Project 3

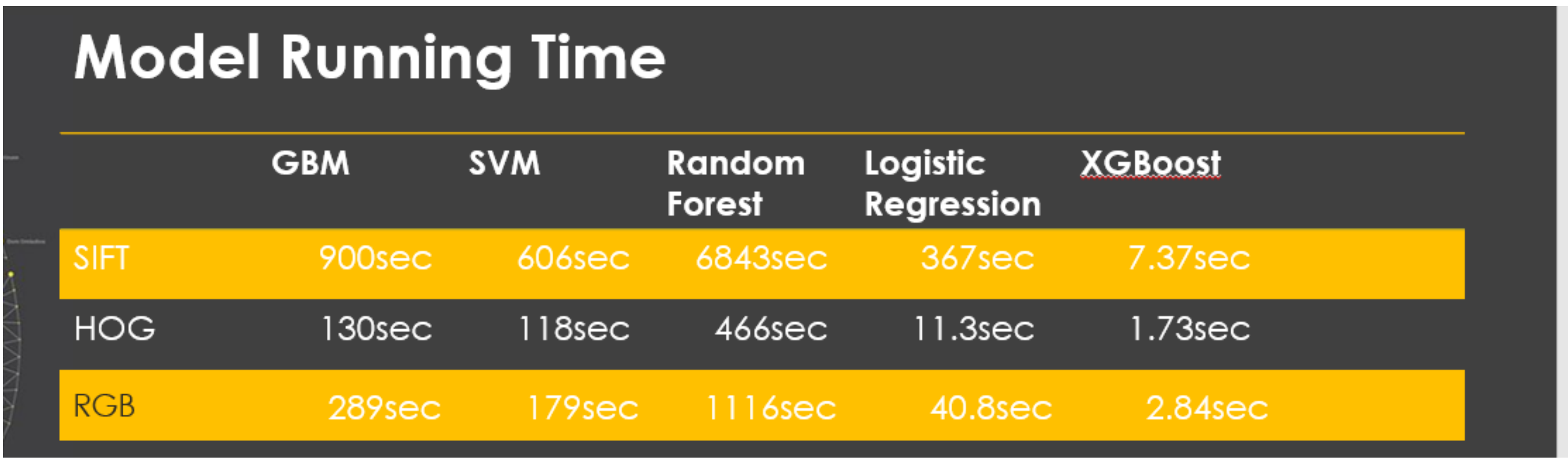
Group 6

2018/3/27

Topic: Dogs, Fried Chicken or Blueberry Muffins?

Project summary:

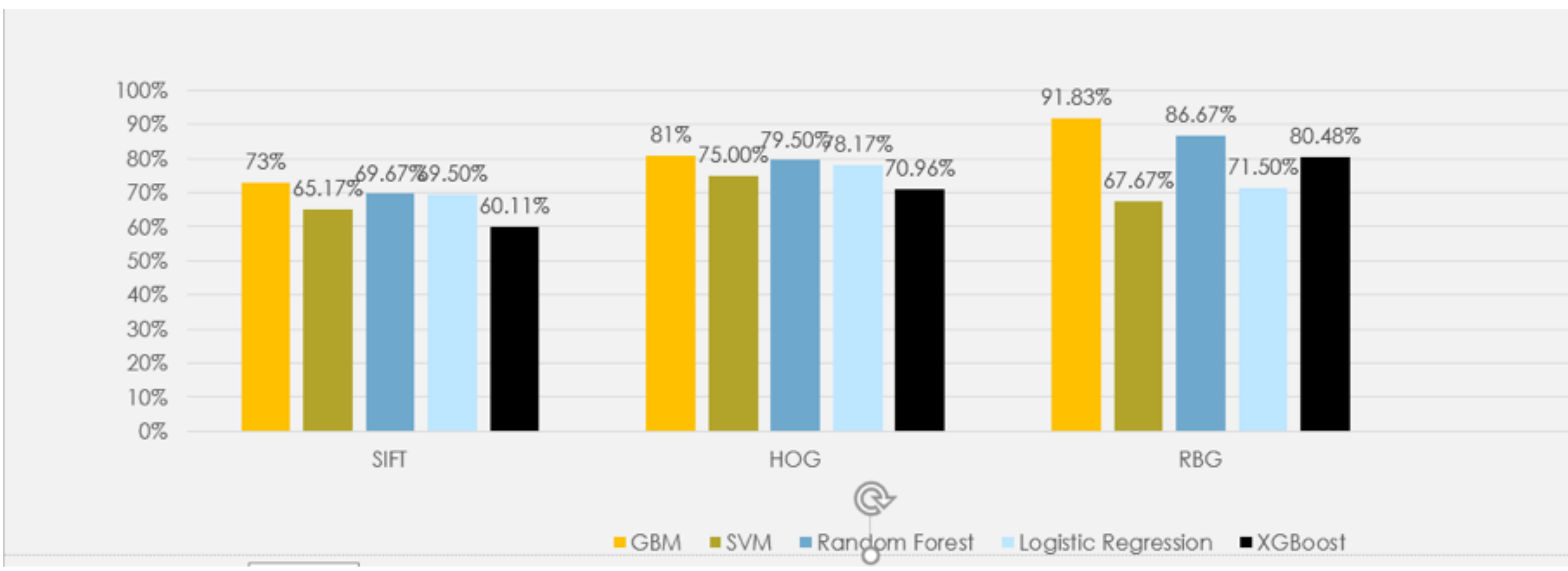
In this project, we created a classification engine for images of dogs versus fried chicken versus blueberry muffins. We tried classifiers (GBM, Logistic Regression, SVM, Random Forest, XGboost and Neural Networks) under different feature extraction(Sift, HoG, RGB). By comparing the accurancy rate as well as the processing time, we finally chose the best classification method.



Model Running Time

	GBM	SVM	Random Forest	Logistic Regression	XGBoost
SIFT	900sec	606sec	6843sec	367sec	7.37sec
HOG	130sec	118sec	466sec	11.3sec	1.73sec
RGB	289sec	179sec	1116sec	40.8sec	2.84sec

Time consuming



Accuracy

Step 0. Install, Load packages

Because of using the package caret, when we trained model we also perform cross validation by using the `tuneGrid` and `traincontrol` arguement, so we ignore creating cross validation R file.

Hide

Hide

```
packages.used=c("caret","gbm","EBImage","e1071", "DMwR", "nnet", "randomForest","O
penImageR","DT", "caTools", "pbapply", "ggthemes","xgboost")
packages.needed=setdiff(packages.used,
intersect(installed.packages()[,1],
packages.used))
if(length(packages.needed)>0){
 install.packages(packages.needed, dependencies = TRUE,repos = "http://cran.us.r-p
roject.org")
}
library(caret)
```

```
载入需要的程辑包：lattice
载入需要的程辑包：ggplot2
```

Hide

Hide

```
library(gbm)
```

```
载入需要的程辑包：survival

载入程辑包：‘survival’

The following object is masked from ‘package:caret’:

    cluster

载入需要的程辑包：splines
载入需要的程辑包：parallel
Loaded gbm 2.1.3
```

Hide

Hide

```
library(EBImage)
library(caret)
library(gbm)
library(e1071)
library(DMwR)
```

```
载入需要的程辑包：grid
```

Hide

Hide

```
library(randomForest)
```

```
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.

载入程辑包：‘randomForest’

The following object is masked from ‘package:EBImage’:

    combine

The following object is masked from ‘package:ggplot2’:

    margin
```

Hide

Hide

```
library(nnet)
library(OpenImageR)
```

```
载入程辑包：‘OpenImageR’

The following objects are masked from ‘package:EBImage’:

    readImage, writeImage
```

Hide

Hide

```
library(DT)
library(caTools)
library(EBImage)
library(pbapply)
library(ggthemes)
source("../lib/train.R")
source("../lib/test.R")
source("../lib/data_split.R")
```

Step 1. Model Comparsion Based on SIFT Feature

Step 1.1. Load Feature

We devided the whole training set into ‘df_train’-training data (80%) & ‘df_test’-testing data (20%)

Hide

Hide

```
datasplit_sift <- data_split("SIFT")
train_sift<- datasplit_sift$df_train
test_sift <- datasplit_sift$df_test
```

Step 1.2. Baseline-GBM (GBM+SIFT)

Hide

Hide

```
load("../output/baseline.result.RData")
baseline.time <- baseline.result$time
baseline.time
```

```
Time difference of 16.51173 mins
```

Hide

Hide

```
baseline.test.result <- test_gbm(baseline.result, datasplit_sift$df_test)
baseline.test.accuracy <- 1 - mean(baseline.test.result != datasplit_sift$df_test[
,1])
baseline.test.accuracy
```

```
[1] 0.73
```

Step 1.3. SVM (SVM + SIFT)

Step 1.3.1 Training Process of SVM model

Hide

Hide

```
#svm_SIFT.result <- train_svm(SIFT_train)
#save(svm_SIFT.result,file="../output/svm_SIFT.result.RData")
```

Step 1.3.2 Test of SVM model

Hide

Hide

```
load("../output/svm_SIFT.result.RData")
svm_SIFT.result.time <- svm_SIFT.result$time
svm_SIFT.result.time
```

```
Time difference of 10.12355 mins
```

Hide

Hide

```
svm_SIFT.test.result <- test(svm_SIFT.result, datasplit_sift$df_test)
svm_SIFT.test.accuracy <- 1 - mean(svm_SIFT.test.result != datasplit_sift$df_test[
,1])
svm_SIFT.test.accuracy
```

```
[1] 0.6516667
```

Step 1.4. Random Forest (Random Forest + SIFT)

Step 1.4.1 Training Process of SVM model

Hide

Hide

```
#rf_SIFT.result <- train_rf(SIFT_train)
#save(rf_SIFT.result,file="../output/rf_SIFT.result.RData")
```

Step 1.4.2 Test of Random Forest Model

Hide

Hide

```
load("../output/rf_SIFT.result.RData")
rf_SIFT.result.time <- rf_SIFT.result$time
rf_SIFT.result.time
```

```
Time difference of 114.05 mins
```

Hide

Hide

```
rf_SIFT.test.result <- test(rf_SIFT.result, datasplit_sift$df_test)
rf_SIFT.test.accuracy <- 1 - mean(rf_SIFT.test.result != datasplit_sift$df_test[,1
])
rf_SIFT.test.accuracy
```

```
[1] 0.6966667
```

Step 1.5. Logistic Regression (Logistic Regression + SIFT)

Step 1.5.1 Training Process of Logistic Regression model

Hide

Hide

```
#lr_SIFT.result <- train_lr(SIFT_train)
#save(lr_SIFT.result,file="../output/lr_SIFT.result.RData")
```

Step 1.5.2 Test of Logistic Regression Model

Hide

Hide

```
load("../output/lr_SIFT.result.RData")
lr_SIFT.result.time <- lr_SIFT.result$time
lr_SIFT.result.time
```

```
Time difference of 6.134673 mins
```

Hide

Hide

```
lr_SIFT.test.result <- test(lr_SIFT.result, datasplit_sift$df_test)
lr_SIFT.test.accuracy <- 1 - mean(lr_SIFT.test.result != datasplit_sift$df_test[,1
])
lr_SIFT.test.accuracy
```

```
[1] 0.695
```

Step 1.6. XGBoost (XGBoost + SIFT)

Hide

Hide

```
library(xgboost)
df <- read.csv('../data/SIFT_train.csv', header=FALSE)
labels <- read.csv('../data/label_train.csv')
df$label <- as.factor(labels$label)
df$V1 <- NULL
# Relabel factors for XGBoost specific num_classes requirement
levels(df$label)[levels(df$label)=="1"] <- "0"
levels(df$label)[levels(df$label)=="2"] <- "1"
levels(df$label)[levels(df$label)=="3"] <- "2"
#XGBoost Algorithm
set.seed(031918)
test.i <- sample(1:nrow(df), .3*nrow(df), replace=FALSE)
test.data <- df[test.i,]
train.data <- df[-test.i,]
target.i <- which(colnames(df) == 'label')
train.data <- df[-test.i, -target.i]
train.target <- df[-test.i, target.i]
t1=Sys.time()
model <- xgb.cv(data = as.matrix(train.data), label = train.target, nfold=10,
                 nrounds = 2, objective = "multi:softmax", num_class = 4)
```

```
[1] train-merror:0.255025+0.008268  test-merror:0.459027+0.034534
[2] train-merror:0.181903+0.007269  test-merror:0.406677+0.036113
```

Hide

Hide

```
1-model$evaluation_log$test_merror_mean[2]
```

```
[1] 0.5933229
```

Hide

Hide

```
t2=Sys.time()
t2-t1
```

```
Time difference of 9.280802 secs
```

Hide

Hide

```
load("../output/XGBoost_SIFT.result.RData")
XGBoost_SIFT.test.accuracy <- 1-model$evaluation_log$test_merror_mean[2]
XGBoost_SIFT.test.accuracy
```

```
[1] 0.5933229
```

Hide

Hide

```
model$time
```

```
Time difference of 9.280802 secs
```

Step 2. Model Comparsion Based on HOG Feature

Step 2.1 Retrieve and split the training and test data from the dataset

Hide

Hide

```
datasplit_hog <- data_split("hog_extraction1")
train_hog <- datasplit_hog$df_train
test_hog <- datasplit_hog$df_test
```

Step 2.2 GBM (GBM + HOG)

Step 2.2.1 Training Process of SVM model

Hide

Hide

```
# GBM_hog <- train_gbm(train_hog)
# save(GBM_hog,file="../output/GBM_hog.RData")
```

Step 2.2.2 Test of GBM Model

Hide

Hide

```
load("../output/GBM_hog.RData")
GBM_hog.time <- GBM_hog$time
GBM_hog.time
```

```
Time difference of 2.172558 mins
```

Hide

Hide

```
GBM.test.result_hog <- test_gbm(GBM_hog, test_hog)
GBM.test.accuracy_hog <- mean(GBM.test.result_hog == test_hog[,1])
GBM.test.accuracy_hog
```

```
[1] 0.8083333
```

Step 2.3 SVM (SVM + HOG)

Step 2.3.1 Training Process of SVM model

Hide

Hide

```
# SVM_hog <- train_svm(train_hog)
# save(SVM_hog,file="../output/SVM_hog.RData")
```

Step 2.3.2 Test of SVM Model

Hide

Hide

```
load("../output/SVM_hog.RData")
SVM_hog.time <- SVM_hog$time
SVM_hog.time
```

```
Time difference of 1.974376 mins
```

Hide

Hide

```
SVM.test.result_hog <- test(SVM_hog, test_hog)
SVM.test.accuracy_hog <- mean(SVM.test.result_hog == test_hog[,1])
SVM.test.accuracy_hog
```

```
[1] 0.75
```

Step 2.4 Random Forest (Random Forest + HOG)

Step 2.4.1 Training Process of Random Forest model

Hide

Hide

```
# RF_hog <- train_rf(train_hog)
# save(RF_hog,file="../output/RF_hog.RData")
```

Step 2.4.2 Test of Random Forest Model

Hide

Hide

```
load("../output/RF_hog.RData")
RF_hog.time <- RF_hog$time
RF_hog.time
```

```
Time difference of 7.774015 mins
```

Hide

Hide

```
RF.test.result_hog <- test(RF_hog, test_hog)
RF.test.accuracy_hog <- mean(RF.test.result_hog == test_hog[,1])
RF.test.accuracy_hog
```

```
[1] 0.7966667
```

Step 2.5 Logistic Regression (Logistic Regression + HOG)

Step 2.5.1 Training Process of Logistic Regression model

Hide

Hide

```
# LR_hog <- train_lr.cv(train_hog)
# save(LR_hog,file="../output/LR_hog.RData")
```

Step 2.5.2 Test of Logistic Regression Model

Hide

Hide

```
load("../output/LR_hog.RData")
LR_hog.time <- LR_hog$time
LR_hog.time
```

```
Time difference of 11.31186 secs
```

Hide

Hide

```
LR.test.result_hog <- test(LR_hog, test_hog)
LR.test.accuracy_hog <- mean(LR.test.result_hog == test_hog[,1])
LR.test.accuracy_hog
```

```
[1] 0.78
```

Step 2.6. XGBoost (XGBoost + HOG)

Hide

Hide

```
# the procedure would be as the same as the SIFT feature part but change the featu
re file int othe HOG features
library(xgboost)
df <- read.csv('../output/hog_extraction1.csv', header=FALSE)
labels <- read.csv("../data/label_train.csv")
df$label <- as.factor(labels$label)
df$V1 <- NULL
# Relabel factors for XGBoost specific num_classes requirement
levels(df$label)[levels(df$label)=="1"] <- "0"
levels(df$label)[levels(df$label)=="2"] <- "1"
levels(df$label)[levels(df$label)=="3"] <- "2"
# XGBoost Algorithm
set.seed(031918)
test.i <- sample(1:nrow(df), .3*nrow(df), replace=FALSE)
test.data <- df[test.i,]
train.data <- df[-test.i,]
target.i <- which(colnames(df) == 'label')
train.data <- df[-test.i, -target.i]
train.target <- df[-test.i, target.i]
t1=Sys.time()
model2 <- xgb.cv(data = as.matrix(train.data), label = train.target, nfold=10,
                 nrounds = 2, objective = "multi:softmax", num_class = 4)
```

```
[1] train-merror:0.110582+0.004262  test-merror:0.314257+0.033457
[2] train-merror:0.074234+0.003477  test-merror:0.275672+0.029517
```

Hide

Hide

```
t2=Sys.time()
t2-t1
```

```
Time difference of 2.460956 secs
```

Hide

Hide

```
model2$time = t2-t1
save(model2,file="../output/XGBoost_HOG.result.RData")
```

Hide

Hide

```
load("../output/XGBoost_HOG.result.RData")
XGBoost_HOG.test.accuracy <- 1-model2$evaluation_log$test_merror_mean[2]
XGBoost_HOG.test.accuracy
```

```
[1] 0.7243284
```

Hide

Hide

```
model2$time
```

```
Time difference of 2.460956 secs
```

Step 3. Model Comparsion Based on RGB Feature

Step 3.1 Retrieve and split the training and test data from the dataset

Hide

Hide

```
datasplit_rgb <- data_split("rgb_feature")
train_rgb <- datasplit_rgb$df_train
test_rgb <- datasplit_rgb$df_test
```

Step 3.2 GBM (GBM + RBG)

Hide

Hide

```
# GBM_rgb <- train_gbm(train_rgb)
# save(GBM_rgb,file="../output/GBM_rgb.RData")
```

Hide

Hide

```
load("../output/GBM_rgb.RData")
GBM_rgb.time <- GBM_rgb$time
GBM_rgb.time
```

```
Time difference of 4.8336 mins
```

Hide

Hide

```
GBM.test.result_rgb <- test_gbm(GBM_rgb, test_rgb)
GBM.test.accuracy_rgb <- mean(GBM.test.result_rgb == test_rgb[,1])
GBM.test.accuracy_rgb
```

```
[1] 0.9183333
```

Step 3.3 SVM (SVM + RGB)

Hide

Hide

```
# svm_rgb.result <- train_svm(train_rgb)
# save(svm_rgb.result,file="../output/svm_rgb.result.RData")
```

Hide

Hide

```
load("../output/svm_rgb.result.RData")
svm_rgb.result.time <- svm_rgb.result$time
svm_rgb.result.time
```

```
Time difference of 2.991129 mins
```

Hide

Hide

```
svm_rgb.test.result <- test(svm_rgb.result, test_rgb)
svm_rgb.test.accuracy <- 1 - mean(svm_rgb.test.result != test_rgb[,1])
svm_rgb.test.accuracy
```

```
[1] 0.6766667
```

Step 3.4 Random Forest (Random Forest + RGB)

Hide

Hide

```
# rf_rgb.result <- train_rf(train_rgb)
# save(rf_rgb.result,file="../output/rf_rgb.result.RData")
```

Hide

Hide

```
load("../output/rf_rgb.result.RData")
rf_rgb.result.time <- rf_rgb.result$time
rf_rgb.result.time
```

```
Time difference of 18.63612 mins
```

Hide

Hide

```
rf_rgb.test.result <- test(rf_rgb.result, test_rgb)
rf_rgb.test.accuracy <- 1 - mean(rf_rgb.test.result != test_rgb[,1])
rf_rgb.test.accuracy
```

```
[1] 0.8666667
```

Step 3.5 Logistic Regression (Logistic Regression + RGB)

Hide

Hide

```
# lr_rgb.result <- train_lr.cv(train_rgb)
# save(lr_rgb.result,file="../output/lr_rgb.result.RData")
```

Hide

Hide

```
load("../output/lr_rgb.result.RData")
lr_rgb.result.time <- lr_rgb.result$time
lr_rgb.result.time
```

```
Time difference of 40.81696 secs
```

Hide

Hide

```
lr_rgb.test.result <- test(lr_rgb.result,test_rgb)
lr_rgb.test.accuracy <- 1 - mean(lr_rgb.test.result !=test_rgb[,1])
lr_rgb.test.accuracy
```

```
[1] 0.715
```

Step 3.6. XGBoost (XGBoost + RGB)

Hide

Hide

```
# library(xgboost)
# df <- read.csv("../output/rgb_feature.csv", header=FALSE)
# labels <- read.csv("../data/label_train.csv")
# df$label <- as.factor(labels$label)
# df$V1 <- NULL
#
# # Relabel factors for XGBoost specific num_classes requirement
# levels(df$label)[levels(df$label)=="1"] <- "0"
# levels(df$label)[levels(df$label)=="2"] <- "1"
# levels(df$label)[levels(df$label)=="3"] <- "2"
#
# #XGBoost Algorithm
# set.seed(031918)
# test.i <- sample(1:nrow(df), .3*nrow(df), replace=FALSE)
# test.data <- df[test.i,]
# train.data <- df[-test.i,]
#
# target.i <- which(colnames(df) == 'label')
# train.data <- df[-test.i, -target.i]
# train.target <- df[-test.i, target.i]
#
# t1=Sys.time()
# model3 <- xgb.cv(data = as.matrix(train.data), label = train.target, nfold=10,
#                  nrounds = 2, objective = "multi:softmax", num_class = 4)
# t2=Sys.time()
# t2-t1
# model3$time = t2-t1
# save(model3,file="../output/XGBoost_RGB.result.RData")
```

Hide

Hide

```
load("../output/XGBoost_RGB.result.RData")
XGBoost_RGB.test.accuracy <- 1-model3$evaluation_log$test_merror_mean[2]
XGBoost_RGB.test.accuracy
```

```
[1] 0.8047873
```

Hide

Hide

```
model3$time
```

```
Time difference of 3.404678 secs
```