Machine Learning Weight Lifting Technique Prediction

Joy Flowers

Executive Summary

This analysis develops predictions as to whether bicep curls were performed correctly by men using accelerometers (sensors) in a similar method to how a Fitbit or Jawbone records data. The prediction using 75% of the training set with a random forest machine learning algorithm predicted within 97.5% accuracy on a validation set, and 90% accuracy on a test set. The other algorithms attempted had less favorable results. Once the model was modified using all training set observations (and more trees), the model predicted the Test set at 100% accuracy.

Introduction

Six men participated in a study regarding proper weight lifting techniques when doing bicep curls. Here is the website that gives an overview of the study: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). While using a Human Activity Recognition (HAR) device such as Fitbit, Jawbone or Nike FuelBand, they were asked to perform bicep curls in both correct and incorrect ways according to the following categories:

- Class A: exactly according to the specification (Right)
- Class B: throwing the elbows to the front (Wrong)
- Class C: lifting the dumbbell only halfway (Wrong)
- Class D: lowering the dumbbell only halfway (Wrong)
- Class E: throwing the hips to the front (Wrong)

With the sensor measurement data given from the training link https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv, the goal is to predict, using machine learning, which entries recorded a correctly performed bicep curl. The Test set data is found at https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Analysis of Data

```
library("caret")
library("dplyr")
library("knitr")
cache_modelFit = TRUE
cache_modFit2 = TRUE
```

```
#rm(list=ls())
set.seed(3433)
fileUrlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileUrlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
pmlTrain <- download.file(fileUrlTrain,destfile="training.csv",method="curl",mode="wb")
dateDownloaded <- date()
orig_training <- read.csv("training.csv",header=TRUE)
pmlTest <- download.file(fileUrlTest,destfile="test.csv",method="curl",mode="wb")
dateDownloaded <- date()
```

```
orig_test <- read.csv("test.csv",header=TRUE)
dim(orig_training)
```

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

The Class category is held in the *classe* variable. In the training set, there are 19622 observations and 160 variables. A quick glance at the variables using the View function shows that most data is categorized into roll, pitch, and yaw. Roll, pitch and yaw are used to define movement direction. For a great visual description of how roll, pitch, and yaw are used in airplane flight see http://howthingsfly.si.edu/flight-dynamics/roll-pitch-and-yaw. The definitions for roll, pitch and yaw are:

- Roll is rotation around the front-to-back axis.
- Pitch is rotation around the side-to-side axis.
- Yaw is rotation around the vertical axis.

There were four sensors put on the body to define motion. These were placed at the arm (bicep), belt (front center waist), forearm (wrist), and on the dumbbell itself, each measuring in the x,y, and z directions.

To identify the variables that do not contribute toward the prediction, the nearZeroVar function was used and those that do not contribute were further investigated.

```
ZCkTest <- nearZeroVar(orig_training, saveMetrics=TRUE)
ZeroVr <- subset(ZCkTest,nzv==TRUE)
```

According to the nearZeroVar results, there are many columns that could be eliminated such as anything with kurtosis, skewness and summary columns as well as username and times. There appear to be columns of both summary and detail information. Summary information appears to include:

- Kurtosis
- Skewness
- Maximum Value (Max)
- Minimum Value (Min)
- Amplitude
- Average Value (Avg)
- Standard Deviation (Stddev)
- Variance (Var)

Also, there appear to be rows of summary data when the variable new window = "yes". In order to get the best predictions, the data must be cleaned to keep only detailed data and eliminate summary data.

Preprocessing the Data

The training and test data will now be processed to remove the summary columns and rows while retaining the detail columns and rows.

First, eliminate summary rows:

```
traincleanup <- subset(orig_training,new_window!="yes")
testcleanup <- subset(orig_test,new_window!="yes")
```

Now, eliminate summary columns, so this leaves only detail columns beginning with *gyros*, *accel*, *magnet*, *roll*, *pitch*, *yaw*, and *total* plus the *classe* response variable for a total of 53 variables to keep.

```
train_clean <- select(traincleanup,starts_with("gyros"),starts_with("accel"),starts_with("magnet"),star
test_clean <- select(testcleanup,starts_with("gyros"),starts_with("accel"),starts_with("magnet"),starts
```

Now test for correlation of remaining variables - 38 variables are 80% correlated. So use Principal Component Analysis (PCA) and reduce to 25 Principal Components (PCs).

```
M <- abs(cor(train_clean[,-53]))
diag(M) <- 0
corr_matrix <- which(M > 0.8,arr.ind=TRUE)
head(corr_matrix,3)
```

```
##                  row col
## gyros_arm_y        5   4
## gyros_arm_x        4   5
## gyros_dumbbell_z   9   7
```

```
PCAnTrain <- preProcess(train_clean[,-53], method = "pca")
PCDShow <- PCAnTrain$rotation
```

Training the Data

Now to perform a preliminary test, for cross-validation using the hold-out method, split the training set into a training (75%) and validation (25%) set, and train the model using the PCs as predictors.

```
split_train <- createDataPartition(y=train_clean$classe,p=0.75,list=FALSE)
training <- train_clean[split_train,]
validat <- train_clean[-split_train,]
PCAnTrain <- preProcess(training[,-53], method = "pca",pcaComp=25)
trainpc <- predict(PCAnTrain,training[,-53])
modelFit <- train(training$classe ~ .,method="rf",data=trainpc,ntree=100)
modelFit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.78%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4054   19   14   14    3  0.01218324
## B   57 2680   44    2    6  0.03908211
## C    3   37 2432   40    2  0.03261734
## D    7    6   91 2251    6  0.04659043
## E    2   13   20   14 2597  0.01851852
```

Prediction and Evaluation

Now run the model on the validation set. The accuracy found on Validation set is 97.5%. Kappa is 96.8%. It takes about 15 min to train the model using 100 trees (and 25 minutes for 250 trees). With 250 trees, accuracy increased by only 0.5%. At 10 trees, accuracy decreased by almost 10%. Reducing the PCs from 25 to 5 decreased accuracy by nearly 15%, so the decision was to run with 100 trees and 25 PCs.

```
valpc <- predict(PCAnTrain,validat[,-53])
predictions_val <- predict(modelFit,valpc)
confusionMatrix(predictions_val,validat$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1358   17    5    3    2
##          B    6  891   13    1    4
##          C    1   20  812   37    3
##          D    1    0    8  742    4
##          E    1    1    0    3  869
##
## Overall Statistics
##
##                Accuracy : 0.9729
##                  95% CI : (0.9679, 0.9773)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9657
##  Mcnemar's Test P-Value : 0.0001076
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9934   0.9591   0.9690   0.9440   0.9853
## Specificity            0.9921   0.9938   0.9846   0.9968   0.9987
## Pos Pred Value         0.9805   0.9738   0.9301   0.9828   0.9943
## Neg Pred Value         0.9974   0.9902   0.9934   0.9891   0.9967
## Prevalence             0.2847   0.1935   0.1745   0.1637   0.1837
## Detection Rate         0.2828   0.1855   0.1691   0.1545   0.1810
## Detection Prevalence   0.2884   0.1905   0.1818   0.1572   0.1820
## Balanced Accuracy      0.9928   0.9764   0.9768   0.9704   0.9920
```

Now run the model on the test set. The accuracy found on 20 case Test set was only 18 out of 20 or 90% on first run. A subsequent run shown here has an accuracy of 19 out of 20 or 95%.

```
test <- test_clean
testpc <- predict(PCAnTrain,test[,-53])
predictions_test <- predict(modelFit,testpc)
predictions_test
```

```
##  [1] B A C A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

So now to get a better prediction, instead of using 75% of the training data, use all of the training set to predict the test set, using 500 trees and the model predicted 20 out of 20, which is 100% accuracy.

```r
modFit2 <- train(classe ~ .,method="rf",data=train_clean)
modFit2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.46%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5470    1    0    0    0 0.0001827819
## B   13 3700    5    0    0 0.0048413125
## C    0   18 3333    1    0 0.0056682578
## D    0    0   43 3102    2 0.0142993327
## E    0    0    0    5 3523 0.0014172336
```

```r
predict_test2 <- predict(modFit2,test_clean)
```

```r
predict_test2
```

```
##  [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
```

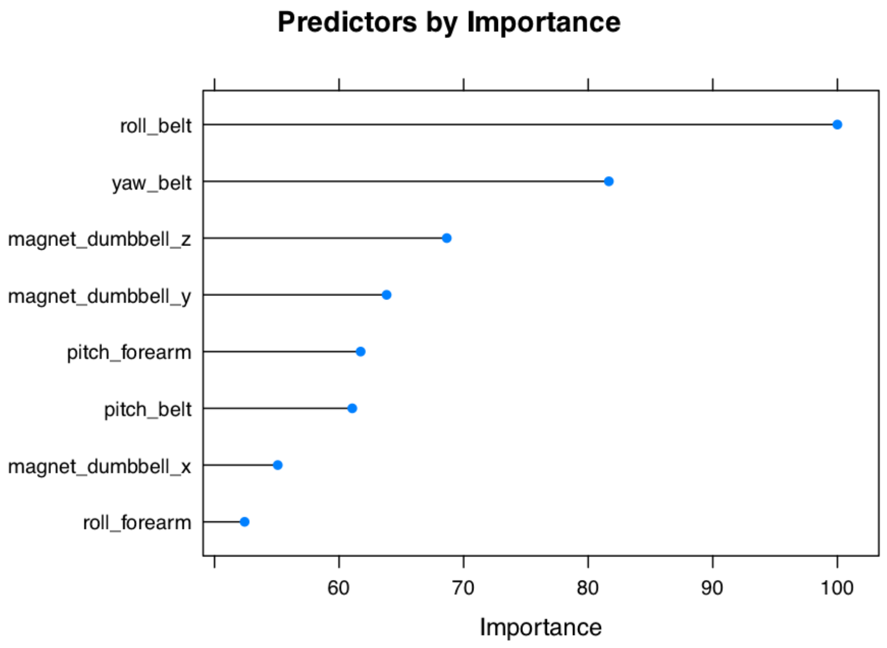
```r
missClass <- function(values,prediction){sum(prediction != values)/length(values)}
out_of_samp_err <- round(missClass(validat$classe, predictions_val),3)
```

The out of sample error misclassification for the validation set is 0.027. The out of sample error for the final Test set is zero.

```r
VarIm <- (varImp(modFit2))
Va <- data.frame(VarIm$importance)
VI_Importance <- abs(sort(-round(VarIm$importance$Overall,0)))
Predictor <- rownames(Va)[order(Va$Overall, decreasing=TRUE)]
VI <- data.frame(cbind(Predictor,VI_Importance))
head(VI,8)
```

```
##          Predictor VI_Importance
## 1        roll_belt           100
## 2         yaw_belt            82
## 3 magnet_dumbbell_z           69
## 4 magnet_dumbbell_y           64
## 5     pitch_forearm           62
## 6        pitch_belt           61
## 7 magnet_dumbbell_x           55
## 8      roll_forearm           52
```

```
plot(VarIm,main="Predictors by Importance",top=8)
```



Of the first eight variables that were most important in predicting the bicep technique, the roll, yaw, and pitch (discussed above) on the belt sensor were among the top six, which indicates that the belt sensor (and therefore movement from the waist) was the most important sensor in terms of prediction.

Other Models Attempted

There were several other models that were attempted such as this Learning Vector Quantization (lvq) algorithm below. It yielded 54.2% accuracy. The Linear Discriminate Analysis (lda) model yielded about 46% accuracy (yet it is noted that this data should not fit a linear model). Certain other models such as rpart, and svm took too much time to train - more than an hour each and so were abandoned.

```
learning vector quantizatiom like a neural network
control <- trainControl(method="repeatedcv", number=10, repeats=3)
modelLvq <- train(train$classe~., data=trainpc, method="lvq", trControl=control)
predictions_vallvq <- predict(modelLvq,valpc)
confusionMatrix(predictions_vallvq,validat$classe)
```

Credits

Data set creation is credited to: 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.

Conclusion

A random forest machine learning algorithm was able to predict with a high degree of accuracy (Validation Set 97% and Test Set 100%) whether the men in the study correctly performed bicep curls, given their sensor data.