Deep Learning for Computer Vision

Homework #0

經濟四 李品樺 **B07303024**

Collaborators: None

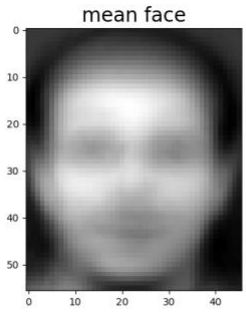
Problem 1

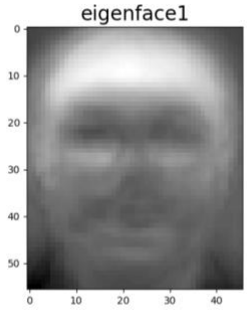
1. Perform PCA on training set. Plot mean face and first four eigenfaces.

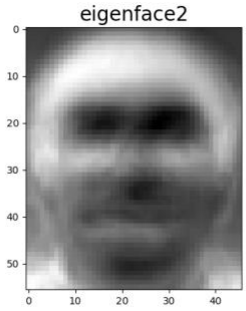
- Divide the dataset into two partitions, training set and testing set. Select first nine pictures of each class as the training set, the last picture of each class as the testing set. Then, we create labels using numpy array for training set and testing set for further usage.
- Read the image using **cv2**. Note that we read the images into **grayscale by passing a flag=0** in the cv2.imread method, and change them into numpy array
- Calculate the mean face value using **mean** method of numpy array, reshape it into original size so as to write the image using **pyplot**.
- Subtract the original training image by the mean face value
- Using PCA function from **sklearn** and fit the result from previous step
- The eigenfaces can be obtained by selecting **components_** which is an attribute of the fitting result
- Since the values obtained are negative, we need to normalize them to the scale of (0,255) as below:

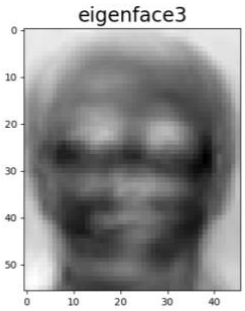
```
a = 255/(np.max(eigenfaces, axis=1) - np.min(eigenfaces, axis=1))
a = np.expand_dims(a, -1)
b = -a*np.expand_dims(np.min(eigenfaces,axis=1), -1)
eigenfaces = (eigenfaces*a + b).reshape(360, 56, 46)
```

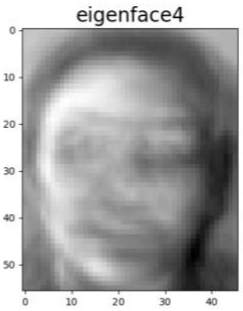
- Finally, reshape them back to original size, and save the images









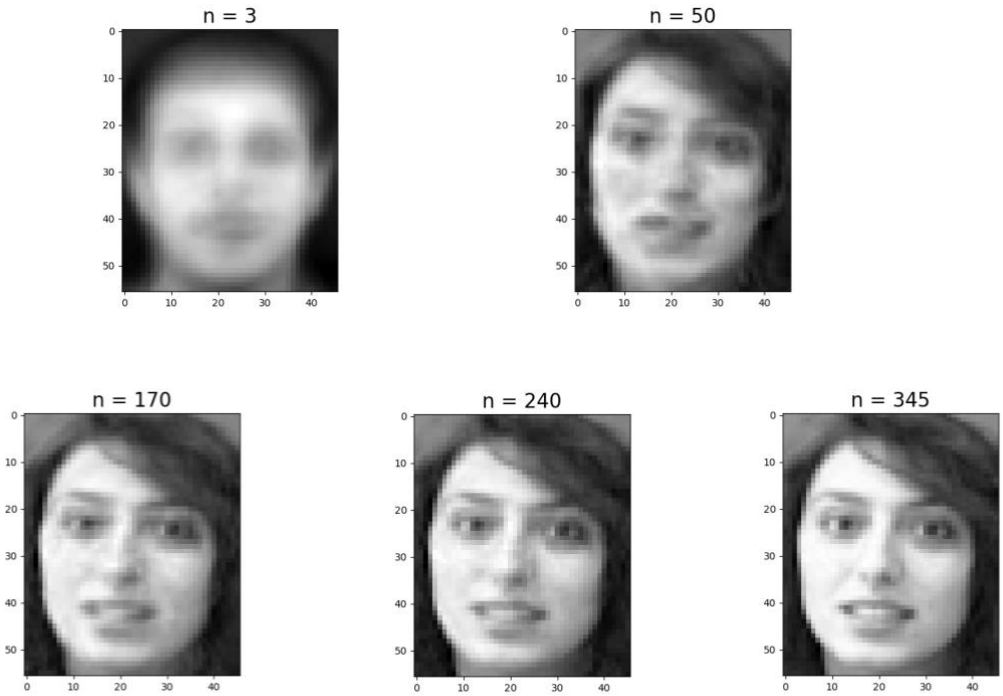


- Reference:
  - Scikit learn - PCA
  - GeekforGeeks - Face Recognition using PCA Implementation

2. Project person8 image1 onto the PCA eigenspace.

Reconstruct this image using the first n= 3,50,170,240,345 eigenfaces.

- Read the given image and subtract it by the mean face
- Fit the subtracted result with the PCA model
- Use the transform method of PCA to get the projected value
- Using first n eigenfaces, reconstruct the image by doing matrix multiplication of the projected result and the eigenfaces in the previous problem
- Reconstructed images are obtained by adding mean face to the multiplication result
- The reconstructed images are shown below:



- We can find out that the image is clearer and closer to the original image when reconstructed using more eigenfaces
- Reference:
    - [Face Recognition with Eigenfaces – Python Machine Learning]

3. Compute the mean squared error (MSE) between the reconstructed image and the original image.

- Subtract the original image and the reconstructed result pixel-wise, square the value and compute the mean. The result is shown in the table below.
- As we observe from the table, using more eigenfaces (i.e. larger n), the reconstructed mean squared error is less.

n	MSE
3	1566.347198469649
50	134.03183261341115
170	39.84704055061051
240	21.477817770862064
345	3.042216175564436

4. Apply the k-nearest neighbors algorithm to classify the testing set images. Determine the best k and n values by 3-fold cross-validation.

- First, perform PCA on the training set
- Set k as the parameter in **KNeighborsClassifier** model from sklearn, project the training set onto the n dimensions, then use the classifier to vote for the predicted result
- We use **cross_val_score** function from sklearn to implement 3-fold cross validation. Set the parameter **cv as 3** and **scoring as "accuracy"**
- Choose **k =1, n=50** for hyperparmeters since the accuracy is the highest of all
- The 3-fold validation result is shown as the screenshot below:

```
k=1, n=3  , Acc:0.650000
k=1, n=50 , Acc:0.961111
k=1, n=170, Acc:0.955556
k=3, n=3  , Acc:0.611111
k=3, n=50 , Acc:0.900000
k=3, n=170, Acc:0.888889
k=5, n=3  , Acc:0.561111
k=5, n=50 , Acc:0.847222
k=5, n=170, Acc:0.822222
```

- Reference:
  - Scikit learn - CrossValidation
  - Scikit learn - Nearest Neighbors
  - DataCamp - KNN Classification using Scikit-learn

5. Report the recognition rate of the testing set using the hyperparameter choice in 4.

- Subtract testing set by mean face value, perform PCA transform to reduce dimension
- Set k as 1 for **KNeighborsClassifier**
- Fit the classifier with projected training set(dimension equals to 50) and the training label
- Compute the recognition rate using projected testing set(dimension equals to 50) and testing label with the method **KNeighborsClassifier.score**
- **Recognition rate: 92.5%**