IITM-CS5691 : Pattern Recognition and Machine Learning — Release Date: October 9, 2023
Assignment II — Due Date : October 23, 2023, 23:59

Roll No : CS23M034 — Name: Kushagra Jain
Collaborators (if any): Ravindra narayan Bhidwe (CS23M054)
References/sources (if any):

- Use LaTeX to write-up your solutions (in the solution blocks of the source LaTeX file of this assignment), submit the resulting rollno.asst2.answers.pdf file at Crowdmark by the due date, and propery drag that pdf's answer pages to the corresponding question in Crowdmark (do this propery, otherwise we won't be able to grade!). (Note: **No late submissions** will be allowed, other than one-day late submission with 10% penalty or four-day late submission with 30% penalty.)
- Please upload to moodle a rollno.zip file containing three files: rollno.asst2.answers.pdf file mentioned above, and two code files for the programming question (rollno.ipynb file and rollno.py file). Do not forget to upload to Crowdmark your results/answers (including Jupyter notebook **with output**) for the programming question.
- Collaboration is encouraged, but all write-ups must be done individually and independently, and mention your collaborator(s) if any. Same rules apply for codes written for any programming assignments (i.e., write your own code; we will run plagiarism checks on codes).
- If you have referred a book or any other online material or LLMs (Large Language Models like ChatGPT) for obtaining a solution, please cite the source. Again don't copy the source *as is* - you may use the source to understand the solution, but write-up the solution in your own words (this also means that you cannot copy-paste the solution from LLMs!). Please be advised that *the lesser your reliance on online materials or LLMs* for answering the questions, *the more your understanding* of the concepts will be and *the more prepared you will be for the course exams*.
- Points will be awarded based on how clear, concise and rigorous your solutions are, and how correct your answer is. The weightage of this assignment is 12% towards the overall course grade.

1. (8 points) [SPECTRAL CLUSTERING - LAPLACIAN EIGENMAP] Consider a simple undirected graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges. Let A be the binary adjacency matrix of the graph (i.e., the symmetric 0-1 matrix where 1 indicates the presence of the corresponding edge; diagonal entries of A are zero). Let $x \in \mathbb{R}^n$ denote the node scores.

   Let the graph Laplacian matrix be $L = D - A$ seen in class, with D being the diagonal matrix of node degrees. Let $\lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_n$ denote the eigen values of the graph Laplacian L (sometimes also referred to as L_G to explicitly mention the graph).

   (a) (2 points) Show that $x^T L x = \sum_{(i,j)\in E}(x_i - x_j)^2$, and hence argue very briefly why $\lambda_i \geqslant 0$ for $i = 1, 2, \ldots, n$?

Solution:
Let's rewrite the equation in different form :

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}(x_i - x_j)^2$$

We include A_{ij} term to clear visualize the thing and it does not change the result because A_{ij} is only take value of 0 and 1.And we also include the $\frac{1}{2}$ for simplify our calculation and it is also not affect our result because it is just a constant.

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}x_i^2 + A_{ij}x_j^2 - 2A_{ij}x_ix_j$$

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}x_i^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}x_j^2 - \sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}x_ix_j$$

Know Solve them individually

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}x_i^2 = \frac{1}{2}\sum_{i=1}^{n}\left(A_{i1} + A_{i2} + ... + A_{iN}\right)x_i^2$$

$$\left(A_{i1} + A_{i2} + \ldots + A_{iN}\right) \text{ is equal to degree } d_i$$

So,

$$= \frac{1}{2}\sum_{i=1}^{n}d_ix_i^2$$

$$= \frac{1}{2}X^TDX \qquad (1)$$

Similarly,

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}x_j^2 = \frac{1}{2}X^TDX \qquad (2)$$

know the Last term ,

$$\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}x_ix_j = \sum_{i=1}^{n}x_i\sum_{j=1}^{n}A_{ij}x_j$$

$$= \sum_{i=1}^{n}x_i\left(A_{i1}x_1 + A_{i2}x_2 + ... + A_{iN}x_N\right)$$

$$= \sum_{i=1}^{n} x_i \left(A_{i1} + A_{i2} + ... + A_{iN}\right) [x_1 \;\; x_2 \;\; .... \;\; x_N]^T$$

$$= \sum_{i=1}^{n} x_i \left(A_{i1} + A_{i2} + ... + A_{iN}\right) X$$

$$= x_1[A_{11} \;\; A_{12} \;\; ... \;\; A_{1N}]X + x_2[A_{21} \;\; A_{22} \;\; ... \;\; A_{2N}]X + ... + x_N[A_{N1} \;\; A_{N2}... \;\; A_{NN}]X$$

$$= X^T A X \tag{3}$$

Putting all Equation we will get

$$= \frac{1}{2}X^T D X + \frac{1}{2}X^T D X - X^T A X$$

$$= X^T D X - X^T A X$$

$$= X^T [D - A] X$$

$$= X^T L X$$

Hence $x^T L x = \sum_{(i,j)\in E}(x_i - x_j)^2$ and also $x^T L x \geqslant 0$ because $\sum_{(i,j)\in E}(x_i - x_j)^2$ is always positive because we are taking the square of each index.
$x^T L x \geqslant 0$,So it satisfy the condition of semi positive definite matrix .Therefore L is **semi positive definite matrix**.
Now to find out optimum value of X who minimize this equation

$$= \arg\min_X \;\; X^T L X \;\; \text{where} \;\; X^T X = 1$$

Use Lagrange Multiplers to solve the system of equation

$$F(X, \lambda) = X^T L X - \lambda(X^T X - 1)$$

Partial differentiate with respect to X

$$\frac{dF(X, \lambda)}{dX} = 2LX - 2\lambda X$$

Now equating it to Zero to find the optimum value of X

$$2LX - 2\lambda X = 0$$

$$LX = \lambda X$$

So, X is the eigenvector and λ is the eigenvalue of L.

To prove it is a point of minima double differentiate with respect to X

$$\frac{d^2F(X,\lambda)}{dX^2} = 2L$$

We already prove L is a semi positive definate matrix So, $2L \geqslant 0$.Hence it is a point of minima.

We find out that L is a semi positive definite matrix then its eigenvalue will always be $\geqslant 0$, i.e $\lambda_i \geqslant 0 \quad \forall$ i=1,2,3,....

(b) (1 point) If G has 3 connected components, what is the multiplicity of the eigen value 0 of L_G, and what are the corresponding eigen vectors?

Solution: The Multiplicity of the eigen Value 0 of L_G is equal to Number of Connected Component.
If number of connected Component is k
Then,

$$\text{The multiplicity of eigenvalue 0 of } L_G = k$$

Here the number of Component is 3.
So,

$$\text{The multiplicity of eigenvalue 0 of } L_G = 3$$

And the corresponding eigenvectors are the **indicator vectors** i.e if the vertex in the same component then we mark 1 otherwise 0, of the connected components.

(c) (2 points) Let's add one edge to G to obtain a new graph G'. What can you say about the multiplicity of eigen value 0 of $L_{G'}$ relative to that of L_G? Will the sum of eigen values of $L_{G'}$ change compared to that of L_G; and if so, by what amount?

Solution: When we Add one edge to G to obtain a new graph G'. The new graph G' that will be formed will have same no. of component or different no. of component.

Case 1 : Number of Component remain same
When we add extra edge into the same component our adjacency matrix will change but the number of component remain the same. So, the Laplacian matrix

$$L = D - A$$

will also change and our new Laplacian matrix is $L_{G'}$.
The multiplicity of eigenvalue 0 of $L_{G'}$ is equal to number of component (K).
Here, the number of Component is K so The **multiplicity** of eigenvalue 0 of $L_{G'}$ is **same** as L_G which is equal to K.

Case 2: Number of Component changes
When we add extra edge between the different component our adjacency matrix will change and the number of component decreases by 1 .
If number of Component is K then by adding extra edge between the component the number of component is decrease by 1.
So, the graph G' will have number of component = $K - 1$.
The multiplicity of eigen value 0 of $L_{G'}$ is equal to number of component.
Here, the number of Component is $K - 1$ so The **multiplicity** of eigen value 0 of $L_{G'}$ is **different** and decrease by 1 as comparision to L_G which is equal to $K - 1$.

when we add extra edge the adjacency matrix A is changed.Suppose an extra edge is add between vertex V_1 and vertex V_2. then adjacency matrix will have one extra edge means two 1's is coming in the matrix because an edge is indicating in matrix by 1.
So, the sum of degree of all vertex in Graph G' is increase by 2.

$$\sum_{i=1}^{N} d_i' = \sum_{i=1}^{N} d_i + 2$$

The Laplacian matrix is Given by

$$L = D - A$$

Sum of EigenValue is equal to trace of Laplacian matrix.
The trace of Laplacian matrix is Equal to sum of degree of all vertex because the diagonal entry of Adjacency matrix A is equal to zero.
So, the Diagonal entry of Laplacian matrix will be d_1', d_2',..... d_N'.

$$\boxed{\text{Trace} = \sum_{i=1}^{N} d_i' = \sum_{i=1}^{N} d_i + 2}$$

(d) (3 points) If G is a complete graph on n nodes, we know that the multiplicity of eigen value 0 of L_G is 1; prove in this case that the multiplicity of eigen value n of L_G is $n-1$.
(Hint: Let v be an eigen vector of L_G orthogonal to the (all-ones) eigen vector of L corresp. to eigen value 0. Assume, without loss of generality, that $v(1) \neq 0$. Now compute the first coordinate of $L_G v$, and then divide by $v(1)$ to compute eigen value λ.)

Solution:

Consider a simple undirected graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges. Let A be the binary adjacency matrix of the graph (i.e., the symmetric $0-1$ matrix where 1 indicates the presence of the corresponding edge; diagonal entries of A are zero). Let $x \in \mathbb{R}^n$ denote the node scores. Let the graph Laplacian matrix be $L = D - A$, with D being the diagonal matrix of node degrees. Let $\lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_n$ denote the eigenvalues of the graph Laplacian L (sometimes also referred to as L_G to explicitly mention the graph).

If G is a complete graph on n nodes, we know that the multiplicity of eigenvalue 0 of L_G is 1. We will prove in this case that the multiplicity of eigenvalue n of L_G is $n-1$.

Proof:

1. **Laplacian Matrix for a Complete Graph:**

   For a complete graph on n nodes, the Laplacian matrix L_G has the following form:

$$L_G = \begin{bmatrix} n-1 & -1 & -1 & \ldots & -1 \\ -1 & n-1 & -1 & \ldots & -1 \\ -1 & -1 & n-1 & \ldots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \ldots & n-1 \end{bmatrix}$$

   In this matrix, each diagonal entry is $n-1$ (because each node is connected to $n-1$ other nodes), and all off-diagonal entries are -1.

2. **Find Eigenvectors and Eigenvalues:**

   Let v be an eigenvector of L_G orthogonal to the all-ones eigenvector corresponding to eigenvalue 0. Assume, without loss of generality, that $v(1) \neq 0$.

   We want to compute the first coordinate of $L_G v$, and then divide by $v(1)$ to compute the eigenvalue λ.

$$(L_G v)_1 = \sum_{i=1}^{n} L_{1i} v_i = (n-1)v(1) - v(2) - v(3) - \ldots - v(n)$$

   Dividing by $v(1)$, we get:

$$\lambda = n - 1 - \frac{v(2)}{v(1)} - \frac{v(3)}{v(1)} - \ldots - \frac{v(n)}{v(1)}$$

3. **Analyze λ:**

   Since v is an eigenvector, λ must be a constant. This implies that all the terms on the right-hand side must be equal. Since $v(1) \neq 0$, we have:

   $$\frac{v(2)}{v(1)} = \frac{v(3)}{v(1)} = \ldots = \frac{v(n)}{v(1)} = c$$

   Therefore, $v(2) = cv(1)$, $v(3) = cv(1)$, and so on.

4. **Conclude the Multiplicity:**

   We have shown that if v is an eigenvector of L_G orthogonal to the all-ones eigenvector corresponding to eigenvalue 0, then all non-zero entries of v must be equal. This means that there are $n - 1$ linearly independent eigenvectors corresponding to eigenvalue n, which establishes the multiplicity as $n - 1$.

2. (8 points) [PRINCIPAL COMPONENT ANALYSIS - NUMERICAL] Consider the following dataset D of 8 datapoints:

data #	x	y
1	5.51	5.35
2	20.82	24.03
3	-0.77	-0.57
4	19.30	19.38
5	14.24	12.77
6	9.74	9.68
7	11.59	12.06
8	-6.08	-5.22

You need to reduce the data into a single-dimension representation. You are given the first principal component: $\mathrm{PC1} = (-0.694, -0.720)$.

(a) (2 points) What is the xy coordinate for the datapoint reconstructed (approximated) from data #2 (x=20.82, y=24.03) using the first principal component of D? What is the reconstruction error of this PC1-based approximation of data #2?

Solution:

Given the dataset:

data #	x	y
1	5.51	5.35
2	20.82	24.03
3	−0.77	−0.57
4	19.30	19.38
5	14.24	12.77
6	9.74	9.68
7	11.59	12.06
8	−6.08	−5.22

The new data point that is reconstructed using the PC's is given by:

$$\tilde{X}_{new} = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

Here M=1,

$$\tilde{X}_{new} = \sum_{i=1}^{1} z_{ni} u_i + \sum_{i=2}^{2} b_i u_i$$

The reconstruction of the data point #2 (x=20.82, y=24.03) using the PC1 is:

$$z_{n1} = x^T u_1$$

$$z_{n1} = \begin{bmatrix} 20.82 & 24.03 \end{bmatrix} \cdot \begin{bmatrix} -0.694 \\ -0.720 \end{bmatrix} = -31.75068(\alpha_1)$$

$$z_{n1} u_1 = (-31.75068) \cdot \begin{bmatrix} -0.694 \\ -0.720 \end{bmatrix} = \begin{bmatrix} 22.035 \\ 22.860 \end{bmatrix}$$

$$b_2 = \tilde{x}^T u_2 \quad \text{where } \tilde{x} \text{ is mean}$$

$$b_2 = \begin{bmatrix} 9.294 & 9.685 \end{bmatrix} \cdot \begin{bmatrix} -0.720 \\ 0.694 \end{bmatrix} = 0.02971(\alpha_2)$$

$$b_2 u_2 = (0.02971) \cdot \begin{bmatrix} -0.720 \\ 0.694 \end{bmatrix} = \begin{bmatrix} -0.0214 \\ 0.0206 \end{bmatrix}$$

$$\tilde{X}_{new} = \begin{bmatrix} 22.035 \\ 22.860 \end{bmatrix} + \begin{bmatrix} -0.0214 \\ 0.0206 \end{bmatrix}$$

$$\boxed{\tilde{X}_{new} = \begin{bmatrix} 22.0136 \\ 22.8806 \end{bmatrix}}$$

So the xy coordinate for the data point reconstructed (approximated) from data #2 (x=20.82, y=24.03) using PC1 is x_{new} = **22.0136** and y_{new} = **22.8806**

$$X = x - \tilde{x}_{\text{new}} = \begin{bmatrix} 22.0136 \\ 22.8806 \end{bmatrix} - \begin{bmatrix} 20.82 \\ 24.03 \end{bmatrix} = \begin{bmatrix} 1.1936 \\ -1.1494 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1.1936 & -1.1494 \end{bmatrix} \begin{bmatrix} 1.1936 \\ -1.1494 \end{bmatrix} = 2.7456$$

Reconstruction error = Euclidean distance

Reconstruction Error = $\sqrt{2.7456}$

$$\boxed{\text{Reconstruction Error} = \mathbf{1.66}}$$

(b) (2 points) What is the second principal component of the dataset D? How will you represent data #2 as a linear combination of the two principal components? What is the reconstruction error of this $(PC1, PC2)$-based representation of data #2?

Solution:

Orthonormal vector are perpendicular to each other and their length is 1.
So, To find orthonormal vector to $PC1 = (-0.694, -0.720)$, we can use rotation matrix to clockwise 90-degree rotation,and we can find another orthonormal vector.

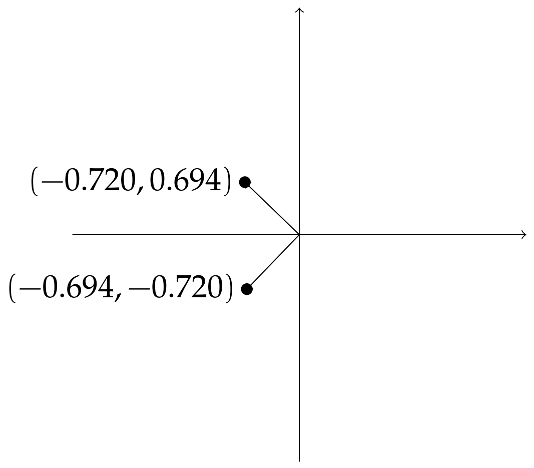
For a 90-degree clockwise rotation, we use the rotation matrix:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Let $v = (-0.694, -0.720)$. Applying the rotation:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} -0.694 \\ -0.720 \end{bmatrix} = \begin{bmatrix} -0.720 \\ 0.694 \end{bmatrix}$$

The vector $u = (-0.720, 0.694)$ is orthogonal to v and has a length of 1, making it an orthonormal vector.



To confirm that both the vector are orthonormal their dot product is equal to zero.

$$\mathbf{u} \cdot \mathbf{v} = (-0.720) \cdot (-0.694) + 0.694 \cdot (-0.720) = 0$$

So, **PC2** = (-0.720,0.694)
The new data point that is reconstructed using the PC's is given by:

$$\tilde{X}_{new} = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

Here M=2

$$\tilde{X}_{new} = \sum_{i=1}^{2} z_{ni} u_i$$

So,

$$z_{n1} = x^T u_1 = \begin{bmatrix} 20.82 & 24.03 \end{bmatrix} \cdot \begin{bmatrix} -0.694 \\ -0.720 \end{bmatrix} = -31.75068(\alpha_1)$$

$$z_{n1} u_1 = (-31.75068) \cdot \begin{bmatrix} -0.694 \\ -0.720 \end{bmatrix} = \begin{bmatrix} 22.035 \\ 22.860 \end{bmatrix}$$

$$z_{n2} = x^T u_2 = \begin{bmatrix} 20.82 & 24.03 \end{bmatrix} \cdot \begin{bmatrix} -0.720 \\ 0.694 \end{bmatrix} = 1.68642(\alpha_2)$$

$$z_{n2} u_2 = (1.68642) \cdot \begin{bmatrix} -0.720 \\ 0.694 \end{bmatrix} = \begin{bmatrix} -1.2142 \\ 1.1704 \end{bmatrix}$$

$$\tilde{X}_{new} = \begin{bmatrix} 22.035 \\ 22.860 \end{bmatrix} + \begin{bmatrix} -1.2142 \\ 1.1704 \end{bmatrix}$$

$$\tilde{X}_{new} = \begin{bmatrix} 20.82 \\ 24.03 \end{bmatrix}$$

So the xy coordinate for the data point reconstructed (approximated) from data #2 (x=20.82, y=24.03) using both PC's is x_{new} = **22.82** and y_{new} = **24.03**

$$X = x - \tilde{x}_{\text{new}} = \begin{bmatrix} 20.82 \\ 24.03 \end{bmatrix} - \begin{bmatrix} 20.82 \\ 24.03 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Reconstruction error = $X^T X$

$$\boxed{\text{Reconstruction error = } \mathbf{0}}$$

(c) (2 points) Let D' be the mean-subtracted version of D. What will be the first and second principal components $PC1$ and $PC2$ of D'? What is the xy coordinate of data #2 and its $PC1$-based reconstruction in D'? What is the associated reconstruction/approximation error of data #2?

Solution: The mean ($\bar{x}$) for x and ($\bar{y}$) for y is calculated as:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

where n is the number of data points.
For x:

$$\bar{x} = \frac{5.51 + 20.82 - 0.77 + 19.30 + 14.24 + 9.74 + 11.59 - 6.08}{8} = 9.294$$

For y:

$$\bar{y} = \frac{5.35 + 24.03 - 0.57 + 19.38 + 12.77 + 9.68 + 12.06 - 5.22}{8} = 9.685$$

The dataset D' is:

data #	$x - \tilde{x}$	$y - \tilde{y}$
1	−3.784	−4.335
2	11.526	14.345
3	−10.064	−10.255
4	10.006	9.695
5	4.946	3.085
6	0.446	−0.005
7	2.296	2.375
8	−15.374	−14.905

The Covariance matrix is given by :

$$\text{Covariance} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \tilde{x})(y_i - \tilde{y})$$

The Covariance matrix is same for mean subtracted data and for the standard data.
So the eigenvalues and eigenvectors of covariance matrix are also same which means PC1 and PC2 for mean subtracted data is same as PC1 and PC2 of standard data.
So, PC1 and PC2 for Mean subtracted data is :
PC1 $= (-0.694, -0.720)$.
PC2 $= (-0.720, 0.694)$.
The new data point that is reconstructed using the PC's is given by:

$$\tilde{X}_{new} = \sum_{i=1}^{M} z_{ni}u_i + \sum_{i=M+1}^{D} b_i u_i$$

Here M=1,

$$\tilde{X}_{new} = \sum_{i=1}^{1} z_{ni}u_i + \sum_{i=2}^{2} b_i u_i$$

For Mean subtracted data the term $\sum_{i=2}^{2} b_i u_i = 0$ because $b_i = (\tilde{x} - \tilde{x})u_i$ and it is equal to zero.
The reconstruction of the data point #2 (x=20.82, y=24.03) whose equivalent mean subtracted data is #2 (x=11.526, y=14.345) that is projected using the PC1 is:

$$z_{n1} = x^T u_1$$

$$z_{n1} = \begin{bmatrix} 11.526 & 14.345 \end{bmatrix} \cdot \begin{bmatrix} -0.694 \\ -0.720 \end{bmatrix} = -18.327444(\alpha_1)$$

$$z_{n1}u_1 = (-18.327444) \cdot \begin{bmatrix} -0.694 \\ -0.720 \end{bmatrix} = \begin{bmatrix} 12.71925 \\ 13.19576 \end{bmatrix}$$

$$\tilde{X}_{new} = \begin{bmatrix} 12.71925 \\ 13.19576 \end{bmatrix}$$

The Mean subtracted data projected on PC1 and its projected value is (x_{new} = 12.715, y_{new} = 13.216) .So the xy coordinate for the data point reconstructed (approximated) from data #2 (x = 20.82, y = 24.03) using PC1 is

$$\tilde{x}_{\text{new}} = 12.71925 + \text{mean} = 12.71925 + 9.294 = 22.0132$$

$$\tilde{y}_{\text{new}} = 13.19576 + \text{mean} = 13.19576 + 9.685 = 22.8807$$

So the xy coordinate for the data point reconstructed (approximated) from data #2 (x=20.82, y=24.03) using PC1 is x_{new} = **22.0136** and y_{new} = **22.8806**

$$X = x - \tilde{x}_{\text{new}} = \begin{bmatrix} 22.0136 \\ 22.8806 \end{bmatrix} - \begin{bmatrix} 20.82 \\ 24.03 \end{bmatrix} = \begin{bmatrix} 1.1936 \\ -1.1494 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1.1936 & -1.1494 \end{bmatrix} \begin{bmatrix} 1.1936 \\ -1.1494 \end{bmatrix} = 2.7456$$

Reconstruction error = Euclidean distance

Reconstruction Error = $\sqrt{2.7456}$

$$\boxed{\text{Reconstruction Error} = \mathbf{1.66}}$$

(d) (2 points) Let D'' be a dataset extended from D by adding a third feature z to each datapoint. It so happens that this third feature is a constant value (3.5) across all 8 datapoints. Then, what will be the three principal components of D'', and what is the xyz coordinate of the $PC1$-based reconstruction of data #2 in D'' and the associated reconstruction error?

Solution:
Given the dataset D″:

data #	x	y	z
1	5.51	5.35	3.5
2	20.82	24.03	3.5
3	−0.77	−0.57	3.5
4	19.30	19.38	3.5
5	14.24	12.77	3.5
6	9.74	9.68	3.5
7	11.59	12.06	3.5
8	−6.08	−5.22	3.5

The Mean of the dataset D″ is $\tilde{x} = 9.294, \tilde{y} = 9.685$ and $\tilde{z} = 3.5$.
So the mean centric data of D″ is :

data #	$x - \tilde{x}$	$y - \tilde{y}$	$z - \tilde{z}$
1	−3.784	−4.335	0.00
1	11.526	14.345	0.00
1	−10.064	−10.245	0.00
1	10.006	9.695	0.00
1	4.946	3.085	0.00
1	0.446	−0.005	0.00
1	2.446	2.375	0.00
1	−15.368	−14.905	0.00

The Covariance matrix S of the dataset D″ is calculated using the following formula :

$$\text{Covariance matrix} = \frac{1}{n} \sum_{n=1}^{N} (X - \tilde{X})^T (X - \tilde{X})$$

The covariance matrix calculated using the formula is given by :

$$\text{Covariance matrix (S)} = \begin{bmatrix} 76.85809844 & 78.97715625 & 0 \\ 78.97715625 & 82.630775 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The eigenvalue and its Corresponding eigenvector (Which is the principal Component) is :

$$\lambda_1 = 0.714, \quad PC_1 = \begin{bmatrix} -0.694 \\ -0.720 \\ 0 \end{bmatrix}$$

$$\lambda_2 = 158.774, \quad PC_2 = \begin{bmatrix} -0.720 \\ 0.694 \\ 0 \end{bmatrix}$$

$$\lambda_3 = 0, \quad PC_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The new data point that is reconstructed using the PC's is given by:

$$\tilde{X}_{new} = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

Here M=1,

$$\tilde{X}_{new} = \sum_{i=1}^{1} z_{ni} u_i + \sum_{i=2}^{3} b_i u_i$$

For Mean subtracted data the term $\sum_{i=2}^{3} b_i u_i = 0$ because $b_i = (\tilde{x} - \tilde{x})u_i$ and it is equal to zero.
The reconstruction of the data point #2 (x=20.82, y=24.03, z=3.5) whose equivalent mean subtracted data is #2 (x=11.526, y=14.345 , z=0) that is projected using the PC1 is:

$$z_{n1} = x^T u_1$$

$$z_{n1} = \begin{bmatrix} 11.526 & 14.345 & 0 \end{bmatrix} \cdot \begin{bmatrix} -0.694 \\ -0.720 \\ 0 \end{bmatrix} = -18.327444(\alpha_1)$$

$$z_{n1} u_1 = (-18.327444) \cdot \begin{bmatrix} -0.694 \\ -0.720 \\ 0 \end{bmatrix} = \begin{bmatrix} 12.71925 \\ 13.19576 \\ 0 \end{bmatrix}$$

$$\tilde{X}_{new} = \begin{bmatrix} 12.71925 \\ 13.19576 \\ 0 \end{bmatrix}$$

The Mean subtracted data projected on PC1 and its projected value is (x_{new} = 12.715, y_{new} = 13.216 , z_{new} = 0) .So the xy coordinate for the data point reconstructed (approximated) from data #2 (x = 20.82, y = 24.03) using PC1 is

$$\tilde{x}_{\text{new}} = 12.71925 + \text{mean} = 12.71925 + 9.294 = 22.0132$$

$$\tilde{y}_{\text{new}} = 13.19576 + \text{mean} = 13.19576 + 9.685 = 22.8807$$

$$\tilde{z}_{\text{new}} = 13.19576 + \text{mean} = 0 + 3.5 = 3.5$$

So the xy coordinate for the data point reconstructed (approximated) from data #2 (x=20.82, y=24.03 , z=3.5) using PC1 is x_{new} = **22.0136** , y_{new} = **22.8806** and z_{new} = **3.5**

$$X = x - \tilde{x}_{\text{new}} = \begin{bmatrix} 22.0136 \\ 22.8806 \\ 3.5 \end{bmatrix} - \begin{bmatrix} 20.82 \\ 24.03 \\ 3.5 \end{bmatrix} = \begin{bmatrix} 1.1936 \\ -1.1494 \\ 0 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1.1936 & -1.1494 & 0 \end{bmatrix} \begin{bmatrix} 1.1936 \\ -1.1494 \\ 0 \end{bmatrix} = 2.7456$$

Reconstruction error = Euclidean distance

Reconstruction Error = $\sqrt{2.7456}$

$$\boxed{\text{Reconstruction Error = } \mathbf{1.66}}$$

3. (8 points) [LINEAR REGRESSION]

(a) (4 points) The error function in the case of ridge regression is given by:

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2 + \frac{\lambda}{2}w^Tw$$

Show that this error function is convex and is minimized by:

$$w^* = (\lambda I + \phi^T\phi)^{-1}\phi^T t$$

Also show that $(\lambda I + \phi^T\phi)$ is invertible for any $\lambda > 0$.

(Note 1: To simplify and keep your solution concise, use vector/matrix format (e.g., gradient, Hessian, etc.,) for your expressions.

Note 2: Here, the target vector $t \in \mathbb{R}^N$ and the matrix $\phi \in \mathbb{R}^{N\times d'}$ represents all the N input datapoints after transformation by the feature-mapping function $\phi(.) : \mathbb{R}^d \to \mathbb{R}^{d'}$. For example, the $\phi(.)$ for performing k-degree polynomial regression on a d-dimensional input for $k = 2, d = 2$ is given by $\phi([x_1, x_2]) = [1, x_1, x_2, x_1^2, x_2^2, x_1x_2]$.)

Solution: The Error is represent by :

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T\phi(x_n))^2 + \frac{\lambda}{2}w^T w$$

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}(w\phi(x_n) - t_n)^T(w\phi(x_n) - t_n) + \frac{\lambda}{2}w^T w$$

know consider,

$$E_d(w) = (w\phi - t)^T(w\phi - t)$$
$$E_w(w) = w^T w$$

We have to find the value of w which minimize the $\tilde{E}(w)$. This problem is basically equivalent to minimize $E(d)$ such that $E(w)$ is $\leqslant$ some constant C.
Use the Lagrange Multiplier

$$F(w, \lambda) = \frac{1}{2}E(d) + \frac{\lambda}{2}E(w)$$

We take $\frac{1}{2}$ to simplify our calculation.

$$F(w, \lambda) = \frac{1}{2}(w\phi - t)^T(w\phi - t) + \frac{\lambda}{2}w^T w$$

To find the optimal **w** that minimizes $F(\mathbf{w}, \lambda)$, we differentiate it with respect to **w** and set the derivative to zero.

$$\frac{\partial F}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2}\left((w\phi - t)^T(w\phi - t) + \lambda w^T w\right)\right)$$

Using the chain rule and the properties of matrix derivatives, we get:

$$\frac{\partial F}{\partial w} = \phi^T(w\phi - t) + \lambda w$$

Setting this derivative to zero:

$$\phi^T(w\phi - t) + \lambda w = 0$$
$$w(\phi^T\phi + \lambda I) - \phi^T t = 0$$
$$w(\phi^T\phi + \lambda I) = \phi^T t$$

$$w^* = (\phi^T\phi + \lambda I)^{-1}\phi^T t$$

know to prove it is a convex function we have to double differentiate with respect to w

$$\frac{\partial^2 F}{\partial w^2} = \phi^T\phi + \lambda I$$

The $\phi^T\phi$ is a positive semi definite matrix because it is special matrix that is called gram matrix and $\lambda > 0$ which is given so $\phi^T\phi + \lambda I$ is a positive quantity hence it is a convex function. So, $w = (\phi^T\phi + \lambda I)^{-1}\phi^T t$ is a point of minima.

To show that $\phi^T\phi + \lambda I$ is invertible i.e their is a no non-zero vector x for which $(\phi^T\phi + \lambda I)x$ = 0 ,means the equation has only the trivial solution x = 0.
Proof : Assume for contradiction that $(\phi^T\phi + \lambda I)x$ = 0 has a nontrivial solution x $\neq$ 0 .

$$\phi^T\phi x + \lambda I x = 0$$

$$\phi^T\phi x + \lambda x = 0$$

$$x = -\frac{\phi^T\phi x}{\lambda}$$

Now, let's take the dot product of both sides with x^T .

$$x^T x = -\frac{x^T\phi^T\phi x}{\lambda}$$

since x $\neq$ 0 , $x^T x \neq= 0$ as well.Therefore we can divide both side by $x^T x$:

$$1 = -\frac{x^T\phi^T\phi x}{\lambda(x^T x)}$$

$\phi^T\phi$ is a semi positive definite matrix and $x^t x$ is positive quantity because it is a norm of x and $x^T\phi^T\phi x$ is also positive because $\phi^T\phi$ is positive semi definite matrix.
since $\lambda > 0$ this implies $\frac{x^T\phi^T\phi x}{(x^T x)} < 0$, which is a contradiction. The left hand side is a positive number ,but the right- hand side it is a negative number which cannot happen.
Hence our Assumption that x is nonzero nust be incorrect. Therefore, the only solution to $(\phi^T\phi + \lambda I)x$ = 0 is the trivial solution x = 0, which means that $\phi^T\phi + \lambda I$ is invertible.

(b) (4 points) Given a dataset

$$\mathsf{X} = \begin{bmatrix} -2 & 4 \\ -1 & 2 \end{bmatrix} \mathsf{t} = \begin{bmatrix} 3 & -1 \end{bmatrix}$$

find all minimizers of w of $\mathsf{E}(w) = \frac{1}{2}||\mathsf{X}w - \mathsf{t}||^2$, and indicate the one with the smallest norm. How does your answer change if you are looking for minimizers of $\tilde{\mathsf{E}}(w)$ instead (assuming λ = 1)?

Solution: The least square error is given by :

$$E(w) = \frac{1}{2}||Xw - t||^2$$

To minimize the error the optimum value of w is :

$$(X^T X)w = X^T t \qquad --- (1)$$

$$X^T X = \begin{bmatrix} -2 & -1 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} -2 & 4 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 5 & -10 \\ -10 & 20 \end{bmatrix}$$

$$|X^T X| = |5 \cdot 20 - (-10) \cdot (-10)| = |100 - 100| = 0$$

$|X^T X|$ is equal to 0 so inverse is not possible, So for w infinite many solution is possible because we find the projection of t which lie in the column space of X so only unique or infinite many solution is possible.

$$X^T t = \begin{bmatrix} -2 & -1 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} -5 \\ 10 \end{bmatrix}$$

putting the value of $X^T t$ and $X^T X$ in equation (1)

$$\begin{bmatrix} 5 & -10 \\ -10 & 20 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -5 \\ 10 \end{bmatrix}$$

By solving the System of linear equation we get :

$$w_0 = 2 * w_1 - 1$$

The Norm of matrix w is given by :

$$F = w_0^2 + w_1^2$$

So,

$$F(w1) = (2 * w_1 - 1)^2 + w_1^2$$

$$F(w1) = 5w_1^2 - 4w_1 + 1$$

To find out the optimum value of w_1 that minimise the norm ,we differentiate with respect to W_1 and set the derivative to zero.

$$F'(w_1) = 10w_1 - 4$$

$$10w_1 - 4 = 0$$

$$w_1 = 4/10 = 0.4$$

To prove it is a convex funtion or it is a point of minima we have to double differentiate with respect to w

$$F''(w_1) = 10$$

So, $F''(w_1) > 0$ so it is a point of minima.
Calculate the value of w_0 by putting the value of w_1 in ($w_0 = 2 * w_1 - 1$)

$$w_0 = -0.2$$

So the optimum value of w of E(w) who has smallest norm :

$$\boxed{w = \begin{bmatrix} -0.2 \\ 0.4 \end{bmatrix}}$$

The Regularized least squae error is given by:

$$\tilde{E}(w) = \frac{1}{2}||Xw - t||^2 + \frac{\lambda}{2} w^t w$$

To minimize the error, the optimum value of w is :

$$w = (X^T X + \lambda I)^{-1} X^T t = (X^T X + I)^{-1} X^T t \qquad , \lambda = 1 \text{ (given)}$$

$$X^T X = \begin{bmatrix} 5 & -10 \\ -10 & 20 \end{bmatrix}$$

$$X^T X + I = \begin{bmatrix} 6 & -10 \\ -10 & 21 \end{bmatrix}$$

$$(X^T X + I)^{-1} = \frac{1}{26} \begin{bmatrix} 21 & 10 \\ 10 & 6 \end{bmatrix}$$

.

$$X^T t = \begin{bmatrix} -5 \\ 10 \end{bmatrix}$$

$$w = \frac{1}{26} \begin{bmatrix} 21 & 10 \\ 10 & 6 \end{bmatrix} \begin{bmatrix} -5 \\ 10 \end{bmatrix} = \frac{1}{26} \begin{bmatrix} -5 \\ 10 \end{bmatrix}$$

So the optimum value of w who minimise the $\tilde{E}$(w) :

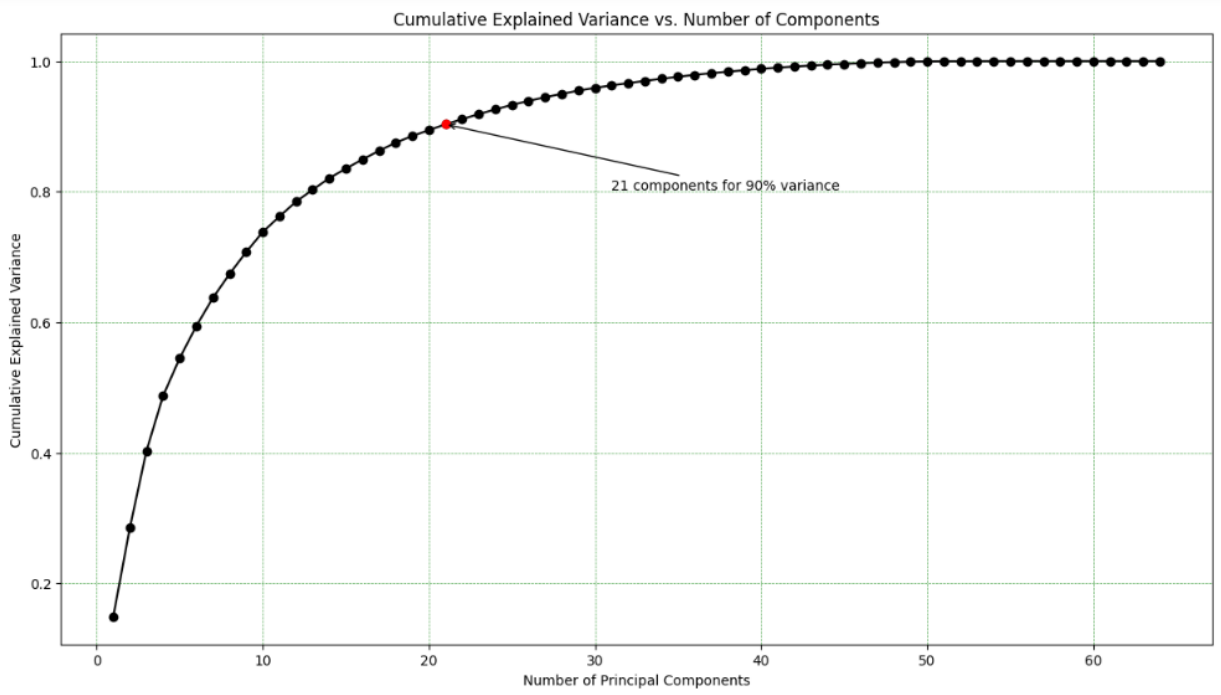
$$\boxed{w = \frac{1}{26} \begin{bmatrix} -5 \\ 10 \end{bmatrix}}$$

4. (8 points) [Life in lower dimensions...] You are provided with a dataset of 1797 images in a folder here - each image is 8x8 pixels and provided as a feature vector of length 64. You will try your hands at transforming this dataset to a lower-dimensional space, and clustering the images in this reduced space.

   Please use the template.ipynb file in the same folder to prepare your solution. Provide your results/answers in the pdf file you upload to Crowdmark, and submit your code separately in this moodle link. The code submitted should be a rollno.zip file containing two files: rollno.ipynb file (including your code as well as the exact same results/plots uploaded to Crowdmark) and the associated rollno.py file.

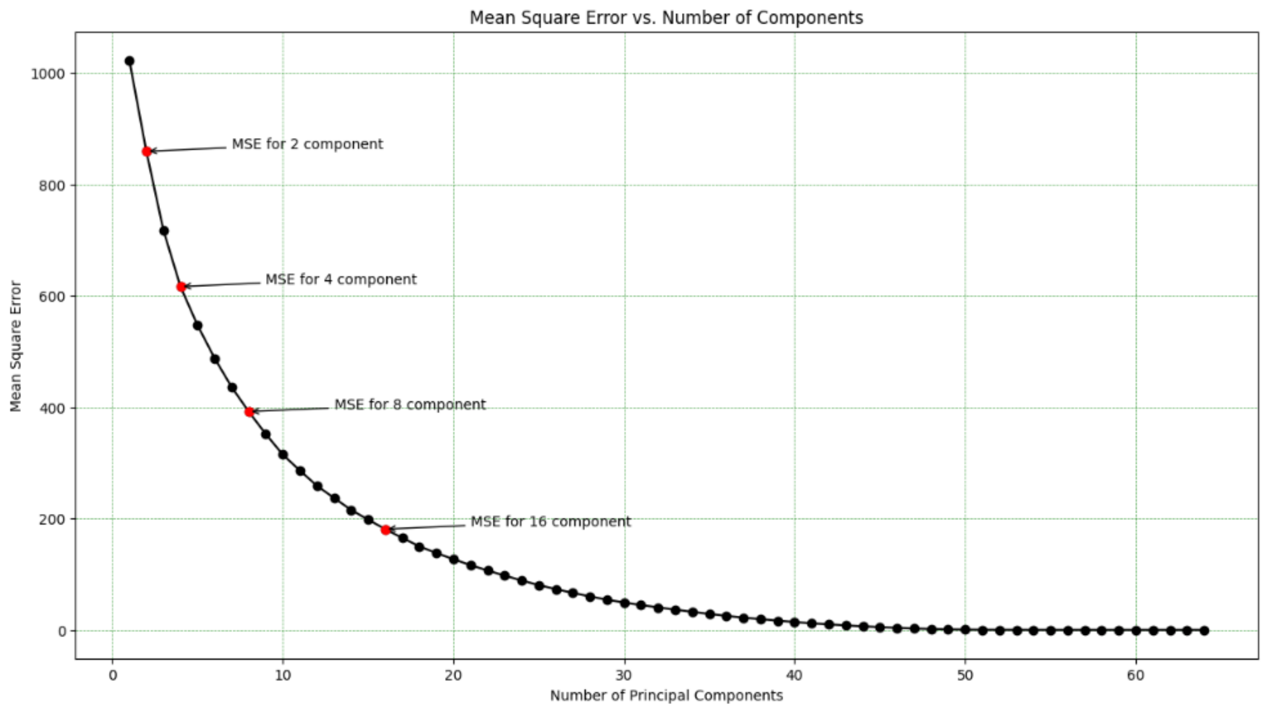
   Write the code from scratch for both PCA and clustering. The only exception is the computation of eigenvalues and eigenvectors for which you could use the numpy in-bulit function.

   (a) (4 points) Run the PCA algorithm on the given dataset. Plot the cumulative percentage variance explained by the principal components. Report the number of principal components that contribute to 90% of the variance in the dataset.



```
The number of principal components that contribute to 90% of the variance is: 21
```

(b) (4 points) Perform reconstruction of data using the small number of components: [2,4,8,16]. Report the Mean Square Error (MSE) between the original data and reconstructed data, and interpret the optimal dimension $\widehat{d}$ based on the MSE values.



```
MSE for d=2: 858.9447808487271
MSE for d=4: 616.191130056277
MSE for d=8: 391.7947361149742
MSE for d=16: 180.93970325737757
The optimal dimension is 16 based on MSE values.
```

4. (8 points) [LIFE IN LOWER DIMENSIONS...] You are provided with a dataset of 1797 images in a folder here - each image is 8x8 pixels and provided as a feature vector of length 64. You will try your hands at transforming this dataset to a lower-dimensional space, and clustering the images in this reduced space. Please use the template.ipynb file in the same folder to prepare your solution. Provide your re- sults/answers in the pdf file you upload to Crowdmark, and submit your code separately in this moodle link. The code submitted should be a rollno.zip file containing two files: rollno.ipynb file (including your code as well as the exact same results/plots uploaded to Crowdmark) and the associated rollno.py file. Write the code from scratch for both PCA and clustering. The only exception is the computation of eigenvalues and eigenvectors for which you could use the numpy in-bulit function. (a) (4 points) Run the PCA algorithm on the given dataset. Plot the cumulative percentage vari- ance explained by the principal components. Report the number of principal components that contribute to 90% of the variance in the dataset. (b) (4 points) Perform reconstruction of data using the small number of components: [2,4,8,16]. Report the Mean Square Error (MSE) between the original data and reconstructed data, and interpret the optimal dimensionˆd based on the MSE values.

Import required Header File

In [2]:
```python
import numpy as np
import pandas as pd
from numpy import load
from matplotlib import pyplot as plt
```

Loading of Dataset

In [3]:
```python
data = load('Data.npz')
keys = data.keys()
images = data['arr_0']
#print(images.shape)
```

Printing Some dataset value

In [4]:
```python

num_images_to_plot = 2

# Loop through and plot the images
for i in range(num_images_to_plot):
    # Reshape the flat vector to 8x8
    image = images[i].reshape(8, 8)

    plt.figure(figsize=(3, 3))  # Adjust the figsize as needed
    plt.imshow(image, cmap='gray')  # Assuming the images are grayscale
    plt.title(f'Image {i+1}')
    plt.axis('off')
    plt.show()

```

Image 1

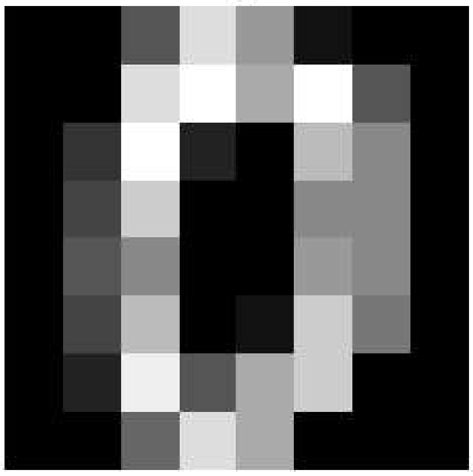
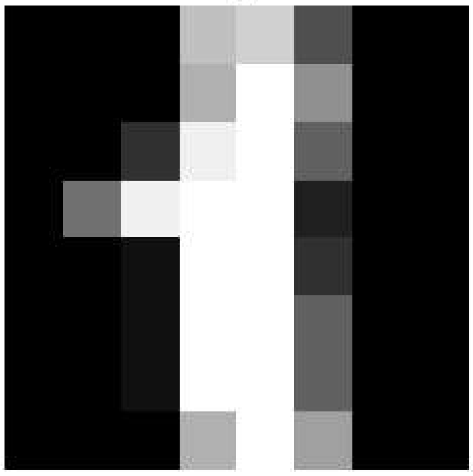


Image 2



Function for Mean, Covariance and PCA

In [5]:

```python
def Mean(data):
    data_point = data.shape[0]
    features = data.shape[1]

    mean =np.zeros(features)

    for i in range(features):
        sum=0
        for j in range(data_point):
            sum=sum+data[j][i]
        mean[i]=sum/data_point

    return mean


def covariance_matrix(data):

    data_point = data.shape[0]
    features = data.shape[1]
    mean=Mean(data)
    covariance=np.zeros((features,features))
    for i in range(features):
        for j in range(features):
            for k in range(data_point):
                covariance[i][j] += (data[k][i] - mean[i]) * (data[k][
            covariance[i][j]/=(data_point-1)

    return covariance




def PCA(data):

    mean=Mean(data);
    data_mean_subtracted= data-mean

    covariance = covariance_matrix(data_mean_subtracted)


    eigenvalues , eigenvectors = np.linalg.eigh(covariance)

    sorted_index = np.argsort(eigenvalues)[::-1]
    sorted_eigenvalue = eigenvalues[sorted_index]
    sorted_eigenvectors = eigenvectors[:,sorted_index]

    explained_variance_ratio=sorted_eigenvalue/ np.sum(sorted_eigenvalu


    sum=0 # contains cumalative variance
    PCs=0 # Contains Number of PC which having cumalative variance is m
    for i in explained_variance_ratio:
        sum+=i
        PCs+=1
        if(sum>=0.90):
            break


    return covariance,explained_variance_ratio,PCs


```

Calculating the covariance , variance ratio and PCA

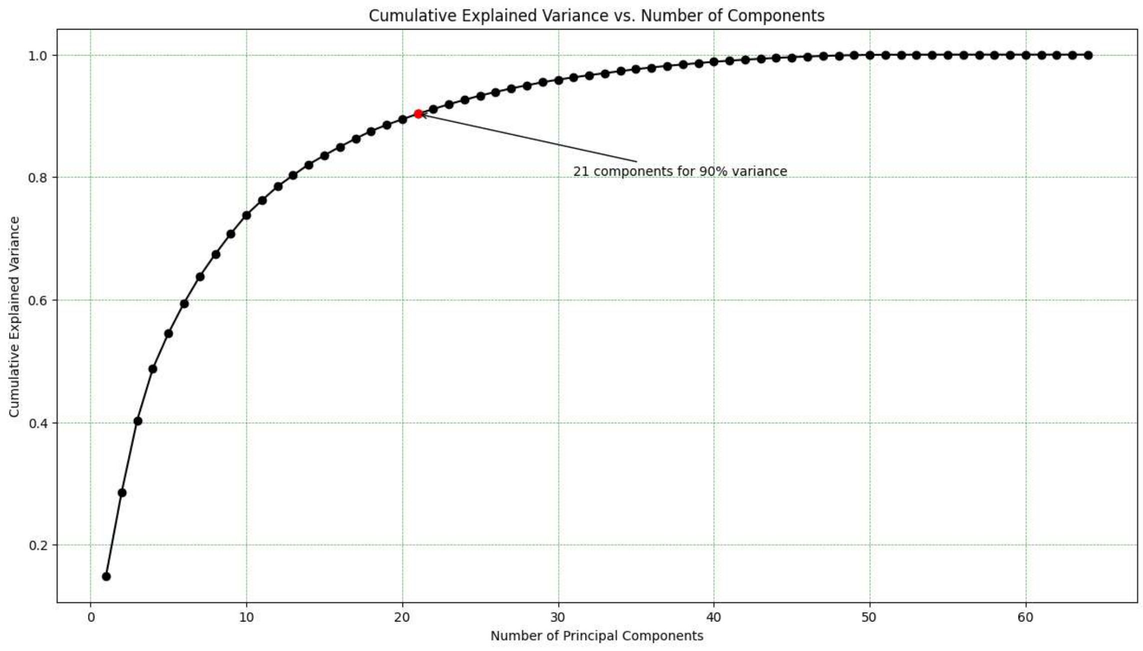
In [6]:

```
covariance,explained_variance_ratio , PCs =  PCA(images)
```

Solution of Part (a)

In [7]:

```
#Solution 1

cumulative_explained_variance=[]
sum=0
for i in explained_variance_ratio:
    sum=sum+i
    cumulative_explained_variance.append(sum)

# Plot cumulative explained variance
component_numbers = range(1, len(cumulative_explained_variance) + 1)
plt.figure(figsize=(15,8))
plt.plot(component_numbers,cumulative_explained_variance,marker='o', li
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Cumulative Explained Variance vs. Number of Components')


plt.plot(PCs, cumulative_explained_variance[PCs-1], 'ro')
plt.annotate(f'{PCs} components for 90% variance',
             xy=(PCs, cumulative_explained_variance[PCs-1]),
             xytext=(PCs + 10, cumulative_explained_variance[PCs-1] - 0
             arrowprops=dict(arrowstyle='->'))
plt.grid(color='green', linestyle='--', linewidth=0.5, alpha=0.7)
plt.show()


print(f"The number of principal components that contribute to 90% of th
```

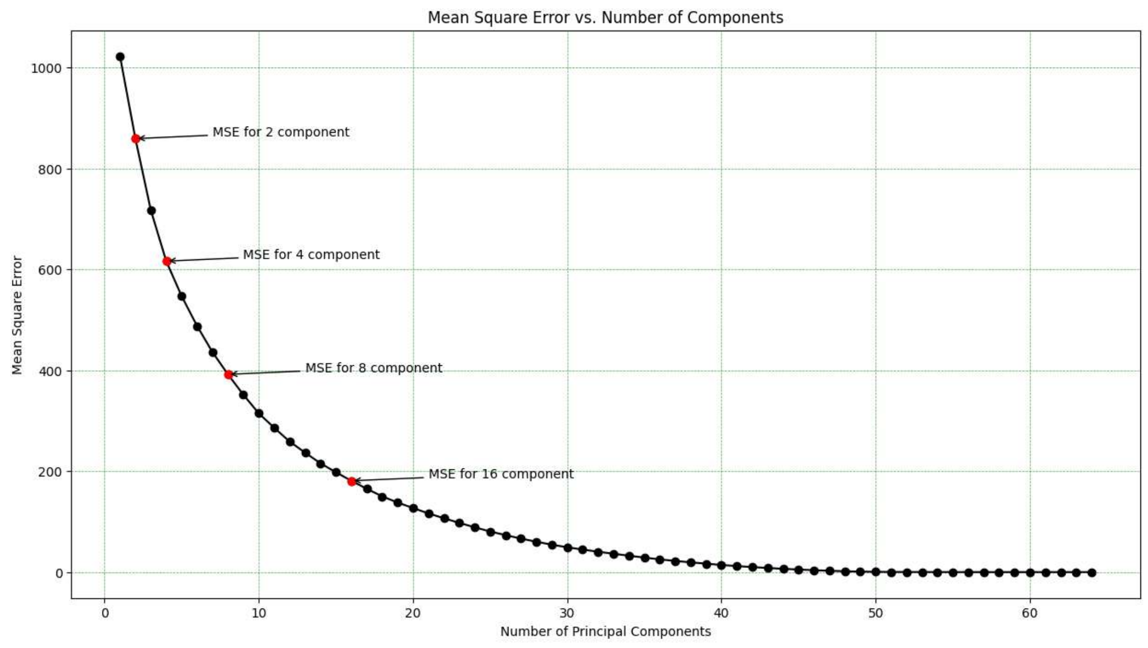


```
The number of principal components that contribute to 90% of the variance
is: 21
```

Solution of Part (b)

In [9]:

```python

def reconstruct_data(data_mean_subtracted, num_components,sorted_eigenv

    M_Eigenvectors=sorted_eigenvectors[:,:num_components]
    reconstructed_data=np.dot(np.dot(data_mean_subtracted, M_Eigenvecto
    reconstructed_data=reconstructed_data+mean

    return reconstructed_data







mean=Mean(images);
data_mean_subtracted= images-mean

#covariance = covariance_matrix(data_mean_subtracted)
eigenvalues , eigenvectors = np.linalg.eigh(covariance)


sorted_index = np.argsort(eigenvalues)[::-1]
sorted_eigenvalue = eigenvalues[sorted_index]

sorted_eigenvectors = eigenvectors[:,sorted_index]
#print(sorted_eigenvectors)

num_components_list = [2, 4, 8, 16]
MSE_values = []
x=[]


for num_components in num_components_list:
    reconstructed_data = reconstruct_data(data_mean_subtracted, num_com
    y=(images - reconstructed_data) ** 2
    a=y.shape[0]
    b=y.shape[1]
    sum=0
    for i in range(a):
        for j in range(b):
            y[i][j]=y[i][j]/(a)
            sum+=y[i][j]


    MSE=sum
    MSE_values.append(MSE)

sum=0
for num_components in range(1,65):
    reconstructed_data = reconstruct_data(data_mean_subtracted, num_com
    y=(images - reconstructed_data) ** 2
    a=y.shape[0]
    b=y.shape[1]
    sum=0
    for i in range(a):
        for j in range(b):
            y[i][j]=y[i][j]/(a)
            sum+=y[i][j]
    MSE=sum
    #print(num_components," ",sum)
    x.append(MSE)


component_numbers = range(1, len(cumulative_explained_variance) + 1)
plt.figure(figsize=(15,8))
plt.plot(component_numbers,x,marker='o', linestyle='-', color='black')
plt.xlabel('Number of Principal Components')
plt.ylabel('Mean Square Error')
plt.title('Mean Square Error vs. Number of Components')

for num_components in num_components_list:
    plt.plot(num_components, x[num_components-1], 'ro')
    plt.annotate(f'MSE for {num_components} component',
    xy=(num_components, x[num_components-1]),
    xytext=(num_components+5 , x[num_components-1] + 5.5),
    arrowprops=dict(arrowstyle='->'))


plt.grid(color='green', linestyle='--', linewidth=0.5, alpha=0.7)
plt.show()

for i, j in enumerate(num_components_list):
    print(f'MSE for d={j}: {MSE_values[i]}')

optimal_dimension = num_components_list[np.argmin(MSE_values)]
print(f"The optimal dimension is {optimal_dimension} based on MSE value
```



```
MSE for d=2: 858.9447808487271
MSE for d=4: 616.191130056277
MSE for d=8: 391.7947361149742
MSE for d=16: 180.93970325737757
The optimal dimension is 16 based on MSE values.
```