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

Roll No: EE19B094 — Name: Manvar Nisharg
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 $\{\lambda_i := \sigma_i^2\}_{i=1,\ldots,N}$ are called the singular values $\{\sigma_i\}_{i=1,\ldots,N}$ of X.)

Solution: For $X^T X$, from the definition of eigenvectors (ϕ) and eigenvalues (λ),

$$\begin{aligned}(X^T X)\phi &= \lambda\phi \\ X(X^T X)\phi &= X\lambda\phi \\ XX^T(X\phi) &= \lambda(X\phi)\end{aligned}$$

Thus, XX^T and $X^T X$ have same set of non-zero eigenvalues.

Now, from the definition of co-variance matrix we know it is symmetric and real and also from the following,

$$\begin{aligned}S &= E\{XX^T\} \\ u^T S u &= u^T E\{XX^T\} u \\ &= E\{u^T XX^T u\} \\ &= E\{v^2\} \\ &= \sigma_v^2\end{aligned}$$

where, σ_v^2 is the variance of the random variable v.

Now, as $u^T S u$ is always positive, we can say S is always positive semi-definite. Also, as rows of X are linearly independent S would be positive definite. Thus all eigenvalues of the matrix will be positive and as order of XX^T is $N \times N$, there will be N eigenvalues. Also, as all eigenvalues of XX^T are positive, multiplicity of zero eigenvalue for it is zero. While for $X^T X$, it has the same set of non-zero eigenvalues as XX^T, but it has total D eigenvalues, thus rest D-N eigenvalues are zero.

(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 $X^T X$. 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 $X^T X$; 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: As proved in the last question, S is real and symmetric. Now, we prove that eigenvectors of a real and symmetric matrix are orthogonal.
Let u,v be eigenvectors corresponding to α, β, respectively. Namely we have Su=αu and Sv=βv. Now, let us calculate α(u· v),

$$\begin{aligned}
\alpha(u \cdot v) &= (\alpha u) \cdot v \\
&= Su \cdot v = (Su)^T v \\
&= u^T S^T v = u^T S v \\
u^T \beta v &= \beta(u^T v) \\
&= \beta(u \cdot v) \\
\therefore (\alpha - \beta)(u \cdot v) &= 0 \\
\therefore u \cdot v &= 0
\end{aligned}$$

Now, from the last question we saw that, for eigenvector u_i and eigenvalue λ of XX^T

$$\begin{aligned}
XX^T(u_i) &= \lambda(u_i) \\
X^T X X^T(u_i) &= X^T \lambda(u_i) \\
X^T X(X^T u_i) &= \lambda(X^T u_i)
\end{aligned}$$

Thus, the eigenvetor v_i of $X^T X$ can be computed as $X^T u_i$ i.e. directly from u_i and X without having to do eigenvalue decomposition of $X^T X$.

(c) (2 points) Applying PCA on the matrix X would be computationally difficult as it would involve finding the eigenvectors of $S = \frac{1}{N} X^T X$, 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: If we use SVD to decompose the eigenvectors of the covariance matrix, then for a matrix A of size MxN, SVD(A) costs $\mathcal{O}(M^2N + MN^2 + N^3)$. Thus, if we take A = $X^T X$ whose size will be, DxD and thus SVD will take $\mathcal{O}(D^3)$ time.

However we saw that the eigenvalues of $X^T X$ and XX^T are same and eigenvectors can also be calculated from eigenvalues of XX^T. And as size of XX^T is NxN, it's SVD will be $\mathcal{O}(N^3)$ and thus we do SVD on XX^T.

Now for the eigenvetors of X^TX, we multiply the eigenvectors obtained from SVD of XX^T by X^T, which we take $\mathcal{O}(DN^2)$. Finally to normalize this DxN matrix, will take $\mathcal{O}(DN)$. Thus the overall time taken to calculate the eigenvectors of X^TX is $\mathcal{O}(N^3 + DN^2 + DN)$ i.e. $\mathcal{O}(N^3 + DN^2)$.

(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:

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

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: Firstly, as the gradient of Lagrangian with respect to H is implicitly assumed to be zero, so that the constraint that U is orthonormal holds for both the specific and general solutions.
Secondly, $S\widehat{U} = \widehat{U}H$ i.e. $\widehat{U^T}S\widehat{U} = H$. Now, as S is symmetric, H will be symmetric too as $(\widehat{U^T}S\widehat{U})^T = \widehat{U^T}S\widehat{U}$
Thirdly, if we substitute $S\widehat{U} = \widehat{U}H$ back into $\widetilde{J}$, we get

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

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

Now lets consider the case of specific solution of $S\widehat{U} = \widehat{U}H$ i.e. where H is a diagonal matrix of eigenvalues.

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

$$= \textit{Sum of eigenvalues of S}$$

Now in the case for general solution of $S\widehat{U} = \widehat{U}H$, we have already seen that H is symmetric so H is diagonalizable $(H = P^{-1}DP)$. So, the equation becomes

$$S\widehat{U} = \widehat{U}P^{-1}DP$$

$$S\widehat{U}P^{-1} = \widehat{U}P^{-1}D$$

Now, expanding this equation column wise, if v_i is the column of $\widehat{U}P^{-1}$ and u_i is the corresponding i^{th} diagonal element of D, then we get $Sv_i = v_i u_i$, which implies that u_i is eigenvalue of S. Thus, D is a diagonal matrix of eigenvalues of S.

Now again the value of $\widehat{J}$ will be

$$\widetilde{J} = Tr\{H\}$$
$$\widetilde{J} = Tr\left\{P^{-1}DP\right\}$$
$$\widetilde{J} = Tr\{D\}$$
$$= \text{Sum of eigenvalues of } S$$

Hence, both cases give same value of $\widetilde{J}$.

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 basic idea of the algorithm remains same i.e. for every point assign a cluster whose cluster centre is closest to it and then in the next step update the cluster centres. However this time, we do not update the cluster centres to the mean/centroid of the data points but instead to the median of the points (dimension wise).

$$\text{Assignment} : r_k^{(n)} = \begin{cases} 1 & \text{if } m^{(k)} \text{ if closest to } x^{(n)} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Update} : m_i^{(k)} = \text{median}(x_i^{(n)}) \text{ for all dimensions } i \; \forall x^{(n)} \text{ whose } r_k^{(n)} = 1$$

We will definitely prefer using ℓ_1 norm if our data contains outliers than ℓ_2 norm. Robustness is defined as resistance to outliers in a dataset. The more able a model is to ignore extreme values in the data, the more robust it is.
The ℓ_1 norm is more robust than the ℓ_2 norm, for fairly obvious reasons: the ℓ_2 norm squares values, so it increases the cost of outliers exponentially while the ℓ_1 norm only takes the absolute value, so it considers them linearly.

   (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: For, the given case of soft K-means, the assignment and the updates steps are,

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

where $\beta = \frac{1}{2\sigma^2}$

$$Update : \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$$

Now, the only difference between this algorithm and hard K-means is that here the assignment is soft i.e. any value between 0 to 1 for all clusters, whereas in the hard K-means it is always 0 or 1.

Now, lets assume $\sigma \to 0$ i.e. $\beta \to \infty$

Now in the assignment step, let $m^{(M)}$ be the nearest cluster to $x^{(n)}$, then dividing numerator and denominator by $exp(-\beta d(m^{(M)}, x^{(n)}))$, then the assignment step becomes,

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

Clearly, $exp(-\beta(d(m^{(k)}, x^{(n)}) - d(m^{(M)}, x^{(n)})) = 0$ if $k \neq M$.

Thus,

$$r_k^{(n)} = \frac{exp(-\beta(d(m^{(k)}, x^{(n)}) - d(m^{(M)}, x^{(n)}))}{\sum_{k'} exp(-\beta(d(m^{(k')}, x^{(n)}) - d(m^{(M)}, x^{(n)}))} = 0 \quad when\ k \neq M$$

$$r_k^{(n)} = \frac{exp(-\beta(d(m^{(k)}, x^{(n)}) - d(m^{(M)}, x^{(n)}))}{\sum_{k'} exp(-\beta(d(m^{(k')}, x^{(n)}) - d(m^{(M)}, x^{(n)}))} = 1 \quad when\ k = M$$

Thus, here too we have a hard assignment i.e. the assignment step either assigns a value of 0 or 1 which is same as assignment step of hard K-means And as the Update step was already same as hard K-Means, we proved that as $\sigma \to 0$, soft K-means becomes 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: As n=4 and k=3, we have to pair two points, and as x_3 and x_4 are closest, pairing them would be most optimum. Thus the clusters are, (1) (3) (6,7)

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: Suppose we initalize our cluster centres to 2 , 6 and 7. Then our clusters will be (1,3) (6) (7) and as we can see all data points are closest to their cluster centres then to others, and thus there be no more updates and this is the final form.

However, clearly this is not the most optimum solution as the most optimum solution is the one discussed in the previous question. The value of the objective function in the previous case was $2 * (0.5)^2 = 0.5$ whereas in this case it is $2 * (1)^2 = 2$.

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: Let us suppose the centres for the optimal k-means algorithms are $C_1 \leqslant C_2 \leqslant \cdots \leqslant C_m$. Now, let us assume a point x_i assigned to a cluster with cluster centre C_i. Now, we need to prove that all points between C_i and x_i would be assigned to C_i as well. We shall do this by contradiction, suppose any point x_j between x_i and C_i is assigned to cluster centre C_j. This for any optimal k-means clustering would imply that x_j is closer to C_j than to C_i.

There are three cases possible,

Case-1
C_j lies between C_i and x_i. This however, would mean that the order of points are $C_i \ldots C_j \ldots x_j \ldots x_i$ or $C_i \ldots x_j \ldots C_j \ldots x_i$ either in increasing or decreasing order. This, would then contradict our initial assumption that C_i is the closest cluster center to x_i as clearly C_j is more closer. Therefore this case is ruled out.

Case-2
C_j lies outside the range of C_i and x_i on the side of x_i. Thus, the order of points would be $C_i \ldots x_j \ldots x_i \ldots C_j$. Now, as x_j is assigned to C_j over C_i would mean that x_j is closer

to C_j than to C_i. Now, let us consider the point x_i, as it lies between x_j and C_j, the distance it to from C_j would be less than the distance from C_j to x_j. Thus, if C_j is the closest centre to x_j it must be even closer for x_i. However again it contradicts our initial assumption of C_i being the closest centre to x_i.

Case-3
C_j lies outside the range of C_i and x_i on the side of C_i. Thus, the order of points would be $C_j \dots C_i \dots x_j \dots x_i$. Now, here as C_i lies between C_j and x_j, clearly x_j is more closer to C_i than C_j, thus any optimal k-means algorithm would not assign x_j to C_j over C_i, thus this case is ruled out as well.

Thus, we proved that if a point x_i is assigned to C_i then all points between them must be assigned to C_i as well. Now if we choose x_i to be the farthest points from C_i on either side to be assigned to C_i, all points between them must be assigned to C_i as well and thus all clusters would be continuous.

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

Solution: Let us first define the sum of square of distance between points and the cluster centre for all cluster as closeness. And let $d(x_j, \dots, x_i)$ be the sum of squared distances to their means for $x_j \dots x_i$

Now, as in all dynamic programming algorithms we then define a sub-problem as finding the minimum closeness of clustering $x_1, \dots, x_i$ into m clusters. We store the closeness for that case in a matrix D of size n+1 by k+1 in entry D[i,m]. The matrix is initialized as D[i,m] = 0, when m = 0 or i = 0. Thus D[n, k] entry denotes the minimum closeness value to the original problem. Let j be the index of the smallest number in cluster m (last cluster) in an optimal solution to D[i,m]. It is evident that D[j 1,m 1] must be the optimal closeness for the first j 1 points in m 1 clusters, for otherwise one would have a better solution to D[i,m]. This then leads us to the recursive relation

$$D[i, m] = \min_{m \leqslant j \leqslant i} \{D[j-1, m-1] + d(x_j, \dots, x_i)\}$$

Here note that $d(x_j, \dots, x_i)$ can be calculated in constant time from $d(x_{j+1}, \dots, x_i)$ already calculated before in the recursion. Thus, each entry requires $O(n^2)$ time and thus total time would be $O(kn^2)$.

To find exact points of cluster centres corresponding to the minimum closeness of

D[n.k], let us define a matrix A of size n by k to record the index of smallest number in cluster m.

$$A[i, m] = \underset{m \leqslant j \leqslant i}{\operatorname{argmin}} \{D[j-1, m-1] + d(x_j, \ldots, x_i)\}$$

Then we backtrack from A[n, k] to obtain the starting and ending indices for all clusters. This can be done in $O(k)$ time.

Thus the overall time is $O(kn^2 + k) = O(kn^2)$ time.

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,

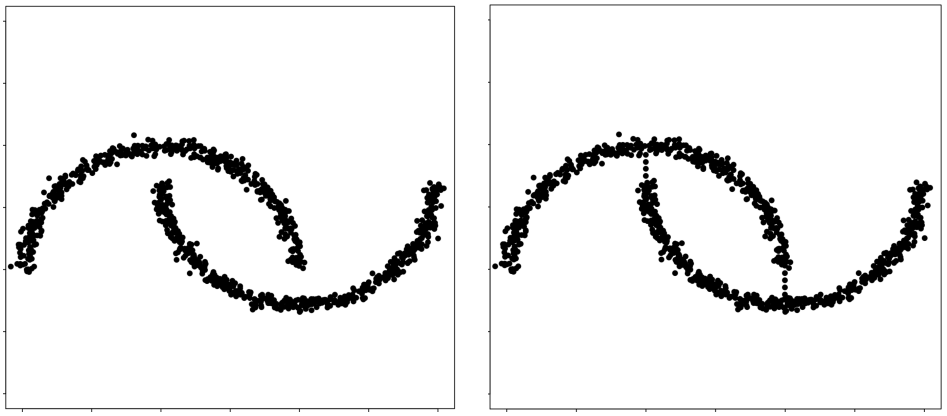
     $$\frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \|a_i - b_j\|$$

   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: The K-means algorithm while updating its cluster centres, takes the centroid of the points assigned to that cluster as the new cluster centre. Thus, it takes the average of the position values across dimensions. The mean distance metric in the case of hierarchical clustering uses a similar technique albeit the average distance in K-means is between points intra cluster whereas in the hierarchical clustering it is inter-cluster. However, the max and min distance metrics are extreme versions and disregard the average distribution which is at the core of K-means, therefore the average distance metric would give the most close clusters to K-means although they might differ significantly in specific cases.

(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: For the figure 1a, intuitively there are two clusters and distance between any point and it's closest neighbour inside a cluster is less than minimum distance between the clusters (the gap distance) i.e. intra-distance between closest points in a cluster is less than the minimum inter-distance between two clusters. Thus, the minimum distance metric of distance would give the best result.

Whereas in figure 1b, the above arguments fails as the extra added points, make the inter-cluster distance almost as small as intra-distance cluster. Thus, none of the metrics would actually give desirable clustering.

(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.)

Solution:

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

We begin clustering nodes with minimum distances between them and keep updating the distance matrix.

cluster 1 - merge A and B

	AB	C	D	E
AB	0	4.95	2.9	3.45
C	4.95	0	2.24	1.41
D	2.90	2.24	0	1
E	3.45	1.41	1	0

cluster 2 - merge D and E

	AB	C	DE
AB	0	4.95	2.9
C	4.95	0	1.41
DE	2.90	1.41	0

cluster 3 - merge C and DE

	AB	CDE
AB	0	2.9
CDE	2.9	0

cluster 4 - merge AB and CDE

	ABCDE
ABCDE	0

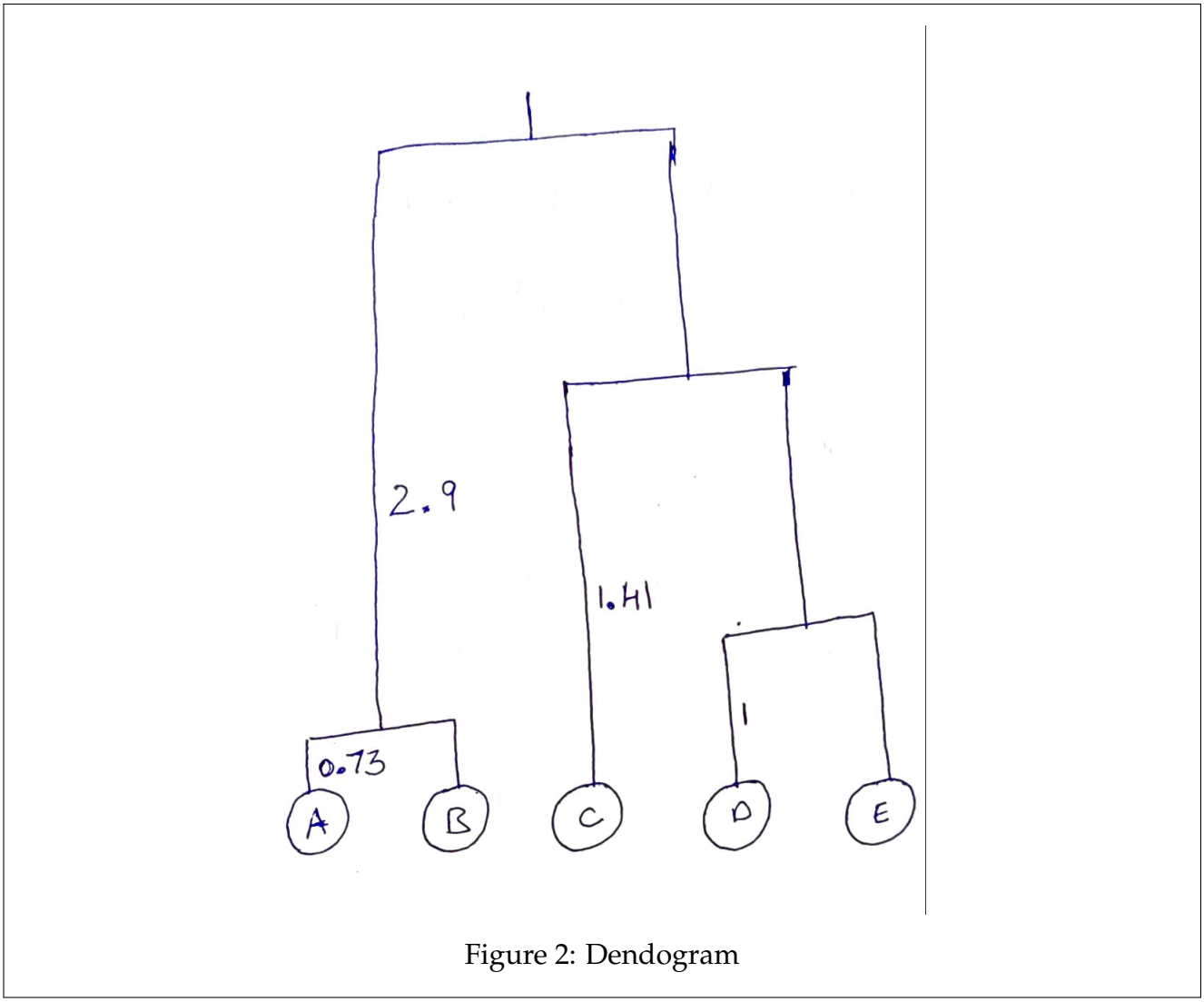
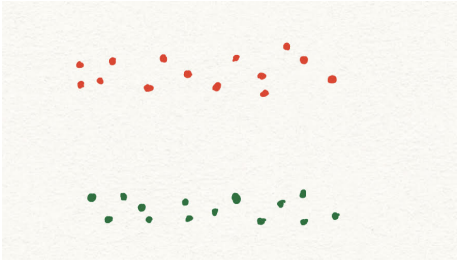


Figure 2: Dendogram

(d) (2 points) Which distance metric (minimum, maximum and average) is more likely to generate following results given in Figure ?? 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: Similar to part B of this question, here too the distance between points and their closest neighbour in all clusters are less than the minimum distance between two clusters i.e. intra-distance between closest points in a cluster is less than the minimum inter-distance between two clusters. Thus, the minimum distance metric of distance would give the desired result.

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: Clearly in this definition of cut cost, as the cut cost increases with increasing number of edges passing through the partitions, this method will inherently favour partitioning small sets of points which are relatively isolated. Thus often this technique would give partitions whose size differ greatly.

   (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:

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

   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: For any row i in the Laplacian matrix, its entries would be $(-w_{i1}, -w_{i2}, \ldots, (\sum_{n=1}^{n=N} w_{in}) - w_{ii}, \ldots, -w_{iN})$.

Now we see that sum of all elements along a row is 0. Thus if, we replace any column in the matrix with sum of all the columns (i.e. $C_i \rightarrow C_1 + C_2 + \cdots + C_N$), we know that the determinant value of the matrix won't change, and that column will have all elements 0, and thus matrix is singular.

(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: We have the equation

$$(D - W)y = \lambda Dy$$

. If we substitute $y = D^{-1/2}z$ then the equation becomes,

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

Now, we proved in class that L is positive semi-definite. Also as D is a diagonal matrix of degrees of vertices which are always non-negative, thus D is a diagonal positive semi definite matrix. And thus so would be $D^{-1/2}$.
Now if we calculate $v^T D^{-1/2} L D^{-1/2} v$

$$v^T D^{-1/2} L D^{-1/2} v = (D^{-1/2}v)^T L (D^{-1/2}v) \geqslant 0 \quad \text{for any } v$$

Thus, $D^{-1/2}LD^{-1/2}$ will also be symmetric positive semi-definite.

Now as we saw above the sum of L along rows is 0, thus $(D - W)\mathbf{1}$ will be $\mathbf{0}$.
Thus, from this we can see that $z = D^{1/2}\mathbf{1}$ will be a eigenvector of $D^{-1/2}LD^{-1/2}$ with eigenvalue 0. Also, as we know that $D^{-1/2}LD^{-1/2}$ is symmetric positive semi-definite, this corresponds to the eigenvector corresponding to the smallest eigenvalue.

(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: Now, let $y_0 = D^{1/2}\mathbf{1}$ be the eigenvector corresponding to the smallest eigenvalue and y_1 be corresponding to the second smallest. Also, as we substituted $y = D^{-1/2}z$, the correspondng eigenvectors of $(D - W)y = \lambda Dy$ are $y_0 = D^{-1/2}z_0 = \mathbf{1}$ and $y_1 = D^{-1/2}z_1$. Now, we know that $D^{-1/2}LD^{-1/2}$ is symmetric and thus its eigencevtors will be orthogonal i.e.

$$z_1^T z_0 = 0$$

$$y_1^T D^{1/2} D^{1/2} y_0 = 0$$

$$y_1^T D\mathbf{1} = 0$$

Now from the fact that if 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, then z_1 satisfies,

$$z_1 = min_{z_1^T z_0 = 0} \quad \frac{z^T D^{-1/2}(D - W)D^{1/2}z}{z^T z}$$

$$\therefore y_1 = min_{y^T D\mathbf{1} = 0} \quad \frac{y^T (D - W)y}{y^T Dy}$$

which is the condition we desire. Thus we proved that the eigenvector corresponding to the second smallest minimizes the GRQ constraint to $y^T D\mathbf{1} = 0$.

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 first 21 dimensions with the highest eigenvalues, contribute to 90% of the variance.

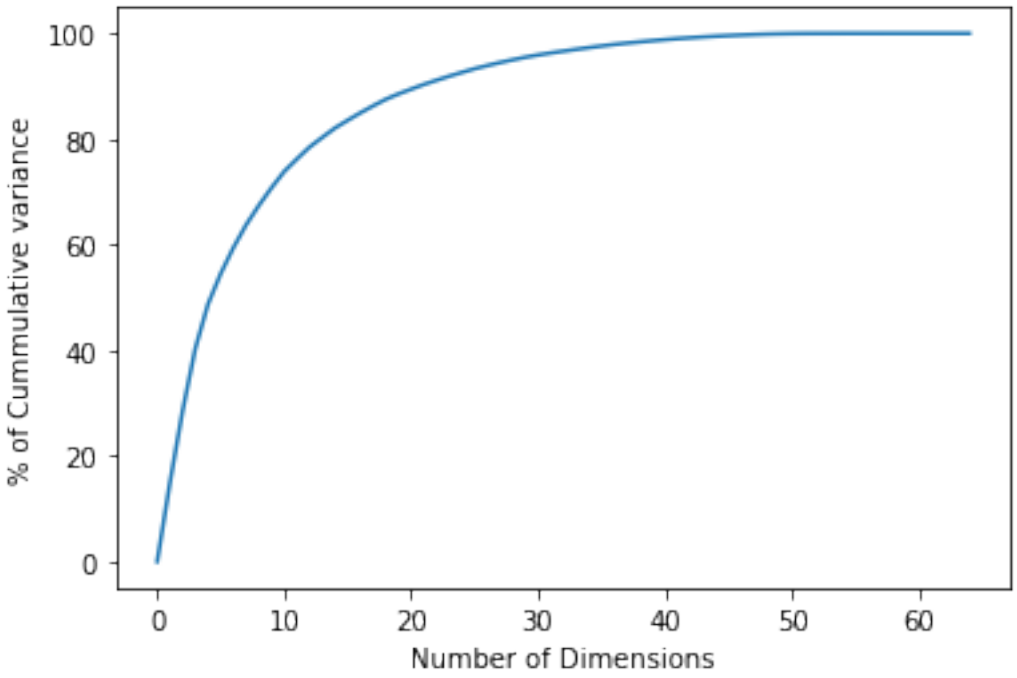


Figure 4: % cumulative variance vs number of dimensions

(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 $\hat{d}$ based on the MSE values.

Solution:

```
Dimension : MSE
2 : 858.9447808487348
3 : 717.2352446162687
4 : 616.1911300562691
5 : 546.7166473621047
6 : 487.6410153666709
8 : 391.79473611497616
12 : 258.70583438481384
16 : 180.9397032573787
24 : 88.82087637084612
32 : 40.42470493509364
```

```
40 : 14.174164665139777
50 : 0.5441328712389208
```

(c) (3 points) Apply K-means clustering on the reduced dataset from last subpart (b) (i.e., the $\mathbb{R}^{64}$ to $\mathbb{R}^{\hat{d}}$ 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 briefy why you chose that method.
Also, show the 2D scatter plot (consider only the first two dimensions of optimal $\hat{d}$) of the datapoints based on the cluster predicted by K-means (use different color for each cluster).

Solution:

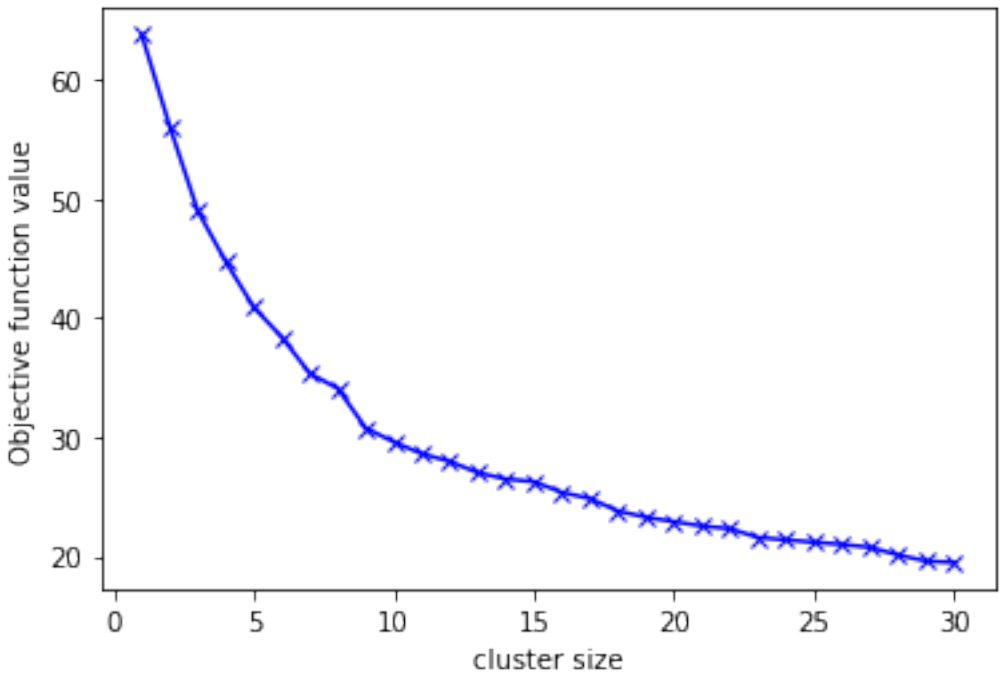


Figure 5: Objective function value of K-means Vs Cluster size

To decide the optimal value of cluster we use the elbow method. It works on the principle that initially the objective function decreases rapidly with cluster size but then it gets flatter after one point. That point is called the elbow point and we choose this point as our optimal cluster size as increase in cluster size further wouldn't affect the objective function much.
From the graph above, the elbow point is around n = 9.
Thus, for n = 9, the clustered points plotted considering only the first two dimensions are,

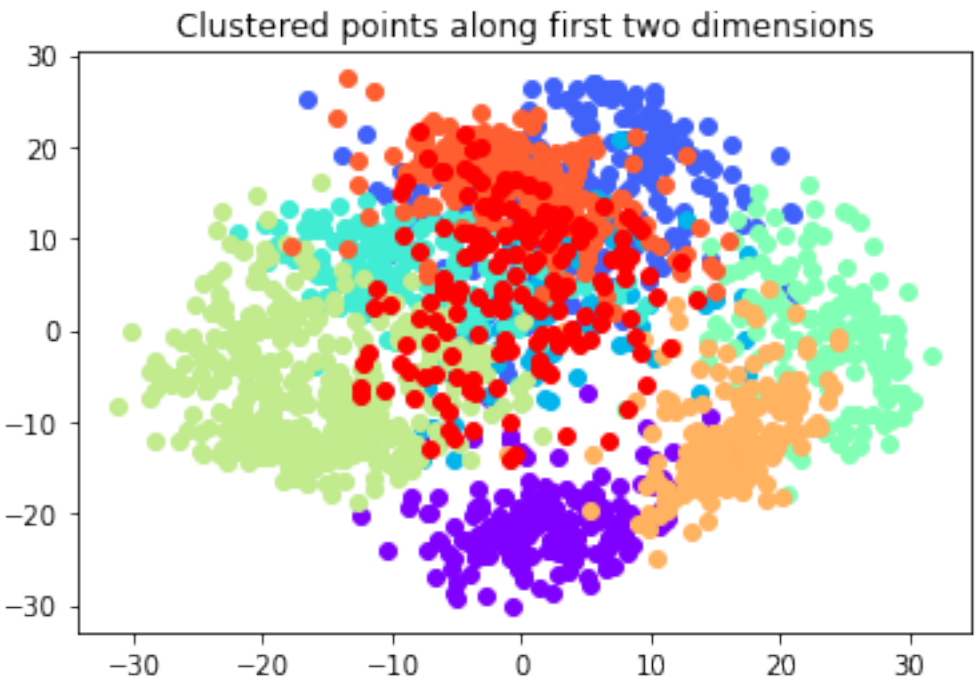


Figure 6: Clustered points

(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: As we increase the number of dimensions in PCA, less will be the error between the original and reconstructed data. Thus, we have a trade off between memory space and error in reduced data.

While for K-means, increasing cluster size will also decrease the value of the objective function, however the decrease is rapid at first, and it flattens out eventually. Thus, we choose the cluster size where the biggest change in slope is observed.

Now, when we look at images, we mainly observe the features that have a big contribution towards the overall structure of the image. They represent the features with a big eigenvalues in our PCA. Then for clustering, we observe the above features and cluster images based on how close the features lie. This represents our K-means algorithm where we cluster our images based on the features selected from the PCA step.
Thus, the overall PCA + K-means is very similar to how our brain would cluster the images.