Activity Recognition through Machine Learning

Junxiong

5/8/2019

Summary

For this project, we will predict the manner in which people did the exercise by using the data from accelerometers on the belt, forearm, arm, and dumbell. We mainly use the methods of random forest and decision tree from machine learning.

Data

The data for this project come from this source, s. We will first load the data as below.

```
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if(!"train.csv" %in% dir()){
    download.file(trainUrl, "train.csv")
}
if(!"test.csv" %in% dir()){
    download.file(testUrl, "test.csv")
}

train <- read.csv("train.csv")
test <- read.csv("test.csv")
```

Data Processing

After taking a look at the data, we find there are too many variables and also many empty entries. Thus, we will remove some of the low variable covariates and null-value covariates.

```
library(caret)
```

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

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

```
set.seed(314)
nsv <- nearZeroVar(train)
trainNSV <- train[,-nsv]
trainNSV <- trainNSV[,(colSums(is.na(trainNSV))==0)]
```

The first six columns of trainNSV contain informtaion about time stamp and users, which are not necessary for prediction. Therefore, we will remove them as well.

```
trainNSV <- trainNSV[,-c(1:6)]
```

For cross validation, we will subsample the training set.

```
inTrain <- createDataPartition(y=trainNSV$classe, p=0.6, list=FALSE)
inTrainNSV <- trainNSV[inTrain, ]
cvTrainNSV <- trainNSV[-inTrain, ]
```

Model selection

```
# Random forest
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
```

```
mod_rf <- randomForest(classe~., data=inTrainNSV)
pred_rf <- predict(mod_rf, cvTrainNSV)
confusionMatrix(pred_rf, cvTrainNSV$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   17    0    0    0
##          B    1 1500    7    0    0
##          C    0    1 1361    7    0
##          D    0    0    0 1277    1
##          E    0    0    0    2 1441
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9937, 0.9968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9881   0.9949   0.9930   0.9993
## Specificity            0.9970   0.9987   0.9988   0.9998   0.9997
## Pos Pred Value         0.9924   0.9947   0.9942   0.9992   0.9986
## Neg Pred Value         0.9998   0.9972   0.9989   0.9986   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1912   0.1735   0.1628   0.1837
## Detection Prevalence   0.2865   0.1922   0.1745   0.1629   0.1839
## Balanced Accuracy      0.9983   0.9934   0.9968   0.9964   0.9995
```

The out of sample error is

```
1-confusionMatrix(pred_rf, cvTrainNSV$classe)[[3]][1]
```

```
##    Accuracy
## 0.004588325
```

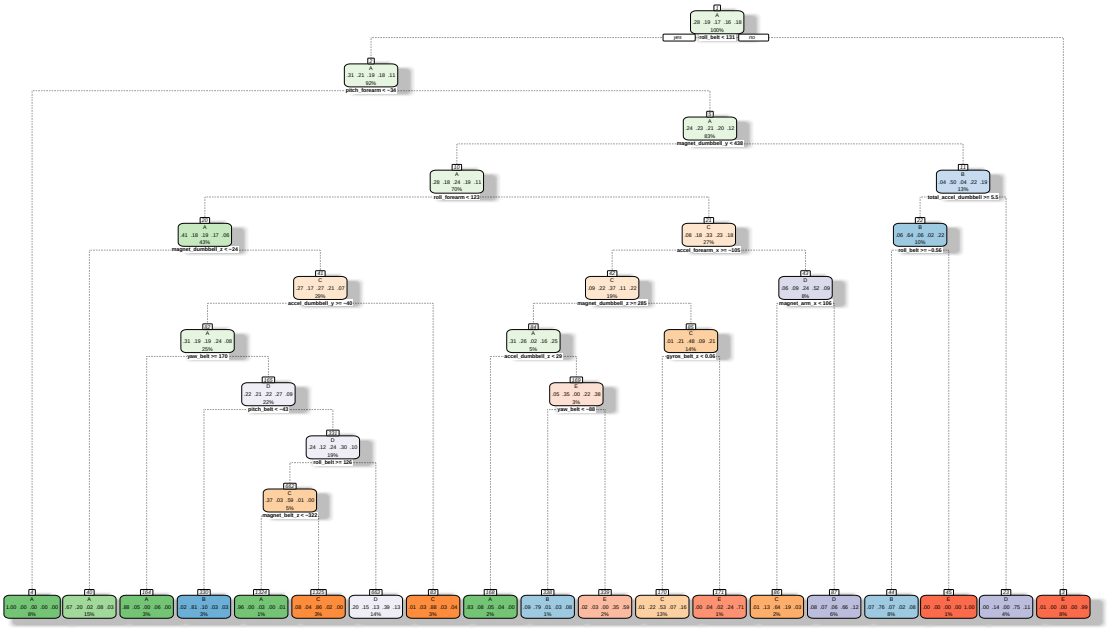
```
# Decision tree
library(rpart)
mod_rp <- rpart(classe~., data=inTrainNSV)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
fancyRpartPlot(mod_rp)
```



Rattle 2019−May−09 23:16:41 Junxiong

```
pred_rp <- predict(mod_rp, cvTrainNSV, type="class")
confusionMatrix(pred_rp, cvTrainNSV$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1898  290   37  114   36
##          B   36  717   63   30   70
##          C   41  267 1085  134  176
##          D  252  236  182  933  256
##          E    5    8    1   75  904
```

```
##
## Overall Statistics
##
##                Accuracy : 0.7057
##                  95% CI : (0.6955, 0.7158)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6281
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8504  0.47233   0.7931   0.7255   0.6269
## Specificity            0.9150  0.96855   0.9046   0.8588   0.9861
## Pos Pred Value         0.7992  0.78275   0.6371   0.5019   0.9104
## Neg Pred Value         0.9390  0.88442   0.9539   0.9410   0.9215
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2419  0.09138   0.1383   0.1189   0.1152
## Detection Prevalence   0.3027  0.11675   0.2171   0.2369   0.1266
## Balanced Accuracy      0.8827  0.72044   0.8489   0.7922   0.8065
```

The out of sample error is

```
1-confusionMatrix(pred_rp, cvTrainNSV$classe)[[3]][1]
```

```
##  Accuracy
## 0.2942901
```

From the output, we can see that random forest is better.

Prediction

```
predict(mod_rf, test)
```

```
##  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
```