elapsed
##  361.50    2.45  366.13
```

```
## summary(fitrf$finalModel)
predrf <- predict(fitrf,validdata)
cfmrf <- confusionMatrix(predrf, validdata$classe)
cfmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    9    0    0    0
##          B    2 1127    8    0    1
##          C    1    2 1013   13    1
##          D    0    1    5  951    7
##          E    0    0    0    0 1073
##
## Overall Statistics
##
##                Accuracy : 0.9915
##                  95% CI : (0.9888, 0.9937)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9893
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9895   0.9873   0.9865   0.9917
## Specificity            0.9979   0.9977   0.9965   0.9974   1.0000
## Pos Pred Value         0.9946   0.9903   0.9835   0.9865   1.0000
## Neg Pred Value         0.9993   0.9975   0.9973   0.9974   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1915   0.1721   0.1616   0.1823
## Detection Prevalence   0.2855   0.1934   0.1750   0.1638   0.1823
## Balanced Accuracy      0.9980   0.9936   0.9919   0.9919   0.9958
```

```
sprintf("Confusion matrix accuracy of random forest is %.4f",cfmrf$overall[1])
```

```
## [1] "Confusion matrix accuracy of random forest is 0.9915"
```

### Prediction using gradient boosting model

```
ctrlgbm <- trainControl(method='repeatedcv',number=3,repeats=1)
system.time(fitgbm <- train(classe~.,data=builddata,method='gbm',
                                  trControl=ctrlgbm,verbose=FALSE))
```

```
##    user  system elapsed
##  150.89    0.67  151.86
```

```
## summary(fitgbm$finalModel)
predgbm <- predict(fitgbm,validdata)
cfmgbm <- confusionMatrix(predgbm, validdata$classe)
cfmgbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   46    0    1    2
##          B   12 1058   37    0   12
##          C   11   32  975   29    5
##          D    2    1   12  926   12
##          E    2    2    2    8 1051
##
## 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 : 4.242e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9289   0.9503   0.9606   0.9713
## Specificity            0.9884   0.9871   0.9842   0.9945   0.9971
## Pos Pred Value         0.9711   0.9455   0.9268   0.9717   0.9869
## Neg Pred Value         0.9936   0.9830   0.9894   0.9923   0.9936
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1798   0.1657   0.1573   0.1786
## Detection Prevalence   0.2882   0.1901   0.1788   0.1619   0.1810
## Balanced Accuracy      0.9861   0.9580   0.9672   0.9775   0.9842
```

```
sprintf("Confusion matrix accuracy of boosted trees is %.4f",cfmgbm$overall[1])
```

```
## [1] "Confusion matrix accuracy of boosted trees is 0.9613"
```

Comparing the four models above we see that random forest yields the best accuracy yet it takes the longest time, which is followed by gradient boosting, decision trees and linear discriminant analysis. In fact the performance of random forest and gradient boosting machine are comparable yet random forest takes almost twice as long time to finish. Although this may not be a big issue for such a small data set the tradeoff should be made between accuracy and computational cost when choosing from random forest and gradient boosting for big data classification.

## Predicting Test Data

We now use the best prediction model found above to predict test data.

```
testing <- read.csv('pml-testing.csv')
testing <- testing[,varcol]
```

```
test <- testing[,colindx]
quizans <- predict(fitrf,test)
quizans
```

```
##  [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
```

## Appendix

Among the four models only the decision tree yields a fairly interpretable model. The rest works like a black box. We hence only plot the decision tree model below.

```
library(rpart.plot)
prp(fittree)
```