CS215 Assignment 2

Dhananjay Raman
Shantanu Welling

October 2022

Contents

Problem 1

1. This is similar to sampling random points within a unit circle uniformly.
   Sample angle co-ordinate uniformly between $[0, 2\pi)$ by mapping $[0, 1)$ uniform distribution to $[0, 2\pi)$.
   Next, we sample the radial co-ordinate. (By considering the angular symmetry of the polar angle co-ordinate) Since we want a uniform distribution over a circle and the circumference of circle is a linear function of radius and number of points on a circle of radius r is proportional to the circumference of the circle, the PDF of random variable R which represents the radial co-ordinate is a linear function of r.
   Since r goes from 0 to 1, we can establish that the PDF of R is $2r$.
   Therefore, CDF is r^2. CDF is $P(R \leq r)$.
   Using Transformation of RVs CDF^{-1} method...
   For a uniform RV U on $[0, 1]$, $P(U \leq u) = u$.
   $\implies F_R(r) = P(R \leq r) = P(U \leq F_R(r)) = P(F_R^{-1}(U) \leq r)$.
   So, $R = F_R^{-1}(U)$. Therefore $R = \sqrt{U}$.
   Sample radial co-ordinate accordingly.Get x and y co-ordinates by transformation of polar co-ordinates to cartesian.
   Scale the x and y axes to the axes lengths of the ellipse.

2. Plot of sampled points for $N = 10^7$:

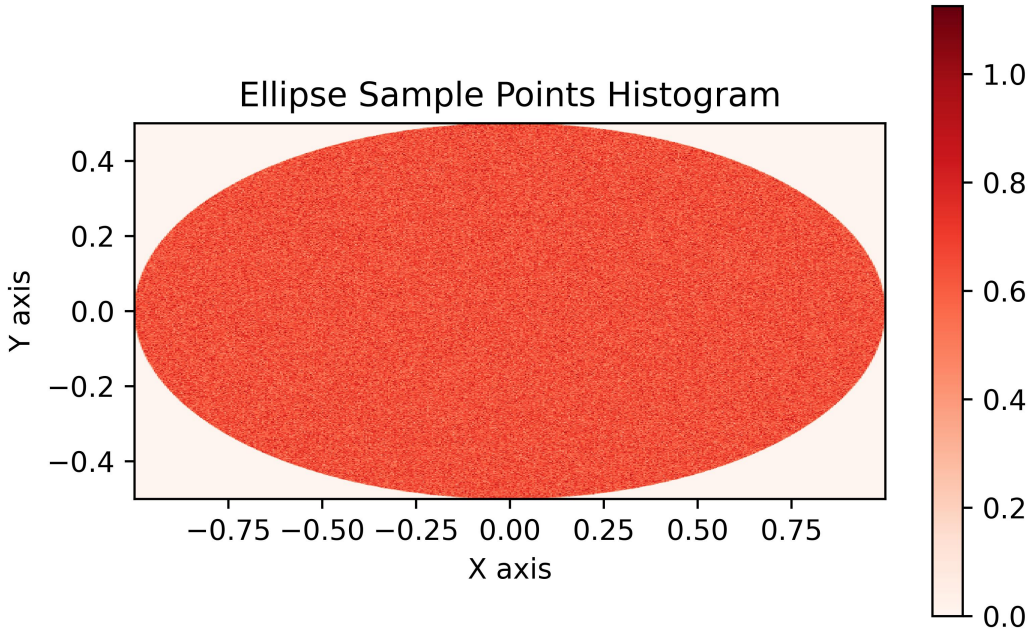

Figure 1: Uniform Sampling inside ellipse

3. This is like sampling random points inside a right angled triangle with vertices (0,0), (0,1) & (1,0).
   We calculate the vectors u, v which represent two adjacent sides of the triangle (which intersect at $(0, 0)$), uniformly sample points in the range$(0, 1)$, and take them in pairs (x, y). Then, it is easy to see that $xu + yv$ gives us a uniform distribution on the parallelogram formed by taking u and v as sides.
   If this point lies inside the given triangle (which represents half of the parallelogram), we are done, otherwise it must lie in the other half, and reflecting the point across the center of the parallelogram gives us a valid sample. In case of the triangle given in the question, we scale the vectors u & v to the adjacent sides' lengths of the corresponding triangle which coincide at (0,0) and take the direction of these vectors along the corresponding sides of the triangle.

4. Plot of sampled points for $N = 10^7$:

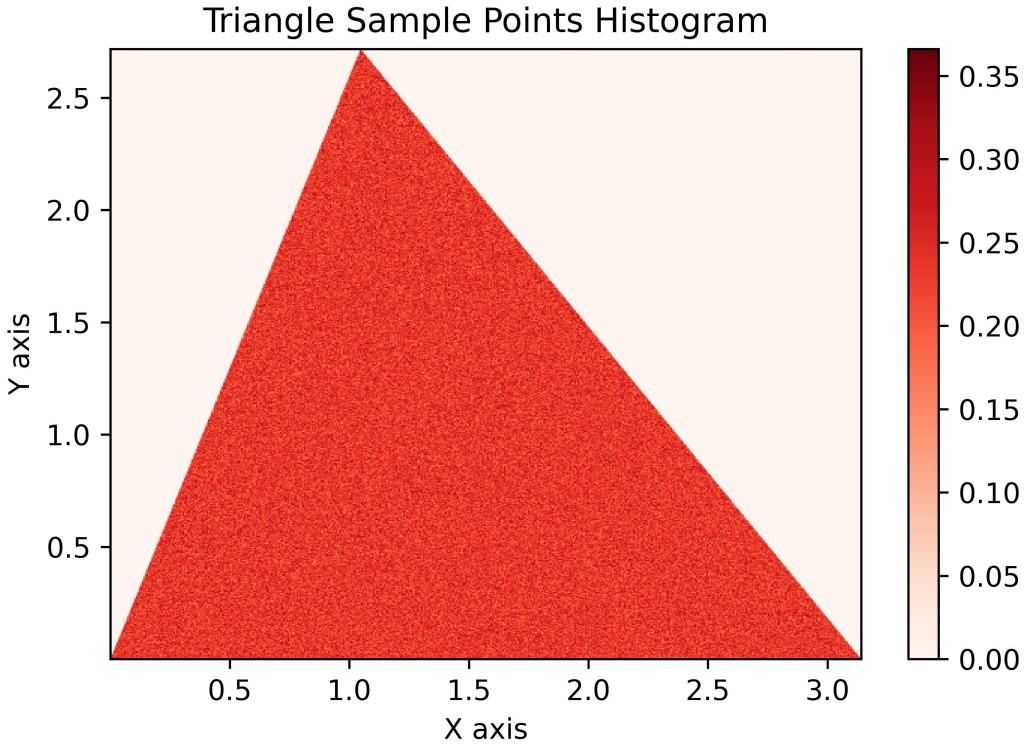

Figure 2: Uniform Sampling inside triangle

Problem 2

1. We know that $C = AA^T$, where A is a given matrix, and C is its covariance matrix. Here we want to compute A, given C.
   Since C is a real symmetric matrix, by the **Spectral Theorem** there exists matrices Q and D such that

   $$C = QDQ^T$$

   and in fact we can find them by just computing the eigenvectors and eigenvalues of C. Hence, Q is a matrix with the columns as distinct eigenvectors of C, while D is a diagonal matrix with each diagonal value as the square root of the corresponding eigenvalue. We can also write

   $$C = (QD^{1/2})(D^{1/2}Q^T) = AA^T$$

   where $A = QD^{1/2}$ (since $D^{1/2}$ is diagonal, its transpose is itself). This is how we computed A in our answer, and now we just sample standard normal gaussian 2×1 vector W, and set our RV as (where μ is the mean vector)

   $$X = AW + \mu$$

2. Boxplot of the error between the true mean μ and the ML estimate $\bar{\mu}_N$ vs $log_{10}N$ (taking measure of error as standardized l^2 norm):

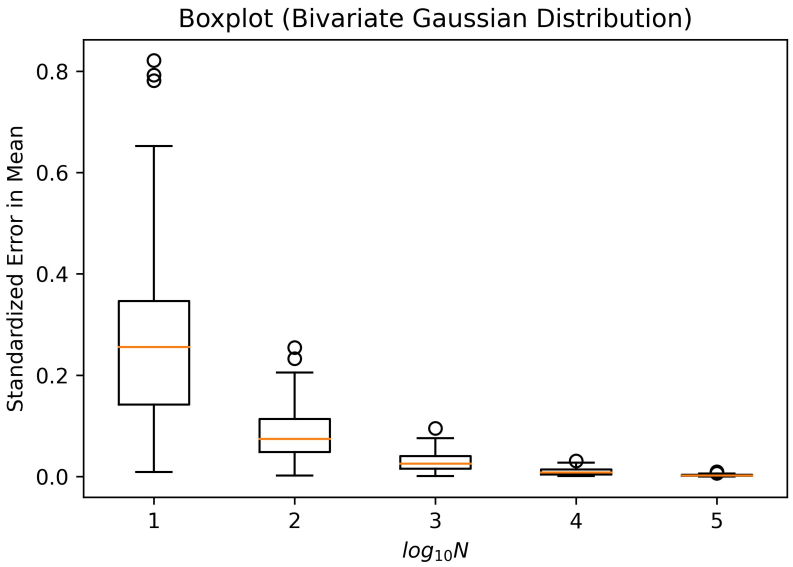

Figure 3: Error in estimating mean of Bivariate Gaussian Distribution

3. Boxplot of the error between the true covariance C and the ML estimate $\bar{C}_N$ vs $log_{10}N$ (taking measure of error as standardized Frobenius norm):

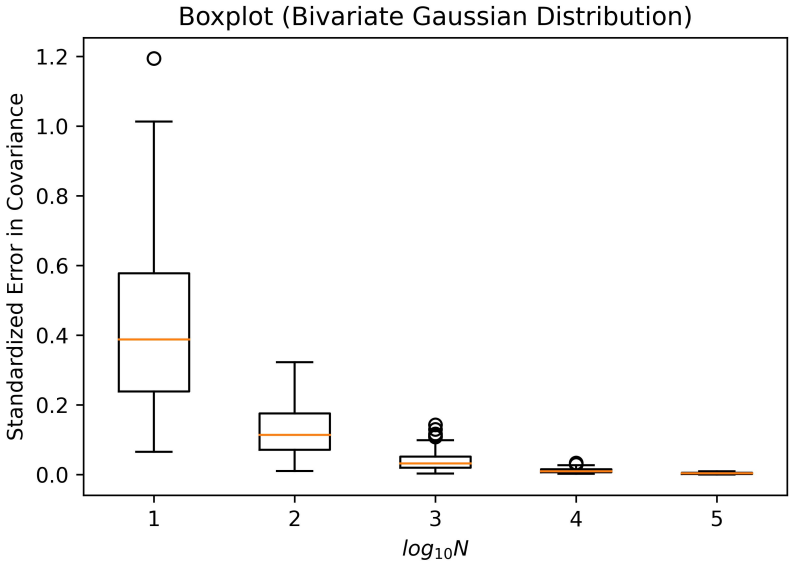

Figure 4: Error in estimating covariance of Bivariate Gaussian Distribution

4. Scatterplot of random samples of Bivariate Gaussian vs $log_{10}N$, marked with scaled principal modes of variation

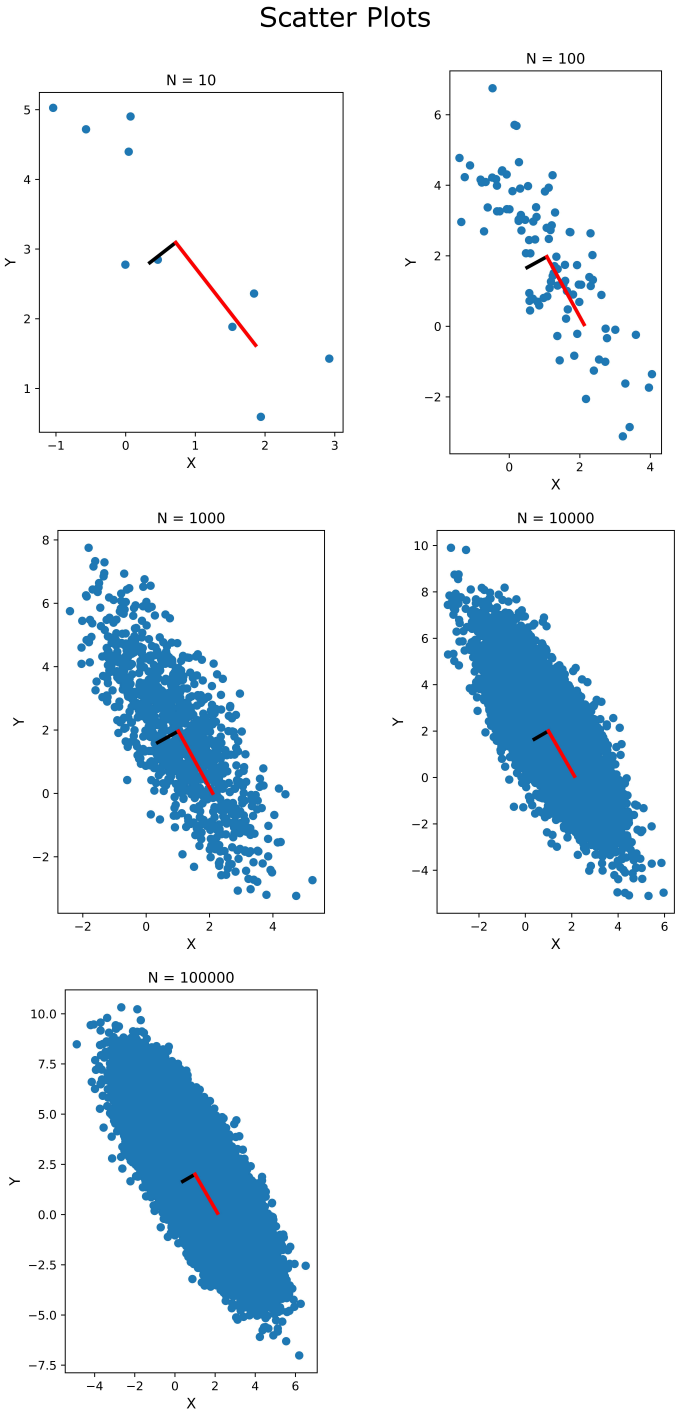

Figure 5: Distribution of points for a Bivariate Gaussian, red and black vectors show principal modes of variation

Problem 3

1. We see that there is a linear relationship between X and Y. Applying PCA on a distribution gives us the principal mode of variation for that distribution. In a linear relationship, the variation is only along one direction which is the principal mode of variation. Hence applying PCA on (X, Y) will give us a principal mode of variation which coincides with the slope of the relationship. To find a point which passes through the line, we take the sample mean of the distribution. Since the linear relationship between 2 RVs X & Y is given by $Y_i = \alpha + \beta X_i + \eta_i$ (where errors η_i are 0 mean i.i.d Gaussian RVs), the point $(\bar{x},\bar{y})$ which is the sample mean of the data, lies on the linear model.
   To apply PCA without using the built-in *pca*() function, we calculate the sample mean and covariance using *sum*() and matrix multiplication. Then we use *eig*() on the covariance matrix, and take the vector which has the largest eigenvalue (measure of spread) compared to the other, as the slope of the line.

2. Scatterplot of distribution of X vs Y in set 1, overlaid with principal mode of variation:

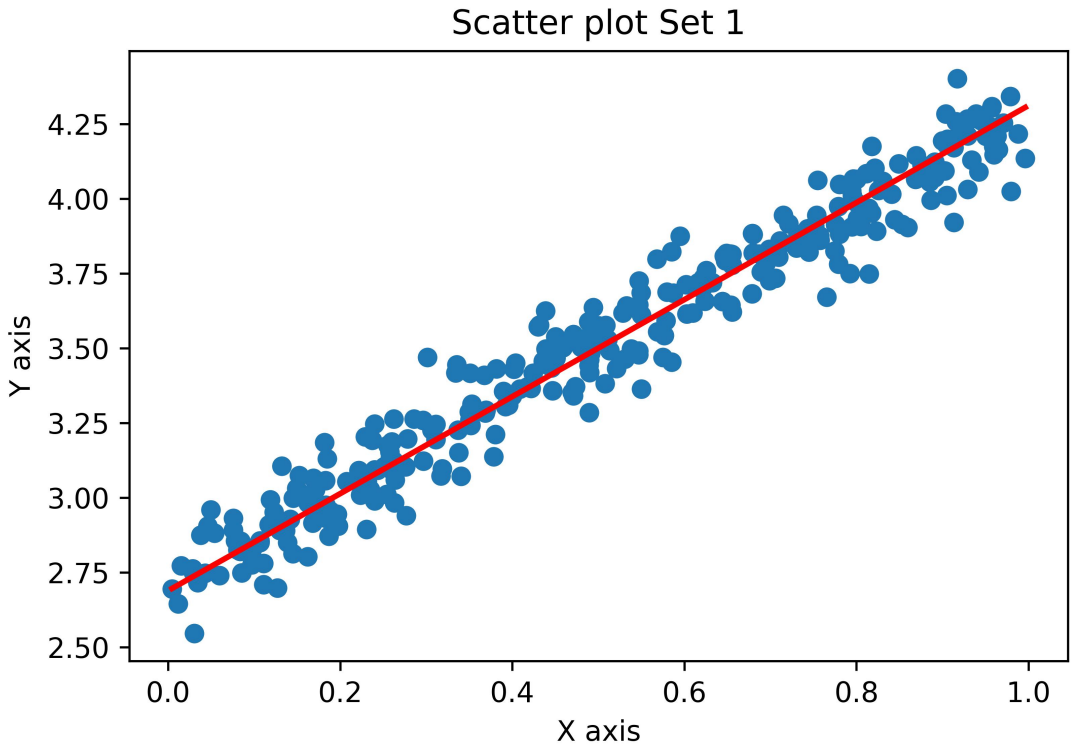

Figure 6: Principal model of variation computed for given distribution (set 1)

3. Scatterplot of distribution of X vs Y in set 2, overlaid with principal mode of variation:

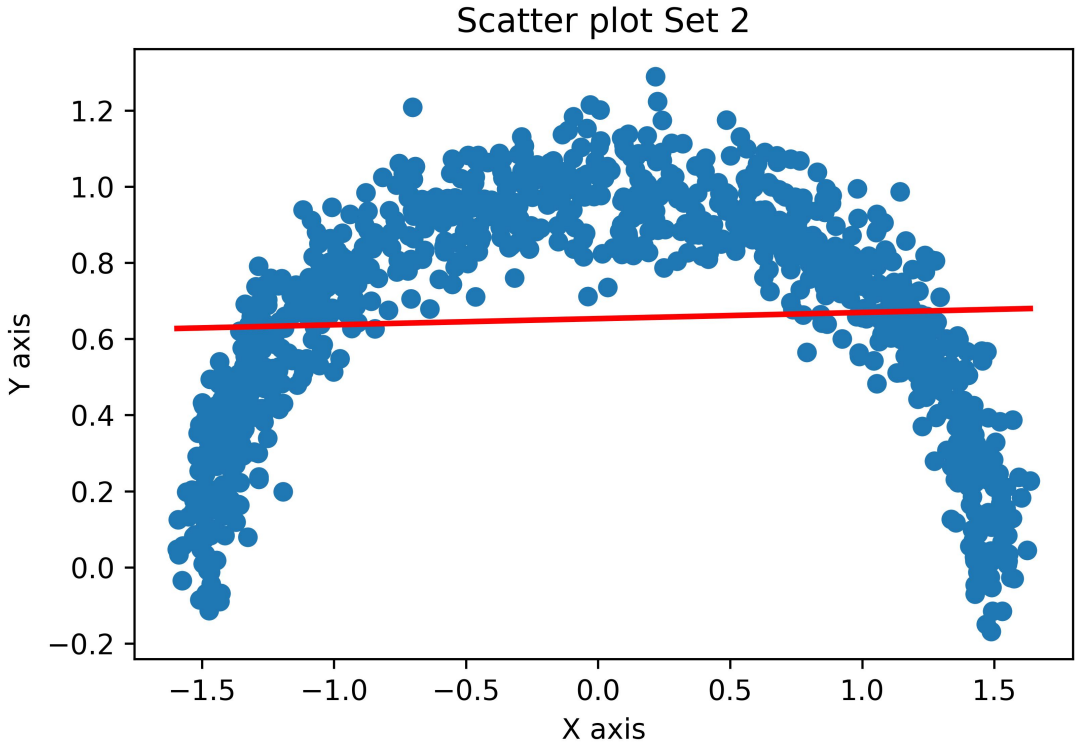

Figure 7: Principal model of variation computed for given distribution (set 2)

Applying PCA on the first set gave a sufficiently good linear relationship approximation as the variation of the distribution was majorly along one direction but PCA on the second distribution does not give us relevant information about the initial distribution, as our assumption that relationship between X and Y is linear is clearly incorrect (by observation, and also since both eigenvalues are comparable in this case), and the variation of the distribution is not majorly along one direction.

Problem 4

1. Graph of eigenvalues for each digit, sorted:

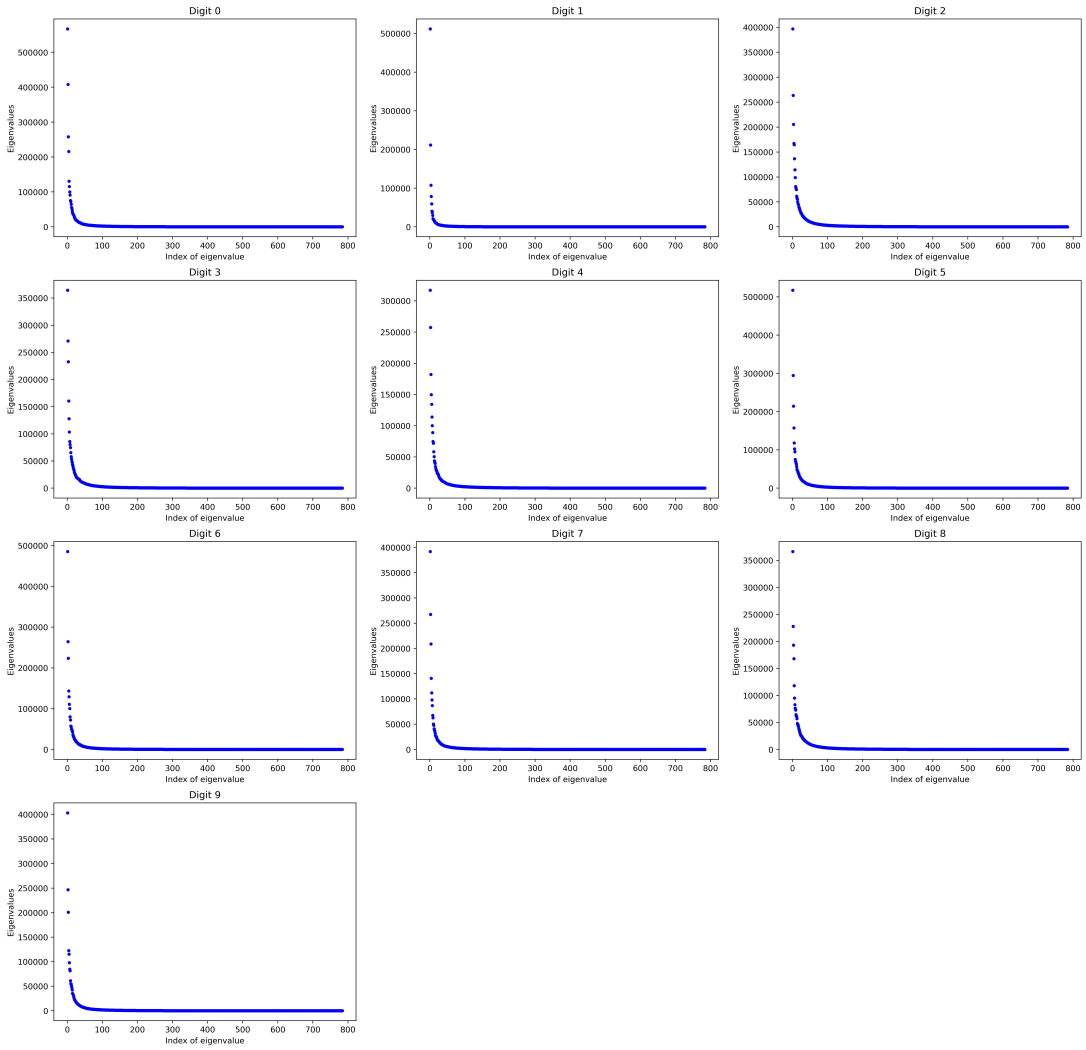

Figure 8: Eigenvalues computed for each digit

Analysis:
We see that there are very few significant eigenvalues compared to the vast majority of negligible eigenvalues.
On average there are 8-10 significant eigenvalues/modes of variations for each digit, which is far less than 784.
Hence, we can conclude that everyone writes the same digit with very less variation. The rest represent very minute variation in digit writing style, reflected in the very small variance/eigenvalue for them.

2. Images depicting principal mode of variation about the mean for each digit:

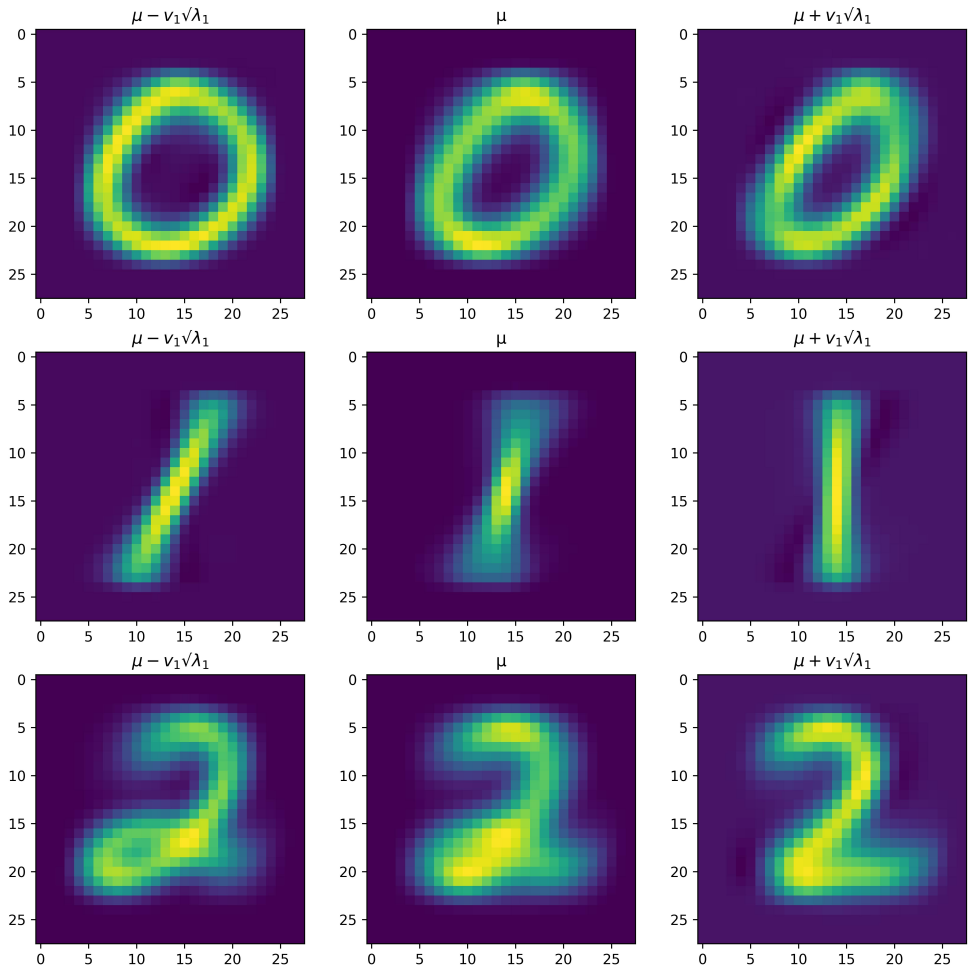

Figure 9: Principal mode of variation for digits 0-2

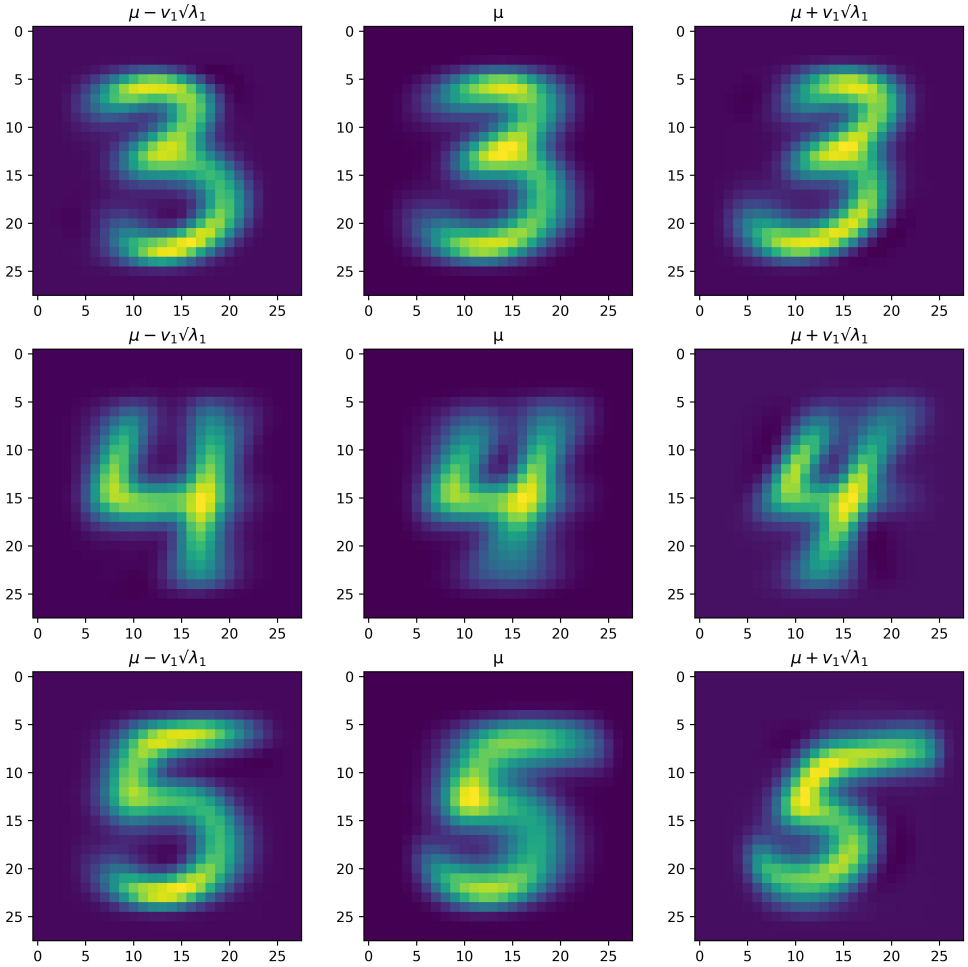

Figure 10: Principal mode of variation for digits 3-5

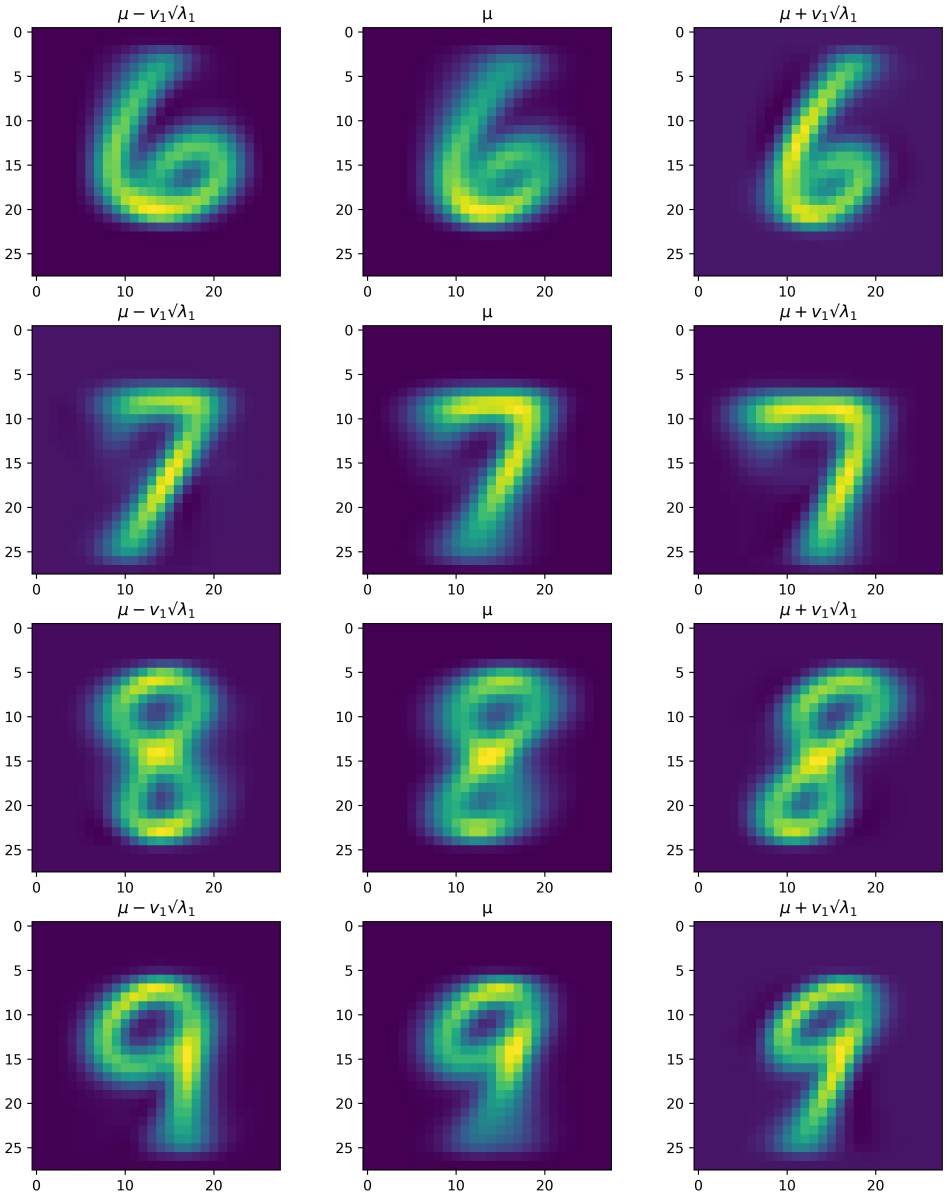

Figure 11: Principal mode of variation for digits 6-9

Analysis:
We observe the many different styles in which people tend to write the same digit in, and compare and contrast between them using the principal mode of variation.

Every digit appears to be wobbling about the mean image. The mean represents how people draw the digit on an average. The two images on either side show the extremes of the drawing style of people for that digit.
For digits like $1, 3, 4, 5, 8, 9$, the variation shows us that people either tend to write these digits in the upright position, or slanted slightly in the clockwise direction.
For digits like $0, 2$ the size of a loop in the shape tends to change.
Yet for digits like $6, 7$ the length of the straight portion of the shape varies from person to person.

Problem 5

1.
```
def reduce_dimensions(digits_train, labels_train):
    #Function to find 84 principal eigenvalues and corresponding
        ↪ eigenvectors
    digits=[[] for i in range(10)] #Store data of each digit
    digit_count=[0 for i in range(10)] #Number of traincases for
        ↪ each digit
    for i in range(len(digits_train)):
        digit_count[int(labels_train[i])]+=1
        digits[int(labels_train[i])].append(digits_train[i])

    meanmat=[np.zeros((28*28,1)) for i in range(len(digits))] #Mean
        ↪ vector for each digit

    for digit in range(len(digits)): #Compute mean
        for i in range(digit_count[digit]):
            meanmat[digit]+=np.reshape(digits[digit][i],(28**2,1))
        meanmat[digit]/=digit_count[digit]

    covmat=[np.zeros((28*28,28*28)) for i in range(len(digits))] #
        ↪ Covariance matrix for each digit
    for digit in range(len(digits)): #Compute covariance matrix
        for i in range(digit_count[digit]):
            covmat[digit]+=np.matmul(np.reshape(digits[digit][i
                ↪ ],(28*28,1)),np.transpose(np.reshape(digits[digit][
                ↪ i],(28*28,1))))
        covmat[digit]/=digit_count[digit]
        covmat[digit]-=np.matmul(meanmat[digit],np.transpose(meanmat[
            ↪ digit]))

    evecs_all=[] #List of list of eigenvectors of each digit
    evalues_all=[] #List of list of eigenvalues of each digit
    for i in range(10):
        evalues, evecs = np.linalg.eigh(covmat[i]) #Find eigenvalues
            ↪ and eigenvectors for each digit
```

```
        evec_eval_pair = {evalues[j]:evecs[:,j] for j in range(len(
            ↪ evalues))}
        evalues = np.flip(np.sort(evalues)) #Sort eigenvalues
        evalues = evalues[0:84] #get top 84 eigenvalues
        evecs = np.array([evec_eval_pair[evalue] for evalue in
            ↪ evalues]) #get corresponding eigenvector
        evecs_all.append(evecs)
        evalues_all.append(evalues)

    return (evecs_all, evalues_all, meanmat) #Return tuple of
        ↪ eigenvectors & eigenvalues of all digits and the mean of
        ↪ all digits

(evecs, evals, meanmat) = reduce_dimensions(arrays['digits_train'],
    ↪ arrays['labels_train'][0])
#Find first 84 principal modes of variation for all digits from 0-9

def compute_coordinates(image, image_label, evecs, m):
    return np.matmul(np.transpose(image-m),np.transpose(evecs[
        ↪ image_label]))
#Function to reduce dimensionality of each image and compute those
    ↪ 84 co-ordinates
#Takes input an image array, label of the image, and top 84
    ↪ eigenvectors of that corresponding digit generated from
    ↪ reduce_dimensions function
#m denotes mean vector of that digit
```

2. We know the coordinates of the image in the reduced 84-dimension space. We scale each of the 84 principal eigenvectors of 784 dimensions that generated this space with its corresponding coordinate (which is a real number) and take the vector sum of all these scaled vectors, then finally add the mean vector to it. This is the reconstructed image as we reduced the dimensionality of the original image to 84 by taking the projection of the original image onto the 84 dimensional hyperplane generated by 84 principal eigenvectors.

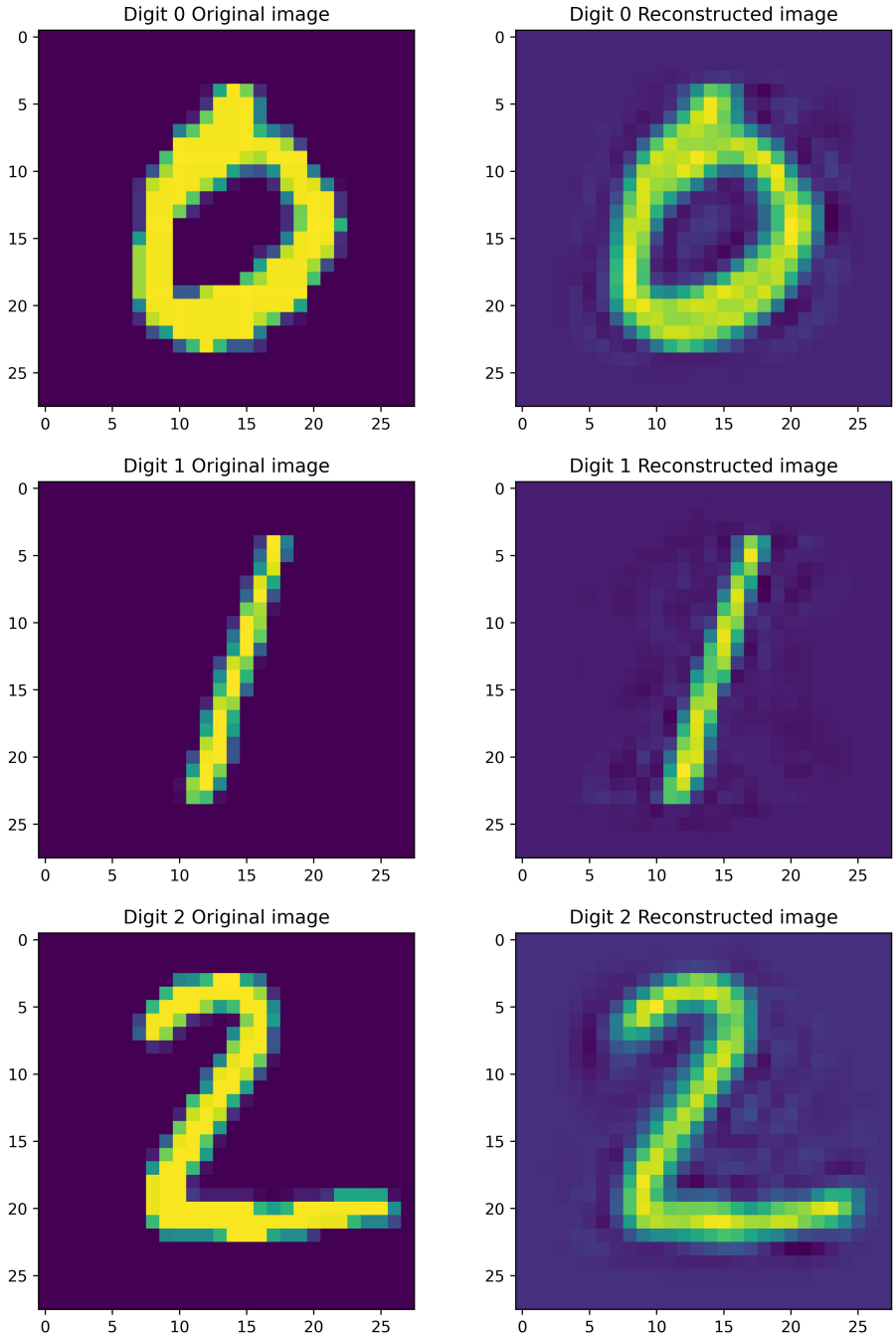

Figure 12: Original & Reconstructed images for digits 0-2

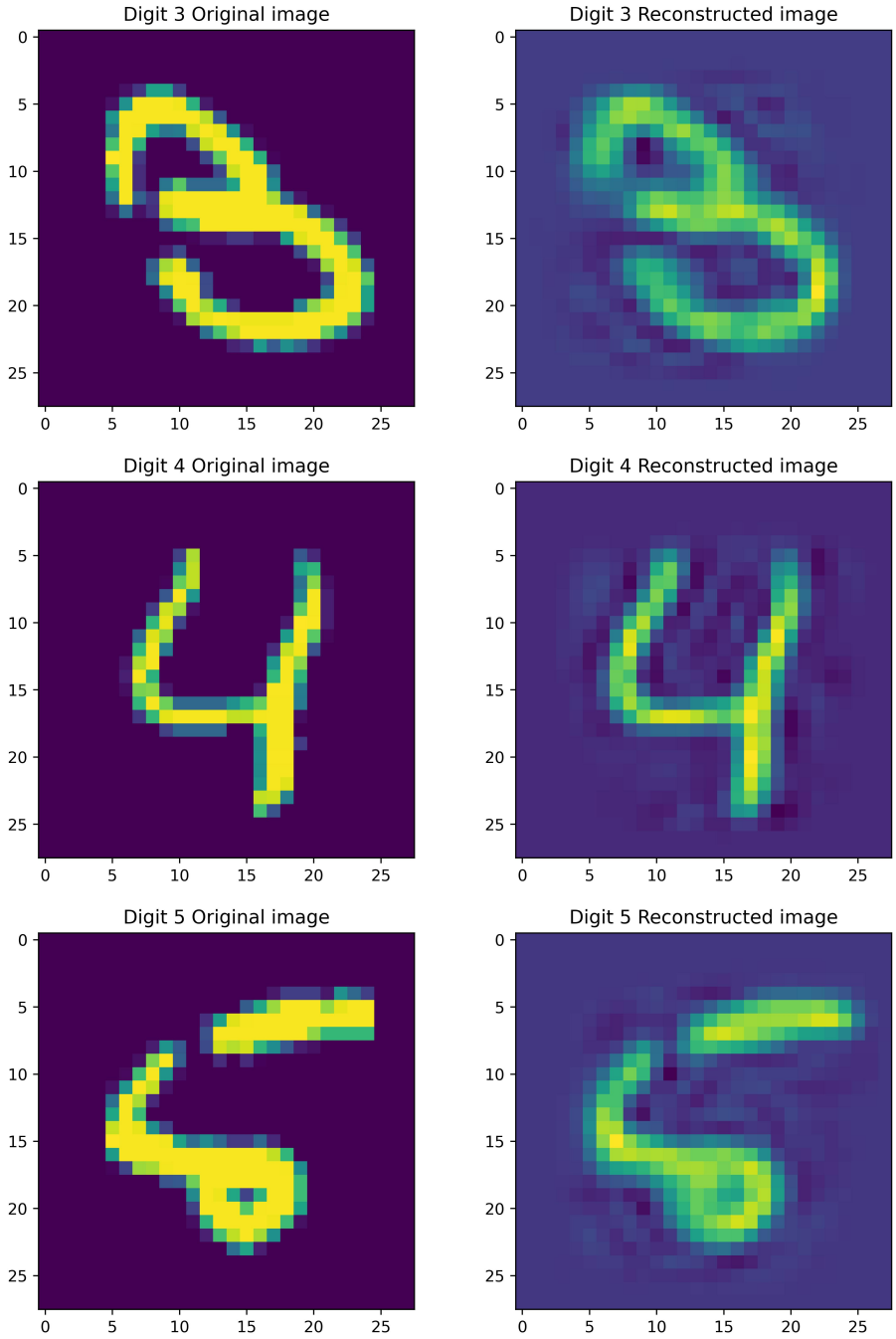

Figure 13: Original & Reconstructed images for digits 3-5

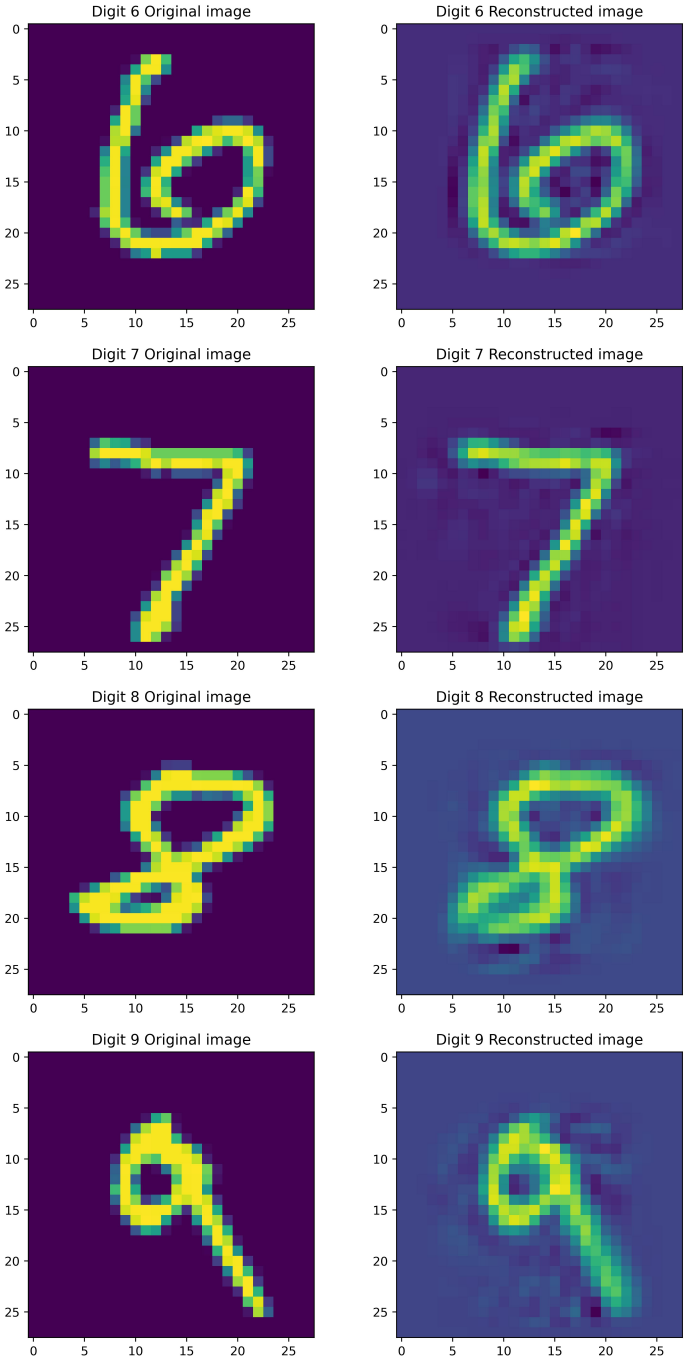

Figure 14: Original & Reconstructed images for digits 6-9

Problem 6

1. Pictorial representation of mean and principal eigenvectors, and graph showing top 10 eigenvalues:

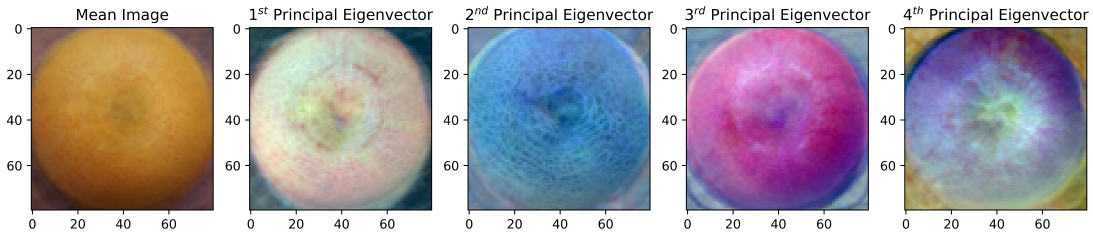

Figure 15: Mean & Top 4 Eigenvector Images

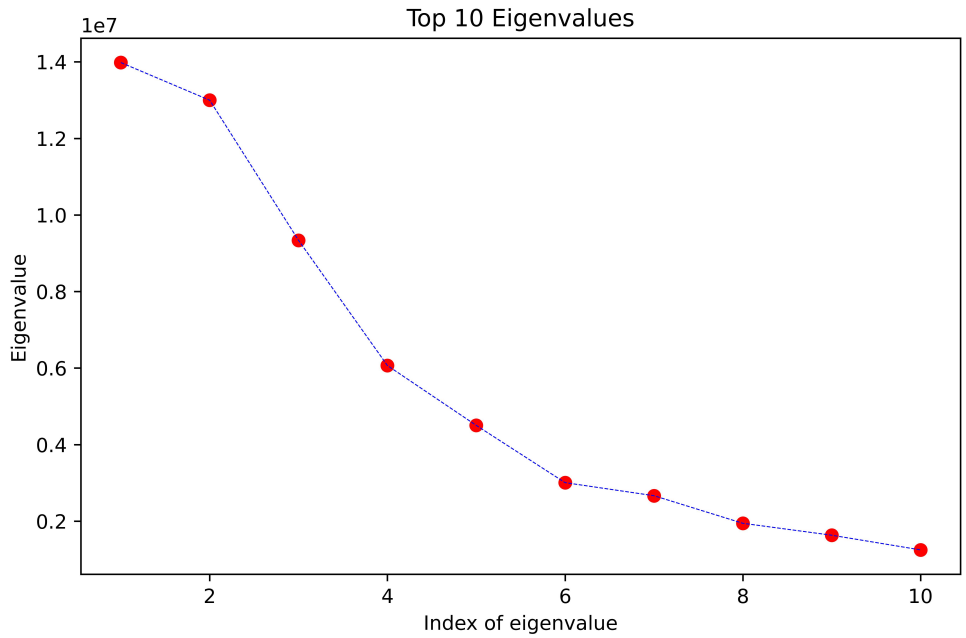

Figure 16: Top 10 Eigenvalues Graph

2. We know that every image can be represented as a linear combination of all eigenvectors (normalized) of the covariance matrix C (Spectral Theorem, also tells us that these will be orthogonal), call them $\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4, \ldots$ in decreasing order of eigenvalues. We want to find the closest representation of input image I in the vector space of $\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4$.
Let $\bar{v} = \bar{\mu} + c_1\bar{u}_1 + c_2\bar{u}_2 + c_3\bar{u}_3 + c_4\bar{u}_4 + \bar{x}$ represent the given image, where $\bar{\mu}$ is the mean image for that digit and $\bar{x}$ is a linear combination of the rest of the eigenvectors. We claim that $\bar{v}' = \bar{\mu} + c_1\bar{u}_1 + c_2\bar{u}_2 + c_3\bar{u}_3 + c_4\bar{u}_4$ is the "closest" to $\bar{v}$.

This is because $\bar{v} - \bar{v}' = \bar{x}$, and $\bar{x}$ is free from any components along $\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4$, so for any other vector $\bar{v}'' = \bar{\mu} + c_1'\bar{u}_1 + c_2'\bar{u}_2 + c_3'\bar{u}_3 + c_4'\bar{u}_4$ (here $||\bar{x}||$ represents Frobenius Norm of $\bar{x}$),

$$\begin{aligned} ||\bar{v} - \bar{v}''|| &= \sqrt{(c_1' - c_1)^2 + (c_2' - c_2)^2 + (c_3' - c_3)^2 + (c_4' - c_4)^2 + ||\bar{x}||^2} \\ &\geq ||\bar{x}|| \\ &\geq ||\bar{v} - \bar{v}'|| \end{aligned}$$

Hence, $\bar{v}'$ is the closest to $\bar{v}$. To find the coefficients c_1, c_2, c_3 & c_4 is the same as taking a projection of vector $\bar{v}$ along the relevant eigenvector (similar to Gram Schmidt Process since eigenvectors are orthogonal), i.e.

$$\bar{v}' = \bar{\mu} + ((\bar{v} - \bar{\mu}) \cdot \bar{u}_1)\bar{u}_1 + ((\bar{v} - \bar{\mu}) \cdot \bar{u}_2)\bar{u}_2 + ((\bar{v} - \bar{\mu}) \cdot \bar{u}_3)\bar{u}_3 + ((\bar{v} - \mu) \cdot \bar{u}_4)\bar{u}_4$$

This can be computed using a single matrix product expression.

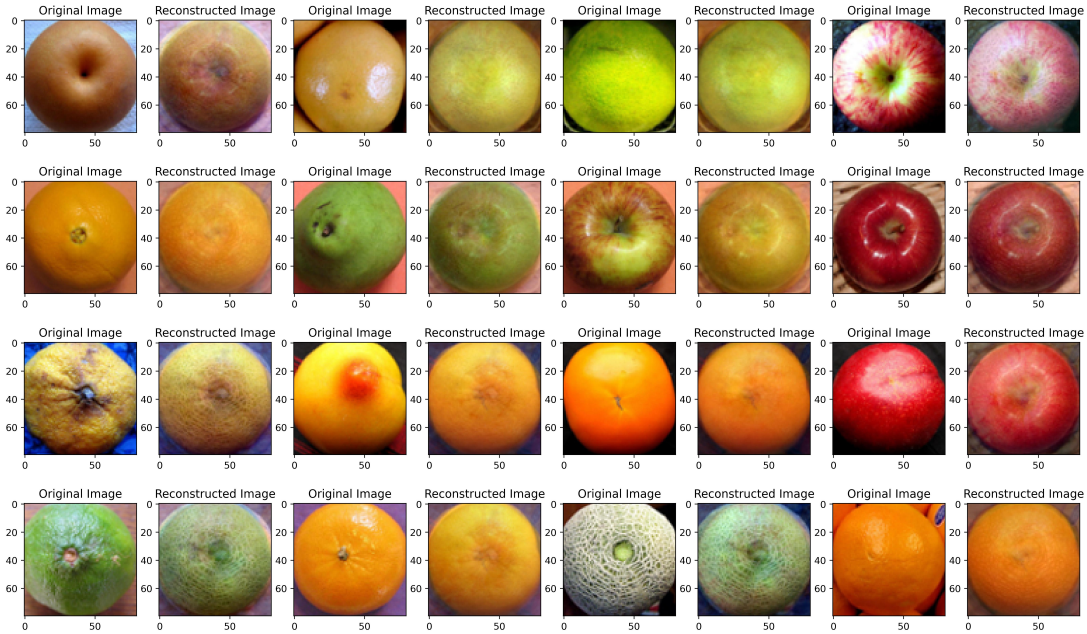

Figure 17: Original Fruit Images & their Closest Representation

3. We assume a multivariate gaussian distribution across all eigenvectors. This gives us a way to sample random variables by transformation of 4 dimensional vector of i.i.d uni-variate standard normal random variables. Let such a random variable be $W = [W_1, W_2, W_3, W_4]^T$.
   We scale each W_i with the square root of the corresponding eigenvalue (or just set the scale parameter as the square root of the eigenvalue since eigenvalue represent the variance along that eigenvector direction), and scale $\bar{u}_i$ by this factor. Adding the mean image vector to this gives us a sample of a new image of a fruit. Since, we are adding vectors randomly sampled within the scaled span of the 4 principal eigenvectors to the mean vector, the resultant sample varies along the 4 principal modes of variation from the mean image, i.e.

$$\bar{v}_{new} = W_1\sqrt{\lambda_1}\bar{u}_1 + W_2\sqrt{\lambda_2}\bar{u}_2 + W_3\sqrt{\lambda_3}\bar{u}_3 + W_4\sqrt{\lambda_4}\bar{u}_4$$

   Again this can be computed using a single matrix vector product expression.

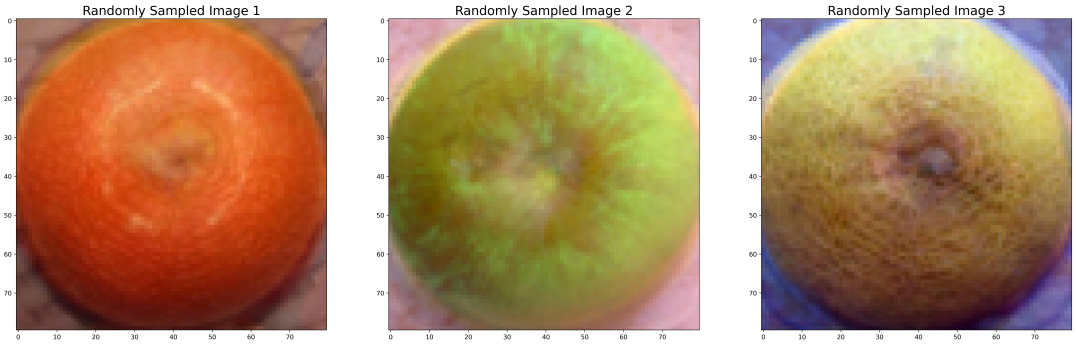

Figure 18: 3 Random Sample Images Representing New Generated Fruits