Shape

Description automatically generated

**LINEAR ALGEBRA PROJECT REPORT**

Prepared by

**Huzaifa Adnan (200901092)**

**Mujtaba Abbas (200901070)**

Submitted to

**Dr. Majid Khan**

June 28, 2022

# Singular Value Decomposition & Image Compression

**Introduction:**

The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three other matrices. This concept may be found in linear algebra. It reveals fundamental geometrical and theoretical insights regarding linear transformations, in addition to having some fascinating algebraic features, and it does so in a concise manner. Additionally, it has several essential applications in the field of data science.

There are three MATLAB code files given,

1. readDatabase
2. meanImage
3. svd\_face\_recognition

**ReadDatabase File:**

We also build a function known as readDatabase in this file, and it requires an array of trainMat, testMat, trainLabel, testLabel, irows, and icols as its inputs. It is the responsibility of the function to read the database.

**trainMat** is going to be used to learn SVD

**testMat** queries are going to be used to test the performance of the system.

**trainLabel** is going to give us the id of the person to whom the picture belongs

**testLabel** is going to give us the id of the person whose image we are testing.

There are a total of 15 individuals in this collection, and 11 photos are associated with each person (4 images for testing and 7 for training to learn SVD).

Then, we set limages (length of images) to textscan (reads formatted data from text file or string) the file that was read from the dataset. Finally, we assign cwd to pwd (identifies the current position) and assign fid to read the file from the provided location of the dataset. After that, we give the limages the value that corresponds to the first position in the matrix of limages.

After that, we make a memory allocation, read the first photo from the path of the dataset, and put those images into a variable called im. After that, the irows and icols variables, both of which are declared inside the method, are allocated to the size function, which contains im as an array of objects. size(obj,dim) is a function that returns a value based on whether obj is an object or an array of objects and dim is the size of obj.

After that, the trainMat and trainLabel variables, together with the testMat and testLabel variables, are utilised with the zeros built-in function. The zeros function generates an array that contains only zeros. trainLabel is used for the storage of training labels, whereas testLabel is put to use for the storage of testing labels.

Then the value 1 is assigned to each of the variables that will be utilised in the for loop when it has been setup. in for loop beginning at k = 1, continuing until the length of the photos is reached. During this iteration of the loop, the picture is taken from the dataset and saved in the variable known as im. After that, the information in im has a variable named image attached to it.

We select the mod of k with 11 as our starting point because to the fact that in the dataset provided, each subject has 11 photos. Then, in accordance with the stipulated requirement that the values of mod k with 11 are between the range of larger than or equal to 1 and less than or equal to 4, the values and computations will be carried out inside the loop, and the count of various variables will also be brought up to date. If the modulo of k multiplied by 11 reaches zero, then the value of the picture id will be increased by 1, which indicates that the id of the person will change after eleven photographs, and then the loop will finish. And with that, the event has reached its conclusion.

Following the execution of this code file inside MATLAB, the database is loaded, and the application is then used to read the data from the dataset.

**Code**:

function [trainMat,testMat, trainLabel, testLabel,irows,icols]=readDatabase

% function reads the database and returns the data split in two matrices

% referred as trainMat and testMat.

% trainMat will be used to learn SVD or (will serve as Naive Reference matrix),

% whereas testMat rows will serve

% as our queries to test how well our system performs

%trainLabel will provide us the Id of person to which image belongs, similarly

%testLabel will provide us the Id of person we are testing...

%---------------------------------------------------------------

% Remember in the given database there are 15 subjects and each subject has 11 images.

% We will use each person 4 images for testing purposes and remaining

% 7 images for training purposes (serve as database) to learn the SVD.

% We will store each image as a row of the database matrix, we will also use label

% vectors to store the id (label) of each person, so the total dimensions of the database matrix will

% be (number of images) X (number-of-pixels-in-each-image );

%Open List file

fid=fopen('./dataset/images-original.txt','r');

cwd=pwd;

limages=textscan(fid,'%s\n');

limages=limages{1};

%Allocate Memory

im=imread([cwd './dataset/' limages{1}]); %

irows=size(im,1); % each image rows and columns

icols=size(im,2);

% create a big matrix to load the complete database...

%

trainMat=zeros(15\*7,irows\*icols); % rows= number of images, columns = # of pixels

trainLabel=zeros(1,15\*7); % to store the training labels...

testMat=zeros(15\*4,irows\*icols); % rows= number of images, columns = # of pixels

testLabel=zeros(1,15\*4); % to store the testing labels...

%Read the images in respective containers...

imid=1;

testcount=1;

traincount=1;

count=1;

for k=1:length(limages) % iterate over list of images

im=imread([cwd './dataset/' limages{k}]); % read the image..

%the code to populate the the trainMat,testMat, testLabel,trainLabel, ...

image = im(:);

image = image';

if( mod(k,11) >= 1 && mod(k,11) <=4 )

testMat(testcount,:) = image;

testLabel(1,testcount) = imid;

testcount = testcount + 1;

else

trainMat(traincount,:) = image;

trainLabel(1,traincount) = imid;

traincount = traincount + 1;

end

if mod(k,11) == 0

imid= imid+1 % id of person will change after eleven images...

end

end

**Output:**

**Table

Description automatically generated**

**Mean Image:**

The mean picture file is what's utilised to calculate the average of all the photographs included in the data set that's been provided.

In this step, we define a function called im that takes the form of an array and call it meanImage. After that, we create a variable called fid that opens the file (using fopen) from the dataset for which we provide the location. Additionally, it opens and reads the particular file that is included inside the dataset. Then, we create a variable called cwd = pwd, where pwd is an inbuilt function that determines the location of the current folder. After that, we give the dataset file a textscan function and establish a variable called limages, which stands for length of images. The Textscan function is used to read data that has been prepared from a text file or string. After that, we place limages in the first element of the matrix that contains limages.

After that, an empty array that is allocated to mim is created (which will store the mean of images). After that, we will create a for loop in which we will assign a variable k to the length of the limages, with k starting at 1. in which we supply the function name im to be equivalent to the imread command of the path from the dataset in which the limages will be counted. In the initial loop, one picture will be read, and then each subsequent image will be read after an increment of k times.

Then, we construct a condition that states that if the value of k is equal to 1, then the mim variable (which is an empty array) is given the value of double (im), where im is reading from the dataset, and double(x) returns the value of X with double precision. Then, in the otherwise loop, the value of mim will be changed, and it will be set to mim = double(im) + mim, where mim is the mean of the pictures; after this, the loop will be ended/executed.

Following the execution of the loop, the value of the variable mim will be set to contain the double datatype representation of the overall mean of the photos that were retrieved from the dataset.

After that, the value of the matrix mim is multiplied (./) and given to the im function so that it reflects the process by which matrices are multiplied. First row with first column, then each subsequent row with each subsequent column, and so on, until the whole length of the limages has been reached (read from the dataset). (./ denotes right division of the array).

Now im has been updated to include the average of all the photos that were retrieved from the dataset.

After that, we make use of the imshow() method to display the mean picture that is now saved in im. As a consequence of this, the image that is shown is the image that represents the mean of all of the total photos that have been read from the dataset. After then, our function is finished.

**Code:**

% script computes the mean image of the given database...

% It reads the database images listed in the file one by

function [im]=meanImage

fid=fopen('./dataset/images-original.txt','r');

cwd=pwd;

limages=textscan(fid,'%s\n');

limages=limages{1};

mim=[];

for k=1:length(limages)

im=imread([cwd './dataset/' limages{k}]);

if k==1

mim=double(im);

else

mim=double(im)+mim;

end

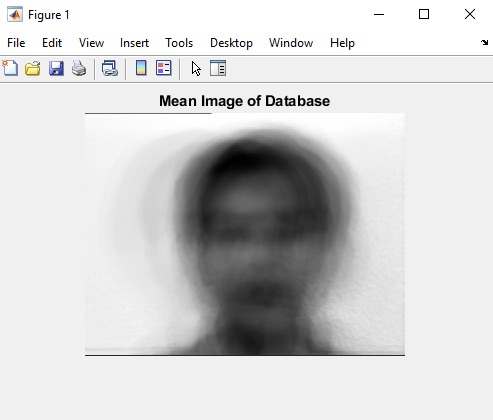
end

im=mim./length(limages);

figure,imshow(im,[]),title('Mean Image of Database');

end

**Output:**

****

**SVD face recognition:**

Following the execution of the readDatabase file comes the execution of this svd code file. The readDatabase function is used to read each and every element of an array. After that, we construct a db and assign zeros function, which generates an array of zeros, and then we assign values to variables such as queryimages, database images, and so on.

An economy-size decomposition of the m-by-n matrix A is produced by the expression [U,S,V] = svd(A,'econ'). After then, a variety of loops, conditions, and computations are carried out inside the loops. A variety of different scenarios, including for loops, while loops, and a lot of others, are tested, and various computations are carried out. As a consequence of this, a variety of goods and values are produced.

After all of this has been completed, the image is reformatted into distinct rows and columns. Then, im is changed into the uint8(datatype), and uint8(array) transforms the contents of an array into unsigned 8-bit (1-byte) integers of the class uint8. After then, the picture is shown.

The precision is then set to 0 before being computed, which comes about after a few iterations of computations within the loop. And then, on accuracy, Array right division (./) is conducted with the size of the testMat. After that, the accuracy is multiplied by 100, and then the result is written in the display message. Finally, on accuracy, Array right division (./) is performed with the size of the testMat.

[./ ===> Right division of the array The matrix containing the entries A(i,j)/B is denoted by the letter A./B. (i,j). Both A and B are required to have the same dimensions, with the exception of the case when one of them is a scalar.

**Code:**

% first read the database (both training and testing...)...

% test the accuracy of the system...

[trainMat,testMat, trainLabel, testLabel,rows,cols]=readDatabase;

queryImages = testMat';

databaseImages = trainMat';

n = size(testMat)

pause

%train SVD labels...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

meanImg=meanImage;

meanImg=meanImg(:);

meanImg=double(meanImg);

db = zeros( size(databaseImages,1) , size(databaseImages,2));

for i=1:size(databaseImages,2)

db(:,i) = databaseImages(:,i) - meanImg;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[U,S,V] = svd(db,'econ');

A=(S>0);

sumTotalSigmaValues = sum(S(:));

A=zeros(size(S,1) , size(S,1));

count =1;

A(:,count) = S(:,count);

sumCurrSigmaValues = sum(A(:));

while (sumCurrSigmaValues/sumTotalSigmaValues) \* 100 < 95

count = count+1;

A(:,count) = S(:,count);

sumCurrSigmaValues = sum(A(:));

end

count=count

W=U(:,1:count);

W=W';

queryImagest=zeros(count,size(queryImages,2));

for i=1:size(queryImages,2)

queryImagest(:,i) = W\*queryImages(:,i);

end

databaseImagest=zeros(count,size(databaseImages,2));

for i=1:size(databaseImages,2);

databaseImagest(:,i) = W\*databaseImages(:,i);

end

result = zeros( size(queryImagest,2), size(databaseImagest,2) ); % query image distance from database image

for qImage=1:size(queryImagest,2)

q1 = queryImagest(:,qImage);

for dImage=1:size(databaseImagest,2)

d1 = databaseImagest(:,dImage);

distance = sum((q1 - d1).^ 2,1).^0.5;

result(qImage,dImage) = distance;

end

end

imageNo = 7; % query image used

% displaying query image

im=queryImages(:,imageNo);

im=reshape(im,rows,cols);

im=uint8(im);

figure;

imshow(im);

% displaying closest database image

minDistance = min(result(imageNo,:));

index = find(result(imageNo,:) == minDistance);

im = databaseImages(:,index);

im=reshape(im,rows,cols);

im=uint8(im);

figure;

imshow(im);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% finding accuracy %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

accuracy =0;

for imageNo = 7:size(queryImages,2)

minDistance = min(result(imageNo,:));

index = find(result(imageNo,:) == minDistance );

if testLabel(1,imageNo ) == trainLabel(1,index)

accuracy = accuracy + 1;

end

end

accuracy = accuracy ./ size(testMat,1);

accuracy = accuracy\* 100;

disp('The accuracy in percentage is: ');

accuracy

**Output:**

**Graphical user interface, application

Description automatically generated**Graphical user interface, application

Description automatically generated