Predicting Exercise Activity Quality from Activity Monitors

Ken Daniels

November 22, 2015

Executive Summary

Given the Groupware@Les Weight Lifting Exercises training dataset from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har), we seek to obtain a model that predicts the quality of excercise activity for new samples. After further subdividing the dataset into equal-sized training and testing datasets, we employ a random forest machine learning algorithm to obtain a model with an expected error rate no greater than 0.7%.

Analysis

We begin by loading the raw data into variables “trainRaw” and “test”:

```
trainRaw <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv", na.strings=c("NA",""))
testFull <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.c
sv", na.strings=c("NA",""))
```

We then transform “trainRaw” into variable “train” by exluding the first 6 incidental columns, then removing all columns for which fewer than half the rows contain values, leaving us with a training dataset having 19622 observations of 54 attributes each:

```
train<-trainRaw[,7:dim(trainRaw)[2]]
train <- train[,apply(!is.na(train),2,sum)>=(dim(train)[1])/2]
dim(train)
```

```
## [1] 19622    54
```

Now we further subdivide the original training set into two equally sized sets, trainSub and testSub:

```
library(AppliedPredictiveModeling)
```

```
library(caret)
```

```
set.seed(4810)
trainSubIndex <- createDataPartition(y=train$classe,p=0.5,list=FALSE)
trainSub <- train[trainSubIndex,]
testSub <- train[-trainSubIndex,]
dim(trainSub)
```

```
## [1] 9812   54
```

```
dim(testSub)
```

```
## [1] 9810   54
```

We use trainSub to develop our caret-based random forest (“rf”) model, employing 3-fold cross validation (“cv”):

```
library(caret)
rffit<-train(classe~.,data=trainSub,method="rf",trControl=trainControl(method="cv",numb
er=3),prox=TRUE,allowParallel=TRUE)
```

```
print(rffit)
```

```
## Random Forest
##
## 9812 samples
##   53 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 6542, 6540, 6542
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9853246  0.9814312  0.002798233  0.003543616
##   27    0.9924586  0.9904595  0.003894894  0.004927093
##   53    0.9861395  0.9824645  0.004284783  0.005418237
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
print(rffit$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE,      allowParallel
= TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.35%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 2790    0    0    0    0 0.000000000
## B   11 1886    1    1    0 0.006845708
## C    0    5 1704    2    0 0.004091175
## D    0    0    9 1599    0 0.005597015
## E    0    0    0    5 1799 0.002771619
```

This results in a model with a theoretical expected error rate ranging between 0% (for predicting variable A) and 0.7% (for predicting variable B). This aligns with the model accuracy of 0.9925 (mtry=27) with standard deviation of 0.00389. Note that we could have improved the accuracy of the model by increasing the k-fold value, but this could have led to overfitting and greater machine processing times.

We now verify our training-based model against our testSub dataset:

```
predictedClasse <- predict(rffit,newdata=testSub)
testClasse <- testSub$classe
numIncorrectPredictions <- length(testClasse[predictedClasse != testClasse])
numTestSamples <- length(testClasse)
missRate <- numIncorrectPredictions / numTestSamples
print(missRate)
```

```
## [1] 0.004587156
```

Thus, we obtain a miss rate of 0.4587156% when applying our random forest model against the test data subset. This falls within the theoretical error rate of no greater than 0.7% predicted by our model.