practicalmachinelearning

Ian Chua

11/13/2019

Background

This document aims to explore the use of machine learning techniques to classify the weight lifting exercise (WLE) a person is doing, based on the various measurements placed at different parts of the body while the exercise is being performed.

Data loading

Over here, we do a preliminary exploration after loading the data, and found that there are many NAs and many blanks. It is also noted that the blanks and NAs largely congregate by columns, not by rows.

```
training <- read.csv(file = "pml-training.csv",as.is = TRUE)
testing <- read.csv(file = "pml-testing.csv", as.is = TRUE)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

```
library(e1071)
dim(training)
```

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

```
sum(is.na(training))
```

```
## [1] 1287472
```

```
dim(testing)
```

```
## [1]  20 160
```

Data cleaning and preprocessing

As mentioned, the blanks and NAs are usually by a whole column, not rows. Hence we remove the columns which have a huge number of blanks and NAs. Then, the training dataset is split into two, to reserve some data for cross-validation.

```r
training <- training[,colSums(is.na(training))==0]
training <- training[,colSums(training=="")==0]
training <- training[,-(1:7)]
training$classe <- as.factor(training$classe)
inTrain <- createDataPartition(training$classe,p=0.7)[[1]]
traindata <- training[inTrain,]
valdata <- training[-inTrain,]
```

Simple decision tree

Next, a simple decision tree is fitted to the data we have, in order to predict the "classe", which indicates the WLE. This model is then used to predict the classe of the validation dataset. A confusion matrix is then contructed to determine the accuracy of the model.

```r
modfittree <- rpart(classe~.,data=traindata)
predtree <- predict(modfittree,newdata=valdata,type="class")
confusionMatrix(predtree,valdata$classe)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1528  171   21   56   12
##          B   48  672   85   86   95
##          C   50  167  831  143  126
##          D   18   79   61  612   53
##          E   30   50   28   67  796
##
## Overall Statistics
##
##                Accuracy : 0.7543
##                  95% CI : (0.7431, 0.7652)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6885
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9128   0.5900   0.8099   0.6349   0.7357
## Specificity            0.9383   0.9338   0.9000   0.9571   0.9636
## Pos Pred Value         0.8546   0.6815   0.6310   0.7436   0.8198
## Neg Pred Value         0.9644   0.9047   0.9573   0.9305   0.9418
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2596   0.1142   0.1412   0.1040   0.1353
## Detection Prevalence   0.3038   0.1675   0.2238   0.1398   0.1650
## Balanced Accuracy      0.9255   0.7619   0.8550   0.7960   0.8496
```

The accuracy of the model on the validation set is about 75%. While it is quite high, it is not high enough for use in our prediction, as we would like an accuracy of at least 80% to predict the testing dataset.

Random forest

Next, a random forest model is fitted to the training data. Similar methods are used to evaluate the accuracy of the model.

```r
modfitrf <- randomForest(classe~.,data=traindata,ntree=200,mtry=5)
predrf <- predict(modfitrf,newdata=valdata,type="class")
confusionMatrix(predrf,valdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    0 1131   10    0    0
##          C    0    4 1016   10    0
##          D    0    0    0  954    3
##          E    1    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9930   0.9903   0.9896   0.9972
## Specificity            0.9991   0.9979   0.9971   0.9994   0.9998
## Pos Pred Value         0.9976   0.9912   0.9864   0.9969   0.9991
## Neg Pred Value         0.9998   0.9983   0.9979   0.9980   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1922   0.1726   0.1621   0.1833
## Detection Prevalence   0.2850   0.1939   0.1750   0.1626   0.1835
## Balanced Accuracy      0.9992   0.9954   0.9937   0.9945   0.9985
```

Now, we can see that the model accuracy is much better, at about 99%. Hence, we will use the random forest model to predict on the testing dataset.

```r
predict(modfitrf,newdata=testing,type="class")
```

```
##  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
```