Project 3

Keran Li, Mingming Liu, Zhongxing Xue, Yuhan Zha, Junkai Zhang

March 19, 2018

Step 0: Prepare packages

```
# Load the libraries
packages.used <- c("EBImage","gbm","caret","nnet","xgboost","xgboost",
                   "haven","tidyverse","mlr","plyr","ggplot2","randomForest",
                   "adabag","neuralnet","grDevices","kernlab","tree")

#if(!require("EBImage")){
#  source("https://bioconductor.org/biocLite.R")
#  biocLite("EBImage")
#}

# Check packages
packages.needed=setdiff(packages.used,
                        intersect(installed.packages()[,1],
                                  packages.used))
# install needed packages
if(length(packages.needed)>0){
  install.packages(packages.needed, dependencies = TRUE)
}

library("EBImage")
library("gbm")
library("kernlab")
library("tree")
library("caret")
library("nnet")
library("xgboost")
library("haven")
library("tidyverse")
library("mlr")
library("plyr")
library("ggplot2")
library("randomForest")
library("adabag")
library("neuralnet")
library("grDevices")
```

Step 0: specify directories.

Set the working directory to the image folder. Specify the training and the testing set. For data without an independent test/validation set, you need to create your own testing data by random subsampling. In order to obain reproducible results, set.seed() whenever randomization is used.

```
setwd("~/Desktop/Spring2018-Project3-Group3/")
```

```
experiment_dir <- "../data/"
img_train_dir <- paste(experiment_dir, "train/", sep="")
img_test_dir <- paste(experiment_dir, "test/", sep="")
```

Step 1: Set up controls for evaluation experiments.

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

(T/F) process features for training set (T/F) run evaluation on an independent test set (T/F) process features for test set

```
run.feature.train = TRUE
run.test = TRUE
run.feature.test = TRUE
```

Set up controls indicating which model to run

```
run.xgboost = TRUE
run.adaboost = FALSE
run.tree = FALSE
run.gbm = FALSE
run.lg = FALSE
run.nnt = FALSE
run.rf = FALSE
run.svm = FALSE
```

Step 2: Extract features

```
source("../lib/SelectFeature.R")

# tm_feature_train <- NA
# if(run.feature.train){
#   tm_feature_train <- system.time(dat_train <- SelectFeature(SIFTname, SubGroup))
# }
# cat("Time for constructing training features=", tm_feature_train[1], "s \n")

#tm_feature_test <- NA
#if(run.feature.test){
#  tm_feature_test <- system.time(dat_test <-SelectFeature())
#}
# Running time
#cat("Time for constructing testing features=", tm_feature_test[1], "s \n")

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

Step 3: Import training and test data of best features.

```
tr_class <- read.csv('../output/Feature_v3_TrainClass.csv', header = TRUE)
tr_data <- read.csv('../output/Feature_v3_TrainClass.csv', header = TRUE)
te_class <- read.csv('../output/TestClass_SIFT.csv', header = TRUE)
```

```
te_data <- read.csv('../output/TestData_SIFT.csv', header = TRUE)

train_all_data <- rbind(tr_data, te_data)
train_all_class <- rbind(tr_class, te_class)

# Training data
dat_train <- cbind(train_all_class[,-1], train_all_data[,-1])
colnames(dat_train)[1] <- "class"

# testdata <- cbind(te_class[,-1], te_data[,-1])
# colnames(test)[1] <- "class"
# dim(dat_train)
```

Step 4: Train a classification model with training images

Call the train model and test model from library.

`train.R` and `test.R` should be wrappers for all your model training steps and your classification/prediction steps. + `train.R` + Input: a path that points to the training set features. + Input: an R object of training sample labels. + Output: an RData file that contains trained classifiers in the forms of R objects: models/settings/links to external trained configurations. + `test.R` + Input: a path that points to the test set features. + Input: an R object that contains a trained classifier. + Output: an R object of class label predictions on the test set. If there are multiple classifiers under evaluation, there should be multiple sets of label predictions.

```
source("../lib/train.R")
source("../lib/test.R")
```

- Train the model with the entire training set using the selected model.

```
if(run.xgboost){
  xgboost_train<- train_xgboost(dat_train)
  cat("Time for testing model=", xgboost_train$time, "s \n")
}
if(run.adaboost){
  adaboost_train <- train_adaboosting(dat_train)
  cat("Time for testing model=", adaboost_train$time, "s \n")
}
if(run.tree){
  tree_train <- train_tree(dat_train)
  cat("Time for testing model=", tree_train$time, "s \n")
}
if(run.gbm){
  gbm_train <-  train_GBM(dat_train)
  cat("Time for testing model=", gbm_train$time, "s \n")
}
if(run.lg){
  lg_train <- train_logistic(dat_train)
  cat("Time for testing model=", lg_train$time, "s \n")
}
if(run.nnt){
  nnt_train <- train_nn(dat_train)
  cat("Time for testing model=", nnt_train$time, "s \n")
}
```

```
if(run.rf){
  rf_train <- train_rf(dat_train)
  cat("Time for testing model=", rf_train$time, "s \n")
}
if(run.svm){
  svm_train <- train_svm(dat_train)
  cat("Time for testing model=", svm_train$time, "s \n")
}
```

Step 5: Make prediction and summarize Running Time

Feed the final training model with the completely holdout testing data. Prediction performance matters, so does the running times for constructing features and for training the model, especially when the computation resource is limited.

```
tm_xgboost_test=NA
tm_adaboost_test=NA
tm_tree_test=NA
tm_gbm_test=NA
tm_lg_test=NA
tm_nnt_test=NA
tm_rf_test=NA
tm_svm_test=NA

if(run.xgboost){
  load(file="../lib/XgBoost/XgBoost.RData")
  tm_xgboost_test <- system.time(xgboost.pred <- predict(fit_model, dat_test))
  save(xgboost.pred, file="../output/xgboost.pred.RData")
  cat("Time for testing model=", tm_xgboost_test[1], "s \n")
}
if(run.adaboost){
  load(file="../lib/AdaBoost/adaboost.RData")
  tm_adaboost_test <- system.time(ada.pred <- predict(adaall, dat_test))
  save(ada.pred , file="../output/ada.pred .RData")
  cat("Time for testing model=", tm_adaboost_test[1], "s \n")
}
if(run.tree){
  load(file="../lib/ClassficationTree/tree.RData")
  tm_tree_test <- system.time(tree.pred <- predict(tree, dat_test))
  save(tree.pred, file="../output/tree.pred.RData")
  cat("Time for testing model=", tm_tree_test[1], "s \n")
}
if(run.gbm){
  load(file="../lib/GBM/GBM.RData")
  tm_gbm_test <- system.time(gbm.pred <- predict(gbm_fit, dat_test))
  save(gbm.pred, file="../output/gbm.pred.RData")
  cat("Time for testing model=", tm_gbm_test[1], "s \n")
}
if(run.lg){
  load(file="../lib/Logistic/logistic.RData")
  tm_lg_test <- system.time(lg.pred <- predict(fit, dat_test))
  save(lg.pred, file="../output/lg.pred.RData")
  cat("Time for testing model=", tm_lg_test[1], "s \n")
```

```
}
if(run.nnt){
  load(file="../lib/NeuronNetwork/NeuronNetwork.RData")
  tm_nnt_test <- system.time(nnt.pred <- predict(pr.nn, dat_test))
  save(nnt.pred, file="../output/nnt.pred.RData")
  cat("Time for testing model=", tm_nnt_test[1], "s \n")
}
if(run.rf){
  load(file="../lib/RandomForest/randomforest.RData")
  tm_rf_test <- system.time(rf.pred <- predict(rf_all, dat_test, type="response"))
  save(rf.pred , file="../output/rf.pred .RData")
  cat("Time for testing model=", tm_rf_test[1], "s \n")
}
if(run.svm){
  load(file="../lib/SVM/svm.RData")
  tm_svm_test <- system.time(svm.pred <- predict(rad_svm_fit, dat_test))
  save(svm.pred , file="../output/svm.pred.RData")
  cat("Time for testing model=", tm_svm_test[1], "s \n")
}
```