Practical Machine Learning Project

Ivy Wang

2017/6/28

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.

Data source

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

The data for this project come from http://groupware.les.inf.puc-rio.br/har. Full source:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. "Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13)". Stuttgart, Germany: ACM SIGCHI, 2013.

I really apprieciate all above mentioned authors for being so generous in allowing their data to be used for this kind of assignment.

Load and process data

Firstly upload the necessary libraries to create a analysis environment.

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## 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
```

```
library("corrplot", lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
library(repmis)
set.seed(2333)
```

Secondly load and process data,make it tidy and clean.

```
training <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
##eliminate variables most of NA
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
##remove columns only variables
training <- training[,-c(1:7)]
testing<- testing[, -c(1:7)]

dim(training)
```

```
## [1] 19622    53
```

```
dim(testing)
```

```
## [1] 20 53
```

Thirdly, split tidy data. I prefer to set a 70/30 proportion to compute the out of sample error. Cut the the tidy training data into two part: training set (70%) for prediction and validation set (30%)

```
inTrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train <- training[inTrain, ]
valid <- training[-inTrain, ]
dim(train)
```

```
## [1] 13737    53
```

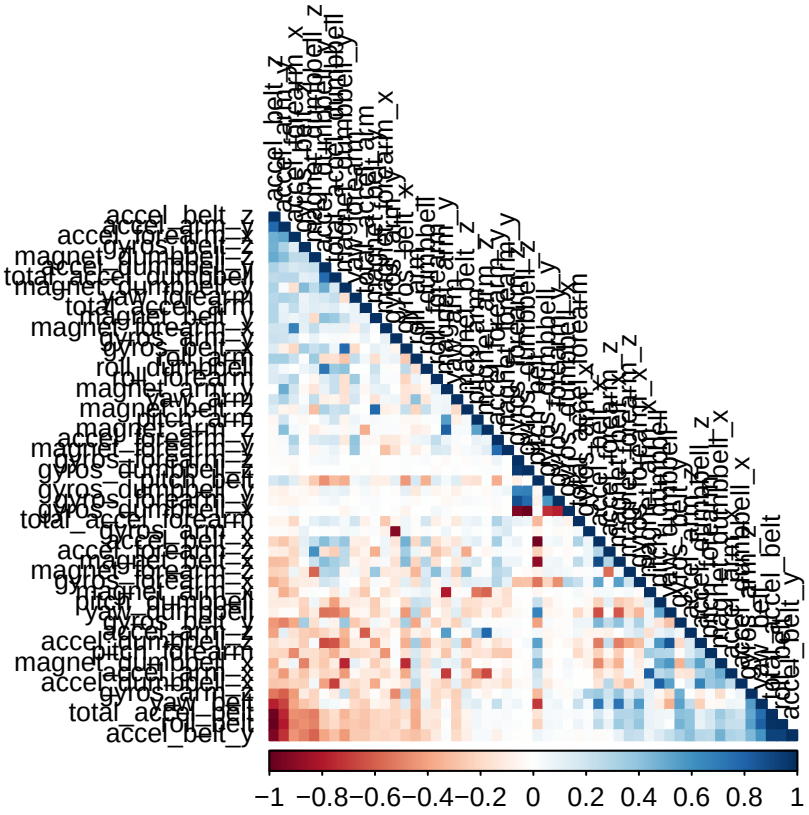
```
dim(valid)
```

```
## [1] 5885   53
```

Modeling and exploratory analysis

Before we proceed model procedules, we'd better check the correation among variables. The darker color means the higher corelation.

```
corMatrix <- cor(train[, -53])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



Random forests

```
control <- trainControl(method = "cv", number = 5)
fit_rf <- train(classe ~ ., data = train, method = "rf",
                trControl = control)
print(fit_rf, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10989, 10991, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9909    0.9885
##   27    0.9896    0.9868
##   52    0.9846    0.9805
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```r
# predict outcomes using validation set
predict_rf <- predict(fit_rf, valid)
# Show prediction result
(conf_rf <- confusionMatrix(valid$classe, predict_rf))
```

```
## 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   10 1014    2    0
##          D    0    0   15  946    3
##          E    0    0    0    2 1080
##
## 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            1.0000   0.9913   0.9854   0.9958   0.9972
## Specificity            1.0000   1.0000   0.9975   0.9964   0.9996
## Pos Pred Value         1.0000   1.0000   0.9883   0.9813   0.9982
## Neg Pred Value         1.0000   0.9979   0.9969   0.9992   0.9994
## Prevalence             0.2845   0.1952   0.1749   0.1614   0.1840
## Detection Rate         0.2845   0.1935   0.1723   0.1607   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9956   0.9915   0.9961   0.9984
```

Accuracy reaches 0.991 and out of sample error rate is 0.009, so random forest method is pretty good. It shows many predictors are highly correlated. Random forests chooses a subset of predictors at each split and decorrelate the trees.

Prediction on testing set

Since the result of random forest is quite reliable, we can predict now.

```r
predict(fit_rf, testing)
```

```
##  [1] 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
```