Project 3 - Applied Data Science

Grp 1

Oct 21, 2017

Load and install necessary packages.

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

```
## Loading required package: EBImage
```

```
packages.used=c("gbm", "MASS", "OpenImageR", "jpeg", "ggplot2", "reshape2", "randomForest")

# check packages that need to be installed.
packages.needed=setdiff(packages.used,
                        intersect(installed.packages()[,1],
                                  packages.used))
# install additional packages
if(length(packages.needed)>0){
  install.packages(packages.needed, dependencies = TRUE,
                   repos='http://cran.us.r-project.org')
}

## Loading packages
library("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("MASS")
library("OpenImageR")
```

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

```
library("jpeg")
library("ggplot2")
library("reshape2")
library("randomForest")
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:EBImage':
##
##     combine
```

Step 0: specify directories.

Set working directory as where this .rmd file is located.

```
#setwd("~/Desktop/Fall2017-project3-fall2017-project3-grp1-master/doc")
```

Set directories of train and test datasets.

```
experiment_dir <- "../data/training_set/" # This will be modified for different data sets.
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) 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
- (T/F) boolean variable indicating to train model or not

```
run.cv = F # run cross-validation on the training set
K <- 5  # number of CV folds
run.feature.train = T # process features for training set
run.test = TRUE # run evaluation on an independent test set
run.feature.test = TRUE # process features for test set
model.train = T
```

Boolean variables indicating which feature extraction method to perform

```
run.pca = FALSE
run.hogs = FALSE
run.cnn = FALSE
run.sift = TRUE
```

Boolean variables indicating which model to run

```
run.gbm = TRUE
run.svm = FALSE
run.rf = FALSE
run.lda = FALSE
```

Using cross-validation or independent test set evaluation, we compare the performance of different classifiers or classifiers with different specifications. In the baseline model, we use GBM with different `shrinkage` parameter values. In this chunk, we list, in a vector, setups (in this case, `depth`) corresponding to models that we will compare. Additionally, we define the parameters to be tuned in the Random Forest model.

```
model_values<-seq(0.01,0.25,0.05) # for GBM
svm_gamma_values <- seq(0, 0.5, by = 0.1) # for SVM
svm_gamma_labels = paste("SVM gamma =", svm_gamma_values) # for SVM
rf_par = expand.grid(mtry = c(10, 20, 30), ntree = c(1000, 2000) ) # for RF
```

Step 2: import training images class labels.

```
label_train <- read.csv("../data/training_set/label_train.csv", header=T)
```

Step 3: construct visual feature

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

if( !run.sift ){

  tm_feature_train <- NA
  if(run.feature.train){
    tm_feature_train <- system.time(dat_train <- feature(img_train_dir,
                                                         n_pixel_row = 300,
                                                         n_pixel_col = 300,
                                                         n_dig = 4,
                                                         n_hogs = 54,
                                                         desired_variance = 0.9,
                                                         run.pca = run.pca,
                                                         run.hogs = run.hogs,
                                                         run.cnn = run.cnn,
                                                         run.lbp = run.lbp,
                                                         export=TRUE) )
  }

  tm_feature_test <- NA
  if(run.feature.test){
    tm_feature_test <- system.time(dat_test <- feature(img_test_dir,
                                                         n_pixel_row = 300,
                                                         n_pixel_col = 300,
                                                         n_dig = 4,
                                                         n_hogs = 54,
                                                         desired_variance = 0.9,
                                                         run.pca = run.pca,
```

```
                                                    run.hogs = run.hogs,
                                                    run.cnn = run.cnn,
                                                    run.lbp = run.lbp,
                                                    export=TRUE) )
  }

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

}

if(run.sift){
  dat_test <- read.csv("../data/training_set/sift_test.csv", head = T)
  dat_test <- dat_test[,-1]
  }
```

Step 4: Train a classification model with training images

Call the train model and test model from library.

- `train.R`
- Input: a path that points to the training set features.
- Input: an R object of training sample labels.
- Input: boolean variable indicating which model to run
- 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.
- Input: boolean variable indicating which model to test
- 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.

Load 'train' and 'test' scripts.

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

load("../output/feature_train.RData")
#load("../output/feature_test.RData")
```

Model selection with cross-validation

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

# if SIFT feature extraction method selected TRUE
```

```
if( run.sift ){
  baseline_dat_train <- read.csv("../data/training_set/sift_train.csv", header=T)
  baseline_dat_train <- baseline_dat_train[,-1]
  dat_train <- baseline_dat_train
}

# Boolean variables to indicate which model to perform cross-validation.
cv.svm = F
cv.gbm = F
cv.rf = T
cv.lda = F

if(run.cv) {

  if( cv.gbm ){
    err_cv <- array(dim=c(length(model_values), 2))

    for(k in 1:length(model_values)){
      cat("k=", k, "\n")
      err_cv[k,] <- cv.function( as.data.frame(dat_train), label_train, model_values[k], K, cv.gbm = T)
     }
  }

  if( cv.svm ){

  err_cv <- array(dim=c(length(svm_gamma_values), 2))

   for(k in 1:length(svm_gamma_values)){
    cat("k=", k, "\n")
     err_cv[k,] <- cv.function( dat_train, label_train, d = svm_gamma_values[k], K = K, cv.svm = T)
   }

  }

  if( cv.rf ){
    err_cv <- array(dim=c(nrow(rf_par), 2))
    model_values = rf_par

    for(k in 1:nrow(rf_par)){
      cat("k=", k, "\n")
      err_cv[k,] <- cv.function( dat_train, label_train, rf_par[k,], K, cv.rf = T)
    }
  }

  save(err_cv, file="../output/err_cv.RData")
}
```

Visualize cross-validation results.

```
if(run.cv){

  if(cv.gbm){
```

```
    load("../output/err_cv.RData")
    #pdf("../fig/cv_results.pdf", width=7, height=5)
    plot(model_values, err_cv[,1], xlab="Interaction Depth", ylab="CV Error",
         main="Cross Validation Error", type="n", ylim=c(0, 1))
    points(model_values, err_cv[,1], col="blue", pch=16)
    lines(model_values, err_cv[,1], col="blue")
    arrows(model_values, err_cv[,1]-err_cv[,2], model_values, err_cv[,1]+err_cv[,2],
    length=0.1, angle=90, code=3)
    #dev.off()
    }

  if(cv.svm){
    load("../output/err_cv.RData")
    #pdf("../fig/cv_results.pdf", width=7, height=5)
    plot(svm_gamma_values, err_cv[,1], xlab="Interaction Depth", ylab="CV Error",
         main="Cross Validation Error", type="n", ylim=c(0, 1))
    points(svm_gamma_values, err_cv[,1], col="blue", pch=16)
    lines(svm_gamma_values, err_cv[,1], col="blue")
    arrows(svm_gamma_values, err_cv[,1]-err_cv[,2], svm_gamma_values, err_cv[,1]+err_cv[,2],
    length=0.1, angle=90, code=3)

  }
  if(cv.rf){
    load('../output/err_cv.RData')
    models <- cbind(model_values, err_cv)
    colnames(models) <- c('mtry','ntree','err','sd')
    models <- models[,-4]
    #models <- melt(models, id.vars='ntree', value.name="err", variable.name="mtry")
    ggplot(data=models, aes(x=mtry, y=err, group = as.factor(ntree), colour = as.factor(ntree))) +
    geom_line() +
    geom_point( size=4, shape=21, fill="white")
  }

}
```

- Choose the “best”" parameter value

```
if( run.cv ){

  if( !run.rf ){
    model_best=model_values[1]
  } else {
    model_best=model_values[1,]
  }

  # Best parameter for GBM
  if(run.gbm){
    model_best<- model_values[which.min(err_cv[,1])]
  }
  # Best parameter for SVM
  if(run.svm){
    model_best <- svm_gamma_values[which.min(err_cv[,1])]
  }
  # Best parameter for Ranfom Forest
```

```
  if(run.rf){
    model_best <- rf_par[which.min(err_cv[,1]),]
  }

}
```

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

```
if( !run.cv ){

  if( run.gbm){
    model_best <- 0.06
  }

  if( run.rf ){
    model_best <- list(mtry = 20, ntree = 2000)
  }

  if( run.svm ){
    model_best <- 0.1
  }

}

if( model.train ){
  # Train LDA
  if( run.lda ){
    tm_train_lda <- system.time(model.lda <- train( dat_train, label_train, run.lda = TRUE ))
    save(model.lda, file="../output/model_lda.RData")
  }

  # Train GBM
  if( run.gbm ){
    tm_train_gbm <- system.time(model.gbm <- train( as.data.frame(dat_train), label_train, params = mod
    save(model.gbm, file="../output/model_gbm.RData")
  }

  # Train SVM
  if( run.svm ){
    tm_train_svm <- system.time(model.svm <- train( dat_train, label_train, run.svm = TRUE, params = mo
    save(model.svm, file="../output/model_svm.RData")
  }

  # Train Random Forest
  if( run.rf ){
    tm_train_rf <- system.time(model.rf <- train(dat_train, label_train, model_best, run.rf = TRUE ))
    save(model.rf, file="../output/model_rf.RData")
  }
}
```

Step 5: Make prediction and Summarize running time

```
#Import Test labels
#label_test <- read.csv("../data/training_set/label_test.csv", header=T)

if( !run.sift ){
  cat("Total Time for constructing training features=", tm_feature_train[1], "s \n")

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

# LDA
if( run.lda ){
  load(file="../output/model_lda.RData")
  tm_test_lda <- system.time(pred.lda <- test(model.lda, dat_test, test.lda = T))
  lda.error <- sum((pred.lda != label_test[,2]))/(length(label_test[,2]))
  cat("Test error =", lda.error * 100, "% \n")
  if( model.train ) cat("Time for training model=", tm_train_lda[1], "s \n")
  cat("Time for testing model=", tm_test_lda[1], "s \n")
}

# GBM
if( run.gbm ){
  load(file="../output/model_gbm.RData")
  tm_test_gbm <- system.time(pred.gbm <- test(model.gbm, dat_test, test.gbm = T))
  #gbm.error <- sum((pred.gbm != label_test[,2]))/(length(label_test[,2]))
  #cat("Test error =", gbm.error * 100, "% \n")
  cat("Time for training model=", tm_train_gbm[1], "s \n")
  #cat("Time for testing model=", tm_test_gbm[1], "s \n")
}
```

```
## Time for training model= 413.75 s
```

```
# SVM
if( run.svm ){
  load(file="../output/model_svm.RData")
  tm_test_svm <- system.time(pred.svm <- test(model.svm, dat_test, test.svm = T))
  svm.error <- sum((pred.svm != label_test[,2]))/(length(label_test[,2]))
  cat("Test error =", svm.error * 100, "% \n")
  cat("Time for training model=", tm_train_svm[1], "s \n")
  cat("Time for testing model=", tm_test_svm[1], "s \n")
}

# Ranfom Forest
if( run.rf ){
  load(file="../output/model_rf.RData")
  tm_test_rf <- system.time(pred.rf <- test(model.rf, dat_test, test.rf = T))
  rf.error <- sum((pred.rf != label_test[,2]))/(length(label_test[,2]))
  cat("Test error =", rf.error * 100, "% \n")
  if( model.train ) cat("Time for training model=", tm_train_rf[1], "s \n")
  cat("Time for testing model=", tm_test_rf[1], "s \n")
}
```

```r
write.csv(pred.gbm, "../output/baseline_predictions.csv")
```