Weight Lifting Exercises Activity Recognition

B. Tyson Dube

January 31, 2016

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 which allows to quantify how much of a particular activity has been done. However, the quality or way of how well a particular activity was performed is rarely evaluated.

Executive Summary

The goal of this project is to use a machine learning algorithm as part of an attempt to identify mistakes made in weight lifting exercises. The dataset used was made available the authors* of the 'Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements paper.

The unique and highly controlled circumstances around the collection of the data allow us to use machine learning to identify with compelling accuracy the mistakes made in weight lifting exercises.

Getting Started: Exploratory Analysis

First we load the data in a fully reproducible way.

Inital summary reporting shows that there is incomplete data that must be accounted for before moving forward. We remove these variables from the selected features.

```
naCol <- (colSums(is.na(trainRaw)) > 0)
str(trainRaw[, naCol], list.len=3)
```

```
## 'data.frame':    19622 obs. of  67 variables:
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

On further analysis we see that the dataset contains several mislabeled factor variables. These variables can also be removed as they do not add to our analysis.

```
facCol <- unlist(sapply(trainRaw[1,], is.factor))
facCol[length(facCol)] <- FALSE
str(trainRaw[,facCol], list.len=3)
```

```
## 'data.frame':    19622 obs. of  36 variables:
##  $ user_name              : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2
2 2 2 2 2 2 ...
##  $ cvtd_timestamp         : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9
9 9 9 9 9 9 ...
##  $ new_window             : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1
...
##   [list output truncated]
```

Indices and timestamps can also be removed.

```
remTag <- rep(F, ncol(trainRaw))
remTag[1:7] <- T
str(trainRaw[,remTag],list.len=3)
```

```
## 'data.frame':    19622 obs. of  7 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name           : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2
2 2 2 2 ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 13230
84232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##   [list output truncated]
```

Create the processed training and validation data.

```
train <- trainRaw[,!(naCol|facCol|remTag)]
predNames <- names(train)
predID <- grep("^classe", predNames, invert=T)
predNames <- predNames[predID]
```

```
suppressMessages(library(caret))
```

```
## Warning: package 'ggplot2' was built under R version 3.2.3
```

```
set.seed(1983)
trainMod <- createDataPartition(y=train$classe,p=0.8, list=F)
training <- train[trainMod,]
crossV <- train[-trainMod,]
```

Model Fitting

The data seems well suited for a Random Forest algorithm. This method should work well for this type of classification problem.

```
suppressMessages(library(randomForest))
rf <- randomForest(classe~., data=training)
```

Cross validation

We use a 2-fold cross validation.

```
cvP <- predict(rf,crossV[,predNames])
summaryCv <- confusionMatrix(crossV[, "classe"], cvP)
summaryCv
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    2  756    1    0    0
##          C    0    2  681    1    0
##          D    0    0    3  640    0
##          E    0    0    1    3  717
## 
## Overall Statistics
## 
##                Accuracy : 0.9967
##                  95% CI : (0.9943, 0.9982)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9958
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9974   0.9927   0.9938   1.0000
## Specificity            1.0000   0.9991   0.9991   0.9991   0.9988
## Pos Pred Value         1.0000   0.9960   0.9956   0.9953   0.9945
## Neg Pred Value         0.9993   0.9994   0.9985   0.9988   1.0000
## Prevalence             0.2850   0.1932   0.1749   0.1642   0.1828
## Detection Rate         0.2845   0.1927   0.1736   0.1631   0.1828
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9991   0.9982   0.9959   0.9964   0.9994
```

The confusion matrix show a 99.67% accuracy level.

Prediction

Using the Test cases we are able to further test the predictive value of our model.

```
result <- predict(rf, testRaw[, predNames])
result
```

```
##  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
```

Conclusion

We were able to achieve a high classification accuracy of 99.67% using a rather basic random forests method.

References

- 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.