Practical Machine Learning - Coursera Project

Dwight Kruger

October 13, 2015

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.

The goal of this project is to build a prediction model which predicts the manner in which the participants did the exercise. The data for this project was generously provided by http://groupware.les.inf.puc-rio.br/har.

Background

Although people regularly quantify *how much* of a particular activity they do, they rarely quantify *how well* they do it. In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict how well they did the excercise. Training data was obtained from subjects who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Our model will predict which of the 5 classes each subject belongs to.

More information is available from the website at: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). A data dictionary can be found at http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf. The training set contains a variable called *classe* whose values are defined as follows:

- A - Exactly according to specification.
- B - Throwing the Elbow to the front.
- C - Lifting the Dumbbell only halfway.
- D - Lowering the Dumbbell only halfway.
- E - Throwing the Hips to the front.

The training data is from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data is from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Exploratory Analysis

Browsing the training data and the data dictionary it is clear that some data will not be useful when building models (such as the subject's name, row identifier, etc.). First we will preprocess the training data and remove clealy irrelevant data such as summary rows as well as columns for which all data is NA, the row identifier, the subject's name, date/time values and window identifiers.

```r
# Load the data.
if (!file.exists("./data"))
{
  dir.create("./data")

  trainingDataUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  testDataUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(trainingDataUrl, destfile="./data/trainingData.csv", method="auto")
  download.file(testDataUrl, destfile="./data/testData.csv", method="auto")
}

trainingData = read.csv("./data/trainingData.csv", na.strings=c("", "NA", "NULL"))
testData = read.csv("./data/testData.csv", na.strings=c("", "NA", "NULL"))

# Remove irrelevant rows and columns.
trainingData <- trainingData[trainingData$new_window != "yes",] # Ignore summary rows
trainingData <- trainingData[,colSums(is.na(trainingData)) < nrow(trainingData)]  # Remove NA columns
trainingData <- select(trainingData, -X)                         # Remove X column
trainingData <- select(trainingData, -raw_timestamp_part_1)      # Remove raw_timestamp_part_1 column
trainingData <- select(trainingData, -raw_timestamp_part_2)      # Remove raw_timestamp_part_2 column
trainingData <- select(trainingData, -cvtd_timestamp)            # Remove cvtd_timestamp column
trainingData <- select(trainingData, -new_window)                # Remove new_window column
trainingData <- select(trainingData, -num_window)                # Remove num_window column
trainingData <- select(trainingData, -user_name)                 # Remove user_name column
```

Next, we will to look for variables in the training set which have near zero variance, i.e. those which have basically the same value for all measurements for all subjects. If we find any, we will want to also remove those variables as well from our training set.

```
countZeroVar <- nearZeroVar(trainingData)
if (any(countZeroVar))
  message("Some variables have near zero variance.") else
  message("None of the variables have a near zero variance.")
```

```
## None of the variables have a near zero variance.
```

Split the training data into a training set (75%) and a test set (25%)

```
inTrain <- createDataPartition(y=trainingData$classe, p=0.75, list=FALSE)
training <- trainingData[inTrain,]
testing <- trainingData[-inTrain,]
```

We may have variables which are highly correlated with each other and hence may lead to overfitting. A visualization of the correlation between each pair of variables is given in figure 1 in the appendix figure 1.

Build the Random Forest model

Using Principal Component Analysis (PCA) we will choose the most useful predictors and build a random forest model.

```
prComp <- preProcess(training[-53], method="pca", thresh = 0.99)     # PCA to find useful predictors
trainPC <- predict(prComp, training[-53])
rfModel <- randomForest(training$classe ~ ., data=trainPC, importance=TRUE) # Build the random forest
```

Run the test set through the prediction model and create a confusion matrix to examine how well the random forest predicted the correct outcomes. This will dermine the cross-validation error.

```
predTestData <- predict(rfModel,  predict(prComp, testing[,-53]))
cmTestData <- confusionMatrix(testing$classe, predTestData)
cmTestData$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1363    2    0    2    0
##          B   18  903    7    1    0
##          C    0   11  823    4    0
##          D    1    2   37  744    2
##          E    0    3    9    2  868
```

Calculate the accuracy of our results, by counting the number of times the model correctly predicted *classe* (accuracy), and the out of sample error. I would be satistifed with a out of sample error of 5% or less.

```
accuracy <- cmTestData$overall[1]
classeAccuracy <- accuracy[[1]]
outOfSampleError <- 1 - classeAccuracy
```

The random forest model predicted *classe* 97.9% of the time correctly (accuracy) and the out of sample error is 1-accuracy = 2.1%

Predict *classe* of the subjects in the test data

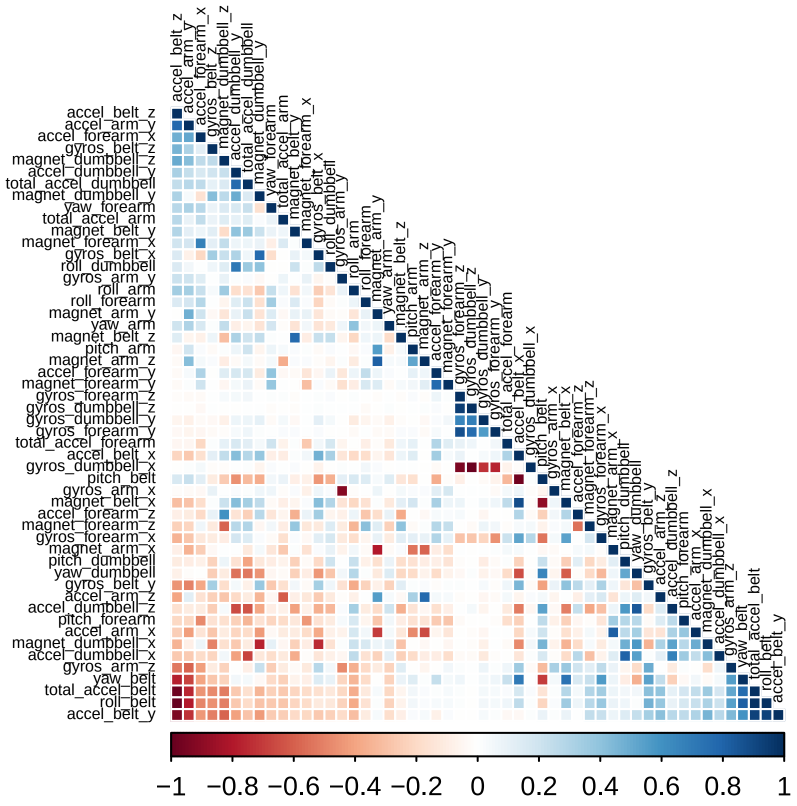
Finally predict the *classe* of the test subjects using the random forest model.

```
classe_prediction <- predict(rfModel, predict(prComp, testData))
result <- cbind(testData[2], classe_prediction)
```

```
##    user_name classe_prediction                             Description
## 1      pedro                 B       Throwing the elbows to the front
## 2     jeremy                 A Exactly according to the specification
## 3     jeremy                 B       Throwing the elbows to the front
## 4     adelmo                 A Exactly according to the specification
## 5     eurico                 A Exactly according to the specification
## 6     jeremy                 E         Throwing the hips to the front
## 7     jeremy                 D     Lowering the dumbbell only halfway
## 8     jeremy                 B       Throwing the elbows to the front
## 9   carlitos                 A Exactly according to the specification
## 10   charles                 A Exactly according to the specification
## 11  carlitos                 B       Throwing the elbows to the front
## 12    jeremy                 C      Lifting the dumbbell only halfway
## 13    eurico                 B       Throwing the elbows to the front
## 14    jeremy                 A Exactly according to the specification
## 15    jeremy                 E         Throwing the hips to the front
## 16    eurico                 E         Throwing the hips to the front
## 17     pedro                 A Exactly according to the specification
## 18  carlitos                 B       Throwing the elbows to the front
## 19     pedro                 B       Throwing the elbows to the front
## 20    eurico                 B       Throwing the elbows to the front
```

Appendix

Figure 1: Correlations between pairs of variables in the training set.



The darker colors indicate features which are correlated with each other, where blue is a positive correlation and red is a negative correlation. Clearly not all of the variables are necessary since some are highly correlated with each other.