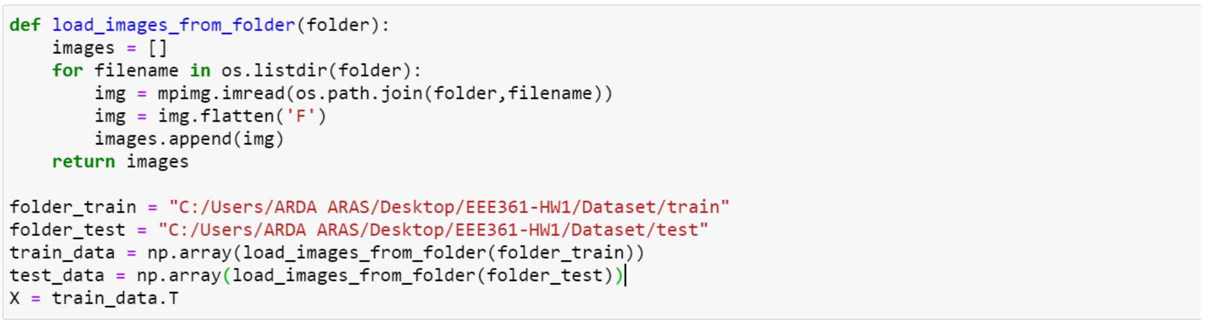
EEE361 HOMEWORK-1

Arda Can Aras

21704036

Bilkent EEE

The very first thing that need to be accomplished is to pre-process data to obtain desired **X**. With the given reference in the homework, the matrix composed as follows in the code segment below.



```
def load_images_from_folder(folder):
    images = []
    for filename in os.listdir(folder):
        img = mpimg.imread(os.path.join(folder,filename))
        img = img.flatten('F')
        images.append(img)
    return images

folder_train = "C:/Users/ARDA ARAS/Desktop/EEE361-HW1/Dataset/train"
folder_test = "C:/Users/ARDA ARAS/Desktop/EEE361-HW1/Dataset/test"
train_data = np.array(load_images_from_folder(folder_train))
test_data = np.array(load_images_from_folder(folder_test))
X = train_data.T
```

Observe that for the first question we are only dealing with the training data. Therefore, **X** is only composed of training data. It has 361 rows and 2429 columns. Since all images in training set has 19x19 pixels resolution, when we vectorize them they become 361x1. Therefore, every column of **X** corresponds to single sample image in vectorized form which obtained from the training data set.

Question 1

We have two parts for this question. First, we are asked to factorize matrix by using SVD and then by using NMF.

In this part, X matrix is formed by flattening the images in 'F' order which is concatenation in column order. Then, flattened images are added as columns of X vector. Therefore, last shape of X is (361,2429).

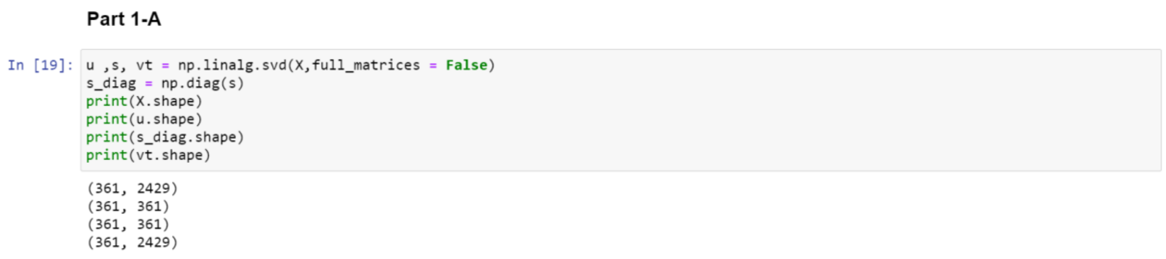
SVD factorization should be done as:

$$X = U\Sigma V^T$$

where Σ is (361,361), U is (361,361) and V is (2429,361). Σ has diagonal entries σ_i 1 <= i < = 361.

Part 1

For the first part of the question the first task is to factorize **X** by using SVD. The following code block will accomplish this task.



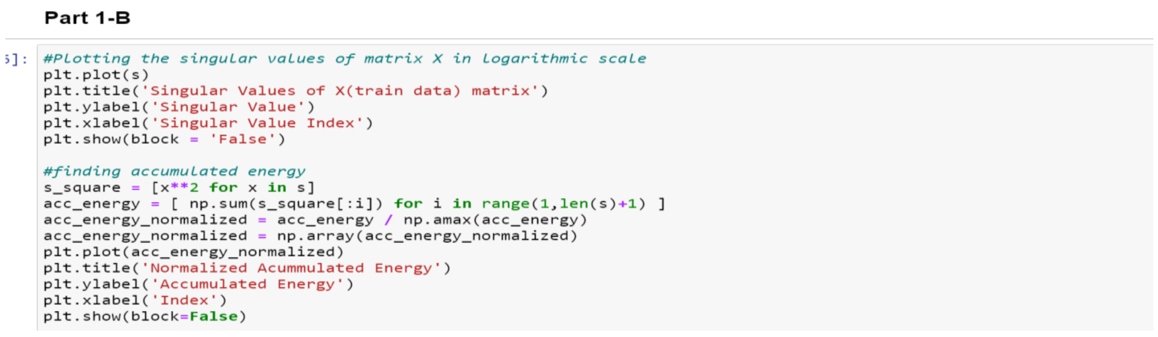
Part 1-A

```
In [19]: u ,s, vt = np.linalg.svd(X,full_matrices = False)
         s_diag = np.diag(s)
         print(X.shape)
         print(u.shape)
         print(s_diag.shape)
         print(vt.shape)

         (361, 2429)
         (361, 361)
         (361, 361)
         (361, 2429)
```

From the figure above we can individually observe the size of matrixes. Where first entry in the output corresponds the row and the second one is column.

Then we are asked to plot the singular values of **X** and plot the accumulated energy. Following lines of codes used for that purpose.



Part 1-B

```
5]: #Plotting the singular values of matrix X in logarithmic scale
    plt.plot(s)
    plt.title('Singular Values of X(train data) matrix')
    plt.ylabel('Singular Value')
    plt.xlabel('Singular Value Index')
    plt.show(block = 'False')

    #finding accumulated energy
    s_square = [x**2 for x in s]
    acc_energy = [ np.sum(s_square[:i]) for i in range(1,len(s)+1) ]
    acc_energy_normalized = acc_energy / np.amax(acc_energy)
    acc_energy_normalized = np.array(acc_energy_normalized)
    plt.plot(acc_energy_normalized)
    plt.title('Normalized Acummulated Energy')
    plt.ylabel('Accumulated Energy')
    plt.xlabel('Index')
    plt.show(block=False)
```

We can obtain following plots when we execute the cell above.

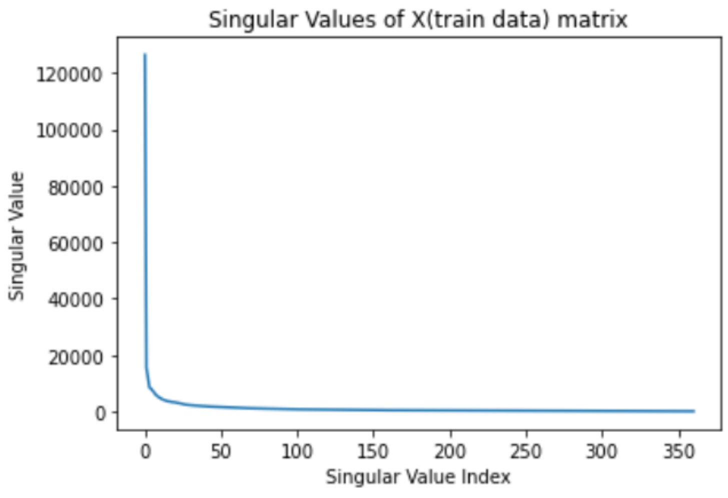


Figure 1: Singular Values of X versus Singular Value Index

From figure above we can observe that there is a significance decrease between the singular values. So, we expect to have more importance in the very first columns of **U**. This issue will be discussed later.

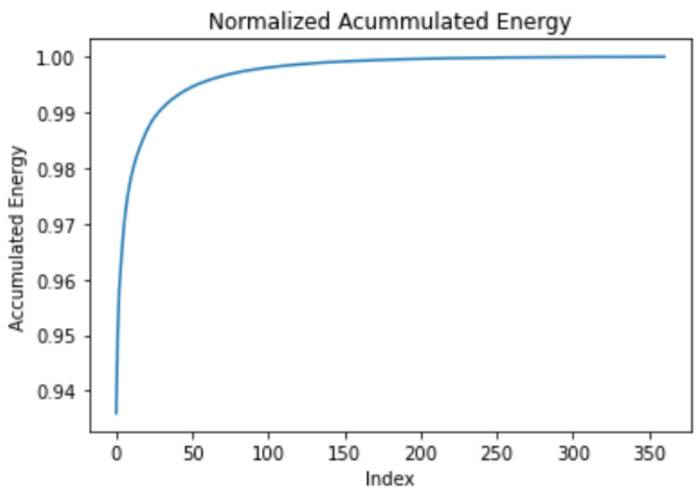
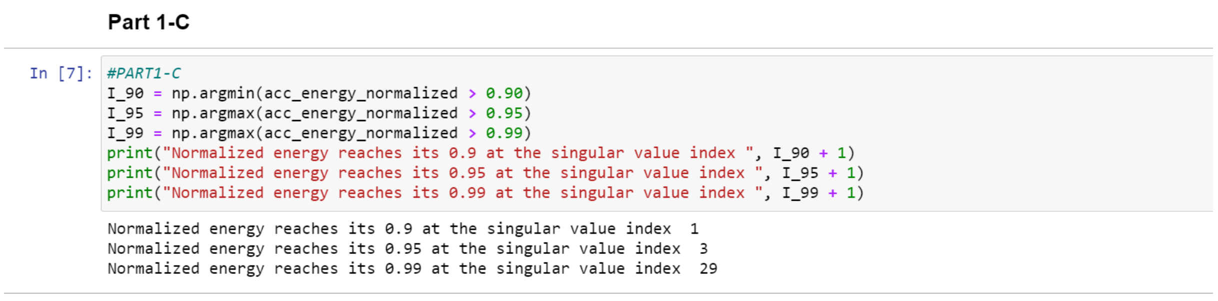


Figure 2: Normalized Accumulated Energy versus Index

We can also understand from the accumulated energy graph that, the energy reaches its 90 percentage by only considering the first index.

Then we are asked to identify some indices as defined in the homework. Following lines of code will be used to do that.



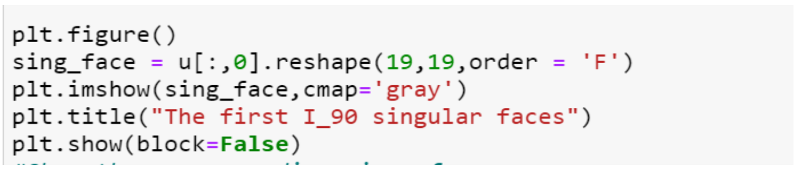
Part 1-C

```
In [7]: #PART1-C
        I_90 = np.argmin(acc_energy_normalized > 0.90)
        I_95 = np.argmax(acc_energy_normalized > 0.95)
        I_99 = np.argmax(acc_energy_normalized > 0.99)
        print("Normalized energy reaches its 0.9 at the singular value index ", I_90 + 1)
        print("Normalized energy reaches its 0.95 at the singular value index ", I_95 + 1)
        print("Normalized energy reaches its 0.99 at the singular value index ", I_99 + 1)

        Normalized energy reaches its 0.9 at the singular value index  1
        Normalized energy reaches its 0.95 at the singular value index  3
        Normalized energy reaches its 0.99 at the singular value index  29
```

From the output we can understand the corresponding indexes *of* I_{90}, I_{95} and I_{99}.

Then we are asked to check the first I_{90} columns of **U**. The following lines of code can be used to accomplish this task.



```
plt.figure()
sing_face = u[:,0].reshape(19,19,order = 'F')
plt.imshow(sing_face,cmap='gray')
plt.title("The first I_90 singular faces")
plt.show(block=False)
```

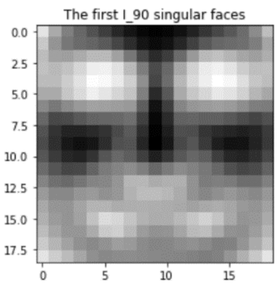


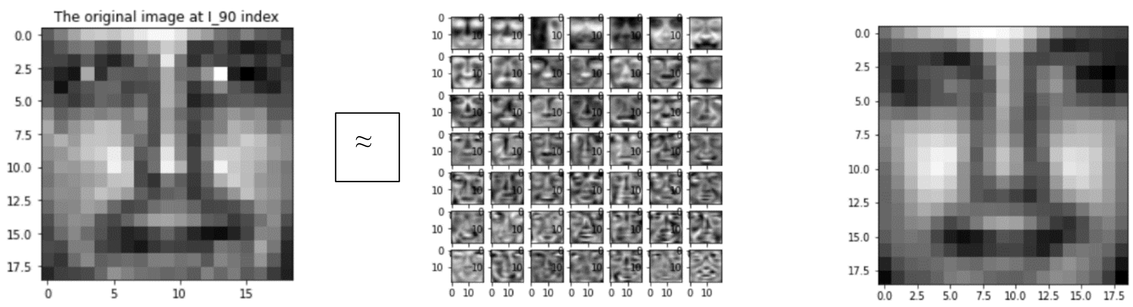
Figure 3: The first I_90 singular faces.

Since I_{90} equals to 1, we only have a single image. So, the matrix U includes the eigen faces in each column that we can totally recover and single image by using all the columns (eigen faces) in U. However, what is more interesting is with less effort we can really approximate the image by using the first r columns of U for $1 \leq r \leq 361$ for our data set. As r reaches to 361, we can totally recover the image. Let us observe this affect by using r = 49.

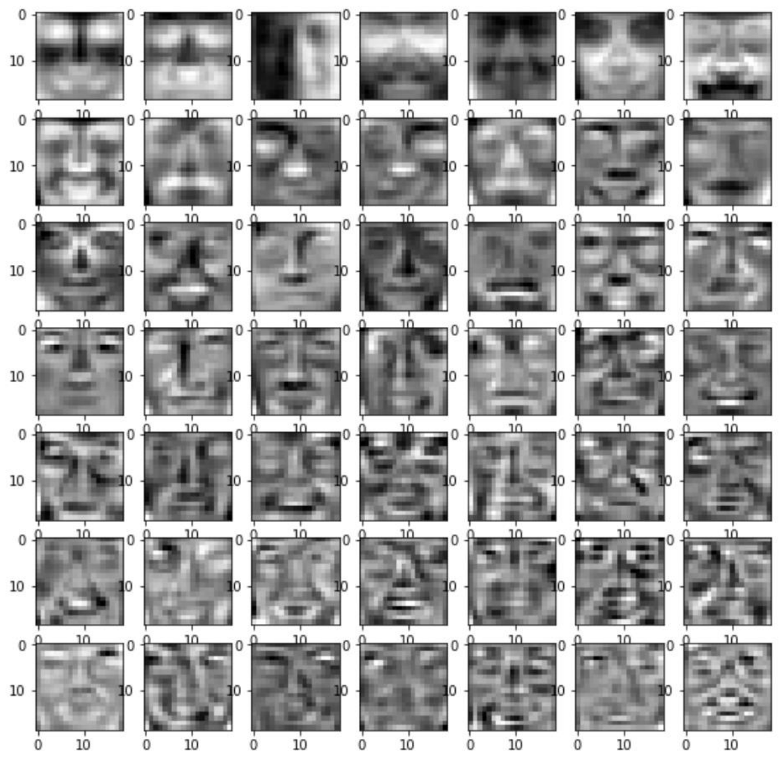
$$X_{approx} = U_{361xr}\, S_{rxr}\, V^T_{rx2429}$$

By using the formula above, we can approximate whole matrix and retrieve any image we want by simply selecting the specific column.

$$X(:,j) \approx X\_approx(:,j) \text{ (where X\_approx given as above)}$$



The figure in the middle corresponds the first r = 49 singular faces. Note that this approximation is different than the visualization in the reference [1] at the homework. In that scenario we use all the columns of W to approximate the single image to show that our X ≈ WH. Since SVD is exact factorization and we can fully recover the first image by using all the columns of U (d) section of Question 1 part1 need to be treated differently. Since the mentioned Figure 1 of [1] has different kind of approximation. The following figure is the bigger version of the singular faces above.



When I observe the eigen faces, I found that they have distributed features. We can observe that all the structures of face distributed evenly among the structures like eyes, nose, and eyebrows. When we investigate the features of W in NMF the meaning of distributed features will be clearer. Also entries of the matrix are non-negative.

Part 2

In that part we are asked to use Non-negative Matrix Factorization (NMF) to factorize matrix X. First, we are asked to use technique called HALS for factorization. Following line of codes can be used to implement this method and its updating policies. But first we need to initialize by using SVD based technique. I simply followed the instructions given in the reference [1] of homework to implement these functions. To improve the computation time, for loops in summations can be replaced by direct matrix multiplications. Therefore, I used some derivations to obtain better result.

For this part I initialize with r=49. Therefore, W matrix has 49 columns and H has 49 rows. Since r is user defined choice, I designed algorithm such that it can be computed for any r values. I obtained the following error versus iteration graph. Please check the Appendix A to see full derivations that implemented in code.

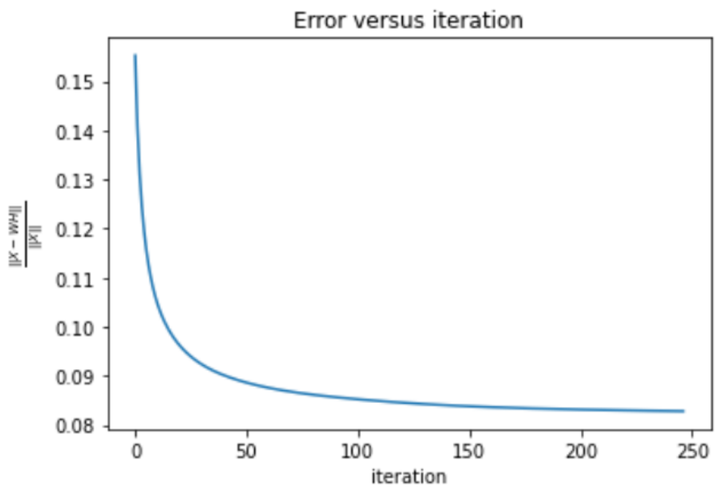


Figure 4: Error versus iteration graph for r=49.

From the figure above we can see that even the first iteration does not have large error. As we keep updating the W and H matrices, our error converges approximately to 0.10. Therefore, we can conclude that our approximation done a great job. Next, we can plot the corresponding eigenfaces which are columns of W.

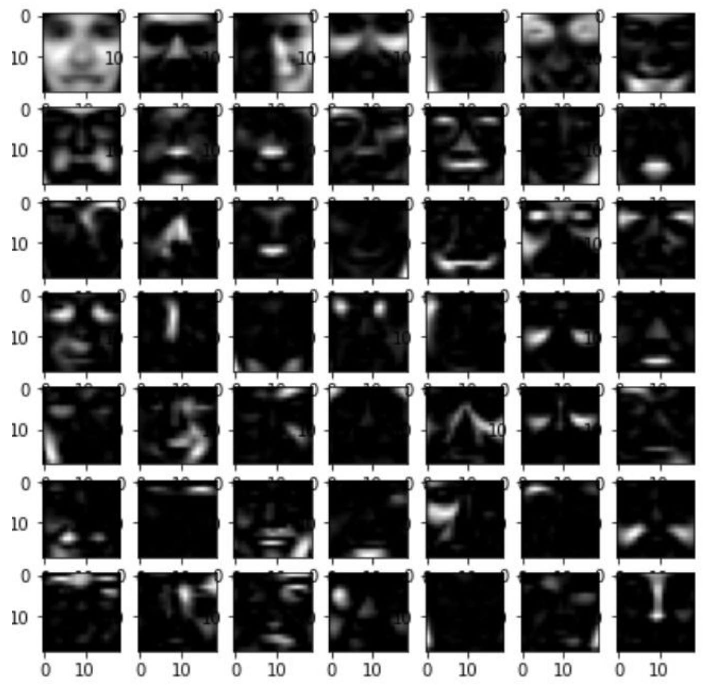
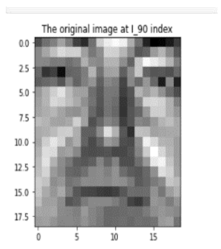
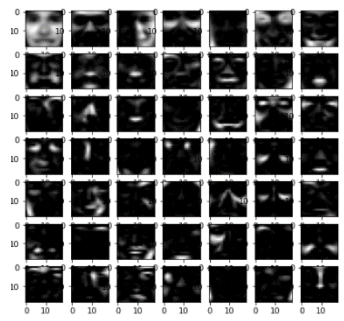


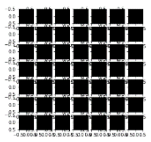
Figure 5: Eigenfaces (columns of W).

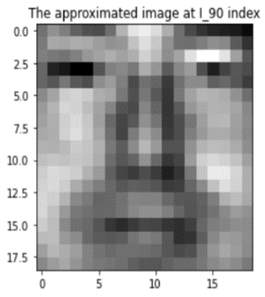
We can observe that only a single portion of face (mouth, lips, eyebrows) is lighted. When I observe the entries of the columns of W they are composed of non-negative values and positive values clumps together. So, we can conclude that it has localized feature with tries to enlighten a specific area in the face. Therefore, we can conclude that we reached our task which was to minimize the error with the constraint both W and H having non-negative entries. As it was shown in the reference [1] of the homework we can obtain the image and its approximate by using W and H.

$$\underbrace{X(:,j)} \approx \sum_{k=1}^{r} \underbrace{W(:,k)} \underbrace{H(k,j)} = \underbrace{WH(:,j)} .$$









jth facial image

facial features

importance of features in jth image

approximation of jth image

Question 2

For that question we are asked to reconstruct the noisy version of the images at the test data set and observe the effect of noise. At the previous part we had factorize our training data matrix by SVD and NMF. Since we compose singular faces and eigenfaces matrices from them, we expect to see that for a given image in test data set, by using some columns of U and W we can obtain the approximate image. For the first part we will deal with SVD based reconstruction. When I decompose the matrix X by using SVD and reconstruct the noisy version images from the columns of U, I obtained the following graphs for different noise levels.

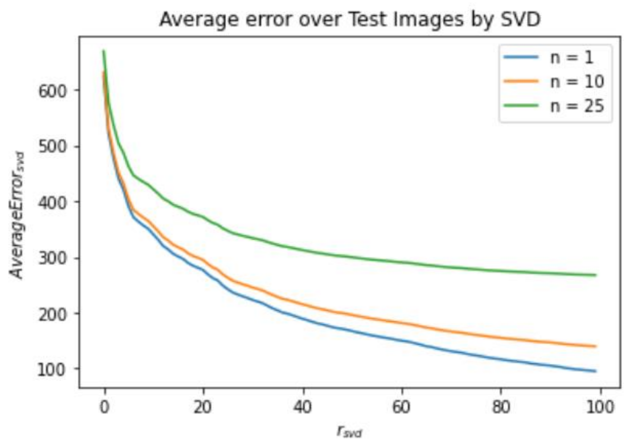


Figure 6: Error of SVD for different values of r when n = 1,10 and 25

From the figure above, we can conclude that as we include more columns from the matrix U, our approximation gets better and better. There is an exponential decay trend as it can be seen from the graphs in the figure above. To fasten the computations, we can find simple matrix multiplication to replace summation and dot products. Check the appendix B to see full derivations.

Then we are asked to reconstruct noisy images from the columns of W. After I followed the instructions of the reconstruction, I obtained the following graphs.

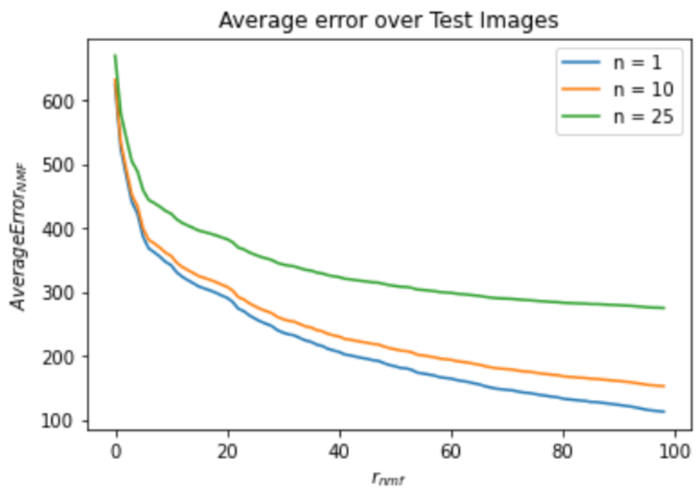


Figure 7: Error of NMF for different values of r when n = 1,10 and 25

We can also observe the same trend above like. However, initial error for NMF is larger than SVD.

Question 3

For this question we are asked to plot two graphs showing the error versus r values for SVD and NMF factorization. Since we already factorize the matrix X, we need to compose masked array and follow the instructions in homework. As a result, I obtained following graphs.

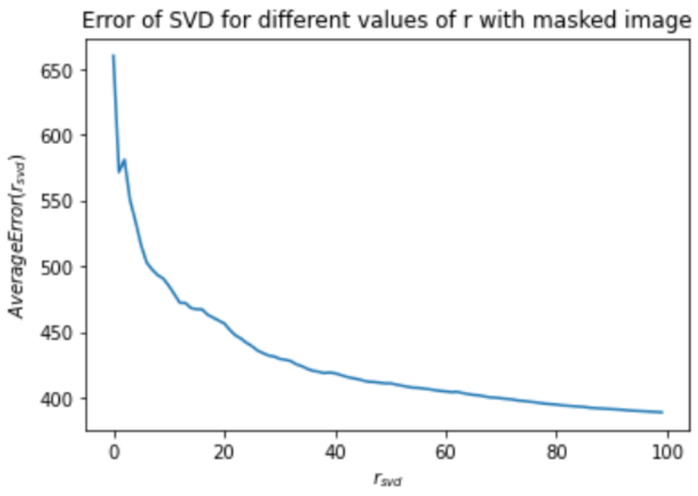


Figure 8: Error of SVD for different values of r with masked image.

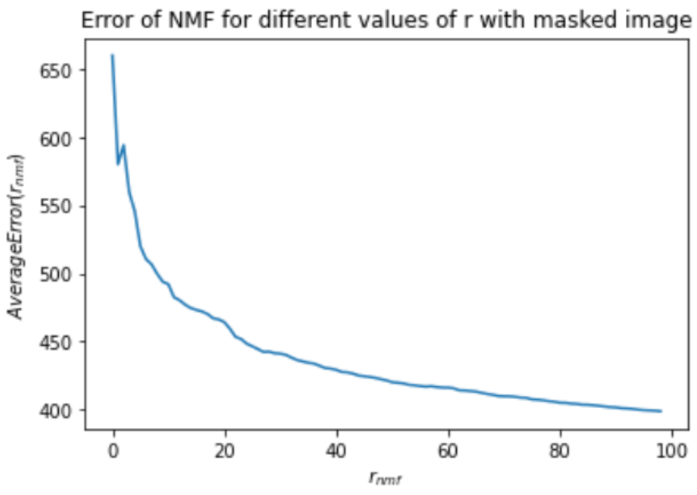


Figure 9: Error of NMF for different values of r with masked image.

When we compare the figures above, they have similar trends. As we increase the r, our approximation gets better and error decreases. Even we are asked to determine for first 100 r values, derivations of SVD speed up to process so I can compute for all the r ranging from 1 to 361. The following graph is obtained.

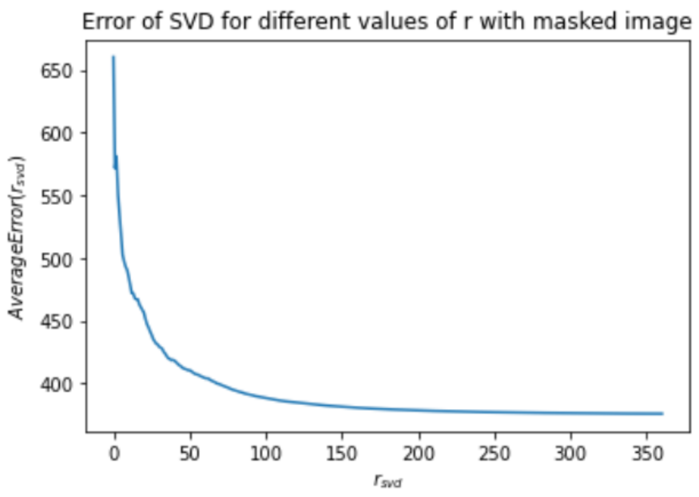
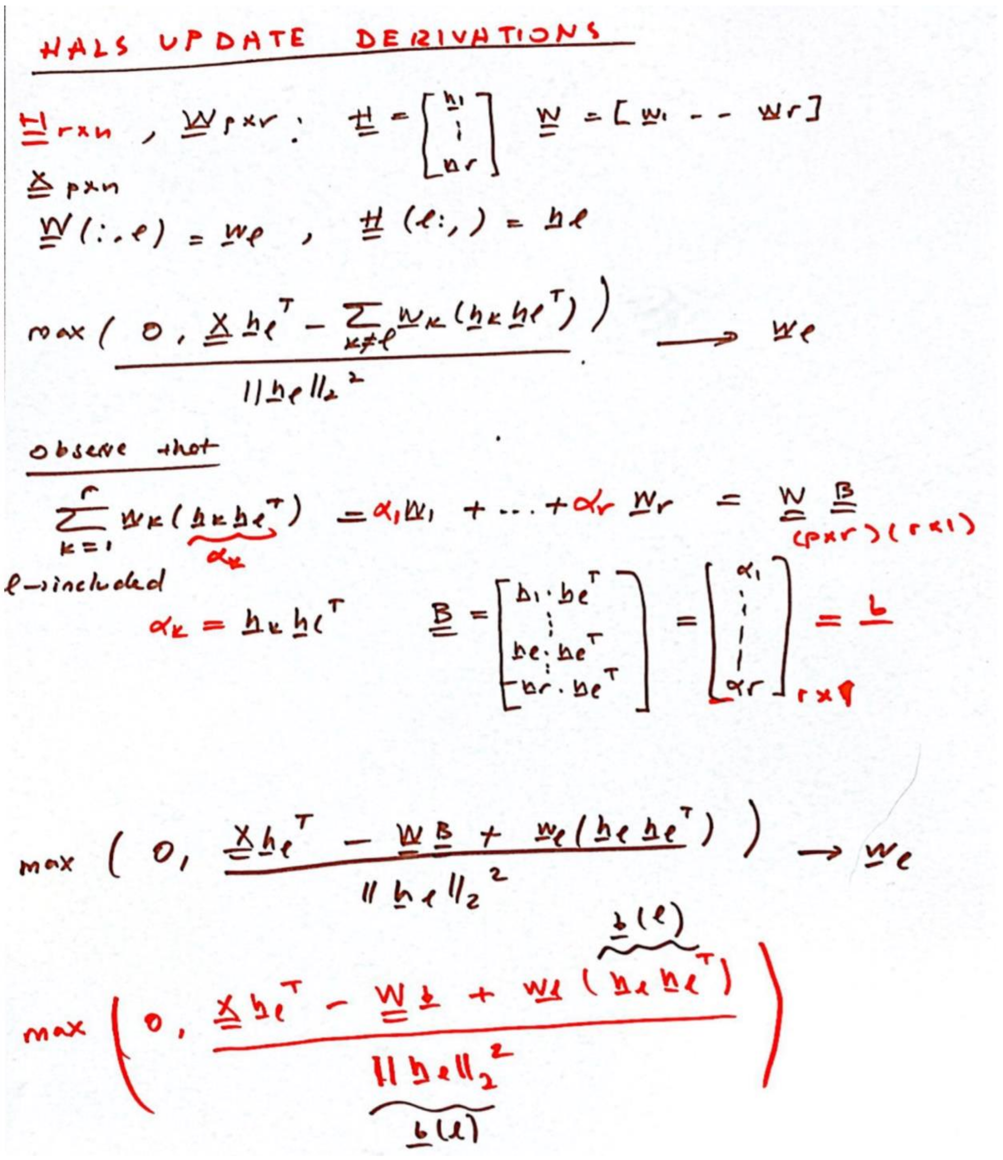


Figure 10: Error of SVD for all values of r with masked image.

At the end error converges to 376.09 when we use all of the columns and this is the best approximate we can obtain by using SVD.

Appendices

Appendix A.



HALS UPDATE DERIVATIONS

$\underline{\underline{H}}_{r\times n}$, $\underline{\underline{W}}_{p\times r}$: $\underline{\underline{H}} = \begin{bmatrix} h_1 \\ \vdots \\ h_r \end{bmatrix}$ $\underline{\underline{W}} = [\, w_1 \, - - \, w_r \,]$

$\underline{\underline{X}}_{p\times n}$

$\underline{\underline{W}}(:,\ell) = w_\ell$, $\underline{\underline{H}}(\ell:,) = h_\ell$

$$\max\left(0, \frac{\underline{\underline{X}}\, \underline{h}_\ell^T - \sum_{k\neq\ell} w_k (\underline{h}_k \underline{h}_\ell^T)}{\|\underline{h}_\ell\|_2^2}\right) \longrightarrow w_\ell$$

observe that

$$\sum_{k=1}^{r} w_k (\underbrace{\underline{h}_k \underline{h}_\ell^T}_{\alpha_k}) = \alpha_1 w_1 + \dots + \alpha_r w_r = \underset{(p\times r)}{\underline{\underline{W}}}\ \underset{(r\times 1)}{\underline{\underline{B}}}$$

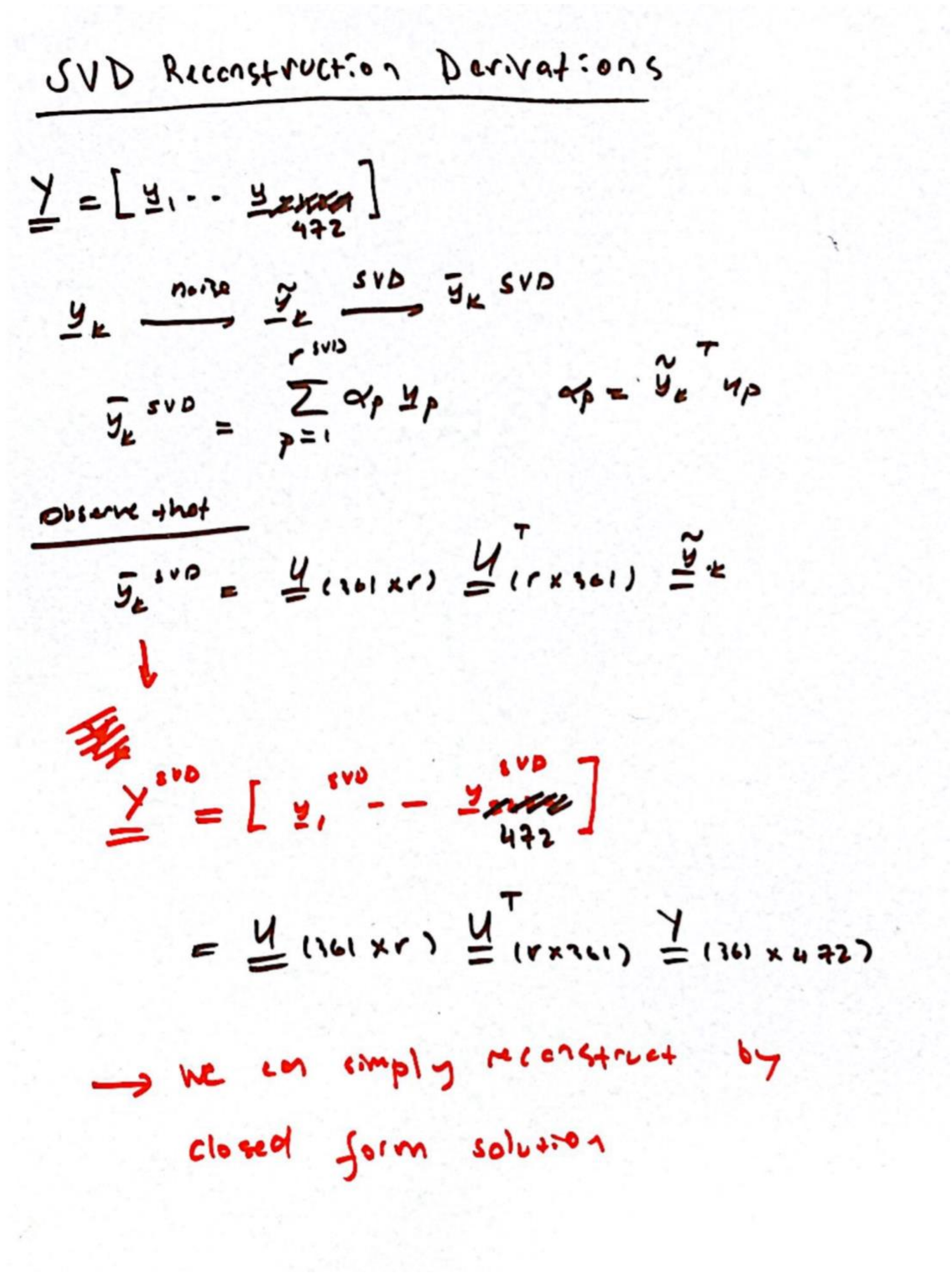
$\ell \rightarrow$ included

$$\alpha_k = \underline{h}_k \underline{h}_\ell^T \qquad \underline{\underline{B}} = \begin{bmatrix} h_1 \cdot h_\ell^T \\ \vdots \\ h_\ell \cdot h_\ell^T \\ \vdots \\ h_r \cdot h_\ell^T \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \vdots \\ \alpha_r \end{bmatrix}_{r\times 1} = \underline{b}$$

$$\max\left(0, \frac{\underline{\underline{X}}\, \underline{h}_\ell^T - \underline{\underline{W}}\, \underline{\underline{B}} + w_\ell (\underline{h}_\ell \underline{h}_\ell^T)}{\|\underline{h}_\ell\|_2^2}\right) \rightarrow w_\ell$$

$$\max\left(0, \frac{\underline{\underline{X}}\, \underline{h}_\ell^T - \underline{\underline{W}}\, \underline{b} + w_\ell \overbrace{(\underline{h}_\ell \underline{h}_\ell^T)}^{\underline{b}(\ell)}}{\underbrace{\|\underline{h}_\ell\|_2^2}_{\underline{b}(\ell)}}\right)$$

Appendix B.



SVD Reconstruction Derivations

$$\underline{\underline{Y}} = [\,\underline{y}_1 \cdots \underline{y}_{472}\,]$$

$$\underline{y}_k \xrightarrow{\text{noise}} \underline{\tilde{y}}_k \xrightarrow{\text{SVD}} \bar{y}_k^{\text{SVD}}$$

$$\bar{y}_k^{\text{SVD}} = \sum_{p=1}^{r^{\text{SVD}}} \alpha_p \underline{u}_p \qquad \alpha_p = \tilde{y}_k^T u_p$$

Observe that

$$\bar{y}_k^{\text{SVD}} = \underline{\underline{U}}_{(361 \times r)} \underline{\underline{U}}^T_{(r \times 361)} \underline{\tilde{y}}_k$$

$\downarrow$

$$\underline{\underline{Y}}^{\text{SVD}} = [\,\underline{y}_1^{\text{SVD}} - - \underline{y}_{472}^{\text{SVD}}\,]$$

$$= \underline{\underline{U}}_{(361 \times r)} \underline{\underline{U}}^T_{(r \times 361)} \underline{\underline{Y}}_{(361 \times 472)}$$

$\rightarrow$ we can simply reconstruct by closed form solution