q1

April 3, 2020

1 Question 1, PCA ANALYSIS

[40]:
```
url="/home/abhishek/dev/Semester_2/SMAI/Assignments/Assignment_3/dataset"
```

[41]:
```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from skimage import transform,io
import warnings
warnings.filterwarnings('ignore')
from mpl_toolkits.mplot3d import Axes3D
```

[42]:
```
class PCA:

    def read_image(self,url):
        print("Reading Image from directory...")
        dir_list = os.listdir(url)
        faces = []
        image_labels = []
        for image in dir_list:
            labels = image.split("_")
            image_labels.append(labels[0])

        for image in dir_list:
            img = io.imread(url+"/"+image)
            img = img.astype(np.uint8)
            # converting to grayscale
            rgb_weights = [0.2989, 0.5870, 0.1141]
            grayscale_image = np.dot(img[...,:3], rgb_weights)
            #normalizing image
            small_grey = transform.resize(grayscale_image, (64,64),␣
 ↪mode='symmetric', preserve_range=True)
            reshape_img = small_grey.reshape(1, 4096)
            faces.append(reshape_img[0])
        X_train = np.asarray(faces)
```

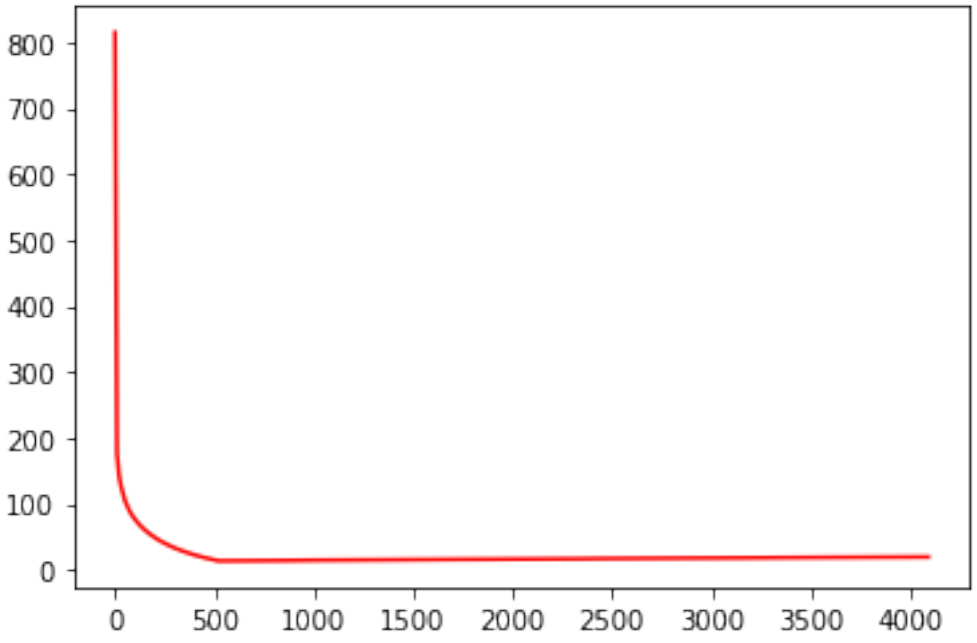
```
        # print(self.X_train[0])
        print(X_train.shape)
        print("Reading Image from directory Done...")
        return image_labels,X_train

    def apply_pca(self,X):
        print("Applying PCA.......")
        eig_val, eig_mat = np.linalg.eig(np.cov(X))
        idx = eig_val.argsort()[::-1]
        eig_val = eig_val[idx]
        eig_mat = eig_mat[:,idx]
        print("Eigen Matrix calculation done..")
        # taking principal components
        M = []
        print(eig_mat.shape)
        full_pc = eig_mat.shape[0]
        for numpc in range(0,full_pc,10):
            eigen_coeff = eig_mat[:,range(numpc)]
            score_mat = np.dot(eigen_coeff.T,X)
            final_score = np.dot(eigen_coeff,score_mat).T
            # print(final_score.shape)
            val = np.linalg.norm(X.T-final_score,'fro')
            M.append(val)
            # print(val)

        mse = np.asarray(M)
        print("Applying PCA done..............")
        return mse,eig_mat,eig_val

    def plot_mse_vs_principal_component(self,mse,eig_mat):
        full_pc = eig_mat.shape[1]
        plt.figure()
        plt.plot(range(0,full_pc,10),mse,'r')
        plt.show()

    def no_of_component_such_that_mse_less_than_20(self,eig_mat,X):
#         N = 20 #from observation no of principal compenent = 50
        eigen_coeff = eig_mat[:,range(24)]
        score_mat = np.dot(eigen_coeff.T,X)
        final = np.dot(eigen_coeff,score_mat).T
        final = final*255
        final_score = final.astype(int)
        fig,axes = plt.subplots(4,5,figsize=(9,9),subplot_kw={'xticks':[],␣
↪'yticks':[]},gridspec_kw=dict(hspace=0.01, wspace=0.01))
        for i, ax in enumerate(axes.flat):
            ax.imshow(final_score[i].reshape(64,64),cmap='gray')
```

```
        plt.show()
        return final_score


    def scatter_plot_pca(self,X_new,X):
        #1-D plot
        val=0
        plt.plot(X[:,0],np.zeros_like(X[:,0])+val,'o')
        plt.plot(X_new[:,0],np.zeros_like(X_new[:,0])+val,"x")
        plt.show()

        # 2-D plot
        plt.scatter(X[:,0],X[:,1], alpha=0.2)
        plt.scatter(X_new[:,0],X_new[:,1], alpha=0.8)#=self.image_labels)
        plt.show()

        #3-D plot
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(X[:,0], X[:,1], X[:,2], marker='o')
        ax.scatter(X_new[:,0], X_new[:,1], X_new[:,2], marker='^')
        ax.set_xlabel('X Label')
        ax.set_ylabel('Y Label')
        ax.set_zlabel('Z Label')
        plt.show()
```

[43]:
```
pca = PCA()
image_label,X_train = pca.read_image(url)
#normalization
X = X_train
X_train = X_train/255
#applying PCA
mse,eig_mat,eig_val = pca.apply_pca(X_train.T)
```

```
Reading Image from directory…
(520, 4096)
Reading Image from directory Done…
Applying PCA…
Eigen Matrix calculation done..
(4096, 4096)
Applying PCA done…
```

[44]:
```
pca.plot_mse_vs_principal_component(mse,eig_mat)
```



[45]:
```
final_score = pca.no_of_component_such_that_mse_less_than_20(eig_mat,X_train.T)
```

[46]:
```
## Check N has Mean Squared error has less than 20%
```

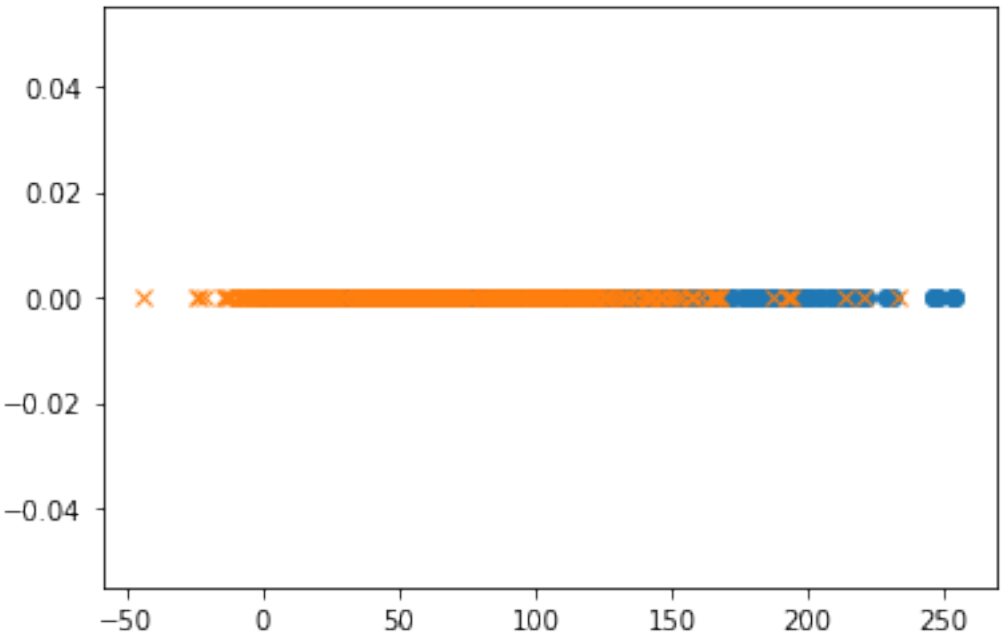
[47]:
```
eigen_val_n = eig_val[0:24]
x = np.sum(eigen_val_n)
y = np.sum(eig_val)
print(x/y)
```

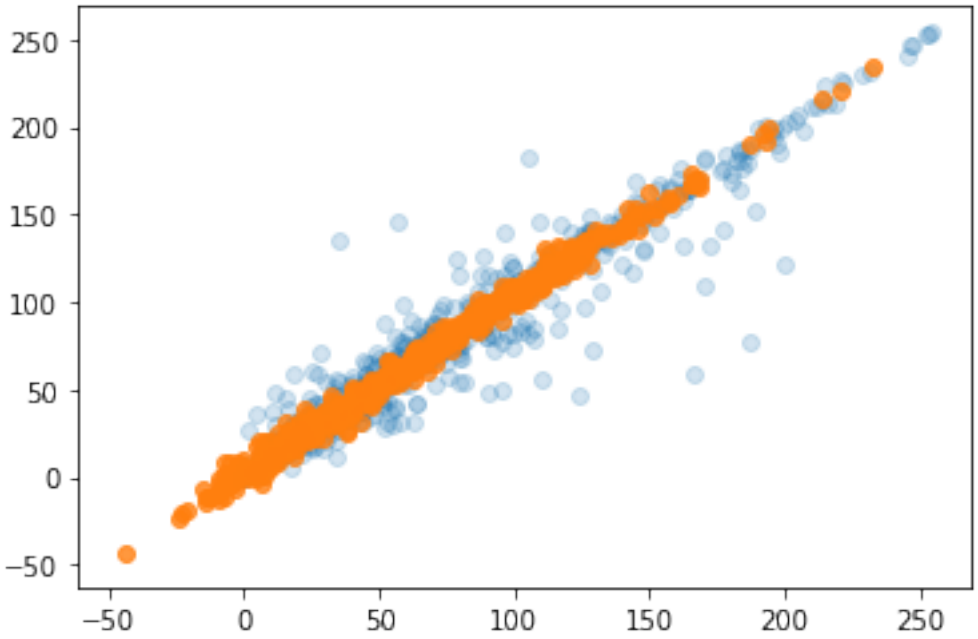
```
(0.804354886673874-5.750783086893538e-33j)
```

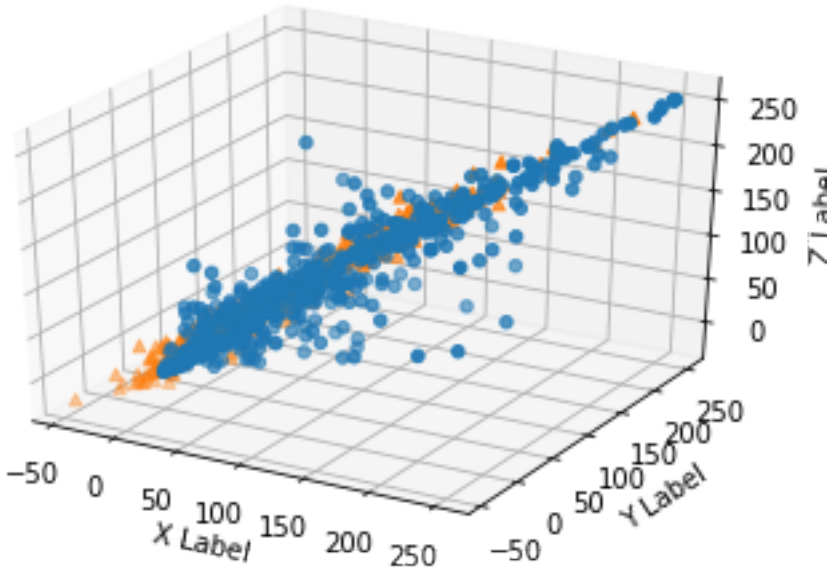
[48]:
```
## for N = 24, mean square error is less than 20%
```

1.1 pca.scatter_plot_pca(final_score,X)

[49]:
```
pca.scatter_plot_pca(final_score,X)
```






250
200
150
100
50
0
Z Label
−50
0
50
100
150
200
250
X Label
250
200
150
100
50
0
−50
Y Label

[ ]: