Pratical Machine Learning - Course Project

Haijie Cao

27 mars 2017

1 Setting things up

1.1 Load useful packages

```
library(caret)
library(rpart)
library(randomForest)
```

1.2 Download files

```
filename.1<-"pml-training.csv"
if(!file.exists(filename.1)){
        urlfile.1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
        download.file(urlfile.1,destfile = filename.1,method = "curl")
}

filename.2<-"pml-testing.csv"
if(!file.exists(filename.2)){
        urlfile.2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
        download.file(urlfile.2,destfile = filename.2,method = "curl")
}
```

1.3 Read the dataset in R

```
## reading the data
trainset<-read.csv(filename.1,header = T,na.strings = c("NA","","#DIV/0!"))
testset<-read.csv(filename.2,header = T,na.strings = c("NA","","#DIV/0!"))
```

1.4 Clean the dataset

Our project goal is to predict the manner that people do their exercices. There are five classes for the prediction, and we should choose the variables to predict with.
At the fist view of the dataset, we observe that, the 7 first variables are infomations not necessary about the exercices' performance. We can leave them out. And there are many variables with lots of NA data, we could eliminate them too.

```
##leave out the first 7 variable.
trainset<-trainset[,8:160]
testset<-testset[,8:160]
## eliminate the variables which have many NA data. (more than 90% NA data)
list<-c()
for (i in 1:153) {
      if (sum(is.na(trainset[,i]))/nrow(trainset)>=0.9) {
              list<-c(list,-i)
      }
}
trainset<-trainset[,list]
testset<-testset[,list]
dim(trainset)
```

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

Finaly, we will keep 53 variables.

2 Fitting models

2.1 Build the taining and testing set

```
## data split, training set has 60% of the data
set.seed(12345)
trainIndex<-createDataPartition(trainset$classe,p=0.6,list=F)
training<-trainset[trainIndex,]
testing<-trainset[-trainIndex,]
```

2.2 Cross-validation set up for training set.

```
set.seed(234)
trainControl<-trainControl(method = "cv",number = 10)
```

2.3 Decision tree

Let's start by building a decision tree, it may well be enough to predict the classes.

```
## we set "accuracy" as our model evaluation metrics
metric<-"Accuracy"
## training the data
set.seed(123)
fit.tree<-train(classe~.,data=training,method="rpart",metric=metric,trControl=trainControl)
print(fit.tree)
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10599, 10599, 10598, 10598, 10599, 10598, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03909587  0.5160537  0.38363590
##   0.03946369  0.5160537  0.38363590
##   0.11449929  0.3384859  0.08270477
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03946369.
```

We observe that the best accuracy is only 51%, it's not a good performance, it's not even necessary to evaluate the testing set. We could reject this model directly.
We have to try something else.

2.4 Ramdom forest

Fitting a random forest model seems a good option because it will resample and bootstrap data for each node it creates. But it will take some time to compute.

```
fit.rf<-train(classe~.,data=training,method="rf",metric=metric,
              trControl=trainControl,prox=F)
print(fit.rf)
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10599, 10597, 10600, 10597, 10599, 10599, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9903196  0.9877538
##   27    0.9902354  0.9876481
##   52    0.9868381  0.9833505
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
## challenging our model with the test set
testRF<-predict(fit.rf,newdata=testing)
confRF<-confusionMatrix(testing$classe,testRF)
confRF$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    4    0    0    0
##          B   10 1505    3    0    0
##          C    0    9 1359    0    0
##          D    0    0   18 1264    4
##          E    0    0    2    4 1436
```

```
confRF$overall[1]
```

```
##  Accuracy
## 0.9931175
```

This model's out of sample accuracy is over 99%, which is way more satisfying than the decision tree's accuracy. Out of sample error :

```
1-confRF$overall[1]
```

```
##    Accuracy
## 0.006882488
```

The out of sample error rate is of 0.8%, which is excellent.

3 Quizz prediction

Now let's predict the Quizz set classes :

```
predict(fit.rf,newdata = testset)
```

```
##  [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
```