CS215 Assignment 2

Sarthak Mittal & Yash Patil

Contents

1 EUCLIDEAN PLANE

1.1 Ellipse

1.1.1 Algorithm

The equation of the region enclosed by an ellipse centred at origin and having a and b as lengths of semi-major and semi-minor axes respectively is given by (assuming X and Y to be the cardinal axes):

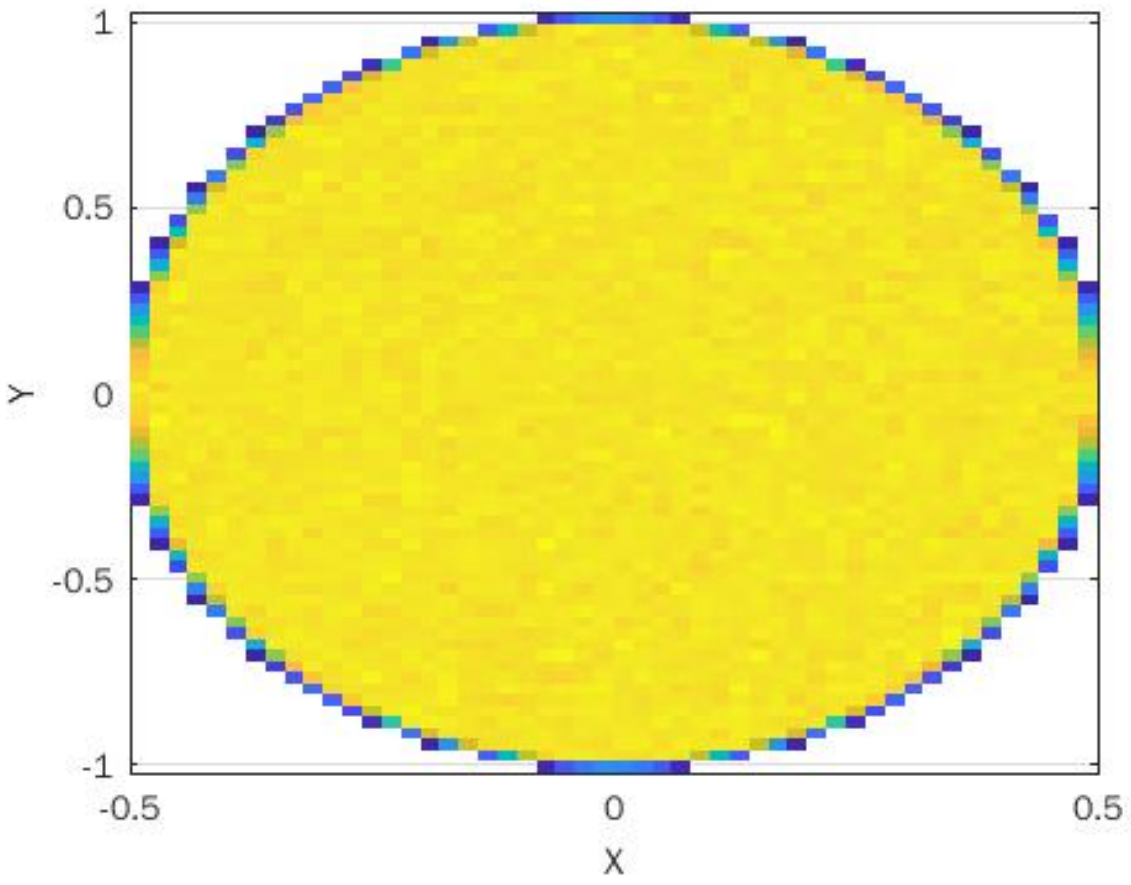
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1.$$

This equation can be parametrized as $x = ar\cos(\theta),\ y = br\sin(\theta)$, where $0 \leq r^2 \leq 1$ and $0 \leq \theta < 2\pi$.

To obtain $N = 10^7$ sample points that are uniformly distributed over the elliptic region, we choose uniformly (single dimension) distributed values of r^2 and θ (in their respective ranges) and then scale the resulting values (as per the parametrization) to obtain samples for (x, y). For each fixed value of θ, the transformation is just a linear scaling of r (for which all values in $(0, 1)$ are equi-probable) and hence the uniformity will be maintained, since θ is also linearly distributed, and because of the fact that linear transformation is invertible.

1.1.2 Histogram Plot

By running the file `ellipse.m`, we got the following plot:



1.2 Triangle

1.2.1 Algorithm

The region enclosed by the triangle inside the rectangle defined as:

$$\{(x, y) \mid 0 \leq x \leq \pi,\ 0 \leq y \leq e\}$$

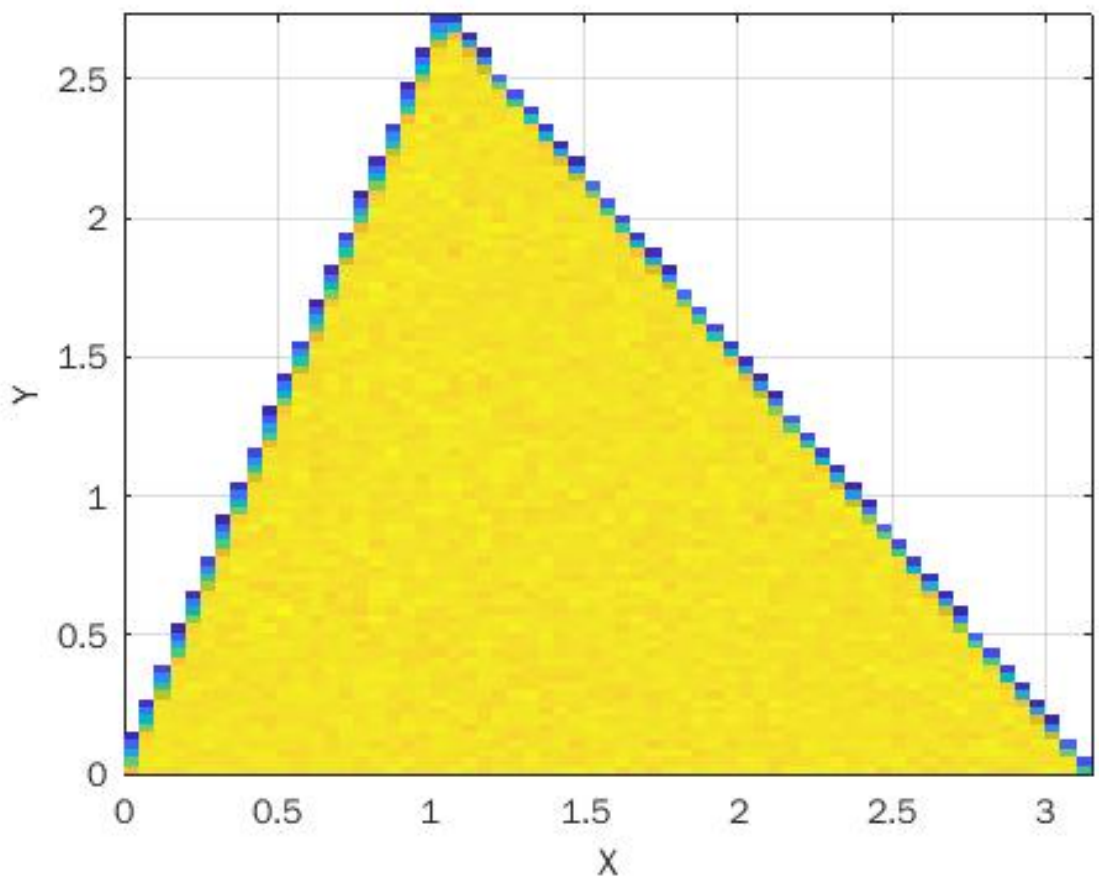
satisfies:

$$\{(x, y) \mid 0 \leq x \leq \frac{\pi}{3},\ 0 \leq y \leq \frac{3ex}{\pi}\} \bigcup \{(x, y) \mid \frac{\pi}{3} \leq x \leq \pi,\ 0 \leq y \leq \frac{3e(\pi - x)}{2\pi}\}$$

To obtain $N = 10^7$ sample points that are uniformly distributed over the triangular region, we choose uniformly (single dimension) distributed values of x and y (in their respective ranges in the bounding rectangle) and then filter out the points (using conditions as per the restrictions) to obtain valid samples for (x, y). The number of valid sample points will be roughly half (ratio of area of triangle to rectangle) the number of sample points taken inside the rectangular region (due to uniform distribution over areas). Hence we choose twice $(2N)$ the number of points inside the rectangle before filtering them out. Since we are only choosing a subset of the original uniform distribution, the final sample obtained will also be uniform.

1.2.2 Histogram Plot

By running the file `triangle.m`, we got the following plot:



2 MULTIVARIATE GAUSSIAN

2.1 Generating Sample

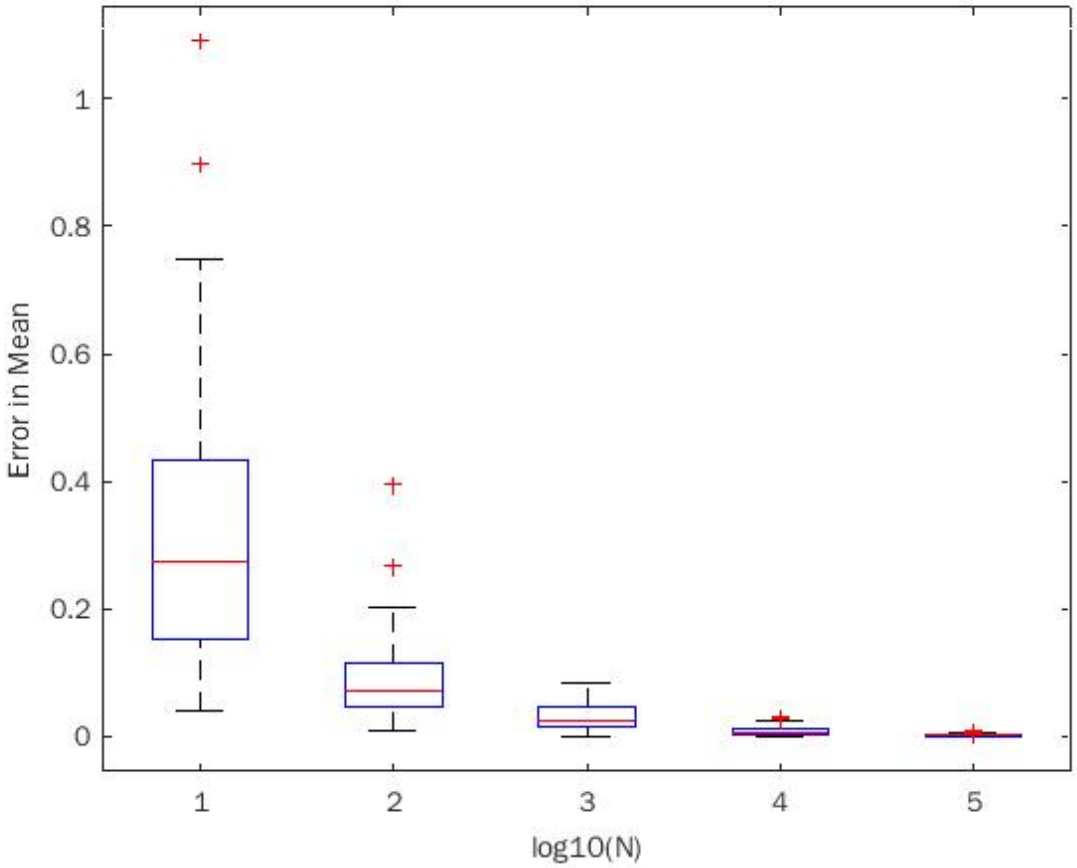
We have been given the covariance matrix C and mean μ. We know that for a $d - dimension$ multivariate Gaussian distribution, the sample can be represented by a matrix equation of the form:

$$X = Aw + \mu$$

where w is a $d \times N$ sample obtained from standard $d - dimension$ multivariate Gaussian distribution, and μ is the sample mean. The covariance of X is then given by $C = AA^T$. Hence, to obtain the matrix A from given covariance matrix C, we need to solve the matrix equation. The lower-triangular solution is obtained using the Cholesky Decomposition. Once we obtain A, we can generate random samples from standard distribution and use the above matrix equation to obtain sample points for X (since A, w and μ will be known/evaluated). The sample is stored in the matrix `X` generated by the file `mvgaussian.m`. The ML estimates and their relevant formulae have been used in the code submitted.

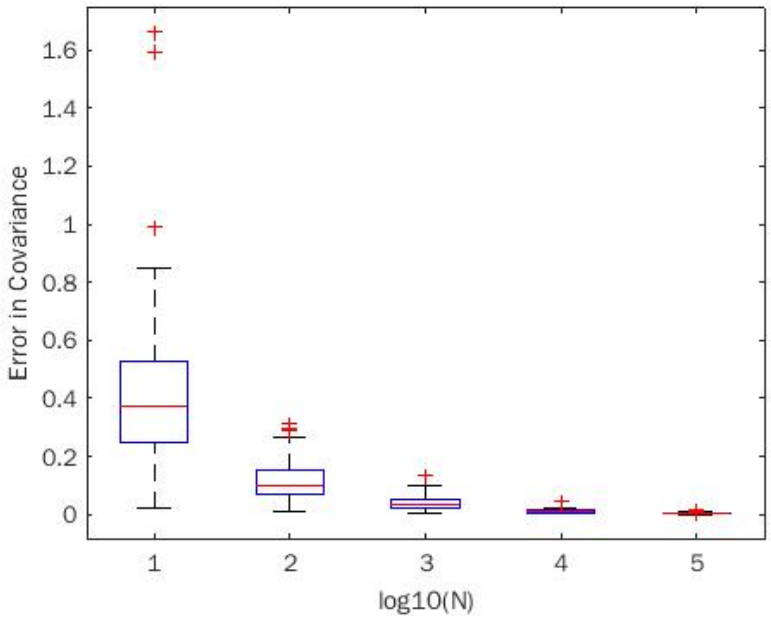
2.2 Plot of Error in Mean

By running the file `meanerror.m`, we got the following plot:



2.3 Plot of Error in Covariance

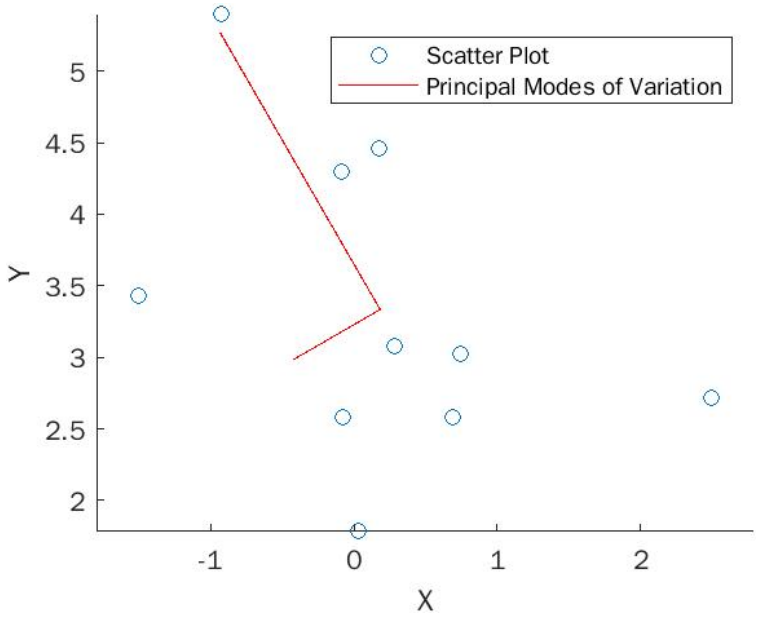
By running the file `coverror.m`, we got the following plot:



2.4 Plots of Principal Modes of Variation

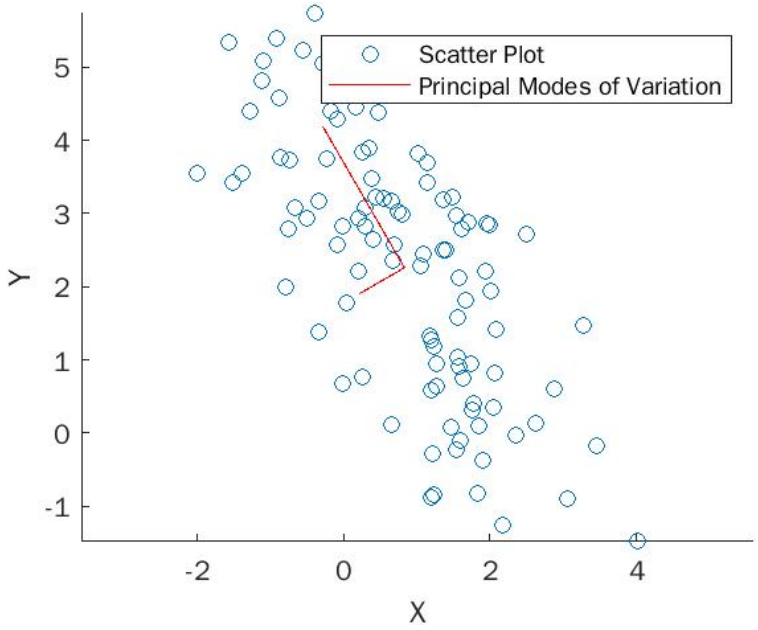
2.4.1 Plot for N = 10

By running the file `pmvgaussian1.m`, we got the following plot:



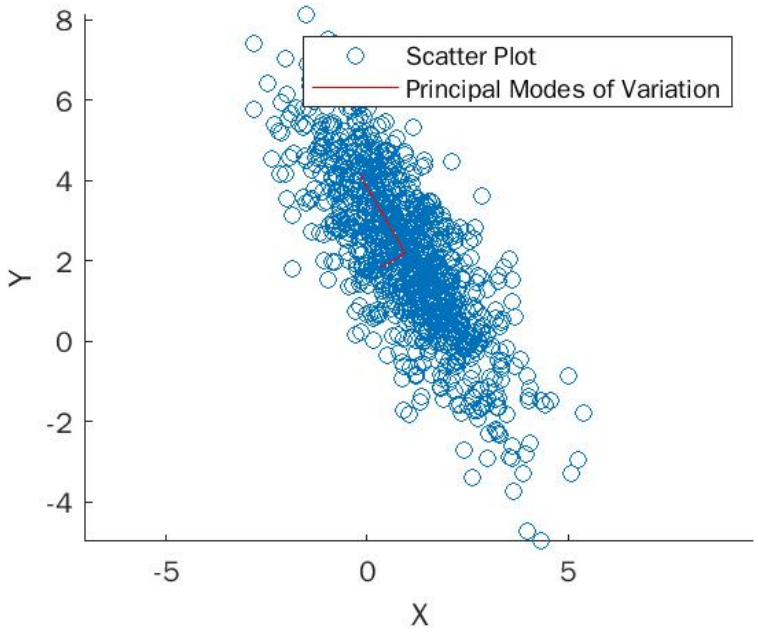
2.4.2 Plot for N = 100

By running the file `pmvgaussian2.m`, we got the following plot:



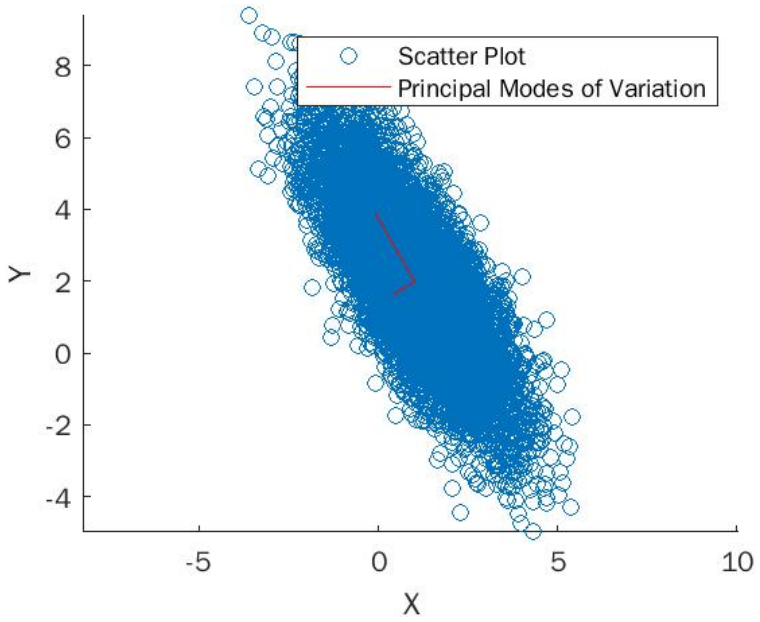
2.4.3 Plot for N = 1000

By running the file `pmvgaussian3.m`, we got the following plot:



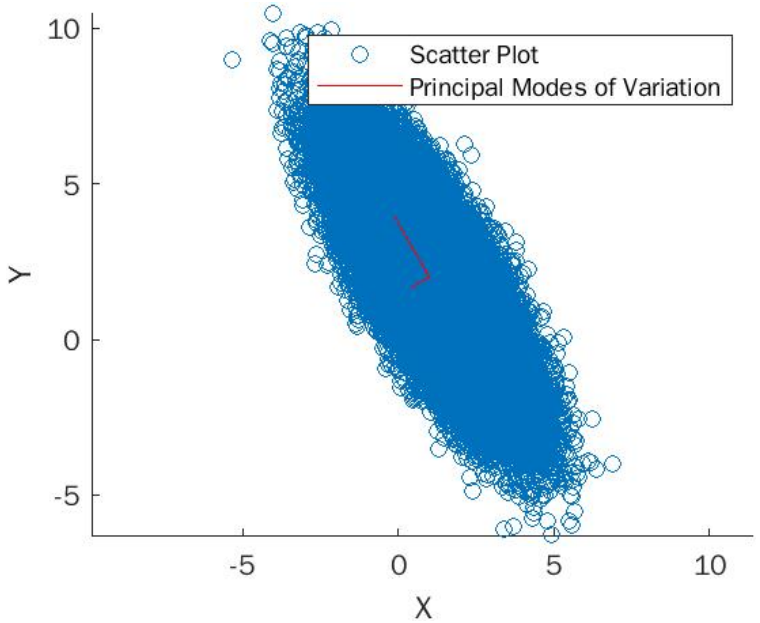
2.4.4 Plot for N = 10000

By running the file `pmvgaussian4.m`, we got the following plot:



2.4.5 Plot for N = 100000

By running the file `pmvgaussian5.m`, we got the following plot:



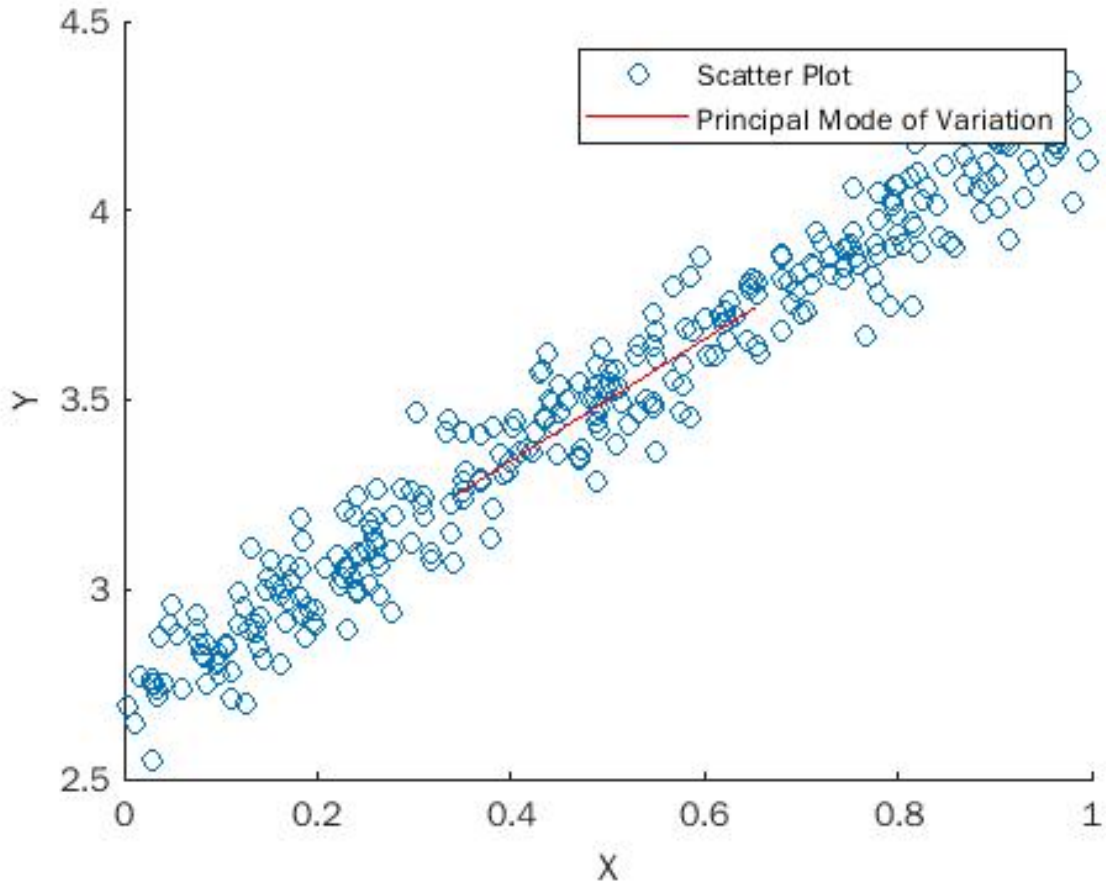
3 PCA AND HYPERPLANE FITTING

3.1 Approximation using PCA

We first convert the sample to mean-centric and then find out the ML estimate for the covariance. The diagonal entries of the covariance matrix give us a rough idea of the spread and non-diagonal ones give the extent of correlation of components. Then we calculate the eigenvectors and eigenvalues for covariance matrix, which are of great significance since their direction remains unaltered upon action (pre-multiplication) of the covariance matrix. Generally the largest eigenvalue (and the corresponding eigenvector) are the ones primarily responsible for the rough trend of the sample. This eigenvector (also called first principal direction) is the most significant direction of the data set. Since it accounts for the largest fraction of the total variance, and generally the largest eigenvalue is much greater than the rest. PCA essentially reduces the dimension of the data set and provides a linear approximation (along this eigenvector) of the relation between the random variables. **The line shown in the graphs is only a segment of the complete line**.

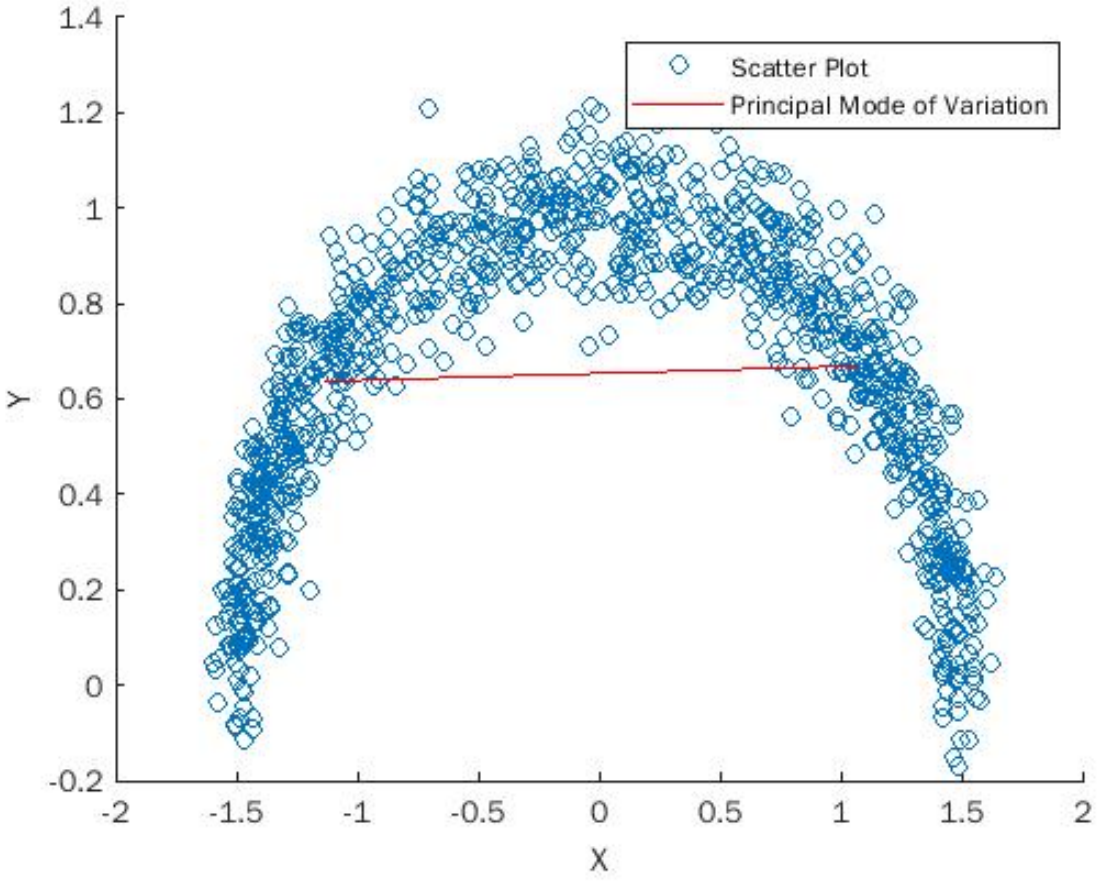
3.2 Plot for Data Set 1

By running the file `pcahypfit1.m`, we got the following plot:



3.3 Plot for Data Set 2

By running the file `pcahypfit2.m`, we got the following plot:



3.4 Quality of Approximation

Since the first data set already has a fairly linear relation, the principal mode of variation gave a line that passes near a lot of points, because maximum variance upon projection will be obtained for this line. The second data set is highly curved and non-linear, and so the maximum variance will be obtained when the points farther away from the mean (which will lie near $x = 0$ due to symmetry) are projected (almost) vertically. Hence the slope of the line is almost zero. The approximation of second data set is less accurate than first data set mainly because of the large variation in average rate of change of Y with increase in X in the data set.

4 PRINCIPAL COMPONENT ANALYSIS (PCA)

To begin with, the major requirement was to load the data from the `mnist.mat` file. We used the `load` function for that. The data was stored in the matrix named `data_train` having data of $28^2 \times 60000$ sample of images. We looped on the digits to match it with `labels_train` and stored the sample into `digitData`. We used the function `im2double` to convert the data set into a double matrix. The rest of the first set of loop code is just manipulation of `digitData` and `image2D`.

4.1 Calculation of Mean

We used the `reshape` function on `digitData` and then evaluated the mean using `sum` function along rows, and used `size` function to get row size.

4.2 Calculating Covariance Matrix

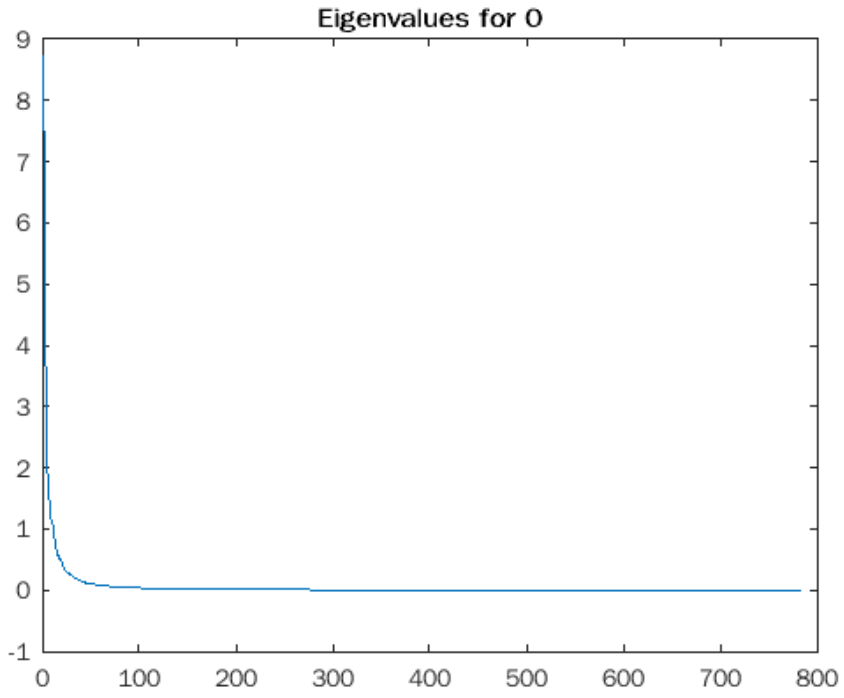
Using the mean calculated above we centered the data to its mean (`meanCentric = digitData - digitMean`). Then we found the MLE of covariance matrix (`cov = meanCentric * meanCentric' / size(digitData, 2)`).

4.3 Principal Mode of Variation

We used the `eig` function to calculate the eigenvectors for covariance matrix (`[V,D] = eig(cov)`). Then we sorted the eigenvalues (in descending order) (`[~,X] = sort(diag(D),'descend')`) and accordingly sorted eigenvectors also (`Vsort = V(:,X)`). Then we picked the first eigenvalue and first column of eigenvectors.

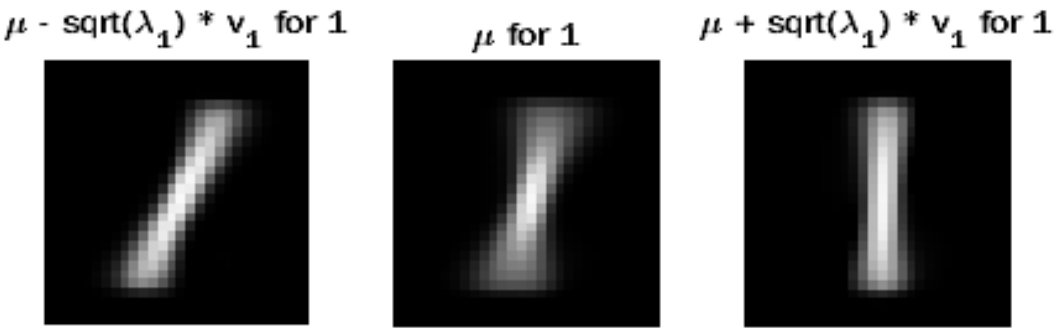
4.4 Significant Modes of Variations

For further calculations, we converted each 28×28 pixel image matrix to a $28^2 \times 1$ vector by concatenating its columns.



Observations: There is a drastic decrease in the number of large eigenvalues. This is the general observation with all of the digit graphs. Roughly there are around 50-80 principal mode of variations for each digit, and the graph becomes nearly 0 beyond that. The observed number of principal modes of variations are far less than 784. This is specifically because all the significant eigenvalues are covered in far lesser dimensions (50-80), and so majority of the variance is covered by them, and PCA is able to dimensionally reduce the data.

4.5 Principal Mode of Variation around Mean



Observations: For the digits, the mean shows us how people write the digit on an average. The other two plots show the plots which are deviated by the principal eigenvector, thus covering maximum variance region. Hence the plots show the tendency of people in writing the digit in a slightly inclined way (for digit 1).

NOTE: The remaining plots and images have been submitted in the problem4 folder in results.

5 PCA FOR DIMENSIONALITY REDUCTION

5.1 Function for Dimension Reduction

We defined a function that would take the data (the image vector and the dimension for the basis) as input and would lead to the transformation of the data. We calculate the mean, covariance matrix and the eigenvectors and eigenvalues of the data, roughly as done in previous question. The procedure to order the eigenvalues and eigenvectors also carries forward.

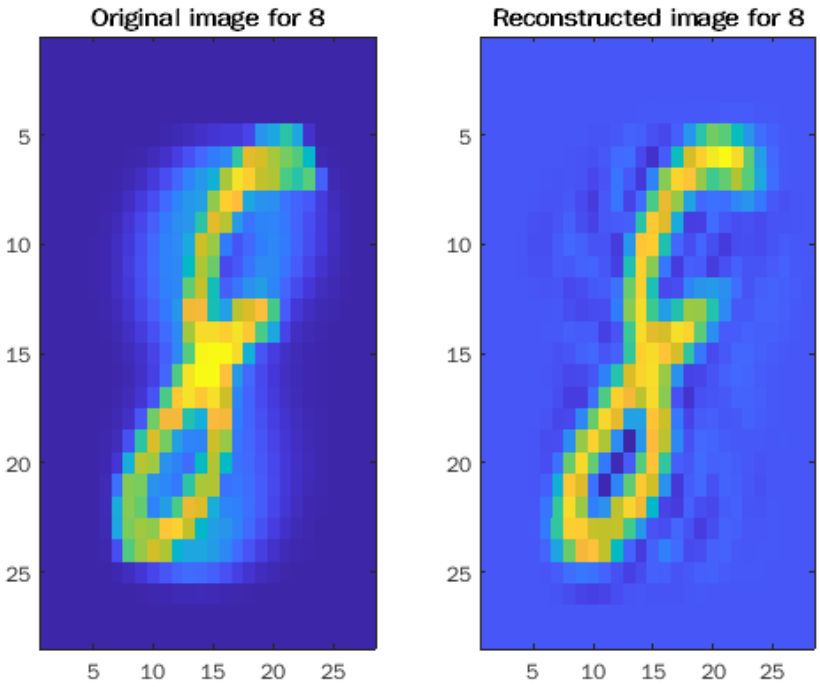
While defining the basis for the new vector space, we used the first n eigenvectors, since in PCA, the cardinal axes align with the columns of the eigenvector matrix.

For the projection matrix, the transpose of the basis matrix is multiplied with the original data matrix so as to map the data to n dimensions. After multiplying as above the value of R_{ij} where R is the reduced data matrix, is the inner product of the i^{th} eigenvector with the j^{th} sample (the j^{th} column in digits). This is how the function reproduces a reduced matrix of n dimension.

5.2 Algorithm for Regeneration

When aligning/projecting the data for the new system we multiplied the the transpose of the n dimentional eigenvector matrix with the data which in turn gave us the data aligned with the new cardinal axis (the new principal modes of variation). Then we multiply it with the basis to get to a suitable linear combination of the original data.

Essentially this would ensure that the j^{th} column in reconstructed matrix is the linear combination of the bases with coefficients from inner products with the j^{th} sample.



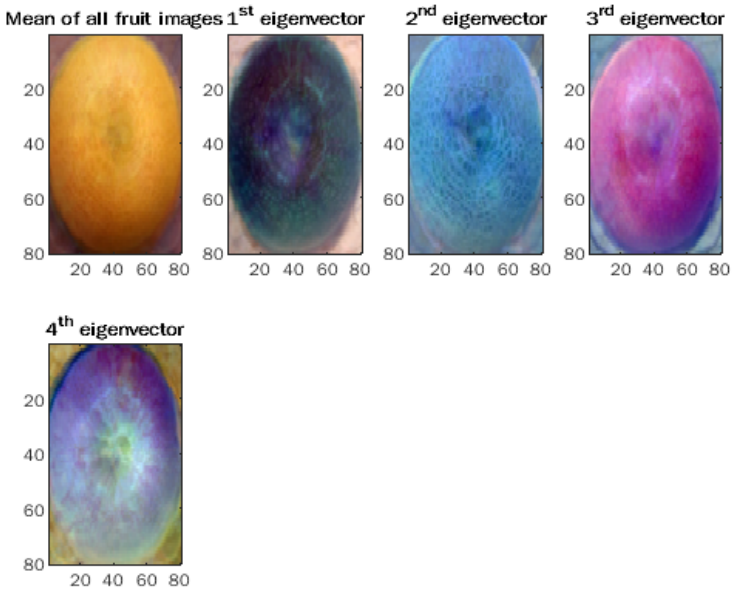
NOTE: The remaining plots and images have been submitted in the problem5 folder in results.

6 PCA FOR ANOTHER IMAGE DATASET

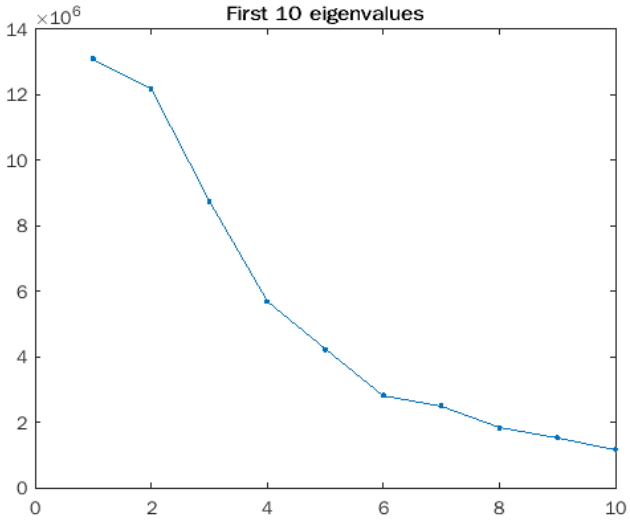
We loaded the data using `images = dir(fullfile(Dir,"*.png"))`. Then we looped through the images and rescaled and reshaped them (`final-image(:,i) = reshape(im,[19200, 1])`).

6.1 Mean, Covariance and PCA

As before, we calculated the mean, made the data mean-centric and calculated covariance matrix. Then we took the top 4 largest eigenvalues and eigenvectors (`[eigen-vector, eigen-values] = eigs(covariance,4)`).

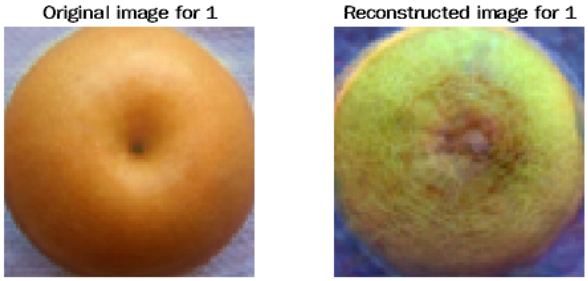


Then we took top 10 eigenvalues (`first-10-eigen-values = eigs(covariance,10)`).



6.2 Closest Representation

The closest representation will be the projection of the image on the vector space formed by the four eigenvectors. This is because whenever we use PCA, the closest representation of the linear combination of n eigenvectors is the multiplication of the transpose of the n eigenvectors with the original data matrix which creates a projection matrix when this matrix is multiplied with the basis of observation ie n eigen vector it then creates the image of the required dimension. We found the projection matrix as before, and then took dot product of eigenvectors with the generated projection matirx of image to find coefficients of linear combination, and then added the mean to the projected matrix to make the re-structured images.

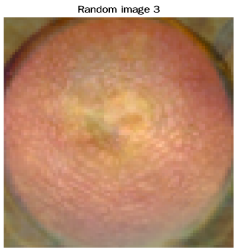


6.3 Random Images

We used the `rng` function to generate random images, and obtained relative weights for them (after making them comparable to the norm of the mean). We constructed the new image with dimension 19200×1 by re-structuring the random one as done earlier. Here we have used random linear combination of the top 4 eigenvectors to generate new fruits that are not part of the data set. Since these variables should be in accordance with the data set which is given hence while calculating the projection matrix did the

w = randn(4,1) * norm(imageMean);

now the reconstructed matrix for the image is generated by multiplying the basis of interest here the top four eigen vector matrix with this **w**(projection) matrix to get the final constructed image



NOTE: The remaining images have been submitted in the problem6 folder in results.