Assignment-2 REPORT

200050040-200050075

QUESTION-6

Contents

PCA for Another Image Dataset

1 Finding Mean, Covariance and Eigen vectors,values

Similiar to the previous question analysis, We have the data set of 16 fruits each of which is a **19200x1** column vector.So our total data set is represented by a single matrix **D** = **19200x16**, each column represents a fruit.

All the steps of finding mean,Covariance etc are Exactly same as **Q4**, Mean μ = average of all the 16 columns = (sum of all 16 columns)/16.So mean can be found easily and same as previous questions.Also the covariance matrix can be found from **D** and μ. If **S = D** - μ, then **C = S*S**^{T}**/N**.Finding them is same.

Now using **C** I found the top-4 Eigen **vectors** of **C** and the top-10 **Eigen values** of C.

Displaying the **mean** and the 4-**eigen vectors** as Images:

I reshaped those vectors to 80x80x3 and scaled to [0,1] by dividing with 255.Then used the image() function to visualise those images. Also I used imwrite() to save those images in the current directory.I used the subplot() to plot the 4 eigen-vector images in a single plot.

The obtained images are:

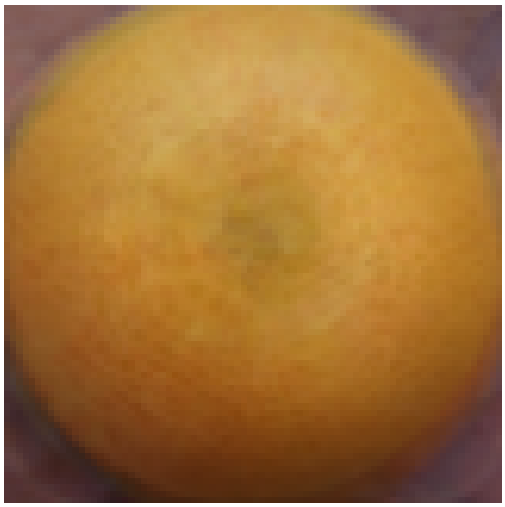


Figure 1: Mean of all the Fruits

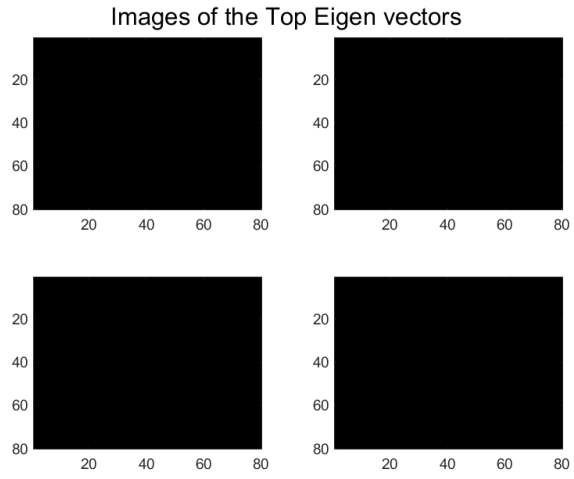


Figure 2: Top-4 eigen vectors as images

Plotting the Top-10 eigen values:

I found the top10 eigen values of **C** using the eigs(C) function.And plotted them. The obtained plot is:

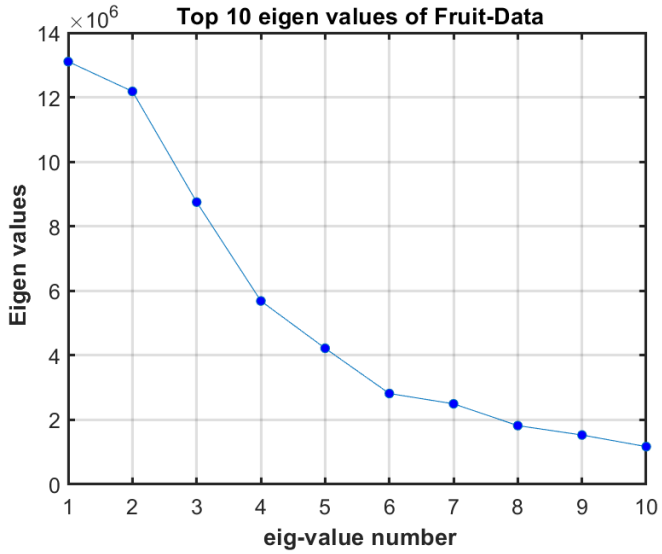


Figure 3: Top-10 eigen values

2 Closest Representations of the fruit images.

Let $\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, \mathbf{u_4}$ be the **unit** eigen vectors (The top-4 eigen unit vectors) and let μ be the mean.So now we need to find coefficients a_1, a_2, a_3, a_4 so that the vector **c** $= \mu + a_1u_1 + a_2u_2 + ... + a_4u_4$ is as close to the original Fruit data vector **x** as possible. Here we perform computations assuming all are Column vectors, i.e, we represent the images as 19200x1 column vectors.

For closeness, we need the norm ||**x-c**|| to be as small as possible.norm is nothing but square root of the sum of squares of the vector.So we need the sum of squares of components of **x-c** to be as small as possible.

$$\sum_{i=1}^{19200}(x_i - \mu_i - a_1u_{1i} + ... + a_4u_{4i})^2 \qquad \textbf{minimum}$$

Now if we expand the sum, then we get the following

$$||\mathbf{x}||^2 + ||\mu||^2 + \sum_{k=1}^{4}[(a_k)^2||\mathbf{u_k}||^2 - 2a_k(\mathbf{u_k}\cdot\mathbf{x})] + \sum_{i=1}^{4}\sum_{j=1\neq i}^{4} a_ia_j(\mathbf{u_i}\cdot\mathbf{u_j})$$

First 2 terms are constant and also the 4rd term is zero since $\mathbf{u_i}, \mathbf{u_j}$ are orthogonal eigen vectors.So we need the 3rd term to be minimum for all k = 1,4. Also $||u_k|| = 1$as we have choosen **unit** eigen vectors.

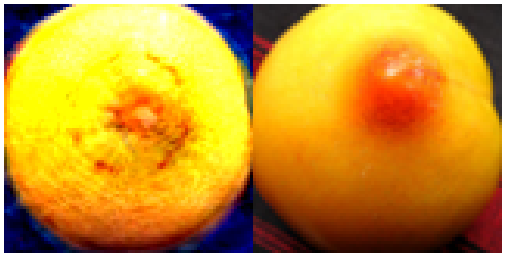
So it will be minimum if $a_k^2 - 2a_k(u_k.x)$ is minimum.It happens when $\mathbf{a_k} = \mathbf{u_k}\cdot\mathbf{x}$

Hence the closest vector to **x** is **c** $= \mu + \sum_{k=1}^{4}(\mathbf{u_k}.\mathbf{x})\mathbf{u_k}$

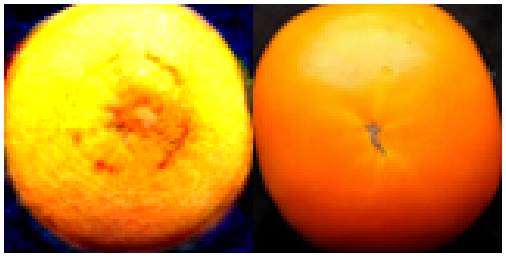
So I implemented the same for all images in a loop and Plotted the image of both **x** and **c** side-by-side for comparison using the image() function after rescaling and reshaping them to 80x80x3. I also saved those images in the directory using the imwrite() function.The images that I obtained are: **Original: RIGHT**

Fruit: 1

Fruit: 2



Fruit: 3

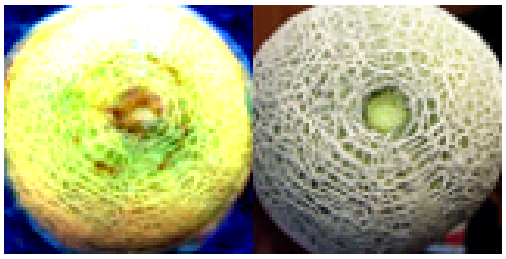


Fruit: 4

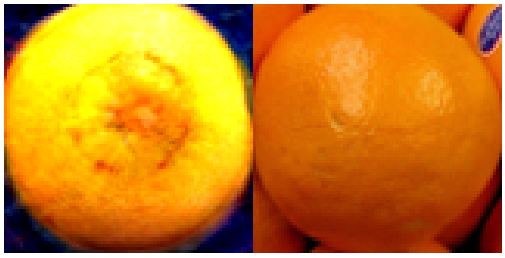
Fruit: 5

Fruit: 6

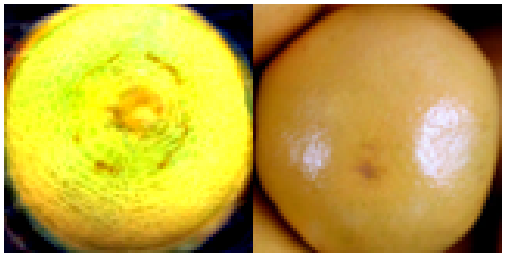
Fruit: 7



Fruit: 8



Fruit: 9



Fruit: 10

Fruit: 11

Fruit: 12

Fruit: 13

Fruit: 14

Fruit: 15

Fruit: 16

3 Sampling new fruit images.

So now we need to sample random images from the same distribution given by the μ and **C**. We have to use the 4 eigen vectors $\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, \mathbf{u_4}$ and the mean to sample new 'fruits'. So we have to generate vectors $\mathbf{x} = \mu + \mathbf{a_1 u_1} + \mathbf{a_2 u_2} + ... + \mathbf{a_4 u_4}$. where a_1, a_2, a_3, a_4 are the scalar real coefficients of the 4 **unit** eigen vectors.

So our job is essentially to generate the 4 coefficients randomly *based on the given fruit distribution data*.

Now when we project all our data(in the 19200 dimensions) on a line in the direction of $\mathbf{u_1}$, what we are capturing is the coefficient of $\mathbf{u_1} = a_1$ in all our data points.So when we project data vectors **x** along a **Mode of variation** i.e, along the eigen vectors, We just require the coefficient of $\mathbf{u_1}$ in **x** - μ which determines the *distance* of the projected point on the line from the mean μ.

Also when we caluculate the variance of the **projected** data along the Mode of variation determined by say $\mathbf{u_1}$, what we are finding is essentially the **variance of the component** $\mathbf{a_1}$ among our whole data points. And we know that variance along a direction is equal to the **eigen value** of the eigen-vector along that direction.

Hence for a **large data sample**, the coefficients vary as following:

Variance($\mathbf{a}_k$) $= \lambda_k(eigen\ value\ of\ \mathbf{u_k})$ and the Mean($\mathbf{a}_k$) $= 0$ for all k = 1,2,3,4 because a_k is the *displacement from the mean* hence it's average/mean = 0.

So now as we know the variance and means of the 4 coefficients, we try to **model** or **sample** them from their distributions. We only know the **Variance** and the **mean** of the coefficients we don't know any exact distribution, so let's **assume** that it is a **Gaussian** with that Variance and mean(= 0).

So we **sample** draws for $\mathbf{a_k}$ from a **Gaussian** with Variance = $\boldsymbol{\lambda_k}$ and mean = 0 for k = 1,2,3,4. So I used the randn() function to generate draws $\mathbf{w}_k$ from the standard Gaussian with variance 1. Then scaled those values obtained to get a Variance = $\lambda_\mathbf{k}$. i.e, multiplied $\mathbf{w_k}$ by $\sqrt{\lambda_\mathbf{k}}$ to get the coefficient $\mathbf{a_k}$, i.e $\mathbf{a_k} = \sqrt{\lambda_\mathbf{k}}\mathbf{w_k}$

And hence using the 4 coefficients generated , we sample a new image vector $\mathbf{x} = \mu + \mathbf{a_1u_1} + \mathbf{a_2u_2} + ... + \mathbf{a_4u_4}$.Then reshaped it to 80x80x3 and scaled it to [0,1] then viewed its image using the image() function.Also I used imwrite() to save the image in the folder.

The Sampled 3 Fruits are:

Sampled Fruit: 1

Sampled Fruit: 2

Sampled Fruit: 3

I have set the rng() to a fixed value.And if we change that, the program will generate new fruits

4 Code Running Instructions.

`Run the Q6_fruit.m file in 'code' folder to produce all the above images.`They can also be found in the results folder.

1. The Mean image fruit_mean.png
2. The Mean image Eigen_Values.png
3. The image of 4 Eigen vectors Eigvectors_img.png
4. Comparison images of each Fruit, "fruit_comparison_**N**.png" for N = 1 - 16
5. Newly sampled 3 Fruits , "generated_fruit_**N**.png" for N = 1 - 3

Also I kept the 16 Image data files **image_N.png**(taken as input) in the same code folder, Which are read by the program when run.

NOTE: The program is actually running for a long time,I tried to optimise it, but still it is taking like around 2-3 minutes.So please take care of that.