```python
#
# svd_faces.py
#

import matplotlib.pyplot as plt

import numpy as np
import numpy.linalg as LA
import argparse
import  scipy.misc   as misc
import glob, os
import cv2
from sklearn import decomposition
import time

def reconstr_matrix(U,D,V,k):
    rec_mat=np.dot(U[:,:k],np.dot(D[:k,:k],V[:k,:]))
    return rec_mat

def rgb2gray(rgb):

    if(len(rgb.shape)>2):
        r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]
        L = 0.2989 * r + 0.5870 * g + 0.1140 * b
    else:
        L = rgb
    return L

def ParseArguments():
    parser = argparse.ArgumentParser(description="Project ")
    parser.add_argument('--data-dir', default="", required=True, help='data dir  (default: %(default)s)')
    parser.add_argument('--alg', default="pca", required=False, help='pca, nmf  (default: %(default)s)')
    args = parser.parse_args()

    return  args.data_dir, args.alg


data_dir, alg =  ParseArguments()

classes=[]

for file in glob.glob(data_dir+"/**"):
        tmp=file.split('/')
        print(tmp)
        classes.append(tmp[len(tmp)-1])

print("classes =", classes)

data_classes=np.zeros(0,dtype=np.int)

classes_names=classes;

counter_tmp=0;

points=np.array([])

for classs in classes:
    list1=glob.glob(data_dir+"/"+classs+"/**");
    for file in list1:


        face0 = plt.imread(file)
        face = rgb2gray(face0)

        h=face.shape[0]
        w=face.shape[1]


        face2 = np.reshape(face,face.shape[0]*face.shape[1]);


        print("class = ", classs, "file = ", file," , shape = ", face.shape," , shape2 = ", face2.shape)

        #data_classes=np.append(data_classes,int(classs));
        data_classes=np.append(data_classes,int(counter_tmp));


        if(points.size==0):
            points=face2;
        else:
```

```
            points=np.vstack((points,face2))

    counter_tmp=counter_tmp+1

X=points.transpose()


U, Sigma, VT = np.linalg.svd(X, full_matrices=False)
D=np.diag(Sigma)
print("X:", X.shape)
print("U:", U.shape)
print("Sigma:", Sigma.shape)
print("V^T:", VT.shape)



ile=5
nr=1


fig0 = plt.figure(1)



fig = plt.figure(1)
st = fig.suptitle("Eigenfaces ", fontsize="x-large")



for i in range(0,ile):
    for j in range(0, ile):
        plt.subplot(ile,ile,nr);
        #plt.title("Eigenface " + str(nr))
        eigenface=U[:,nr-1].reshape(h,w)
        #if(nr==1):
        eigenface=-eigenface
        #eigenface=X[:,nr-1].reshape(h,w)
        plt.imshow(eigenface, cmap=plt.get_cmap('gray'))

        plt.axis('off')
        nr=nr+1


nr_rec = 4
nr_rec2 = 6

#reconstruct
k=15



nr=1



fig2 = plt.figure(2)
st = fig2.suptitle("Reconstructed faces, max dim="+str(D.shape[0]), fontsize="x-large")


for i in range(0,nr_rec):
    im_nr = np.random.randint(X.shape[1])

    face_orig = X[:,im_nr].reshape(h,w)
    plt.subplot(nr_rec,nr_rec2+1,nr)

    plt.imshow(face_orig, cmap=plt.get_cmap('gray'))
    plt.title("Person "+str(im_nr))
    plt.axis('off')
    nr=nr+1
    dims=2
    for j in range(0,nr_rec2):
        plt.subplot(nr_rec,nr_rec2+1,nr)
        if(j==nr_rec2-2):
            dims=int(0.6*D.shape[0]-1)

        if(j==nr_rec2-1):
            dims=int(0.8*D.shape[0]-1)


        X_rek=np.dot(U[:,:dims],np.dot(D[:dims,:dims],VT[:dims,:]))
```

```
        face_rek = X_rek[:,im_nr].reshape(h,w)
        plt.imshow(face_rek, cmap=plt.get_cmap('gray'))

        plt.title("dim = "+str(dims))
        plt.axis('off')
        nr=nr+1
        dims=dims+2




# ~ model = decomposition.NMF(n_components=10, init='random', random_state=0)
# ~ W=model.fit_transform(X)
# ~ H=model.components_


if(alg=="nmf" or alg=="NMF"):


    p_red=min(int(h/2),int(w/2))

    print("Calculating NMF...", end="", flush=True)
    start_time = time.time()

    model = decomposition.NMF(n_components=p_red, init='random', random_state=0)
    W=model.fit_transform(X)
    H=model.components_
    print("\t\t took %s seconds " % round((time.time() - start_time),5))



    nr=1;
    ile=4
    how_much_rec=0
    for i in range(0,ile):
        for j in range(0, ile):
            how_much_rec+=8;
            fig5=plt.figure(5);
            plt.subplot(ile,ile,nr);
            k=how_much_rec
            #rr_eigenimage = np.dot(U[:,k:(k+1)],V[k:(k+1),:])
            M_recons=np.dot(W[:,0:k],H[0:k,:])
            plt.imshow(M_recons, cmap=plt.get_cmap('gray'))
            plt.title("i = " + str(how_much_rec))
            plt.axis('off')
            nr=nr+1

    fig5.suptitle("NMF reconstructed")


    nr=1;
    ile=4
    how_much_rec=0
    for i in range(0,ile):
        for j in range(0, ile):
            how_much_rec+=1;
            fig6=plt.figure(6);
            plt.subplot(ile,ile,nr);
            k=how_much_rec
            #rr_eigenimage = np.dot(U[:,k:(k+1)],V[k:(k+1),:])
            M_recons=np.dot(W[:,(k-1):k],H[(k-1):k,:])
            plt.imshow(M_recons, cmap=plt.get_cmap('gray'))
            plt.title("i = " + str(how_much_rec))
            plt.axis('off')
            nr=nr+1

    fig6.suptitle("NMF eigenimages")

plt.show()


quit()
```