Human Activity Recognition Project

Jeremy Berg

January 31, 2016

Executive Summary

A major challenge for analyzing data from wearable devices is recognition not just of what exercise an individual is performing, but also how the individual is doing it. A dataset from accelerometers on the belt, forearm, arm, and dumbell of 6 participants from http://groupware.les.inf.puc-rio.br/har was analyzed. These individuals were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The dataset was divided into a training set (75%) and a test/cross validation set (25%). A random forest model was developed based on 60 variables that correctly classified all cases in the training set accurately. This model was then applied to a cross validation set to yield 99.4% accuracy.

Introduction

(From http://groupware.les.inf.puc-rio.br/har) 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. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Data Loading and Cleaning

```
## Loading required package: lattice
## Loading required package: ggplot2
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

The dataset (pml-training.csv) was first loaded.

```
setwd("~/Data_Science_Course/jeremyberg.github.io")
train_set <- read.csv("pml-training.csv", stringsAsFactors = FALSE, na.strings = c("NA", "#DIV/0!", ""))
num_row <- nrow(train_set)
num_col <- ncol(train_set)
```

The original dataset has 19622 rows and 160 columns. As a first step in cleaning up the dataset, the dataset was checked for missing data.

```
# Eliminate columns with large numbers of missing data
# Define a function to count number of non-NA values
count_nonNA <- function(vec) {
        num_nonNA <- length(vec[!is.na(vec)])
```

```
}
ncount <- integer(num_col)
col_to_keep <- integer(0)
for (i in 1:num_col) {
        ncount[i] <- count_nonNA(train_set[, i])
        if (ncount[i] == num_row) {
              col_to_keep <- c(col_to_keep, i)
        }
}
train_set_1 <- train_set[, col_to_keep]
num_col_1 <- ncol(train_set_1)
train_set_2 <- train_set_1[, 8:num_col_1]
num_col_2 <- ncol(train_set_2)
train_set_2[, num_col_2] <- as.factor(train_set_2[, num_col_2])
for (i in 1:(num_col_2 -1)) {
        train_set_2[, i] <- as.numeric(train_set_2[, i])
}
```

A substantial number of the columns were found to be incomplete, that is, have values only for a subset of the rows. For further analysis, the dataset was restricted to the 53 columns by elimnation of these incomplete columns as well as those that did not contain measurement information.

The next step in cleaning up the dataset, the data were examined for data points that were substantially outside the distribution for the corresponding variable. The data for each variable were normalized by subtracting the mean and dividing by the standard deviation. These normalized data were examined for data points that were more than a threshold, set at 20, times the standard deviation away from the mean.

```
thres <- 20.0
for (i in 1:(num_col_2 - 1)) {
mean_col <- mean(train_set_2[, i])
sd_col <- sd(train_set_2[, i])
        for (j in 1:(num_row - 1)) {
                if((abs(train_set_2[j, i] - mean_col) / sd_col) > thres) {
#                       print(j)
#                       print(train_set_2[j, i])
#                       print(mean_col)
#                       print(sd_col)
                }
        }
}
```

A single data point was found to lie outside the distribution for several variables and was removed from the dataset for further analysis.

```
train_set_2 <- train_set_2[-5373, ]
```

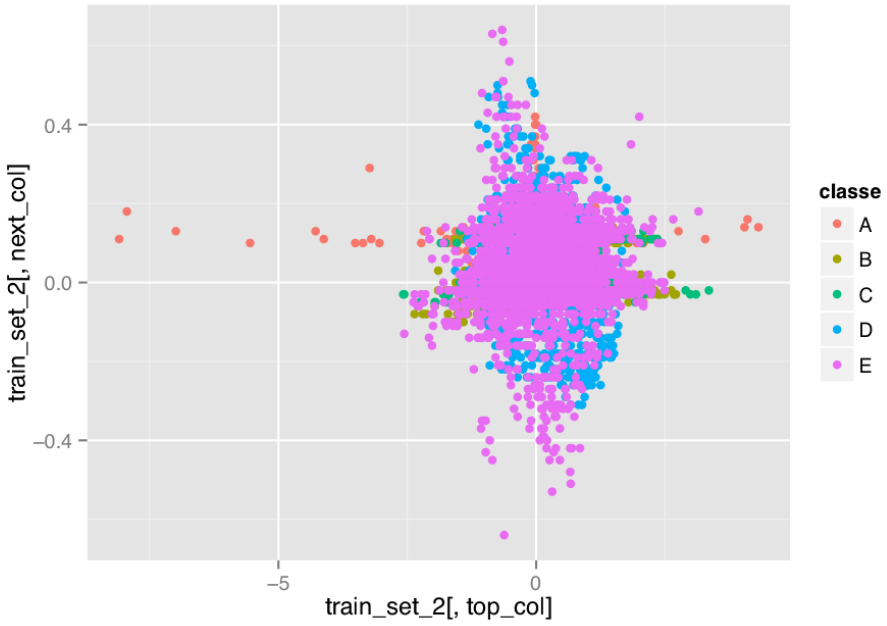
Data Analysis

```
reg_var <- numeric(length = (num_col_2 - 1))
for (i in 1:(num_col_2 - 1)) {
        reg_var[i] <- (max(train_set_2[, i]) - min(train_set_2[, i]))/ sd(train_set_2[, i])
```

```
}
top_var <- sort(reg_var)[(num_col_2 - 1)]
top_col <- which(reg_var == top_var)
next_var <- sort(reg_var)[(num_col_2 - 2)]
next_col <- which(reg_var == next_var)
```

The two variables with the largest normalized ranges were plotted against each other.

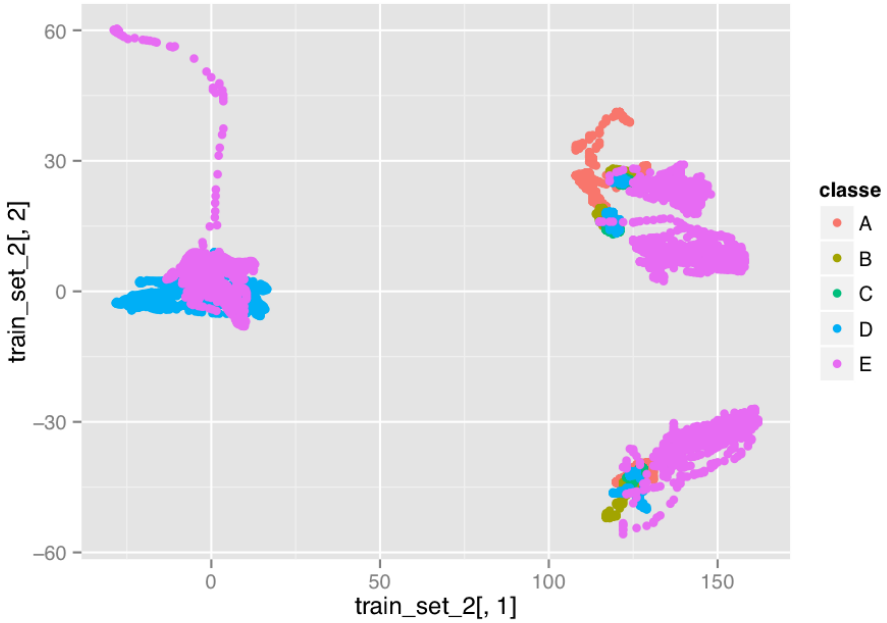
```
p_1 <- ggplot(train_set_2, aes(x = train_set_2[, top_col], y = train_set_2[, next_col], color = classe))
p_1 <- p_1 + geom_point()
p_1
```



These two variables appear to have approximately normal distributions.

For comparison, the first two variables in the dataset were plotted against each other.

```
p_2 <- ggplot(train_set_2, aes(x = train_set_2[, 1], y = train_set_2[, 2], color = classe))
p_2 <- p_2 + geom_point()
p_2
```



This plot revealed that the distributions for both variables were multimodal. This may limit the effectiveness of linear models.

Model Construction

A random forest model was selected as the initial choice because this method tends to be relatively accurate. The dataset was first divided into a training set with 75% of the data and cross-validation testing set with 25% of the data. A random forest model was these contructed using all of the variables remaininig in the dataset.

```
inTrain <- createDataPartition(train_set_2$classe, p = 0.75, list = FALSE)
train_set_to_use <- train_set_2[inTrain, ]
test_set_to_use <- train_set_2[-inTrain, ]
set.seed(0325)
modelFit_rf <- train(classe ~., data = train_set_to_use, method = "rf")
```

Results

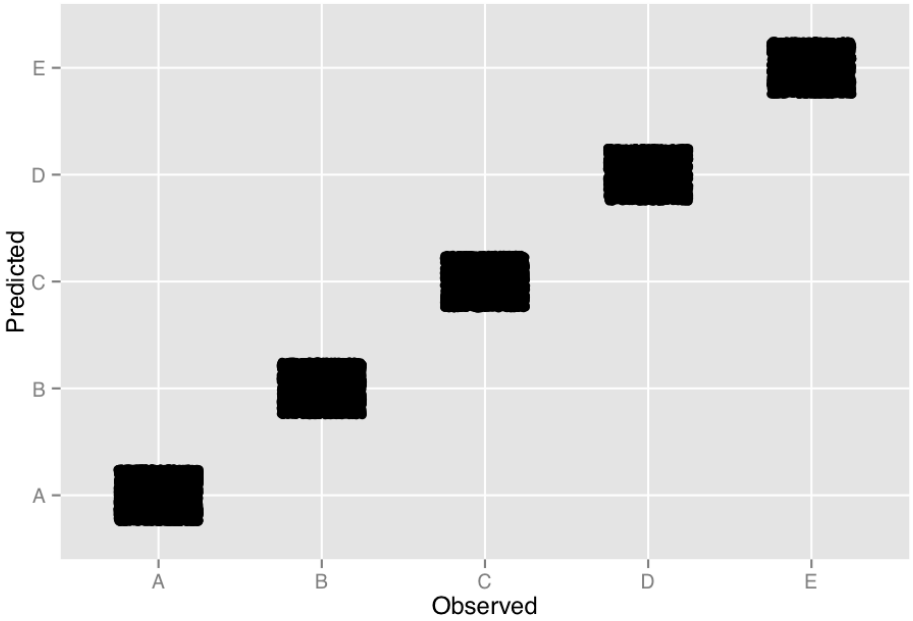
The results of this random forest model on the training dataset are shown below:

```
predictions_rf <- predict(modelFit_rf, train_set_to_use)
confusionMatrix(predictions_rf, train_set_to_use$classe)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
results_rf <- data.frame(train_set_to_use$classe)
results_rf <- cbind(results_rf, predictions_rf)
colnames(results_rf) <- c("Observed", "Predicted")
p_3 <- ggplot(results_rf, aes(x = Observed, y = Predicted))
p_3 <- p_3 + geom_jitter(position = position_jitter(width = 0.25, height = 0.25))
p_3
```



The random forest model was 100% accurate on the training set. This is highly encouraging although this outcome does suggest the possibility of overfitting.

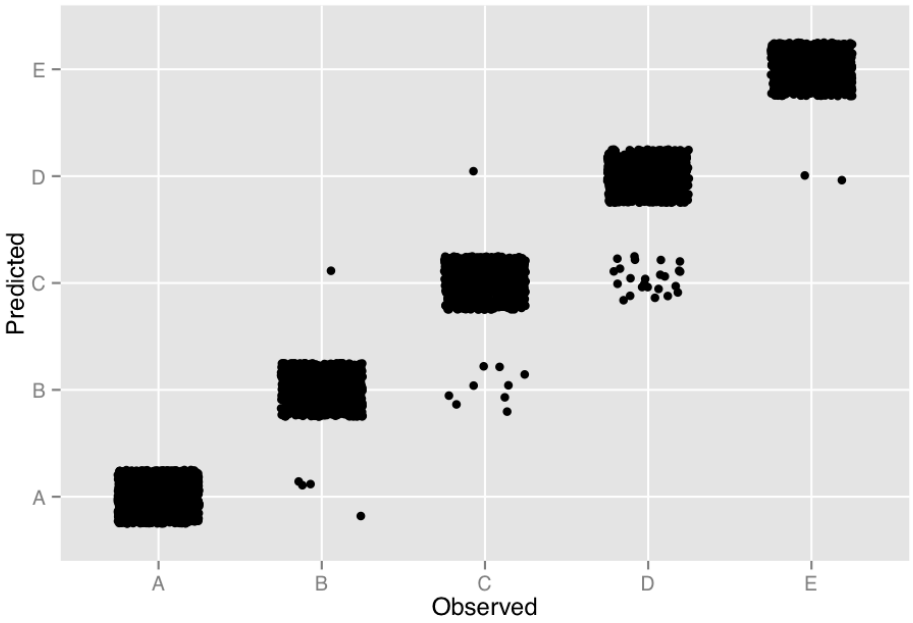
The same random forest model was applied to the test set for the purpose of cross-validation. The results are shown below:

```
predictions_rf_test <- predict(modelFit_rf, test_set_to_use)
confusionMatrix(predictions_rf_test, test_set_to_use$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    4    0    0    0
##          B    0  944    9    0    0
##          C    0    1  845   24    0
##          D    0    0    1  780    2
##          E    0    0    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.9916
##                  95% CI : (0.9887, 0.994)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9894
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9883   0.9701   0.9978
## Specificity            0.9989   0.9977   0.9938   0.9993   1.0000
## Pos Pred Value         0.9971   0.9906   0.9713   0.9962   1.0000
## Neg Pred Value         1.0000   0.9987   0.9975   0.9942   0.9995
## Prevalence             0.2843   0.1936   0.1744   0.1640   0.1838
## Detection Rate         0.2843   0.1925   0.1723   0.1591   0.1834
## Detection Prevalence   0.2851   0.1944   0.1774   0.1597   0.1834
## Balanced Accuracy      0.9994   0.9962   0.9911   0.9847   0.9989
```

```
results_rf_test <- data.frame(test_set_to_use$classe)
results_rf_test <- cbind(results_rf_test, predictions_rf_test)
colnames(results_rf_test) <- c("Observed", "Predicted")
p_4 <- ggplot(results_rf_test, aes(x = Observed, y = Predicted))
p_4 <- p_4 + geom_jitter(position = position_jitter(width = 0.25, height = 0.25))
p_4
```



The results were quite impressive with an overall accuracy of 99.4% and accuracies of over 99% in distinguishing all groups. Because of the success of this model, no additional models were pursued.

Application to Test Data

The test dataset was loaded and then processed in the same manner as the training set above. The random forest model was applied to the test dataset to produce predictions for the 20 data points.

```r
setwd("~/Data_Science_Course/jeremyberg.github.io")
test_set <- read.csv("pml-testing.csv", stringsAsFactors = FALSE, na.strings = c("NA", "#DIV/0!", ""))
test_set_1 <- test_set[, col_to_keep]
test_set_2 <- test_set_1[, 8:num_col_1]
test_set_2[, num_col_2] <- as.factor(test_set_2[, num_col_2])
for (i in 1:(num_col_2 -1)) {
        test_set_2[, i] <- as.numeric(test_set_2[, i])
}
```

```r
predictions_rf_test <- predict(modelFit_rf, test_set_2)
print(predictions_rf_test)
```

```
##  [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
```