Prediction Assignment - Practical Machine Learning

Dongli Liu

February 1, 2016

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

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Data preparation

```r
library(dplyr)
library(data.table)
library(caret)
library(doParallel)
library(iterators)
library(parallel)
library(foreach)

set.seed(01312016)

pmlTraining <- fread("pml-training.csv", drop = c(1, 3:5))
pmlTesting <- fread("pml-testing.csv", drop = c(1, 3:5, 160))
```

First attempt, with NearZeroVariable from caret

While it is a natural idea to eliminate NearZeroVariable with following code:

```r
nzv <- nearZeroVar(pmlTraining)
pmlTrainingFilter <- select(pmlTraining, -nzv)
```

Then fit it with a random forest model:

```r
trainIndex <- createDataPartition(pmlTrainingFilter$classe,
                                  p = .6,
                                  list = FALSE)
```

```
training <- pmlTrainingFilter[trainIndex,]
validation <- pmlTrainingFilter[-trainIndex,]

cluster <- makeCluster(detectCores())
registerDoParallel(cluster)
modelControl <- trainControl(method="cv",
                             number=5,
                             allowParallel = TRUE)

rfFit <- train(classe ~ .,
               data = training,
               method = "rf",
               trControl = modelControl)
```

However, the performance is very poor:

```
print(rfFit)
```

```
## Random Forest
##
## 11776 samples
##    95 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 190, 192, 192, 190, 192
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.7580547  0.6921616  0.07109911   0.09145634
##   50    0.7452888  0.6776359  0.07098114   0.09051711
##   99    0.7451151  0.6770458  0.05663318   0.07132661
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

So let's forget it.

Second attempt, fine-tune with missing data elimination

It seems that there are so many missing data in the previous model, so let's try to remove them and have aonther try, with the knowledge from book *R in Action*:

```
library(mice)
md <- md.pattern(pmlTrainingFilter)
```

So it is clear that there is a *Not missing at random*, as described in above-mentioned book; which means removing those missing could be good:

```r
filterPredictor <- colnames(md)[c(1:53, 95, 96)]
pmlTrainingFilter2 <- select(pmlTraining, one_of(filterPredictor))

trainIndex2 <- createDataPartition(pmlTrainingFilter2$classe,
                                   p = .6,
                                   list = FALSE)
training2 <- pmlTrainingFilter2[trainIndex,]
validation2 <- pmlTrainingFilter2[-trainIndex,]

cluster <- makeCluster(detectCores())
registerDoParallel(cluster)
modelControl <- trainControl(method="cv",
                             number=5,
                             allowParallel = TRUE)

rfFit2 <- train(classe ~ .,
               data = training2,
               method = "rf",
               trControl = modelControl)
```

This time the performance is perfect:

```r
print(rfFit2)
```

```
## Random Forest
##
## 11776 samples
##    54 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9421, 9423, 9420, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9929512  0.9910832  0.001298866  0.001642302
##   30    0.9960085  0.9949513  0.001428463  0.001806920
##   58    0.9929507  0.9910833  0.002454783  0.003104513
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 30.
```

And the prediction on validation set is perfect as well:

```r
pred <- predict(rfFit2, newdata = validation2)
confusionMatrix(pred, validation2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##           A 2232   11    0    0    0
##           B    0 1506    5    0    0
##           C    0    1 1363   16    0
##           D    0    0    0 1270    2
##           E    0    0    0    0 1440
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9955          
##                  95% CI : (0.9938, 0.9969)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9944          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9963   0.9876   0.9986
## Specificity            0.9980   0.9992   0.9974   0.9997   1.0000
## Pos Pred Value         0.9951   0.9967   0.9877   0.9984   1.0000
## Neg Pred Value         1.0000   0.9981   0.9992   0.9976   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1919   0.1737   0.1619   0.1835
## Detection Prevalence   0.2859   0.1926   0.1759   0.1621   0.1835
## Balanced Accuracy      0.9990   0.9957   0.9969   0.9936   0.9993
```

Apply to Testing Set

Finally, apply this model to our test and get the answer:

```
pred2 <- predict(rfFit2, newdata = pmlTesting)
print(pred2)
```

```
##  [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
```