

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 accuracy 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 argument, so we ignore creating cross validation R file.

```
packages.used=c("caret","gbm", "e1071", "DMwR", "nnet", "randomForest","OpenImageR","DT", "caTools", "EBImage")
# 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 divided 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 1.579168 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 0.768333333333333"
```

Step 1.3. Predict result

We used GBM, Logistic Regression, SVM, Random Forest to construct the classification process separately. 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 dimensions, is extremely over-weighted, especially 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
- 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 dimensions, 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 Regression
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

```
source('../lib/cnn_feature.R')
cnn_feature = extract_feature(img_dir)
```

Step 2.3.2 Neural Networks Model Training

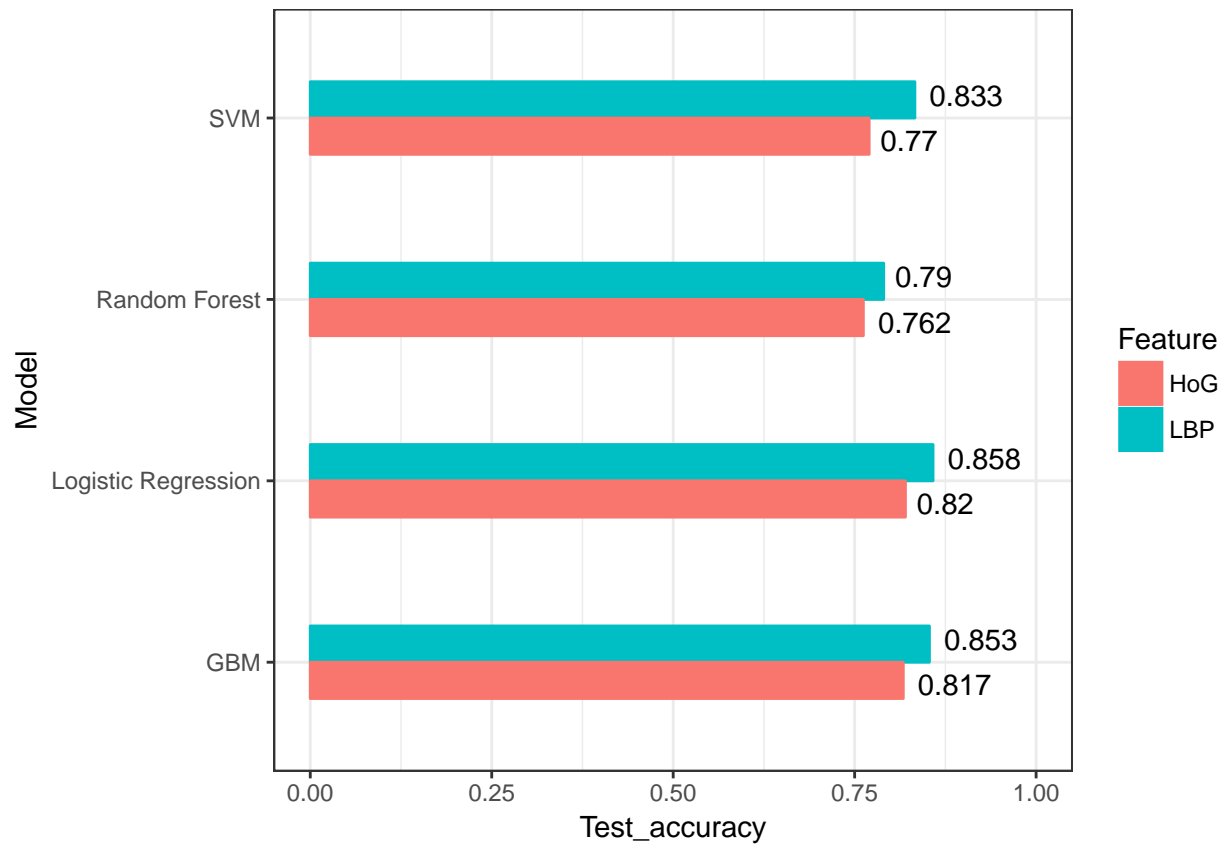
```
cnn = train_cnn(cnn_feature)
save(cnn, "../output/cnn.fit.Rdata")
```

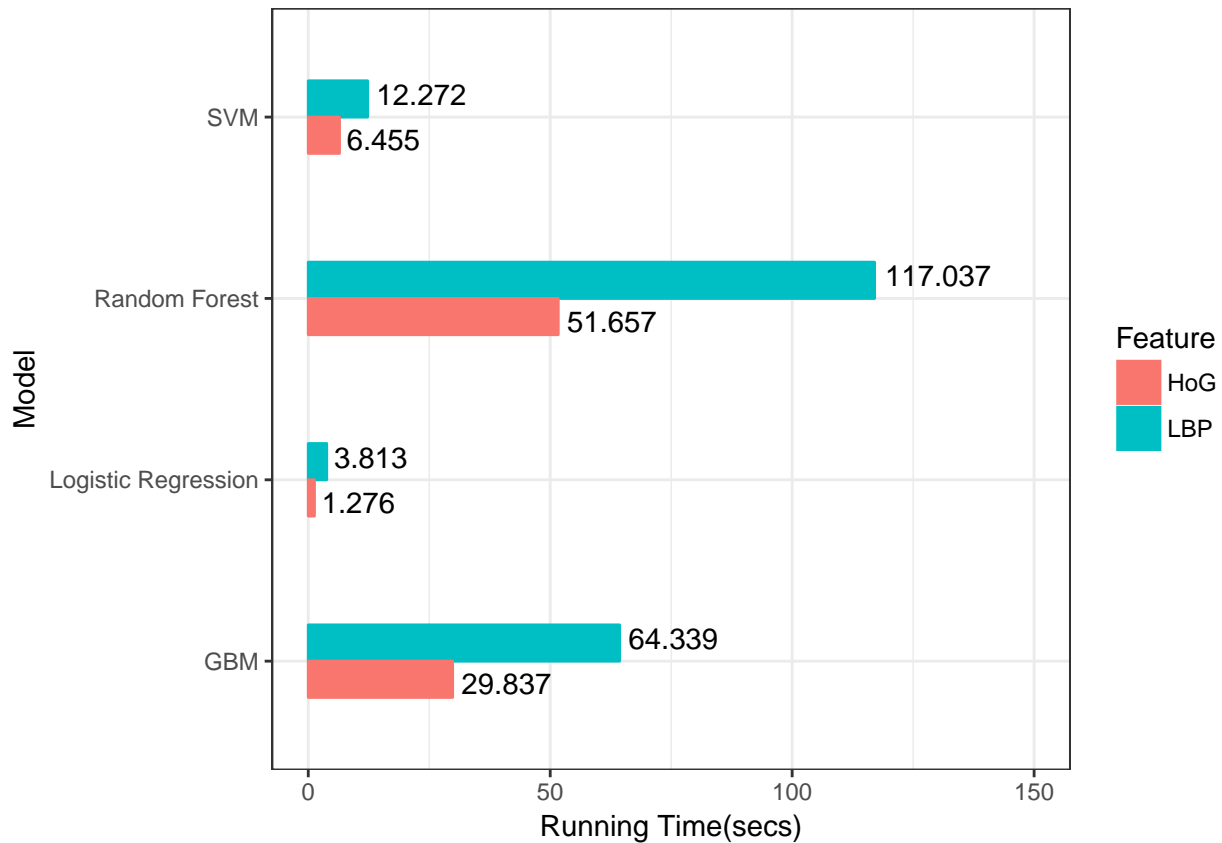
Step 2.3.2 Neural Networks Model Training

```
load(file = '../output/cnn.test.Rdata')
df_cnn

##   training_error test_error running_time
## 1           0.963         0.842         32min
```

Step 3. Model Selection





```
## Warning in `[<-factor`(`*tmp*`, ri, value = "CNN"): invalid factor level,
## NA generated
```

##	Model	Feature	Train_accuracy	Test_accuracy
## 1	Random Forest	LBP	1	0.79
## 2	Logistic Regression	LBP	0.8908333333333333	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	0.8029166666666667	0.7683333333333333
## 10	<NA>	NULL	1	1

##	Running_Time	Parameters
## 1	117.037 secs	Ntree=
## 2	3.813 secs	Multinomial
## 3	12.272 secs	Cost= , Margin=
## 4	64.339 secs	GBM
## 5	51.657 secs	Ntree=
## 6	1.276 secs	Multinomial
## 7	6.455 secs	Cost= , Margin=
## 8	29.837 secs	GBM
## 9	1.57916771570841 secs	gbm
## 10	1 secs	Layer=

Step 4. Final Test

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

Step. 4.1. Feature Extraction

```
# load("../output/lbp_test.csv")  
# output name df_test
```

Step. 4.2. Test Process under LBP

```
load("../output/lr.fit.LBP.Rdata")  
load("../output/rf.fit.LBP.Rdata")  
load("../output/svm.fit.LBP.Rdata")  
load("../output/gbm.fit.LBP.Rdata")  
  
test_lr_accuracy = mean(test(lr, df_test))  
test_rf_accuracy = mean(test(rf, df_test))  
test_svm_accuracy = mean(test(svm, df_test))  
test_gbm_accuracy = mean(test(gbm, df_test))
```

Step 4.3. Final Analysis Table