Project 3

Real-time Face Recognition using SVD

December 23, 2022

Group #8
Tamar Leiser
Lakshmikar Reddy Polamreddy

Numerical Methods, AI-5003
Topics in Scientific Computing, MAT-5402
Fall 2022
Yeshiva University

Contents

1 Abstract **1**

2 Introduction **2**

2.1 Singular Value Decomposition 2

3 Activity 1 **3**

3.1 Loading the Images 3

4 Activity 2 **4**

4.1 Vector Representation of Images 4

4.2 Mean Face 4

5 Activity 3 **5**

5.1 Splitting the Dataset 5

6 Activity 4 **6**

6.1 Query Function 6

7 Activity 5 **7**

7.1 Analysis of Full Image Space Recognition for 6-5 Split 7

7.2 Analysis of Full Image Space Recognition for Different Splits 7

7.3 Analysis of Full Image Space Recognition for Random Split 8

8 Activity 6 **8**

9 Activity 7 **9**

10 Activity 8 **10**

10.1 Analysis of Reduced Face Space with $p = 10$ 10

10.2 Analysis of Reduced Face Space with More p-Values 10

10.3 Plotting the Reduced Face Space 11

10.4 Analysis of Reduced Face Space Recognition for Random Split 12

10.5 Summary of Results 12

11 Conclusion **13**

References **13**

1 Abstract

In this project, we have explored the application of SVD (Singular Value Decomposition) to real-time face recognition problem. Initially, we made an attempt to identify faces in the full image space and calculated metrics such as accuracy and time taken to perform face recognition. Then, we projected the original faces into the reduced image space formed by using the SVD technique. When we performed face recognition in the reduced image space, we found that accuracy has slightly decreased but time taken has drastically reduced. After comparing the results of face recognition in both spaces, we have come to a conclusion that the reduced space obtained using SVD is more efficient and better than that of full image space, despite the small reduction in accuracy, due to the large improvement of efficiency which is more applicable for real-time face recognition.

2 Introduction

Face recognition refers to establishing the identity of an individual by processing the facial image and comparing it with the existing database of images. In recent years, it has attracted wide attention and a lot of research is happening in this field because of various applications associated with it. For instance, we are also using this technique in our day to day activities like unlocking one's phone and biometric access to offices. However, just comparing each face with a database of faces takes a long time, due to the large size of images and the large number of comparisons that then need to be conducted. Real-time face recognition requires a much faster analysis of faces, because in these cases there are many faces being compared at once and results are needed quickly. Researchers in this field are aiming to perform face recognition more efficiently with still a high accuracy. In order to reduce the time taken for this activity, we have implemented Singular Value Decomposition in this project to compress the images of faces. Using this process decreases the time needed to run facial recognition. To analyze the viability of using SVD for real-time face recognition, we compare the accuracy and efficiency of face recognition using the full original face space to those when using SVD and a reduced face space. If we find that we don't lose too much accuracy while gaining much efficiency once we move to a reduced face space, then it will be viable to use the reduced face space for real-time face recognition.

2.1 Singular Value Decomposition

Singular Value Decomposition is a dimensionality reduction technique which allows us to compress images and represent each face by a smaller number of values, thus making the comparison process between faces quicker. The SVD process is detailed below:

If A is an m x n matrix where $m \geq n$, then there exists an SVD:

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

where:

U is an m x m orthogonal matrix,

V is an n x n orthogonal matrix, and

Σ is an m x n diagonal matrix, with the diagonal values the singular values $s_1 \geq s_2 \geq ... \geq s_r > 0$

To use SVD for dimensionality reduction and image compression, we need the best rank-p approximation of A, which is from the first p terms of:

$$A_p = \sum_{i=1}^{p} s_i u_i v_i^T$$

We use this p-value to reduce the dimension of the U matrix by taking the first p columns of U. We then project the data (A) into the reduced face space (U_p), to represent the data by the smaller matrix (B_p) of size p x n:

$$B_p = U_p^T A$$

Once the data is represented by a smaller matrix, we can conduct face recognition in the reduced face space, where each face is represented by fewer values. This will reduce the time needed to compare faces because we have to compare fewer values for each face.

3 Activity 1

3.1 Loading the Images

First, we loaded the 165 images from the Yale database, 11 faces for 15 different subjects, and made sure the images loaded correctly. The first 25 faces are shown in Fig. 3.1

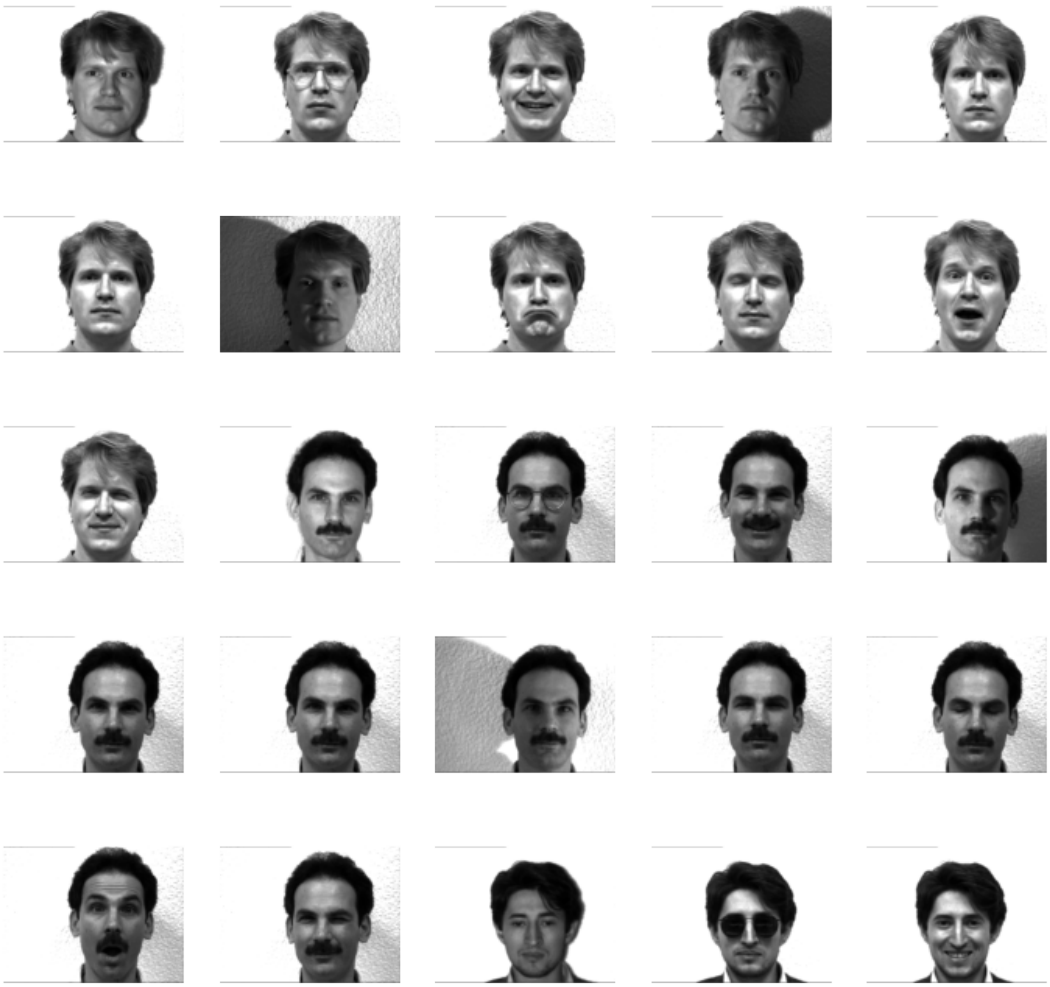

Figure 3.1 First 25 Faces from Yale Database

4 Activity 2

4.1 Vector Representation of Images

To be able to compare faces to each other for face recognition, we need vector representations of each image using the pixels of the faces. For this purpose, we created a large array (77760x165) where each column is the 1d vector representation of each face. Originally, each image is of the size 243x320, so we need to reshape each image to a 1d vector of size 77760.

To accomplish this, we created two functions, img2vec and vec2img, which convert images to 1d vectors and 1d vectors to images, respectively. Using the img2vec function, we were able to convert all 165 images to 1d vectors of size 77760 and make them the columns of the large array *dataset*.

4.2 Mean Face

To compare each face accurately, we also need to compute the mean face and normalize *dataset*. This is found by the equation:

$$\overline{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$$

We used the NumPy mean function to compute the mean face across all columns of *dataset*. The mean face is shown in Fig 4.2. We were then able to normalize *dataset* by subtracting the mean face from each column vector.

Figure 4.2 Mean Face

5 Activity 3

5.1 Splitting the Dataset

Our next step was to split *dataset* into a set of known faces and a set of unknown faces to query. We want to make sure to split *dataset* by subject so that each subject has the right amount of faces in the known and unknown. Otherwise, we could end up with all faces of one subject in the unknown set, for example, which would obviously mean that face recognition could never give us the correct answer.

We created functions to accomplish this in two different ways: manually or randomly. The manual way will split each subject by making the first x faces known, and the last $11 - x$ faces unknown. The random way will split each subject randomly, making a random x faces known and the other $11 - x$ faces unknown.

6 Activity 4

Our goal in this activity is to run face recognition in the original, full image space. To do this, we need to compare the vector representation of each face in the query set to all faces in the known set, and find the face which is closest in distance.

6.1 Query Function

To compare faces, we created a query function which takes in a queried face and compares it to every face in the known database. The vector representation of each face is subtracted from the queried face, and the l_2 norm is taken. The known face with the smallest norm distance is taken as the correct subject. We tested this function for two different faces, the results of which are shown in the figures below.

Figure 6.11 Queried face: subject01

Figure 6.12 Chosen face: subject01

Label of queried face: sad

Figure 6.13 Queried face: subject07

Figure 6.14 Chosen face: subject15

Label of queried face: rightlighting

The face with the extreme attribute of right lighting was not recognized correctly, while the face with a small difference (sad vs glasses) was recognized correctly. We can see that this function will not always be accurate, especially with those faces with more extreme attributes, but the function will be more accurate with the other faces without extreme attributes.

7 Activity 5

To test the face recognition using the full image space, we test each face in the query set against the known database and check if the subject chosen was indeed the correct subject of the queried face. We then report the accuracy as a percentage of the number of faces recognized correctly, and the efficiency as a measure of the total time taken to test every query face.

7.1 Analysis of Full Image Space Recognition for 6-5 Split

We first test this process by splitting *dataset* manually 6-5, meaning the first 6 faces of each subject is considered known, and the last 5 faces are considered unknown. The results found are:

Accuracy = 89.333%

Efficiency = $1.679s$

7.2 Analysis of Full Image Space Recognition for Different Splits

Next, we test the same process by splitting *dataset* 4-6, so 4 faces in the known database and 6 faces in the unknown query set, and by splitting *dataset* 8-3. The comparison of results are shown in the following table:

Table 1: Accuracy and Efficiency per Split

Known Set Size	**Accuracy** (%)	**Efficiency** (s)
6	89.333	1.679
4	88.571	1.690
8	97.778	1.441

We find that when we have a larger known set, there are less faces to query and more faces to compare against, so the accuracy increases, while the opposite is true for a smaller known set. With a larger known set and smaller unknown set, less faces are being tested so the efficiency is better because the entire process takes less time to complete. However, splitting the dataset this way means we are testing only a small number of faces, which is not accurate to real-time face recognition.

For future analysis, we will split *dataset* 6-4. This way, we are querying an adequate number of faces, while keeping a higher accuracy and efficiency.

7.3 Analysis of Full Image Space Recognition for Random Split

We now want to look at the accuracy and efficiency when splitting *dataset* randomly, as described in Section 3.1. Because each time we randomly split *dataset*, the results differ due to different random selections of faces for the known/unknown sets, we will look at the average accuracy and efficiency when completing this process 30 times. The comparison of results are shown in the following table:

Table 2: Accuracy and Efficiency for Manual vs. Random Split

Type	**Accuracy** (%)	**Efficiency** (s)
Manual	89.333	1.679
Random	79.289	1.646

Randomly splitting the data gives us a lower accuracy with about the same efficiency. We see that for the Yale database, manually splitting the faces gives us a better accuracy, but in general random splitting is more accurate to real-time face recognition so it is worthwhile to see this effect on the accuracy.

8 Activity 6

We now want to look at conducting face recognition using a reduced face space from SVD. The idea is to represent each face by a smaller dimension, which should improve the efficiency of the process.

To reduce dimensionality, we first have to compute the singular value decomposition: $F = U\Sigma V^T$. The reduced face space, or "base faces", is the first p columns of the U matrix.

We calculate the SVD using the known set as opposed to the entire dataset, because using the entire dataset will not give us more accuracy in face recognition as we're still projecting the same data into the reduced face space. This would only increase the time needed to reduce the dimension and conduct the face recognition, which is the opposite of what we want. Therefore, we compute the SVD of only the known set.

9 Activity 7

To choose our value of p for the reduced face space, we will plot the singular values from Σ as a function of values of p. This plot is shown below:

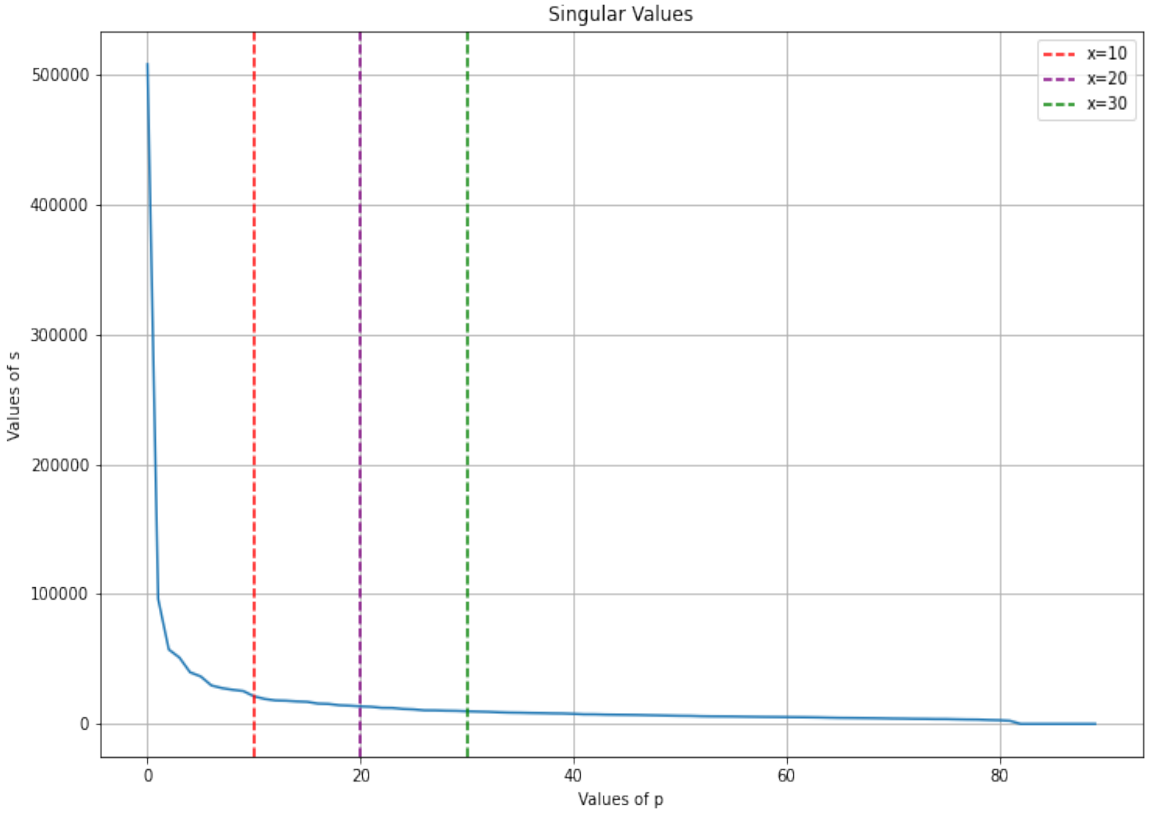

Figure 7 Singular Values

The vertical lines show the fraction of the face space that will be captured with each value of p. For $p = 10$, we already capture most of the singular values, so we will start with $p = 10$ and reduce the phase space. To do this, we take only the first 10 columns of the U matrix.

10 Activity 8

10.1 Analysis of Reduced Face Space with $p = 10$

To reduce the dimensionality, we project the data into the reduced face space by multiplying the transposed reduced matrix (U) with the known set (F) and the unknown set (Q):

Projected known set = $U^T F$

Projected unknown set = $U^T Q$

We now repeat the same process as in Activity 5 to look at the accuracy and efficiency of the face recognition, this time in the reduced space. The results found are:

Accuracy = 81.333%

Efficiency = $0.114s$

10.2 Analysis of Reduced Face Space with More p-Values

We now repeat the process with more values of p to see how this effects accuracy. We expect more values of p to give a better accuracy because the faces are represented by slightly larger vectors and are thus more easily recognizable. The results are compared in the following table:

Table 3: Increase in Accuracy Using SVD per p-Value

p	**Accuracy** (%)
1	22.667
10	81.333
20	84.000
30	86.667
40	86.667
50	86.667
60	88.000
70	86.667
80	89.333
90	89.333

We indeed find that higher p-values gives higher accuracy. The accuracy value, which is at 22.667% when $p = 1$, increased rapidly to 81.333% with p=10. It then increased to 86.667% with $p = 30$ and then slowly increased to 89.333% with $p = 70$, and remained constant from there. With a high enough p-value, we are close to the original full face space and we recover the accuracy found in Activity 5.

While the accuracy grows with p-values, the efficiency worsens. For each increase in p, the necessary computation to compare each face grows linearly, because we need to find the difference using one more value in the vector representing each face. For this reason, we want to reduce the face space enough that we gain efficiency, but not too much that we lose too much accuracy. Therefore, we will continue with $p = 30$, which gives an accuracy of 86.667% and efficiency of $0.064s$.

10.3 Plotting the Reduced Face Space

We can plot the base faces to visualize our reduced face space. To do this we first have to normalize the base faces to values $[0, 255]$, since after using SVD they are between $[0, 1]$. Once we do this, we have the following images:

Figure 7 Base Faces

10.4 Analysis of Reduced Face Space Recognition for Random Split

As in Activity 5, we also look at face recognition in the reduced face space using SVD for datasets split randomly. Once again, we look at the average accuracy and efficiency over 30 iterations, and the results found are:

Accuracy = 77.733%

Efficiency = $0.088s$

As expected, similar to in Activity 5 with the full face space, randomly splitting the data gives us a worse accuracy than when the data is split manually.

10.5 Summary of Results

A summary of results are compared in the following table:

Table 4: Accuracy and Efficiency Summary

Face Space	**Split Type**	**Known Set Size**	**Accuracy** (%)	**Efficiency** (s)
Full	Manual	4	88.571	1.690
Full	Manual	6	89.333	1.679
Full	Manual	8	97.778	1.441
Full	Random	6	79.289	1.646
Reduced	Manual	6	86.667	0.064
Reduced	Random	6	77.733	0.088

From these results, we can clearly see that using the reduced face space greatly improves efficiency without a big loss in accuracy as compared to similar analysis using the full face space. In both cases, randomly splitting the data gave a lower accuracy and efficiency.

11 Conclusion

In this project, we first performed face recognition in the full face space by splitting the given faces manually into known and unknown sets. We noticed that accuracy and efficiency (time taken) improved with an increase in the number of faces in the known set. When we have randomly split the faces, we achieved less accuracy when compared to the former. Secondly, we performed the same activity in a reduced face space using SVD and achieved a slightly lower accuracy when compared to the full face space but the time taken, in seconds, has reduced by a great margin as evident in the results shown above. We looked at different ways of reducing the face space to different sizes and compared these results, finding that a balanced middle will allow for a high enough accuracy with a high enough efficiency. We also once again looked at randomly splitting the faces as we did earlier, and found that the accuracy and efficiency lowered as compared to a manual split. However, even with a random split we found that efficiency greatly improved using the reduced face space as compared to the original face space. Based on our analysis of the results, we would recommend performing face recognition in the reduced face space obtained by using the SVD approach because of a slight compromise in accuracy for the benefit of a much higher efficiency, which is more applicable to real-time face recognition needs.

References

[1] Sauer, Timothy (2017). Numerical Analysis, 3rd. edition. Pearson. ISBN-13: 978-0134696454

[2] Class Notes: "12-Eigenvalues and SVD.pdf"

[3] `https://en.wikipedia.org/wiki/Singular_value_decomposition`

[4] `https://en.wikipedia.org/wiki/Dimensionality_reduction`

[5] `https://en.wikipedia.org/wiki/Facial_recognition_system`