MultiLayerNN.FaceRecognition

June 10, 2021

[1]:

```
                                 # CAP 5615 2021 Summer, X. Zhu, June 10 2021
                                # Multi Layer Neural Network Face Recognition
# some codes were adopated from https://www.kaggle.com/serkanpeldek/
 ↪face-recognition-on-olivetti-dataset
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.utils import shuffle
```

[2]:

```
data=np.load("olivetti_faces.npy")
target=np.load("olivetti_faces_target.npy")
```

[3]:

```
# data (400 images, each 64x64)
print(data.shape)
# labels
print(target.shape)
print(target)
```

```
(400, 64, 64)
(400,)
[ 0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  2  2  2  2
  2  2  2  2  2  2  3  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4
  4  4  5  5  5  5  5  5  5  5  5  5  6  6  6  6  6  6  6  6  6  6  7  7
  7  7  7  7  7  7  7  7  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9
  9  9  9  9 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11 11 11 11
 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13 13 13 13 14 14 14 14
 14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 16 16 16 16 16 16 16 16
 16 16 17 17 17 17 17 17 17 17 17 17 18 18 18 18 18 18 18 18 18 18 19 19
 19 19 19 19 19 19 19 19 20 20 20 20 20 20 20 20 20 20 21 21 21 21 21 21
 21 21 21 21 22 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23 23 23 23 23
 24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 26 26 26 26
 26 26 26 26 26 26 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28
 28 28 29 29 29 29 29 29 29 29 29 29 30 30 30 30 30 30 30 30 30 30 31 31
 31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32 32 32 33 33 33 33 33 33
 33 33 33 33 34 34 34 34 34 34 34 34 34 34 35 35 35 35 35 35 35 35 35 35
 36 36 36 36 36 36 36 36 36 36 37 37 37 37 37 37 37 37 37 37 38 38 38 38
 38 38 38 38 38 38 39 39 39 39 39 39 39 39 39 39]
```

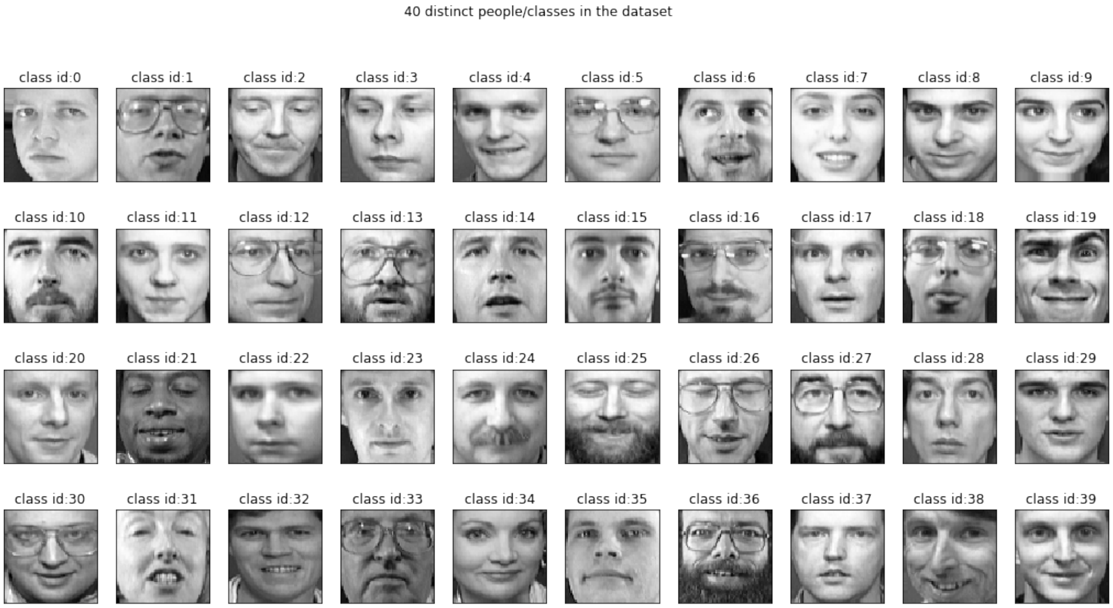
[4]:

```
def show_a_random_face_per_class(images, unique_ids):
    #Creating 4X10 subplots in  18x9 figure size
    fig, axarr=plt.subplots(nrows=4, ncols=10, figsize=(18, 9))
    #For easy iteration flattened 4X10 subplots matrix to 40 array
    axarr=axarr.flatten()

    #iterating over user ids
    rand=np.random.randint(10)
    for unique_id in unique_ids:
        image_index=unique_id*10+rand
        axarr[unique_id].imshow(images[image_index], cmap='gray')
        axarr[unique_id].set_xticks([])
        axarr[unique_id].set_yticks([])
        axarr[unique_id].set_title("class id:{}".format(unique_id))
    plt.suptitle("40 distinct people/classes in the dataset")
```

[5]:

```
show_a_random_face_per_class(data, np.unique(target))
```



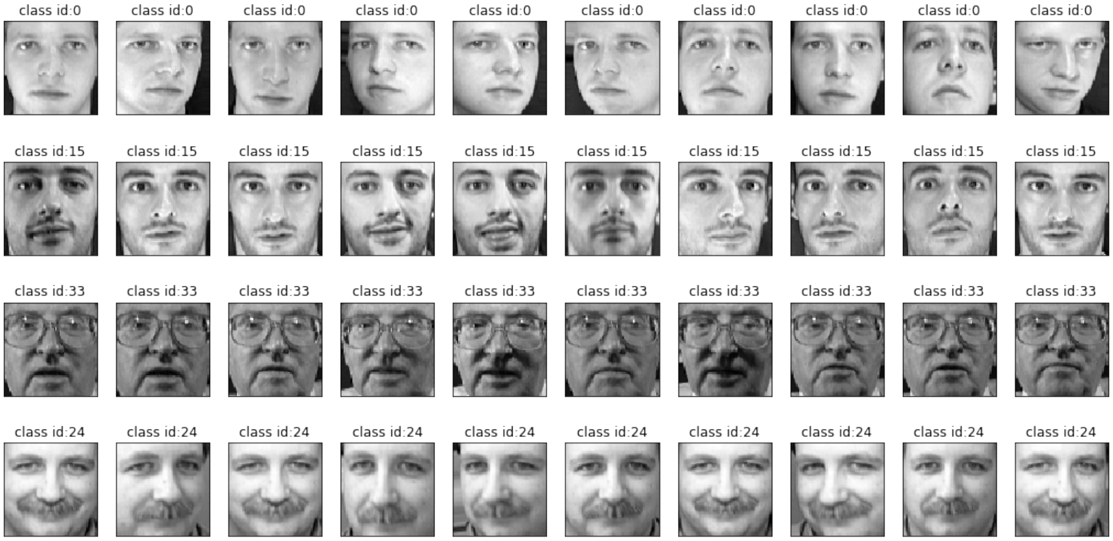
[6]:

```
# show all images of selected class
def show_all_faces_of_selected_subjects(images, subject_ids):
    cols=10# each subject has 10 distinct face images
    rows=(len(subject_ids)*10)/cols #
    rows=int(rows)

    fig, axarr=plt.subplots(nrows=rows, ncols=cols, figsize=(18,9))
    #axarr=axarr.flatten()
    for i, subject_id in enumerate(subject_ids):
        for j in range(cols):
            image_index=subject_id*10 + j
            axarr[i,j].imshow(images[image_index], cmap="gray")
            axarr[i,j].set_xticks([])
            axarr[i,j].set_yticks([])
            axarr[i,j].set_title("class id:{}".format(subject_id))
```

[15]:

```
show_all_faces_of_selected_subjects(images=data, subject_ids=[0,15, 33, 24])
```



[8]:

```
# now we flatten each 64x64 image as a single vector 64x64=4096 (for training␣
 ↪NN)
X=data.reshape((data.shape[0],data.shape[1]*data.shape[2]))
X.shape
```

[8]:

```
(400, 4096)
```

[9]:

```
# now we slpilt training and test data.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X, target, test_size=0.2,␣
 ↪stratify=target, random_state=1)
print(X_train.shape)
print(X_test.shape)
```

```
(320, 4096)
(80, 4096)
```

[24]:

```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=500,␣
 ↪random_state=42,activation='logistic',max_iter=1000)
clf.fit(X_train, y_train)
```

[24]:

```
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=500, learning_rate='constant',
              learning_rate_init=0.001, max_iter=1000, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=42, shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

[25]:

```
y_pred=clf.predict(X_test)
print(y_test)
print(y_pred)
```

```
[18 23 39  6 38 28 19 22  4 24 37  7 34 32  8 31 25 34 27 29 24  5  2 26
 11 26 11 33  5 35 35 13 39  4  2 10 30 36 15 17  9 29  9  1 14  7 14  1
  6  3 15 30  0  3 27 16 20 32 12 28 21 25 19 38 22 16 33 31 17 23 13 10
 37 20  8 12 21 36 18  0]
[18 23 39  6 28 28 19 24  5 24 37  7 34 32  8 31 25 34 31 29 24  5  2 26
 11 26 11 33  5 35 35 13 39 20  2 10 30 36 15 17  9 29  9 31 14  7 14  1
  6 14 15 30  0  3 27 16 20 32  0 28 21 24 19 29 22 16 33 30 17 23 13 10
 37 20 22  2 21 36 18  0]
```

[26]:

```
from sklearn.metrics import confusion_matrix
cf=confusion_matrix(y_test, y_pred)
cf
```

[26]:

```
array([[2, 0, 0, …, 0, 0, 0],
       [0, 1, 0, …, 0, 0, 0],
       [0, 0, 2, …, 0, 0, 0],
       …,
       [0, 0, 0, …, 2, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 2]], dtype=int64)
```

[27]:

```
# use scikit-learn to calculate accuracy.
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

[27]:

```
0.8375
```