Project 3: Image Classification

1. Baseline Model--Boosted Decision Stump

Import necessary libraries

Set os.environ["R_USER"] to user name in windows

In [1]:
```
import pandas as pd
import numpy as np
import pyreadr
import import_ipynb
import time
```

In [2]:
```
import os
os.environ["R_USER"] = "Jiyoung Sim" # user name
```

In [3]:
```
from rpy2 import robjects
from rpy2.robjects import pandas2ri
from rpy2.robjects import numpy2ri
```

Step 0: Provide directories for training/testing images. Images and fiducial points will be in different subfolders.

In [4]:
```
train_dir = './data/train_set/' # This will be modified for different data set
s.
train_image_dir = train_dir + 'images/'
train_pt_dir = train_dir + 'points/'
train_label_path = train_dir + 'label.csv'
```

In [5]:
```
test_dir = './data/test_set/' # This will be modified for different data sets.
test_image_dir = test_dir + 'images/'
test_pt_dir = test_dir + 'points/'
test_label_path = test_dir + 'label.csv'
```

Step 1: set up controls for evaluation experiments.

Set baseline_tt_split = True if you want to do train_test_split for data in the same directory. Set it to False if you have test data in a different directory

In [6]:
```
baseline_feature_train = True # process features for training set
baseline_train = True # train model
baseline_cv= True # hyperparameter tuning by GridSearchCV when training
baseline_feature_test = True # process features for test set
baseline_test = True # run evaluation on an independent test set
baseline_tt_split = True
```

Step 2: import data and train-test split if wanted

In [7]:
```
info = pd.read_csv(train_label_path)
if(baseline_tt_split):
    from sklearn.model_selection import train_test_split
    train_idx_py, test_idx_py = train_test_split(range(len(info)), test_size=
0.2, random_state = 0)
    train_idx_r = [i+1 for i in train_idx_py]
    test_idx_r = [i+1 for i in test_idx_py]
    info_test = info
else:
    info_test = pd.read_csv(test_label_path)
    train_idx_py = list(range(len(info)))
    train_idx_r = [i+1 for i in train_idx_py]
    test_idx_py = list(range(len(info_test)))
    test_idx_r = [i+1 for i in test_idx_py]
```

Step 3: construct features and responses

In [8]:
```
# Import feature.R
feature = robjects.r(
    '''
    source('./lib/feature.R')
    '''
)[0]
```

In [12]:
```
# function to read fiducial points
def readMat(index):
    import scipy.io
    numpy2ri.activate()
    try:
        mat = np.round(scipy.io.loadmat(train_pt_dir + '{:04n}.mat'.format(ind
ex))['faceCoordinatesUnwarped'])
    except KeyError:
        mat = np.round(scipy.io.loadmat(train_pt_dir + '{:04n}.mat'.format(ind
ex))['faceCoordinates2'])
    nr,nc = mat.shape
    mat_r = robjects.r.matrix(mat, nrow=nr, ncol=nc)
    robjects.r.assign("mat", mat_r)
    return mat_r

#load fiducial points
start = time.time()
n_files = len(os.listdir(train_pt_dir))
fiducial_pt_list = [readMat(index) for index in range(1, n_files+1)]
end = time.time()
tm_fid_pt_train = end-start

if(baseline_tt_split):
    fiducial_pt_test = fiducial_pt_list
    tm_fid_pt_test = tm_fid_pt_train
else:
    start = time.time()
    n_files = len(os.listdir(test_pt_dir))
    fiducial_pt_test = [readMat(index) for index in range(1, n_files+1)]
    end = time.time()
    tm_fid_pt_test = end-start
```

In [13]:
```
# convert pandas dataframe to R dataframe
from rpy2.robjects import pandas2ri
pandas2ri.activate()
info_rdf = pandas2ri.py2ri(info)
info_test_rdf = pandas2ri.py2ri(info_test)
```

In [14]:
```
# extract features from fiducial points
as_factor = robjects.r('''as.factor''')
if(baseline_feature_train):
    start = time.time()
    dat_train_r = feature(fiducial_pt_list, train_idx_r, info_rdf)
    end = time.time()
    dat_train_py = pandas2ri.ri2py_dataframe(dat_train_r)
    dat_train_r[-1] = as_factor(dat_train_r[-1])
    tm_feature_train_baseline = end - start + tm_fid_pt_train
#     dat_train_py.to_csv('dat_train_py.csv', index=False)

if(baseline_feature_test):
    start = time.time()
    dat_test_r = feature(fiducial_pt_test, test_idx_r, info_test_rdf)
    end = time.time()
    dat_test_py = pandas2ri.ri2py_dataframe(dat_test_r)
    dat_test_r[-1] = as_factor(dat_test_r[-1])
    tm_feature_test_baseline = end - start + tm_fid_pt_test
#     dat_test_py.to_csv('dat_test_py.csv', index=False)
```

Step 4: Train a classification model with training features and responses

In [21]:
```
# train baseline model
baseline_dir = 'baseline_train_main2.sav' #'baseline_train_alldata.sav'#'basel
ine_train_main.sav'
if (baseline_train==True):
    import train_baseline
    tm_train_baseline, baseline = train_baseline.gbm_fn(dat_train_py.iloc[:,:-
1], dat_train_py.iloc[:,-1], baseline_cv)

    from sklearn.externals import joblib
    joblib.dump(baseline, baseline_dir) # save the model to disk

# test
if (baseline_test==True):
    import test_baseline
    start= time.time()
    baseline_acc = test_baseline.test_clf(dat_test_py.iloc[:,:-1], dat_test_py
.iloc[:,-1], baseline_dir)
    end = time.time()
    tm_test_baseline = end-start
```

Step 5: Summarize Running Time and Accuracy

In [15]:
```
print('training feature extraction took: {}'.format(tm_feature_train_baseline
))
print('testing feature extraction took: {}'.format(tm_feature_test_baseline))
print('model training took: {}'.format(tm_train_baseline))
print('model testing took: {}'.format(tm_test_baseline))
print('model accuracy: {}'.format(baseline_acc))
```

```
training feature extraction took: 2.778940200805664
testing feature extraction took: 2.13370418548584
model training took: 972.6331098079681
model testing took: 0.9105648994445801
model accuracy: 0.45
```

2. Improved Model--Voting Classifier (Combines Light GBM (dart), Logistic Regression, Linear SVM, and Random Forest)

Step 1: set up controls for evaluation experiments.

Set baseline_tt_split = True if you want to do train_test_split for data in the same directory. Set it to False if you have test data in a different directory

In [16]:
```
voting_feature_train = True # process features for training set
voting_train = True # train model
voting_cv= True # hyperparameter tuning by GridSearchCV when training
voting_feature_test = True # process features for test set
voting_test = True # run evaluation on an independent test set
voting_tt_split = True
```

Step 2: import data and train-test split if wanted--identical to Step 2 in previous part

Step 3: construct features and responses

In [17]:
```
# import myfeature2.R
myfeature2 = robjects.r(
    '''
    source('./lib/myfeature2.R')
    '''
)[0]
```

In [19]:
```
# extract features from fiducial points
start = time.time()
myfeature_train_r = myfeature2(info_rdf, fiducial_pt_list)
myfeature_train_py = pandas2ri.ri2py_dataframe(myfeature_train_r)
myfeature_train_r[-1] = as_factor(myfeature_train_r[-1])
end = time.time()
tm_feature_train_voting = end - start + tm_fid_pt_train

if(voting_tt_split):
    train_df = myfeature_train_py.iloc[train_idx_py].reset_index(drop=True)
    test_df = myfeature_train_py.iloc[test_idx_py].reset_index(drop=True)
    tm_feature_test_voting = tm_feature_train_voting
else:
    train_df = myfeature_train_py

    start = time.time()
    myfeature_test_r = myfeature2(info_test_rdf, fiducial_pt_test)
    myfeature_test_py = pandas2ri.ri2py_dataframe(myfeature_test_r)
    myfeature_test_r[-1] = as_factor(myfeature_test_r[-1])
    end = time.time()

    tm_feature_test_voting = end - start + tm_fid_pt_test
    test_df = myfeature_test_py
```

Step 4: Train a classification model with training features and responses

In [22]:
```
# train improved model
voting_dir = 'voting_train_main2.sav' #'voting_train_alldata.sav'#'voting_trai
n_main.sav'
if (voting_train==True):
    import train_voting
    tm_train_voting, voting = train_voting.train_fn(train_df.iloc[:,:-1], trai
n_df.iloc[:,-1], voting_cv)

    from sklearn.externals import joblib
    joblib.dump(voting, voting_dir) # save the model to disk

# test the model
if (voting_test==True):
    import test_voting
    start= time.time()
    voting_acc = test_voting.test_fn(test_df.iloc[:,:-1], test_df.iloc[:,-1],
voting_dir)
    end = time.time()
    tm_test_voting = end-start
```

Step 5: Summarize Running Time and Accuracy

In [20]:
```
print('training feature extraction took: {}'.format(tm_feature_train_voting))
print('testing feature extraction took: {}'.format(tm_feature_test_voting))
print('model training took: {}'.format(tm_train_voting))
print('model testing took: {}'.format(tm_test_voting))
print('model accuracy: {}'.format(voting_acc))
```

```
training feature extraction took: 2.9130449295043945
testing feature extraction took: 2.9130449295043945
model training took: 168.33163595199585
model testing took: 2.0300350189208984
model accuracy: 0.474
```