Practical Machine Learning Project

Guangsheng Liang

July 25, 2014

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Date Preprocessing

1. Download the data

Now there are two csv file in our directory.

```
dir('~/Desktop/project package')
```

```
## [1] "testing.csv"  "training.csv"
```

2. Load the data and require packages

```
setwd('~/Desktop/project package')
training <- read.csv("training.csv", na.strings = c("#DIV/0!",""))
testing <- read.csv("testing.csv", na.strings = c("#DIV/0!",""))
library(caret)
```

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

Now the training and testing file are in the global enviornment.

```
ls()
```

```
## [1] "metadata" "testing"  "training"
```

3. Train data partition

We cut off the training data into two parts: 60% of the data for training, and 40% of the data for testing our model.

```
dim(train); dim(test)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Let's see how many predictors are there in the data.

```
dim(train)
```

```
## [1] 11776   160
```

4. Cleaning the data

As we seen, there are too many variables in the data, which will cause overfitting and slow down the speed of machine learning.
Thus, we need to find out which variables should be exclude from our model. Simply screening data, we find there are variables that are just for description and some simple statistical measurements. The description variables do no help to the model fiting, and the statistical measurements highly correlate with other original measurements, which will cause confounding. So We want to exclude all these variables from our data.

```
dim(train.pre)
```

```
## [1] 11776    53
```

We now lower down our predictors to only 52 ('classe' is our ourcome).

5. Lower down predictors

We now have 52 predictors, but that is not less enough. To keep lowering down the predictor number, We can run 'nearZeroVar' to find out variables that have few unique values relative to the data. We set a threshold 10% to cut off.

```
dim(train.tar)
```

```
## [1] 11776    17
```

```
names(train.tar)
```

```
##  [1] "pitch_belt"       "yaw_belt"         "roll_arm"
##  [4] "pitch_arm"        "yaw_arm"          "magnet_arm_x"
##  [7] "magnet_arm_z"     "roll_dumbbell"    "pitch_dumbbell"
## [10] "yaw_dumbbell"     "roll_forearm"     "pitch_forearm"
## [13] "yaw_forearm"      "magnet_forearm_x" "magnet_forearm_y"
## [16] "magnet_forearm_z" "classe"
```

Now we have 16 variables, and I think it is good enough to use them to fit into the model.

6. Model fitting

Random forest has the highest accuarcy and the longest calculation time among the methods we learned. Since we have lowered down the predictors to 16, it won't cost too much time and we can get a highest precdiction.

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.1.1
```

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

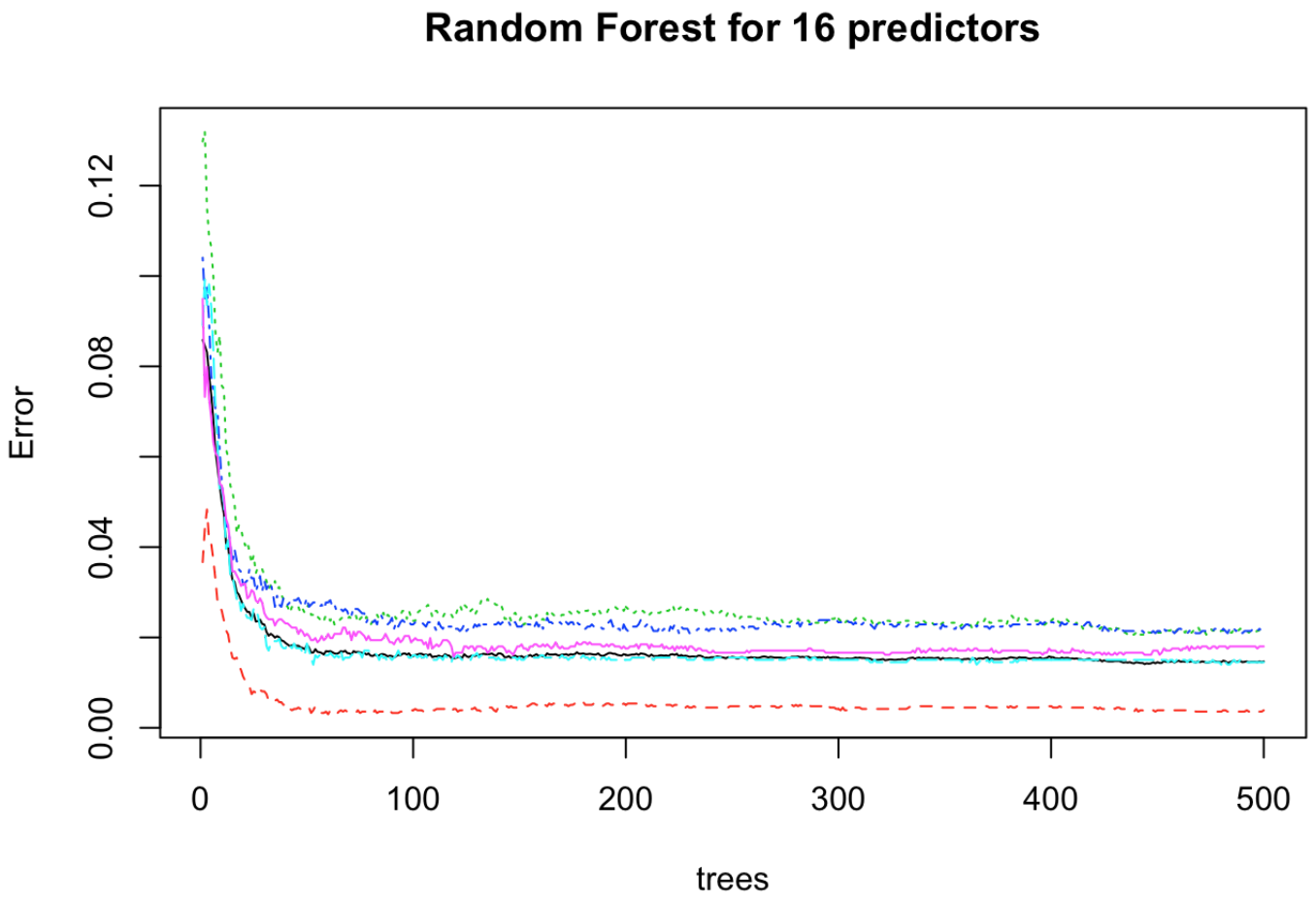
```
fit.nf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 1.47%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3335    9    2    1    1    0.003883
## B   18 2231   27    1    2    0.021062
## C    0   20 2009   20    5    0.021908
## D    2    4   18 1902    4    0.014508
## E    1   12    8   18 2126    0.018014
```

```
varImp(fit.nf$finalModel)
```

```
##                   Overall
## pitch_belt         1322.3
## yaw_belt           2075.6
## roll_arm            479.0
## pitch_arm           198.6
## yaw_arm             300.2
## magnet_arm_x        286.4
## magnet_arm_z        258.6
## roll_dumbbell       637.9
## pitch_dumbbell      313.4
## yaw_dumbbell        536.5
## roll_forearm        759.3
## pitch_forearm       963.0
## yaw_forearm         218.3
## magnet_forearm_x    248.0
## magnet_forearm_y    255.6
## magnet_forearm_z    457.4
```

```
plot(fit.nf$finalModel, main = "Random Forest for 16 predictors")
```



7. Test our model

We do the same screening for our testing data, and test the model prediction accuarcy.

```
testmodel
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2225    7    0    0    0
##          B   21 1484    9    0    4
##          C    0    6 1351   10    1
##          D    5    2   15 1264    0
##          E    0    9    7    7 1419
##
## Overall Statistics
##
##                Accuracy : 0.987
##                  95% CI : (0.984, 0.989)
##     No Information Rate : 0.287
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.983
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.988    0.984    0.978    0.987    0.996
## Specificity             0.999    0.995    0.997    0.997    0.996
## Pos Pred Value          0.997    0.978    0.988    0.983    0.984
## Neg Pred Value          0.995    0.996    0.995    0.997    0.999
## Prevalence              0.287    0.192    0.176    0.163    0.181
## Detection Rate          0.284    0.189    0.172    0.161    0.181
## Detection Prevalence    0.284    0.193    0.174    0.164    0.184
## Balanced Accuracy       0.994    0.989    0.987    0.992    0.996
```

The test model shows that our model has a 98.7% accuarcy to predict the classe, which is fairly high enough.

Summary

We use the random forest to build up a model, which has a 98% accuarcy to predict the action class.