Group 9 _Project 3: Predictive Modeling

Chen Wang, Xin Gao, Kanyan Chen, Haoyu Zhang, Zack Abrams

10/30/2019

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

if(!require("dplyr")){
  install.packages("dplyr")
}
if(!require("readxl")){
  install.packages("readxl")
}

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

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

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

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

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

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

if(!require("e1071")){
  install.packages("e1071")
}

if(!require("gbm")){
```

```
  install.packages("gbm")
}

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

if(!require("h2o")){
  install.packages("h2o")
}

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

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

library(kernlab)
library(R.matlab)
library(readxl)
library(dplyr)
library(EBImage)
library(caret)
library(xgboost)
library(readr)
library(stringr)
library(car)
library(e1071)
library(gbm)
library(h2o)
library(mlr)
library(randomForest)
```

Step 0 set work directories, extract paths

```
set.seed(0)
setwd("/Users/Chen/Desktop/GR5243/fall2019-proj3-sec2--grp9/")

train_dir <- "../train_set/"
train_image_dir <- paste(train_dir, "images/", sep="")
train_pt_dir <- paste(train_dir,  "points/", sep="")
train_label_path <- paste(train_dir, "label.csv", sep="")
```

```
# this chunck is set to (eval = FALSE), where we construct our features from 6006 to 92
info <- read.csv("../train_set/label.csv")
info$emotion_idx = as.factor(info$emotion_idx)
n <- nrow(info)
n_train <- round(n*(4/5), 0)
train_idx <- sample(info$Index, n_train, replace = F)
test_idx <- setdiff(info$Index,train_idx)
```

```
readMat.matrix <- function(index){
     return(round(readMat(paste0(train_pt_dir, sprintf("%04d", index), ".mat"))[[1]],0))
}
fiducial_pt_list <- lapply(1:2500, readMat.matrix)
```

Step1 Feature Selection

```
# this chunck is set to (eval = FALSE), where we construct our features from 6006 to 92
start.time <- Sys.time()
xi = c(rep(1,8),rep(10,8),rep(25,7),rep(33,7),rep(37,14),rep(59,13),rep(37,8),rep(71,6),44,56,4,13,6,11
xj = c(2:9,11:18,19:24,26:32,34:36,38:58,60:67,75:78,68:70,72:74,52,71,25,33,39,49)
xi = rep(xi,2)
xj = rep(xj,2)
xoy = c(rep(1,77),rep(2,77))
df_label = data.frame(xi,xj,xoy)
shape_matrix = matrix(nrow = 2500,ncol = 154)
for(row in 1:2500){
  for(k in 1:154){
    shape_matrix[row,k] = fiducial_pt_list[row][[1]][df_label[k,1],df_label[k,3]] - fiducial_pt_list[ro
  }
}
```

```
select_feature = c(1,5,9,11:14,17:20,25,26,34,36,38,40,44,45,49,50,52:55,57,67:69,72:77,79,80,84,87:89,
end.time <- Sys.time()
end.time - start.time
train_X = shape_matrix[train_idx,select_feature]
train_y = info$emotion_idx[train_idx]
test_X = shape_matrix[test_idx,select_feature]
test_y = info$emotion_idx[test_idx]
dat_tr <- data.frame(train_X, train_y)
dat_test <- data.frame(test_X, test_y)
save(dat_tr, file="../output/train.RData")
save(dat_test, file="../output/test.RData")
```

Since we used the same features for both the training and testing data, the feature selection time in total is 33.46879 seconds.

Step2 Train and test a classification model with training features and responses

1. Baseline: GBM

```
set.seed(0)
load("../output/train.RData")
load("../output/test.RData")
train_X <- as.matrix(dat_tr[,-93])
test_X <- as.matrix(dat_test[,-93])
train_y <- dat_tr$train_y
test_y <- dat_test$test_y
```

```
tm.train <- system.time(mod_gbm <- gbm(train_y ~.,
              data = dat_tr,
              distribution = "multinomial",
              cv.folds = 5,
              shrinkage = 0.1,
              n.minobsinnode = 10,
              interaction.depth = 1,
              n.trees = 200))

tm.train
```

```
##    user  system elapsed
##  11.599   0.202  50.927
```

```
#train on the training set
set.seed(0)
pred.train <- predict.gbm(object = mod_gbm,
                  newdata = dat_tr,
                  n.trees = 200,
                  type = "response")
emotion.train <- colnames(pred.train)[apply(pred.train,1,which.max)]
accuracy.train <- sum(emotion.train == train_y)/2000
accuracy.train
```

```
## [1] 0.8225
```

```
set.seed(0)
tm.test <-system.time(pred <- predict.gbm(object = mod_gbm,
                  newdata = dat_test,
                  n.trees = 200,
                  type = "response"))
tm.test
```

```
##    user  system elapsed
##   0.036   0.001   0.036
```

```
emotion.pred <- colnames(pred)[apply(pred, 1, which.max)]
accuracy <- sum(emotion.pred==test_y)/500
accuracy
```

```
## [1] 0.462
```

The baseline model has the following results: for the training part, the user time is 11.249 seconds, with the training accuracy of 82.25%; for the testing part, the user time is 0.037 seconds, with the testing accuracy of 46.2%.

2 Improved method: KSVM

```
set.seed(0)
tm.clf <- system.time(clf <- ksvm(train_X,
               train_y,kernel="rbfdot",
               kpar=list(sigma=0.0005),
               cross=5,C = 50))
tm.clf
```

```
##    user  system elapsed
##  10.802   1.598  12.722
```

```
set.seed(0)
tm.clf.test <- system.time(test <- predict(clf,test_X))
tm.clf.test
```

```
##    user  system elapsed
##   0.690   0.147   0.843
```

```
accuracy.ksvm <- sum(test==test_y)/500
accuracy.ksvm
```

```
## [1] 0.52
```

For the final improved method using SVM, the training user time is 8.779 seconds, the testing time is 0.744 seconds, and testing accuracy is 52%.

3. Random Forest (this is a method we tried, but not the final improved method we picked)

```
# Random forest model:

traintask <- makeClassifTask(data = dat_tr,target = "train_y")
testtask <- makeClassifTask(data = dat_test,target = "test_y")
rf.lrn <- makeLearner("classif.randomForest")
rf.lrn$par.vals <- list(ntree = 100L, importance=TRUE)
rdesc <- makeResampleDesc("CV",iters=5L)
r <- resample(learner = rf.lrn, task = traintask, resampling = rdesc, measures = list(acc), show.info =
```

```
## Resampling: cross-validation
## Measures:             acc
## [Resample] iter 1:    0.4250000
## [Resample] iter 2:    0.4525000
## [Resample] iter 3:    0.3650000
## [Resample] iter 4:    0.4025000
## [Resample] iter 5:    0.4550000
##
## Aggregated Result: acc.test.mean=0.4200000
##
```

```
params <- makeParamSet(makeIntegerParam("mtry",lower = 10,upper = 50),makeIntegerParam("nodesize",lower
ctrl <- makeTuneControlRandom(maxit = 5L)
#tune parameters
tune <- tuneParams(learner = rf.lrn, task = traintask, resampling = rdesc, measures = list(acc), par.se
```

```
## [Tune] Started tuning learner classif.randomForest for parameter set:
##             Type len Def   Constr Req Tunable Trafo
## mtry     integer   -   - 10 to 50   -    TRUE     -
## nodesize integer   -   - 10 to 50   -    TRUE     -
```

```
## With control class: TuneControlRandom
## Imputation value: -0
## [Tune-x] 1: mtry=44; nodesize=20
## [Tune-y] 1: acc.test.mean=0.4125000; time: 0.2 min
## [Tune-x] 2: mtry=43; nodesize=12
## [Tune-y] 2: acc.test.mean=0.4295000; time: 0.2 min
## [Tune-x] 3: mtry=41; nodesize=20
## [Tune-y] 3: acc.test.mean=0.4130000; time: 0.2 min
## [Tune-x] 4: mtry=23; nodesize=39
## [Tune-y] 4: acc.test.mean=0.3990000; time: 0.1 min
## [Tune-x] 5: mtry=41; nodesize=25
## [Tune-y] 5: acc.test.mean=0.4145000; time: 0.2 min
## [Tune] Result: mtry=43; nodesize=12 : acc.test.mean=0.4295000
```

```
#the best is mtry=36; nodesize=19 acc.test.mean=0.4240000

#control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
#set.seed(0)
#metric <- "Accuracy"
#tunegrid <- expand.grid(.mtry=c(20:50))
#rf_gridsearch <- train(as.factor(train_y)~., data=dat_tr1, method="rf", metric=metric, tuneGrid=tunegr
#print(rf_gridsearch)
#plot(rf_gridsearch)



model2 <- randomForest(as.factor(train_y) ~ ., data = dat_tr, ntree = 65, mtry = 30, importance = TRUE)
model2
```

```
##
## Call:
##  randomForest(formula = as.factor(train_y) ~ ., data = dat_tr,      ntree = 65, mtry = 30, importanc
##                Type of random forest: classification
##                      Number of trees: 65
## No. of variables tried at each split: 30
##
##         OOB estimate of  error rate: 60.35%
## Confusion matrix:
##     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
## 1  52  0 16  5  0  2  2  0  0  3  0  1  2  1  3  7  0  0  1  0  0  1
## 2   0 75  0  0  0  0  0  8 12  0  0  0  0  0  0  0  0  0  0  2  0  0
## 3  16  0 53  5  0  3  0  0  0 12  0  4  6  0  1  5  0  0  0  0  1  4
## 4   4  0  7 48  0  4  0  0  0  9  4 11 10  0  0  0  0  0  0  0  0  2
## 5   0  0  0  0 46  0  4  3  0  0  0  0  0  7  7  0  5  6  0  1  1  0
## 6   2  0  7 13  0 25  1  0  1  1  4 13  4  4  0  4  0  0  0  1  2  1
## 7   4  3  0  1  7  1 30  5  1  0  0  0  1  6  1  3  9  4  5  3  4  3
## 8   0 13  0  0  6  0  3 59  2  0  0  0  0  1  1  0  6  2  0  0  0  0
## 9   0 19  0  0  0  2  3  1 46  1  1  0  0  0  0  0  0  0  0  4  3  1
```

```
## 10  8  0 19 13  0  3  0  0  0 25  5  4 12  0  0  1  0  0  0  0  1  0
## 11  2  0  5 13  1  7  0  0  1  7 30 17  7  1  0  0  0  0  0  0  1  3
## 12  1  0  1  7  0  9  1  0  1  2  9 21 20  3  2  5  0  0  0  1  0  1
## 13  3  0  7 12  0  5  0  0  2  7  4 18 21  0  3  2  0  0  0  0  1  2
## 14  0  1  0  0  3  1  1  0  0  0  0  0  0 56 16  0  3  2  1  3 12  1
## 15  1  0  1  0  4  1  2  0  1  0  0  0  0 27 29  0  1  0  5  3  3  0
## 16  9  0  4  3  0  0  0  1  0  2  1  2  3  1  0 56  1  2  1  0  1  2
## 17  0  0  0  0  5  0  6 10  0  0  0  0  0  8  2  1 32 25  4 10  1  0
## 18  2  0  2  0  9  0  3  0  0  1  0  0  0  2  3  1 31 24  3  3  2  1
## 19  2  2  2  0  4  0  5  1  3  1  0  0  0  7  7  1 10  6  7  7 12  3
## 20  0  4  0  0  1  2  5  5  5  0  0  0  0 11  3  0 11  4  3 21 15  3
## 21  1  4  1  1  0  0  1  1  8  0  0  0  1 17  4  1  2  0 14 11 24  3
## 22  3  3 13  6  0  2  0  0  2  5  1  6  2  5  1  1  2  1  2 10 10 13
##    class.error
## 1    0.4583333
## 2    0.2268041
## 3    0.5181818
## 4    0.5151515
## 5    0.4250000
## 6    0.6987952
## 7    0.6703297
## 8    0.3655914
## 9    0.4320988
## 10   0.7252747
## 11   0.6842105
## 12   0.7500000
## 13   0.7586207
## 14   0.4400000
## 15   0.6282051
## 16   0.3707865
## 17   0.6923077
## 18   0.7241379
## 19   0.9125000
## 20   0.7741935
## 21   0.7446809
## 22   0.8522727
```

```
predTrain<-predict(model2, data=dat_tr,type="class")
sum(predTrain == dat_tr$train_y)/length(dat_tr$train_y)
```

```
## [1] 0.3965
```

```
predValid <- predict(model2, dat_test, type = "class")
sum(predValid == dat_test$test_y)/length(dat_test$test_y)
```

```
## [1] 0.458
```

```
rftime_train<- system.time(model2 <- randomForest(train_y ~ ., data = dat_tr, ntree = 70, mtry = 36, imp
rftime_test<- system.time(predValid <- predict(model2, dat_test, type = "class"))
rftime_train
```

```
##    user  system elapsed
##   3.438   0.016   3.469
```

```
rftime_test
```

```
##    user  system elapsed
```

```
##   0.012   0.000   0.013
```

For the random forest method, the training time is 3.503 seconds, with the training accuracy of 39.65%. The testing time is 0.012 seconds, with the testing accuracy of 45.8%. (with the aggregated CV result of 42%)

test prediction in class

```
test_dir <- "../test_set_sec2/"
test_pt_dir <- paste(test_dir,  "points/", sep="")
info.test <- read.csv("../test_set_sec2/labels_prediction.csv")

readMat.matrix1 <- function(index){
     return(round(readMat(paste0(test_pt_dir, sprintf("%04d", index), ".mat"))[[1]],0))
}
fiducial_pt_list1 <- lapply(1:2500, readMat.matrix1)

xi = c(rep(1,8),rep(10,8),rep(25,7),rep(33,7),rep(37,14),rep(59,13),rep(37,8),rep(71,6),44,56,4,13,6,11
xj = c(2:9,11:18,19:24,26:32,34:36,38:58,60:67,75:78,68:70,72:74,52,71,25,33,39,49)
xi = rep(xi,2)
xj = rep(xj,2)
xoy = c(rep(1,77),rep(2,77))
df_label = data.frame(xi,xj,xoy)
shape_matrix1 = matrix(nrow = 2500,ncol = 154)
for(row in 1:2500){
  for(k in 1:154){
    shape_matrix1[row,k] = fiducial_pt_list1[row][[1]][df_label[k,1],df_label[k,3]] - fiducial_pt_list1
  }
}

select_feature = c(1,5,9,11:14,17:20,25,26,34,36,38,40,44,45,49,50,52:55,57,67:69,72:77,79,80,84,87:89,9
test_X = shape_matrix1[,select_feature]

set.seed(0)
tm.clf.test.test <- system.time(test1 <- predict(clf,test_X))
tm.clf.test.test


set.seed(0)
test_X.df <- data.frame(test_X)
tm.test <-system.time(pred.test <- predict.gbm(object = mod_gbm,
                   newdata = test_X.df,
                   n.trees = 200,
                   type = "response"))
tm.test
emotion.pred.test <- colnames(pred.test)[apply(pred.test, 1, which.max)]

index.test <- info.test[,1]
final <- cbind(index.test,test1,emotion.pred.test)
colnames(final) <- c("Index","Baseline","Advanced")
write.csv(final,file = "labels_prediction_grp9.csv",row.names = FALSE)
```