Practical Machine Learning

Jeff Grady

9/6/2017

Overview

For the Practical Machine Learning course project, here we build a machine learning system to judge how well a participant executed a particular weightlifting exercise based on data collected from movement sensors.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. They have been very generous in allowing their data to be used for this kind of assignment.

Exploratory Data Analysis, Cleaning

```
library(parallel)
library(doParallel)
library(caret)
```

First, we download the training and testing data sets. Once downloaded, we parse and load them into memory.

```
training_filename <- "pml-training.csv"
testing_filename <- "pml-testing.csv"
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = trainin
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = testing_
training <- read.csv(training_filename)
testing <- read.csv(testing_filename)
```

In examining the data, for example, with the `whatis()` function from the `YaleToolkit` library, we find that many variables have substantial missing data:

```
> whatis(training)
           variable.name         type missing distinct.values precision
1                      X      numeric       0           19622     1e+00
2              user_name  pure factor       0               6        NA
...
18          max_roll_belt      numeric   19216             195     1e-01
19         max_picth_belt      numeric   19216              22     1e+00
```

We see here that for `max_roll_belt`, `max_picth_belt`, and several others, they're all missing **19216** values out of **19622** total observations. For other variables such as `kurtosis_roll_arm` were interpreted as **factors** when read from csv, and thus they have missing values, but as empty strings ("") instead of `NA`:

```
> sum(training$kurtosis_roll_arm == "")
[1] 19216
```

Our first task is to write a function to remove these variables with substantial missing data. This function `cleanData()` takes a `data.frame` as an argument, makes a copy of it, removes:

- Columns 1-7, which were not related to accelerometer or other movement data
- Any column where more than half of the values are `NA`
- Any column where more than half of the values are empty string ("")

…and then returns the resulting `data.frame`.

```r
cleanData <- function(myData){
    # trim columns that aren't accelerometer data
    output <- myData[,-c(1:7)]
    # stores column indexes for removal
    exclude <- c()
    totalRows <- nrow(output)
    # subtract 1, because the last column is 'classe'
    numCol <- ncol(output) - 1
    for (i in 1:numCol) {
        numNA <- sum(is.na(output[,i]))
        numBlank <- sum(output[,i] == "")
        # If the field has more than half of its data missing, exclude it.
        if ((numNA > (totalRows * 0.5)) |
            (numBlank > (totalRows * 0.5))) {
            exclude <- c(exclude, i)
        }
    }
    output <- output[,-exclude]
    output
}
```

Here we set up parallel processing to make our model training take less time to compute:

```r
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
```

Here, we run `cleanData()` and we set up **10**-fold cross-validation.

```r
set.seed(1234)
cleaned <- cleanData(training)
fitControl <- trainControl(method = "cv",
                           number = 10,
                           allowParallel = TRUE)
```

Training

Now we train using a **random forest** with our cross-validation parameters.

```r
modRF <- train(classe ~ ., method="rf",
               trControl=fitControl, data=cleaned)
```

And, we train using **boosted regression** and with our cross-validation parameters.

```r
modGBM <- train(classe ~ ., method="gbm",
                trControl=fitControl, data=cleaned)
```

Finally, we train using **linear discriminant analysis** and with our cross-validation parameters.

```r
modLDA <- train(classe ~ ., method="lda",
                trControl=fitControl, data=cleaned)
```

Cross-Validation

Note that we had the `train()` function do **10**-fold cross-validation for us when we passed in `fitControl` with parameters of `method = "cv"` and `number = 10`.

Model Accuracy and Error Estimation

Let's calculate our accuracy for each of the models. We'll take the mean of all the k-folds:

```
modRFAccuracy <- mean(modRF$resample$Accuracy)
modGBMAccuracy <- mean(modGBM$resample$Accuracy)
modLDAAccuracy <- mean(modLDA$resample$Accuracy)
totalAccuracy <- (modRFAccuracy + modGBMAccuracy + modLDAAccuracy)/3
combinedAccuracy <- choose(3, 2) * totalAccuracy**2 * (1 - totalAccuracy)**1 +
    choose(3, 3) * totalAccuracy ** 3 * (1 - totalAccuracy) ** 0
```

Now, we have:

- `modRFAccuracy`: **99.51%**
- `modGBMAccuracy`: **96.36%**
- `modLDAAccuracy`: **70.12%**

We can use the binomial theorem to calculate the probabilities our combined models:

$$\binom{3}{2} * totalAccuracy^2 * (1 - totalAccuracy)^1 + \binom{3}{3} * totalAccuracy^3 * (1 - totalAccuracy)^0$$

Predictions

Now that we're finished computing our models, let's use them to predict **`classe`** for our **`testing`** data.

```
myAnswersRF <- predict(modRF, testing)
myAnswersGBM <- predict(modGBM, testing)
myAnswersLDA <- predict(modLDA, testing)
myAnswers <- c()
for (i in 1:nrow(testing)) {
    if ((myAnswersRF[i] == myAnswersGBM[i]) ||
        (myAnswersRF[i] == myAnswersLDA[i])) {
        myAnswers <- c(myAnswers, myAnswersRF[i])
        next
    } else if (myAnswersGBM[i] == myAnswersLDA[i]) {
        myAnswers <- c(myAnswers, myAnswersGBM[i])
        next
    }
    myAnswers <- c(myAnswers, myAnswersRF[i])
}
# correctAnswers <- c("B", "A", "B", "A", "A", "E", "D", "B", "A", "A",
#                     "B", "C", "B", "A", "E", "E", "A", "B", "B", "B")
correctAnswers <- c(2, 1, 2, 1, 1, 5, 4, 2, 1, 1,
                    2, 3, 2, 1, 5, 5, 1, 2, 2, 2)
numCorrect <- sum(correctAnswers == myAnswers)
numTotal <- nrow(testing)
```

Predicting the results of the test data with our model, **numCorrect** is **20** out of **20** test cases, or **100**%.

```
stopCluster(cluster)
registerDoSEQ()
```