Main

group 10

In your final repo, there should be an R markdown file that organizes **all computational steps** for evaluating your proposed Facial Expression Recognition framework.

```
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)
library(randomForest)
library(mlbench)
```

Step 0 set work directories

```
set.seed(0)
setwd("~/Desktop/fall2019-proj3-sec2--grp10/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.

```
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
```

Step 2: import data and train-test split

```
#train-test split
set.seed(0)
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)
```

```
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
#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

`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

```
library(mlbench)
load("../output/fiducial_pt_list.RData")
```

```r
source("../lib/feature.R")
# load the data
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.test){
  tm_feature_test <- system.time(dat_test <- feature(fiducial_pt_list, test_idx))
}

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

feature selection using train set rfe method

```r
# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
tm_feature_selection <- system.time(results <- rfe(dat_train1[,-6007], dat_train1[,6007],size = c(1:20)
                                                   rfeControl=control))

#tm_selectin
 #user    system   elapsed
#13310.534    88.755 13984.268
```

###feature selection using train set 2nd method

```r
set.seed(0)
#38 min
rPartMod <- train(emotion_idx ~ ., data=dat_train1, method="rpart")
rpartImp <- varImp(rPartMod)
rpartImp
save(rpartImp,file = "../output/feature_selection2.RData")
save(results, file="../output/feature_selection.RData")
#first five same as rfe
```

Step 4: Train a classification model with training features and responses

```r
load("../output/feature_selection.RData")
load("../output/feature_selection2.RData")
feature = rownames(rpartImp$importance)[1:20]
position = c()
for (i in 1:length(feature)){
  position[i] = which(predictors(results) ==  feature[i])
}

load("../output/feature_train.RData")
load("../output/feature_test.RData")
set.seed(0)

# summarize the results
```

```r
#print(results)
# list the chosen features
# plot the results
#plot(results, type=c("g", "o"))
```

Model selection with cross-validation

- Do model selection by choosing among different values of training model parameters.

Baseline GBM

```r
model_values <- list(depth = c(3,6,9), lr = 10^c(-2:-3))
depth = c(3,6,9)
lr = 10^c(-2:-3)
model_labels_gbm = paste('GBM with depth =', rep(model_values$depth, rep(4,3)), ', learning rate =',
                         rep(model_values$lr, 5))

source("../lib/cross_validation_gbm.R")
if(run.cv){
    err_cv_gbm <- matrix(0, nrow = length(depth), ncol = length(lr))
    for(i in 1:length(depth)){
        cat("depth=", depth[i], "\n")
        for(j in 1:length(lr)){
            cat("lr=",lr[j],"\n")
            err_cv_gbm[i,j] <- cv.function(dat_train, K, depth[i],lr[j])
        }
    }
    save(err_cv_gbm, file="../output/err_cv_gbm.RData")
}
```

- Choose the "best" parameter value

Baseline GBM

```r
model_values <- list(depth = c(3,6,9), lr = 10^c(-2:-3))
depth = c(3,6,9)
lr = 10^c(-2:-3)
model_labels_gbm = paste('GBM with depth =', rep(model_values$depth, rep(4,3)), ', learning rate =', rep
load("../output/err_cv_gbm.RData")
if(run.cv){
  inds = which(err_cv_gbm == min(err_cv_gbm), arr.ind=TRUE)
  depth_best = depth[inds[,1]]
  lr_best = lr[inds[,2]]
}
par_best_gbm = list(depth = depth_best, lr = lr_best)
```

- Train the model with the entire training set using the selected model (model parameter) via cross-validation.

GBM Baseline

```r
source("../lib/train_gbm.R")
tm_train_gbm=NA
```

```r
tm_train_gbm <- system.time(fit_train_gbm <- train(dat_train, par_best_gbm))
save(fit_train_gbm, file="../output/fit_train_gbm.RData")
```

Step 5: Run test on test images

baseline gbm

```r
source("../lib/test_gbm.R")
tm_test_gbm=NA
if(run.test){
  load(file="../output/fit_train_gbm.RData")
  tm_test_gbm <- system.time(pred_gbm <- test(fit_train_gbm, dat_test))
}
```

```
## Loaded gbm 2.1.5
```

- evaluation

baseline gbm

```r
pbest.pred <- apply(pred_gbm, 1, which.max)
accu <- mean(dat_test$emotion_idx == pbest.pred)
cat("The accuracy of baseline gbm:" ,"is",paste('GBM with depth =', par_best_gbm[[1]],
                                              ', learning rate =', par_best_gbm[[2]]), accu*100, "%.\n
```

```
## The accuracy of baseline gbm: is GBM with depth = 6 , learning rate = 0.01 43.8 %.
```

improved model

Model selection with cross-validation

- Do model selection by choosing among different values of training model parameters.

```r
load("../output/feature_train.RData")
load("../output/feature_test.RData")
set.seed(0)
f = 500
dat_train = dat_train[,c(predictors(results)[1:f],"emotion_idx",feature[11:20])]
dat_test = dat_test[,c(predictors(results)[1:f],"emotion_idx",feature[11:20])]
```

Random forest

```r
source("../lib/cross_validation_rf.R")
para = c(400,500,600,700,800)
model_labels = paste("Random Forest with number of trees =", para)
if(run.cv){
  err_cv <- matrix(0, nrow = length(para), ncol = 2)
  for(i in 1:length(para)){
    cat("number of trees=", para[i], "\n")
    err_cv[i,] <- cv.function(dat_train, K, para[i])
  save(err_cv, file="../output/err_rf.RData")
  }
}
```

- Choose the "best" parameter value

Random forest

```
para = c(400,500,600,700,800)

load("../output/err_rf.RData")
colnames(err_cv) <- c("mean cv.error","sd cv.error")
rownames(err_cv) <- para
if(run.cv){
  model_best <- para[which.min(err_cv[,1])]
}

#save(model_best,file = "../output/model_best_rf.Rdata")
```

- Train the model with the entire training set using the selected model (model parameter) via cross-validation.

Random Forest

```
load("../output/model_best_rf.Rdata")
par_best <- model_best
source("../lib/train_rf.R")
tm_train_rf=NA
tm_train_rf <- system.time(fit_train <- train_rf(dat_train,par_best))
#save(fit_train, file="../output/fit_train_rf.RData")
```

Step 5: Run test on test images

random forest

```
source("../lib/test_rf.R")
tm_test_rf=NA
if(run.test){
  load(file="../output/fit_train_rf.RData")
  tm_test_rf <- system.time(pred <- test_rf(fit_train,dat_test))
}
```

- evaluation

random forest

```
model_labels = paste("Random Forest with number of trees =", para)
accu <- mean(dat_test$emotion_idx == pred)
cat("The accuracy of random forest:", model_labels[which.min(err_cv[,1])], "is", accu*100, "%.\n")
```

```
## The accuracy of random forest: Random Forest with number of trees = 600 is 46.6 %.
```

Note that the accuracy is not high but is better than that of ramdom guess(4.5%).

Summarize Running Time

Prediction performance matters, so does the running times for constructing features and for training the model, especially when the computation resource is limited.

```
cat("Time for constructing training features=", tm_feature_train[1], "s \n")
```

```
## Time for constructing training features= 3.248 s
```

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

```
## Time for constructing testing features= 0.254 s
```

```r
cat("Time for training baseline gbm model=", 1784, "s \n")
```

```
## Time for training baseline gbm model= 1784 s
```

```r
cat("Time for testing baseline gbm model=", tm_test_gbm[1], "s \n")
```

```
## Time for testing baseline gbm model= 23.71 s
```

```r
cat("Time for training random forest model=", tm_train_rf[1], "s \n")
```

```
## Time for training random forest model= 353.114 s
```

```r
cat("Time for testing random forest model=", tm_test_rf[1], "s \n")
```

```
## Time for testing random forest model= 0.336 s
```

```r
#cat("Time for selecting features using training sets=", tm_feature_selection[1], "s \n")
```

Reference

- Du, S., Tao, Y., & Martinez, A. M. (2014). Compound facial expressions of emotion. Proceedings of the National Academy of Sciences, 111(15), E1454-E1462.