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

Roll No: EE19B132 — Name: Sagnik Ghosh
Collaborators (if any): ME19B012 (Dhananjay)
References (if any): CMB

- 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: Consider an arbitrary non-zero eigenvalue λ of XX^T with corresponding eigenvector v.

$$XX^Tv = \lambda v$$
$$X^T(XX^Tv) = X^T\lambda v$$
$$(X^TX)(X^Tv) = \lambda(X^Tv)$$

From the above equation, it is clear that λ is an eigenvalue of X^TX with corresponding eigenvector X^Tv.

As we have taken a general non-zero eigenvalue λ, we can conclude that for any non-zero eigenvalue of XX^T, it is also an eigenvalue of X^TX. Hence, they have the same set of non-zero eigenvalues.

Now, let us prove that the non-zero eigenvalues are positive. Consider an arbitrary non-zero eigenvalue λ of XX^T with corresponding eigenvector v.

$$XX^Tv = \lambda v$$
$$v^TXX^Tv = v^T\lambda v$$
$$||X^Tv||^2 = \lambda||v||^2$$

The above equation is only true if λ is positive. As we have taken an arbitrary non-zero eigenvalue, all non-zero eigenvalues are positive. As we proved earlier, XX^T and X^TX have the same set of non-zero eigenvalues, the non-zero eigenvalues of X^TX are also positive.

Consider an eigenvector of A corresponding to the zero eigenvalue,

$$Av = (0)v$$
$$Av = 0$$

Hence, v belongs to the null-space of A.

Hence, the multiplicity of zero eigenvalue of XX^T is the nullity of XX^T and the multiplicity of zero eigenvalue of X^TX is the nullity of X^TX.

We know, rank(XX^T) = rank(X^TX) = rank(X).

We are also given that the rows of X are linearly independent. Hence, rank(X) = N.

Hence, rank(XX^T) = rank(X^TX) = N.

We also know, (Number of non-zero eigenvalues) + (Multiplicity of zero eigenvalue) = N

Comparing the above with rank + nullity = N and nullity = multiplicity of zero eigenvalue, we can see that number of non-zero eigenvalues = rank(XX^T) = rank(X^TX) = N.

For XX^T, rank(XX^T) + nullity(XX^T) = N.

Hence, N + nullity(XX^T) = N

nullity(XX^T) = 0

Hence, the multiplicity of zero eigenvalue of XX^T is 0.

Similarly for X^TX,

rank($X^T X$) + nullity($X^T X$) = D
N + nullity($X^T X$) = D
nullity($X^T X$) = D − N
Hence, the multiplicity of zero eigenvalue of $X^T X$ 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 $X^T X$. 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: $(XX^T)^T = XX^T$ and $(X^T X)^T = X^T X$.
Both XX^T and $X^T X$ are real symmetric matrices.
Let us consider a real symmatric matrix A with distinct eigenvalues λ_1, λ_2 with eigenvectors v_1, v_2.

$$Av_1 = \lambda_1 v_1$$
$$Av_2 = \lambda_2 v_2$$
$$\lambda_1 < v_1, v_2 >=< \lambda_1 v_1, v_2 >$$
$$=< Av_1, v_2 >$$
$$< v_1, A^T v_2 >$$
$$< v1, \lambda_2 v_2 >$$
$$\lambda_2 < v_1, v_2 >$$

Hence, $\lambda_1 < v_1, v_2 >= \lambda_2 < v_1, v_2 >$, but $\lambda_1 \neq \lambda_2$. This is only possible if $< v_1, v_2 >= 0$, i.e., v_1 is orthogonal to v_2.
We have proved that for a real symmetric matrix A, the eigenvectors corresponding to distinct non-zero eigenvalues are orthogonal. If A also has zero eigenvalue with some multiplicity, the eigenvectors corresponding to the zero eigenvalue span the null space of A and we can always find an orthonormal basis for the null space of A. Hence, we can choose an orthonormal set of eigenvectors for any real symmetric matrix.
As both XX^T and $X^T X$ are real symmetric, we can choose orthonormal sets of eigenvectors $\{u_i\}_{i=1,\ldots,N}$ and $\{v_j\}_{j=1,\ldots,D}$ respectively.
Let us consider an arbitrary non-zero eigenvalue λ_i of XX^T with corresponding eigenvector u_i.

$$XX^T u_i = \lambda_i u_i$$
$$X^T(XX^T u_i) = X^T \lambda_i i_i$$
$$(X^T X)(X^T u_i) = \lambda_i (X^T u_i)$$

From the above equation, it is clear that λ_i is also an eigenvalue of $X^T X$ with eigenvector $v_i = X^T u_i$.

In conclusion, we find the eigenvectors of XX^T as $\{u_i\}_{i=1,\ldots,N}$ and then find all the required eigenvalues $\{v_j\}_{j=1,\ldots,D}$ of $X^T X$ using $v_i = X^T u_i$. The eigenvectors of $X^T X$ corresponding to the zero eigenvalue ($D - N$ of them), can be found by finding an orthonormal set of its null space.

(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: The following are the exact steps involved in applying PCA in $O(N^3)$ time.

- Find the eigenvalues λ_i's and the eigenvectors u_i's of the matric $S' = \frac{1}{N} XX^T$.
- The eigenvalues λ_i's found in the previous step are all non-zero as proved in part (a) of this problem. The covariance matrix S has the same set of eigenvalues as S' and also the eigenvalue zero with a multiplicity of $D-N$. Now we have all the eigenvalues of the covariance matrix.
- As proved in the previous part, we can find the eigenvectors v_i's corresponding to non-zero eigenvalues of the covariance matrix S from the eigenvectors of S', u_i's as follows,

$$v_i = X^T u_i$$

- Now, we have the eigenvectors of the covariance matrix corresponding to the non-zero eigenvalues. The eigenvectors corresponding to the zero eigenvalue span the null space. Hence, we can find the remaining eigenvectors by finding an orthogonal basis set for the null space of the covariance matrix.
- We now have all the eigenvectors of the covariance matrix S. We can **normalize** each of the eigenvectors by dividing them by their norm as follows,

$$v_i = \frac{v_i}{||v_i||}$$

- Arrange the v_i's in descending order of their corresponding eigenvalues.
- Now we can apply PCA by projecting the datapoints onto a suitable number of v_i's in descending order of their eigenvalues.
- The above algorithm works because we are now doing an eigenvalue decomposition of an N dimensional square matrix S' which takes $O(N^3)$ time instead of an eigenvalue decomposition of the D dimensional covariance matrix which takes $O(D^3)$ time.

(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: We have,

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

Let us simplify the above expression by using $S\widehat{U} = \widehat{U}H$.

$$\begin{aligned}\widetilde{J} &= Tr\left\{\widehat{U}^\top\widehat{U}H\right\} + Tr\{H\} - Tr\left\{H\widehat{U}^\top\widehat{U}\right\} \\ &= Tr\left\{\widehat{U}^\top\widehat{U}H\right\} + Tr\{H\} - Tr\left\{\widehat{U}^\top\widehat{U}H\right\} \\ &= Tr\{H\}\end{aligned}$$

Note that the above expression is valid for **any general solution**.

For the **specific solution**, columns of $\widehat{U}$ are eigenvectors of S, the H is a diagonal matrix with diagonal entries equal to the eigenvalues of S corresponding to the eigenvectors in $\widehat{U}$.

Hence, for the **specific solution**, the expression simplifies to,

$\widetilde{J} = Tr\{H\} =$ (Sum of eigenvectors of S corresponding to the eigenvectors as columns of $\widehat{U}$)

To minimize $\widetilde{J}$, we need to choose $\widehat{U}$ such that it's columns are the eigenvector corresponding to the $D - M$ smallest eigenvalues of S.

Hence, $\widetilde{J}$ is equal to the sum of $D - M$ smallest eigenvalues of S for the **specific solution**.

We now need to prove that for any general solution to $S\widehat{U} = \widehat{U}H$, $\tilde{J}$ is equal to the sum of $D - M$ smallest eigenvalues of S.

Before proving that, we will first prove that H can be assumed to be symmetric, and hence diagonalizable.

For any general solution, we proved that $\tilde{J} = \text{Tr}\{H\}$. But $\text{Tr}\{H\} = \text{Tr}\{H^T\}$. Hence, even if we substitute H with H^T, the final solution remains the same. Hence, we can asssume H to be symmetric and hence diagonalizable.

Let us now diagonalize H as follows,

$$H = ABA^{-1}$$

where B is a diagonal matrix.

Let us consider $S\widehat{U} = \widehat{U}H$ and substitute for H,

$$S\widehat{U} = \widehat{U}ABA^{-1}$$
$$S\widehat{U}A = \widehat{U}AB$$
$$S(\widehat{U}A) = (\widehat{U}A)B$$

Because B is a diagonal matrix, it is clear from the above expression that the diagonal entries of B are the eigenvalues of S corresponding to the eigenvectors which are the columns of $\widehat{U}A$.

We can also choose $\widehat{U}A$ such that its columns are eigenvectors corresponding to the $D - M$ smallest eigenvalues of S

Hence, $\text{Tr}\{B\}$ = sum of $D - M$ smallest eigenvalues of S

But $\text{Tr}\{H\} = \text{Tr}\{ABA^{-1}\} = \text{Tr}\{AA^{-1}B\} = \text{Tr}\{B\}$.

Hence, $\text{Tr}\{H\}$ = sum of $D - M$ smallest eigenvalues of S.

Hence, $\tilde{J}$ = sum of $D - M$ smallest eigenvalues of S.

This is exactly the same result we got for the **specific 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: Suppose we have K clusters and N data points. Then the objective function using ℓ_1 norm would be as follows:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|$$

where r_{nk}'s are the responsibilities (assignments) and μ_k's are the mean vectors corresponding to each cluster. Note that $\|.\|$ is used to denote ℓ_1 norm in this problem.

Minimizing the objective function with respect to r_{nk} we get the **assignment equations** as follows:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \left\|\mathbf{x}_n - \boldsymbol{\mu}_j\right\| \\ 0 & \text{otherwise.} \end{cases}$$

Let us now try maximizing the objective function with respect to the mean vectors μ_k's. To do so, let us consider each cluster individually. Let us consider and arbitrary cluster with mean vector μ_p. This cluster has q data points assigned to it and let us assume that all data points assigned to this cluster form a set P. The objective function corresponding to this cluster is as follows:

$$\sum_{x_n \in P} \left\|\mathbf{x}_n - \boldsymbol{\mu}_p\right\|$$

Taking derivative of this with respect to μ_p, we find that all terms in this summation are either $+1$ or -1 depending on whether norm of x_n is lesser than or greater than μ_p respectively. Hence, the derivative will become zero if there an equal number of $+1$'s and -1's. To achieve this, we need to choose a μ_p such that its ℓ_1 norm is greater than half the x_n's and lesser than the other half.

In conclusion, in the **update step**, for a particular cluster with mean vector μ_p, we update μ_p by choosing it such that it's ℓ_1 norm is greater than half the data points assigned to it and less than the other half. Note that this chosen μ_p is not unique as it can be chosen in many different ways to satisfy the above condition.

If the data contains outliers, ℓ_1 norm should be preferred over ℓ_2 norm. It has been reasoned above that for the case of K Means with ℓ_1 norm, we choose the mean vectors such that the norm of the mean vector is less then half the data points assigned to that cluster and greater than the other half. It can be noticed that in the update step with ℓ_1 norm, we do not consider the exact magnitude of the norm of the data points but we only care about whether it's norm is greater than or less then the mean vector's norm. From this, we can conclude that outliers should not affect the update step of K Means with ℓ_1 norm significantly.

Reference: Stack Exchange - Derivative of L1 norm

(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 responsibilities of soft K-Means in terms of the current parameters are as follows,

$$\gamma\left(z_{nk}\right) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}$$

If we take $\Sigma_r = \sigma^2 I$, then we will prove that $\gamma\left(z_{nr}\right)$ tends to r_{nr}, the responsibilities of hard K-means as σ tends to zero.

Note that we are considering k clusters with mean vectors, $\mu_1, \mu_2, ..., \mu_k$.
The responsibilities simplify to the following,

$$\gamma(z_{nr}) = \frac{\pi_r \exp\left(\frac{-1}{2\sigma^2}\|x_n - \mu_r\|^2\right)}{\pi_1 \exp\left(\frac{-1}{2\sigma^2}\|x_n - \mu_1\|^2\right) + \cdots + \pi_k \exp\left(\frac{-1}{2\sigma^2}\|x_n - \mu_k\|^2\right)}$$

This simplifies to,

$$\gamma(z_{nr}) = \frac{\pi_r}{\pi_1 \exp\left(\frac{-1}{2\sigma^2}\left(\|x_n - \mu_1\|^2 - \|x_n - \mu_r\|^2\right)\right) + \cdots + \pi_r + \cdots + \pi_k \exp\left(\frac{-1}{2\sigma^2}\left(\|x_n - \mu_k\|^2 - \|x_n - \mu_r\|^2\right)\right)}$$

If we consider the case when $r = \arg\min_j \left\|\mathbf{x}_n - \boldsymbol{\mu}_j\right\|^2$, and as σ tends to zero, the above expression simplifies to,

$$\gamma(z_{nr}) = \frac{\pi_r}{0 + \ldots + \pi_r + \ldots + 0}$$
$$\gamma(z_{nr}) = 1$$

If we consider the case when $r \neq \arg\min_j \left\|\mathbf{x}_n - \boldsymbol{\mu}_j\right\|^2$, then the expression simplifies to,

$$\gamma(z_{nr}) = \frac{\pi_r}{\infty}$$
$$\gamma(z_{nr}) = 0$$

Hence,

$$\gamma(z_{nr}) = \begin{cases} 1 & \text{if } r = \arg\min_j \left\|\mathbf{x}_n - \boldsymbol{\mu}_j\right\|^2 \\ 0 & \text{otherwise} \end{cases}$$

The above expression is exactly the same as the responsibilities of the hard K-means algorithm.

The above expression for the responsibilities coupled with the update expression for the cluster means, the soft K-means becomes the 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: If $k = 3$, then $x_1 = 1$ should belong to Cluster-1, $x_2 = 3$ should belong to Cluster-2 and $x_3 = 6, x_4 = 7$ should belong to Cluster-3.

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: We are given, $x_1 = 1, x_2 = 3, x_3 = 6, x_4 = 7$.
Let us initialize the mean vectors for each cluster as $\mu_1 = 1, \mu_2 = 6, \mu_3 = 7$ for the 3 clusters.
Let us calculate the responsibilities and represent it as a table. In the following table,

the entry in the i^{th} row and j^{th} column is 1 if the i^{th} datapoint belongs to the j^{th} cluster and 0 otherwise.

1	0	0
1	0	0
0	1	0
0	0	1

Iteration-1:
The mean vectors can now be calculated as follows:
$\mu_1 = \frac{1+3}{2} = 2$
$\mu_2 = 6$
$\mu_3 = 7$
Let us recompute the responsibilities and represent it as a table:

1	0	0
1	0	0
0	1	0
0	0	1

Iteration-2:
The mean vectors can now be calculated as follows:
$\mu_1 = \frac{2+2}{2} = 2$
$\mu_2 = 6$
$\mu_3 = 7$
Let us recompute the responsibilities and represent it as a table:

1	0	0
1	0	0
0	1	0
0	0	1

In iteration-2, the mean vectors and responsibilities have not changed, hence K-means has converged.
From the table we can see that $x_1 = 1, x_2 = 3$ belong to Cluster-1, $x_3 = 6$ belongs to Cluster-2 and $x_4 = 7$ belongs to Cluster-3. This is not the same as the optimal cluster assignment in the previous part. Hence, it is stuck in a suboptimal cluster.

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: We are given data such that $x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n$.
The update equation for K-means is as follows,

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j (\mathbf{x}_i - \mu_j)^2 \\ 0 & \text{otherwise.} \end{cases}$$

In other words, any x_i belongs to cluster with clustercenter μ_k if $k = \arg\min_j (\mathbf{x}_i - \mu_j)^2$. Any optimal clustering satisfies the above condition.
Let us consider 2 datapoints x_a and x_b such that $x_a < x_b$ and both x_a and x_b belong to the same cluster with cluster center μ_l. Let us also consider another arbitrary point x_c such that $x_a < x_c < x_b$. We will now prove that x_c also belongs to the same cluster with cluster center μ_l.
Writing the above conditions, mathematically,

$$(\mathbf{x}_a - \mu_l)^2 < (\mathbf{x}_a - \mu_j)^2 \text{ for all } j$$
$$(\mathbf{x}_b - \mu_l)^2 < (\mathbf{x}_b - \mu_j)^2 \text{ for all } j$$

The above can be rewritten as follows:

$$2x_a(\mu_j - \mu_l) < \mu_j^2 - \mu_l^2$$
$$2x_b(\mu_j - \mu_l) < \mu_j^2 - \mu_l^2$$

As they are inequalities, we need to consider 2 cases.
Case 1: $\mu_j > \mu_l$ Hence,

$$x_a < \tfrac{\mu_j + \mu_l}{2}$$
$$x_b < \tfrac{\mu_j + \mu_l}{2}$$

As $x_a < x_c < x_b$, we can write,

$$x_a < x_c < x_b < \tfrac{\mu_j + \mu_l}{2}$$
$$x_c < \tfrac{\mu_j + \mu_l}{2}$$

As $\mu_j > \mu_l$, we can multiply both sides by $(\mu_j - \mu_l)$ without reversing the inequality.

$$2x_c(\mu_j - \mu_l) < \mu_j^2 - \mu_l^2$$

Rearranging the terms and adding x_c^2 to both sides,

$$x_c^2 + \mu_l^2 - 2x_c\mu_l < x_c^2 + \mu_j^2 - 2x_c\mu_j$$
$$(\mathbf{x}_c - \mu_l)^2 < (\mathbf{x}_c - \mu_j)^2$$

We have now proved that for any x_c such that $x_a < x_c < x_b$, $(\mathbf{x}_c - \mu_l)^2 < (\mathbf{x}_c - \mu_j)^2$ for any j such that $\mu_j > \mu_l$.
Case 2: $\mu_j > \mu_l$

$$x_a > \tfrac{\mu_j + \mu_l}{2}$$
$$x_b > \tfrac{\mu_j + \mu_l}{2}$$

As $x_a < x_c < x_b$, we can write,

$$\tfrac{\mu_j + \mu_l}{2} < x_a < x_c < x_b$$
$$x_c > \tfrac{\mu_j + \mu_l}{2}$$

As $\mu_j < \mu_l$, we can multiply both sides by $(\mu_j - \mu_l)$ and the inequality gets reversed.

$$2x_c(\mu_j - \mu_l) < \mu_j^2 - \mu_l^2$$

Rearranging the terms and adding x_c^2 to both sides,

$$x_c^2 + \mu_l^2 - 2x_c\mu_l < x_c^2 + \mu_j^2 - 2x_c\mu_j$$

$$(\mathbf{x}_c - \mu_l)^2 < (\mathbf{x}_c - \mu_j)^2$$

We have now proved that for any x_c such that $x_a < x_c < x_b$, $(\mathbf{x}_c - \mu_l)^2 < (\mathbf{x}_c - \mu_j)^2$ for any j such that $\mu_j < \mu_l$.

Combining **Case 1 and Case 2**, we have proved that for any x_c such that $x_a < x_c < x_b$, $(\mathbf{x}_c - \mu_l)^2 < (\mathbf{x}_c - \mu_j)^2$ for any j.

Hence, for any x_c such that $x_a < x_c < x_b$ and x_a and x_b belong to the same cluster with cluster center μ_l, even x_c belongs to the same cluster.

The above condition can only be satisfied if each cluster in an optimal clustering consists of points $x_a, x_{a+1}, \ldots, x_b$ for some $1 \leqslant a \leqslant b \leqslant n$.

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

Solution: As we proved in the previous part, any optimal K-means clustering for 1D data which has been sorted, the clusters would be in intervals. This means that once we have sorted data, we only need to check the edges of the intervals. This is full algorithm:

- Sort the 1D data in ascending order.
- Initialise the cluster centers as any k random points in the dataset.
- Assign each datapoint to its nearest cluster by finding the distance to each cluster. This will give you clusters which are continuous intervals.
- Now, update the cluster centers as the mean of all the datapoints assigned to it.
- Now, check **only the points at the edge of each cluster interval** and reassign them to the closest cluster.
- Repeat the previous 2 steps for a suitable number of iterations until convergence.

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).

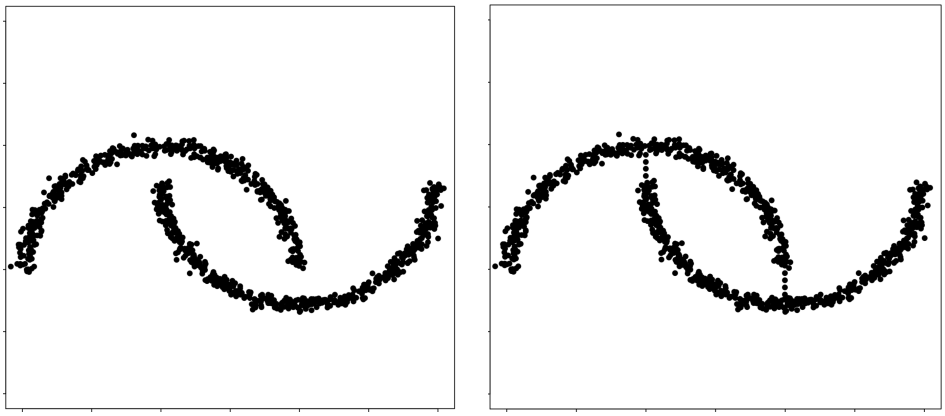
(a) (2 points) Which of the three distance/dissimilarity metrics described above would most likely result in clusters most similar to those given by K-means? (Consider the hierarchical clustering method as described in class and further *cut* the tree to obtain K clusters. Assume K is power of 2.) Explain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

Solution: The **average distance metric** would result in clusters most similar to K-means clustering.

Suppose we have used hierarchical clustering with average distance metric and we have cut the tree to obtain K clusters. As we used the average distance metric, each of the K clusters were obtained from subclusters whose average distance was the least. Hence, the intra-cluster distance is minimized. Even in K-means, we assign each datapoint to a mean vector which is closest to the datapoint. Hence, the intra cluster distance would be minimum. This gives us an intuitive explanation for clustering with average distance metric being similar to K-means clustering.

Another alternate, simpler explanation would be that clustering with minimum and maximum distance metric consider only the distance between two representative points before updating them. Whereas, clustering with average distance metric and K-means clustering consider all the datapoints before updating.

(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: Hierarchical clustering using the minimum distance measure is also called single linkage clustering. Single linkage clustering considers the distance between the closest points between 2 clusters as the distance measure before merging. As it only considers the closest points between two clusters before merging, points within a cluster maybe far apart after merging. Hence, single linkage clustering often results in 'chain like clusters' and it is not at all robust to outliers because even a single outlier can cause 2 clusters to merge.

On the other hand, hierarchical clustering with maximum distance measure (complete link clustering) considers the distance between the 2 farthest points of 2 clusters before merging and hence two points within a cluster are close together after merging. Hence, complete link clustering results in clump like spherical clusters with maximum diameter.

Hierarchical clustering with average distance measure is called average link clustering. As average link clustering considers distance between all pairs of points between clusters, it results in clusters which are hybrids of single link and complete link.

Figure 1a has two clusters which are chains and they are completely separable, i.e., there are no outliers. From the above discussion it is clear that the **minimum distance measure or single link clustering** will separate the two clusters.

Figure 1b is the same as 1a but the two clusters are not separable, i.e., there are outliers. Hence, single link clustering will fail to separate the two moons/clusters. The two measures will also fail as the clusters are in the form of chains. Hence, none of the three measures will work for Figure 1b.

(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:

	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

	AB	C	D	E
AB	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

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

	AB	CDE
AB	0	2.90
CDE	2.90	0

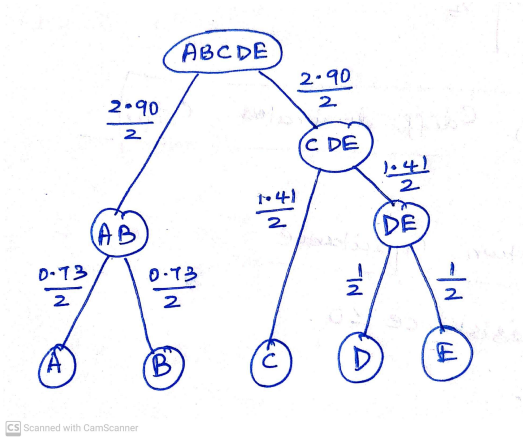
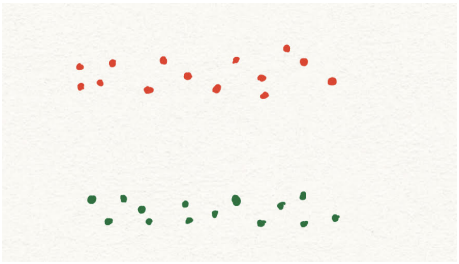


Figure 2: Dendrogram

(d) (2 points) Which distance metric (minimum, maximum and average) is more likely to generate following results given in Figure 3a 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: The minimum distance metric is more likely to generate the results seen in the figure. In the figure, we can notice two long and narrow clusters and there is no overlap between the two clusters, i.e., there are no outliers. It was already explained in part (b) of this problem that clustering with the minimum distance metric results in 'chain-like' clusters as it only considers the two closest points between any two clusters before merging. On the other hand, clustering with average and maximum distance metric tend to form clusters which are larger in diameter and form clumps instead of chains. We also discussed that clustering with minimum distance metric is not at all robust to outliers and in the given figure, there are no outliers, i.e., the clusters are completely separable. Hence, clustering with minimum distance metric is more likely to have generated the given results.

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: The minimum cut method cuts the graph at a place which has the least cost/ sum of edge weights. Consider a scenrario where there is one single datapoint which is connected to the rest of the graph. Cutting the graph at this edge would lead to the minimum cut cost not because it has low edge weights but simply because the number of edges is the least possible through the cut and hence this method will cluster the dataset into a huge cluster and one datapoint. While the above scenario is an extreme case, it is possible that this method partitions the dataset into one large cluster and one small cluster just because there are lesser number of edges in the cut which reduces the cut cost even if the edge weights are significant.

   (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: We are given that the degree matrix D is a diagonal matrix and its diagonal elements are given by $d_{ii} = \Sigma_j W_{ij}$ where W is the similarity matrix. The above expression basically means that each diagonal element of D is the sum of the values in the corresponding row in W. As D is a diagonal matrix, sum of elements of each row in D is equal to the sum of elements of the corresponding row in W.

We are given the Laplacian Matrix L equal to $D - W$. As sum of elements in each row in D is equal to the sum of elements in the corresponding row in W, the sum of elements of each row in $D - W = L$ is equal 0. This means that the sum of all columns of L is equal to the zero vector. Hence, we have found a linear combination of the columns of L which is equal to zero. Hence, the columns of L are linearly dependent. Hence, L, the Laplacian 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 are given $(D - W)y = \lambda Dy$ or $Ly = \lambda Dy$. Let us make a substitution $y = Az$. We get,

$$LAz = \lambda DAz$$
$$\Rightarrow ALAz = \lambda ADAz$$

We want the equation in the form $ALAz = \lambda z$. Comparing the two equations,

$$\lambda ADAz = \lambda z$$
$$\Rightarrow ADA = I$$
$$\Rightarrow A = D^{-\frac{1}{2}}$$

We have expressed A in terms of D, now we need to express z. We know that $y = Az$ from the initial substitution and we know $A = D^{-\frac{1}{2}}$. Hence, $z = D^{\frac{1}{2}}y$.

Now, we have to find the eigenvector of $M = ALA = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ corresponding to its lowest eigenvalue. We know that the Laplacian L is Positive Semi Definite. Using this fact we will try to prove that M is also Positive Semi-Definite. Let us consider $x^TALAx = x^TD^{-\frac{1}{2}}LD^{-\frac{1}{2}}x$ for some arbitrary vector x. We have,

$$x^TD^{-\frac{1}{2}}LD^{-\frac{1}{2}}x = (D^{-\frac{1}{2}}x)^TL(D^{-\frac{1}{2}}x)$$

The above quantity is always greater than equal to zero because L is Positive Semi-Definite. As x^TALAx is non-negative for any x, $M = ALA$ is also Positive Semi-Definite. This means that the smallest eigenvalue $M = ALA$ can have is zero.

Let us consider the original equation $Ly = \lambda Dy$. Substituting $\lambda = 0$ in this equation we get, $Ly = 0$. We already know that the columns of L add up to give the zero vector and hence $y = \mathbf{1}$ where each element of $\mathbf{1}$ is 1, satisfies the equation $Ly = \lambda Dy$. As this equation is the same as $(ALA)z = \lambda z$ and $y = \mathbf{1}$, using the fact that $z = D^{\frac{1}{2}}y$, we get $z = D^{\frac{1}{2}}\mathbf{1}$.

Hence, the eigenvector corresponding to the smallest eigenvalue $\lambda = 0$ is $D^{\frac{1}{2}}\mathbf{1}$.

(d) (4 points) Now, let's bring back the constraint that $y^TD\mathbf{1} = 0$ (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: We are given 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.

The GRQ given is $\frac{y^T(D-W)y}{y^T Dy}$. We are supposed to prove that this is minimized when y is the eigenvector corresponding to the second smallest eigenvalue of the generalized eigenvalue system $(D - W)y = \lambda Dy$ subject to $y^T D\mathbf{1} = 0$.

In light of the given fact, let us express the GRQ in the form given in the fact. We know from the previous part that $z = D^{\frac{1}{2}}y$. Substituting this in the GRQ, we get the GRQ as $\frac{z^T(D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}})z}{z^T z}$ where $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ is clearly symmetric (this can be verified easily by noticing that both $D - W$ and $D^{-\frac{1}{2}}$ are symmetric).

The eigenvalues of $(D-W)y = \lambda Dy$ are the same as $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}z = \lambda z$ (or $ALAz = \lambda z$) as proved in the previous part. Hence, if we can prove that the GRQ is minimized when z is the eigenvector corresponding to the second smallest eigenvalue of $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$, we have proved the required result.

This can be done as follows. Using the given fact, we can prove that GRQ is minimized if z is the eigenvector corresponding to the second smallest eigenvalue of $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ by proving that z is orthogonal to the eigenvector corresponding to the smallest eigenvalue of $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$. We have already proved in the previous part that $D^{\frac{1}{2}}\mathbf{1}$ is the eigenvector corresponding to the smallest eigenvalue of $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$. Now we need to prove that z is orthogonal to $D^{\frac{1}{2}}\mathbf{1}$.

The constraint we are given is $y^T D\mathbf{1} = 0$. This constraint can be rewritten as $(D^{\frac{1}{2}}y)^T(D^{\frac{1}{2}}\mathbf{1}) = 0$. We know that $z = D^{\frac{1}{2}}y$. Hence, $z^T(D^{\frac{1}{2}}\mathbf{1}) = 0$. This means that z is orthogonal to $D^{\frac{1}{2}}\mathbf{1}$ which is the eigenvector corresponding to the smallest eigenvalue.

Hence, the GRQ is minimized when z is the eigenvector corresponding to the second smallest eigenvalue of the system $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}z = \lambda z$ or y is the eigenvector corresponding to the second smallest eigenvalue of the generalized eigenvalue system $(D - W)y = \lambda Dy$ as both the systems are equivalent.

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:

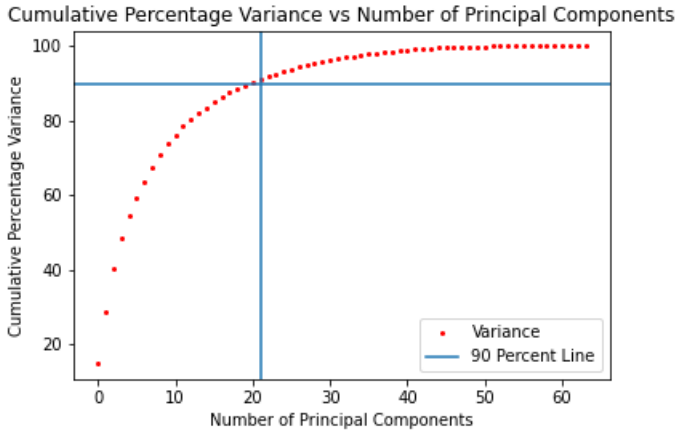


Figure 4: Cumulative Percentage Variance vs Number of Principal Components

From the above plot, it is clear that the number of principal components that contribute to 90 percent of the variance in the dataset 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 $\hat{d}$ based on the MSE values.

Solution: Please refer to the Python Notebook submitted for the code. The MSE values computed in the Python notebook for dimensions [2,4,8,16] are as follows:

```
The MSE value considering 2 dimensions is 858.94
The MSE value considering 4 dimensions is 616.19
The MSE value considering 8 dimensions is 391.79
The MSE value considering 16 dimensions is 180.94
```

The optimal dimension $\widehat{d}$ would be 16 based on the above MSE values. This is because, not only does $\widehat{d} = 16$ give a lower MSE value than $\widehat{d} = 8$, but it gives a considerably lower MSE value.

(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 briefy 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: To find the optimal number of clusters k, we used the **Elbow Method**. In this method we find the sum of average distances between the datapoints in each cluster to the cluster centers, called the distortion. When we plot this for different k, we notice that the distortion reduces for increasing k but after a particular k, it almost saturates. This is the optimal k.

This method works because for any k less than the optimal k, there would be clusters which are actually distinct/separable but they are taken as one cluster because of the smaller number of clusters. For such a cluster, the average distance between any two points would be high, which can be reduced by taking a larger number of clusters. When we take a k which is greater than the optimal k, there would be clusters which are actually the same cluster but are separated because of the high value of k. This would not cause the distortion to change significantly from the value obtained for the optimal k because calculating the distortion for the two clusters separately would be equivalent to calculating it together as the inter cluster distance is small.

The distortion plot obtained for the given dataset is as follows:

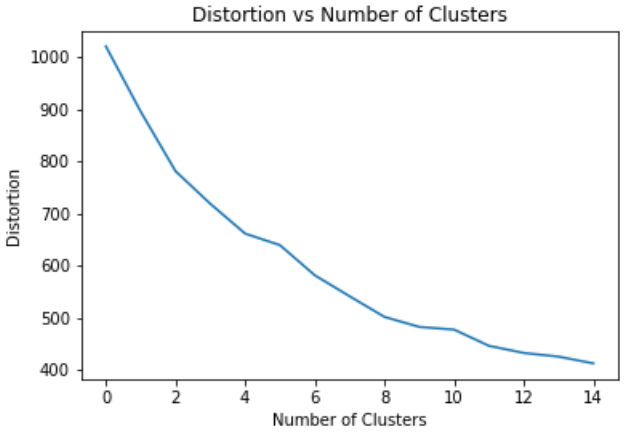


Figure 5: Distortion vs Number of Clusters

We can notice in the above plot that the distortion value starts saturating around $k = 10$ and hence 10 is the optimal number of clusters. This goes well with our intuition because we are dealing with a dataset which contains images of 10 different digits.

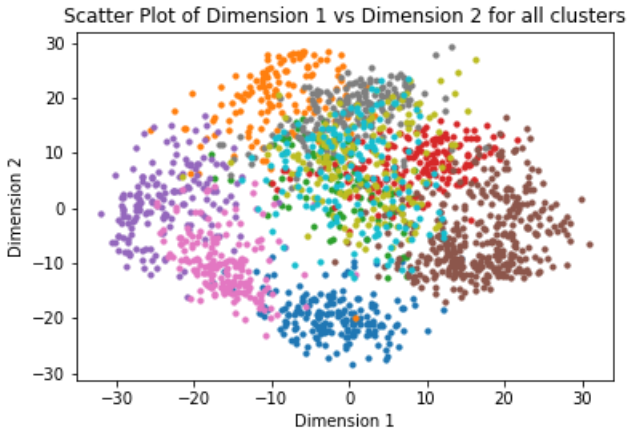


Figure 6: Scatter Plot of the First Two Dimensions

(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:

- In part (a) of this problem, we plotted the Cumulative Percentage Variance explained by each of the principal components. We obtained a monotonically increasing, con-

cave curve which tends to saturate for higher number of principal components. The curve increases because we are plotting a cumulative value which is always non-negative (variance).

- The curve was concave and tended to saturate for higher number of principal components because the principal components corresponding to the larger eigenvalues would contribute to higher variance as is consistent with the theory of PCA.
- In part (b), we calculated the Mean Squared Error between the actual datapoints and the reconstructed datapoints. The MSE value for a dimension d should be the sum of the $d - m$ smallest eigenvalues of the covariance matrix. The MSE values obtained are in accordance to the above fact.
- We concluded from the MSE values that $\widehat{d} = 16$ was the optimal $\widehat{d}$ because it gave the least MSE value and the MSE value was significantly lower than for $\widehat{d} = 8$.
- Next, we applied K-means to the dataset and we obtained the optimal number of clusters by using the Elbow Method. In this method, the average squared distance between the datapoints to the respective cluster centers. Using this, the optimal number of clusters obtained was 10 and this goes well with our intuition because we are dealing with a dataset which has images of 10 different digits.
- From the Scatter Plot it is clear that the clustering is consistent with how our brains would cluster the datapoints as we can notice distinct clusters with different colours. Although there is some overlap between the clusters, this overlap can be explained by considering the fact that we plotted only the first two dimensions and the clustering was done on the basis of 14 other dimensions.