Project 3 - Dogs versus Fried Chicken versus Blueberry Muffins

Xinhu Wang, Xin Luo, Sihui Shao, Yina Wei

November 1, 2017

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

This file is currently a template for running evaluation experiments of image analysis (or any predictive modeling). You should update it according to your codes but following precisely the same structure.

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


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

library("EBImage")
library("gbm")
library("adabag")
```

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/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/")
```

```
img_train_dir <- paste("~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-projec
t3-grp9","/data/training_set/images/",sep="")
img_test_dir <- "../data/"#going to be changed when testing
```

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) run the training process
- (T/F) use the new dataset (or just original sift data)
- (T/F) use the baseline model
- (number) number of trees adaboost use

```
run.feature.train=TRUE # process features for training set
run.feature.test=FALSE # process features for test set
run.test=FALSE # run evaluation on an independent test set
run.train=TRUE #run the training process
new.features=TRUE #use the new dataset (or just original sift data)
baseline=FALSE # Use the baseline model
ada_depth=c(30,70,90)[1]
#you could change the depth of trees in adaboost, generally speaking, larger value means longe
r timse
```

Step 2: import training images class labels.

For the example of zip code digits, we code digit 9 as “1” and digit 7 as “0” for binary classification.

```
label_train <- read.csv("~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-proje
ct3-grp9/data/training_set/label_train.csv")[,2]
```

Step 3: construct visual feature

For this simple example, we use the row averages of raw pixel values as the visual features. Note that this strategy **only** works for images with the same number of rows. For some other image datasets, the feature function should be able to handle heterogeneous input images. Save the constructed features to the output subfolder.

`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 that are required by all the models you are going to evaluate later.

```
tm_feature_train <- NA
source("~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/lib/feat
ures.R")
if(!new.features){
  dat_test<-read.csv('~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3
-grp9/data/training_set/sift_train.csv')
}else{
  if(run.feature.train){

    tm_feature_train <- system.time(dat_train<- features(img_train_dir,"train"))
    save(dat_train, file="~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-proj
ect3-grp9/output/feature_train.RData")
  }else{
    load("~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/outpu
t/feature_train.RData")
  }
}
tm_feature_test <- NA
if(run.feature.test){
if(!new.features){
  dat_test<-read.csv('./data/test_set/sift_test.csv')
}else{

  tm_feature_test <- system.time(dat_test <- features(img_test_dir,"test"))
  save(dat_test, file="./output/feature_test.RData")
}
}
```

Step 4: Train a classification model with training images

Call the train model and test model from library.

```
source("~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/lib/train.R")
source("~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/lib/test.R")
```

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

```
tm_train=NA
#load('~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/output/feature_train.RData')
if(!run.train){
  load(paste0("./output/fit_train_",ada_depth,".RData",sep="" ))
}else{
  if(baseline){

    tm_train <- system.time(fit_train_base <- train_baseline(dat_train, label_train, new.features))
    save(fit_train_base, file="../output/fit_train_new.RData")
  }else{
    tm_train=NA
    tm_train <- system.time(fit_train_ada <- train_ada(dat_train, label_train,ada_depth,new.features))
    save(fit_train_ada, file="~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/output/fit_train_new.RData")
  }
}
```

Step 5: Make prediction

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

```
tm_test=NA
if(run.test){
  load(file="./output/feature_test.RData")
  if(baseline){
    tm_test <- system.time(pred_test <- test_baseline(fit_train_ada, dat_test,new.features))
    save(pred_test, file="../output/pred_test.RData")
  }else{
    tm_test <- system.time(pred_test <- test_ada(fit_train_base, dat_test,new.features))
    save(pred_test, file="~/Desktop/[ADS]Advanced Data Science/Fall2017-project3-fall2017-project3-grp9/output/pred_test.RData")
  }
}
```

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= 639.574 s
```

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

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

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

```
## Time for training model= 1924.125 s
```

```
cat("Time for making prediction=", tm_test[1], "s \n")
```

```
## Time for making prediction= NA s
```