IITM-CS5691 : Pattern Recognition and Machine Learning — Release Date: Apr 5, 2021
Assignment 2 — Due Date: Apr 18, 2021 (11.59pm)

Roll No: EE18B035 — Name: Nithin Uppalapati
Collaborators (if any):
References (if any):

- 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 $\{\lambda_i := \sigma_i^2\}_{i=1,\dots,N}$ are called the singular values $\{\sigma_i\}_{i=1,\dots,N}$ of X.)

Solution:
i) Proving that XX^T and X^TX has same eigenvalues.
Assume that λ is an arbitrary non-zero eigenvalue of the XX^T

$$XX^Tu = \lambda u$$

Now, left multiply both sides with X^T

$$X^TXX^Tu = \lambda X^T.u$$

$$(X^TX)X^Tu = \lambda X^Tu$$

X^Tu is eigenvector of X^TX corresponding to the eigenvalue λ
So, we can say that every non-zero eigenvalue of XX^T is an eigenvalue of X^TX.

Now, assume that λ' is an arbitrary non-zero eigenvalue of the X^TX

$$X^TXu = \lambda' u$$

Now, left multiply both sides with X

$$XX^TXu = \lambda' X.u$$

$$(XX^T)Xu = \lambda' Xu$$

Xu is eigenvector of XX^T corresponding to the eigenvalue λ'
So, we can say that every non-zero eigenvalue of X^TX is an eigenvalue of XX^T.

Hence, XX^T and X^TX have same non-zero eigenvalues.

ii) Proving that the eigenvalues are positive:
To show that the eigenvalues are non-negative, for the matrix XX^T: Consider, uXX^Tu^T. This caan be re-written as, $(uX)(X^Tu^T) = (uX)(uX)^T$
As the above expression is norm of (uX), it is always non-negative.
Thus, $uXX^Tu^T \geqslant 0$. So, XX^T is a positive semi-definite matrix. As rank of X is same as rank of XX^T, which is same as that of X^TX, which is N. (Given that the rows of X are linearly independent.)
So by Rank-Nullity theorem, we can say that the nullity of XX^T is,
Nullity (of XX^T) = N (no. of columns) - N (Rank).
Nullity (of XX^T) = 0. Hence XX^T is a positive definite matrix, multiplicity of 0 eigenvalue of the matrix XX^T is 0

Nullity (of X^TX) = D (no. of columns) - N (Rank).
Nullity (of X^TX) = D-N. Hence the multiplicity of 0 eigenvalue of the matrix X^TX is D-N.

(b) (2 points) We can choose the set of eigenvectors $\{u_i\}_{i-=1,...,N}$ of XX^T to be an orthonormal set and similarly we can choose an orthonormal set of eigenvectors $\{v_j\}_{j=1,...,D}$ 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 $i = 1, \dots, N$ 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: As, from the above part, we shown that if u is eigenvector of XX^T, then X^Tu is eigenvector of X^TX. Similarly, we shown that if v is eigenvector of X^TX, then Xv is eigenvector of XX^T.

From the above information, then if we choose the eigenvectors $\{u_i\}_{i-=1,...,N}$ of XX^T to be an orthonormal set, then we can write v in terms of u as follows,

$$v = \frac{X^Tu}{||X^Tu||}$$

Consider ($i \neq j$):

$$v_i^Tv_j = \frac{(X^Tu_i)^T(X^Tu_j)}{||X^Tu_i||.||(X^Tu_j)||}$$

$$v_i^Tv_j = \frac{(u_i^TXX^Tu_j)}{||X^Tu_i||.||(X^Tu_j)||}$$

As u is eigenvector of XX^T, we get:

$$v_i^Tv_j = \frac{(\lambda u_i^Tu_j)}{||X^Tu_i||.||(X^Tu_j)||} = 0$$

Hence, the set of vectors $\{v_i\}_{j=1}^D$ are also orthonormal.

(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: Given that the complexity of finding the eigenvectors of $S = \frac{1}{N}X^TX$, is $O(D^3)$
So, complexity of finding the eigenvectors of $S = \frac{1}{N}XX^T$, would take $O(N^3)$ (As the size of matrix is of NxN)

Let's say these eigenvectors are represented by $\{u_i\}_{i=1}^{N}$
Now, we have to find the vectors $\{v_i\}_{i=1}^{D}$ which are eigenvectors of X^TX.
For finding v_i we have to compute $\frac{X^Tu}{||X^Tu||}$ As the size of the vector u is of N components, so computation of the vector X^Tu has the complexity of O(DN). And as we have to do this for all set of u vectors, the order becomes $O(DN^2)$.
Where as the denominator (norm) takes order O(D), as the dimension of the vector X^Tu is D. So the overall complexity is $O(N^3 + N^2D)$. As given that $D >> N$, thus, the final complexity is $O(N^2D)$.

(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:

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

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 <u>specific solution</u> 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:

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: The aim is to minimize the objective function in deciding to which cluster center should we assign the data point x_n.

The objective function is given as follows:
obj $= \sum_n \sum_k r_k^{(n)}.d(x^{(n)}, m^{(k)})$.
Generally the distance we use is Euclidean Distance, which is, $d(x, m) = \sum_{i=1}^{I}(x_i - m_i)^2$.
Here I is the dimension of the vector x, $x \in \mathbf{R}^I$

But in this problem we consider l_1 distance in the objective function.
l_1 distance is defined as, $d_1(x, m) = \sum_{i=1}^{I} |x_i - m_i|$

Our goal is to assign the data point $x^{(n)}$ to that cluster center $m^(k)$ where, $d_1(x^{(n)}, m^{(k)})$ is minimized. In mathematical terms, Assignment Step:
$r_k^{(m)} = 1$ If, $m^{(k)}$ is the minimizer of $d_1(x^(n), m^(k))$ among all $m^(i)$'s
$r_k^{(m)} = 0$, otherwise

Update Step:
$m^{(k)} = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$
The difference comes in deciding the assignment of data point $x^{(n)}$ to the nearest cluster center, as now the distance metric is different, which is l_1 distance. So, the minimizer will be the median instead of mean (mean in Euclidean distance case). As the minimizer - y of data points $x^{(n)}$ is given by y = median of all the datapoints.

As median is not significantly affected to outliers w.r.t. mean, the l_1 distance metric performs better w.r.t. Euclidean distance metric.

(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:

$$r_k^{(n)} = \frac{\exp\left(-\beta d\left(\mathbf{m}^{(k)}, \mathbf{x}^{(n)}\right)\right)}{\sum_{k'} \exp\left(-\beta d\left(\mathbf{m}^{(k')}, \mathbf{x}^{(n)}\right)\right)}$$

Where, $\beta = \frac{1}{\sigma^2}$

If $\sigma \to 0$, $\beta \to \infty$.

The above equation can be re-written as,

$$r_k^{(n)} = \frac{1}{\sum_{k'} \exp\left(\beta\left(d\left(\mathbf{m}^{(k)}, \mathbf{x}^{(n)}\right) - d\left(\mathbf{m}^{(k')}, \mathbf{x}^{(n)}\right)\right)\right)}$$

Now, we choose a k such that, $d\left(\mathbf{m}^{(k)}, \mathbf{x}^{(n)}\right)$ is minimal among all the possible values of k, let say this k is k_0. And for all the other values of k (other than k_0), $d\left(\mathbf{m}^{(k)}, \mathbf{x}^{(n)}\right)$ will be greater than $d\left(\mathbf{m}^{(k_0)}, \mathbf{x}^{(n)}\right)$

Hence for all k (other than k_0), we get

$$d\left(\mathbf{m}^{(k^0)}, \mathbf{x}^{(n)}\right) - d\left(\mathbf{m}^{(k)}, \mathbf{x}^{(n)}\right) < 0$$

So, for the datapoint $x^{(n)}$, we know that the optimal cluster is $\mathbf{m}^{(k_0)}$ and if we calculate $r_{k^0}^{(n)}$, we get,

$$r_{k^0}^{(n)} = \frac{1}{\exp(\beta.(\text{-ve quantity})) + \ldots + \exp(\beta.(0)) + \ldots + \exp(\beta.(\text{-ve quantity}))}$$

Zero occurs when $k' = k^0$.
So, $r_{k^0}^{(n)} = 1/1 = 1$

Now, if we choose any k which is not k^0, then we get(for datapoint $x^{(n)}$, we end up getting $d\left(\mathbf{m}^{(k)}, \mathbf{x}^{(n)}\right) - d\left(\mathbf{m}^{(k^0)}, \mathbf{x}^{(n)}\right) > 0$. Hence we get at least one of the terms as ∞ in the denominator.
Hence we get $r_k^{(n)} = 1/\infty = 0$, for $k \neq k^0$.

Re-writing the above statements gives us, $r_k^{(m)} = 1$ If, $m^{(k)}$ is the minimizer of $d_1(x^(n), m^(k))$ among all $m^(i)$'s
$r_k^{(m)} = 0$, otherwise

Which is same as Hard K-Means

(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: Clusters are : {1}, {3}, {6,7}.
Means (cluster centers) being $m_1 = 1$, $m_2 = 3$, $m_3 = 6.5$

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: If we initialize the means with, $m_1 = 2$, $m_2 = 6.1$, $m_3 = 6.9$. Then we end up at,
$m_1 = \frac{1+3}{2} = 2$

$m_2 = 6/1 = 6$
$m_3 = 7/1 = 7$
$m_1 = 2$, $m_2 = 6$, $m_3 = 7$, which is not optimal solution, but is a sub-optimal solution.

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: Assume that in one of the cluster set denoted by $\mathcal{K}$ and it's cluster center m_k, and in this minimum and maximum values are denotes by $v_{k,min}$ and $v_{k,max}$, respectively.
Now consider an arbitrary data point x, such that, $v_{k,min} \leqslant x \leqslant v_{k,max}$ and let's assume that the data point x is not there in the cluster set $\mathcal{K}$.

To prove that the claim: "x is not there in the cluster set $\mathcal{K}$" is False

As we said $v_{k,min}$ and $v_{k,max}$ are the min and max values in the cluster set $\mathcal{K}$ so it follows that $v_{k,min}$ and $v_{k,min}$ are closest point to m_k than any other $m_i, i \neq k$. As x is defined such that, $v_{k,min} \leqslant x \leqslant v_{k,max}$ so the closest center lying to x is also m_k.
$v_{k,min} \leqslant x \leqslant v_{k,max}$
$d(x - m_k) \leqslant d(v_{k,max} - m_k)$
$d(x - m_k) \leqslant d(v_{k,min} - m_k)$

So, x also lies in cluster $\mathcal{K}$
So, the claim that x lies in cluster other than that of $\mathcal{K}$ is False.

Hence the clustering partitions are contiguous.

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

Solution:

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,

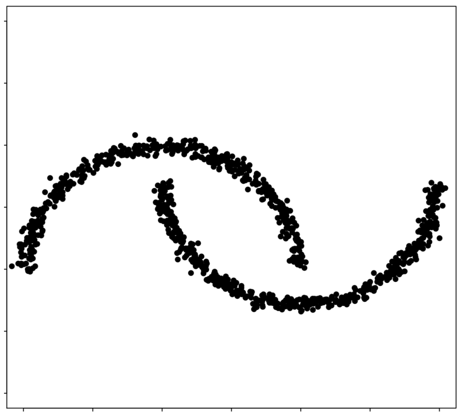
$$\frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \|a_i - b_j\|$$

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:
Out of the three metrics mentioned, I would choose the 3^{rd} one, which is **Average distance**. The reason for doing so is, if we consider the Minimum distance metric, there may be outliers/exceptions in dataset which causes the "outlier/exception" to be included in the dataset. Whereas, if we consider the Maximum distance metric, the nearer points points may not be grouped into a cluster, hence a good cluster may not always form. Hence Average distance metric is preferred over the other two.

(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)

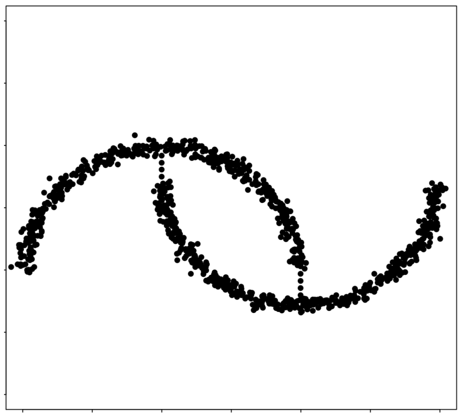

(b)

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

Solution:
Figure 1a: For this dataset, we use Minimum distance metric, as all the points in one of the crest will merge into a single cluster (as desired) if and only if the distance metric is Minimum type. The reason is the gap between the two crests is larger than the distances between the points in a given crest, hence if the number of specified clusters are two, then we finally get the two crests as a separate cluster, if we chose the Minimum distance as the metric for hierarchical clustering.

Figure 1b: In this case, the answer is None of the above metrics will be able to separate the cluster as specified.
The answer is None, because in this case the data is such that, the intermediate points joining the two "moon crests" has the intermediate distance comparable to that of the intermediate distances between the points which are lying within the crests. Hence minimum metric won't meet the expectations.
If we choose avg. metric, then we get thee boundary something like a diagonal line. So, avg. metric also won't meet the expectations.
If we choose max. metric, the clusters formed will be non-intuitive, hence none of the above.

(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:

First Step:

	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

Second Step:

	{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

Third Step:

	{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

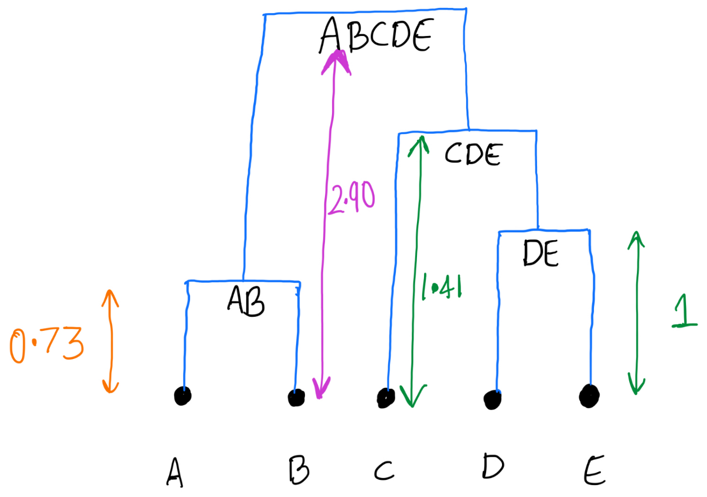
Fourth Step:

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

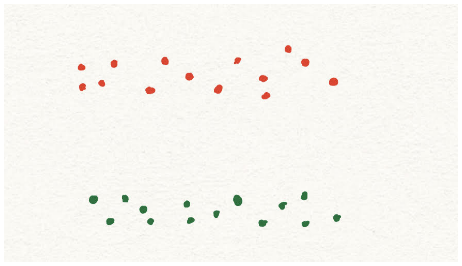
Fifth Step:

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

The final dendogram is shown below,



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



(a)

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

Solution: Using Minimum metric the above cluster is more likely to get generated. If we consider the Minimum metric, as the distance between two neighbouring points in a given cluster are smaller than that of the distance between the points lying in two different clusters, so it is more likely that the nearer points will get combined into a single cluster as shown in figure. Hence Minimum cluster is more likely to generate the above cluster for that dataset.

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: One of the disadvantage is that the *cut cost* is not comparable, i.e., it is not normalised. Consider if one of the partition's size increases, accordingly the number of links among the partitions increases, increasing the cut cost.
   Another disadvantage is that, the optimal way to minimize the cost (among N no. of data points) is to partition a single data point such that the sum of $N-1$ links to the single data point is minimum, which is in-accurate.

   (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 **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: Given that D is defined as a diagonal matrix, whose entries are given as, $d_{ii} = \Sigma_j W_{ij}$.
So, If we consider the matrix $D - W$,
Consider row transformation of matrix $D - W$,
$D - W = [c_1, c_2, \ldots, c_N]$
Where, $c_i^T = i^{th}$ column of D - i^{th} column of W.
If we take sum of all elements in a column, then the sum will be 0.
Consider a column of D. The only non zero element in the i^{th} column of D is d_{ii}. Similarly consider the i^{th} column of W. The sum of all elements in this column is $\Sigma_j W_{ij}$, and this is the i^{th} element in the diagonal matrix D.
Hence, when subtracted, the net sum of all elements in a column of the matrix D-W is 0.
Hence, the Laplacian matrix (D-W) is 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: Consider $(D - W)y = \lambda Dy$.
As the elements of matrix D are positive, the matrix D is invertible. (As the edge weights are positive)

$$(D - W).D^{-1}.D.y = \lambda Dy$$

Now, multiply $\left\{D^{1/2} = (\frac{1}{\sqrt{d}})I\right\}$ from left side on both sides of equation.

$$D^{-1/2}.(D - W).D^{-1/2}.D^{1/2}.y = \lambda D^{-1/2}.Dy$$

$$D^{-1/2}.(D - W).D^{-1/2}.D^{1/2}.y = \lambda D^{1/2}y$$

So, from above equation we get (by comparison), $A = D^{-1/2}$ and $z = D^{1/2}y$

Now, for eigenvector of the smallest eigenvalue, we have to find the smallest eigenvalue corresponding to matrix, ALA.

$$ALA = D^{-1/2}.(D - W).D^{-1/2}$$

Positive-Semi definiteness check of ALA:
As from earlier part, we proved that det(L) = 0.

As we know, L is positive semi-definite matrix, and $u^T.ALA.u$ is same as $(Au)^T L(L)^{Au}$. Assume that $Au = w$, and we know that $u^T.L.u \geqslant 0$ (As L is a positive semi-definite matrix), so we can also say that $w^T.L.w \geqslant 0$

So, we get $|A|.|L|.|A| = 0$
As, $|ALA| = 0$, the expression $|ALA - \lambda I| = 0$ get satisfied for $\lambda = 0$
So, the smallest eigenvalue is 0.

Computing the eigenvector corresponding to $\lambda = 0$:
From the above part, we know that the eigenvector of the Laplacian matrix is **1**. (As adding all the columns of the Laplacian matrix, gives us a **0** vector.
Now, we have to solve for z, which satisfies, $ALAz = 0$

$$D^{-1/2}.(L).D^{-1/2}z = 0$$

$$L.D^{-1/2}z = 0$$

Which means that, $D^{-1/2}z$ is null vector of thee Laplacian Matrix. So, $D^{-1/2}z = \mathbf{1}$
Which gives us, $z = D^{1/2} = (\sqrt{d})I$

(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 $(D - W)y = \lambda D y$.
(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 $\frac{x^T A x}{x^T x}$ is minimized when x is the eigenvector corresponding to the j^{th} smallest eigenvalue.)

Solution: Let us re-write GRQ in terms of Rayleigh's quotient.

$$\mathsf{GRQ} = \frac{y^T(D-W)y}{y^T D y}$$

$$\mathsf{GRQ} = \frac{y^T D^{+1/2} D^{-1/2}(D-W) D^{-1/2} D^{+1/2} y}{y^T D^{1/2} D^{1/2} y}$$

Let $x = D^{1/2} y$,

$$\mathsf{GRQ} = \frac{x^T D^{-1/2}(D-W) D^{-1/2} x}{x^T x}$$

When we compare the above expression with Rayleigh's quotient we get,
$A = D^{-1/2}(D-W)D^{-1/2} and dx = D^{1/2}y$
Here A is real symmetric matrix, as D and W are symmetric matrices, A is symmetric matrix. Also A is real values, because the entries in D and W are real and positive valued.

Also consider the equation - $(D-W)y = \lambda D y$. If we re-write this equation in terms of A and x, we get,
(Multiply $D^{-1/2}$ from left side on both sides of equation.)

$$D^{-1/2}(D-W)(D^{-1/2}x) = \lambda D^{-1/2}(y)$$

$$Ax = \lambda x$$

So, we can say that A and L have same eigenvalues.
The condition that y must satisfy is: $y^T D \mathbf{1} = 0$
Re-writing it in terms of x gives: $x^T D^{1/2} \mathbf{1} = 0$
So, x is orthogonal to the vector, $D^{1/2}$, which is the eigenvector of A (M in above part) corresponding to the eigenvalue $\lambda = 0$. So if we find a eigenvector corresponding to j=2, then that will be the minimiser to the Rayleigh's quotient. (As we proved above that it satisfies the necessary conditions to be a minimiser of the Rayleigh's quotient). Which is,

$$Ax = \lambda' x$$

And this x is minimiser of Rayleigh's quotient.

As also we proved above that A and L {(D-W)} have same set of eigenvalues. So, correspondingly, the eigenvector corresponding to λ' (which is the second smallest eigenvalue) is the minimiser of the GRQ Quotient.

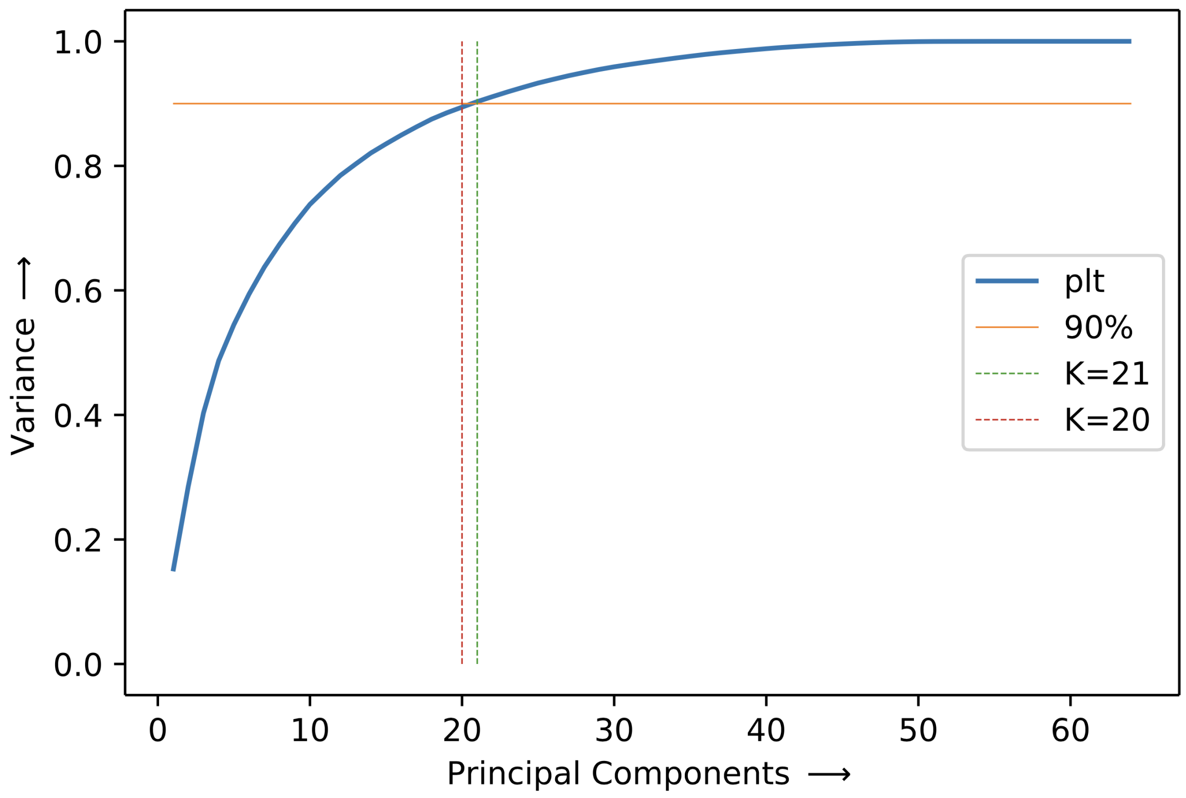
5. (10 points) [LIFE IN LOWER DIMENSIONS...] You are provided with a dataset of 1797 images in a folder here - each image is 8x8 pixels and provided as a feature vector of length 64. You will try your hands at transforming this dataset to a lower-dimensional space, and clustering the images in this reduced 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: The number of components that contribute to 90% is 21.



   (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: Mean-Squared Values Table:

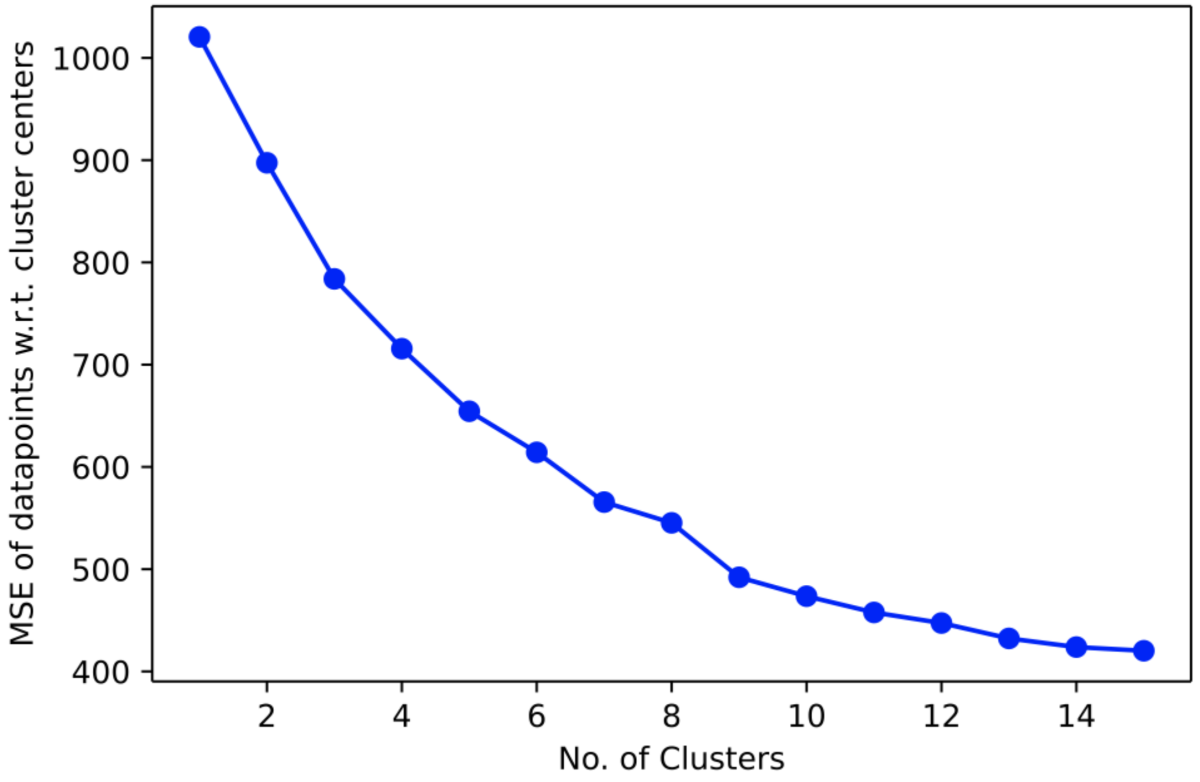
Dimensions	2	4	8	16
MSE	858.9447	616.19	391.79	180.939

As, the MSE is lesser for d=16, (among d=[2,4,8,16]). So, the optimal $\widehat{d}$, I would choose is,

$$\widehat{d} = 16$$

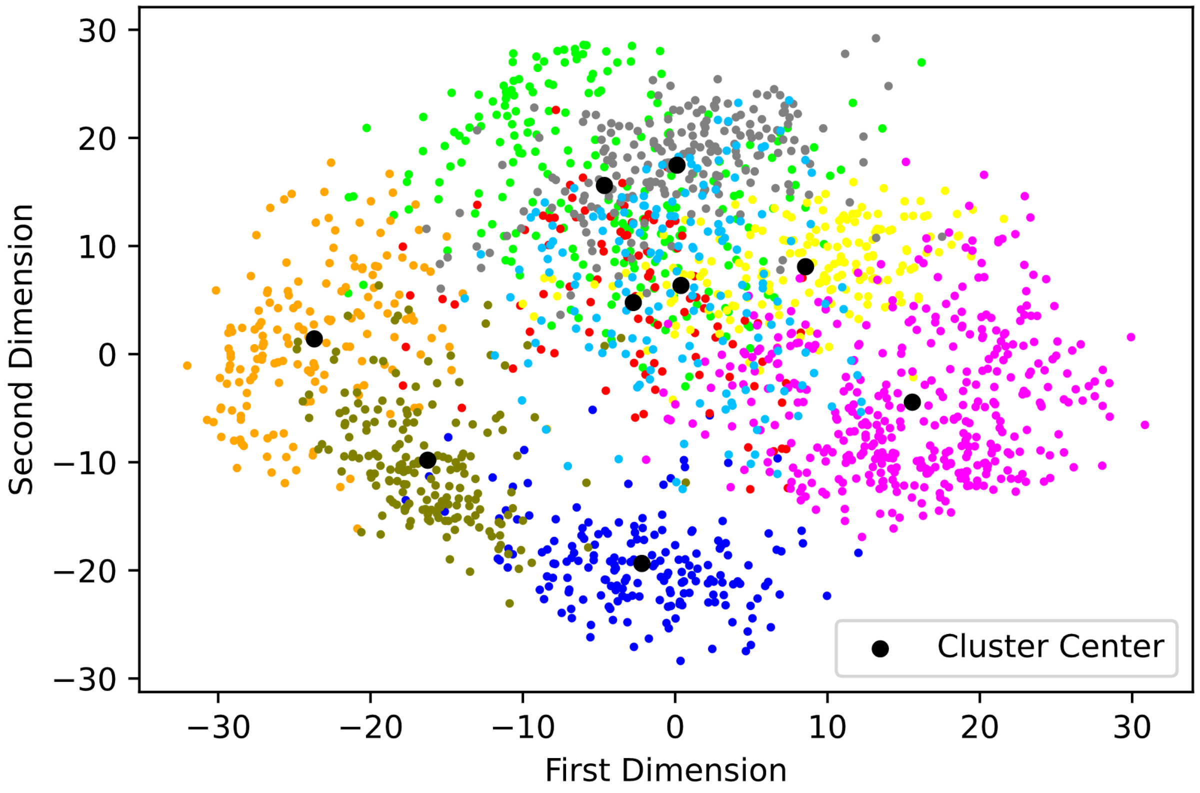
(c) (3 points) Apply K-means clustering on the reduced dataset from last subpart (b) (i.e., the $\mathbb{R}^{64}$ to $\mathbb{R}^{\widehat{d}}$ reduced dataset; pick the initial k points as cluster centers during initialization). Report the optimal choice of K you have made from the set [1...15]. Which 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 data points based on the cluster predicted by K-means (use different color for each cluster).

Solution:
Below is thee plot for MSE (w.r.t. cluster centers) vs the number of clusters.



We observe a steep from k=8 to k=9, which indicates that k=9 cluster is more optimal when compared to that of k=8.
But, if we analyse the graph from k=9 to k=10, wee observe not much steep decrease, and rather a linear fashion of decreasing monotonicity, with lesser slope.
So, by Elbow method, we can arrive at a conclusion that k=9 is the optimal number of clusters.

Below is the scatter plot for 9 clusters of the dataset.

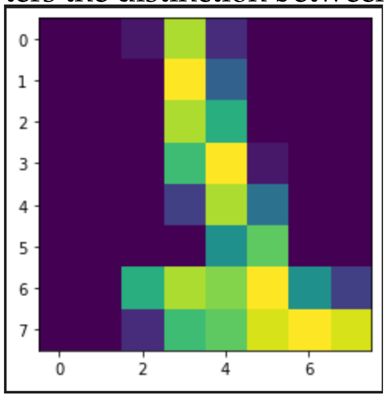


The black dots are the cluster centers

(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: From the above part, although we get the optimum no. of clusters as k=9, I intuitively think that the optimum no. of clusters should be k=10.
The reason is, as there are 10 digits in total, which are [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] and ideally there should be 10 clusters. The probable reason for why we got 9 optimal clusters is, maybe there are few ill-defined (not properly specified numbers) numbers in the dataset

(for example, please see the image below). And maybe due to these kind of noisy characters the distinction between the numbers is lost.



For Example, The above image looks like a rotated and mirrored image of 7. But my algorithm doesn't check for rotated or mirrored images, hence the noise in clusters.

Yes, PCA+K-means clustering is consistent with how my intuition works. (Verified by printing the images in all cluster sets.)