Practical Machine Learning Project

Lim Wei Ping

March 21, 2015

Introduction

The aim of this project is to build a model to predict the manner in which 6 participants perform the barbell lifts. Twenty test cases are provided for prediction.

The data used in this project is from the Weight Lifting Exercise dataset (http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises). In the experiment, the 6 particpants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Readings from 4 sensors on their belt, forearm, arm, and dumbell were recorded.

Data Exploration and Cleaning

A set of training data consisting of 19622 records and 160 variables. The variable "classe"" represent the manner in which the participants lift the dumbell. It is a factor consisting of 5 levels - "A", "B", "C", "D" and "E".

There are 38 variables associated with each of the 4 sensors (belt, forearm, amr and dumbell) that record readings such as the roll, pitch, yaw, acceleration, gyro. There are also 7 variables that are not related to signals collected from the on-body sensors. We observe that there are many variables with multiple NAs or empty records.

```
library(dplyr)
library(caret)

data_train <- read.csv("pml-training.csv", header = TRUE)
str(data_train)
```

The variables not from the sensors are the index, name of participants, timestamps etc, confined within the first 7 variables of the dataset. These variables are removed.

```
## Remove non-sensor variables
data_train <- select(data_train,-c(1:7))
```

If the number of NA or empty records for a certain variable is reasonable, we can potentially impute the missing records for example, by using the K nearest neighbour algorithm introduced in the course. However we found that all the variables with NA/empty records have very high proportions of NA/empty records - about 98% of the total records. In this case, all variables with NA/empty records are discarded.

```
## Compute the number of NA/empty records for each variable
narecords <- sapply(data_train, function(x){sum(is.na(x) | x == "")})
narecords <- narecords[narecords > 0]

## Remove variables with NA/empty records
data_train <- select(data_train, -one_of(names(narecords)))
```

Excluding the variable "classe", we are left with 52 variables for prediction - the same 13 variables from each sensor on readings for the roll, pitch, yaw, total acceleration as well as reading for the X, Y and Z directions for the gyroscope, accelerometer and magnetometer.

Next, we subset 30% of the data for cross-validation later.

```
set.seed(101)
inTrain <- createDataPartition(y = data_train$classe, p = 0.7, list = FALSE)
data_testing <- data_train[-inTrain, ]
data_training <- data_train[inTrain, ]
```

Building Prediction Model Using Random Forest

Random Forest is introduced in the class as a highly accurate prediction model. Hence, we will start with building a Random Forest model using the training set. Preliminary testing using the default settings took a very long time, exceeding 12 hours. We therefore use reduce the number of folds used in cross validation to 4 to reduce the time taken to train the model.

We note that several variables at dispropotionately more important than others. This means that we could run Random Forest again with reduced number of features.

```
model_rf <- train(classe ~ ., method="rf", trControl=trainControl(method = "cv", number = 4), data = da

impt <- varImp(model_rf)$importance
```

Testing the accuracy of the model against the test set, we note that the model obtained a high out-of-sample accuracy of 99.35% (i.e an out-of-sample error of 0.65%)

```
result_testing <- predict(model_rf, data_testing)
confusionMatrix(data_testing$classe, result_testing)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    2    1    0    1
##          B    8 1125    6    0    0
##          C    0    2 1022    2    0
##          D    0    0    5  956    3
##          E    0    0    2    6 1074
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9911, 0.9954)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9965   0.9865   0.9917   0.9963
## Specificity            0.9990   0.9971   0.9992   0.9984   0.9983
## Pos Pred Value         0.9976   0.9877   0.9961   0.9917   0.9926
## Neg Pred Value         0.9981   0.9992   0.9971   0.9984   0.9992
## Prevalence             0.2851   0.1918   0.1760   0.1638   0.1832
## Detection Rate         0.2838   0.1912   0.1737   0.1624   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9971   0.9968   0.9928   0.9950   0.9973
```

Comparing Against a Random Forest Model with Reduced Number of Features

In practice, it is desirable to reduce the number of features used into the prediction model for simplicity and also to lower cost. We can try to build another Random Forest modesl using only variables with importance more than 10.

```
imptvariables <- mutate(impt, variable = rownames(impt)) %>%
        filter(Overall > 10)

data_training_reduced <- select(data_training, one_of(imptvariables$variable), matches("classe"))
```

The out-of-sample accuracy is slightly lower than the model built using 52 variables but is still very high at 98.64% (i.e out-sample-error of 1.36%).

```
model_rf_reduced <- train(classe ~ ., method="rf", trControl=trainControl(method = "cv", number = 4), da

result_testing_reduced <- predict(model_rf_reduced, data_testing)
confusionMatrix(data_testing$classe, result_testing_reduced)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1666    6    0    1    1
##          B   14 1107   16    1    1
##          C    0   11 1011    4    0
##          D    0    0   12  950    2
##          E    0    0    2    9 1071
##
## Overall Statistics
##
##                Accuracy : 0.9864
##                  95% CI : (0.9831, 0.9892)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9828
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9917   0.9849   0.9712   0.9845   0.9963
## Specificity            0.9981   0.9933   0.9969   0.9972   0.9977
## Pos Pred Value         0.9952   0.9719   0.9854   0.9855   0.9898
## Neg Pred Value         0.9967   0.9964   0.9938   0.9970   0.9992
## Prevalence             0.2855   0.1910   0.1769   0.1640   0.1827
## Detection Rate         0.2831   0.1881   0.1718   0.1614   0.1820
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9949   0.9891   0.9840   0.9908   0.9970
```

Model Selection to Predict the 20 Test Cases

As accuracy of the model is the sole consideration in this project, we will choose to run the 20 test cases using the first Random Forest model with 52 features. The results of the prediction are:

```
data_test <- read.csv("pml-testing.csv", header = TRUE)
predict(model_rf, data_test)
```

```
##  [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
```