Exercise Prediction

Nathan E. Wendling

January 31, 2016

Introduction

Given our input data, the goal of this project was to predict the manner in which exercises were done from the performance data. To do this, machine learning was utilized and then analyzed for it's accuracy.

Libraries

The utilized libraries are as follows:

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

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

```r
suppressMessages(library(gbm))
suppressMessages(library(AppliedPredictiveModeling))
suppressMessages(library(forecast))
suppressMessages(library(ggplot2))
suppressMessages(library(randomForest))
```

Read in the data Reading in the data from the provided .csv files:

```r
datain <- read.csv("pml-training.csv", stringsAsFactors = FALSE)
validation <- read.csv("pml-testing.csv", stringsAsFactors = FALSE)
```

Eliminate Spurious Data Next, we look to reduce the number of variables we are computing over by looking at the percentage of the variable's record that is NA.

```r
nacols <- 0 #vector has to exist first
for(i in 1:length(colnames(datain))) {
   nacols[i] <- sum(is.na(datain[,i]))/length(datain[,i])
}
filtrain <- datain[,nacols<.9]
```

Due to the length of the output (160 lines), it is supressed in this report but 67 of the columns from the .csv were found to have over 90% of their data to be NA. Thus we save computing time by only computing over 93 variables.

Cut into train and test Now we take out available data in datain and create training and testing sets (as the classe-less pml-test.csv file serves as validation)

```r
inTrain <- createDataPartition(y=datain$classe, p=.8, list=FALSE)
trainset <- filtrain[inTrain,]
testset <- filtrain[-inTrain,]
dim(trainset); dim(testset)
```

```
## [1] 15699    93
## [1] 3923   93
```

Train the model Now we train several models on the TRAINING set in order to have something to predict with. We attempted to use the random forest method as for this project interpretability and computation times are not high proirities. However, in the course of doing the project, quicker methods were needed, but glm had problems with the data. Hence, gbm with a very small ntree.

```
fitRF <- train(classe ~ ., data=trainset, method="gbm", ntree=10)
```

Out of Sample Error Once we have the trained model, we see how well they predict the, "classe" on the testing data set.

```
predRF <- predict(fitRF, testing)
accuracy <- confusionMatrix(predRF, testing$classe)$overall[1]
accuracy <- accuracy*100 #human readable
```

Thus we get an 78.565 percent accuracy on the testing subset of the data. Not great, but acceptable for our purposes. If random forest would have been allowed to run, >85% would have been expected.

Validation We then predict on the 20 validation cases.

```
predict(fitRF, validation)
```

Conclusions

While a random forest would have computed the results better, due to the time constraints (the random forest ran for hours without results) and not using PCA analysis on the data, we managed to get 15 of the 20 validation problems correct, which should be acceptable for this report.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.