Weight Lifting Prediction

John Jennings

Thursday, March 12, 2015

Executive Summary

A Human Activity Recognition study was performed on six weigh lifters. Three aspects of that study pertain to qualitative activity recognition and were investigated: the problem of specifying correct execution, the automatic and robust detection of execution mistakes, and how to provide feedback on the quality of execution to the user.[1]

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).[1]

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. The dataset contains five classes (sitting-down, standing-up, standing, walking, and sitting). These five classes will be the focus of our continued investigation.[1]

The goal of this report 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.

Full documentation of the intitial study are cited in the sources section of this report.

Setup the Workspace and Download the Data

```
rm(list=ls()) # Clear the Workspace
setwd("C:/Users/John/Github/Practical Machine Learning") # Setup the Workspace
set.seed(100)
```

Setup the environment

```
library(caret)
library(randomForest)
library(rpart)
```

Load the required libraries

```
url.train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

# Download the training dataset.  This may take several minutes.  Skip if it already exists.
file.train <- "./pml-training.csv"
        if (!file.exists(file.train)) {
                download.file(url.train, destfile = file.train, mode = "wb")
                }

url.test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Download the training dataset.  This may take several minutes.  Skip if it already exists.
file.test <- "./pml-testing.csv"
        if (!file.exists(file.test)) {
                download.file(url.test, destfile = file.test, mode = "wb")
                }
```

Download the data

Explore and Clean the Data

```
# Read the training and test datasets, this may take several minutes
data.train <- read.csv("pml-training.csv", sep=",")
data.test <- read.csv("pml-testing.csv", sep=",")
```

Read the data

```
#
#Results suppressed using results='hide' in order to shorten the length of the report
#
str(data.train, list.len=1000) #Use list.len to view more than 99 variables
table(data.train$classe)  #How many observations for each class?
```

Exploring the data

```
# Read the training and test datasets, this may take several minutes
data.train <- read.csv("pml-training.csv", sep=",", na.strings=c("NA","","#DIV/0!"))
data.test <- read.csv("pml-testing.csv", sep=",")
```

Reload the data, Removing NAs

```
data.train.tidy <- data.train[,-c(1:7)]  #Remove unnecessary variables
data.train.tidy <- data.train.tidy[,colSums(is.na(data.train.tidy)) == 0]  #Remove all columns with zer
data.test.tidy <- data.test[,-c(1:7)]  #Remove unnecessary variables
data.test.tidy <- data.test.tidy[,colSums(is.na(data.test.tidy)) == 0]  #Remove all columns with zeros
```

Tidy up the data

```
inTrain <- createDataPartition(y=data.train$classe, p=0.75, list=FALSE)
training <- data.train.tidy[inTrain,]
testing <- data.train.tidy[-inTrain,]
dim(training)
```

Partition the training data

```
## [1] 14718    53
```

Create the Prediction Models

```
model.dt <- rpart(classe ~ ., data=training, method="class")
predict.dt <- predict(model.dt, testing, type = "class")
```

Decision Tree Model Prediction

```
model.rf <- randomForest(classe ~. , data=training, method="class")
predict.rf <- predict(model.rf, testing, type = "class")
```

Random Forest Model Prediction

Estimate the Error With Cross Validation NOTE: In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the run, as follows:

Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree.

Put each case left out in the construction of the kth tree down the kth tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.[2]

```
summary(model.rf)
```

```
##                 Length Class  Mode
## call               4  -none- call
## type               1  -none- character
## predicted      14718  factor numeric
## err.rate        3000  -none- numeric
## confusion         30  -none- numeric
## votes          73590  matrix numeric
## oob.times      14718  -none- numeric
## classes            5  -none- character
## importance        52  -none- numeric
## importanceSD       0  -none- NULL
## localImportance    0  -none- NULL
## proximity          0  -none- NULL
## ntree              1  -none- numeric
## mtry               1  -none- numeric
## forest            14  -none- list
## y              14718  factor numeric
## test               0  -none- NULL
## inbag              0  -none- NULL
## terms              3  terms  call
```

```
confusionMatrix(predict.dt, testing$classe)
```

Compare the Decision Tree model with the Random Forest model.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1279  191   22   90   31
##          B   47  525   85   43  150
##          C   39   87  683  120  115
##          D   20   85   65  517   66
##          E   10   61    0   34  539
##
## Overall Statistics
##
##                Accuracy : 0.7225
##                  95% CI : (0.7097, 0.735)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6471
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9168   0.5532   0.7988   0.6430   0.5982
## Specificity            0.9048   0.9178   0.9108   0.9424   0.9738
## Pos Pred Value         0.7929   0.6176   0.6542   0.6866   0.8370
```

```
## Neg Pred Value         0.9648   0.8954   0.9554   0.9309   0.9150
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2608   0.1071   0.1393   0.1054   0.1099
## Detection Prevalence   0.3289   0.1733   0.2129   0.1535   0.1313
## Balanced Accuracy      0.9108   0.7355   0.8548   0.7927   0.7860
```

```
confusionMatrix(predict.rf, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    4    0    0    0
##          B    0  944    3    0    0
##          C    0    1  851   11    3
##          D    0    0    1  793    1
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9953   0.9863   0.9956
## Specificity            0.9989   0.9992   0.9963   0.9995   1.0000
## Pos Pred Value         0.9971   0.9968   0.9827   0.9975   1.0000
## Neg Pred Value         1.0000   0.9987   0.9990   0.9973   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1735   0.1617   0.1829
## Detection Prevalence   0.2853   0.1931   0.1766   0.1621   0.1829
## Balanced Accuracy      0.9994   0.9970   0.9958   0.9929   0.9978
```

Summary of the Results The Random Forest model has a higher accuracy than the Decision Tree model (0.9951 vs. 0.7225). The Specificity and Sensitivity is also higher for the Random Forest model in all cases. This results in positive predictor values for Random Forest of 0.9971 (Class A), 0.9968 (Class B), 0.9827 (Class C), 0.9975 (Class D), and 1.0000 (Class E) vs. Decision Tree values of 0.7929 (Class A), 0.6176 (Class B), 0.6542 (Class C), 0.6866 (Class D), and 0.8370 (Class E), meaning that the Random Forest model will positively identify how the exercise was performed almost 100% of the time.

I will use the Random Forest model to proceed with the grading assignment.

Prediction for Grading Submission

```
predict.grade <- predict(model.rf, data.test, type="class")
predict.grade
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predict.grade)
```

Create Prediction Files

Source:

[1] Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3UAUtpobC

[2] Breiman, Leo and Cutler, Adele. Random Forests. Retrieved from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#ooberr