CS168 Spring Assignment 5

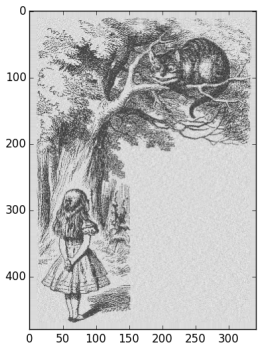
SUNet ID(s): johngold jsalloum
Name(s): John Gold Justin Salloum
Collaborators: None

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Part 1

(a) I think the image will will have not have much resemblence to the original image, but will have concentrated dark and light areas that roughly correspond to the distrubution of black and white pixels in the original image.

(b) Image, k = 150:



(c) That was the size of the original image, so when we ran SVD there are only 342 entries in the diagonal of S. Therefore no more components were possible

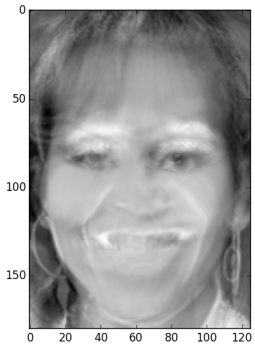
(d) We have 480 X 150 elements in U, 150 elements in S, and 150 X 342 elements in V. The original image had 164,000 elements, but only 123,000 were non-zero. With 150 componenets we are down to 123,000 elements, but about 60,000 are less than .000001. Therefore we only need about 63,000 units of memory (50%) compared to the original full rank image.

(e) Bonus: Most variance shows details, but the components with less variance shows up as gray when we don't include them. The first few components are the most essential to recovering the original image, while the components provided by higher values of k become less crucial to the bigger picture but more important for details - hence as k gets bigger the gray-blurring reduces.

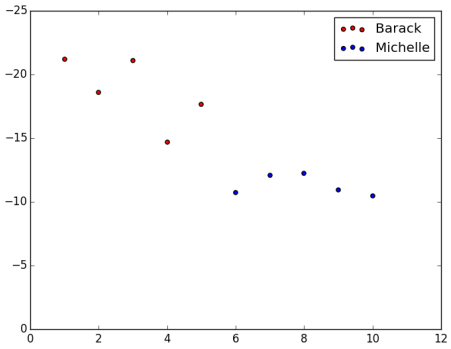
(f) Bonus: Nothing would change if we inverted the colors before SVD and then inverted them again afterwards. This happens because the SVD will return an inverted matrix from the inverted data, which then becomes the same as the original data after the second inversion.

Part 2

(a) "Eigenfaces"



(b) If we are looking at the plot of projections, we can separate the images by looking at the projection values. The projection of Michelle's face onto the first componenet are lower (higher negative value) than Barack's.



Part 3

(a) Why is there a major drop in the singular values M?? Because R is distributed gaussian, many values lie close to 0, so those values in M are also close to 0. However, some values are 1 or higher, and cause large jumps in those corresponding entries of M.

Therefore most of the variance comes from a few cells, but after that everything else is almost meaningless.

(b) The smaller (and better) norm comes from M' - M, so the SVD does indeed help.

(c) 0 is a reasonable choice, because the average value in M is close to 0 (it varies from trial to trial due to the randomness). Naively, if we had 4500 values and were missing 500, it would make the most sense to "guess" the missing values as the average of the 4500 values we had.

```
#p5.py
import scipy.misc
import scipy.linalg
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import sys
import os


def main():
    part = sys.argv[1]
    if(part == "1"):
        part1()
    if(part == "2"):
        part2()
    if(part == "3"):
        part3()



def part3():
    T = np.random.uniform(-1,1,(10,100))
    R = np.random.normal(0,1,(50, 10))
    M = R.dot(T)
    U, s, Vh = scipy.linalg.svd(M)
    #print s #(or plot S)
    np.set_printoptions(threshold='nan')
    #part B
    M_hat = M + np.random.normal(0, .1, (50, 100))

    #randomly set 500 of these to 0
    num_removed = 0
```

```
    while(num_removed < 500):
        row = np.random.uniform(0, 50)
        col = np.random.uniform(0, 100)
        if(M_hat[row, col] != 0):
            M_hat[row, col] = np.average(M)
            num_removed += 1

    #print M_hat

    M_prime = get_SVD_reduced(M_hat, 10)

    print np.linalg.norm(M_prime - M)
    print np.linalg.norm(M_hat - M)


#for each image in the folder
    #read in image
def part2():

    img_data = np.zeros((10,22500))
    for row, filename in enumerate(os.listdir('./p5_dataset/')):
        img = scipy.misc.imread('./p5_dataset/' + filename)[:, :, 0].reshape(
        img_data[row, :] = img

    #center data
    centered = img_data/img_data.sum(axis=0, keepdims=True)
    #run SVD
    #get first component
    U, s, Vh = scipy.linalg.svd(centered)

    if(sys.argv[2] == "a"):
        plt.imshow(Vh[0].reshape(180, 125), cmap = cm.Greys_r)
        plt.show()
    if(sys.argv[2] == "b"):
        #Do projections
        barack = []
        michelle = []
        for row in range(centered.shape[0]):
            if(row < 5):
                barack.append(Vh[0].dot(centered[row]))
            else:
                michelle.append(Vh[0].dot(centered[row]))
```

```
        #Now plot the points
        b = plt.scatter(range(1,6),barack,c="r")
        r = plt.scatter(range(6,11),michelle,c="b")
        plt.legend([b, r], ["Barack", "Michelle"])
        plt.axis((0, 12, 0, -25))
        plt.show()


def get_SVD_reduced(matrix, k):
    U, s, Vh = scipy.linalg.svd(matrix)

    return U[:, :k].dot(np.diag(s[:k])).dot(Vh[:k, :])


def part1():
    k = int(sys.argv[2])
    img = scipy.misc.imread("p5_image.gif")

    inverted = True
    if(inverted):
        img = (img * -1) + 1

    reduced_matrix = get_SVD_reduced(img, k)

    if(inverted):
        reduced_matrix = (reduced_matrix * -1) + 1

    plt.imshow(reduced_matrix, cmap = cm.Greys_r)
    plt.show()



if __name__ == "__main__":
    main()
```