Machine Learning Techniques on a Human Activity Dataset

Robert Hadow

Abstract

The PML dataset described herein was generated in a series of observations of human physical activity: dumbbell lifting. It used magnetic orientation sensors and solid-state accelerometers and gyroscopes mounted on the subject's midsection, upper arm, wrist, and on a 1.25 kg dumbbell. The object of this project is to identify from the data a method that predicts which of five activities were being conducted: a proper dumbbell lift or four improper variants.

After a look at the data, we eliminated variables for which 80% or more of the values were NA. We conducted a principal components analysis, which showed, in our opinion, little duplication. We tried several approaches, including CART, Receiver Operating Characteristics, and Random Forest.

Basis Data

Detectors of this type can directly measure the following at a rate of forty-five observations per second:

- linear acceleration in three dimensions
- orientation in three dimensions
- rotation in three dimensions

The analysts computed certain statistics from the original observations. Not all measures got secondary analysis. We would dump statistics that require too much imputation.

- kurtosis
- skewness
- mean
- sd

First Look

After we unpacked the data, we made an ocular examination of its structure and content. See code segment *preprocess 1*.

The training data set includes 19622 rows of 160 columns. The direct observations are provided as numerics or integers. Calculated values are provided as character strings. There are many missing values (blanks) and NAs.

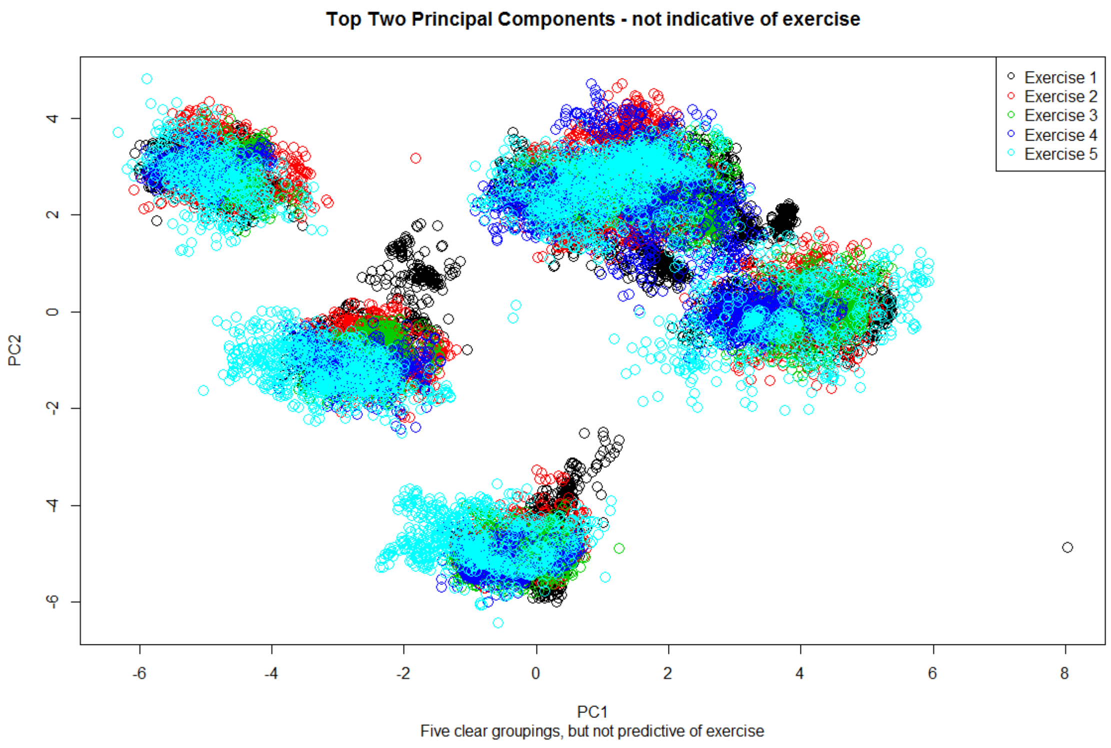
Method

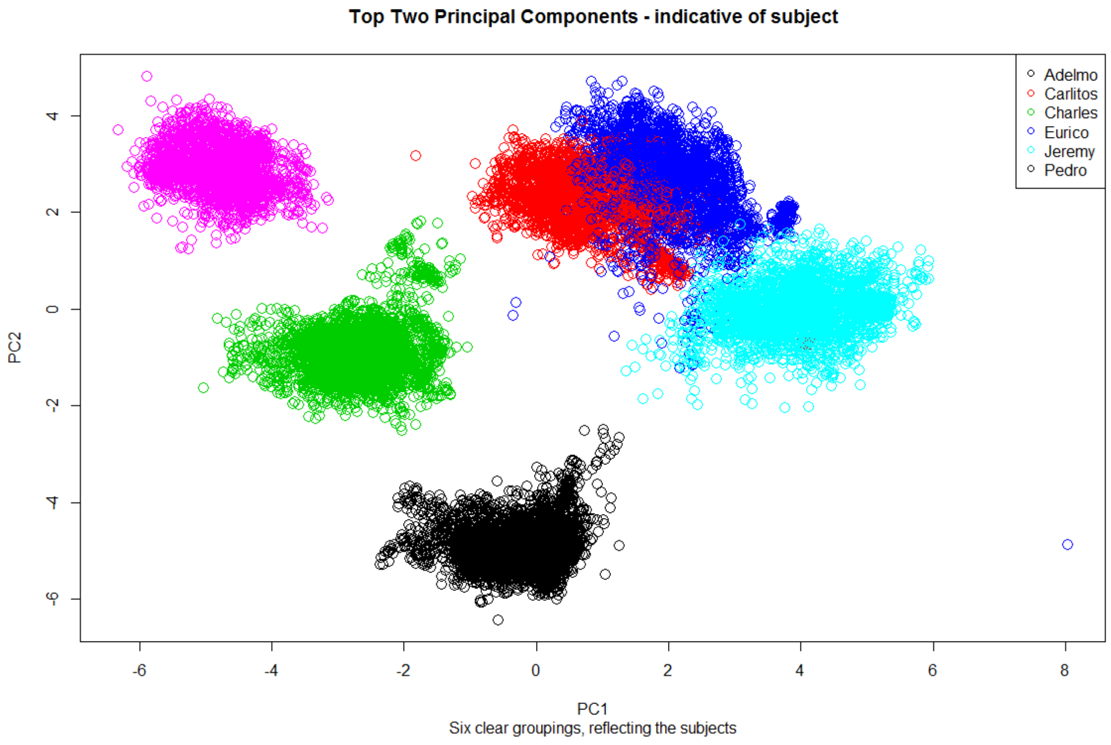
(1) Convert subjects and known exercise type to factors
(2) Convert character columns to numeric
(3) Drop all columns with more than 50% NA values
(4) While testing code, limit data sets to 1.5% of original size
(5) Investigate possibility of combining principal components
(6) Partition data 60% training, 40% testing
(7) Use the Random Forest algorithm twenty-five times to develop a model
(8) Test the model against the reserved testing data

When a satisfactory result is obtained, perform steps 1-3 on the test set provided, then transform it for submission and grading. See code segment *preprocess 2*.

Object of Analysis

We seek to use the PML-Training dataset to learn to identify exercises by their characteristics, then predict from a small sample what exercise was conducted. See code segment *PCA investigation*.





First we reduced the number of variables by PCA analysis. We chose to see how many combinations of columns could explain 80% of the variability of the data set. We were able to distill 80% of the variability of the data set into 14 variables. Even for the first two principal measures, there is little association. Insofar as the two most correlated variables from the dataset still remaining show little correlation, we abandoned our quest to reduce the number of variables further.

Predicting Out-of-Sample Error

True Prediction Error = Training Error + Training Optimism

True Prediction Error = Training Error + f(Model Complexity)

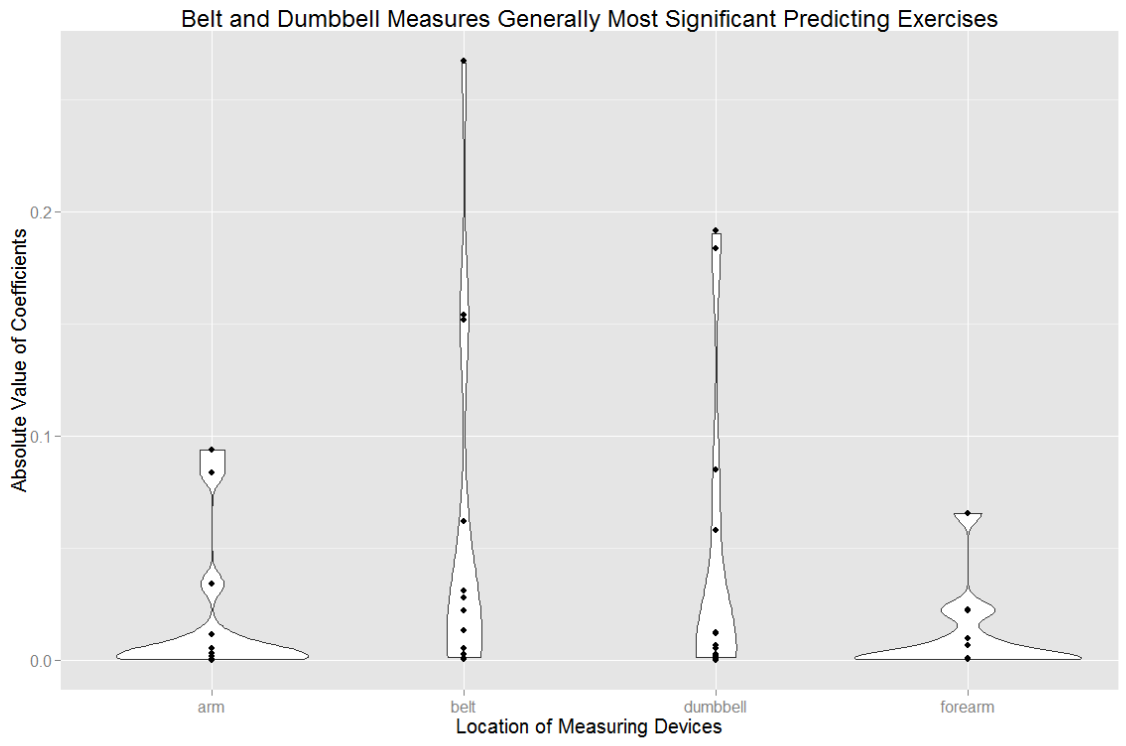
In the assignment, we are asked to use a cross-validation technique to check our model complexity. The simplest measure of complexity is the number of predictors. We used holdout data as a surrogate for fresh data. We used a random sample of 60% of our data as the training set and 40% as the holdout - a test set.

We used our reserved test set to develop an estimate of out-of-sample error.

Leave-One-Out Cross Validation

We chose not to try Leave-One-Out Cross Validation because the data set is of sufficient size not to warrant it.

Another approach to cross validation would be to run a set of analyses with one subject of six left out of the mix. This would be particularly effective if the test set included a completely new subject (which our project does not). See code segment *most significant*.



CART Tree Method

We used the CART Tree method, but with less than satisfactory results. See code segment *CART investigation*.

```
##    user  system elapsed
##    4.37    0.16   18.02

##
## pred    A    B    C    D    E
##    A 1951  363  218  230   36
##    B   38  532   51  226   92
##    C  211  623 1099  781  415
##    D    0    0    0    0    0
##    E   32    0    0   49  899
```

Receiver Operating Characteristics Method

We used the ROCC method, with better results. See code segment *rocc investigation*.

```
##    user  system elapsed 
##    1.37    0.23   47.10 

## [1] "ROCC Prediction Model run against reserved test set"

## 
## pred    A    B    C    D    E
##    A  741   74   81   42   78
##    B 1491 1444 1287 1244 1364
##    C    0    0    0    0    0
##    D    0    0    0    0    0
##    E    0    0    0    0    0
```

Random Forest Method

We used the Random Forest Method with good results. See code segment *forest*

```
##     user   system  elapsed 
##   222.00     3.89 35094.82 

## [1] "Random Forest Prediction Model run against training set (along the x-axis)"

## 
## pred    A    B    C    D    E
##    A 3348    0    0    0    0
##    B    0 2279    0    0    0
##    C    0    0 2054    0    0
##    D    0    0    0 1930    0
##    E    0    0    0    0 2165

## [1] "Random Forest Prediction Model run against reserved test set (along the x-axis)"

## 
## pred    A    B    C    D    E
##    A 2232    4    0    0    0
##    B    0 1510    1    0    0
##    C    0    4 1367    2    0
##    D    0    0    0 1284    0
##    E    0    0    0    0 1442
```

Select Prediction Solution to Problem Set

We eliminated predictors for which 80% of the data was NA. We looked at Principal Components. While fourteen PCA predictors accounted for 80% of the variability, they did not show apparent correlation. We therefore chose not to combine any predictors.

First we tried a CART prediction model, which provided reasonable but not satisfactory results

We tried a Receiver Operating Characteristic (ROC) model, which provided unsatisfactory results.

In the end, a Random Forest model worked best. It is our choice for prediction. See code segment *predict solution*.

```
##   Code             Method Sample_Accuracy Est_OOS_Accuracy
## 1 cart               CART       0.5721807        0.5711190
## 2 rocc Receiver Operating       0.2784477        0.2784859
## 3   rf      Random Forest       1.0000000        0.9985980
```

```
## [1] "Random Forest Prediction Model run against problem set"
```

```
## 
## pred 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
##    A  0  1  0  1  1  0  0  0  1  1  0  0  0  1  0  0  1  0  0  0
##    B  1  0  1  0  0  0  0  1  0  0  1  0  1  0  0  0  0  1  1  1
##    C  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##    D  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##    E  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1  1  0  0  0  0
```

```
##    pmltest.problem_id pred
## 1                  01    B
## 2                  02    A
## 3                  03    B
## 4                  04    A
## 5                  05    A
## 6                  06    E
## 7                  07    D
## 8                  08    B
## 9                  09    A
## 10                 10    A
## 11                 11    B
## 12                 12    C
## 13                 13    B
## 14                 14    A
## 15                 15    E
## 16                 16    E
## 17                 17    A
## 18                 18    B
## 19                 19    B
## 20                 20    B
```

APPENDICES

Code

preprocess_1

```r
suppressPackageStartupMessages(library(knitr))
knitr::opts_chunk$set(fig.width=12, fig.height=8, echo = FALSE)

if(!file.exists("figures")) dir.create("figures")
figDir <- "figures/"
opts_chunk$set(fig.path = figDir)
squelch = TRUE # FALSE if running in console; TRUE if running in knitr

if(!file.exists("data")) dir.create("data")
dataDir <- "data/"

fileURL = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
targetFile = "data/pml-training.csv"
download.file(fileURL, targetFile, mode="wb")
pml.train <- read.csv(targetFile, stringsAsFactors = FALSE,  na.strings=c(" ", "", "NA"))

fileURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
targetFile <- "./data/pml-testing.csv"
download.file(fileURL, targetFile, mode="wb")
pml.test <- read.csv(targetFile, stringsAsFactors = FALSE, na.strings=c(" ", "", "NA"))

cols <- length(colnames(pml.train))
rows <- length(rownames(pml.train))
```

preprocess_2

```r
suppressPackageStartupMessages(library(e1071))
suppressPackageStartupMessages(library(R.utils))
suppressPackageStartupMessages(library(plyr))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(data.table))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(randomForest))
suppressPackageStartupMessages(library(rocc))
suppressPackageStartupMessages(library(rpart))
suppressPackageStartupMessages(library(rmarkdown))

# I used parallel during development, but ran into problems with reproduceability and thras
hing
suppressPackageStartupMessages(library(doParallel))
cl <- makeCluster(detectCores()-2)
registerDoParallel(cl)

pmltrain <- pml.train[, -1] # Column 1 is worthless in any case
pmltest  <- pml.test[, -1]

x <- pml.test # initialize x
Code <- c("cart", "rocc", "rf")
Method <- c("CART", "Receiver Operating", "Random Forest")
Sample_Accuracy <- Est_OOS_Accuracy <- c(0.0, 0.0, 0.0)
```

```r
results <- data.frame(Code, Method, Sample_Accuracy, Est_OOS_Accuracy)

# pml_write_files takes two-column matrix and writes out solution to assignment
pml_write_files = function(x){
  n = nrow(x)
  for(i in 1:n){
    filename = paste0("problem_id_",x[i,1],".txt")
    write.table(x[i,2],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

# accuracy() fit is the result of train(). data is the training or test set 15-11-19
# y is outcome. Should be in quotes.
accuracy <- function(fit = modfit, data = training, y = "classe") {
        prediction <- predict(fit, data)
        data[ , "predRight"] <- data[ , y] == prediction
        sum(data[ , "predRight"])/length(data[,"predRight"])
                }

# Convert some columns to factor
# Absolutely parallel treatment two sets, training and test
# (it would make sense to build a function, but column names are different)
pmltrain[, "user_name"]  <- as.factor(pmltrain[, "user_name"])
pmltest[,  "user_name"]  <- as.factor(pmltest[,  "user_name"])
pmltrain[, "classe"]     <- as.factor(pmltrain[, "classe"])
pmltest[,  "problem_id"] <- as.factor(pmltest[,  "problem_id"])

# Convert character columns to numeric
ccols <- function(x) {
        for (i in 1:ncol(x)) {
        if(class(x[1, i]) == "character") {
                x[,i] <- suppressWarnings(as.numeric(x[,i]))
        }}
        x
}

pmltrain <- ccols(pmltrain)
pmltest  <- ccols(pmltest)

# Establish columns to drop from train. Exactly same ones will be dropped from test.
na_count <-sapply(pmltrain, function(y) sum(length(which(is.na(y)))))

threshhold <- .5 * nrow(pmltrain)
selCol = na_count < threshhold
pmltrain <- pmltrain[ , selCol]
pmltest <- pmltest[ , selCol]

# TEMPORARY  Make smaller dataset
set.seed <- 104729
# pmltrain <- sample_frac(pmltrain, .01)

# end of preprocessing_2
```

PCA_investigation

```r
set.seed <- 104729
# Copy dataset without worthless x column and subject names
selCol  <- c("x")
```

```r
pmltemp <- pmltrain[, -1]

# determine 80% threshhold
preProc <- preProcess(pmltrain[, !colnames(pmltrain) == "classe"], method ="pca", thresh=.8
)
trainPC <- predict(preProc, pmltrain[, !colnames(pmltrain) == "classe"])

PCACount = sum(grepl("^PC", colnames(preProc$rotation)))

# Try the signficant PCAs
preProc <- preProcess(pmltrain[, !colnames(pmltrain) == "classe"], method="pca", pcaComp=PC
ACount)
trainPC <- predict(preProc, pmltrain[, !colnames(pmltrain) == "classe"])

plot(trainPC[, "PC1"], trainPC[, "PC2"],
    main="Top Two Principal Components - not indicative of exercise",
    col=pmltemp[, "classe"], cex = 1.5,
     ylab="PC2",  xlab="PC1",
    sub = "Five clear groupings, but not predictive of exercise")
legend("topright", legend = c("Exercise 1", "Exercise 2", "Exercise 3",
                              "Exercise 4", "Exercise 5"),
                     col = c(1:5), lty = 0, pch = 1)

plot(trainPC[, "PC1"], trainPC[,"PC2"],
    main="Top Two Principal Components - indicative of subject",
    col=as.integer(factor(pmltrain$user_name)),
     ylab="PC2", xlab = "PC1",  cex = 1.5,
    sub = "Six clear groupings, reflecting the subjects")
legend("topright", legend = c("Adelmo", "Carlitos","Charles","Eurico","Jeremy","Pedro"),
                     col = c(1:5), lty = 0, pch = 1)
```

CART investigation

```r
# Replace subject variable with five binary variables to accomodate lm
set.seed <- 104729
temp <- createDataPartition(y = pmltrain$classe, p=.6, list = FALSE)
training <- pmltrain[ temp,]
testing  <- pmltrain[-temp,]
rm(temp)

system.time(modFit <- train(classe ~ ., data = training, method = "rpart"))

pred <- predict(modFit, testing)
results[Code == "cart", 3] <- mean(modFit$resample$Accuracy)
results[Code == "cart", 4] <- mean(modFit$resample$Kappa)
#x <- table(pred, testing$classe)
table(pred, testing$classe)

results[Code == "cart", 3] <- accuracy(modFit, training,"classe")
results[Code == "cart", 4] <- accuracy(modFit, testing,"classe")
```

ROCC investigation

```r
set.seed <- 104729
temp <- createDataPartition(y = pmltrain$classe, p=.6, list = FALSE)
training <- pmltrain[ temp,]
testing  <- pmltrain[-temp,]
rm(temp)
```

```r
system.time(modFit <- train(classe ~ ., data = training, method = "rocc")) #, prox = TRUE))

pred <- predict(modFit, testing)
testing$predRight <- pred == testing$classe
"ROCC Prediction Model run against reserved test set"
table(pred, testing$classe)

results[Code == "rocc", 3] <- accuracy(modFit, training,"classe")
results[Code == "rocc", 4] <- accuracy(modFit, testing,"classe")
```

forest

```r
set.seed <- 104729
temp <- createDataPartition(y = pmltrain$classe, p=.6, list = FALSE)
training <- pmltrain[ temp,]
testing  <- pmltrain[-temp,]
rm(temp)

system.time(modFit <- train(classe ~ ., data = training, method = "rf", prox = TRUE))

"Random Forest Prediction Model run against training set (along the x-axis)"
pred <- predict(modFit, training)
table(pred, training$classe)

"Random Forest Prediction Model run against reserved test set (along the x-axis)"
pred <- predict(modFit, testing)
table(pred, testing$classe)

results[Code == "rf", 3] <- accuracy(modFit, training, "classe")
results[Code == "rf", 4] <- accuracy(modFit, testing,  "classe")
```

Attribution

This analysis relies on a data set kindly provided by:

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

Use *this project* for academic purposes may be restricted by the ethics standards of your institution. Be aware that the text and code in this project includes certain errors and signatures to allow easy identification of derivative works.

Random Forests$\textregistered$ is a trademark of Health Care Productivity, Inc.