Project 1 Machine Learning

Kevin Kientz

June 11, 2016

Overview

The goal of this project is to predict the manner in which six subjects performed an exercise. The prediction is based on data collected from accelerometers on the belt, forearm, arm, and dumbell of the participants.

Using “Human Activity Recognition” data in which participants used an accelerometer while performing a dumbbell exercise, I downloaded a training dataset which I used to develop an appropriate prediction algorithm. That prediction algorithm was then used on a separate test dataset to test the accuracy of the prediction algorithm I used.

The dataset is based on six young healthy male participants aged 20-28 who 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). The accelerometer measured activity while performing these exercises and recorded the class (specification) of the exercise in the “classe” variable of the dataset.

Loading and cleaning the data

Using only the predictors that would be relevant to the project, and removing predictors that have NA values, the dataset is ready for a machine learning algorithm.

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

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

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
trainingDataset <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", na.strings=c("NA","#DIV/0!",""))
testingDataset <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", na.strings=c("NA","#DIV/0!",""))

# only use columns that would be predictors ["belt", "arm", "dumbbell", "forearm"]
trainingData <- trainingDataset[8:160]
testingData <- testingDataset[8:160]

# remove columns with na's -- this removes roughly 100 predictors
trainingDataCleaned <- trainingData[, colSums(is.na(trainingData)) == 0]
```

Building a prediction algorithm using the Random Forest method

First, I split the training data set into a training and testing (i.e., validation) partitions. Doing this will allow me to test a prediction algorithm and obtain out-of-sample errors.

```
set.seed(4162)
inTrain <- createDataPartition(trainingDataCleaned$classe, p = 0.75, list = FALSE)
train <- trainingDataCleaned[inTrain,]
test <- trainingDataCleaned[-inTrain,]
```

Then, I used the random forest methodology with a centering and scaling preprocessing, and then did a cross validation technique with 8 folds and stored the model in the “rfmodel” variable. Using the predict function, I stored the model’s predictions in a “test_predictions” factor variable which allowed me to compare the in-sample true values in the validation test dataset with the predicted values. This was done with a Confusion Matrix.

We see that the random forest methodology I employed was highly accurate (over 99%), meaning that the out-of sample error rate (on the validation test set with 25% of the data) was under 1%. This implies that the expected out-of-sample error rate will be less than 1%. The model turned out to be perfectly accurate at choosing Class A when the full exercise was performed correctly (100% sensitivity for Class A). The model was also perfectly accurate when predicting Class E – that the exercise was performed while throwing the hips out (100% specificity for Class E).

```
rf_model <- train(train$classe ~ ., method="rf", data=train, preProcess=c("center",
 "scale"), trControl=trainControl(method = "cv", number = 8))
test_predictions <- predict(rf_model, newdata=test)
print(confusionMatrix(test_predictions, test$classe), digits=4)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395   10    0    0    0
##          B    0  939    5    0    0
##          C    0    0  843   11    2
##          D    0    0    7  793    2
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9896, 0.9947)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9860   0.9863   0.9956
## Specificity            0.9972   0.9987   0.9968   0.9978   1.0000
## Pos Pred Value         0.9929   0.9947   0.9848   0.9888   1.0000
## Neg Pred Value         1.0000   0.9975   0.9970   0.9973   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1719   0.1617   0.1829
## Detection Prevalence   0.2865   0.1925   0.1746   0.1635   0.1829
## Balanced Accuracy      0.9986   0.9941   0.9914   0.9921   0.9978
```

Testing model predictions against out-of-sample data

Finally, I used the model to predict the type of exercize conducted with out-of-sample test data, which was provided for 20 different test cases.

```
print(predict(rf_model,newdata=testingData))
```

```
##  [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
```