Practical Machine Learning Course Project

Liu Yi Hung

2015/9/24

Introduction

This paper is aim to predict personal activity, the data is from devices such as *Jawbone Up*, *Nike FuelBand*, and *Fitbit*. More informations about data can be found at http://groupware.les.inf.puc-rio.br/har. The data is downloaded from here, training data and testing data, soucre: http://groupware.les.inf.puc-rio.br/har.

In this project, I'll use variables to predict the outcome, which is the *classe* variable in the data. The algorithm I used is *random forest*, more informations about random forest can be found at Random forests]. Basically, the main analysis take the advantage of the *caret* and *randomForest* packages in R.

Why I choose this algorithm is because the advantage of random forest, these features is copied from (Random Forests: Leo Breiman and Adele Cutler)[https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm].

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

Download and Load the Data

First, download and load the data from source in to R.

```
if (!file.exists("pml-training.csv")) download.file("https://d396qusza40orc.cloudfront.net/predmachlearn
    "pml-training.csv", method = "curl")
if (!file.exists("pml-testing.csv")) download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/
    "pml-testing.csv", method = "curl")
train <- read.csv("pml-training.csv", na.strings = c("#DIV/0!", "", "NA"))
test <- read.csv("pml-testing.csv", na.strings = c("#DIV/0!", "", "NA"))
```

Check the dimensions of two dataset

```r
dim(train)
```

```
[1] 19622   160
```

```r
dim(test)
```

```
[1]  20 160
```

Model Building

Split the data

In order to further cross validation, I'll split the **train** data by `p = 0.75` using the `createDataPartition()` function in the *caret* package.

```r
library(caret)
set.seed(555)
inTrain <- createDataPartition(train$classe, p = 0.75, list = FALSE)
training <- train[inTrain, ]
testing <- train[-inTrain, ]
```

Check the dimensions

```r
dim(training)
```

```
[1] 14718   160
```

```r
dim(testing)
```

```
[1] 4904  160
```

Preprocessing

There are many variables have near zero variance, which can cause a poor model, these can be removed using `nearZeroVar()` function, use `saveMetrics = T` to see the results.

```r
# to see the results, uncomment this line nearZeroVar(training, saveMetrics = T)

# removing variables
nsv <- nearZeroVar(training)
trainingSelected <- training[, -nsv]
dim(trainingSelected)
```

```
[1] 14718   126
```

```r
library(dplyr)
trainingSelected <- select(trainingSelected, -(X:num_window))
dim(trainingSelected)
```

```
[1] 14718   120
```

Some variables have many NAs inside, removing these variables.

```r
# to see these variables, uncomment this line colSums(is.na(trainingSelected))

# remove these variables
trainingSelectedRmna <- trainingSelected[, colSums(is.na(trainingSelected)) == 0]
dim(trainingSelectedRmna)
```

```
[1] 14718    53
```

Fit the model

Use the function `randomForest()`, which takes data after preprocessing `data = trainingSelectedRmna` and `method = "class"`.

```r
library(randomForest)
RFmodel <- randomForest(classe ~ ., data = trainingSelectedRmna, method = "class")
RFmodel
```

```

Call:
 randomForest(formula = classe ~ ., data = trainingSelectedRmna,      method = "class")
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 0.42%
Confusion matrix:
     A    B    C    D    E  class.error
A 4184    1    0    0    0 0.0002389486
B   11 2832    5    0    0 0.0056179775
C    0    9 2557    1    0 0.0038955980
D    0    0   26 2384    2 0.0116086235
E    0    0    0    7 2699 0.0025868441
```

Cross Validation

Use the data **testing** splited before to perform cross validation. The process is based on the prediction on the **testing** data, and calculate the accuracy between model and testing data using `confusionMatrix()` funciton.

```r
prediction <- predict(RFmodel, testing, type = "class")
cm <- confusionMatrix(prediction, testing$classe)
cm
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 1394    7    0    0    0
         B    0  940    1    0    0
         C    0    2  853    6    0
         D    0    0    1  798    0
         E    1    0    0    0  901

Overall Statistics

               Accuracy : 0.9963
                 95% CI : (0.9942, 0.9978)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9954
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9993   0.9905   0.9977   0.9925   1.0000
Specificity            0.9980   0.9997   0.9980   0.9998   0.9998
Pos Pred Value         0.9950   0.9989   0.9907   0.9987   0.9989
Neg Pred Value         0.9997   0.9977   0.9995   0.9985   1.0000
Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
Detection Rate         0.2843   0.1917   0.1739   0.1627   0.1837
Detection Prevalence   0.2857   0.1919   0.1756   0.1629   0.1839
Balanced Accuracy      0.9986   0.9951   0.9978   0.9961   0.9999
```

The accuracy is really high, which is 0.9963295.

Out-of-sample error rate

The out-of-sample error rate can be calculated using the following command.

```r
outOfSample <- sum(prediction != testing$classe)/length(testing$classe)
outOfSample
```

```
[1] 0.003670473
```

Prediction

Now, use the model I built to predict the original **test** data. The results is shown below in the array format for 20 data.

```r
predictTest <- predict(RFmodel, test, type = "class")
predictTest
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 
 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
```

Generate Submit Answers

This is only a chunk of codes to generate the files to submit to Coursra.

```
pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE, col.names = FALSE)
    }
}
pml_write_files(predictTest)
```