Practical Machine Learning Assignment

Raymond Kwan

10 October 2015

Loading the data from given files

The following code load the relevant libraries, as well as the training and testing data set from the provided CSV files.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.2
```

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

```
set.seed(12345)
trainingDat <- read.csv("pml-training.csv", sep=",")
testDat <- read.csv("pml-testing.csv", sep=",")
```

The following checks the type of categories of the outcome variable. This is to provide an understanding of the nature and types of categorical outcomes.

```
levels(trainingDat$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

One of the problems associated with the data set is the some variables have very little variability, and so the command *nearZeroVar* can be used to isolate these variables and thereby removing the unneccessary variables. The next step is to select only relevant variables by further removing the participant identities as well as the time-related data.

```
nsv <- nearZeroVar(trainingDat,saveMetrics=T)
trainingDat2 <- trainingDat[,!nsv$nzv]
trainingDat2 <- trainingDat2[,7:100]
```

During the data cleaning process, it was found that there are a large number of NA data. While it is possible to remove these data from the data set, their removal would result in a significantly reduced set of data. Instead, we use the *knnImpute* to impute the NA data set in order to protect the remaining data from being removed.

```
preObj <- preProcess(trainingDat2[,-94],method="knnImpute")
```

The following is to check and make sure that the data types for all variables are as they should be.

```
tmp <- rep(0, ncol(trainingDat2))
for (i in 1:ncol(trainingDat2)) {
    tmp[i] <- class(trainingDat2[,i])
}
```

Once the data type are correct, the preProcess object *preObj* is used to produce the imputed data.

```
trainingDat2[,-94] <- predict(preObj, trainingDat2[,-94])
```

Finally, the resulting set of training data is partitioned into the training and the testing set for the purpose of cross-validation. Thus, once the model is formed using the training data set, the prediction accuracy would be evaluated using the testing set.

```
inTrain <- createDataPartition(y=trainingDat2$classe,
                               p=0.75, list=F)
training <- trainingDat2[inTrain,]
testing <- trainingDat2[-inTrain,]
dim(training)
```

```
## [1] 14718    94
```

It was found that the use of the entire training data set is very computationally intensive, and that training time is very long. Thus, in order to reduce the computation time, only 3000 samples are selected randomly from the original training data set. This gives rise to a trade-off between accuracy and computation time.

```
tmp2 <- sample_n(training, 3000)
```

Random forest and Cross-validation

The following code fits the random forest model. Based on the model, the predicted outcomes based on the testing data set are produced. It is important to note that the testing data set is based on the partitioned training data is, and is *NOT* based on the testing set provided by the file pml-testing.csv in the project. The testing data set from pml-testing.csv is for the final evaluation of the predictions. Also, cross validation has been done to assess the accuracy of the prediction.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.2
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
time_rf <- system.time(modFit_rf <- train(subset(tmp2,select=-c(classe)), tmp2$classe, method="rf", na.a
numOfRowWithNAs <- length(unique(unlist(lapply(testing, function(x) which(is.na(x))))))
dim(testing)
```

```
## [1] 4904   94
```

```
print(numOfRowWithNAs)
```

```
## [1] 0
```

```
result_rf <- predict(object=modFit_rf, newdata=testing)
comTest_rf <- table(result_rf, testing$classe)
accuracy_rf <- sum(as.matrix(comTest_rf) * diag(dim(comTest_rf)[1])) / sum(comTest_rf)
```

Generalized Boosted regression Model with PCA and Cross-Validation

The following code is for fitting the Generalized Boosted regression Model (GBM) algorithm when the principle component analysis (PCA) is used as a preprocessing. The parameter *thresh* is set to 0.99 in order to allow the number of components to capturn 99 percents of the variance. Cross validation has been done to assess the accuracy of the prediction.

```
ctr <- trainControl(preProcOptions=list(thresh=0.99))
time_gbm_pca <- system.time(modFit_gbm_pca <- train(classe~., data=tmp2, method="gbm",
                              verbose=F, preProcess="pca", trControl=ctr) )
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.2
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr
## ------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------
##
## Attaching package: 'plyr'
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
numOfRowWithNAs <- length(unique(unlist(lapply(testing, function(x) which(is.na(x))))))
dim(testing)
```

```
## [1] 4904   94
```

```
print(numOfRowWithNAs)
```

```
## [1] 0
```

```
result_gbm_pca <- predict(object=modFit_gbm_pca, newdata=testing)
comTest_gbm_pca <- table(result_gbm_pca, testing$classe)
accuracy_gbm_pca <- sum(as.matrix(comTest_gbm_pca) * diag(dim(comTest_gbm_pca)[1])) / sum(comTest_gbm_pc
```

Based on the above lines, it can be seen that there are 4904 rows, but there is no rows with NAs as expected. Subsequently, the results and accuracy are stored in their respective variables.

Generalized Boosted regression Model and Cross-Validation

The following code fits the GBM algorithm model in the absence of the PCA as preprocessing. Cross validation has been done to assess the accuracy of the prediction.

```
time_gbm <- system.time(modFit_gbm <- train(classe~., data=tmp2, method="gbm",
                                verbose=F))

numOfRowWithNAs <- length(unique(unlist(lapply(testing, function(x) which(is.na(x))))))
dim(testing)
```

```
## [1] 4904   94
```

```
print(numOfRowWithNAs)
```

```
## [1] 0
```

```
result_gbm <- predict(object=modFit_gbm, newdata=testing)
comTest_gbm <- table(result_gbm, testing$classe)
accuracy_gbm <- sum(as.matrix(comTest_gbm) * diag(dim(comTest_gbm)[1])) / sum(comTest_gbm)
```

Finally, the confusion matrices for the GBM-PCA, GBM, and RF predictions are

```r
print(comTest_gbm_pca)
```

```
##
## result_gbm_pca    A    B    C    D    E
##              A 1272  108   50   32   14
##              B   46  710   64   27   85
##              C   26   74  675   89   68
##              D   37   30   26  611   49
##              E   14   27   40   45  685
```

```r
print(comTest_gbm)
```

```
##
## result_gbm    A    B    C    D    E
##          A 1354   62    5    2    3
##          B   15  838   28    9   18
##          C   10   36  811   53   25
##          D    6    6   10  718   13
##          E   10    7    1   22  842
```

```r
print(comTest_rf)
```

```
##
## result_rf    A    B    C    D    E
##         A 1363   39   16   11    3
##         B    7  876   24    4   12
##         C    9   30  808   41   22
##         D   11    4    5  740   13
##         E    5    0    2    8  851
```

To predict the test data

The accuracies for GBM-PCA, GBM, and RF are

```r
print(accuracy_gbm_pca)
```

```
## [1] 0.8060767
```

```r
print(accuracy_gbm)
```

```
## [1] 0.9304649
```

```r
print(accuracy_rf)
```

```
## [1] 0.9457586
```

respectively. In the test cases provided by the project under file *pml-testing.csv*, there are 20 cases that need to be predicted. Based on the above accuracy results, the estimated number of erroneous prediction for the three methods are $20 \times (1 - accuracy\_gbm\_pca)$=4, $20 \times (1 - accuracy\_gbm)$=1, and $20 \times (1 - accuracy\_rf)$=1 respectively. The results suggest that the use of PCA on top of GBM does not improve the accuracy. Also, the random forest prediction seems to be the most accurate among the three.

The following sets up the data structure of the evaluation test data from the file pml-testing.csv provided in the project. Here, we have

- removed the categorical outcome variable *classe*, and
- used the *preObj* object to impute the NA data in the test set.

```
tmp <- as.vector(names(trainingDat2))
tmp <- tmp[-94]
testDat2 <- subset(testDat, select=tmp)
dim(testDat2)
```

```
## [1] 20 93
```

```
testDat2 <- predict(preObj, testDat2)
```

The following code stores the results into the respective variables.

```
result_gbm_pca_testDat2 <- predict(object=modFit_gbm_pca, newdata=testDat2)
result_gbm_testDat2 <- predict(object=modFit_gbm, newdata=testDat2)
result_rf_testDat2 <- predict(object=modFit_rf, newdata=testDat2)
```

The predicted results are

```
print(result_gbm_pca_testDat2)
```

```
##  [1] B A A A A C D E A A A C B A E E A B B B
## Levels: A B C D E
```

```
print(result_gbm_testDat2)
```

```
##  [1] B A B C A E D B A A B C B A E E A B A B
## Levels: A B C D E
```

```
print(result_rf_testDat2)
```

```
##  [1] B A B A A E D D A A B C B A E E A B B B
## Levels: A B C D E
```

Finally, the following code writes the results into separate files as specified in the project.

```
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
    }
}
pml_write_files(result_rf_testDat2)
```

The predicted data based on random forest were used in the second part of the project. The result suggests that there is one single error out of 20 examples. This observation is consistent with the earlier predicted accuracy of $20 \times (1 - accuracy\_rf)$=1.

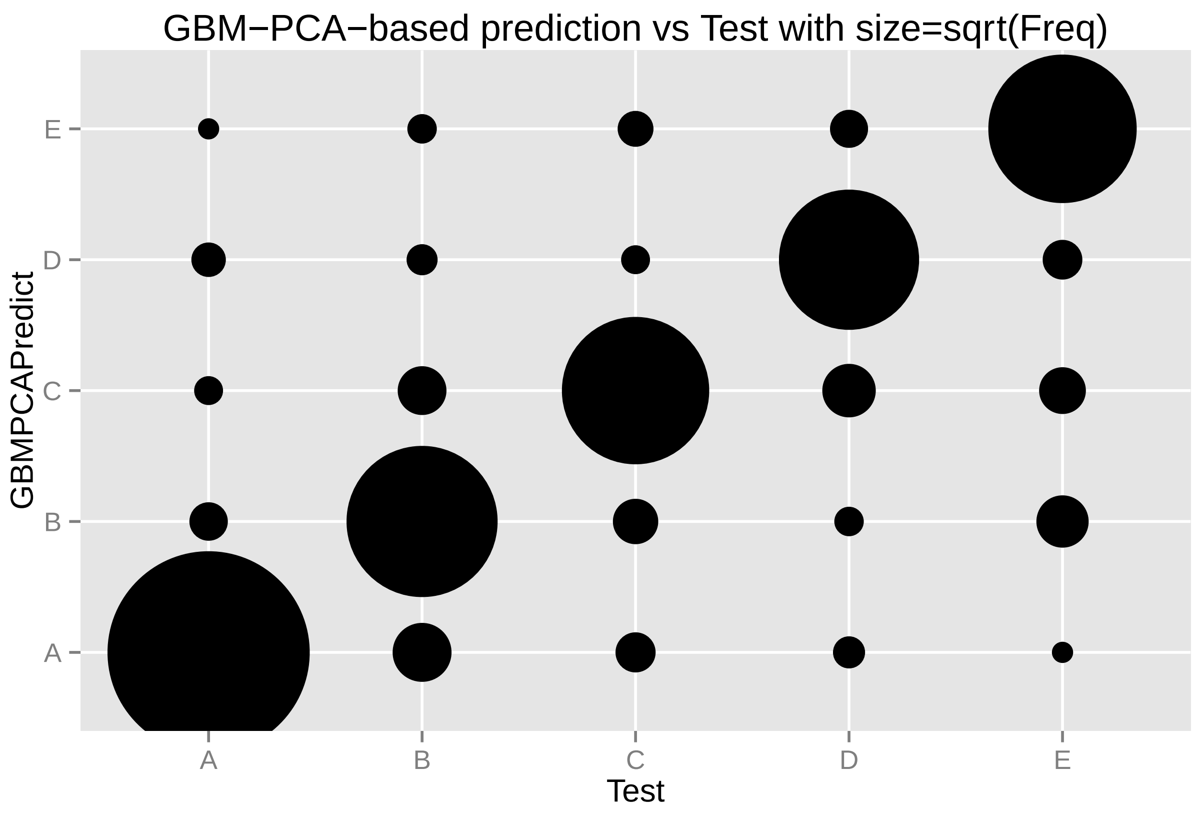
Appendix

In this appendix, three scatter plots are shown comparing the predicted results used for the cross-validation for

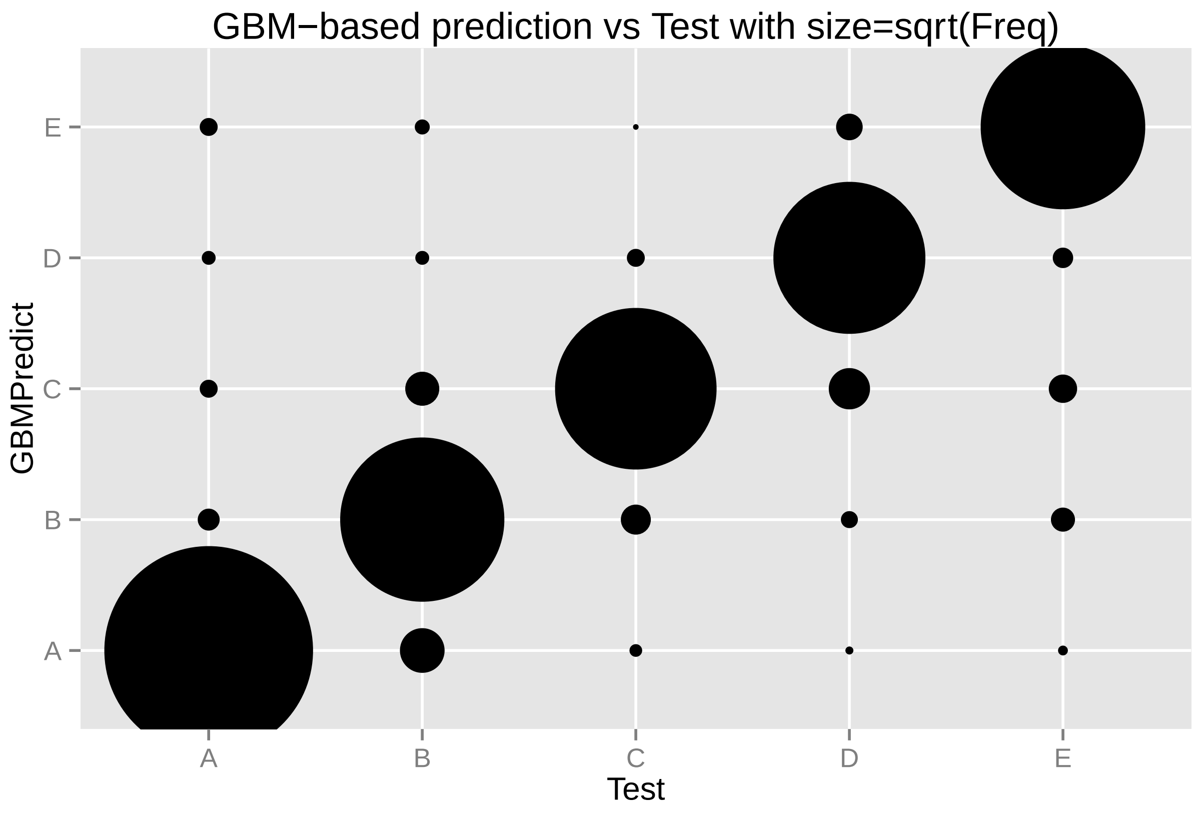
- Generalized Boosted regression Model with PCA (GBM-PCA)
- Generalized Boosted regression Model without PCA (GBM)
- Random Forest (RF)

The size of the point reflects the square-root of the number of occurances.

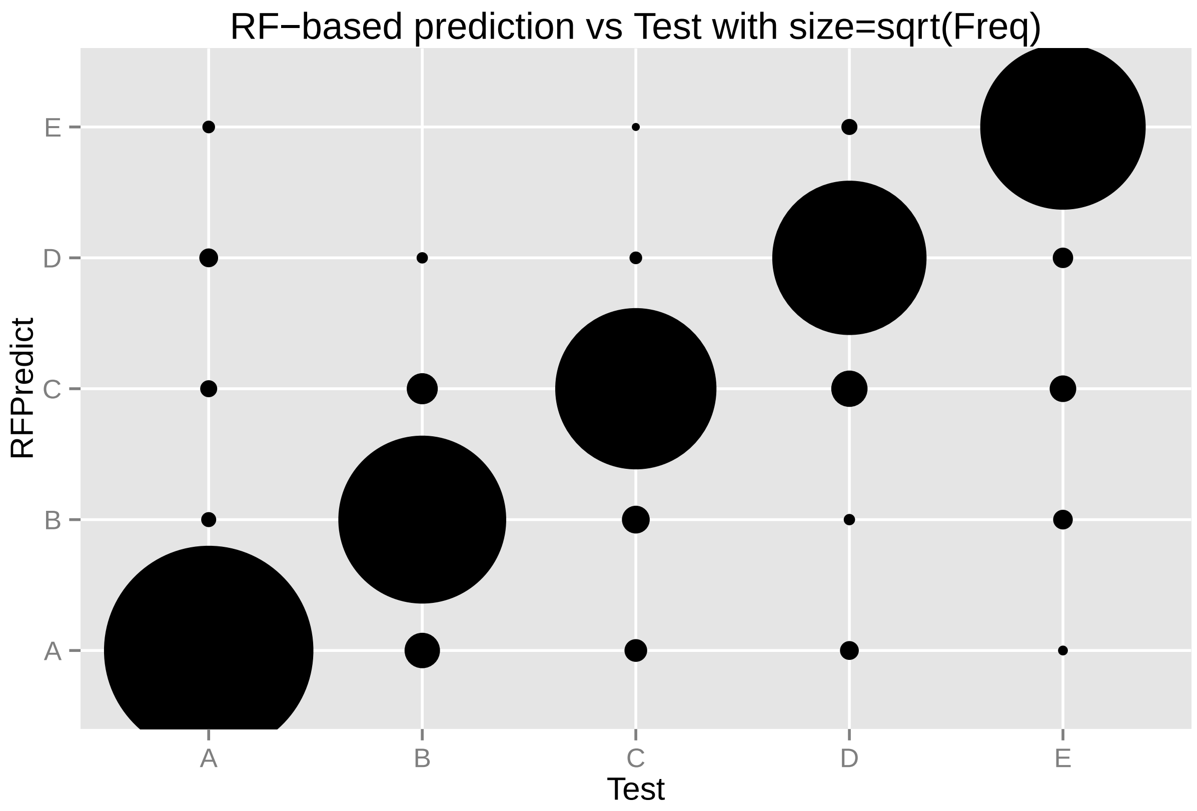
```
tmp <- as.data.frame(comTest_gbm_pca)
names(tmp) <- c("GBMPCAPredict", "Test", "Freq")
ggplot(tmp, aes(x=Test, y=GBMPCAPredict)) +
    geom_point(size=sqrt(tmp$Freq)) + ggtitle("GBM-PCA-based prediction vs Test with size=sqrt(Freq)")
```



```
tmp <- as.data.frame(comTest_gbm)
names(tmp) <- c("GBMPredict", "Test", "Freq")
ggplot(tmp, aes(x=Test, y=GBMPredict)) +
    geom_point(size=sqrt(tmp$Freq)) + ggtitle("GBM-based prediction vs Test with size=sqrt(Freq)")
```



```
tmp <- as.data.frame(comTest_rf)
names(tmp) <- c("RFPredict", "Test", "Freq")
ggplot(tmp, aes(x=Test, y=RFPredict)) +
    geom_point(size=sqrt(tmp$Freq)) + ggtitle("RF-based prediction vs Test with size=sqrt(Freq)")
```


RF−based prediction vs Test with size=sqrt(Freq)
RFPredict
E
D
C
B
A
Test
A
B
C
D
E