Machine Learning Project

JoeY

10/11/2014

Project Summary

The aim of this project is to predict whether a specific exercise was performed correctly, based on analysis of sensor data that has been collected while subjects performed that exercise correctly and with common errors. The present study finds that the random forest method from the R caret package performs quite well, using the model's default training options.

Below is a brief synopsis of the experiment design from the original research paper. The paper provides additional details about the instruments used to collect data, and controls on performance and monitoring.

> 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). Class A corresponds to the specifed execution of the exercise, while the other 4 classes correspond to common mistakes." source: http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf

Load libraries

```r
library(knitr)
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

Read in data

```r
if (!file.exists("pml-training.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
        destfile="pml-training.csv", method="curl")}
if (!file.exists("pml-testing.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
        destfile="pml-testing.csv", method="curl")}
pmlTrain <- read.csv("pml-training.csv")
pmlTest <- read.csv("pml-testing.csv")
```

Tidy up data

The data contain blank values and division by zero error codes, both of which are normalized to NA for this analysis.

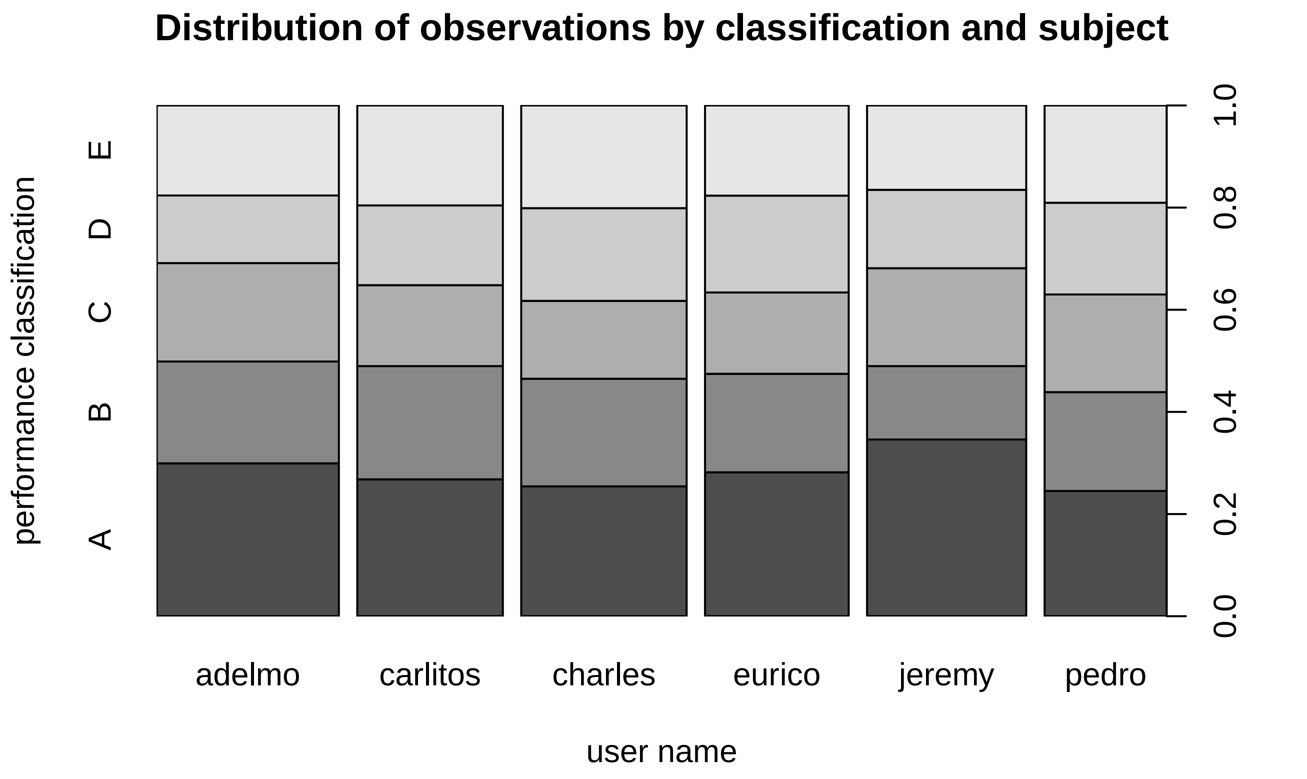
```r
pmlTrain[pmlTrain=="#DIV/0!"] <- NA
pmlTrain[pmlTrain==""] <- NA
dim(pmlTrain)
```

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

Exploratory analysis

The data capture performance by six subjects, identified by first name. The performance is scored in five classes, identified using the letters A through E.

```r
plot(pmlTrain$user_name,pmlTrain$classe, xlab="user name", ylab="performance classification",
     main="Distribution of observations by classification and subject")
```



Remove summary statistic columns

The data include summary statistics on certain lines. These appear to correspond the original researchers' arbitrary division of the data into time series windows. In the data, these lines are coded with the new_window flag, which is set to "yes" on lines with the summary statistics.

For this analysis, we will remove all of those summary statistics, as well as the observation sequence number and the subject (user_name). None of these variables would be meaningful in an application that would provide a gym user real-time feedback.

```r
# use grep to remove files by name pattern. We'll  do that to the test data later.
delcols <- grep("^(X|user|new|var|avg|stddev|kurtosis|skewness|max|min|amplitude)", colnames(pmlTrain))
pmlTrain <- pmlTrain[, -delcols]
dim(pmlTrain)
```

```
## [1] 19622    57
```

Partition data for training and testing

Next we'll need to generate generate a training and test set. The names of source sets were a little misleading, but `pml-testing.csv` did not contain the predicted (`classe`) variable, so it is not suitable for use as a testing dataset. Therefore, the observations in `pml-training.csv` are randomly partitioned for this purpose.

```
set.seed(101)
inTrain = createDataPartition(pmlTrain$classe, p=3/4)[[1]]
training = pmlTrain[inTrain,]
testing = pmlTrain[ -inTrain,]
```

Train a reasonable model and test

Training a random forest classifier on an Ubuntu notebook is very costly, but produces a 100% accuracy rate on the training data. A useful feature of the random forest method is that it takes care of cross-validation implicitly.

```
rfFit <- train(classe ~., data=training, method="rf")
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
## Loading required namespace: e1071
```

```
rfFit
```

```
## Random Forest
##
## 14718 samples
##    56 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    1         1      2e-03        2e-03
##   38    1         1      5e-04        7e-04
##   74    1         1      1e-03        2e-03
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 38.
```

One would expect the out of sample error to be a little larger, and it is. But it is still nearly perfect.

```
rfPred <- predict(rfFit, newdata=testing)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

The out of sample of sample error in this instance is 2.0392 × 10-4. A confusion matrix shows that we get only one error; one type of "bad repetition" incorrectly classified as another type of weight-lifting mistake. So we will accept this as a viable model.

```
confusionMatrix(rfPred,testing$classe)$table
```

```
## Loading required namespace: e1071
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    0    0
##          D    0    0    0  803    0
##          E    0    0    0    1  901
```

Conclusion

The random forest method in the caret package was computationally expensive to run in this instance, but produced an excellent result. If I did not have time to run it overnight, I might have opted to use a much smaller training set or modifications to the training options, accepting a lower accuracy. As it is, the classifier that results from this project should perform adequately in the intended application, assuming equivalent sensor hardware.

In implementing the application, I would suggest giving the user feedback at the end of each repetition of an exercise. The feedback would represent the majority of classifications of observations during the rep. An additional "end of repetition" marker would be needed for that application.

Generate predicted classes for submission.

```
testPred <- predict(rfFit, newdata=pmlTest[, - delcols])
testPred
```

```
##  [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
```