main

Install and load packages we need

```
if(!require("EBImage")){
  source("https://bioconductor.org/biocLite.R")
  biocLite("EBImage")
}
if(!require("R.matlab")){
  install.packages("R.matlab")
}
if(!require("readxl")){
  install.packages("readxl")
}

if(!require("dplyr")){
  install.packages("dplyr")
}
if(!require("readxl")){
  install.packages("readxl")
}

if(!require("ggplot2")){
  install.packages("ggplot2")
}

if(!require("caret")){
  install.packages("caret")
}

library(R.matlab)
library(readxl)
library(dplyr)
library(EBImage)
library(ggplot2)
library(caret)
```

Construct features

Step 0 set work directories, extract paths, summarize

```
set.seed(0)
setwd("~/Desktop/proj3-sec2-group6/doc")
# here replace it with your own path or manually set it in RStudio to where this rmd file is located.
# use relative path for reproducibility
```

Provide directories for training images. Training images and Training fiducial points will be in different subfolders.

```
train_dir <- "../data/train_set/" # This will be modified for different data sets.
train_image_dir <- paste(train_dir, "images/", sep="")
train_pt_dir <- paste(train_dir,  "points/", sep="")
train_label_path <- paste(train_dir, "label.csv", sep="")
```

Step 1: set up controls for evaluation experiments.

In this chunk, we have a set of controls for the evaluation experiments.

- (T/F) cross-validation on the training set
- (number) K, the number of CV folds
- (T/F) process features for training set
- (T/F) run evaluation on an independent test set
- (T/F) process features for test set

```
run.cv=TRUE # run cross-validation on the training set
K <- 5  # number of CV folds
run.feature.train=TRUE # process features for training set
run.test=TRUE # run evaluation on an independent test set
run.feature.test=TRUE # process features for test set
```

Using cross-validation or independent test set evaluation, we compare the performance of models with different specifications. In this Starter Code, we tune parameter k (number of neighbours) for KNN.

```
k = c(5,11,21,31,41,51)
model_labels = paste("KNN with K =", k)
```

Step 2: import data and train-test split

```
#train-test split
info <- read.csv(train_label_path)
n <- nrow(info)
n_train <- round(n*(4/5), 0)
train_idx <- sample(info$Index, n_train, replace = F)
test_idx <- setdiff(info$Index,train_idx)
```

If you choose to extract features from images, such as using Gabor filter, R memory will exhaust all images are read together. The solution is to repeat reading a smaller batch(e.g 100) and process them.

```
n_files <- length(list.files(train_image_dir))

image_list <- list()
for(i in 1:100){
   image_list[[i]] <- readImage(paste0(train_image_dir, sprintf("%04d", i), ".jpg"))
}
```

Fiducial points are stored in matlab format. In this step, we read them and store them in a list.

```
#function to read fiducial points;If you want to process test data(without emotion_idx),use function:feature_test
#input: index
#output: matrix of fiducial points corresponding to the index
readMat.matrix <- function(index){
     return(round(readMat(paste0(train_pt_dir, sprintf("%04d", index), ".mat"))[[1]],0))
}

#load fiducial points
fiducial_pt_list <- lapply(1:n_files, readMat.matrix)
save(fiducial_pt_list, file="../output/fiducial_pt_list.RData")
```

Step 3: construct features and responses

- The follow plots show how pairwise distance between fiducial points can work as feature for facial emotion recognition.
- In the first column, 78 fiducials points of each emotion are marked in order.
- In the second column distributions of vertical distance between right pupil(1) and right brow peak(21) are shown in histograms. For example, the distance of an angry face tends to be shorter than that of a surprised face.
- The third column is the distributions of vertical distances between right mouth corner(50) and the midpoint of the upper lip(52). For example, the distance of an happy face tends to be shorter than that of a face.

Figure1

Figure1

`feature.R` should be the wrapper for all your feature engineering functions and options. The function `feature( )` should have options that correspond to different scenarios for your project and produces an R object that contains features and responses that are required by all the models you are going to evaluate later.

- `feature.R`
- Input: list of images or fiducial point
- Output: an RData file that contains extracted features and corresponding responses

Step 4: We need split data intro training and testing set

```
source("../lib/feature.R")
tm_feature_train <- NA
if(run.feature.train){
  tm_feature_train <- system.time(dat_train <- feature(fiducial_pt_list, train_idx))
}

tm_feature_test <- NA
if(run.feature.train){
  tm_feature_test <- system.time(dat_test <- feature(fiducial_pt_list, test_idx))
}

# summarizing time for feature construction
cat("Time for constructing training feature =", tm_feature_train[1], "s \n")
```

```
## Time for constructing training feature = 0.84 s
```

```
cat("Time for constructing testing feature =", tm_feature_test[1], "s \n")
```

```
## Time for constructing testing feature = 0.206 s
```

```
save(dat_train, file="../output/feature_train.RData")
save(dat_test, file="../output/feature_test.RData")
```

load training and testing Rdata

```
load("../output/feature_train.RData")
load("../output/feature_test.RData")
```

base model

This is how we work out gbm model, it will take more than 3 hours. Thus I saved the model.

```
fitControl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 1)

gbmGrid <- expand.grid(interaction.depth = 2,
                       n.trees = 300,
                       shrinkage = 0.15,
                       n.minobsinnode = 10)

tm_train_gbm<-system.time(gbmFit_base <- train(emotion_idx~., data = dat_train,
                method = "gbm",
                trControl = fitControl,
```

```
                verbose = TRUE,
                tuneGrid = gbmGrid))
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        3.0910             nan     0.1500    0.1113
##      2        2.6288             nan     0.1500    0.0852
##      3        2.4149             nan     0.1500    0.0655
##      4        2.2579             nan     0.1500    0.0077
##      5        2.1108             nan     0.1500    0.0512
##      6        1.9926             nan     0.1500   -0.0080
##      7        1.8970             nan     0.1500    0.0010
##      8        1.8063             nan     0.1500    0.0099
##      9        1.7182             nan     0.1500    0.0041
##     10        1.6368             nan     0.1500   -0.0098
##     20        1.0999             nan     0.1500   -0.0281
##     40        0.5842             nan     0.1500   -0.0072
##     60        0.3346             nan     0.1500   -0.0059
##     80        0.2054             nan     0.1500   -0.0042
##    100        0.1332             nan     0.1500   -0.0029
##    120        0.0883             nan     0.1500   -0.0019
##    140        0.0596             nan     0.1500   -0.0007
##    160        0.0408             nan     0.1500   -0.0008
##    180        0.0283             nan     0.1500   -0.0006
##    200        0.0197             nan     0.1500   -0.0004
##    220        0.0138             nan     0.1500   -0.0003
##    240        0.0099             nan     0.1500   -0.0002
##    260        0.0071             nan     0.1500   -0.0001
##    280        0.0050             nan     0.1500   -0.0001
##    300        0.0036             nan     0.1500   -0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        3.0910             nan     0.1500    0.1676
##      2        2.6402             nan     0.1500    0.1127
##      3        2.4031             nan     0.1500    0.0277
##      4        2.2400             nan     0.1500    0.0385
##      5        2.0983             nan     0.1500    0.0196
##      6        1.9733             nan     0.1500    0.0142
##      7        1.8539             nan     0.1500    0.0103
##      8        1.7688             nan     0.1500    0.0276
##      9        1.6796             nan     0.1500    0.0045
##     10        1.6035             nan     0.1500   -0.0046
##     20        1.0769             nan     0.1500   -0.0199
##     40        0.5604             nan     0.1500   -0.0064
##     60        0.3249             nan     0.1500   -0.0035
##     80        0.2011             nan     0.1500   -0.0041
##    100        0.1312             nan     0.1500   -0.0021
##    120        0.0882             nan     0.1500   -0.0014
##    140        0.0596             nan     0.1500   -0.0011
##    160        0.0408             nan     0.1500   -0.0008
##    180        0.0283             nan     0.1500   -0.0006
##    200        0.0198             nan     0.1500   -0.0004
##    220        0.0141             nan     0.1500   -0.0003
##    240        0.0099             nan     0.1500   -0.0003
##    260        0.0070             nan     0.1500   -0.0002
##    280        0.0050             nan     0.1500   -0.0001
##    300        0.0036             nan     0.1500   -0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        3.0910             nan     0.1500    0.1390
##      2        2.6383             nan     0.1500    0.1462
##      3        2.4111             nan     0.1500    0.0074
##      4        2.2672             nan     0.1500    0.0531
##      5        2.1160             nan     0.1500    0.0503
##      6        1.9841             nan     0.1500    0.0150
##      7        1.8727             nan     0.1500    0.0287
##      8        1.7789             nan     0.1500    0.0018
##      9        1.6948             nan     0.1500    0.0105
##     10        1.6098             nan     0.1500   -0.0131
##     20        1.0844             nan     0.1500   -0.0060
##     40        0.5730             nan     0.1500   -0.0075
##     60        0.3330             nan     0.1500   -0.0048
##     80        0.2052             nan     0.1500   -0.0012
##    100        0.1319             nan     0.1500   -0.0024
##    120        0.0881             nan     0.1500   -0.0016
##    140        0.0605             nan     0.1500   -0.0008
##    160        0.0416             nan     0.1500   -0.0007
##    180        0.0288             nan     0.1500   -0.0004
##    200        0.0202             nan     0.1500   -0.0004
##    220        0.0140             nan     0.1500   -0.0004
##    240        0.0098             nan     0.1500   -0.0002
##    260        0.0070             nan     0.1500   -0.0002
##    280        0.0050             nan     0.1500   -0.0001
##    300        0.0036             nan     0.1500   -0.0001
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        3.0910             nan     0.1500    0.0721
##      2        2.6540             nan     0.1500    0.1066
##      3        2.4689             nan     0.1500    0.0620
##      4        2.2940             nan     0.1500    0.0975
##      5        2.1152             nan     0.1500   -0.0013
##      6        1.9875             nan     0.1500   -0.0073
##      7        1.8890             nan     0.1500   -0.0086
##      8        1.8051             nan     0.1500   -0.0269
##      9        1.7233             nan     0.1500   -0.0003
##     10        1.6430             nan     0.1500   -0.0237
##     20        1.0946             nan     0.1500   -0.0047
##     40        0.5710             nan     0.1500   -0.0088
##     60        0.3321             nan     0.1500   -0.0069
##     80        0.2045             nan     0.1500   -0.0050
##    100        0.1335             nan     0.1500   -0.0022
##    120        0.0895             nan     0.1500   -0.0016
##    140        0.0607             nan     0.1500   -0.0011
##    160        0.0416             nan     0.1500   -0.0007
##    180        0.0285             nan     0.1500   -0.0005
```

```
##    200        0.0200             nan     0.1500   -0.0003
##    220        0.0141             nan     0.1500   -0.0003
##    240        0.0100             nan     0.1500   -0.0002
##    260        0.0071             nan     0.1500   -0.0002
##    280        0.0051             nan     0.1500   -0.0001
##    300        0.0037             nan     0.1500   -0.0001
## 
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        3.0910             nan     0.1500    0.2830
##      2        2.6294             nan     0.1500    0.0996
##      3        2.4057             nan     0.1500    0.0786
##      4        2.2415             nan     0.1500    0.0235
##      5        2.1131             nan     0.1500    0.0295
##      6        1.9881             nan     0.1500    0.0598
##      7        1.8695             nan     0.1500    0.0074
##      8        1.7827             nan     0.1500    0.0209
##      9        1.6939             nan     0.1500    0.0009
##     10        1.6122             nan     0.1500   -0.0168
##     20        1.0747             nan     0.1500   -0.0246
##     40        0.5605             nan     0.1500   -0.0100
##     60        0.3229             nan     0.1500   -0.0045
##     80        0.1991             nan     0.1500   -0.0022
##    100        0.1290             nan     0.1500   -0.0027
##    120        0.0855             nan     0.1500   -0.0013
##    140        0.0581             nan     0.1500   -0.0012
##    160        0.0403             nan     0.1500   -0.0005
##    180        0.0280             nan     0.1500   -0.0006
##    200        0.0197             nan     0.1500   -0.0004
##    220        0.0139             nan     0.1500   -0.0002
##    240        0.0098             nan     0.1500   -0.0003
##    260        0.0071             nan     0.1500   -0.0002
##    280        0.0050             nan     0.1500   -0.0001
##    300        0.0036             nan     0.1500   -0.0001
## 
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        3.0910             nan     0.1500    0.1706
##      2        2.6863             nan     0.1500    0.1284
##      3        2.4730             nan     0.1500    0.0952
##      4        2.3051             nan     0.1500    0.0999
##      5        2.1551             nan     0.1500    0.0544
##      6        2.0367             nan     0.1500    0.0364
##      7        1.9257             nan     0.1500    0.0306
##      8        1.8264             nan     0.1500   -0.0012
##      9        1.7589             nan     0.1500    0.0294
##     10        1.6746             nan     0.1500    0.0200
##     20        1.1805             nan     0.1500   -0.0122
##     40        0.6785             nan     0.1500   -0.0083
##     60        0.4230             nan     0.1500   -0.0068
##     80        0.2747             nan     0.1500   -0.0039
##    100        0.1884             nan     0.1500   -0.0022
##    120        0.1321             nan     0.1500   -0.0011
##    140        0.0937             nan     0.1500   -0.0016
##    160        0.0685             nan     0.1500   -0.0009
##    180        0.0499             nan     0.1500   -0.0008
##    200        0.0372             nan     0.1500   -0.0007
##    220        0.0279             nan     0.1500   -0.0004
##    240        0.0209             nan     0.1500   -0.0004
##    260        0.0157             nan     0.1500   -0.0003
##    280        0.0119             nan     0.1500   -0.0002
##    300        0.0090             nan     0.1500   -0.0001
```

```
saveRDS(gbmFit_base,file = '../lib/gbmFit_base.rds')
```

Load the model.

```
gbmFit_base <- readRDS('../lib/gbmFit_base.rds')
```

Predict the data

```
tm_train_base <- system.time(pred0 <- predict(gbmFit_base, newdata = dat_train))
tm_test_base <- system.time(pred1 <- predict(gbmFit_base, newdata = dat_test))
```

```
accu0 <- mean(dat_train$emotion_idx == pred0)
accu1 <- mean(dat_test$emotion_idx == pred1)
```

```
cat("The accuracy of model: gredient boosting on training set", "is", accu0*100, "%.\n")
```

```
## The accuracy of model: gredient boosting on training set is 100 %.
```

```
cat("The accuracy of model: gredient boosting on testing set", "is", accu1*100, "%.\n")
```

```
## The accuracy of model: gredient boosting on testing set is 43.2 %.
```

```
#confusionMatrix(pred1, dat_test$emotion_idx)
```

summarizing running time for base model

```
#cat("Time for building base model=", tm_train_gbm[1], "s \n")
cat("Time for training base model=", tm_train_base[1], "s \n")
```

```
## Time for training base model= 1.821 s
```

```
cat("Time for testing base model=", tm_test_base[1], "s \n")
```

```
## Time for testing base model= 1.241 s
```

advanced model

normalize training set & PCA on training set

```
# normalize
df_train_X <- scale(dat_train[,!(names(dat_train) %in% 'emotion_idx')])
df_train_Y <- dat_train$emotion_idx

# PCA on training
source('../lib/pca_feature.R')

pca <- pca_feature(df_train_X,threshhold=0.99)

# combine PCA X, Y after PCA
pca_train <- data.frame(pca$data_X_transformed,emotion_idx=df_train_Y)
```

build advance model on training

```
source('../lib/train_advance.R')

run.advance.train=TRUE
tm_advance_train=NA

if(run.advance.train){
  tm_advance_lda2 <- system.time(advance_model <- train_advance(pca_train))
}
```

scale on test & transform test into PC dimensions

```
df_test_X <- scale(dat_test[,!(names(dat_test) %in% 'emotion_idx')])
df_test_Y <- dat_test$emotion_idx

data_test_X_transformed <-  df_test_X  %*% pca$trans_matrix

# combine test X, Y after PCA
pca_test <- data.frame(data_test_X_transformed,emotion_idx=df_test_Y)
```

predict on training and testing

```
source('../lib/test_advance.R')

tm_advance_train <- system.time(pred_train_advance <- test_advance(model=advance_model,dat_pca_test=pca_train))

tm_advance_test <- system.time(pred_test_advance <- test_advance(model=advance_model,dat_pca_test=pca_test))

cat("The accuracy of advance model(lda2) on training set:",confusionMatrix(pred_train_advance,reference = pca_train$emotion_idx)$ove
```

```
## The accuracy of advance model(lda2) on training set: 73.25 %.
```

```
cat("The accuracy of advance model(lda2) on testing set:",confusionMatrix(pred_test_advance,reference = pca_test$emotion_idx)$overal
```

```
## The accuracy of advance model(lda2) on testing set: 50.8 %.
```

summarizing running time

```
cat("Time for building advance model=", tm_advance_lda2[1], "s \n")
```

```
## Time for building advance model= 15.25 s
```

```
cat("Time for training advance model=", tm_advance_train[1], "s \n")
```

```
## Time for training advance model= 0.031 s
```

```
cat("Time for testing advance model=", tm_advance_test[1], "s \n")
```

```
## Time for testing advance model= 0.017 s
```