Exercise Prediction

Dan Wright

April 24, 2016

Background

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 – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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. In this project, your goal will be to use data 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 (see the section on the Weight Lifting Exercise Dataset).

Data

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 data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Peer Review Portion Your submission for the Peer Review portion should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to $<$ 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

Course Project Prediction Quiz Portion Apply your machine learning algorithm to the 20 test cases available in the test data above and submit your predictions in appropriate format to the Course Project Prediction Quiz for automated grading.

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

Reading and cleaning the data

```
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(corrplot)
```

We begin by reading in the data and finding the dimensions of the dataset. We find the training dataset is a 19622 by 160 matrix while the test dataset is a 20 by 160 dataset.

```
test <- read.csv("pml-testing.csv")
train <- read.csv("pml-training.csv")
dim(test); dim(train)
```

```
## [1]  20 160
```

```
## [1] 19622   160
```

We also want to remove the columns with N/A values or timestamp columns, which don't add any predictive results.

```
train <- train[, colSums(is.na(train)) == 0]
test <- test[, colSums(is.na(test)) == 0]
classe <- train$classe
trainRemove <- grepl("^X|timestamp|window", names(train))
train <- train[, !trainRemove]
train <- train[, sapply(train, is.numeric)]
train$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(test))
test <- test[, !testRemove]
test <- test[, sapply(test, is.numeric)]
```

We then split the training set to allow us to create a model and test the model on unseen data.

```
set.seed(224466)
inTrain <- createDataPartition(train$classe, p=0.60, list=F)
trainData <- train[inTrain, ]
testData <- train[-inTrain, ]
```

Building the model

We now build a random forest model on our training dataset. We use 250 trees, however we receive a notification that the model only included 27 because that gave the best results.

```
control <- trainControl(method="cv", 5)
model <- train(classe ~ ., data=trainData, method="rf", trControl=control, ntree=250)
model
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9420, 9421, 9421, 9421
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9870920  0.9836692  0.003698028  0.004677484
##   27    0.9887055  0.9857128  0.003887983  0.004917052
##   52    0.9755428  0.9690605  0.004905322  0.006203196
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

We test the model on the training dataset and find the accuracy is .9936, which surprisingly, is higher than the training dataset, which had an accuracy of .9887.

```
predict <- predict(model, testData)
confusionMatrix(testData$classe, predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    4    0    0    0
##          B    6 1505    7    0    0
##          C    0    4 1361    3    0
##          D    1    0   17 1267    1
##          E    0    2    2    3 1435
##
## Overall Statistics
##
##                Accuracy : 0.9936
##                  95% CI : (0.9916, 0.9953)
##     No Information Rate : 0.2849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9919
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9969   0.9934   0.9813   0.9953   0.9993
## Specificity            0.9993   0.9979   0.9989   0.9971   0.9989
## Pos Pred Value         0.9982   0.9914   0.9949   0.9852   0.9951
## Neg Pred Value         0.9988   0.9984   0.9960   0.9991   0.9998
## Prevalence             0.2849   0.1931   0.1768   0.1622   0.1830
## Detection Rate         0.2840   0.1918   0.1735   0.1615   0.1829
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9981   0.9957   0.9901   0.9962   0.9991
```

We then test the accuracy and out of sample error rate and confirm an out of sample error rate of .0064.

```
accuracy <- postResample(predict, testData$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9936273 0.9919390
```

```
oose <- 1 - as.numeric(confusionMatrix(testData$classe, predict)$overall[1])
oose
```

```
## [1] 0.006372674
```

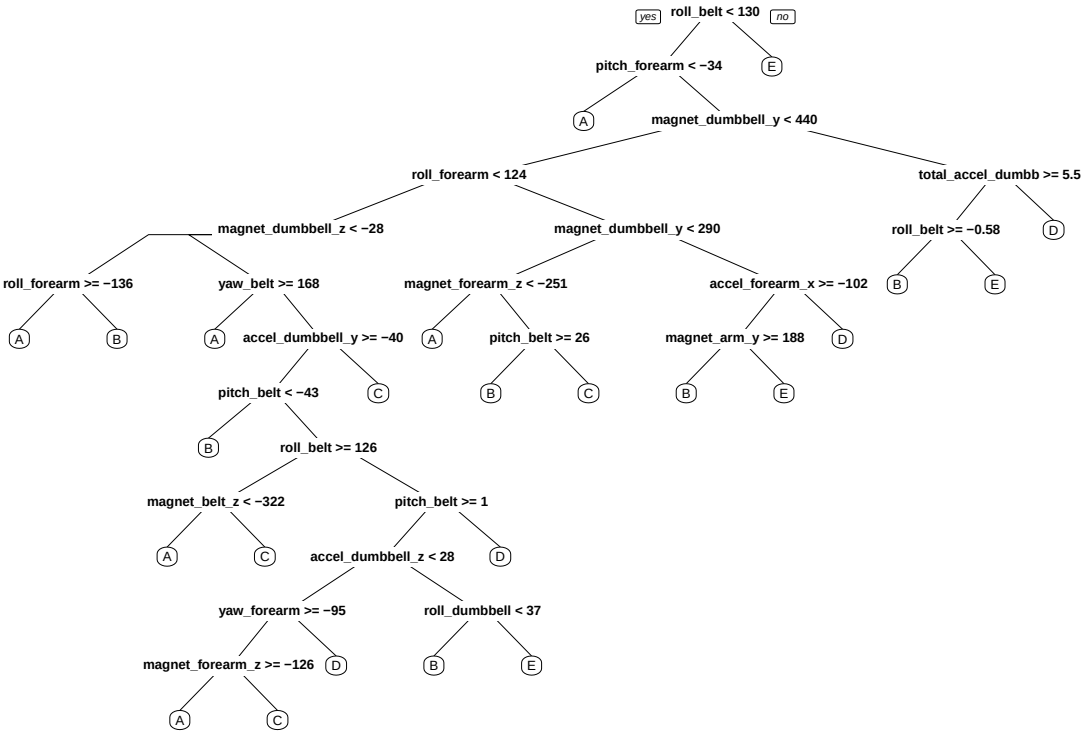
We now test the results on the final dataset of 20 rows.

```
final <- predict(model, test[, -length(names(test))])
final
```

```
##  [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
```

We plot a sample decision tree to understand how the random forest might be splitting the data.

```
treeModel <- rpart(classe ~ ., data=train, method="class")
prp(treeModel)
```


yes
roll_belt < 130
no
pitch_forearm < −34
E
A
magnet_dumbbell_y < 440
roll_forearm < 124
total_accel_dumbb >= 5.5
magnet_dumbbell_z < −28
magnet_dumbbell_y < 290
roll_belt >= −0.58
D
roll_forearm >= −136
yaw_belt >= 168
magnet_forearm_z < −251
accel_forearm_x >= −102
B
E
A
B
A
accel_dumbbell_y >= −40
A
pitch_belt >= 26
magnet_arm_y >= 188
D
pitch_belt < −43
C
B
C
B
E
B
roll_belt >= 126
magnet_belt_z < −322
pitch_belt >= 1
A
C
accel_dumbbell_z < 28
D
yaw_forearm >= −95
roll_dumbbell < 37
magnet_forearm_z >= −126
D
B
E
A
C