Results on the training set provided at the begining

Group2

March 21, 2018

Step 0: Prepare all needed packages

```
need.packages <- c("gbm","EBImage","xgboost","OpenImageR", "dplyr", "grDevices", "ggplot2", "adabag")
new.packages <- need.packages[!(need.packages %in% installed.packages()[,"Package"])]
if(length(new.packages))
  {
   install.packages(new.packages)
   source("https://bioconductor.org/biocLite.R")
   biocLite("EBImage")
  }
library("gbm")
```

```
## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```
library("xgboost")
library("EBImage")
library("xgboost")
library("OpenImageR")
```

```
##
## Attaching package: 'OpenImageR'

## The following objects are masked from 'package:EBImage':
##
##     readImage, writeImage
```

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:EBImage':
##
##     combine

## The following object is masked from 'package:xgboost':
##
##     slice

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union
```

```r
library("grDevices")
library("ggplot2")
```

```r
# loading all needed functions
source("../lib/feature_all.R")
source("../lib/Train_final.R")
```

```
## Loading required package: rpart

## Loading required package: caret

##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster

## Loading required package: foreach

## Loading required package: doParallel

## Loading required package: iterators
```

```r
source("../lib/Test.R")
source("../lib/CV.R")
```

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

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

Step 2: Construct new visual features

```r
img_dir<- "../data/train/images/" # The fold with training images

# rgb
if(run.feature.train)
{
  tm_rgb <- system.time(rgb_feature <- feature_rgb(img_dir, export=TRUE))
}
cat("Time for constructing RGB feature is ", tm_rgb[3], "s \n")
```

```
## Time for constructing RGB feature is  911.326 s
```

```r
# hog
if(run.feature.train)
{
  tm_hog <- system.time(hog_feature <- feature_hog(img_dir, export=TRUE))
}
cat("Time for constructing HOG feature is ", tm_hog[3], "s \n")
```

```
## Time for constructing HOG feature is  337.592 s
```

```r
# sift
train.sift <- read.csv("../data/train/SIFT_train.csv", header = F) # Load SIFT features
train.sift <- train.sift[, -1]

# sift(PC)
if(run.feature.train)
{
  tm_pca_sift <- system.time(pca_sift_feature <- feature_pca(train.sift))
}
cat("Time for constructing resized SIFT feature is ", tm_pca_sift[3], "s \n")
```

```
## Time for constructing resized SIFT feature is  53.683 s
```

```r
# sift(PC) + rgb + hog
train.rsh <- cbind(rgb_feature, hog_feature, pca_sift_feature)
```

Step 3: Models Training and Parameters Selection

```r
# labels of training data
train.label <- read.csv("../data/train/label_train.csv", header = T)

#subsetting
test_d <- sample(1:nrow(train.rsh), 750)
rsh_train <- train.rsh[-test_d, ]
sift_train <- train.sift[-test_d, ]
label_train <- train.label[-test_d, ]
rsh_test <- train.rsh[test_d, ]
label_test <- train.label[test_d, ]
sift_test <- train.sift[test_d, ]
```

Baseline Model: GBM

Baseline Features: SIFT

Train Gradient Boosting Model using SIFT features obtained by Cross-validation.

```r
tm_gbm.sift=NA

if(run.feature.train){
  tm_gbm.sift<- system.time(fit_gbm.sift <- gbm.fit(x = data.frame(sift_train),
                                                     y = label_train[ , 3],
                     n.trees = 800,
                     distribution = "multinomial",
                     interaction.depth = 7,
                      bag.fraction = 0.5,
                      verbose = FALSE))
}
cat("Time for training the best GBM on SIFT features is ", tm_gbm.sift[3], "s \n")
```

```
## Time for training the best GBM on SIFT features is  818.518 s
```

Prefer Model: XGBoost

Prefer Features: SIFT (PC) + RGB + HoG features

Train XGBoost & SIFT (PC) + RGB + HoG features using best parameters obtained from Cross-validation

```
tm_xgb.rsh <- NA

if(run.feature.train){
tm_xgb.rsh <- system.time(fit_xgb_rsh <- xgb_train(rsh_train, label_train))
}
cat("Time for training the best XGBoost on SIFT (PC) + RGB + HoG features is ", tm_xgb.rsh[3], "s \n")
```

```
## Time for training the best XGBoost on SIFT (PC) + RGB + HoG features is  48.083 s
```

Step 4: Make Prediction on Testing Data

Baseline Model: Fit the best GBM model on testing SIFT features

```
if (ncol(label_test) != 1) {
  label_test <- label_test[,3]
} else {
  label_test <- as.numeric(as.character(label_test))
}

gbm.sift.test <- gbm_test(fit_gbm.sift, sift_test) # need to be provided
error_gbm <- mean(gbm.sift.test != label_test)
cat("The prediction error on baseline model is ", error_gbm, "s \n")
```

```
## The prediction error on baseline model is  0.2746667 s
```

Final Model: Fit the best XGBoost model on testing SIFT (PC) + RGB + HoG features

```
# if (ncol(label_test) != 1) {
#   label_test <- label_test[,3]
# } else {
#   label_test <- as.numeric(as.character(label_test))
# }

xgb.rsh.test <- xgb_test(fit_xgb_rsh, rsh_test)
error_xgb <- mean(xgb.rsh.test != label_test - 1)
cat("The prediction error on the final model is ", error_xgb, "s \n")
```

```
## The prediction error on the final model is  0.07466667 s
```