CS 215: Data Analysis and Interpretation

Assignment-2

Name: BVSS Prabandh & V Mahanth Naidu

Roll_no: 210050037 & 210050161

October 10,2022

Contents

1 Problem 1: Sampling within a Euclidean Plane

1.1 Sampling uniformly inside an Ellipse

An ellipse, within a 2D Euclidean plane, with center at the origin, and with major and minor axes of lengths 2 and 1 along the cardinal axes.
Algorithm for generating random points (in 2D) distributed uniformly inside the ellipse.

Set S are the points inside an ellipse

$$S = \{(x, y)| \frac{x^2}{4} + y^2 < 1\} \tag{1}$$

Parameterized:

$x = 2r\cos(\theta)$

$y = r\sin(\theta)$

$0 \leq r \leq 1 \quad 0 \leq \theta \leq 2\pi$

When we say that points should be ”uniformly distributed,” we mean that the probability of generating a point in any finite region is proportional to the area of that region.

```
rng(0);
clearvars;
x = zeros(5000,1);
y = zeros(5000,1);
% not exactly radius but a distance parameter
radius = 1;
% max value of parameter
twopi = 2 * pi;
for i=1:5000
   theta = twopi * rand();
   r = radius * sqrt(rand());
% radius proportional to sqrt(U), U¬U(0,1)
   x(i) = 2*r*cos(theta);
   y(i) = r*sin(theta);
end
scatter(x,y);
```

As area element is proportional to r^2 . radius r must follow sqrt of uniform distribution. While θ is uniformly distributed.

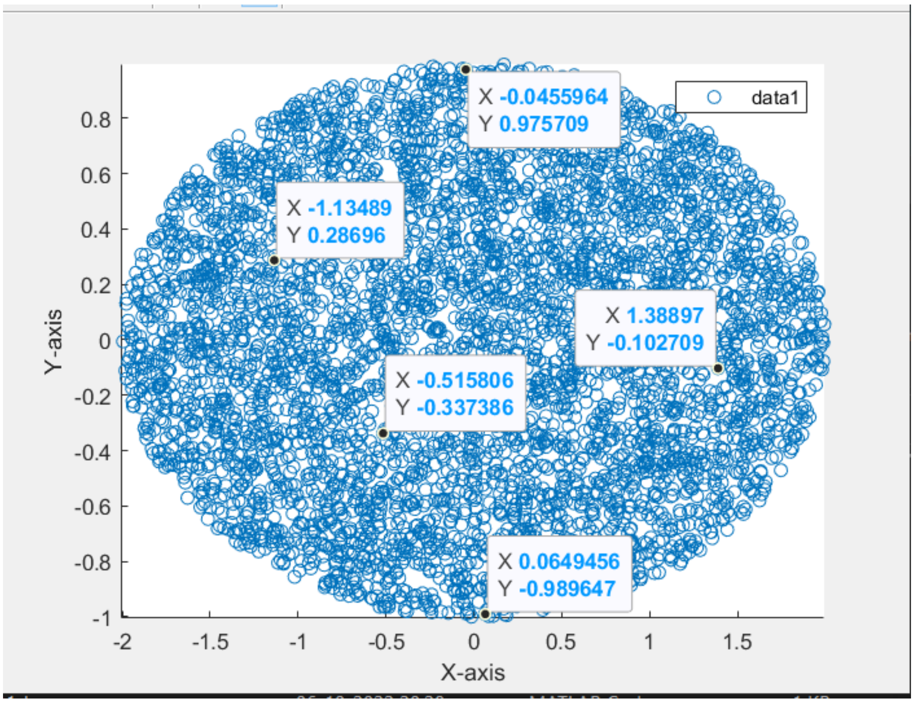


Figure 1: Uniformly distributed ellipse

1.2 histogram of uniformly distributed ellipse

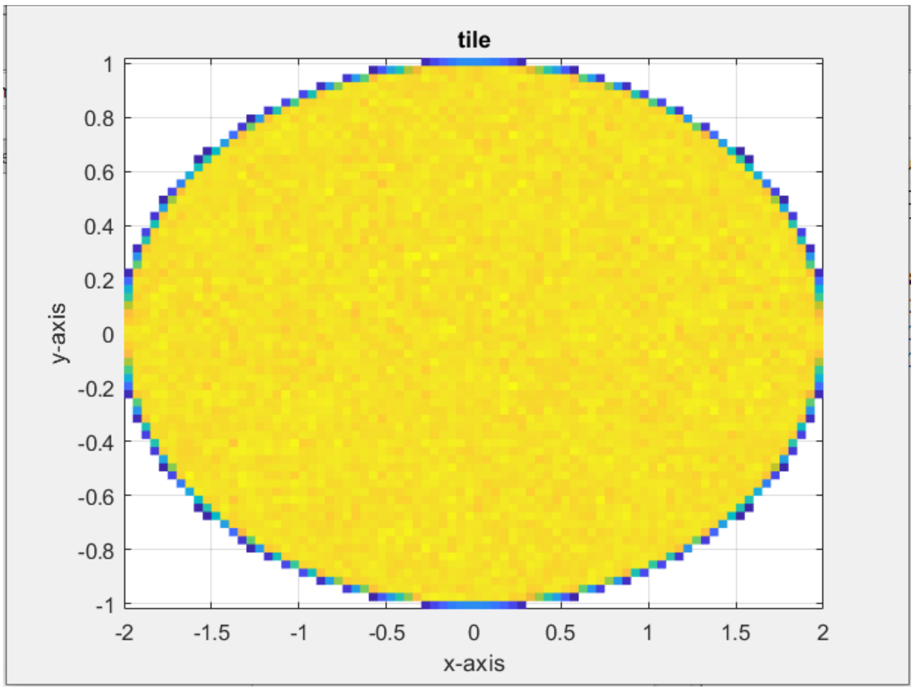


Figure 2: 2D histogram of a uniformly distributed ellipse

1.3 Sampling uniformly inside an Triangle

ALGORITHM

- Define the vectors **a = P2 - P1** and **b = P3 - P1**. The vectors define the sides of the triangle when it is translated to the origin.
- Generate random uniform values u1, u2 U(0,1)
- $Ifu1 + u2 > 1$, apply the transformation $u1 = 1 - u1$ and $u2 = 1 - u2$.
- Form w = u1a + u2b, which is a random point in the triangle at the origin.
- The point w + P1 is a random point in the original triangle.

```
rng(0);
clearvars;
p1=[0,0];
p2=[pi,0];
p3=[pi/3,exp(1)];
x = zeros(5000,1);
y = zeros(5000,1);
a = p2-p1;
b = p3-p1;
for i=1:5000
    u1=rand();
    u2 =rand();
    if(u1+u2>1)
        u1=1-u1;
        u2=1-u2;
    end
    w = p1+a*u1+b*u2;
    x(i)=w(1);
    y(i)=w(2);
end
scatter(x,y);
```

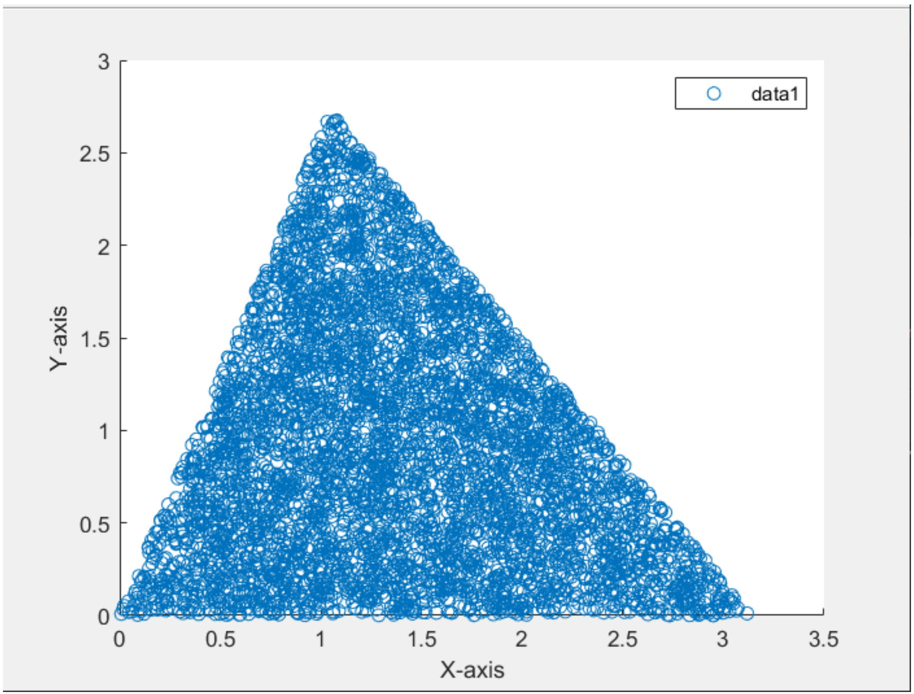


Figure 3: a uniformly distributed triangle

1.4 3D histogram of uniformly distributed ellipse

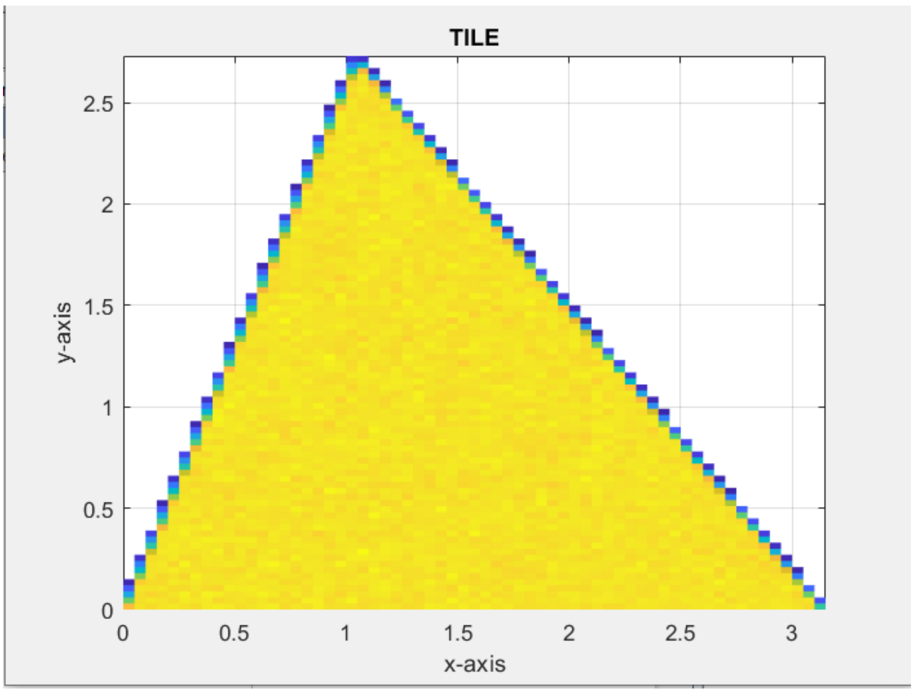


Figure 4: 2D histogram of a uniformly distributed triangle

2 Problem 2: Multivariate Gaussian

2.1 Algorithm to generate Multivariate Gaussian samples

As mentioned in the Question we are only allowed to use randn() and eig() as internal function to generate the data

$$X = AW + \mu$$

$$\mu = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.6250 & -1.9486 \\ -1.9486 & 3.8750 \end{bmatrix}$$

As C is a symmetric Matrix , We know from Spectral Theorem its diagonalizable. so

$$C = VDV^T$$

$$C = AA^T$$

$$AA^T = VDV^T$$

$$AA^T = VD^{\frac{1}{2}}D^{\frac{1}{2}}V^T$$

We can see that $A = VD^{\frac{1}{2}}$ is a solution.
We can geneate W matrix using randn() for x and y as elements of W are i.i.ds
Hence Data is generated

```
rng(0);
clearvars;
N=[10,10^2,10^3,10^4,10^5];
for j=1:5
    X=zeros(N(j),2);
    % taking N samples
    for i=1:N(j)
        W=[randn();randn()];
        % generating two iids as x and y
        C=[1.6250,-1.9486;-1.9486,3.8750;];
        mu=[1;2];
        [V,D] = eig(C);
        % eigen decomposition of C
        A=V*sqrtm(D);
        % A is found
        x=A*W+mu;
        % samples are created
        X(i,1)=x(1);
        X(i,2)=x(2);
    end
    MLE_mean = sum(X)/(N(j));
```

```
    disp(MLE_mean);
    MLE_covariance=zeros(2,2);
    x = X(:,1);
    y = X(:,2);
    sample_mean_X = MLE_mean(1);
    sample_mean_Y = MLE_mean(2);
    MLE_covariance(1,1)=sum((x-sample_mean_X).^2)/(N(j));
    MLE_covariance(2,2)=sum((y-sample_mean_Y).^2)/(N(j));
    Z=((x-sample_mean_X).*(y-sample_mean_Y));
    MLE_covariance(1,2)=sum(Z)/(N(j));
    MLE_covariance(2,1)=MLE_covariance(1,2);
    disp(MLE_covariance);
end
```

Below are the MLE of Mean and covariance matrices of generated data.
(mean as row matrix)
(covariance as 2X2 matrix)

```
-0.3284    3.8600

 3.3055   -3.3554
-3.3554    6.1815
```

Figure 5: N=10

```
 1.0656    1.9989

 1.4690   -1.8433
-1.8433    3.9425
```

Figure 6: N=10^2

```
 1.0656    1.9989

 1.4690   -1.8433
-1.8433    3.9425
```

Figure 7: N=10^3

```
 0.9957    2.0136

 1.6301   -1.9237
-1.9237    3.7641
```

Figure 8: N=10^4

```
 0.9998    2.0025

 1.6140   -1.9369
-1.9369    3.8755
```

Figure 9: N=10^5

2.2 a boxplot of the error between the true mean μ and the ML estimate μ_n

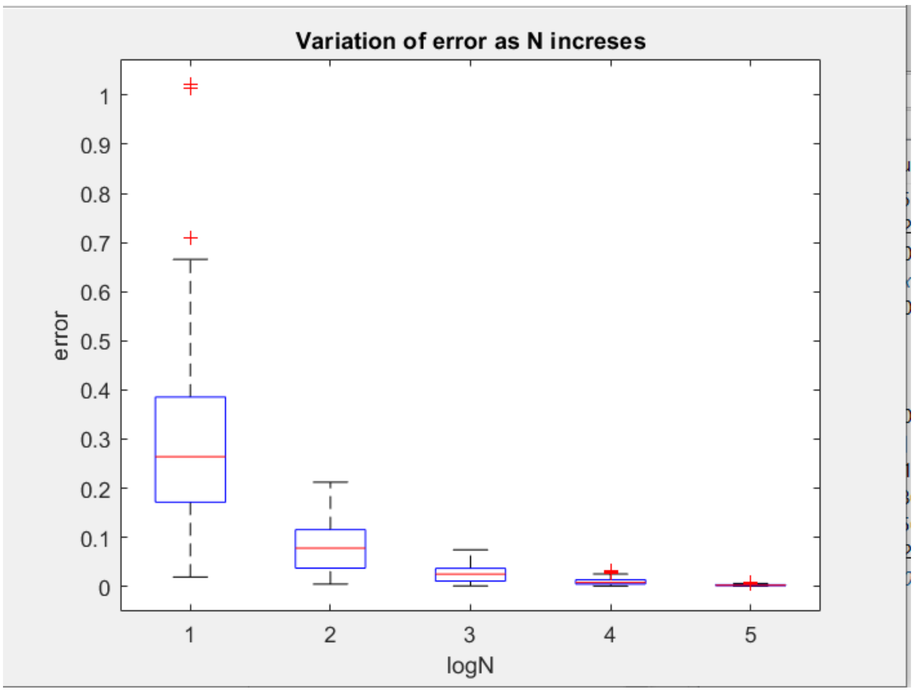


Figure 10: mean MLE error

2.3 a boxplot of the error between the true mean C and the ML estimate C_n

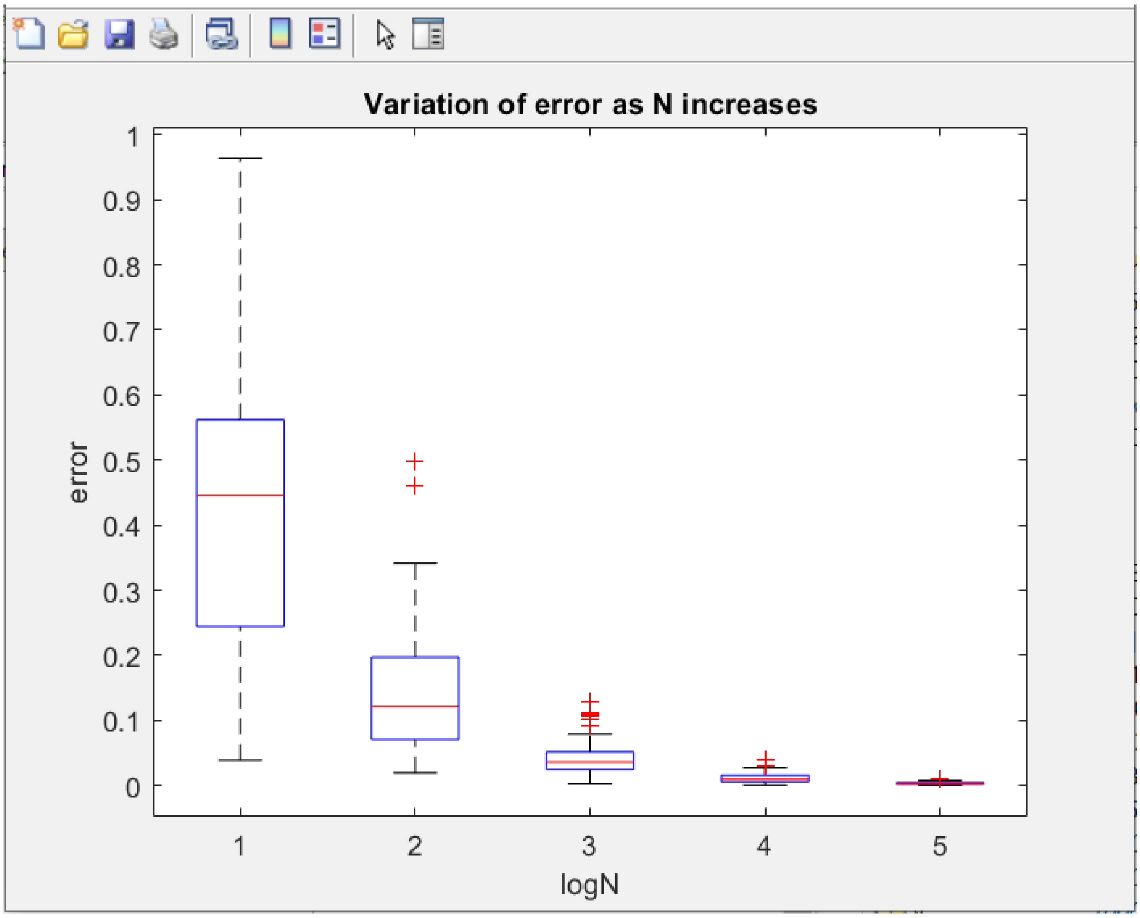


Figure 11: covariance MLE error

2.4 Scatter plots and principle mode of variance

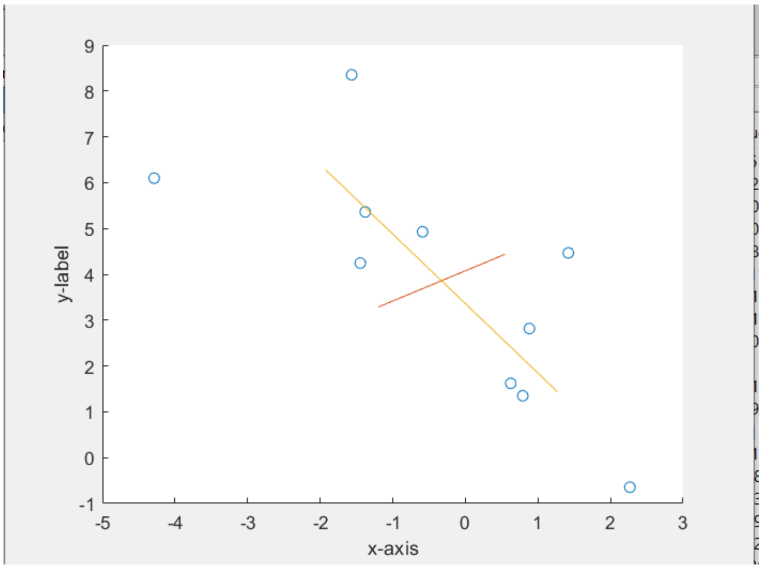


Figure 12: N=10

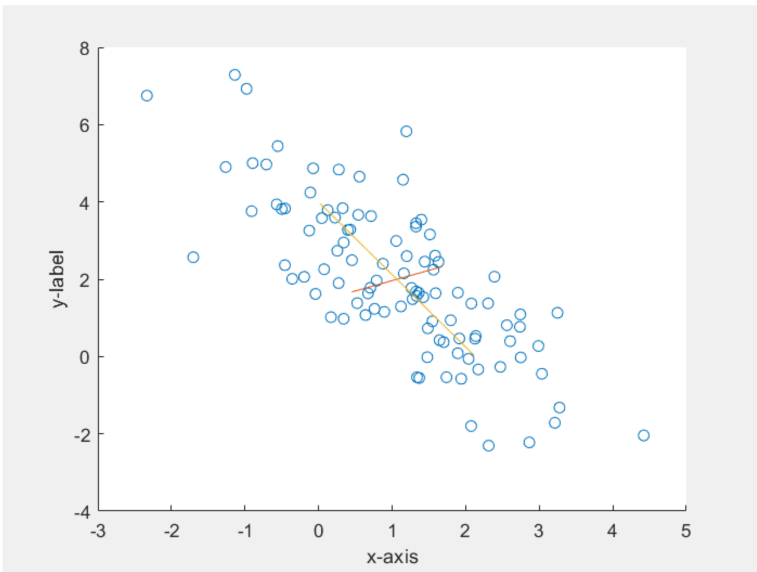


Figure 13: N=10^2

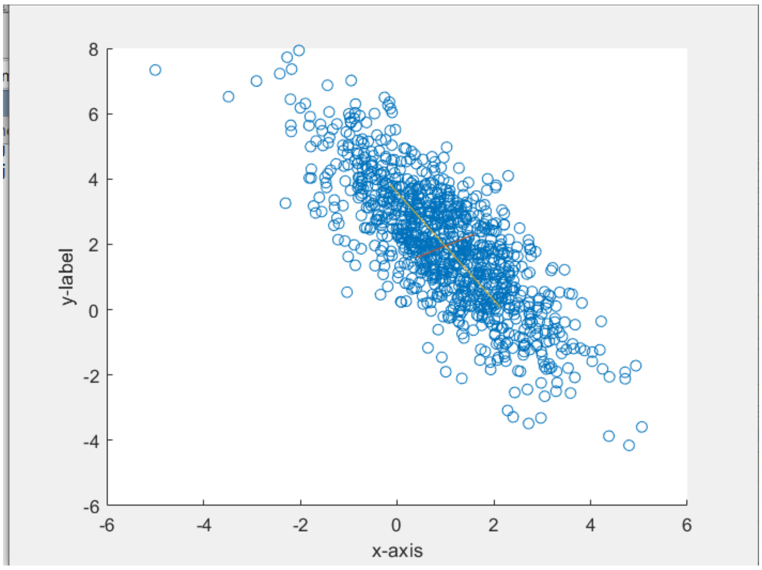


Figure 14: N=10^3

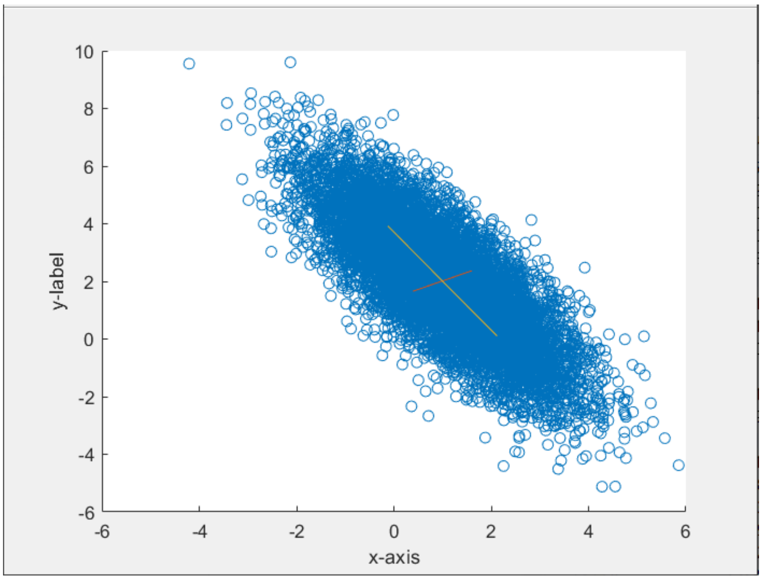


Figure 15: N=10^4

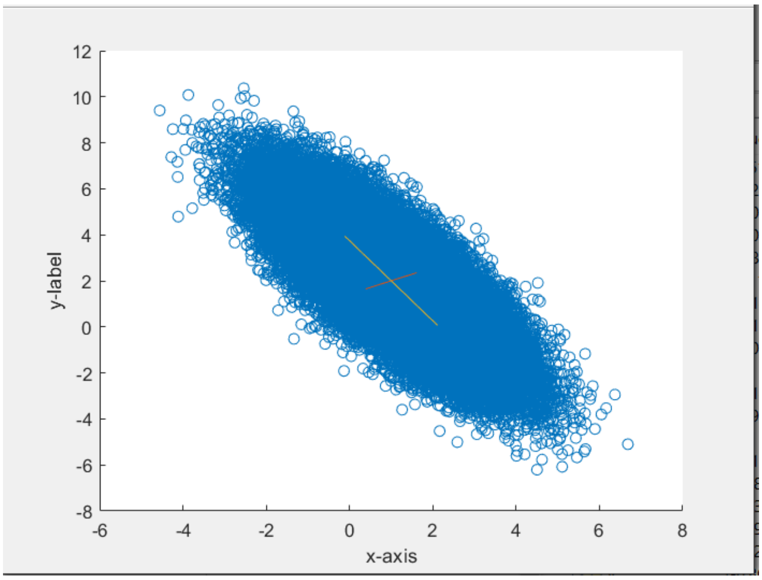


Figure 16: N=10^5

3 Problem 3: PCA and Hyperplane Fitting

3.1 Proof that First mode of Variance is the best fit Line

Below shown is the scatter plot of ”points2D_set1”

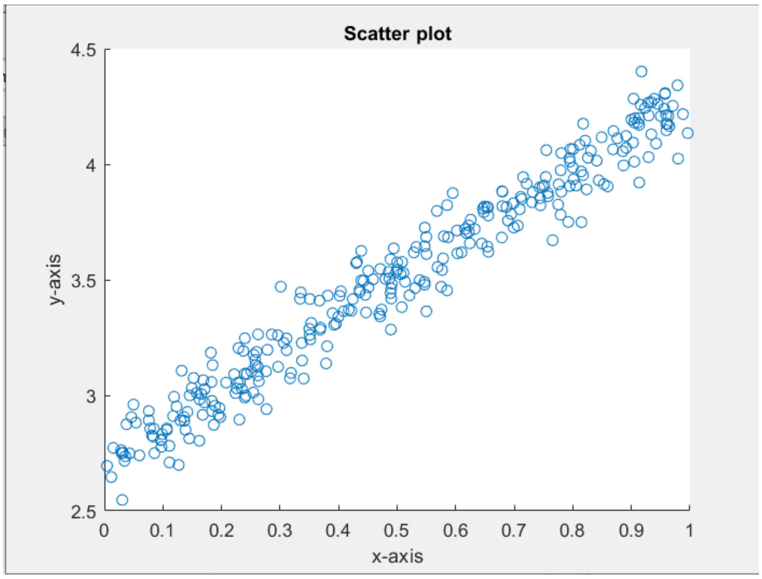


Figure 17: Scatterplot of x and y

I am going to prove that first mode of variance component is the best fit line. We want line which represents best linearity between x and y.

We know line passes through (μ_x, μ_y) Let the best fit line have slope β and intercept α and every points is expressed as

$$Y_i = \beta X_i + \alpha + \eta_i$$

$$\mu_y = \beta\mu_x + \alpha$$

First mode of variation(PCA) also passes through $(\mu_x$,$\mu_y)$

Best Line would be the line which has most of the points and other points are closer to line Mathematically: Sum of square of perpendiculars drawn from points to the line are minimum

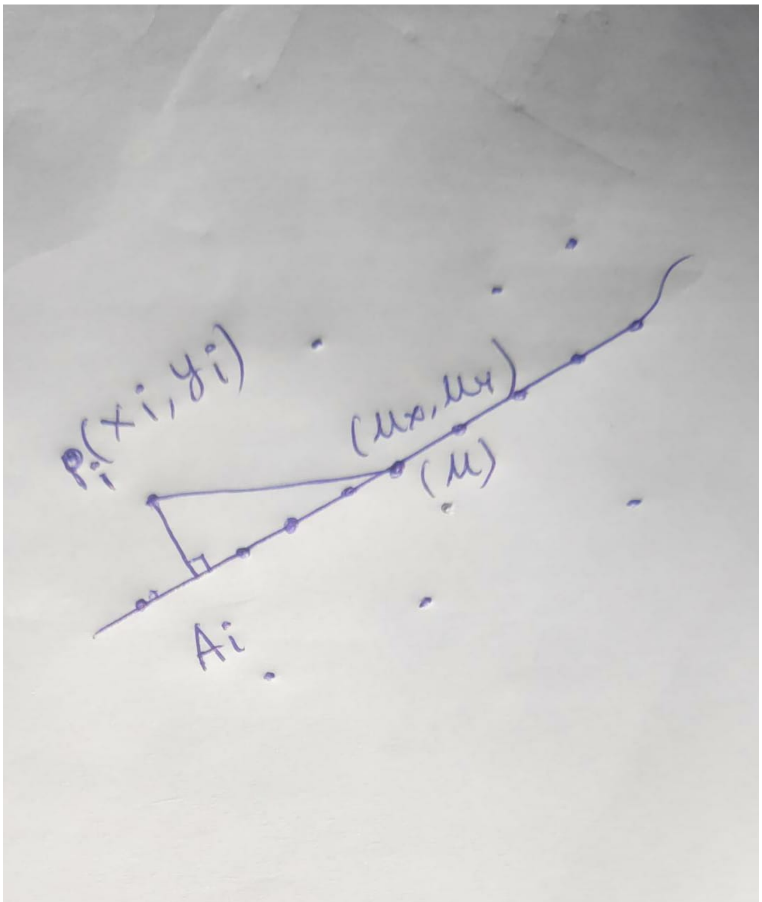


Figure 18: Consider a point in the plane

from Right Triangle $A_i P_i \mu$

$$A_i P_i^2 + A_i \mu^2 = P_i \mu^2$$

Add all these equation from all points

$$\sum_{i=1}^{n} A_i P_i^2 + \sum_{i=1}^{n} A_i \mu^2 = \sum_{i=1}^{n} P_i \mu^2$$

As all lines pass through (μ_x,μ_y) We can say distance $P_i \mu$ is independent of line slope
So is the Summation on RHS

We can see $\sum_{i=1}^{n} A_i P_i^2$ is Sum of square of perpendiculars drawn from points to the line i.e, The quantity we want to minimize.
Now as RHS is constant(Independent of slope of line)

We need $\sum_{i=1}^{n} A_i \mu^2$ to be **MAXIMUM** for $\sum_{i=1}^{n} A_i P_i^2$ to be **MINIMUM**

$$\sum_{i=1}^{n} A_i \mu^2 = Variance\ of\ Projected\ Data\ along\ the\ line \times n$$

i.e, =Variance of Projected Data along the line must be Maximum. which is the first mode of variance (PCA)

3.2 Scatter plot and Best Fit Line For linear data

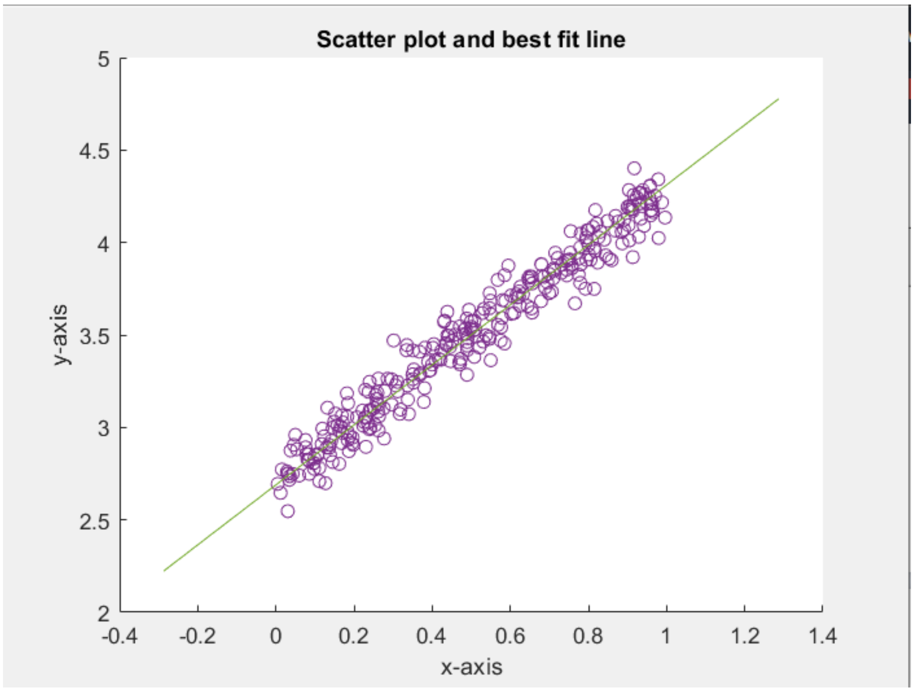


Figure 19: Scatter plot and best fit line for Linear Data

3.3 Scatter plot and Best Fit Line for Quadratic Data

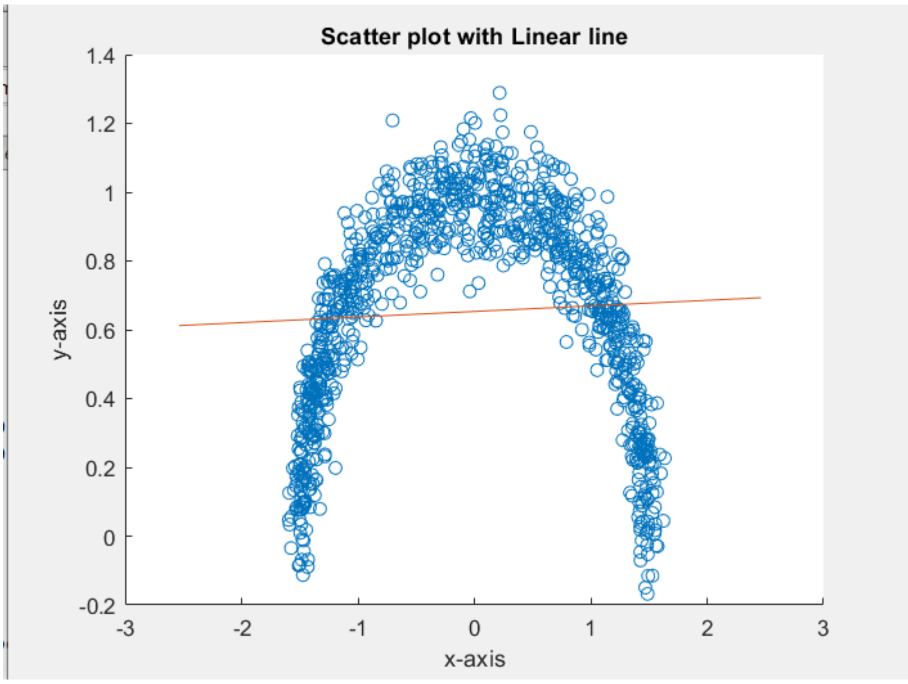


Figure 20: Scatter plot and best fit line for Quadratic Data

We can see Obviously that for ”Points2D_set2” , first mode of variance isnt the best Line,

- Reason being Data is Quadratic rather than Linear.
- PCA gives u number of significant basis required to repersent the Data in Euclidean Space.
- Below are the Eigen values of covariance Matrices.

$$C_{linear} = \begin{bmatrix} 0.0025 & 0 \\ 0 & 0.2925 \end{bmatrix}$$

$$C_{Quadratic} = \begin{bmatrix} 0.096 & 0 \\ 0 & 1.1037 \end{bmatrix}$$

- We can see variance along second mode of variance(eigenvalue) is far less compared to first mode of variance in case of Linear Data , While being comparable for Quadratic Data.
- This indicate Significant PCAs for **Set1** is just first mode of variance

- While we need two modes to Represent **Set2**
- But its noteworthy to Observe, Although PCA line doesn't represent Linearity Between x and y , But Points are distributed almost Symmetrically around the line.

4 Problem 4: Principal Component Analysis (PCA)

4.1 Mean μ ,Covariance Matrix C and Principle mode of variation

```
rng(0);
clearvars;
load("mnist.mat")
axis equal;
digits_train = cast(digits_train,"double");
mean_matrix = zeros(28^2,10);
covariance_Matrix = zeros(28^2,28^2,10);
number_matrix = zeros(10,1);
eigen_mat=zeros(784,784,10);
PCA_matrix = zeros(784,10);
max_eigen = zeros(10,1);
for i=1:60000
    B = reshape(digits_train(:,:,i),[],1);
    mean_matrix(:,labels_train(i)+1)=mean_matrix(:,labels_train(i)+1)+B;
    number_matrix(labels_train(i)+1)= ...
        number_matrix(labels_train(i)+1)+1;
end
for i=1:10
    mean_matrix(:,i)=mean_matrix(:,i)/(number_matrix(i));
    C = reshape(mean_matrix(:,i),[],28);
end
% Creating a Covariance Matrix
for i=1:60000
     B = reshape(digits_train(:,:,i),[],1);
     B=B-mean_matrix(:,labels_train(i)+1);
     covariance_Matrix(:,:,labels_train(i)+1)= ...
         covariance_Matrix(:,:,labels_train(i)+1)+B*B';
end
%%
for i=1:10
    covariance_Matrix(:,:,i)=covariance_Matrix(:,:,i)/(number_matrix(i));
    e = eig(covariance_Matrix(:,:,i));
    [V,D]=eig(covariance_Matrix(:,:,i));
    max_eigen(i)=e(784);
    eigen_mat(:,:,i)=V;
    %scatter(e*0+i-1,e)
    %hold on;
    PCA_matrix(:,i)=V(:,784);
end
```

- (i) Above is the Implementation of Algorithm to generate Mean,Covariance Matrix and Principal mode of Variation
- (ii) mean_matrix(: , i+1) gives μ of digit "i"
- (iii) covariance_matrix (: , : , i+1) gives C of digit "i"
- max_eigen(i) and PCA_matrix(: , i) give max eigen value and eigen vector of digit "i" respectively

4.2 How many "principle" / significant modes of variation do you find, for each digit ?

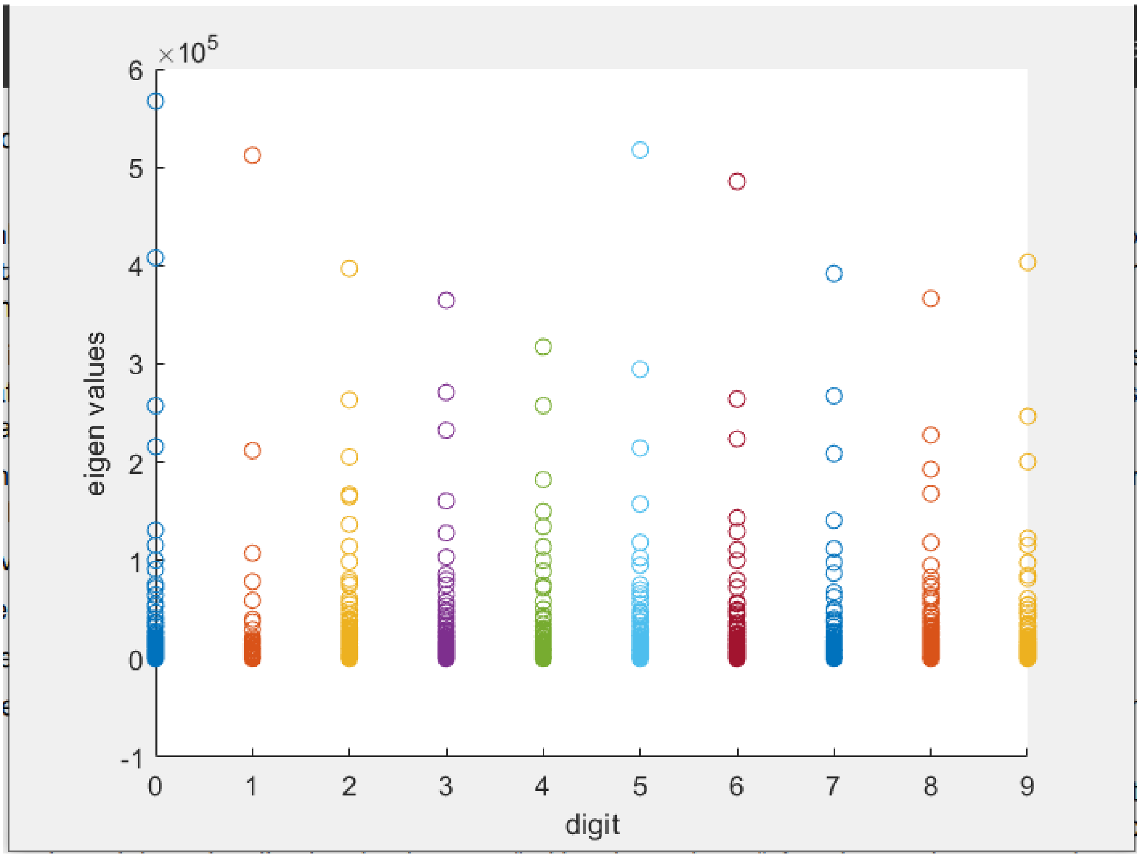


Figure 21: Variation of eigen values with N

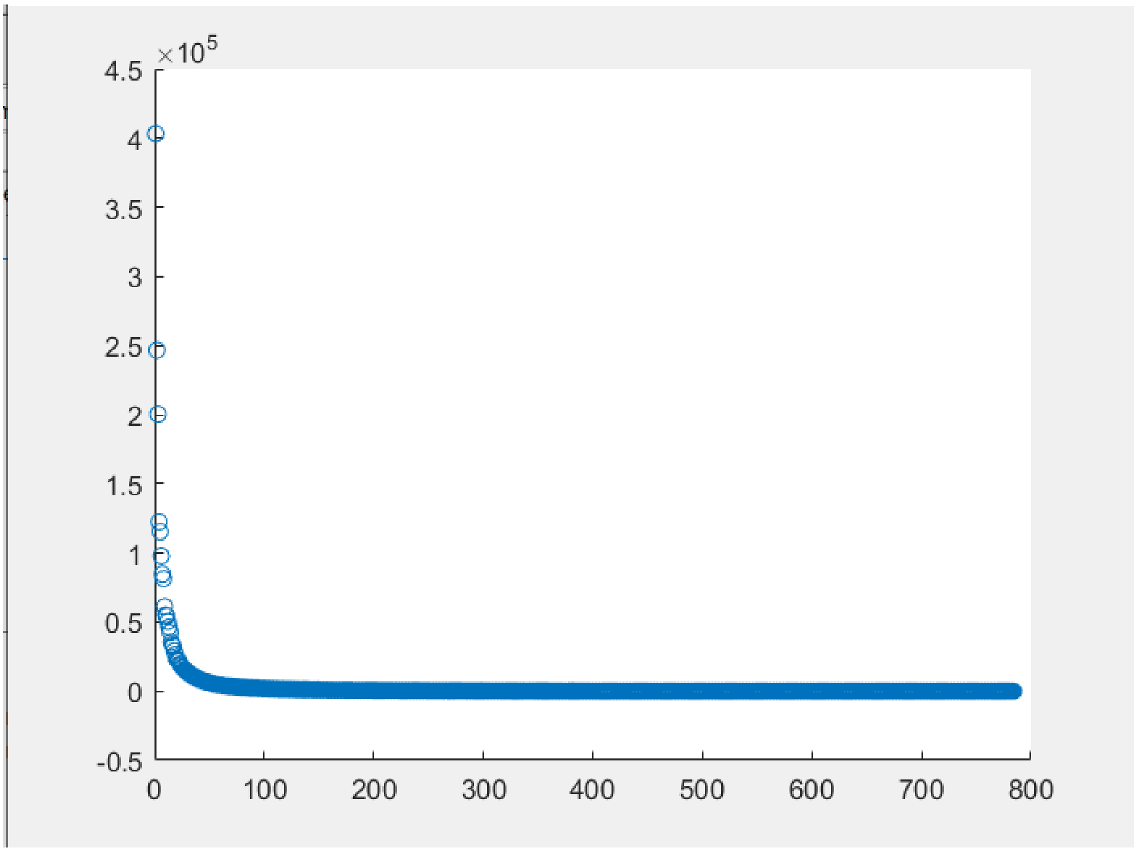


Figure 22: eigen value vs rank for N=9

- As we can see most from the graphs(data) PCA of data is far less than 748
- number of eigen values greater than 1% of maximum eigen value are lessthan 100
- Below are the number of modes having 90% of total data

digit	**siginificant eigen values**
0	63
1	38
2	79
3	80
4	74
5	74
6	62
7	66
8	81
9	62

- This indicates that most of the Data variation are along Specific lines , which is completely understandable because ”most of the people have same way of writing the digit rather each person having his unique style”

4.3 PCA for the digits 0 to 9 - Results of Modes of Variations

- For each digit plotting an image of $\mu^i + \sqrt{\lambda_1^i}v_1^i\ \mu^i\ \mu^i - \sqrt{\lambda_1^i}v_1^i$ after reshaping to 28×28 matrix.
- In results left side image correspond to $\mu^i + \sqrt{\lambda_1^i}v_1^i$ middle image correspond to μ^i and right side image correspond to that of $\mu^i + \sqrt{\lambda_1^i}v_1^i$, So the images of digits are as shown Here μ^i corresponds to mean of the digit i and λ_1^i is the maximum eigen value among all the eigen values and v_1^i is the corresponding eigen vector of λ_1^i

Figure 23: PCA for digit 0

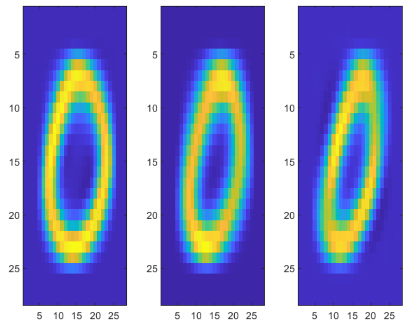


Figure 24: PCA for digit 1

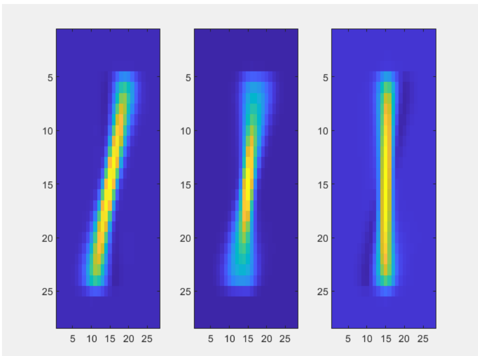


Figure 25: PCA for digit 2

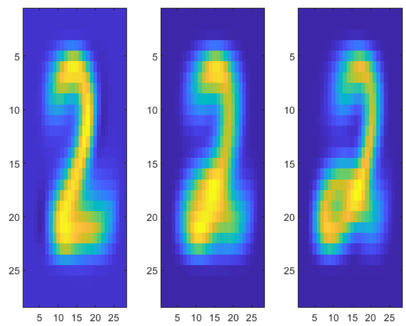


Figure 26: PCA for digit 3

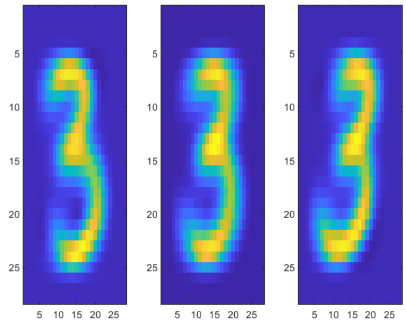


Figure 27: PCA for digit 4

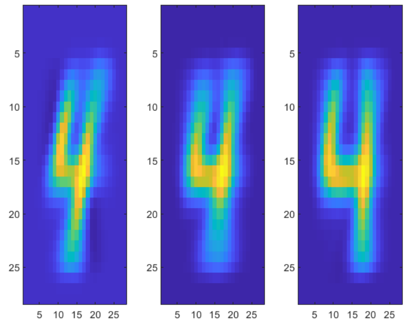


Figure 28: PCA for digit 5

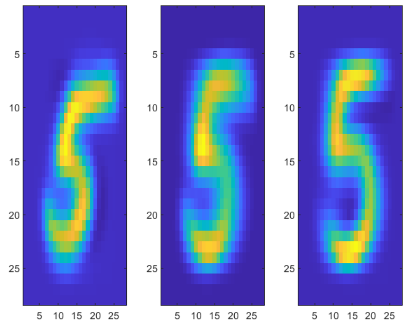


Figure 29: PCA for digit 6

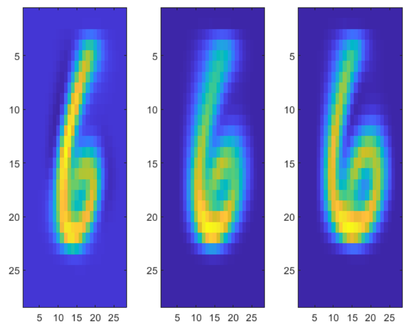


Figure 30: PCA for digit 7

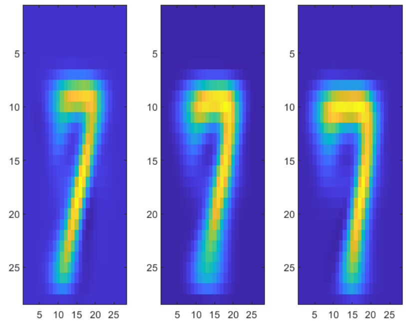


Figure 31: PCA for digit 8

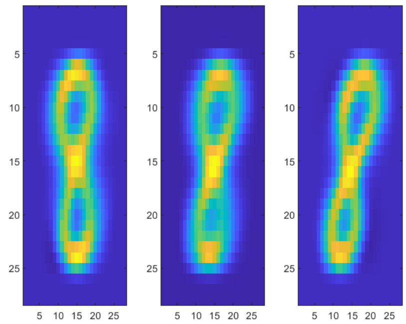
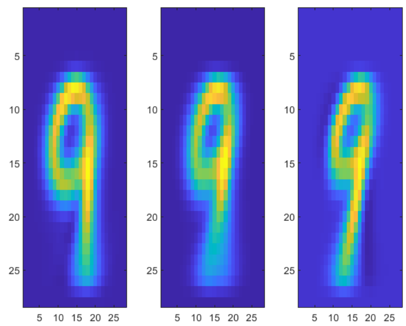


Figure 32: PCA for digit 10



- So if we observe for digit 1 mean image which is in the middle and other two on both sides of the mean image, are the images corresponding to $\mu^1 + \sqrt{\lambda_1^1} v_1^1$, $\mu^1 - \sqrt{\lambda_1^1} v_1^1$ respectively, where μ^1 is the mean of digit 1 from the given set $\sqrt{\lambda_1^1}$ is the maximum eigen value of the covariance matrix corresponding to digit 1 and v_1^1 is it's eigen vector .

- We observed that the images present on either sides are slightly inclined to the mean image,and the image left to the mean image that is the image corresponding to $\mu^1 + \sqrt{\lambda_1^1} v_1^1$ of digit 1 from the given data set is the way how people write the digit 1.

5 Problem 5: Principal Component Analysis (PCA) for Dimensionality Reduction

5.1 Projecting on hyper-Plane formed by first 84-eigen vectors

```
function projection = function_dimension(basis,X)
Y=basis'*X;
projection=Y(701:784);
end
```

- Consider the Euclidean space in Which these 784-dimension points are present,
- Now we want take projection of these points on the 84-dimension hyper plane.
- Take a point X in the space which is represented by basis $e_1, e_2, ...e_{784}$
- eigen vectors for covariance matrix are orthogonal so 784 of them also represent this Euclidean space $w_1, w_2, ...w_{784}$

$$X_{old} = AX_{new} \tag{2}$$

- Where A is the matrix having columns as **new basis** in **old basis system**.
- Upon rotating we can Project the point by just taking the coordinates which are along $w_1, w_2, ...w_{84}$.
- projection is an 84×1 vector in the new basis system.

5.2 Rerotaing the System

```
rng(0);
clearvars;
load("test.mat")
digits_train_modified = zeros(784,1,60000);
for i=1:60000
    B = reshape(digits_train(:,:,i),[],1);
    C=eigen_mat(:,:,labels_train(i)+1);
    B = C'*B;
    B(1:700)=0;
    B=C*B;
    digits_train_modified(:,:,i)=B;
end
```

- We have seen Before How to projrct on the hyperPlane
- essentially all coordinates along $w_{85}, w_{86}, ...w_{784}$ are Zeros
- make a 784×1 vector by adding 700 zeros to 84×1 vector in the new basis.
- Rotate it to normal basis $e_1, e_2, ...e_{784}$ by multiplying A^T this time

$$X_{old} = A^T X_{new} \tag{3}$$

- we get new vectors which are projections of old points on the 84-D hyperplane in old basis
- Below are the images are the comparision in images of new vectors and old vectors.

Figure 33: PCA for digit 0

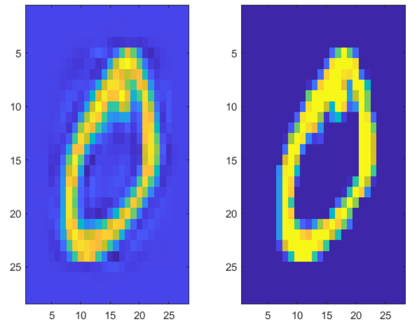


Figure 34: PCA for digit 1

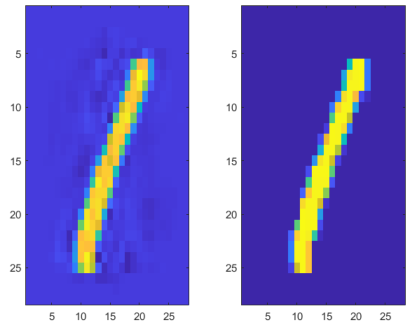


Figure 35: PCA for digit 2

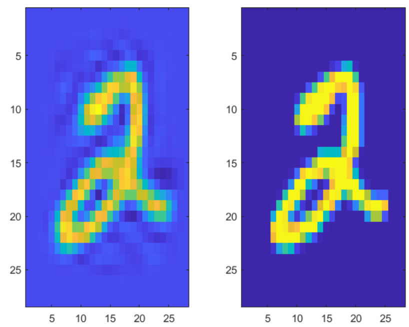


Figure 36: PCA for digit 3

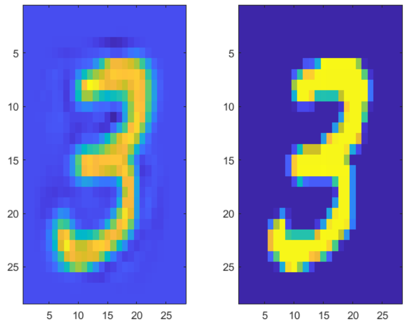


Figure 37: PCA for digit 4

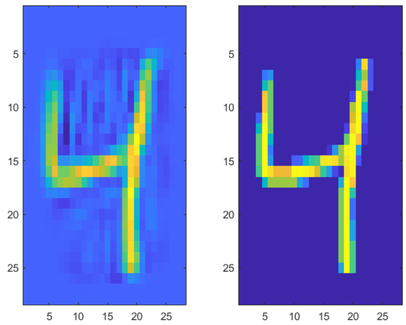


Figure 38: PCA for digit 5

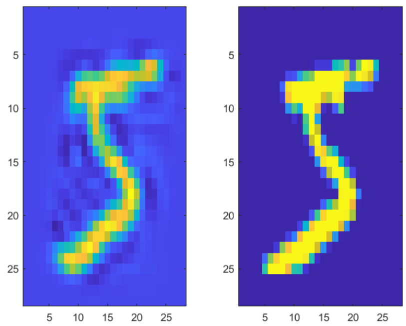


Figure 39: PCA for digit 6

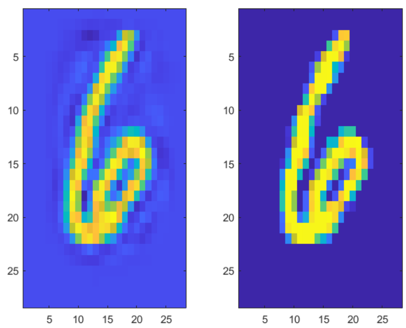


Figure 40: PCA for digit 7

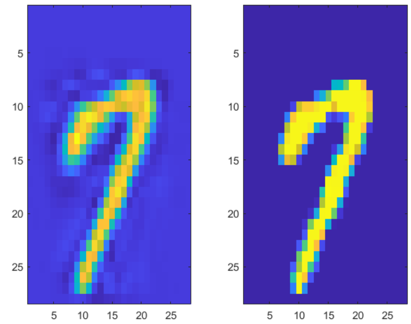


Figure 41: PCA for digit 8

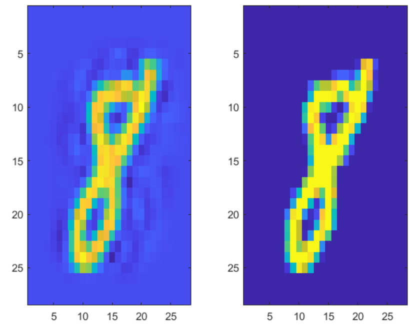
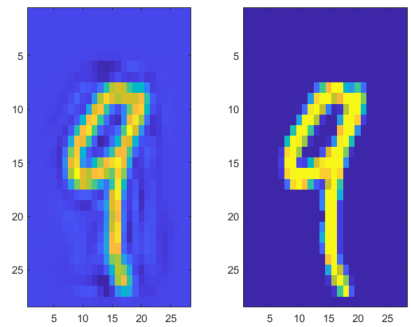


Figure 42: PCA for digit 10



6 Problem 6: Principal Component Analysis (PCA) for Another Image Dataset

Code is in the directory named `q6` inside `code` folder. It is named `q6.m` and can be run just by typing `q6.m` in the console.
The results are all saved in the directory named `q6` inside `results` folder. The results for part (a) are directly in this folder, for part (b) are inside the subdirectory named `fruitComparisons` and for part (c) are inside the subdirectory named `newFruits`.

6.1 Q6(a)

The results for this part are named `q6_meanEigenvectorImages.png` and `q6_eigenvalues.png`.

Figure 43: Mean and EigenVector Images

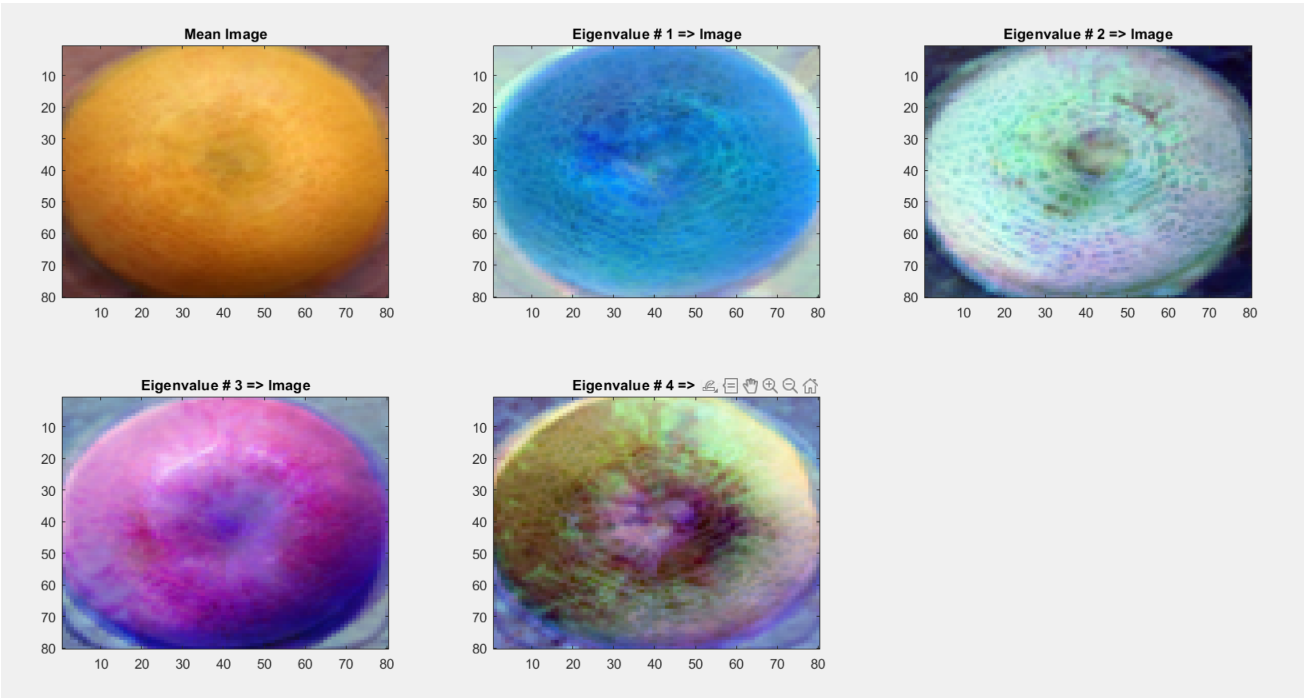
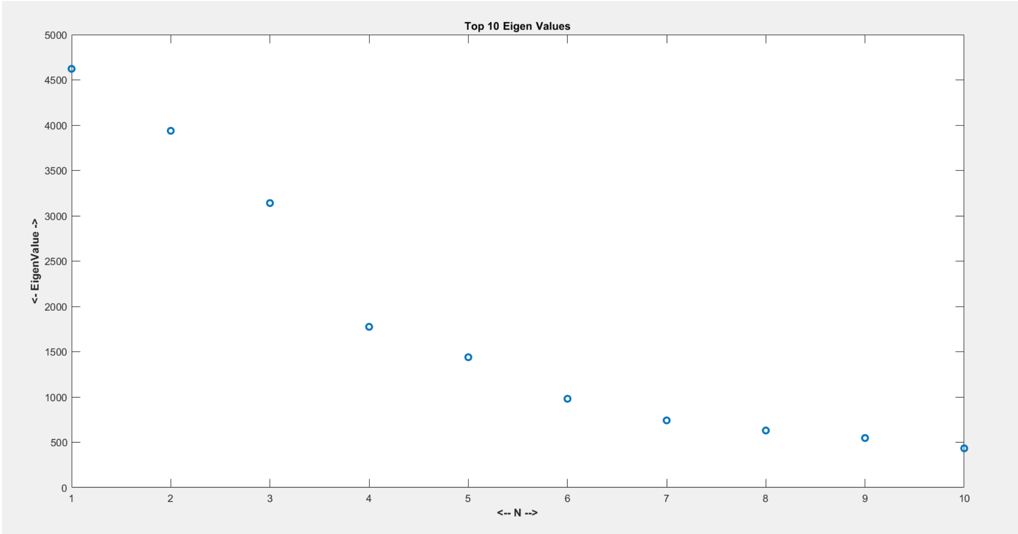


Figure 44: EigenValues



6.2 Q6(b)

Algorithm: Let the original image be I and the closest representation of I be the image J. J is written as a linear combination of the mean vector($\overline{u}$) and the 4 principle eigenvectors($\overline{v_1}$, $\overline{v_2}$, $\overline{v_3}$ and $\overline{v_4}$) of 16 different I's.

- We know that Frobenius norm for a matrix A is defined as

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}$$

- Now, let $J = a_1\overline{u} + a_2\overline{v_1} + a_3\overline{v_2} + a_4\overline{v_3} + a_5\overline{v_4}$.

- As $\overline{u}$ is in the same domain as that of $\overline{v_i}$, we can write
  $u = \sum u_i\overline{v_i}$
  $u_i = \overline{u}.\overline{v_i}$
  $J = \sum_i j_i\overline{v_i}$

- Frobenius norm of the difference between I and J is to be minimized. $\Delta = I - J$

- $$||\Delta||_F = \sum_{i=1}^{4}(j_i - u_i a_1 - a_{i+1})^2 + \sum_{i=5}^{19200}(j_1 - u_i a_1)^2$$

- To make $||\Delta||_F$ minimum, we can make first 4 elements 0.
  $a_2 = j_1 - u_1 a_1$
  $a_3 = j_2 - u_2 a_1$
  $a_4 = j_3 - u_3 a_1$
  $a_5 = j_4 - u_4 a_1$
- We must also differentiate remaining non-zero terms of the expression wrt a_1, and equate it to 0: $\sum (j_i - u_i a_1)(-u_i) = 0$
  On solving, $a_1 = \frac{\overline{J}.\overline{u} - \sum_{i=1}^{4} j_i u_i}{\overline{u}.\overline{u} - \sum_{i=1}^{4} u_i^2} = \frac{\sum_{i=5}^{19200} j_i u_i}{\sum_{i=1}^{5} u_i^2}$
- Hence, we get

$$J = a_1 \overline{u} + (j_1 - u_1 a_1)\overline{v_1} + (j_2 - u_2 a_1)\overline{v_2} + (j_3 - u_3 a_1)\overline{v_3} + (j_4 - u_4 a_1)\overline{v_4}$$

Referred to: Q6 ref 1
The results for this part are named `q6_fruit<i>.png` where i=1,2,..16.

Figure 45: Fruit 1

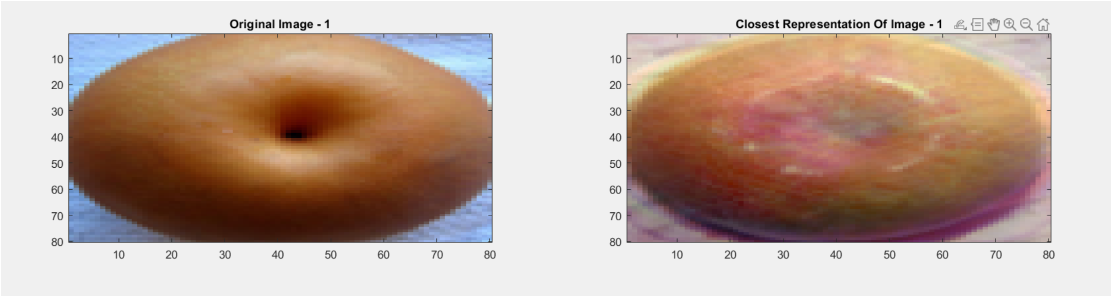


Figure 46: Fruit 2

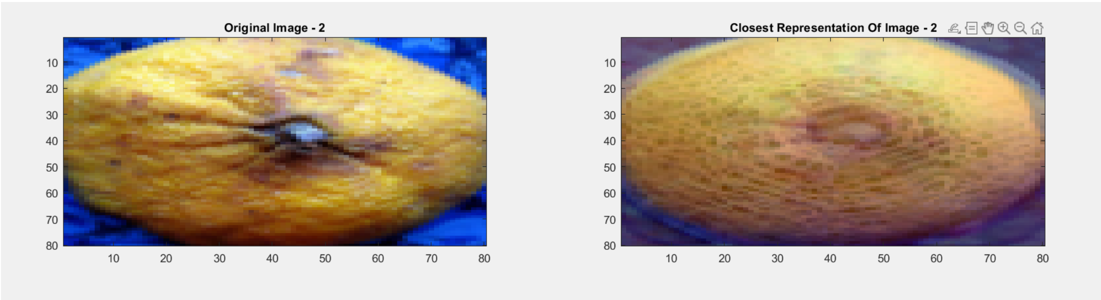


Figure 47: Fruit 3

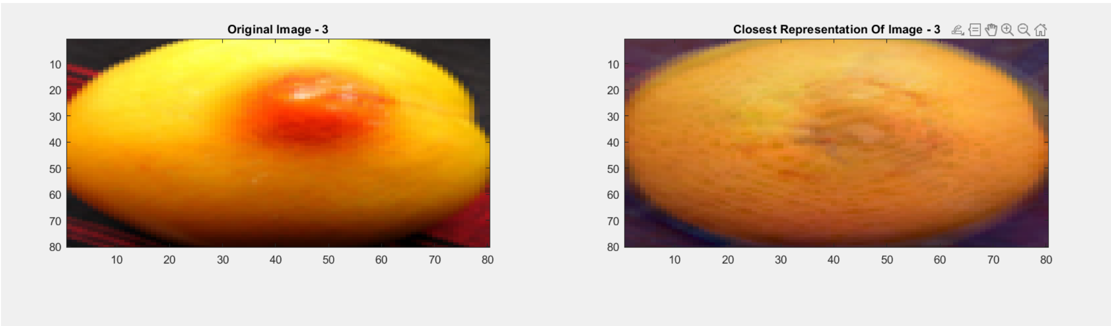


Figure 48: Fruit 4

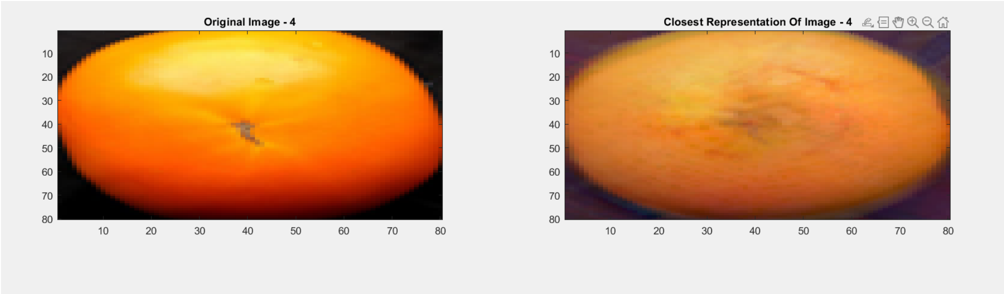


Figure 49: Fruit 5

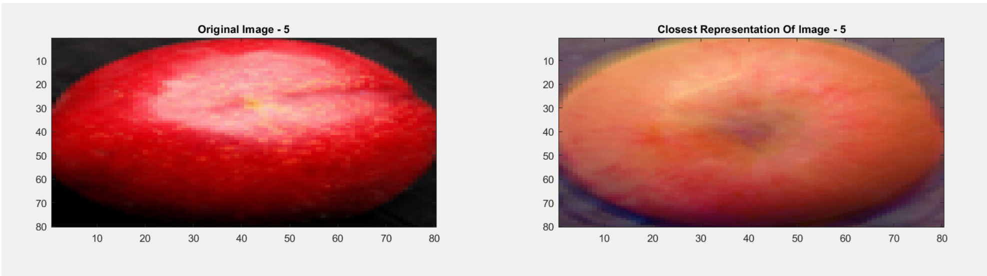


Figure 50: Fruit 6

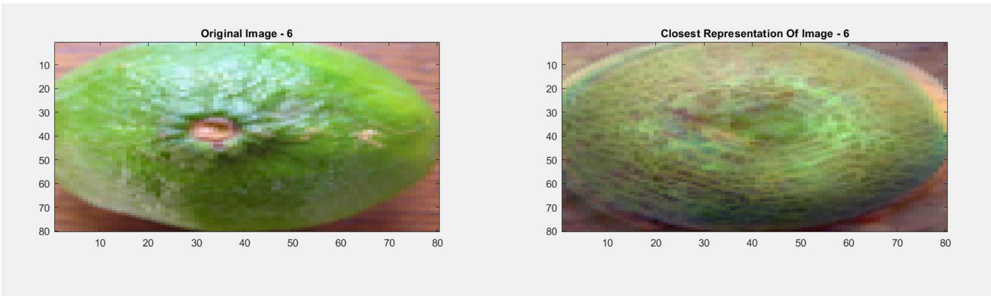


Figure 51: Fruit 7

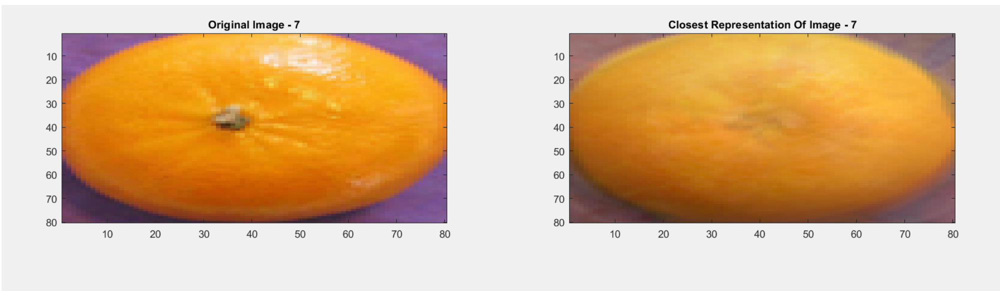


Figure 52: Fruit 8

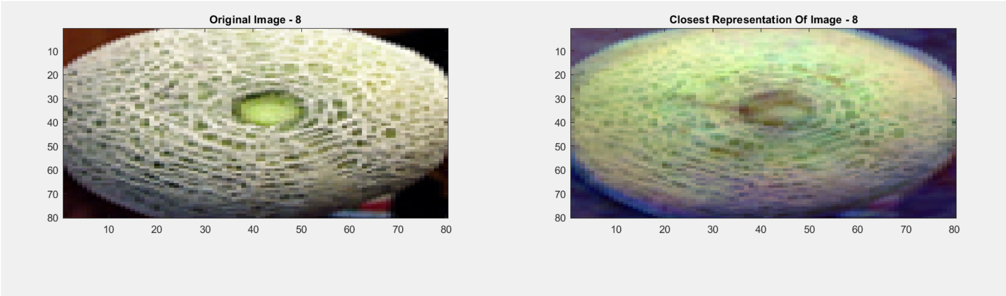


Figure 53: Fruit 9

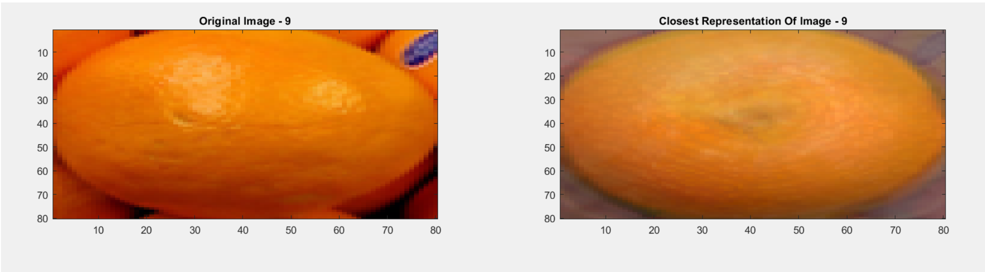


Figure 54: Fruit 10

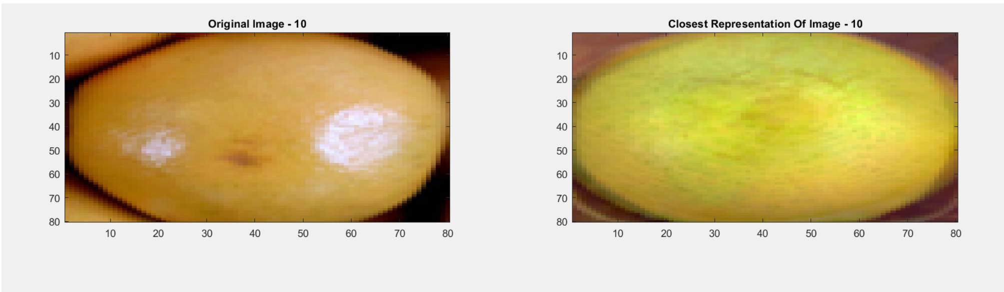


Figure 55: Fruit 11

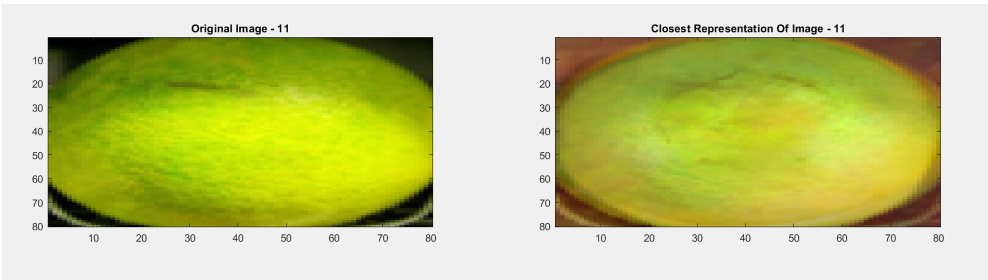


Figure 56: Fruit 12

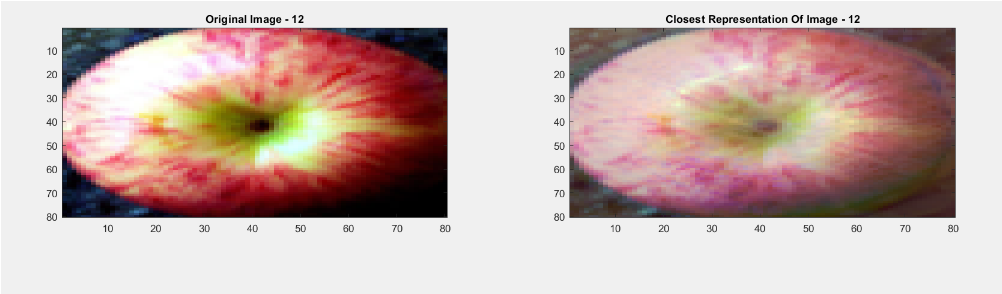


Figure 57: Fruit 13

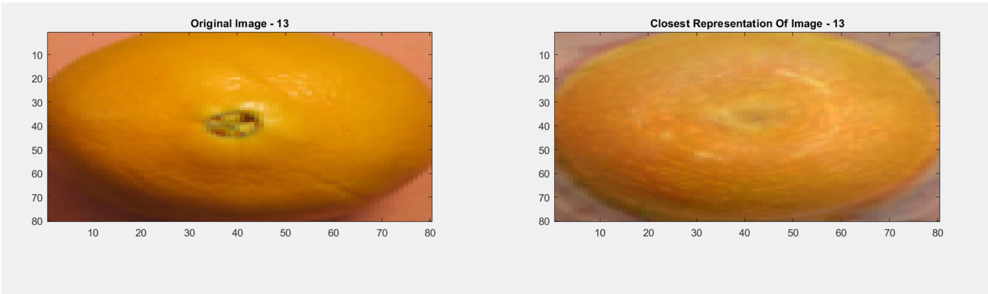


Figure 58: Fruit 14

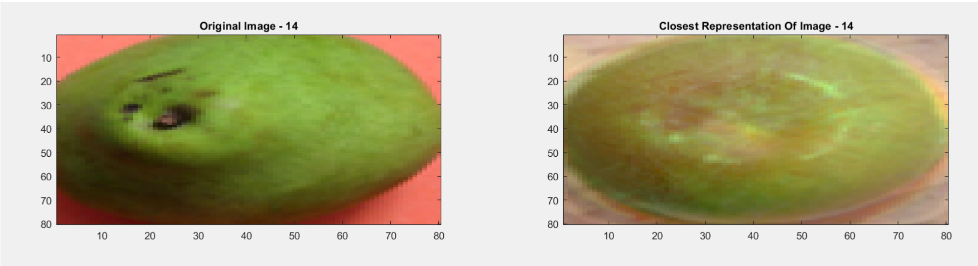


Figure 59: Fruit 15

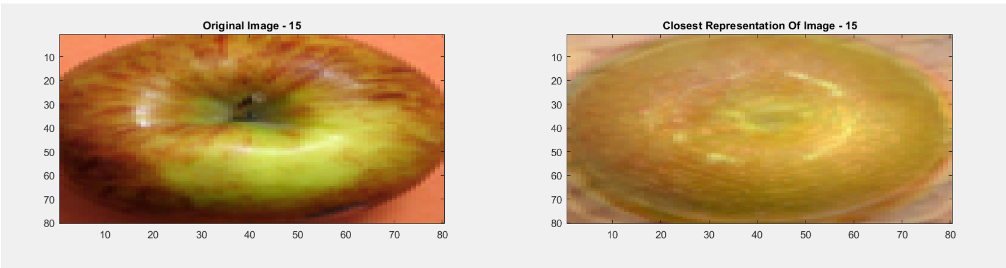
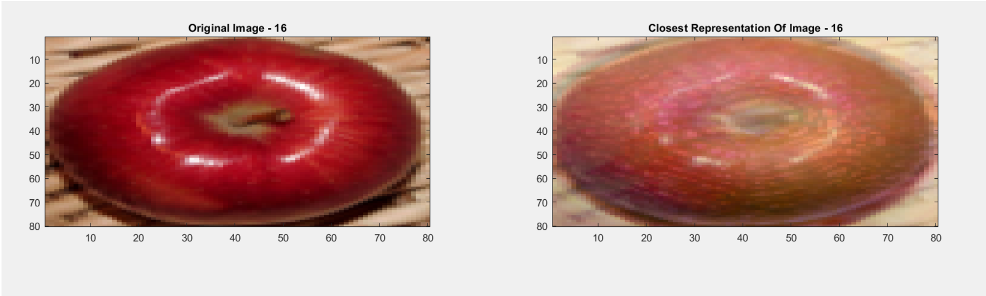


Figure 60: Fruit 16



6.3 Q6(c)

Algorithm: My aim is to get 3 images from the Multi Variate Gaussian to generate new fruits.

- I'll write the new image $J = mean + \sum_{i=1}^{4} \alpha_i V_i$, where $mean$ is the 19200×1 mean vector and $V_i, i = 1, 2, 3, 4$ are the 4 principle eigenvectors.
- To find α_i:
  - Among the 16 image data, consider one. After reshaping it to a column vector, let, vector is I.
  - To standardize this vector I, subtract the mean vector from it, let, vector is M.
  - We can find α_i by then taking dot product of the vector M with corresponding V_i.
- Thus, we have a 19200×1 vector J. We reshape it to $80 \times 80 \times 3$ matrix and display this image.

Referred to: Q6 ref 2
The results for this part are named q6_newFruit<i>.png where i=1,2,3.

Figure 61: New Fruit 1

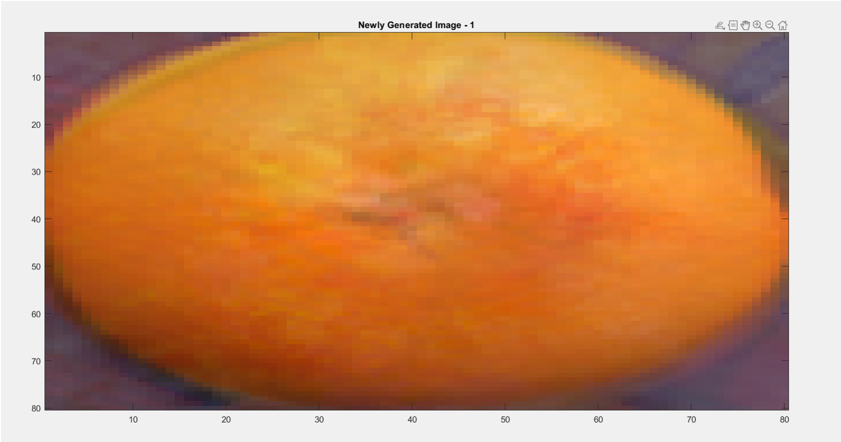


Figure 62: New Fruit 2

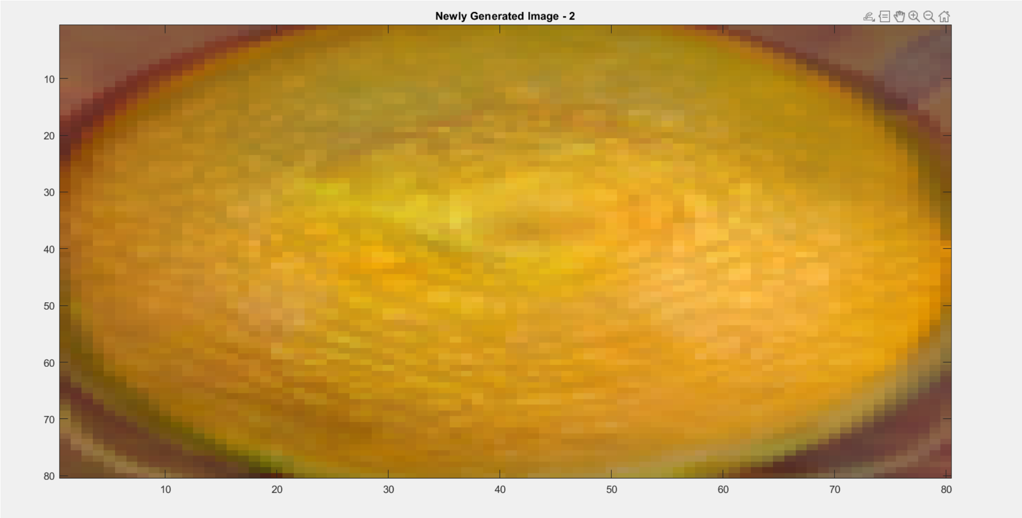


Figure 63: New Fruit 3