Fried chicken or dogs or blueberry muffins

Fall2017-project3-fall2017-project3-grp4

10/30/2017

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 and Neural Networks) under different feature extraction(Sift, LBP, HoG). By comparing the accurancy rate as well as the processing time, we finally chose the best classification method.

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.

```r
packages.used=c("caret","gbm", "e1071", "DMwR", "nnet", "randomForest","OpenImageR","DT", "caTools", "EI
# 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)
}

library(caret)
library(gbm)
library(e1071)
library(DMwR)
library(randomForest)
library(nnet)
library(OpenImageR)
library(DT)
library(caTools)
library(EBImage)
library(mxnet)
library(pbapply)
library(ggthemes)

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

Step 1. Construct Baseline Model

Step 1.1. Load Feature

We devided the whole training set into 'df_train' & 'df_test'

'df_train' is the training data (80%)

'df_test' is the testing data (20%)

```
df = train_test_split("sift")
Sift_train = df$df_train
Sift_test = df$df_test
```

Step 1.2. Training Process of Baseline model

```
baseline.result = train_gbm(Sift_train)
```

```
load("../output/baseline.result.Rdata")
baseline.time = baseline.result[[4]]
baseline.time
```

```
## Time difference of 1.557644 mins
```

```
baseline.accuracy = 1 - baseline.result[[3]]
print(paste0("The accuracy rate of baseline model is ", baseline.accuracy))
```

```
## [1] "The accuracy rate of baseline model is 0.746666666666667"
```

Step 1.3. Predict result

We used GBM, Logistic Regression, SVM, Random Forest to construct the classification process separatly. However, the process really need a long time.

```
# Classifications                    Accuracy Rate
# Sift + GBM                               ~73%
# Sift + Logistic Regression               ~71%
# Sift + Random Forest                     ~73%
# Sift + SVM                               ~33%
```

Step 2. Construct Visual Features

The feature Sift, which has 5000 diamensions, is extremely over-weighted, espacially using classifier SVM. Thus we need to explore some other features. In the following process, we extracted LBP features and HoG features.

Step 2.1. Local Binary Patterns (LBP)

- Divide the examined window into cells (e.g. 16x16 pixels for each cell).

- For each pixel in a cell, compare the pixel to each of its 8 neighbors 2
- Where the center pixel is value is greater than the neighbors value, write 0. Otherwise, write 1
- This gives an 8-digit binary number.
- Compute the histogram over the cell. This histogram can be seen as a 256-dimensional feature vector.
- Optionally normalize the histogram to 59-dimensional feature vector.
- Concatenate histograms of all cells. This gives a feature vector for the entire window.

Step 2.1.1. Creat and Load Feature

We used matlab to extract LBP feature, and finally produce 555 diamensions.

```
df = train_test_split("lbp")
LBP_train = df$df_train
LBP_test = df$df_test
```

Step 2.1.2. Training Process under LBP

LBP + Classifiers

```
# LBP + Random Forest
rf.fit.LBP = train_rf(LBP_train)

# LBP + Logistic Regresstion
lr.fit.LBP = train_lr(LBP_train)

# LBP + SVM
svm.fit.LBP = train_svm(LBP_train)

# LBP + GBM
gbm.fit.LBP = train_gbm(LBP_train)
```

Step 2.1.3. Result table of LBP extraction

```
source("../lib/train_test_accuracy.R")
lbp.result = train_test_accuracy("lbp")
```

```
load("../output/lbp.result.Rdata")

lbp.result
```

```
##                 Model Train_accuracy Test_accuracy    Running_Time
## 1       Random Forest      1.0000000     0.7900000 117.037174 secs
## 2 Logistic Regression      0.8908333     0.8583333   3.813383 secs
## 3                 SVM      0.9762500     0.8333333  12.271676 secs
## 4                 GBM      1.0000000     0.8533333  64.338919 secs
```

Step 2.2. HoG

We used R to extract HoG feature, and finally produce 251 diamensions, and the parameter of HoG should be 5 and 10.

```
load("../output/hog.para.accuracy.Rdata")
hog.para.accuracy
```

```
##         orientation 6 orientation 8 orientation 10
## cells 5     0.7750000     0.7800000      0.8200000
## cells 6     0.7783333     0.7733333      0.8100000
## cells 7     0.7816667     0.7816667      0.8050000
## cells 8     0.7866667     0.8033333      0.8016667
```

Step 2.2.1. Load Feature

```
df = train_test_split("hog510")
HoG_train = df$df_train
HoG_test = df$df_test
```

Step 2.2.2. Training Process under HoG

HoG + Classifiers

```
# HoG + Random Forest
rf.fit.HoG = train_rf(HoG_train)

# HoG + Logistic Regresstion
lr.fit.HoG = train_lr(HoG_train)

# HoG + SVM
SVM.fit.HoG = train_rf(SVM.fit.HoG)

# HoG + GBM
gbm.fit.HoG = train_gbm(HoG_train)
```

Step 2.2.3. Result table of HoG extraction

```
source("../lib/train_test_accuracy.R")
hog.result = train_test_accuracy("hog510")
```

```
load("../output/hog.result.Rdata")

hog.result
```

```
##                  Model Train_accuracy Test_accuracy   Running_Time
## 1        Random Forest           1.00     0.7616667 51.657054 secs
## 2 Logistic Regression           0.87     0.8200000  1.275722 secs
## 3                 SVM           0.93     0.7700000  6.455067 secs
## 4                 GBM           1.00     0.8166667 29.836838 secs
```

Step 2.3. Neural Networks

Neural Network requires large amount of data, and we only have 3000 images in total. Hence we try to duplicate our images by flopping only training images.

In Neural Network feature extraction step, we first shrink the image to the size of 64*64 and convert colored images to gray images.

The structure of our neural network model includes one convolution layer, one activation layer, one pooling layer and 1 dropout layer, which forms our first convolution layers and the second convolution layer is the same as the first one. after two convolution layers, we have 2 full connection layers. The structure of each full connection layer is convolution, activation, and dropout layers.

Step 2.3.1 Neural Networks Feature Extraction

```r
source('../lib/cnn_feature.R')
img_dir="../data/test_set/images"
cnn_feature = extract_feature(img_dir,is_test = T)
```

Step 2.3.2 Train and Save Neural Networks Model Training

```r
train_cnn(cnn_feature)
```

Step 2.3.3 Neural Networks Model Prediction

```r
source('test.R')
test_cnn_prediction = test_cnn("../output/cnn_params/train_cnn", cnn_feature)
write.csv(test_cnn_prediction,"../output/cnn_test_prediction.csv")
```

Step 2.4 Majority Vote

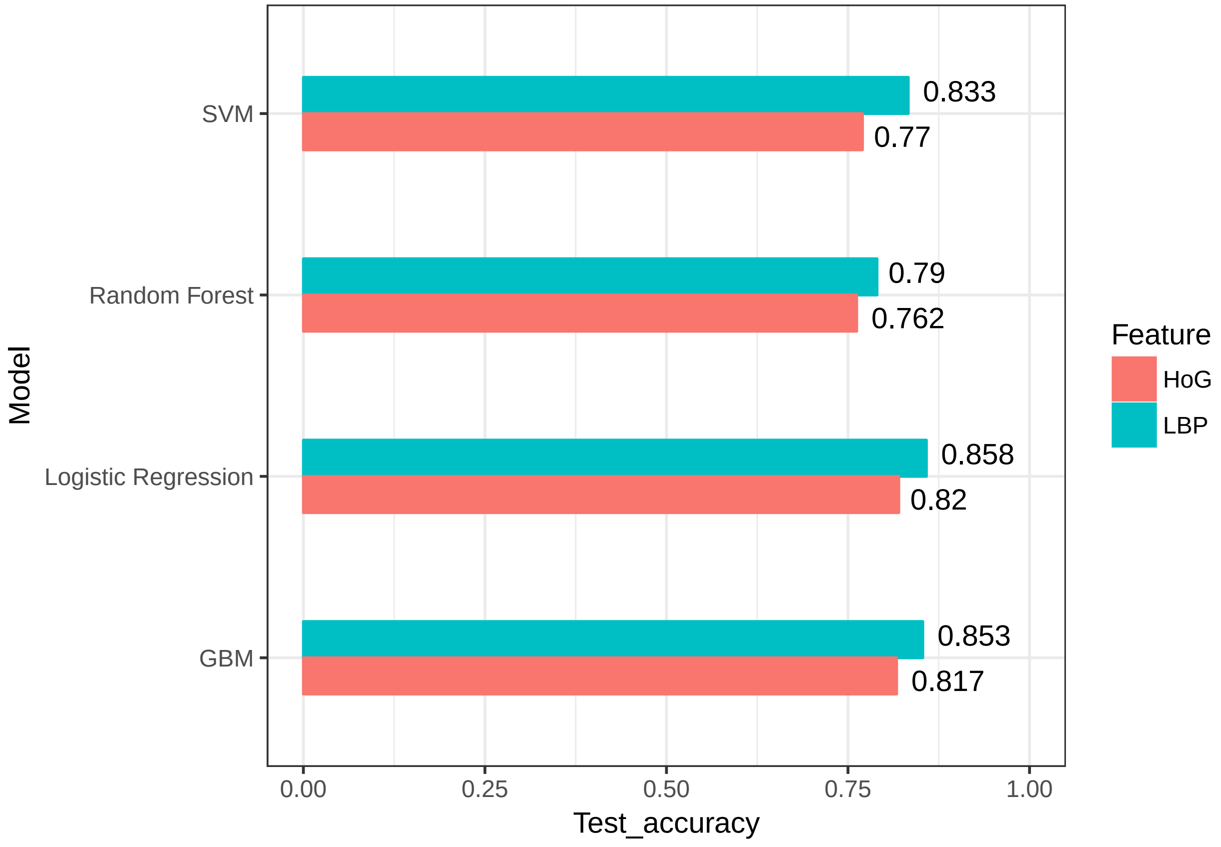
Mojority vote model is the combination of three models of logistic regression, GBM and SVM, which are three best models in our training process . Each model counts 1 vote to the final combo model. The majority label is prediction of our combo model.

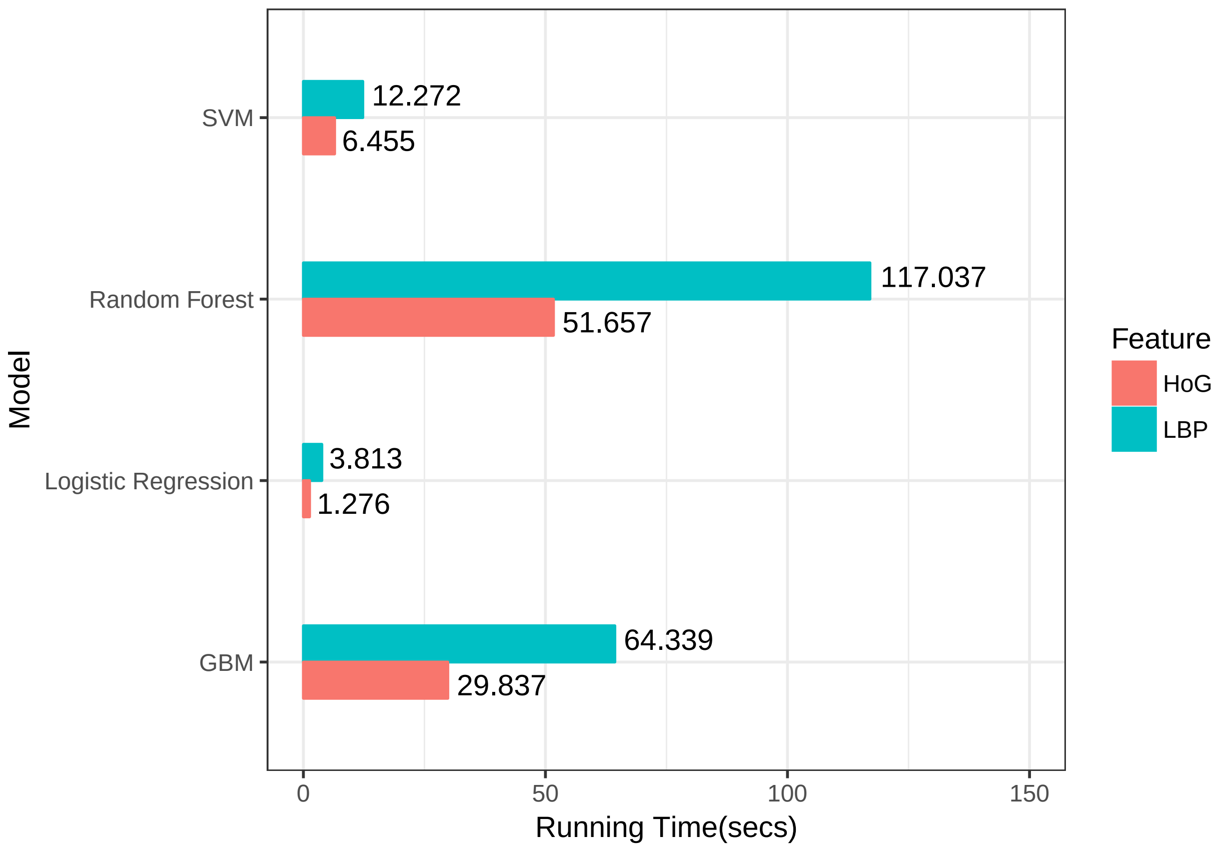
```r
load("../output/lr.fit.LBP.Rdata")
load("../output/svm.fit.LBP.Rdata")
load("../output/gbm.fit.LBP.Rdata")
# Here is the code we use to create majority vote matrix
df = train_test_split('lbp')
svm_pre = test(svm,df$df_test)
lr_pre = test(lr,df$df_test)
gbm_pre = test_gbm(gbm,df$df_test)
```

```
all_train=data.frame(svm_pre,lr_pre,gbm_pre)


for (i in 1:600){
  if (all_train[i,1]==all_train[i,2]){
    all_train[i,4]=all_train[i,1]
  } else if (all_train[i,1]==all_train[i,3]){
    all_train[i,4]=all_train[i,1]
  } else if (all_train[i,2]==all_train[i,3]){
    all_train[i,4]=all_train[i,3]
  } else{
    all_train[i,4]=all_train[i,2]
  }
}


print(paste0("The test accuracy of the majority vote model is ", mean(all_train$V4 == df$df_test$Label)
```

```
## [1] "The test accuracy of the majority vote model is 0.883333333333333"
```

Step 3. Model Selection





```
##                   Model Feature     Train_accuracy     Test_accuracy
## 1         Random Forest     LBP                  1              0.79
## 2   Logistic Regression     LBP 0.890833333333333             0.858
## 3                   SVM     LBP            0.97625             0.833
## 4                   GBM     LBP                  1             0.853
## 5         Random Forest     HoG                  1             0.762
## 6   Logistic Regression     HoG               0.87              0.82
## 7                   SVM     HoG               0.93              0.77
## 8                   GBM     HoG                  1             0.817
## 9                   GBM    Sift 0.792083333333333 0.746666666666667
## 10                  CNN     CNN              0.963             0.842
##              Running_Time
## 1            117.037 secs
## 2              3.813 secs
## 3             12.272 secs
## 4             64.339 secs
## 5             51.657 secs
## 6              1.276 secs
## 7              6.455 secs
## 8             29.837 secs
## 9   1.55764391819636 secs
## 10               1924 secs
```

Step 4. Final Test

Depending on our analysis, finally we choose LBP with classifier Logistic Regresstion as our main method. We also want to test the result using others classification method, here are the results.

Step. 4.1. Feature Extraction

```
df_test = read.csv("../output/lbp_test.csv", header = F)
```

Step. 4.2. Test Process under SIFT and LBP

```
sift_test = read.csv('../data/sift_test.csv', header = T)
load("../output/lr.fit.LBP.Rdata")
load("../output/rf.fit.LBP.Rdata")
load("../output/svm.fit.LBP.Rdata")
load("../output/gbm.fit.LBP.Rdata")
load("../output/baseline.result.Rdata")

baseline.model = list(fit = baseline.result[[1]])

baseline.test.pre = predict(baseline.model, sift_test)

test_lr = test(lr, df_test)
test_rf = test(rf, df_test)
test_svm = test(svm, df_test)
test_gbm_1 = test_gbm(gbm, df_test)
```

Step 4.3. Final Table

```
test_cnn = read.csv('../output/cnn_test_prediction.csv')
#Get the label of the baseline model's prediction
label_test_baseline = data.frame(X = c(1:3000), 'baseline' = test_gbm_1)
write.csv(label_test_baseline, 'label_test_baseline.csv',row.names = F)

#Get the label of other classification model's prediction
all_train=data.frame(test_svm,test_lr,test_gbm_1)

for (i in 1:3000){
  if (all_train[i,1]==all_train[i,2]){
    all_train[i,4]=all_train[i,1]
  } else if (all_train[i,1]==all_train[i,3]){
    all_train[i,4]=all_train[i,1]
  } else if (all_train[i,2]==all_train[i,3]){
    all_train[i,4]=all_train[i,3]
  } else{
    all_train[i,4]=all_train[i,2]
  }
```

```r
}


label_test_all = data.frame(X = c(1:3000), 'GBM' = test_gbm_1, 'SVM' = test_svm, 'Random Forest' = test_
write.csv(label_test_all, 'label_test_all.csv',row.names = F)
```