IITM-CS5691 : Pattern Recognition and Machine Learning
Assignment 2

Release Date: Apr 5, 2021
Due Date: Apr 18, 2021 (11.59pm)

Roll No: EE18B067
Name: Abhishek Sekar
Collaborators (if any): S.Abishek (EE18B001)
References (if any): Bishop PRML book, Class slides and personal notes,link 1,link 2

- Use LaTeX to write-up your solutions (in the solution blocks of the source LaTeX file of this assignment), and submit the resulting single pdf file at GradeScope by the due date. (Note: **No late submissions** will be allowed, other than one-day late submission with 10% penalty or four-day late submission with 30% penalty! Within GradeScope, indicate the page number where your solution to each question starts, else we won't be able to grade it! You can join GradeScope using course entry code **5VDNKV**).
- For the programming question, please submit your code (rollno.ipynb file and rollno.py file in rollno.zip) directly in moodle, but provide your results/answers in the pdf file you upload to GradeScope.
- 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 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.
- Points will be awarded based on how clear, concise and rigorous your solutions are, and how correct your code is. Overall points for this assignment would be **min**(your score including bonus points scored, 50).

1. (10 points) [SINGULARLY PCA!] Consider a dataset of N points with each datapoint being a D-dimensional vector in $\mathbb{R}^D$. Let's assume that:
   - we are in a high-dimensional setting where $D >> N$ (e.g., D in millions, N in hundreds).
   - the $N \times D$ matrix X corresponding to this dataset is already mean-centered (so that each column's mean is zero, and the covariance matrix seen in class becomes $S = \frac{1}{N}X^TX$).
   - the rows (datapoints) of X are linearly independent.

   Under the above assumptions, please attempt the following questions.

   (a) (3 points) Whereas X is rectangular in general, XX^T and X^TX are square. Show that these two square matrices have the same set of non-zero eigenvalues. Further, argue briefy why these equal eigenvalues are all positive and N in number, and derive the multiplicity of the zero eigenvalue for both these matrices.
   (Note: The square root of these equal positive eigenvalues $\{\lambda_i := \sigma_i^2\}_{i=1,\ldots,N}$ are called the singular values $\{\sigma_i\}_{i=1,\ldots,N}$ of X.)

Solution:
Same set of non-zero eigenvalues:
Let us consider the eigenvalues of XX^T which is an NxN matrix.
Let λ_i be a non-zero eigenvalue of XX^T and v_i be its corresponding eigenvector, then we have,

$$\begin{aligned} & XX^T v_i = \lambda_i v_i \\ \Rightarrow & X^T X X^T v_i = \lambda_i X^T v_i && \text{(Multiplying by } X^T \text{ on both sides)} \\ \Rightarrow & X^T X \left(X^T v_i\right) = \lambda_i \left(X^T v_i\right) && \text{(Note that } X^T v_i \neq 0 \text{ as } \lambda_i \text{ is non-zero)} \end{aligned}$$

Therefore, from the above expression, $X^T v_i$ is a legitimate (non-zero) eigenvector corresponding to the eigenvalue λ_i for the matrix $X^T X$.This tells us that all non-zero eigenvalues of $XX^T \subseteq$ non-zero eigenvalues of $X^T X$.
Likewise, considering λ_j, a non-zero eigenvalue of $X^T X$ and u_j its corresponding eigenvector, we have,

$$\begin{aligned} & X^T X u_j = \lambda_j u_j \\ \Rightarrow & XX^T X u_j = \lambda_j X u_j && \text{(Multiplying by X on both sides)} \\ \Rightarrow & XX^T \left(X u_j\right) = \lambda_j \left(X u_j\right) && (X u_j \neq 0 \text{ as } \lambda_j \neq 0\ ) \end{aligned}$$

From this, Xu_j is an eigenvector to the eigenvalue λ_j for the matrix XX^T. Therefore, all non-zero eigenvalues of $X^T X \subseteq$ non-zero eigenvalues of XX^T which implies that both $X^T X$ and XX^T possess the same set of non-zero eigenvalues.

Non-zero eigenvalues are positive:
The non-zero eigenvalues are positive, since the square matrices XX^T and $X^T X$ are positive semi-definite.
We can prove this by verifying if the matrix XX^T satisfies the definition of a positive operator on an N dimensional vector v.

$$\begin{aligned} \text{Consider } & < XX^T, v > && \text{(Where <a,b> denotes the inner product between a and b)} \\ \Rightarrow & < X^T v, X^T v > && \text{(For a real space, } X^T \text{ is the adjoint of X)} \\ \Rightarrow & \parallel X^T v \parallel^2 \geqslant 0 && \text{(Therefore } XX^T \text{ is positive)} \end{aligned}$$

Now, as XX^T is positive, all its eigenvalues are $\geqslant 0$. This implies that its non-zero eigenvalues are all positive.

Non-zero eigenvalues are N in number:

Invoking the Fundamental Theorem of Linear Algebra on XX^T, we have,

$$\dim V = \text{nullity}\left(XX^T\right) + \text{rank}\left(XX^T\right) \qquad \text{(Here dim(V) = N)}$$

As all the rows of X are linearly independant,

$$\text{rank}\left(XX^T\right) = N$$

Using all this to compute nullity, we get,

$$\text{nullity}\left(XX^T\right) = N - N = 0 \tag{1}$$

Nullity is the rank of the null space, therefore as nullity of XX^T is 0, it follows that the zero valued eigenvalues are 0 in number, which means that all the N eigenvalues of XX^T are non-zero.

Multiplicity of zero eigenvalue:
From 1, the multiplicity of zero eigenvalue in XX^T is 0. Now, we have shown that XX^T and X^TX share all their non-zero eigenvalues. Therefore, whatever extra eigenvalues are present in X^TX are all zero valued.This can also be verified by the Fundamental Theorem of Linear Algebra.From this, the multiplicity of zero eigenvalue in X^TX is D-N.

(b) (2 points) We can choose the set of eigenvectors $\{u_i\}_{i-=1,\ldots,N}$ of XX^T to be an orthonormal set and similarly we can choose an orthonormal set of eigenvectors $\{v_j\}_{j=1,\ldots,D}$ for X^TX. Briefly argue why this orthonormal choice of eigenvectors is possible. Can you choose $\{v_i\}$ such that each v_i can be computed easily from u_i and X alone (i.e., without having to do an eigenvalue decomposition of the large matrix X^TX; assume $i = 1, \ldots, N$ so that $\lambda_i > 0$ and $\sigma_i > 0$)?
(Note: $\{u_i\}, \{v_i\}$ are respectively called the left,right singular vectors of X, and computing them along with the corresponding singular values is called the Singular Value Decomposition or SVD of X.)

Solution:
Existence of orthonormal eigenvectors:
We can prove that there exist orthonormal eigenvectors for both matrices by invoking the Real Spectral Theorem.
The Real Spectral Theorem states that, for a self-adjoint(Real and Symmetric) matrix, there exist an orthonormal basis of eigenvectors of that matrix.
So let us check if the matrices XX^T and X^TX are symmetric.

$$\left(XX^T\right)^T = XX^T \qquad \text{Therefore } XX^T \text{ is Real and Symmetric}$$

$$\left(X^TX\right)^T = X^TX \qquad \text{Therefore } X^TX \text{ is Real and Symmetric}$$

Therefore, from the Real Spectral Theorem,

- As XX^T is an NxN matrix, there exist N orthonormal eigenvectors which can be denoted by $\{u_i\}_{i=1,\ldots,N}$.
- As X^TX is a DxD matrix, there exist D orthonormal eigenvectors which can be denoted by $\{v_j\}_{j=1,\ldots,D}$.

Choosing appropriate v_i:
Note that,the easy computation is possible only for the values of i given in the question. ie, $i = 1, \ldots, N$. We can find the higher eigenvectors by computing the null space of X^TX.
As seen in the first subdivision, if λ_i is an eigenvalue of XX^T with corresponding eigenvector u_i, we have,

$$\begin{aligned}
&XX^Tu_i = \lambda_i u_i \\
\Rightarrow &X^TXX^Tu_i = \lambda_i X^Tu_i && \text{(Multiplying by } X^T \text{ on both sides)} \\
\Rightarrow &X^TX\left(X^Tu_i\right) = \lambda_i\left(X^Tu_i\right) \\
\Rightarrow &X^TXw_i = \lambda_i w_i && \text{(Denoting } X^Tu_i \text{ as } w_i\text{)}
\end{aligned}$$

The w_i are the eigenvectors of X^TX.Let us check if they're orthogonal to each other.

$$\begin{aligned}
< w_i, w_j > = &< X^Tu_i, X^Tu_j > && \text{(for } i \neq j\text{)} \\
\Rightarrow &< XX^Tu_i, u_j > \\
\Rightarrow &< \lambda_i u_i, u_j > \\
\Rightarrow &\lambda_i < u_i, u_j >= 0 && \text{(As } u_i \text{ are orthonormal)}
\end{aligned}$$

Now since we know that the w_i are indeed orthogonal, we can compute v_i by normalizing w_i, ie, $v_i = \frac{w_i}{\|w_i\|}$. Therefore, an easy way to compute $\{v_i\}_{i=1\ldots N}$ from u_i and X alone would be,

$$v_i = \frac{X^Tu_i}{\| X^Tu_i \|} \tag{2}$$

(c) (2 points) Applying PCA on the matrix X would be computationally difficult as it would involve finding the eigenvectors of $S = \frac{1}{N}X^TX$, which would take $O(D^3)$ time. Using answer to the last question above, can you reduce this time complexity to $O(N^3)$? Please provide the exact steps involved, including the exact formula for computing the normalized (unit-length) eigenvectors of S.

Solution:
Time complexity and steps:
Observe, that the lower D-N eigenvalues of S matrix are not relevant in the computation of PCA since they don't contribute to the error or variance.ie, if we're taking M principal components after PCA, the resulting error will be, $\Sigma_{i=M+1}^{D}\lambda_i$ which is $\Sigma_{i=M+1}^{N}\lambda_i$ as the other D-N eigenvalues are 0.
Also, it can be noted that, the eigenvectors of S remain the same as that of X^TX but the eigenvalues get scaled by N as shown below.

$$Sv_i = \lambda_i v_i \qquad (\lambda_i \text{ is an eigenvalue}, v_i \text{ its corresponding eigenvector})$$
$$\Rightarrow \frac{1}{N}X^TXv_i = \frac{1}{N}\acute{\lambda_i}v_i \qquad (\text{Where } \acute{\lambda_i} \text{ is an eigenvalue of } X^TX)$$

Therefore,

$$\Rightarrow \lambda_i = \frac{1}{N}\acute{\lambda_i}$$

As the eigenvectors remain intact, we can employ the same procedure to compute them as in the previous subdivision. So, to reduce the time complexity, we can perform the following steps.

- Compute the eigenvalues and eigenvectors of $\frac{1}{N}XX^T$.This results in us finding the eigenvalues λ_i and corresponding orthonormal eigenvectors u_i. This can be done with a time complexity of $O(N^3)$.
- Employ the formula described by 2, $v_i = \frac{X^Tu_i}{\|X^Tu_i\|}$. This computation is of complexity O(DN) as it involves N multiplications for D rows.However, there are N such eigenvectors that are required to be computed. So total time complexity involved in the simplified computation is $O(DN^2)$.

So, overall complexity involved in computing the eigenvectors of S is the computational complexity incurred in the above steps, ie, $O(N^3 + DN^2) = O(DN^2)$ which is much much smaller than the original computational complexity of $O(D^3)$ as $D \gg N$.

(d) (3 points) Exercise 12.2 from Bishop's book helps prove why minimum value of the PCA squared error J, subject to the orthonormality constraints of the set of principal axes/directions $\{u_i\}$ that we seek, is obtained when the $\{u_i\}$ are eigenvectors of the data covariance matrix S. That exercise introduces a modified squared error $\widetilde{J}$, which involves a matrix H of Langrange multipliers, one for each constraint, as follows:

$$\widetilde{J} = Tr\left\{\widehat{U}^TS\widehat{U}\right\} + Tr\left\{H(I - \widehat{U}^T\widehat{U})\right\}$$

where $\widehat{U}$ is a matrix of dimension $D \times (D - M)$ whose columns are given by u_i. Now, any

solution to minimizing $\widetilde{J}$ should satisfy $S\widehat{U} = \widehat{U}H$, and one specific solution is that the columns of $\widehat{U}$ are the eigenvectors of S, in which case H is a diagonal matrix. Show that any general solution to $S\widehat{U} = \widehat{U}H$ also gives the same value for $\widetilde{J}$ as the above specific solution.
(Hint: Show that H can be assumed to be a symmetric matrix, and then use the eigenvector expansion i.e., diagonalization of H.)

Solution:
Showing H can be assumed to be a symmetric matrix
Let us consider the lagrange multiplier matrix H(with real elements).We can observe from the cost function that the element H_{ii} of the lagrange multiplier matrix penalises the term $1 - < u_i, u_i >$ and element H_{ij} penalises $-< u_i, u_j >$ for $i \neq j$. Since we're dealing with real vectors u_i and u_j, $< u_i, u_j >=< u_j, u_i >$.
We know that,

$$Tr\left\{H(I - \widehat{U}^T\widehat{U})\right\} = \sum_i ((1- < u_i, u_i >) \cdot H_{ii}) + \sum_{\substack{i,j \\ i\neq j}} ((- < u_i, u_j >) \cdot H_{ij})$$

By observing the second part of the expression above, we can see that even if $H_{ij} \neq H_{ji}$ in our original matrix H, we can assume a symmetric version of H,represented here by H^S without any loss of generality such that, $H^S_{ij} = H^S_{ji} = \frac{H_{ij}+H_{ji}}{2}$.This replacement will still result in the same value for $\widetilde{J}$. Therefore, we can take H to be a symmetric matrix for all practical purposes.

General solution results in the same value as specific solution:
Firstly let us compute the value of $\widetilde{J}$ obtained from the specific solution.In the specific case,the columns of $\widehat{U}$ are taken as the orthonormal eigenvectors u_i of S. This makes the matrix $\widehat{U}$ an isometry (orthogonal),ie,$\widehat{U}^T\widehat{U} = I$ and therefore $\widetilde{J}$ becomes $Tr\left\{\widehat{U}^TS\widehat{U}\right\}$.Now,using the fact that any solution must satisfy $S\widehat{U} = \widehat{U}H$, we can write, $Tr\left\{\widehat{U}^TS\widehat{U}\right\} = Tr\left\{\widehat{U}^T\widehat{U}H\right\} = Tr\{H\}$.
Therefore the specific solution is,

$$\widetilde{J} = Tr\{H\} \tag{3}$$

General solution:
H can be taken as a real and symmetric matrix without any loss of generality.
Let matrix V represent the matrix, whose columns v_i are the eigenvectors of H. Such a representation is possible as per the Real Spectral Theorem. Then, $HV = V\Lambda$, where Λ is a diagonal matrix with λ_{ii} being the eigenvalue of H for the eigenvector v_i.
Now, any general solution to minimizing $\widetilde{J}$ must satisfy $S\widehat{U} = \widehat{U}H$.

Tweaking this expression a bit,we get,

$$\begin{aligned} S\widehat{U}V = \widehat{U}HV = \widehat{U}V\Lambda \qquad & \text{(Using } HV = V\Lambda\text{)} \\ \Rightarrow S\widetilde{U} = \widetilde{U}\Lambda \qquad & \text{(Where } \widetilde{U} = \widehat{U}V\text{)} \end{aligned}$$

This tells us that the matrix V transforms the matrix $\widehat{U}$ to an orthogonal matrix $\widetilde{U}$ with its columns being the orthonormal eigenvectors of S. Note that V can be chosen appropriately to ensure that the columns of $\widetilde{U}$ possess unit norm.
Now if V is an N dimensional orthogonal matrix with columns having unit norm, then $Tr\{A\} = Tr\left\{V^T A V\right\}$ where A is any N dimensional real matrix. Let a_i represent the row vectors of A,a_{ii} represent the i^{th} diagonal element of A and v_i represent the column vectors of V.
A short proof is given below,

$$\begin{aligned} Tr\left\{V^T(AV)\right\} = Tr\left\{(AV)V^T\right\} \qquad & \text{(As Tr\{AB\} = Tr\{BA\})} \\ \Rightarrow Tr\left\{A(VV^T)\right\} = Tr\{A\} \qquad & \text{(As V is orthogonal, } VV^T = I\text{)} \end{aligned}$$

While we take the gradient of $\widetilde{J}$, we also take it with respect to H. Therefore, the minimized expression of $\widetilde{J}$ becomes $Tr\left\{\widehat{U}^T S\widehat{U}\right\}$ as the H matrix is chosen in such a manner that the rest of $\widetilde{J}$ goes to 0. Like before, let V denote the orthonormal eigenvector matrix of H.

$$\begin{aligned} & Tr\left\{\widehat{U}^T S\widehat{U}\right\} = Tr\left\{\widehat{U}^T\widehat{U}H\right\} && \text{(Using the expression } S\widehat{U} = \widehat{U}H \text{ satisfied by solution)} \\ \Rightarrow & Tr\left\{V^T\widehat{U}^T\widehat{U}HV\right\} = Tr\left\{V^T\widehat{U}^T\widehat{U}V\Lambda\right\} && \text{(Using above proof and } HV = V\Lambda\text{)} \\ \Rightarrow & Tr\left\{\widetilde{U}^T\widetilde{U}\Lambda\right\} = Tr\{\Lambda\} = Tr\{H\} && \text{(Using } \widetilde{U} = \widehat{U}V \text{ and } \widetilde{U}^T\widetilde{U} = I\text{)} \end{aligned}$$

We can see that the minimized value of $\widetilde{J}$ obtained by the general solution matches with that obtained by a specific solution in 3

2. (10 points) [TO SQUARE OR NOT SQUARE WITH K-MEANS]

   (a) (3 points) If instead of squared Euclidean distance, you use ℓ_1 norm in the objective function of (hard) K-means, then what are the new assignment and update equations? If your data contains outliers, would you prefer this over the regular K-means? Justify.

Solution:
First off, let $\|x\|_1$ denote the ℓ_1 norm of a vector x. Let, the cluster center of a cluster i be represented by, $m^{(i)}$.
Assignment step:

The objective function to be minimized can be expressed as,

$$J\left(r, m^{(1)}, m^{(2)}, \ldots, m^{(K)}\right) = \Sigma_{i=1}^{N}\Sigma_{j=1}^{K} r_j^{(i)} \parallel x^{(i)} - m^{(j)} \parallel_1 \tag{4}$$

Here K is the number of clusters and N the number of datapoints. The assignment step remains the same as before but with ℓ_1 norm based distance instead.

$$r_k^{(n)} = \begin{cases} 1 & \text{If } k = \underset{i}{\operatorname{argmin}} \parallel x^{(n)} - m^{(i)} \parallel_1 \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

Essentially, if the k^{th} cluster centroid is the closest to the n^{th}data point, the *partial responsibility* $r_k^{(n)}$ becomes 1 and it is allotted 0 otherwise.

Update step:
Now for the update step, we would like to minimize a variant of the objective function described in 4 that manifests as the sum of distances between the cluster centroid and all data points in the cluster.That is,

$$J_{new}\left(m^{(j)}\right) = \Sigma_{i=1}^{N_k} \parallel x^{(i)} - m^{(j)} \parallel_1 \tag{6}$$

So the update process is basically finding out the new cluster centroid of the N_k points which were allocated (ie $r_i^{(n)} = 1$) to the i^{th} cluster in the assignment step. As this is equivalent to finding $m^{(j)}$ that minimizes the objective function given in 6, we can take the gradient of that with respect to $m^{(j)}$ in order to find the minimizing value.

$$\nabla_{m^{(j)}} J_{new}\left(m^{(j)}\right) = \nabla_{m^{(j)}}\left(\Sigma_{i=1}^{N_k} \parallel x^{(i)} - m^{(j)} \parallel_1\right) = \sum_{i=1}^{N_k} \nabla_{m^{(j)}}\left(\parallel x^{(i)} - m^{(j)} \parallel_1\right)$$

Now,using the definition of ℓ_1 norm, we have

$\parallel x^{(i)} - m^{(j)} \parallel_1 = \Sigma_{n=1}^{D} \left|x_n^{(i)} - m_n^{(j)}\right|$ (x_n refers to component of x in dimension n)

Using this,the gradient simplifies to,

$$\sum_{i=1}^{N_k} \nabla_{m^{(j)}}\left(\parallel x^{(i)} - m^{(j)} \parallel_1\right) = \sum_{i=1}^{N_k}\sum_{n=1}^{D} \frac{d\left(\left|x_n^{(i)} - m_n^{(j)}\right|\right)}{dm_n^{(j)}} \tag{7}$$

$$\text{Observe that, } \frac{d\left(\left|x_n^{(i)} - m_n^{(j)}\right|\right)}{dm_n^{(j)}} = \begin{cases} 1 & m_n^{(j)} > x_n^{(i)} \\ -1 & m_n^{(j)} < x_n^{(i)} \\ 0 & m_n^{(j)} = x_n^{(i)} \end{cases}$$

Therefore,7 will equate to 0 when $m_n^{(j)}$ is such that,the number of cases where the above derivative equates to 1 and -1 are equal across all possible n.This means that the number of data points on either side of $m^{(j)}$ are equal,ie,$m^{(j)}$ represents the median of these data-points. Therefore the update equation is,

$$m^{(j)} = \text{median}\left(\frac{x^{(i)}}{r_j^{(i)}}\right) \forall j \in (1, 2, \ldots, K) \tag{8}$$

Preference in the presence of outliers:
Yes,ℓ_1 norm version of K-Means is preferred in the presence of outliers,if we want to ignore them. This is because, in normal K-Means,the distance to the outlier is computed,which is quadratic in nature and therefore more sensitive to its presence.However,here, we just compute the median of the data points in a cluster to find the cluster centroids and therefore this reduces the influence of the outlier when compared to normal K-Means.

(b) (2 points) Consider a Gaussian mixture model with scalar covariance matrices: $\Sigma_r = \sigma^2 I$ where σ is a fixed parameter, r represents the mixture-component/cluster, and I the identity matrix. Show that for this model as σ tends to zero, the EM-based soft K-means algorithm (i.e., its assignment/update equations) become the same as the hard K-means algorithm.

Solution:
EM algorithm for soft K-Means:
Using the expressions for (Vanilla) Soft K-Means using Gaussian Mixture Models shown in class,while considering the fact that the variance σ is a fixed parameter, we have,
E-Step/Assignment:

$$r_k^{(n)} = \frac{\pi_k \mathcal{N}\left(\frac{x^{(n)}}{m^{(k)},\sigma}\right)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}\left(\frac{x^{(n)}}{m^{(k')},\sigma}\right)} \tag{9}$$

Where, $\mathcal{N}\left(\frac{x^{(n)}}{m^{(k)},\sigma}\right)$ denotes $\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^I \exp\left(\frac{\sum_{i=1}^{I}\left(x_i^{(n)} - m_i^{(k)}\right)^2}{2\sigma^2}\right)$ with I representing the number of dimensions in the datapoints,K denoting the total number of clusters and π_k denoting the prior probability corresponding to cluster k.

M-Step/Update:

The update equation for the cluster centroids is,

$$m^{(k)} = \frac{\sum_{n=1}^{N} r_k^{(n)} x^{(n)}}{R^{(k)}} \tag{10}$$

Where N refers to the total number of datapoints present and $R^{(k)}$ denotes the total responsibility $\sum_{n=1}^{N} r_k^{(n)}$ of cluster k.
The M-Step algorithm for π_k is,

$$\pi_k = \frac{R_k}{\sum_{k'=1}^{K} R_{k'}} \tag{11}$$

When $\sigma \to 0$:
Let us analyze the effect of σ approaching zero on equations 9,10,11.

- **Equation 9:** As $\sigma \to 0$,$\mathcal{N}\left(\frac{x^{(n)}}{m^{(k)},\sigma}\right)$ approaches zero for all data points.However, $\mathcal{N}\left(\frac{x^{(n)}}{m^{(i)},\sigma}\right)$ where $m^{(i)}$ is the closest cluster centroid to the datapoint $x^{(n)}$, decays the slowest when compared to the corresponding gaussians for the other cluster centroids.Using this information in the limit as σ tends to zero, we have,

$$\lim_{\sigma \to 0} r_k^{(n)} = \begin{cases} 1 & \text{if } m^{(k)} \text{ is the closest cluster centroid to } x^{(n)} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

  When $m^{(k)}$ is the closest cluster centroid to the datapoint, the numerator term in 9 is the slowest decaying gaussian. In the limit, the denominator term can be approximated by the slowest decaying gaussian as the other gaussians decay off faster and this leads to a value of one. When $m^{(k)}$ is not the closest cluster centroid, the numerator gaussian decays much faster than that in the denominator leading to a value of zero.

- **Equation 10:** Using the information obtained in 12, equation 10 in the limit becomes,

$$m^{(k)} = \frac{\sum_{n=1}^{N_k} r_k^{(n)} x^{(n)}}{R^{(k)}} \tag{13}$$

Where, N_k is the number of points for which $m^{(k)}$ is the closest cluster centroid, ie,$r_k^{(n)} = 1$ in 12.For the remaining N - N_k datapoints $r_k^{(n)}$ is 0. Therefore, R^k becomes $\sum_{n=1}^{N} r_k^{(n)} = N_k$.

- **Equation 11:**Using the information obtained in 13,substituting for $R^{(k)}$ and using the fact that $\sum_{k'=1}^{K} R_{k'} = N$ equation 11 in the limit becomes,

$$\pi_k = \frac{N_k}{N} \tag{14}$$

Observing equations 12,13,14, it is seen that as σ tends to 0,the E step and M step equations for soft K-Means become the assignment and update equations for hard K-Means algorithm.

(c) (5 points) We will see how K-means clustering can be done in polynomial time if the data points are along a line (1-dimensional or 1D).

i. (1 point) Consider four datapoints: $x_1 = 1, x_2 = 3, x_3 = 6$, and $x_4 = 7$; and desired number of clusters $k = 3$. What is the optimal K-means clustering in this case?

Solution:
Optimal K-Means clustering:
Since the data is already provided in an ordered fashion, we can use the dynamic programming method described in **part (iv)** to compute the optimal K-Means clustering. This results in the following output.

- Cluster 1 with centroid $m^{(1)} = x_1 = 1$ containing the datapoint x_1
- Cluster 2 with centroid $m^{(2)} = x_2 = 3$ containing the datapoint x_2
- Cluster 3 with centroid $m^{(3)} = \frac{x_3+x_4}{2} = 6.5$ containing the datapoints x_3 and x_4
- Objective function value :

$$J\left(r, m^{(1)}, m^{(2)}, m^{(3)}\right) = \Sigma_{i=1}^{4}\Sigma_{j=1}^{3} r_j^{(i)} \left(x^{(i)} - m^{(j)}\right)^2 = (6-6.5)^2 + (7-6.5)^2 = 0.5$$

ii. (1 point) You might think that the iterative K-means algorithm seen in class converges to global optima with 1D datapoints. Show that it can get stuck in a suboptimal cluster assignment for the problem in part (i).

Solution:
Sub optimal cluster assignment:
We can demonstrate by an example that the iterative K-Means algorithm gets stuck in a suboptimal cluster assignment for the problem in the previous part.
Initialization Step:
Let the cluster centroids be initialized to the values given below as part of the process initializing them to random points.

- $m^{(1)} = 2$
- $m^{(2)} = 6$
- $m^{(3)} = 7$

Assignment Step:

- Assign x_1 and x_2 to $m^{(1)}$.
- Assign x_3 to $m^{(2)}$.
- Assign x_4 to $m^{(3)}$.

Update Step:

- $m^{(1)} = \frac{x_1+x_2}{2} = \frac{1+3}{2} = 2$.
- $m^{(2)} = x_3 = 6$.
- $m^{(3)} = x_4 = 7$.

The subsequent iterations don't have any difference in the assignment and the update steps.Therefore, this is the final set of cluster centroids and the data points allotted to it.
Computing the objective function value for this case, we get,

$$J\left(r, m^{(1)}, m^{(2)}, m^{(3)}\right) = \Sigma_{i=1}^{4}\Sigma_{j=1}^{3} r_j^{(i)}\left(x^{(i)} - m^{(j)}\right)^2 = (1-2)^2 + (3-2)^2 = 2 > 0.5$$

As the objective function's value is higher than that obtained in the previous part this is a suboptimal cluster assignment resulting in local optima.

iii. (3 points) Suppose we sort our data such that $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n$. Show then that any optimal K-means clustering partitions the points into contiguous intervals, i.e. prove that each cluster in an optimal clustering consists of points $x_a, x_{a+1}, \ldots, x_b$ for some $1 \leqslant a \leqslant b \leqslant n$.

Solution:
Proof by contradiction:
Let the optimal K-Means algorithm assign clusters such that there exist $x_i \leqslant x_j$ with x_i assigned to the cluster k_1 and x_j assigned to the cluster k_2 such that $m^{(k_1)} > m^{(k_2)}$.These 4 points can be ordered in 5 different ways.Let's look at each of them in a sequential manner.

- **Case 1:** $m^{(k_2)} < m^{(k_1)} \leqslant x_i \leqslant x_j$
  Then, $(x_j - m^{(k_2)})^2 > (x_j - m^{(k_1)})^2$
  Therefore,x_j should've instead been allocated to k_1 as it is closer. Therefore the original assignment was not an optimal one as the above assignment reduces the cost from the objective function.

- **Case 2:** $m^{(k_2)} \leqslant x_i < m^{(k_1)} \leqslant x_j$ or $m^{(k_2)} < x_i \leqslant m^{(k_1)} \leqslant x_j$

  Then, $(x_j - m^{(k_2)})^2 > (x_j - m^{(k_1)})^2$
  Therefore,x_j should've instead been allocated to k_1 as it is closer. Therefore the original assignment was not an optimal one as the above assignment reduces the cost from the objective function.

- **Case 3:** $x_i \leqslant m^{(k_2)} < m^{(k_1)} \leqslant x_j$
  Then, $(x_j - m^{(k_2)})^2 > (x_j - m^{(k_1)})^2$
  Therefore,x_j should've instead been allocated to k_1 as it is closer. Therefore the original assignment was not an optimal one as the above assignment reduces the cost from the objective function.

- **Case 4:** $x_i \leqslant m^{(k_2)} \leqslant x_j < m^{(k_1)}$ or $x_i \leqslant m^{(k_2)} < x_j \leqslant m^{(k_1)}$

  Then, $(x_i - m^{(k_1)})^2 > (x_i - m^{(k_2)})^2$
  Therefore,x_i should've instead been allocated to k_2 as it is closer. Therefore the original assignment was not an optimal one as the above assignment reduces the cost from the objective function.

- **Case 5:** $x_i \leqslant x_j \leqslant m^{(k_2)} < m^{(k_1)}$
  Then, $(x_i - m^{(k_1)})^2 > (x_i - m^{(k_2)})^2$
  Therefore,x_i should've instead been allocated to k_2 as it is closer. Therefore the original assignment was not an optimal one as the above assignment reduces the cost from the objective function.

Thus,all the cases result in an arrangement wherein the objective function's value is made even lesser implying that the assignment we originally thought was optimal isn't

really optimal. Therefore,this shows that, for the one dimensional case, the optimal K-Means algorithm indeed partitions the points into contiguous intervals.

iv. (1 point) [BONUS] Show a $O(kn^2)$ dynamic programming algorithm of k-means when the data is 1-dimensional.

Solution:
Dynamic Programming Algorithm for K-Means
Without loss of generality, let us assume that the input datapoints are received in a sorted manner before clustering, ie, $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n$.
Let us define a few terms. Let TC[i][m] denote the total cost incurred in optimally clustering the first m datapoints into i clusters. Let CC[i][j] denote the cluster cost incurred in allotting the points $x_i, \ldots, x_j$ to one cluster.
Therefore,

$$CC[i][j] = \sum_{k=1}^{k=j} \left(x^{(k)} - m^{(i,j)}\right)^2$$

$$m^{(i,j)} = \frac{1}{j-i+1} \sum_{k=1}^{k=j} x^{(k)}$$

Where, $m^{(i,j)}$ is the cluster centroid of the cluster. Therefore, in a dynamic programming context, we can initialize TC[0][0] with 0 and likewise CC[0][0] with 0 as there is no cost incurred without any data points. For $i > 0$,we have,

$$TC[i][m] = \min_{j=1}^{m} TC[i-1][j-1] + CC[j][m] \tag{15}$$

The above equation computes optimal cost as the cost incurred in allocating j points optimally in i-1 clusters and the rest of the points in a new cluster. Let Ind[i][m] be the value of j that minimizes 15

$$Ind[i][m] = \min \left\{ \operatorname*{argmin}_{j=1}^{m} TC[i-1][j-1] + CC[j][m] \right\} \tag{16}$$

Thus, we can compute Ind[i][m] and then find the data point corresponding to this index and backtrack.Note that $x^{(Ind[i][m])}$ is the first point in the i^{th} cluster from 15.
We can compute the cluster costs and cluster centroids by employing recurrence rela-

tions.The cluster centroid $m^{(i,i)}$ can be initialized by $x^{(i)}$.

$$m^{(i,j)} = \frac{(j-1)\cdot m^{(i,j-1)} + x(j)}{j-i+1} \tag{17}$$

$$CC[i][j] = \sum_{k=1}^{k=j} \left(x^{(k)} - m^{(i,j)}\right)^2 = CC[i][j-1] + \frac{j-1}{j-i+1}\left(x^{(j)} - m^{(i,j-1)}\right)^2 \tag{18}$$

This recurrence relation takes $O(n^2)$ time to compute all values beforehand.Using this, we can compute CC[i][j] in O(1) time and also find out Ind[i][m] in O(1) time.This means that TC[i][m] can be computed in O(n) time finding optimal j.TC is a K x N matrix. Therefore the whole algorithm evaluates in $O(kn^2)$ time.Combining both the cost update steps and the recurrence steps, total complexity = $O(kn^2+n^2) = O(kn^2)$.

3. (10 points) [THINKING HIERARCHICALLY...] Consider some of the most common metrics of distance between two clusters $A = \{a_1, a_2, \ldots, a_m\}$ and $B = \{b_1, b_2, \ldots, b_n\}$.

   - Minimum distance between any pair of points from the two clusters
     $$\min_{a\in A, b\in B} \|a - b\|$$
   - Maximum distance between any pair of points from the two clusters,
     $$\max_{a\in A, b\in B} \|a - b\|$$
   - Average distance between *all* pairs of points from the two clusters,
     $$\frac{1}{m\cdot n}\sum_{i=1}^{m}\sum_{j=1}^{n}\|a_i - b_j\|$$

   As discussed in class, we can obtain clusters by *cutting* the hierarchical tree with a line that crosses at required number of points (K).

   (a) (2 points) Which of the three distance/dissimilarity metrics described above would most likely result in clusters most similar to those given by K-means? (Consider the hierarchical clustering method as described in class and further *cut* the tree to obtain K clusters. Assume K is power of 2.) Explain briefly.

Solution:
K-Means through hierarchial clustering:
Let us briefly explore the clusters obtained using the three dissimilarity metrics.

- **Single Linkage Clustering:** This is the method obtained using the dissimilarity metric of minimum distance between a pair of points belonging to either cluster.This

results in chain(curved or straight) like clusters where the distance between points belonging to the two clusters could be a lot larger than the dissimilarity metric chosen.Therefore, these will not result in clusters similar to K-Means as their objective is different.

- **Complete Linkage Clustering:** This is the method obtained using the dissimilarity metric of maximum distance between a pair of points belonging to either cluster. This results in circular shaped clusters with compact cluster boundaries where the distance between points belonging to the two clusters is not much lesser than the dissimilarity metric chosen.Therefore, these will not result in clusters similar to K-Means as their objective is different.

- **Average Linkage Clustering:** This is the method obtained using the dissimilarity metric of average distance between all pair of points belonging to either cluster.This will result in clusters most resemblant to that obtained through K-Means, since the dissimilarity metric is somewhat similar. In K-Means, we would like the distance of a datapoint to its own cluster centroid(which can be thought of as the average distance of the point to other points belonging to the same cluster) to be lesser than that of another cluster centroid(which can be thought of as the average distance of the point to points belonging to the other cluster). Therefore, the dissimilarity metric produces a result which is somewhat similar to the distance between the two cluster centroids.This is mathematically demonstrated below.

$$\frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \|a_i - b_j\| = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{n} \sum_{j=1}^{n} \|a_i - b_j\| \right)$$

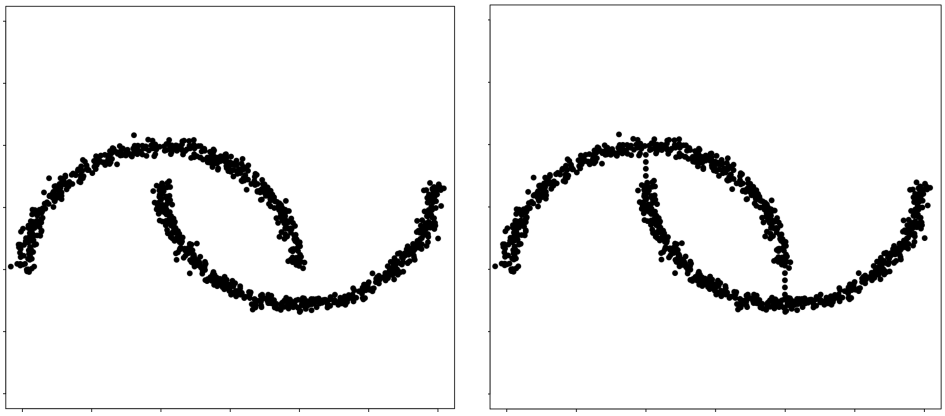
$$\approx \frac{1}{m} \sum_{i=1}^{m} \left(\|a_i - m^{(b)}\| \right) \text{ (Where } m^{(b)} \text{ is the cluster centroid of cluster B)}$$

$$\approx \|m^{(a)} - m^{(b)}\|$$

Average distance of all points belonging to cluster A to $m^{(b)}$ manifests as the effective distance from $m^{(a)}$ to $m^{(b)}$.

Therefore, average linkage clustering or the third dissimilarity metric described will most likely result in clusters resemblant to those obtained through K-Means.

(b) (3 points) Which among the three metrics discussed above (if any) would lead hierarchical clustering to correctly separate the two moons in Figure 1a? How would your answer change (if at all) in case of Figure 1b? Explain briefly.



(a) (b)

Figure 1: (a) Standard moon crescent distribution. (b) Moon crescent distribution with data points in adjoining area.

Solution:
Metric to separate the moons in 1a:
The figure 1a requires to be separated into the two crescent moons.The crescent moons resemble a chain(albeit curved) and therefore the single linkage clustering method described in the above subdivision will be appropriate for correctly separating the two moons as this is the characteristic cluster shape that is obtained through the single linkage clustering method(corresponding to the "minimum distance" dissimilarity metric as described in the above subdivision). The complete linkage clustering clustering method and average linkage clustering method result in compact circular clusters and closely knit collectives respectively.Therefore, these two methods are not suitable for this scenario. Moreover, the crescent moon is long rather than compact.Therefore the "maximum distance" dissimilarity metric does not represent the distance between the two crescent moons. Hence the compact clustering method is not recommended for the given scenario.

Metric to separate the moons in 1b:
The figure 1b requires to be separated into the two crescent moons.However,in this case, neither of the three dissimilarity metrics or clustering methods will work.The average linkage clustering method and the complete linkage clustering method will not work for similar reasons as in 1a.The single linkage clustering method ceases to work as intended here as the two crescent moons are joined, thereby ensuring that the "minimum distance" dissimilarity metric is no longer relevant in distinguishing between the two crescent moons despite our ability to visually distinguish them.

(c) (3 points) Consider the distance matrix in Table 1. Show the hierarchy of clusters created by the minimum distance hierarchical clustering algorithm, along with the intermediate steps.

Finally, draw the dendrogram with edge lengths indicated.
(Note: You can draw the dendrogram on paper and upload the screenshot.)

Table 1: Distance between nodes

	A	B	C	D	E
A	0	0.73	6.65	4.61	5.24
B	0.73	0	4.95	2.90	3.45
C	6.65	4.95	0	2.24	1.41
D	4.61	2.90	2.24	0	1
E	5.24	3.45	1.41	1	0

Solution:
Hierarchy of clusters:
Minimum distance hierarchial clustering algorithm:

- **Identify the clusters closest to each other.**
- **Compute cluster distances to the newly formed node:** This is done by computing the minimum of the distance between the cluster and components of the newly formed node.
- **Compute equivalent distance matrix:** This is done by clubbing the two clusters closest to each other to form a new node.
- **Find height in the dendrogram from the base to the newly formed node:** This is the distance to the newly formed node from the base,ie, $\frac{1}{2}$ the distance between the two clusters clubbed.

Step 1:

- **Closest clusters:** A and B, let the new node be W.
- **Updated cluster distances to** $\{A, B\}$**:**
  - d({A, B},C) = min(6.65,4.95) = 4.95
  - d({A, B},D) = min(4.61,2.90) = 2.90
  - d({A, B},E) = min(5.24,3.45) = 3.45
- **Updated Distance matrix**

	{A,B}	C	D	E
{A,B}	0	4.95	2.90	3.45
C	4.95	0	2.24	1.41
D	2.90	2.24	0	1
E	3.45	1.41	1	0

- **Height of node W from base:** $h_1 = \frac{0.73}{2} = 0.365$

Step 2:

- **Closest clusters:** D and E, let the new node be X.
- **Updated cluster distances to** $\{D, E\}$**:**
  - d($\{D, E\}$,$\{A, B\}$) = min(2.90,3.45) = 2.90
  - d($\{D, E\}$,C) = min(2.24,1.41) = 1.41
- **Updated Distance matrix**

	{A,B}	C	{D,E}
{A,B}	0	4.95	2.90
C	4.95	0	1.41
{D,E}	2.90	1.41	0

- **Height of node X from base:** $h_2 = \frac{1}{2} = 0.5$

Step 3:

- **Closest distance:** C and $\{D, E\}$, let the new node be Y.
- **Updated cluster distances to** $\{C, \{D, E\}\}$**:**
  - d($\{C, \{D, E\}\}$,$\{A, B\}$) = min(4.95,2.90) = 2.90
- **Updated Distance matrix**

	{A,B}	{C,{D,E}}
{A,B}	0	2.90
{C,{D,E}}	2.90	0

- **Height of node Y from base:** $h_3 = \frac{1.41}{2} = 0.705$

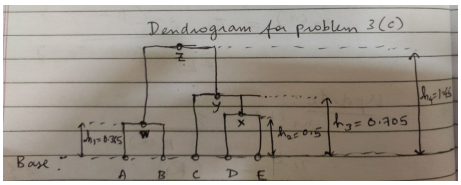
Step 4:

- **Updated Distance matrix**

	{{A,B},{C,{D,E}}}
{{A,B},{C,{D,E}}}	0

- **Height of final node Z from base:** $h_4 = \frac{2.90}{2} = 1.45$

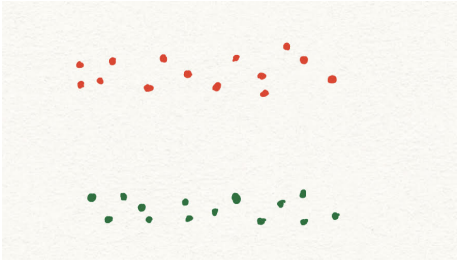
Shown below is the drawn dendrogram.



(a)

Figure 2: Dendrogram for the problem 3(c)

(d) (2 points) Which distance metric (minimum, maximum and average) is more likely to generate following results given in Figure 5a for K = 2? Why?



(a)

Figure 3: Result produced for K = 2 clusters. Red points belong to one cluster and green to the other.

Solution:
Metric to generate 5a
The "minimum distance" metric is more likely to generate the results shown in the figure since the shape of the clusters resemble chains which is the characteristic shape obtained using single linkage clustering method which implements the aforementioned distance metric.
The complete linkage clustering method and average linkage clustering method result in cluster shapes which are compact circles and closely knit collectives respectively.Therefore, either of these methods could not have produced the clusters shown in 5a as the chains are long rather than compact or closely knit blobs.

4. (10 points) [CUTTING SPECTRAL APART] One of the several ways to express a given dataset is by using a *graph*. Each of the N datapoints in the dataset can be thought of as a vertex/node in a graph and any two datapoints can be connected in the graph with an edge whose non-negative weight

W_{ij} indicates the similarity between the ith and jth datapoints. We will look at methods to partition this graph into two clusters, especially one that gained early prominence in computer vision. These methods can be recursively applied to partition the graph into any required number of clusters.

(a) (1 point) A graph cut is a technique that separates a given graph into two disjoint sets of vertices and the degree of similarity (formally called the *cut cost*) between the two sets is given by the sum of weights of the edges between the sets (i.e., edges whose two endpoint nodes lie in different sides of the partition). The obvious method to separate the data into two is by choosing a partition that has the minimum cut cost. What do you think is/are the drawback(s) of this method? (Hint: Think about the sizes of the two sets in the partition.)

Solution:
Drawbacks of minimum cut cost method:
Sometimes, the minimum cut cost method, supports in the formation of disjoint sets, wherein, one of the sets may contain just a few isolated nodes resulting in a non-ideal cut. This is the drawback of the minimum cut method.
Let me support my claim with an example.

Example demonstrating the drawback:
Lets say the graph cut results in the formation of two disjoint clusters of vertices, namely $\mathcal{A}$ and $\mathcal{B}$. Then the cut cost between these two sets is given by $\sum_{\substack{a\in\mathcal{A}\\ b\in\mathcal{B}}} W_{ab}$.

Lets use a parameter, which ensures that the weight between any datapoint i from $\mathcal{A}$ and another arbitrary datapoint j from $\mathcal{B}$, W_{ij} lies between 0 and 1.
Given below are two such parameters which could be employed.

- **Normalized distance:** $W_{ij} = \frac{d\left(x^{(i)},x^{(j)}\right)}{\max\limits_{\substack{n_2\in\mathcal{B}\\ n_1\in\mathcal{A}}} d\left(x^{(n_1)},x^{(n_2)}\right)}$, where we divide the distances between data points by the maximum distance obtained between points from either set.(assuming non-zero maximum distance)
- **Exponential:** $W_{ij} = exp\left(-d\left(x^{(i)},x^{(j)}\right)\right)$, as the distance between any two data points is positive, the exponential ensures that W_{ij} stays between 0 and 1.

Now,let us consider a dataset where we have an equal number of points/nodes in either set if we were to split them ideally,ie, say,$\frac{n}{2}$ points in either set. Let the cut cost for this be J_{ideal}.Then we have, $J_{ideal} \geqslant \frac{n^2}{4}W_{min}$ if W_{min} denotes the minimum weight observed.It is multiplied by $\frac{n^2}{4}$ since there are those many different weights between the points of either set.
Now, consider an alternate split, where we include 1 point in set $\mathcal{A}$ and the rest n-1 points in $\mathcal{B}$ and we call the associated cut cost $J_{example}$. Then, $J_{example}$ takes a value that is atmost

n-1.
If the cut cost method were indeed able to provide us with the ideal split, then $J_{ideal} < J_{example}$. Using the inequalities, we get,

$$W_{min} < \frac{4(n-1)}{n^2} \tag{19}$$

which tells us that, for the values of n, where the above condition isn't satisfied, $J_{example}$ ends up being lower than J_{ideal} and therefore minimum cut cost method results in a non-ideal cut.

(b) (2 points) Due to the above drawback(s), we use a variation of the min cut method called normalized cut to partition the graph into two. The problem of finding the minimum-cost normalized cut can be reduced to this problem:

$$min_y \frac{y^T(D-W)y}{y^TDy} \text{ subject to } y \in \{1,-c\}^N \text{ and } y^TD\mathbf{1} = 0$$

where y_i takes one of the two discrete values $\{1,-c\}$ to indicate which side of the cut/partition the ith datapoint belongs to, W is the symmetric $N \times N$ similarity (non-negative edge-weights) matrix, D is a diagonal matrix called the degree matrix with $d_{ii} = \Sigma_j W_{ij}$, and $\mathbf{1}$ is a vector whose entries are all ones. This expression (not including the constraints) is called the *Generalized Rayleigh's Quotient* (GRQ). The matrix in the numerator, $D - W$ is the called the Laplacian Matrix, denoted by L. Prove that the Laplacian matrix is a singular matrix.

Solution:
Proof for Laplacian matrix being singular:
Let L denote the Laplacian Matrix. We're given that $L = D - W$.
A matrix is said to be singular provided it has a non zero nullity.
Consider the matrix product L$\mathbf{1}$ where $\mathbf{1}$ represents the Nx1 vector with all its entries being 1.

$$L\mathbf{1} = (D-W)\mathbf{1} = \begin{bmatrix} d_{11} - (W_{11} + W_{12} + \ldots + W_{1N}) \\ \vdots \\ d_{NN} - (W_{N1} + W_{N2} + \ldots + W_{NN}) \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \Sigma_j W_{1j} - \Sigma_j W_{1j} \\ \vdots \\ \Sigma_j W_{Nj} - \Sigma_j W_{Nj} \end{bmatrix} \text{ (Using the definition of } d_{ii} \text{ from the degree matrix)}$$

$$\Rightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}$$

This implies that the vector **1** belongs to the null space of the matrix L. Therefore, since L possesses a non zero nullity, it is a singular matrix.

(c) (3 points) As the above minimization problem is NP-hard with the two constraints, we first let go of both constraints. Then, the above GRQ can be minimized over $y \in \mathbb{R}^N$ by solving the generalized eigenvalue system $(D - W)y = \lambda Dy$. Show that this equation can be expressed in the form $(ALA)z = \lambda z$, by expressing A, z in terms of D, W, y. Compute the eigenvector corresponding to the smallest eigenvalue of the matrix $M = ALA$.

Solution:
Expressing equation in the required form:
It has been given that the weights between indices are non-negative. What this means is that, the diagonal entries of the degree matrix D, are non-negative, therefore, it's eigenvalues are all non-negative. This makes D a positive operator and therefore, we can find a positive square root matrix $\sqrt{D}$ such that D = $\sqrt{D}\sqrt{D}$.Note that since the eigenvalues are positive for $\sqrt{D}$, the matrix is invertible.

$$\text{Where } \sqrt{D} = \begin{bmatrix} \sqrt{d_{11}} & 0 & \dots & 0 \\ 0 & \sqrt{d_{22}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{d_{nn}} \end{bmatrix}$$

Where d_{ii} represents the i^{th} diagonal element/eigenvalue of the degree matrix. Using this, we can rewrite the equation $(D - W)y = \lambda Dy$.

$$\begin{aligned} &(D - W)I_N y = \lambda D y && \text{(Where } I_N \text{ is the NxN identity matrix)} \\ \Rightarrow &(D - W)(\sqrt{D}^{-1}\sqrt{D})y = \lambda\sqrt{D}\sqrt{D}y && \text{(Using } \sqrt{D}^{-1}\sqrt{D} = I_N \text{ and } \sqrt{D}\sqrt{D} = D) \\ \Rightarrow &(\sqrt{D}^{-1})(D - W)(\sqrt{D}^{-1})(\sqrt{D}y) = \lambda(\sqrt{D}y) && \text{(Pre-Multiplying by } \sqrt{D}^{-1} \text{ on either side)} \end{aligned}$$

The above equation is of the required form, $(ALA)z = \lambda z$, where comparing equations, we get, $A = \sqrt{D}^{-1}$ and $z = \sqrt{D}y$.

Computing the eigenvector of the smallest eigenvalue:
We are required to compute the eigenvector corresponding to the smallest eigenvalue of the matrix $M = ALA$.Observing that W is a symmetric matrix and so is D by virtue of being diagonal, $L = D - W$ is also a symmetric matrix. Let us show that L is also a positive

matrix.x in the equations below is any N dimensional real vector.

Consider $< Lx, x > = x^T Lx$

Expanding the quadratic form using $L = D - W$,we have

$$\Rightarrow \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix} \begin{bmatrix} d_{11} - W_{11} & -W_{12} & \dots & -W_{1n} \\ -W_{21} & d_{22} - W_{22} & \dots & -W_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -W_{N1} & -W_{N2} & \dots & d_{NN} - W_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix} \begin{bmatrix} d_{11}x_1 - \Sigma_j W_{1j}x_j \\ \vdots \\ d_{NN}x_N - \Sigma_j W_{Nj}x_j \end{bmatrix}$$

Using $d_{ii} = \Sigma_j W_{ij}$, this becomes,

$$\Rightarrow \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix} \begin{bmatrix} \Sigma_j W_{1j}(x_1 - x_j) \\ \vdots \\ \Sigma_j W_{Nj}(x_N - x_j) \end{bmatrix}$$

$$\Rightarrow x_1\Sigma_j W_{1j}(x_1 - x_j) + \dots + x_N\Sigma_j W_{Nj}(x_N - x_j)$$

$$\Rightarrow \Sigma_{ij} W_{ij}(x_i - x_j)^2$$

The final expression can be obtained by rearranging terms in the penultimate step, ie,for every $x_i W_{ij}(x_i - x_j)$ there will be a corresponding $x_j W_{ji}(x_j - x_i)$ term, but $W_{ij} = W_{ji}$ as W is symmetric.
Now, since W_{ij} is non negative, $\Sigma_{ij} W_{ij}(x_i - x_j)^2$ is a non-zero value and therefore, since the inner product results in a non-zero value with L being a symmetric(self-adjoint) matrix it follows that L is positive semi definite. This means, that the smallest eigenvalue of L is 0, as it is singular as shown in the previous subdivision. This means that the smallest eigenvalue of the matrix M = ALA is also 0.
Substituting, y= **1** and $\lambda = 0$ in the expressions for the transform computed above, we get, $z = \sqrt{D}y = \sqrt{D}\mathbf{1}$.

$$(ALA)z = \lambda z = 0 = \sqrt{D}^{-1} L \sqrt{D}^{-1} \sqrt{D}\mathbf{1} = \sqrt{D}^{-1} L\mathbf{1} = 0$$

Thus, the eigenvector for the smallest eigenvalue of M is

$$z = \begin{bmatrix} \sqrt{d_{11}} \\ \vdots \\ \sqrt{d_{NN}} \end{bmatrix} \tag{20}$$

(d) (4 points) Now, let's bring back the constraint that $y^T D\mathbf{1} = 0$ (the constraint that y takes only

two discrete values can remain relaxed as a final real-valued solution $\{y_i\}$ can be clustered using 2-means for instance to identify the desired partition). Prove that the GRQ above (subject to $y^T D\mathbf{1} = 0$) is minimized when y is the eigenvector corresponding to the second smallest eigenvalue of the generalized eigenvalue system $(D - W)y = \lambda Dy$.
(Hint: You can use the following fact. Let A be a real symmetric matrix. Under the constraint that x is orthogonal to the $(j-1)$ eigenvectors corresponding to the $(j-1)$ smallest eigenvalues of A, the Rayleigh's quotient $\frac{x^T Ax}{x^T x}$ is minimized when x is the eigenvector corresponding to the j^{th} smallest eigenvalue.)

Solution:
Minimizing GRQ with said constraint:
From **part b**,relaxing the constraint that y takes only two discrete values,the GRQ can be written as,

$$min_y \frac{y^T(D-W)y}{y^T Dy} \text{ subject to } y^T D\mathbf{1} = 0 \tag{21}$$

Let's observe the numerator,denominator and the constraint of the GRQ expression.

- $y^T(D-W)y$**:**

  $y^T(D-W)y = y^T(A^{-1}A)L(AA^{-1})y$ (For some NxN invertible matrix A)
  $\Rightarrow (y^T A^{-1})(ALA)(A^{-1}y)$ (Rearranging terms)
  Employing the transform $A = \sqrt{D}^{-1}$,$z = \sqrt{D}y$ and $M = ALA$
  $\Rightarrow z^T Mz$ (As $(\sqrt{D})^{-1^T} = (\sqrt{D})^{T^{-1}} = \sqrt{D}^{-1}$)

  Therefore,$y^T(D-W)y$ can be represented by $z^T Mz$.

- $y^T Dy$**:**

  $$y^T Dy = y^T\sqrt{D}\sqrt{D}y = (\sqrt{D}y)^T(\sqrt{D}y) = z^T z \text{ (As } \sqrt{D} \text{ is symmetric)}$$

  Therefore,$y^T Dy$ can be represented by $z^T z$.

- $y^T D\mathbf{1} = 0$:

  $$\begin{aligned} &y^T D\mathbf{1} = 0 \\ \Rightarrow &y^T\sqrt{D}\sqrt{D}\mathbf{1} = 0 \\ \Rightarrow &(y\sqrt{D})^T\sqrt{D}\mathbf{1} = 0 \\ \Rightarrow &z^T v_1 = 0 \end{aligned}$$

  Where v_1 is the eigenvector of M corresponding to the smallest eigenvalue computed in the previous subdivision.

Now using these new expressions, 21 becomes,

$$min_z \frac{z^T M z}{z^T z} \text{ subject to } z^T v_1 = 0 \tag{22}$$

Now $M = ALA$ is a real symmetric matrix, as $A = \sqrt{D}^{-1}$ is symmetric and $L = D - W$, a sum of two symmetric matrices is also symmetric.Therefore, $M^T = (ALA)^T = A^T L^T A^T = ALA = M$.
We can observe that, 22 is a constraint minimization problem where the vector z is orthogonal to the eigenvector corresponding to the smallest eigenvalue of M. Employing the hint given in the question, $\frac{z^T M z}{z^T z}$ with the constraint $z \perp v_1$,is minimized when z is the eigenvector corresponding to the second smallest eigenvalue of M.
Reversing the transform obtained in the previous subdivision, we can see that the GRQ is minimized when y is the eigenvector corresponding to the second smallest eigenvalue of the generalized eigenvalue system $(D - W)y = \lambda D y$.

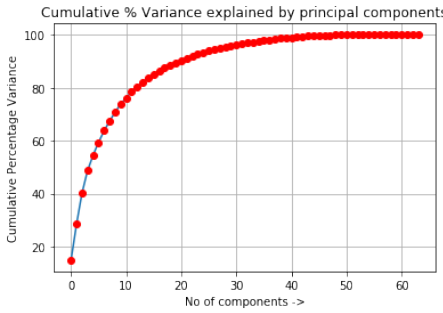
5. (10 points) [LIFE IN LOWER DIMENSIONS...] You are provided with a dataset of 1797 images in a folder here - each image is 8x8 pixels and provided as a feature vector of length 64. You will try your hands at transforming this dataset to a lower-dimensional space, and clustering the images in this reduced 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 GradeScope, 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 Gradescope) 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) (3 points) Run 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.

   Solution:



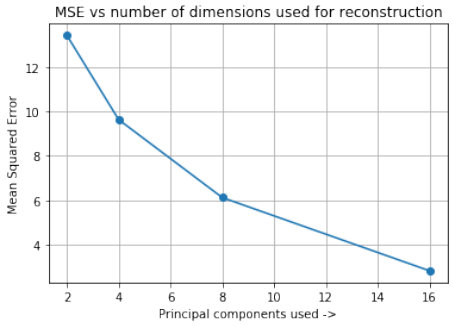
(a)

Figure 4: Cumulative Percentage Variance plotted against 64 components

The first 21 components contribute towards 90.3199% $\approx$ 90% of the variance in the dataset.

(b) (3 points) Perform reconstruction of data using the dimensionality-reduced data considering the number of dimensions [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.

Solution:



(a)

Figure 5: MSE reported for dimensions [2,4,8,16]

- Mean Square Error observed when the reconstructed data is of 2 dimensions = 858.945
- Mean Square Error observed when the reconstructed data is of 4 dimensions = 616.191
- Mean Square Error observed when the reconstructed data is of 8 dimensions = 391.795
- Mean Square Error observed when the reconstructed data is of 16 dimensions = 180.940

Therefore, from the above information,the optimal dimension $\widehat{d}$ is 16 dimensions as it results in the minimum MSE of 180.940 amongst the dimensions. The 16 principal components contribute towards 84.940% $\approx$ 85% variance of the dataset which is good enough to retain the characteristics of the original dataset when reconstructed.

(c) (3 points) Apply K-means clustering on the reduced dataset from last subpart (b) (i.e., the $\mathbb{R}^{64}$ to $\mathbb{R}^{\widehat{d}}$ reduced dataset; pick the initial k points as cluster centers during initialization). Report the optimal choice of K you have made from the set [1...15]. Which method did you choose to find the optimum number of clusters? And explain briefly why you chose that method.
Also, show the 2D scatter plot (consider only the first two dimensions of optimal $\widehat{d}$) of the datapoints based on the cluster predicted by K-means (use different color for each cluster).

Solution:
The optimal number of clusters is 9.
The optimal number of clusters is chosen in a two fold manner.
First the elbow method is employed followed by the silhoutte analysis.

- **The Elbow Method:** We calculate the average WSS (Within cluster sum of square errors) value. This WSS value essentially corresponds to the distance of a datapoint from its respective cluster centroid averaged over the set of datapoints,ie, this is essentially the averaged version of the objective function.The motivation behind this method is as follows, the WSS value reduces with the increase in number of K, however, there comes a point when the decrease in WSS with increase in K diminishes drastically. This point is called the elbow point.Beyond the elbow point, it is not worth taking those many clusters due to the diminished additional returns contributed by the extra clusters as the error stagnates.
- **Silhoutte Analysis:** Accurately identifying the elbow point is hard, therefore, silhoutte analysis can be performed as a check to verify if the optimal cluster number is chosen accurately. The silhoutte score is a measure of how similar a point is to other points from the same cluster when compared to points from the nearest cluster. The silhoutte score is a value between +1 and -1, with a higher value indicating a better fit of datapoints to their respective clusters.

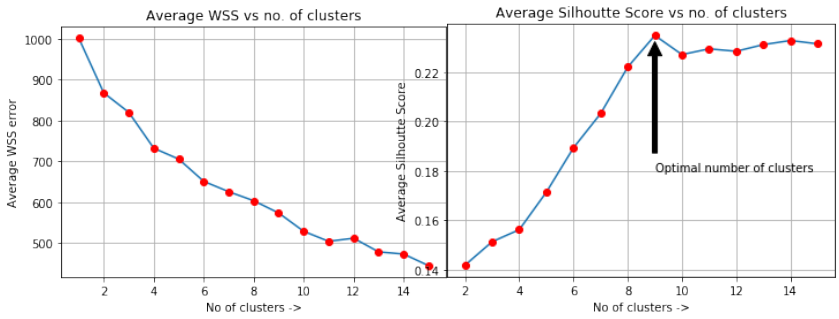


Figure 6: Elbow and Silhoutte methods to identify optimal clusters

From the 6a, we can see that the elbow point is somewhere between 8 and 11. After looking at 6b, we identify the optimal number of clusters as 9.

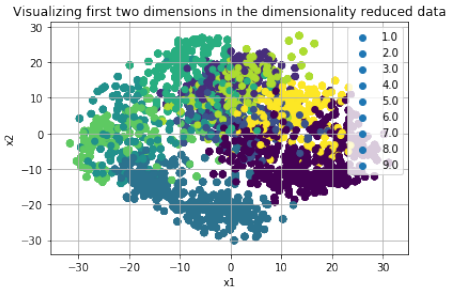


Figure 7: 2D Scatter plot of dimensionality reduced data demarcating different clusters with a different colour

(d) (1 point) Summarise and explain your observations from the above experiments. Is the PCA+K-means clustering consistent with how your brain would cluster the images?

Solution:
Observations:

- We can observe that the mean square error between the reconstructed data after PCA decreases with the increase in number of principal components included.
- We can additionally see that, the rate of change of MSE decreases with increase in the principal components included. This is because later principal components are smaller in magnitude.

- The K-Means algorithm was run only for 50 iterations since it converged very fast.This reassures us of the quickness of the algorithm.

PCA+K-Means:
The PCA+K-Means clustering is not consistent with how our brain would the cluster the images. This is because, we'd expect a convex set of clusters, whereas,as observed in the scatter plot in the previous subdivision,there are points overlapping between clusters. This is because, we are only considering the first two dimensions of the 16 dimensions of the dimensionality reduced data. If we were to include all 16 dimensions,the clusters will be convex and as expected.