Movement Prediction with R Machine Learning

David Harter

11/16/2018

Background

In this project, we will use machine learning techniques in R to identify specfic bodily movements. The data used are measures of movement as gauged from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants.

Libraries

```r
library(caret)
```

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

Gathering, Cleaning, and Partitioning Data

The data comes from the original study, "Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements." For the assessment portion of this project, we will only download and partition the training set.

```r
fitbit <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"), header =
```

Taking a cursory look at this data set, we observe two things: The first 7 columns consist of names, timestamps and other flags; and many of the columns consist largely of "NA" readings. We redefine the data set by omitting these columns, searching for columns where 75 percent or more of the readings are "NA." We then partition into training and testing sets

```r
fitbit <- fitbit[, -c(1:7)]
badCols <- which(colSums(fitbit == "" | is.na(fitbit))/dim(fitbit)[1] > 0.75)
fitbit <- fitbit[, -badCols]

set.seed(2828)
bitCut <- createDataPartition(fitbit$classe, p = 0.8, list = FALSE)
training <- fitbit[bitCut, ]
testing <- fitbit[-bitCut, ]
```

Omitting columns of questionable data pares our training and testing sets from 160 columns down to just 53. Furthermore, it has the added benefit of getting rid of many columns that were classified as factors even though their form suggested they would be better classified as numeric. We are now ready to begin modeling the data.

Modeling

We are starting with a robust data set which has many different predictors and readings which are likely to fall in somewhat similar ranges. For the scope and content restraints of this report, and since time is not a factor, we will stick to the three methods in the caret package that are appropriate for such a set: gradient boosting, decision trees, and random forest. Bagging is not necessary, as the data set is not too small, and Naive Bayes is not appropriate, as we know that these measurements are parts of a physical movement and therefore dependent on each other. We will gauge the accuracy of each method and decide which one to use from those results.

For cross-validation, we will use the k-fold cross validation built into the caret library's train function. Because this data set is large, we will use 10-folds to achieve a high level of accuracy.

Since these methods often take a long time to run, we will simply comment on the results. We start with the gradient boosting method.

```r
set.seed(4325)
boostCtrl <- trainControl(method = "cv", number = 10)
boostFit <- train(classe ~ ., data = training, method = "gbm", trControl = boostCtrl)

boostPred <- predict(boostFit, newdata = testing)
boostMatrix <- confusionMatrix(testing$classe, boostPred)
boostMatrix$overall[1]
```

At 96.2 percent, we see a very high level of accuracy for the gradient boosting method with very few misclassifications.

Next we attempt the decision tree method:

```r
set.seed(856)
treeCtrl <- trainControl(method = "cv", number = 10)
treeFit <- train(classe ~ ., data = training, method = "rpart", trControl = treeCtrl)

treePred <- predict(treeFit, newdata = testing)
treeMatrix <- confusionMatrix(testing$classe, treePred)
treeMatrix$overall[1]
```

The decision tree yielded an accuracy of 50.4 percent, not very impressive at all. We can safely reject this method in favor of one of the other two.

Finally, we attempt the random forest method.

```r
set.seed(1233)
forestCtrl <- trainControl(method = "cv", number = 10)
forestFit <- train(classe ~ ., data = training, method = "rf", trControl = forestCtrl)

forestPred <- predict(forestFit, newdata = testing)
forestMatrix <- confusionMatrix(testing$classe, forestPred)
forestMatrix$overall[1]
```

We see that the random forest, though it took a long time to compute, yields the best accuracy of all at 99.5 percent. Out of 3923 samples, only 19 were classified incorrectly.

Testing the random forest method on the 20 samples from the test set provided in the assignment specifications, we find that all 20 samples were identified correctly. This further confirms the accuracy of the random forest approach.