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.

```r
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
```

```r
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
```

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

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

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

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

```r
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.

No standardisation was carried out on the dataset, as we will be using decision trees and random forest as our classifier model. These methods are insensitive to the order of magnitude of the different variables.

```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 1492  186   21   73   51
##          B   51  611   74   77   93
##          C   37  154  841  124  110
##          D   57   77   62  615   69
##          E   37  111   28   75  759
##
## Overall Statistics
##
##                Accuracy : 0.7337
##                  95% CI : (0.7222, 0.745)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6623
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8913   0.5364   0.8197   0.6380   0.7015
## Specificity            0.9214   0.9378   0.9125   0.9461   0.9477
## Pos Pred Value         0.8184   0.6744   0.6643   0.6989   0.7515
## Neg Pred Value         0.9552   0.8940   0.9599   0.9303   0.9337
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2535   0.1038   0.1429   0.1045   0.1290
## Detection Prevalence   0.3098   0.1540   0.2151   0.1495   0.1716
## Balanced Accuracy      0.9063   0.7371   0.8661   0.7921   0.8246
```

The accuracy of the model on the validation set is about 75%, which means that we can expect the out-of-sample error to be about 25%. Although the accuracy 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.

```
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    8    0    0    0
##          B    1 1130    7    0    0
##          C    0    1 1015    8    0
##          D    0    0    4  956    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9929, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9921   0.9893   0.9917   1.0000
## Specificity            0.9981   0.9983   0.9981   0.9992   1.0000
## Pos Pred Value         0.9952   0.9930   0.9912   0.9958   1.0000
## Neg Pred Value         0.9998   0.9981   0.9977   0.9984   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1920   0.1725   0.1624   0.1839
## Detection Prevalence   0.2856   0.1934   0.1740   0.1631   0.1839
## Balanced Accuracy      0.9988   0.9952   0.9937   0.9954   1.0000
```

Now, we can see that the model accuracy is much better, at about 99%, meaning the out-of-sample error of this model is only 1%. Hence, we will use the random forest model to predict on the testing dataset.

```
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
```