Prediction Assignment1

Ning Guo

2016/12/2

Synopsis

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 (see the section on the Weight Lifting Exercise Dataset).

Loading the data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
library(caret); library(ggplot2); library(randomForest)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

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

```
fileURL1<- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileURL2<- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if(!file.exists("training")|!file.exists("testing")){
        # create the placeholder file
        td = tempdir()
        # download into the placeholder file
        tf1 = tempfile(tmpdir=td)
        download.file(fileURL1, tf1)
        pmltraining<- read.csv(tf1)

        tf2 = tempfile(tmpdir=td)
        download.file(fileURL2, tf2)
        pmltesting<- read.csv(tf2)
}
dim(pmltraining)
```

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

```
dim(pmltesting)
```

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

Data pre-processing

```
## Removing near zero variables
nzv<- nearZeroVar(pmltraining)
pmltrainingtemp<- pmltraining[,-nzv]
## Removing columns mosting with NAs
threshod<- dim(pmltraining)[1]*0.9
badcolumes<- which(apply(is.na(pmltrainingtemp), 2, sum) > threshod)
pmltrainingtidy<- pmltrainingtemp[,-badcolumes]
pmltestingtemp<- pmltesting[,-nzv]
pmltestingtidy<- pmltestingtemp[, -badcolumes]
## Removing non-measurement data
RemoveInx1 <- grepl("X|timestamp|user_name|problem_id", names(pmltrainingtidy))
RemoveInx2 <- grepl("X|timestamp|user_name|problem_id", names(pmltestingtidy))
pmltrainingtidy<- pmltrainingtidy[, which(RemoveInx1==FALSE)]
pmltestingtidy<- pmltestingtidy[, which(RemoveInx2==FALSE)]
```

Data splitting for resampling

```
set.seed(123)
inTrain<- createDataPartition(y=pmltrainingtidy$classe, p=0.7, list = FALSE)
training<- pmltrainingtidy[inTrain,]
testing<- pmltrainingtidy[-inTrain,]
```

Model fitting

I tried three different methods for modeling, including random forest, decision tree, LDA

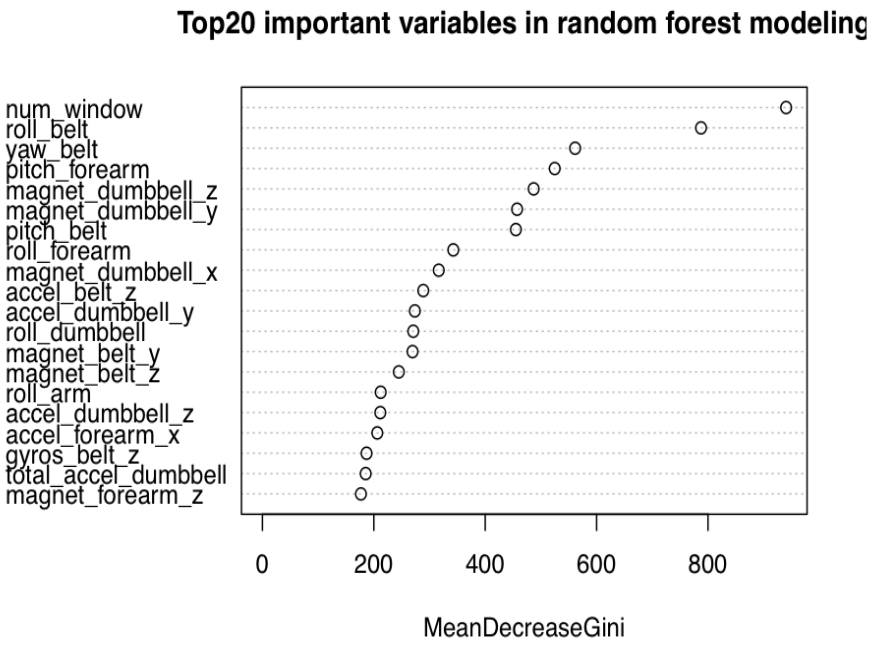
```
## random forest
modfit1<- randomForest(classe~., data = training)
prediction1<- predict(modfit1, testing)
confusionMatrix(testing$classe, prediction1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    8 1018    0    0
##          D    0    0   10  953    1
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9930   0.9903   1.0000   0.9991
## Specificity            1.0000   1.0000   0.9984   0.9978   1.0000
## Pos Pred Value         1.0000   1.0000   0.9922   0.9886   1.0000
## Neg Pred Value         1.0000   0.9983   0.9979   1.0000   0.9998
## Prevalence             0.2845   0.1949   0.1747   0.1619   0.1840
## Detection Rate         0.2845   0.1935   0.1730   0.1619   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9965   0.9943   0.9989   0.9995
```

```
## decision tree
## modfit2<- train(classe~., method="rpart", data = training)
## prediction2<- predict(modfit2, testing)
## confusionMatrix(testing$classe, prediction2)
## LDA
## modfit3<- train(classe~.,  method="lda", data = training)
## prediction3<- predict(modfit3, testing)
## confusionMatrix(testing$classe,  prediction3)
```

Random forest had the best performance with an accuracy of 99.68%, higher than that of 'rpart'(59.85%) and that of 'LDA'(70.43%). So random forest was used as the final model. Due to the limitations of the length of the report, the detailed performances of 'rpart' and 'LDA' were not shown here.

```
## Top20 important variables
varImpPlot(modfit1, n.var = 20, main = "Top20 important variables in random forest modeling")
```



From the dotchart, we found that "num_window" "roll_belt" "yaw_belt" were the TOP3 important variables for modelling.

Predict the test data

```
predanswer<- predict(modfit1, pmltestingtidy)
predanswer
```

```
##  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
```

```
# Write files for submission
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(predanswer)
```