Practical machine learning project

adrian lim

Friday, November 20, 2015

Executive Summary

In this project, we will use machine learning techniques to model exercise data that consists of data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal is to predict the manner in which the participants did the exercise (the "classe" variable in the training set) using the other variables in the set. The data source is from Weight Lifting (see the section on the Weight Lifting Exercise Dataset).

We utilised the random forest technique to predict the exercise classe. The model we built has an out of sample error around 0.66% which is quite good. Further work can be done by fine tuning the model as well as exploring and comparing other techniques for example boosting etc to further improve the model accuracy.

Download test and training data and perform exploratory data analysis on the test set

First lets explore the test dataset to get a feel for the data.

```
library(R.utils)
```

```
## Loading required package: R.oo
## Loading required package: R.methodsS3
## R.methodsS3 v1.7.0 (2015-02-19) successfully loaded. See ?R.methodsS3 for help.
## R.oo v1.19.0 (2015-02-27) successfully loaded. See ?R.oo for help.
##
## Attaching package: 'R.oo'
##
## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods
##
## The following objects are masked from 'package:base':
##
##     attach, detach, gc, load, save
##
## R.utils v2.1.0 (2015-05-27) successfully loaded. See ?R.utils for help.
##
## Attaching package: 'R.utils'
##
## The following object is masked from 'package:utils':
##
##     timestamp
##
## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, inherits, isOpen, parse, warnings
```

```
library(RCurl)
```

```
## Loading required package: bitops
##
## Attaching package: 'RCurl'
##
## The following object is masked from 'package:R.utils':
##
##     reset
##
## The following object is masked from 'package:R.oo':
##
##     clone
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(knitr)
library(caret)
```

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

```
#
# Doun load the data
#
trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#setInternet2(use = TRUE)
#if (!file.exists("./data/train.csv")) {
#    download.file(trainURL,"./data/train.csv",method='curl')
#}
#if (!file.exists("./data/test.csv")) {
#    download.file(testURL,"./data/test.csv",method='curl')
#}
trainRawFile <- read.csv('./data/train.csv', sep=',', stringsAsFactors = FALSE, strip.white = TRUE)
testRawFile <- read.csv('./data/test.csv', sep=',', stringsAsFactors = FALSE,strip.white = TRUE)
```

From inspection, it is clear that there is a lot of missing data in the data set and a fair amount of cleaning is required.

Create a training and a testing set from the downloaded trainset in a 60:40 ratio

```
set.seed(80)
intrain = createDataPartition(trainRawFile$classe,p=0.6,list=FALSE)
trainSet = trainRawFile[intrain,]
testSet = trainRawFile[-intrain,]
dim(trainSet)
```

```
## [1] 11776   160
```

```
dim(testSet)
```

```
## [1] 7846  160
```

Clean the data

We now clean the data in 3 steps:
- removing unneccessary variables
- remove variables which do not have predictive value
- remove variables which have a lot of missing data

A) Remove columns that are not related to movement

The first 7 variables (ID, subject names, timestamp info) can be removed as they are static data and time based information which is not related to what we are trying to predict.

```
trainSet <- trainSet[,-c(1:7)]
testSet <- testSet[,-c(1:7)]
```

B) Remove near to zero variance variables as they do not have predictive value

We use the nearZeroVar function to remove variables that have either one unique value or variables which have very few unique values relative to the number of samples and where the second most common value is prevalent. These kind of variables have close to zero variance and hence are not good predictors.

```
nzv <- nearZeroVar(trainSet, saveMetrics=TRUE)
trainSet <- trainSet[,nzv$nzv==FALSE]
nzv <- nearZeroVar(testSet, saveMetrics=TRUE)
testSet <- testSet[,nzv$nzv==FALSE]

dim(trainSet)
```

```
## [1] 11776   100
```

```
dim(testSet)
```

```
## [1] 7846  103
```

C)Remove fields with NAs

Finally we remove variables with lots of missing data.

```
naNum <- sapply(trainSet, function(x) {sum(is.na(x))})
clean_cols = names(naNum[naNum==0]) #select column names with no NA data
dfTrainCleaned = trainSet[, names(trainSet) %in% clean_cols]

# Ensure ony same variables are in the Cleaned Test Set
dfTestCleaned <- testSet[colnames(dfTrainCleaned)]

dim(dfTrainCleaned)
```

```
## [1] 11776    53
```

```
dim(dfTestCleaned)
```

```
## [1] 7846   53
```

Build a prediction model

We now build a model using random forests with cross validation and 5 folds. This step was repeated a few times to find the model with the smallest out of sample error by adjusting the modelling parameters. To speed up the computation we enable parallel processing using the appropriate libraries

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
## Loading required package: iterators
```

```
clust <- makeCluster(detectCores() - 1)
registerDoParallel(clust)
ctrl <- trainControl(classProbs=TRUE, savePredictions=TRUE, allowParallel=TRUE, method = "cv", number =
model <- train(classe~., method = "rf", trControl = ctrl,data=dfTrainCleaned)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
stopCluster(clust)
```

Evaluate the Model on the training dataset (40% of data set aside originally)

We now evaluate the model on the data set.

```
model
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9421, 9422, 9420, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9876871  0.9844221  0.001499150  0.001898070
##   27    0.9873472  0.9839929  0.002481796  0.003142591
##   52    0.9768177  0.9706734  0.005595295  0.007087291
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
aaa <- mean(predict(model, dfTestCleaned) == dfTestCleaned$classe) * 100
oos <- 100 - aaa # Out of Sample Error Rate
```

The out of sample error rate is 0.6755034%.
The confusion matrix for the model is as shown below.

```
predict1 <- predict(model, dfTestCleaned)
confusionMatrix(predict1, dfTestCleaned$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    8    0    0    0
##          B    1 1507   11    0    0
##          C    0    3 1354   23    0
##          D    0    0    3 1262    1
##          E    2    0    0    1 1441
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9912, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9915
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9928   0.9898   0.9813   0.9993
## Specificity            0.9986   0.9981   0.9960   0.9994   0.9995
## Pos Pred Value         0.9964   0.9921   0.9812   0.9968   0.9979
## Neg Pred Value         0.9995   0.9983   0.9978   0.9964   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1921   0.1726   0.1608   0.1837
## Detection Prevalence   0.2851   0.1936   0.1759   0.1614   0.1840
## Balanced Accuracy      0.9986   0.9954   0.9929   0.9904   0.9994
```

```
save(model,file="./data/model.RData")
```

Run the Model on the test dataset

We now run the model on the test dataset.

```
load(file="./data/model.RData",verbose=TRUE)
```

```
## Loading objects:
##   model
```

```
predict2 <- predict(model, testRawFile)
```

Submit Results of the test dataset

Finally we write the result files for the 20 test cases for submission.

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("./result/problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(predict2)
```

Conclusion

We have build a model using the random forest technique to predict the exercise classe. The model's out of sample error is around 0.66% which is quite good. Further work can be done by tuning the model as well as comparing other techniques for example boosting etc.