Maching Learning Assignment

A Zabrodski

August 21, 2015

Exercise Prediction Using Machine Learning

All data was obtained from the following source:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3jVXbcYoF (http://groupware.les.inf.puc-rio.br/har#ixzz3jVXbcYoF)

The data set is comprised of measurements taken on multiple subjects as they performed a specific exercise, bicep curls. Each subject was instructed to perform the exercise properly, or intentionally wrong with a specific fault. How the exercise was performed is indicated by the classe variable in the data set.

We start by downloading the training data and the test data, then importing the training data into R. Then inspect the training data.

```
url_train = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url_train, "training_data.csv", method = "curl", quiet = TRUE)
train_data <- read.csv("training_data.csv", na.strings=c("NA","#DIV/0!", ""))
download.file(url_test, "test_data.csv", method = "curl", quiet = TRUE)
testing <- read.csv("test_data.csv", na.strings=c("NA","#DIV/0!", ""))
dim(train_data)
```

```
## [1] 19622   160
```

The training set consists of 19622 rows in 160 columns. We are interested in predicting the classe variable. We will also pre-process the data, by removing any instances that have NAs and any columns that do not contain relevant information.

```
train_data = train_data[,colSums(is.na(train_data)) == 0]
testing = testing[,colSums(is.na(testing)) == 0]
train_data = train_data[,-c(1:7)]
testing = testing[,-c(1:7)]
summary(train_data$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Which is a factor variable with 5 levels, with more “A” levels. Since all of the classes are over 3000, if doing convenitonal model validation you could use a smaller proportion of the training data since most samples will contain a representative amount of each class. For our purposes, we will be using a random forest model that will perform cross validation. However, to get a sense of expected accuracy on the test set we will still partition the training data.

We will use the caret package to partition the training data into two sets, training and validation. This way we can test on an unbiased sample of the training set which will give us expected model accuracy. We will use a 65/35 split because we have a large data set with every class having numerous instances.

```
library(caret)
```

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

```
inTrain = createDataPartition(y = train_data$classe, p = 0.65, list = FALSE)
train = train_data[inTrain,]
validation = train_data[-inTrain,]
```

We will now train the model using random forest since it is highly accurate. Unfortunately the algorithm takes a while to run and is not very interpretable. The default is 25 fold cross validation, but we will reduce this to 5 fold for the interest of speed and limit the number of trees to 100. Using a random forest model requires the randomForest package.

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
model = train(classe ~., data = train, method = "rf", trControl=trainControl(numbe
r = 5), ntree = 100)
```

Now we will examine our expected accuracy on the validation training set, and compare it to how our model performed on the training set to get a sense of the level of over fitting.

```
predictions <- predict(model$finalModel, newdata = validation)
confusionMatrix(predictions, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1949    9    0    0    0
##          B    2 1314    8    0    0
##          C    2    4 1187   19    2
##          D    0    1    2 1103    0
##          E    0    0    0    3 1260
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9901, 0.9943)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9904
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9980   0.9895   0.9916   0.9804   0.9984
## Specificity            0.9982   0.9982   0.9952   0.9995   0.9995
## Pos Pred Value         0.9954   0.9924   0.9778   0.9973   0.9976
## Neg Pred Value         0.9992   0.9975   0.9982   0.9962   0.9996
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2839   0.1914   0.1729   0.1607   0.1835
## Detection Prevalence   0.2852   0.1929   0.1768   0.1611   0.1840
## Balanced Accuracy      0.9981   0.9938   0.9934   0.9900   0.9989
```

Even limiting the cross validation and number of trees, we still end up with an overall accuracy of 99.05%. Not bad at all. So the expected out of sample error rate is 1%.

```
model$time$everything[3]/60
```

```
##  elapsed
## 6.116067
```

With the reduced accuracy in the interest of speed, the model still stook almost 6 minutes on a 2015 Mac Book Pro.

With this model, we can now test on the training set and submit for grading.