Project 3 - Example Main Script

Group5: Peifeng Hong, Leo Lam, Shiyu Liu, Qianli Zhu, Sitong Chen

20180322

Package needed

```
#packages <- c("e1071", "dplyr","gbm","randomForest","EBImage","xgboost",
"data.table")


#packages.needed=setdiff(packages,intersect(installed.packages()[,1], pac
kages))

#if(length(packages.needed)>0){install.packages(packages.needed, dependen
cies = TRUE)}

library("e1071")
library("dplyr")
library("gbm")
library("randomForest")
library("EBImage")
library("xgboost")
library("data.table")
```

Step 0: specify directories.

Set the working directory to the image folder.

```
setwd("~/Documents/GitHub/Spring2018-Project3-Group5")
```

Provide directories for raw images. Training set and test set should be in different subfolders. ????????

```
experiment_dir <- "~/Documents/GitHub/Spring2018-Project3-Group5/data/" #
 This will be modified for different data sets.
img_train_dir <- paste(experiment_dir, "train/images/", sep="")
img_test_dir <- paste(experiment_dir, "train/images/", 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

```
K <- 5  # number of CV folds
run.feature.train=F # process features for training pictures
run.feature.test=T # process features for testing pictures
run.train = T # if true, train model on training data, else use saved mod
el
run.test=T # run evaluation on an independent test set
run.train.baseline=T
```

Step 2: import training images class labels.

```
labels<-read.csv("~/Documents/GitHub/Spring2018-Project3-Group5/output/la
bel_train.csv")
label_train<-labels$label
```

Step 3: Construct visual features

```
source("~/Documents/GitHub/Spring2018-Project3-Group5/lib/feature.R")


if(run.feature.train){

# get SIFT feature
  sift_feature_train <- read.csv("~/Documents/GitHub/Spring2018-Project3-
Group5/output/SIFT_feature.csv",header = F)
  sift_feature_train <- sift_feature_train[,-1]
  sift_feature_train$y <- label_train
  #saveRDS(sift_feature_train, file="~/Documents/GitHub/Spring2018-Projec
t3-Group5/output/feature_SIFT_train.RData")

# get RGB feature
  tm_feature <- system.time(rgb_feature_train <- feature(img_train_dir,
        "train",
        data_name="rgb",
        export=TRUE))[3]
} else {
#load SIFT feature
  sift_feature_train<-readRDS("~/Documents/GitHub/Spring2018-Project3-Gro
up5/output/feature_SIFT_train.RData")
#load RGB feature
  rgb_feature_train<-readRDS('~/Documents/GitHub/Spring2018-Project3-Grou
p5/output/feature_rgb_train.RData')

}
```

```
if(run.feature.test){

# get SIFT feature
  sift_feature_test <- read.csv("~/Documents/GitHub/Spring2018-Project3-Group5/output/SIFT_feature.csv",header = F)
  sift_feature_test <- sift_feature_test[,-1]
  #saveRDS(sift_feature_test, file="~/Documents/GitHub/Spring2018-Project3-Group5/output/feature_SIFT_test.RData")

# get RGB feature
  tm_feature_test <- system.time(rgb_feature_test <- readRDS("~/Documents/GitHub/Spring2018-Project3-Group5/output/feature_rgb_train.RData"))[3]
  #saveRDS(rgb_feature_test, file="~/Documents/GitHub/Spring2018-Project3-Group5/output/feature_rgb_test.RData")

}
```

Step 4: Train a baseline model (GBM+SIFT) with training images

Call the train model and test model from library.

```
source("~/Documents/GitHub/Spring2018-Project3-Group5/lib/train_gbm.R")
source("~/Documents/GitHub/Spring2018-Project3-Group5/lib/test_gbm.R")
#source("~/Documents/GitHub/Spring2018-Project3-Group5/lib/cross_validation.R")

if(run.train.baseline){
# process data
  training_y_SIFT <- label_train
  training_x_SIFT<- sift_feature_train[ ,!(colnames(sift_feature_train) %in% c("y"))]

# train model
  tm_train_base <- system.time(base_model <-
              train(training_x_SIFT,training_y_SIFT))[3]
  #saveRDS(base_model,file='~/Documents/GitHub/Spring2018-Project3-Group5/output/base_model.RData')
}else{
  base_model <- readRDS('~/Documents/GitHub/Spring2018-Project3-Group5/output/base_model.RData')
}
base_model <- readRDS('~/Documents/GitHub/Spring2018-Project3-Group5/output/base_model.RData')
```

Step 5: Train advanced model (XGBoost+RGB) with training images

```
source("~/Documents/GitHub/Spring2018-Project3-Group5/lib/train_xgboost.r
")
source("~/Documents/GitHub/Spring2018-Project3-Group5/lib/test_xgboost.r
")

if(run.train){
# process data
  training_y_RGB<- label_train-1
  training_x_RGB<- rgb_feature_train[,!(colnames(rgb_feature_train) %in%
c("y"))]
# train model
  best_para <- xgboost_para(rgb_feature_train,training_y_RGB,K)[[2]]
  tm_train <- system.time(best_model <- xgboost_train(training_x_RGB, tra
ining_y_RGB, best_para))[3]
  #saveRDS(best_model,file='~/Documents/GitHub/Spring2018-Project3-Group5
/output/best_model.RData')
  best_model <- readRDS('~/Documents/GitHub/Spring2018-Project3-Group5/ou
tput/best_model.RData')
}
best_model <- readRDS('~/Documents/GitHub/Spring2018-Project3-Group5/outp
ut/best_model.RData')
```

Step 6: Make prediction

Feed the final training model with the completely holdout testing data.

```
tm_test_base=NA
tm_test=NA
if(run.test){
  #process data
  testing_x_SIFT<- sift_feature_test
  #test model
  tm_test_base <- system.time(base_test_model <- test(base_model,testing_
x_SIFT))[3]
  #saveRDS(base_test_model,'~/Documents/GitHub/Spring2018-Project3-Group5
/output/labelsbase.Rdata')
  #write.csv(file='~/Documents/GitHub/Spring2018-Project3-Group5/output/b
est_test_model_baseline.csv', x=base_test_model)
}
if(run.test){
  testing_x_rgb<- rgb_feature_test
  #best_test_model is the result
  tm_test <- system.time(best_test_model <- xgboost_test_result(best_mode
l, rgb_feature_test)+1)[3]
  #saveRDS(best_test_model,file='~/Documents/GitHub/Spring2018-Project3-G
roup5/output/best_test_model.RData')
```

```
  #write.csv(file='~/Documents/GitHub/Spring2018-Project3-Group5/output/b
est_test_model_advanved.csv', x=best_test_model)
}
```

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 test features=", tm_feature_test, "s \n")

## Time for constructing test features= 0.073 s

#cat("Time for constructing testing features=", tm_feature_test, "s \n")
cat("Time for training baseline model=", tm_train_base, "s \n")

## Time for training baseline model= 489.771 s

cat("Time for training advanced model=", tm_train, "s \n")

## Time for training advanced model= 12.005 s

cat("Time for baseline model to make prediction=", tm_test_base, "s \n")

## Time for baseline model to make prediction= 1.045 s

cat("Time for advanced model to make prediction=", tm_test, "s \n")

## Time for advanced model to make prediction= 0.071 s

accuracy_test <- 1-mean(best_test_model != label_train)
cat("Accuracy rate for advanced model=",accuracy_test,"\n")

## Accuracy rate for advanced model= 0.97
```