Weight Lifting Exercise Prediction Using Machine Learning

Isaac Lawrence

10/25/2014

Using devices such as *Jawbone Up*, *Nike FuelBand*, and *Fitbit* it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data processing

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

We load the training data into a data frame, then remove any columns with NA values, or non-numeric data:

```
training <-  data.frame(fread("pml-training.csv", na.strings=c("#DIV/0!", "")))
training$classe <- as.factor(training$classe)
training  <- training[,colSums(is.na(training)) == 0]
training <- training[,-c(1:7)]
trainingCols <- names(training)
```

We then create training and testing partitions, splitting on the *classe* variable, containing 60% and 40% of the data, respectively:

```
inTrain = createDataPartition(training$classe, p = 0.6)[[1]]
train = training[inTrain,]
test = training[-inTrain,]
```

Machine learning model

Since we don't have much domain background knowledge that would allow us to guess at the underlying model, and we have dozens of numeric variables to use in our prediction, it seems reasonable to try a Random Forest model. We create a Random Forest model to predict the *classe* variable using 50 trees and the numeric data from the training data set:

```
set.seed(123)
modFit <- randomForest(classe ~ .,train,ntree=50)
```

We will see in the following section that the out of sample error rate, as predicted by the testing data set is <1%, so this model seems sufficient.

Cross validation and out of sample error rate

We use our model to predict the *classe* variable for the records in the test data set. We then build a confusion matrix to asses the accuracy of the model:

```
test_predictions <- predict(modFit,test)
CM <- confusionMatrix(test$classe,test_predictions)
```

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

The model seems to work pretty well, as we can see in the table below, which compares predictions to actual values:

```
print(xtable(CM$table), type = 'html')
```

	A	B	C	D	E
A	2229	2	1	0	0
B	10	1502	6	0	0
C	1	7	1358	2	0
D	0	0	15	1270	1
E	0	0	2	1	1439

Furthermore, we can see that specicifity and sensitivity are 99% or greater for all classes:

```
print(xtable(CM$byClass), type = 'html')
```

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: A	1.00	1.00	1.00	1.00	0.29	0.28	0.28	1.00
Class: B	0.99	1.00	0.99	1.00	0.19	0.19	0.19	1.00
Class: C	0.98	1.00	0.99	1.00	0.18	0.17	0.17	0.99
Class:								

D	1.00	1.00	0.99	1.00	0.16	0.16	0.16	1.00
Class: E	1.00	1.00	1.00	1.00	0.18	0.18	0.18	1.00

We see that the accuracy is >99% when our model is applied to the test data set, so we would expect the out of sample error rate to be <1%.

```
CM$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull 
##         0.9939         0.9923         0.9919         0.9955         0.2855 
## AccuracyPValue  McnemarPValue 
##         0.0000            NaN 
```