DD2434 Assignment 1

Yage Hao

December 1, 2020

Problem 1

(i)

Proof. Assume eigenvalue λ for matrix A is complex. Then there exists non-zero eigenvector v with complex entries, since $Av = \lambda v$ and A is a real matrix.
Multiplying Av with the complex conjugate of v gets:

$$\bar{v}^T A v = \bar{v}^T \lambda v = \lambda \bar{v}^T v = \lambda(\bar{v} \cdot v) \tag{1}$$

Since A is symmetric,

$$\begin{aligned} \bar{v}^T A v &= \bar{v}^T A^T v \\ &= (A\bar{v})^T v \end{aligned}$$

and since taking complex conjugate of both sides of $Av = \lambda v$ gives $\bar{A}\bar{v} = \bar{\lambda}\bar{v}$ and implies $A\bar{v} = \bar{\lambda}\bar{v}$ as A is real,

$$\begin{aligned} &= (\bar{\lambda}\bar{v})^T v \\ &= \bar{\lambda}\bar{v}^T v \\ &= \bar{\lambda}(\bar{v} \cdot v) \end{aligned} \tag{2}$$

$$(1) = (2) \implies \lambda(\bar{v} \cdot v) = \bar{\lambda}(\bar{v} \cdot v) \implies \lambda = \bar{\lambda}$$

Contradiction! Therefore, λ is real and real symmetrix matrix has real eigenvalues. □

(ii)

Proof. Suppose v_i and v_j are two arbitrary eigenvectors for matrix A, and their corresponding eigenvalues are λ_i and λ_j for which $\lambda_i \neq \lambda_j$. Then we get:

$$Av_i = \lambda_i v_i \tag{3}$$

$$Av_j = \lambda_j v_j \tag{4}$$

Multiplying (3) by v_j gives:

$$v_j^T A v_i = \lambda_i (v_i \cdot v_j) \tag{5}$$

Multiplying (4) by v_i gives:

$$v_i^T A v_j = \lambda_j (v_i \cdot v_j) \tag{6}$$

$$\begin{aligned} (5) = (6) &\implies \lambda_i(v_i \cdot v_j) - \lambda_j(v_i \cdot v_j) = 0 \\ &\implies (\lambda_i - \lambda_j)(v_i \cdot v_j) = 0 \\ &\implies v_i \cdot v_j = 0 \end{aligned}$$

Therefore, real symmetric matrix has orthogonal eigenvectors. □

Proof. By definition, eigen-decomposition for A is $A = Q\Lambda Q^{-1}$. To prove eigen-decomposition for A is $A = Q\Lambda Q^T$ is equivalent to prove $Q^T = Q^{-1}$, i.e. $Q^T Q = I$.
Since Q is a matrix with each column an eigenvector, we can replace each eigenvector by a normalized one, i.e. length of eigenvector is 1.
And since eigenvectors are mutually orthogonal, $Q^T Q$ gives a diagnal matrix, wich each element equals $\|v_i\|^2 = 1$. Hence $Q^T Q = I$. □

(iii)

Proof. Suppose A is nxn positive semidefinite matrix, then $x^T Ax \geq 0, \forall x \in \mathbb{R}^n$.

$$\begin{aligned}
&\implies x^T(Q\Lambda Q^T)x \geq 0 \text{ by eigen-decomposition of A} \\
&\implies (x^T Q)\Lambda(Q^T x) \geq 0 \\
&\implies \sum_{i=1}^{n} \lambda_i[(Q^T x)_i]^2 \geq 0 \text{ since } \Lambda = diag(\lambda)
\end{aligned}$$

Therefore, $\lambda_i \geq 0, \forall i$. □

(iv)

Proof. Suppose A is a similarity matrix such that $A_{ij} = y_i^T y_j$ for arbitrary $y_i, y_j \in \mathbb{R}^n$.

$$\begin{aligned}
D_{ij} &= A_{ii} + A_{jj} - 2A_{ij} \\
&= y_i^T y_i + y_j^T y_j - 2y_i^T y_j \\
&= (y_i - y_j)^2 \\
&= (\sqrt{(y_i - y_i)^2})^2 \\
&= \|y_i - y_j\|_2^2 \text{ for arbitrary } y_i, y_j \in \mathbb{R}^n.
\end{aligned}$$

□

Problem 2

Proof. PCA method is on data point matrix $Y \in \mathbb{R}^{d \times n}$, where rows are dimensions and columns are points. SVD of Y is $Y = U\Sigma V^T$. Performing PCA by rows by taking k columns of U that corresponding to largest k singular values, then $X = U_k^T Y$.
Performing PCA on Y's columns is equivalent to performing PCA on Y^T's rows. So consider PCA method on $Y^T \in \mathbb{R}^{n \times d}$. SVD of Y^T is $Y^T = (U\Sigma V^T)^T = V\Sigma^T U^T = V\Sigma' U^T$.
Performing PCA on rows by taking k columns of V that corresponding to largest k singular values in Σ', then $X = V_k^T Y^T$.
This is a PCA process with respective to Y^T's rows, which is equivalent to perform PCA on Y's columns. Notice that U and V exchange interpretations as Y transposed. □

Problem 3

Proof. Suppose data matrix $Y \in \mathbb{R}^{d \times n}$ where rows represent d dimensions and columns represent n data points. Assume Y is the result of applying a linear transformation $W \in \mathbb{R}^{d \times k}$ to a latent matrix $X \in \mathbb{R}^{k \times n}$, i.e. $Y = WX$.
$\max_w tr(Y^T W W^T Y)$ *s.t.* $W^T W = I$ is equivalent to problem $\max_{w_i}(w_i^T Y Y^T w_i)$ *s.t.* $w_i^T w_i = 1$, where $w_i \in \mathbb{R}^d$ is a column of W and $i \leq k$.

Rewrite this in Lagrangian form:

$$\begin{cases} L(w_i, \lambda_i) = w_i^T Y Y^T w_i - \lambda_i (w_i^T w_i - 1) \\ \frac{\partial L}{\partial w_i} = 0 \end{cases}$$
$$\implies 2YY^T w_i - 2\lambda_i w_i = 0$$
$$\iff YY^T w_i = \lambda_i w_i \tag{1}$$

Notice that λ_i, w_i are eigenvalue and eigenvector pair of YY^T respectively.
By considering th erelationship of SVD and eigen-decomposition on Y, i.e.

$$YY^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma V^T V \Sigma^T U^T = U\Sigma^2 U^T \coloneqq Q\Lambda Q^{-1}$$

We can rewrite (1) as:

$$\begin{aligned} & YY^T w_i = \lambda_i w_i \\ \iff & U\Sigma^2 U^T w_i = \sigma_i^2 w_i \text{ where } \sigma_i \text{ is a singular value of Y} \\ \iff & w_i^T U\Sigma^2 U^T w_i = \sigma_i^2 \\ \implies & \max_{w_i}(w_i^T YY^T w_i) = \max_{w_i}(w_i^T U\Sigma^2 U^T w_i) = \max(\sigma_i^2) \text{ where } \sigma_i \text{ now is the largest singular value of } Y \\ \implies & \max_W tr(Y^T W W^T Y) = \max_W tr(W^T U\Sigma^2 U^T W) \coloneqq \sum_{i=1}^{k} \sigma_i^2 = \max(\Sigma^2) \end{aligned} \tag{2}$$

in matrix form, where Σ is the diagonal matrix with k largest singular values and $W \in \mathbb{R}^{d\times k}$ is the matrix with each column vector w_i corresponding to σ_i.

From (2), we deduce that the maximizer W satisfies $W^T U = U^T W = I \implies W = U \coloneqq U_k$ where U_k is the k left singular vectors of Y associated with the k largest singular values. □

Problem 4

N.B. This is a proof for the double centering trick.

Proof. Euclidean distance between two points:

$$d_{ij}^2 = (y_i - y_j)^T (y_i - y_j) = y_i^T y_i + y_j^T y_j - 2y_i^T y_j \tag{1}$$

Assume, w.l.o.g. the centroid of the configuration of n data points is at the origin, i.e. $\sum_{i=1}^n y_i = 0, \forall y_i \in \mathbb{R}^d$.
We then have:

$$\begin{aligned} \frac{1}{n}\sum_{j=1}^n d_{ij}^2 &= y_i^T y_i + \frac{1}{n}\sum_{j=1}^n y_j^T y_j - \frac{2}{n} y_i^T \sum_{j=1}^n y_j \\ &= y_i^T y_i + \frac{1}{n}\sum_{j=1}^n y_j^T y_j \end{aligned} \tag{2}$$

Similarly:

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^n d_{ij}^2 &= \frac{1}{n}\sum_{i=1}^n y_i^T y_i + y_j^T y_j - \frac{2}{n}(\sum_{i=1}^n y_i)^T y_j \\ &= \frac{1}{n}\sum_{i=1}^n y_i^T y_i + y_j^T y_j \end{aligned} \tag{3}$$

and

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 = \frac{n}{n^2}\sum_{i=1}^{n} y_i^T y_i + \frac{n}{n^2}\sum_{j=1}^{n} y_j^T y_j - \frac{2}{n^2}(\sum_{i=1}^{n} y_i)^T(\sum_{j=1}^{n} y_j)$$
$$= \frac{1}{n}\sum_{i=1}^{n} y_i^T y_i + \frac{1}{n}\sum_{j=1}^{n} y_j^T y_j \tag{4}$$

(1) − (2) − (3) + (4) gives:

$$d_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2 - \frac{1}{n}\sum_{j=1}^{n} d_{ij}^2 + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2$$
$$= y_i^T y - i + y_j^T y_j - 2y_i^T y_j - y_i^T y_i - \frac{1}{n}\sum_{j=1}^{n} y_j^T y_j - y_j^T y_j - \frac{1}{n}\sum_{i=1}^{n} y_i^T y_i + \frac{1}{n}\sum_{i=1}^{n} y_i^T y_i + \frac{1}{n}\sum_{j=1}^{n} y_j^T y_j$$
$$= -2y_i^T y_j$$
$$\iff y_i^T y_j = -\frac{1}{2}(d_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2 - \frac{1}{n}\sum_{j=1}^{n} d_{ij}^2 + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2) \tag{5}$$

Left hand side of (5) is the definition of an arbitrary entry in similarity matrix (Gram matrix), s_{ij}, while right hand side of (5) shows the process of double-centering, i.e. subtract from each entry of distance matrix the mean of the corresponding row and the mean of the corresponding column and add back the mean of all entries. □

Problem 5

Proof. Suppose the data is in matrix Y of $n \times d$size with observations in rows and features in columns. Suppose Y_c is the centered matrix with subtracted column means, i.e. $Y_c = (I - \frac{1}{n}1_n)Y$, where 1_n is a $n \times n$ matrix of ones.
The PCA process is to apply SVD $Y_c = U\Sigma V^T$, with columns of $U\Sigma$ being principal components. One way to obtain them is to apply eigen-decomposition to the covariance matrix $\frac{1}{n}Y_c^T Y_c$, another way is to perform eigen-decomposition of the Gram matrix $K_c = Y_c^T Y_c = U\Sigma^2 U^T = (I - \frac{1_n}{n})K(I - \frac{1_n}{n})$, where principal components are its eigenvectors scaled by the square roots of the respective eigenvalues.
Now consider a $n \times n$ matrix D of pairwise Euclidean distances with $D_{ij} = \|x_i - x_j\|$. We claim that this matrix can be converted into K_c in order to perform PCA. Below is a short proof.
By the law of cosines, we obtain:

$$D_{ij}^2 = \|x_i - x_j\|^2$$
$$= \|x_i - \bar{x}\|^2 + \|x_j - \bar{x}\|^2 - 2 < x_i - \bar{x}, x_j - \bar{x} >$$
$$= \|x_i - \bar{x}\|^2 + \|x_j - \bar{x}\|^2 - 2[K_c]_{ij}$$

So $-\frac{D^2}{2}$ differs from K_c only by some row and column constants. Meaning that if we double-center it, we will get K_c:

$$K_c = -(I - \frac{1_n}{n})\frac{D^2}{2}(I - \frac{1_n}{n})$$

Which means that starting from the Euclidean distance matrix D, we can perform PCA and get principal components. This is exactly what classical MDS does: $D \to K_c \to U\Sigma$, so its outcome is equivalent to PCA. □

In terms of computation, if we are provided with original data point matrix Y, then directly apply PCA will be easier. If we are provided with a distance matrix or a similarity matrix with original data Y unknown, then it is better to use MDS techniques.

Problem 6

If the data are sampled from multi-class and each of which corresponding to an independent underlying manifold, the traditional Isomap algorithm may yield a disconnected graph, thus failed to find the low dimensional embedding. To improve this, we consider link each of the independent manifold by compute the pairwise distances between their weighted centroids. In detail:
Suppose data set $X \in \mathbb{R}^{M \times N}$ with class labels $l(x_i) \in 1, 2, ..., c$ with $i = 1, 2, ..., N$. Construct neighborghood graph $G_J = (V_j, E_j), j \in 1, 2, ..., c$ for each class respectively. The k-nearest of x_i can be expressed as:

$$k(x_i) = x_j | \left\| x_i - x_j \right\|_2 \leq d_i^k$$

where $\left\| x_i - x_j \right\|_2$ denotes the Euclidean distance between x_i and x_j and d_i^k denotes the Euclidean distance from the k sample points closest to the x_i. In the k neighborhood graph, the edge of the two observation samples e_{ij} can be expressed as:

$$e_{ij} = \begin{cases} 1, \text{ if } x_j \in k(x_i) \text{ or } x_i \in k(x_j) \\ 0, \text{ otherwise} \end{cases}$$

According to e_{ij}, the weight between points and points w_{ij} can be written as:

$$w_{ij} = \begin{cases} \left\| x_i - x_j \right\|_2, \text{ if } e_{ij} = 1 \\ \infty, \text{ otherwise} \end{cases}$$

Then geodesic distance between the individual sample points can be computed by Dijkstra shortest path algorithm, i.e.

$$g_{ij} = \begin{cases} w_{ij}, \text{ if } e_{ij} = 1 \\ min\{w_{ij}, w_{il} + w_{lj}\}, \text{ otherwise} \end{cases}$$

From this, we get the shortest paths between the points of each classes, and then by connecting the density peak points with each other, we can construct a large single neighborhood graph including all data points from different manifold by updating w_{ij} of inter-class using formula below:

$$w_{ij} - w_{ij} \times r, \ r > 1$$

Using the above geodesic distance matrix obtained as the input to the MDS algorithm, we can obtain a low dimensional embedding.

Problem 7 (programming task)

Link to git directory: https://github.com/yagehao/DD2434/tree/master/A1

1. How to Run the Code

Requirements: Jupyter notebook, Python 3
To run the code, please follow the steps below:
1. Open terminal and run
> jupyter notebook
2. open file "a1_problem7" in notebook
3. run all cells

2. Implementation and Development

Detailed implementation and code explanations can be found in the Jupyter notebook code comment sections.

Part 0: Preprocessing

This part includes I/O process and data clearing process, in which we obtained a 101×16 sized data matrix with 101 observations and 16 valid features.

Part 1: PCA

my_pca function is a PCA dimension reduction process with input arguments 'data' (data matrix with data points in rows and attributes in columns) and 'k' (number of dimensions we expected) and it will return 'low_dim_data' (data matrix after dimension reduction).

Part 2: MDS

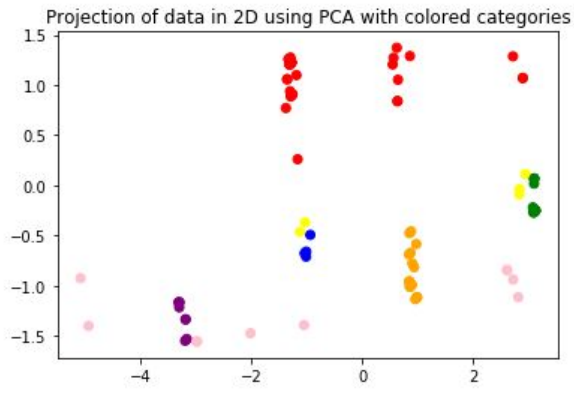
distance_func is a function to compute pairwise Euclidean distance with input argument 'data' (data_matrix, a.k.a. a position matrix) and it returns 'distance_matrix'.
my_mds function is a dimension reduction process using classical MDS algorithm with input 'data' matrix and 'k' embedding dimensions. It outputs 'low_dim_data' of the data matrix after dimension reduction.

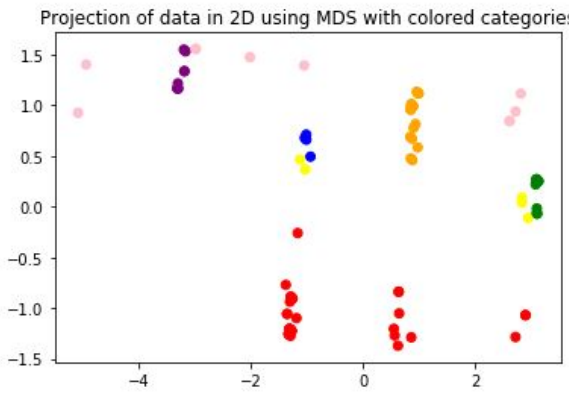
Part 3: Isomap

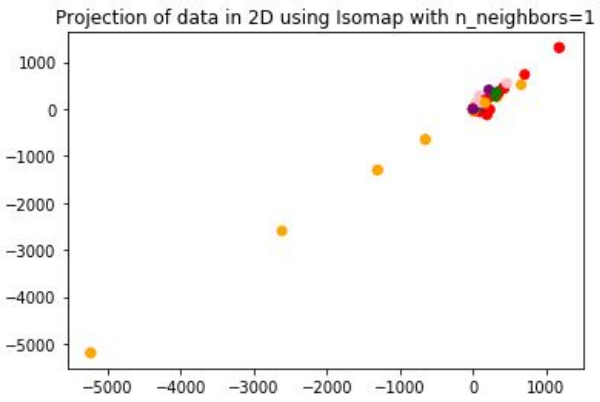
floyd is a function which construct a graph G, where vertices represent points, and each point is connected to its n_neighbors nearest points, according to Euclidean distances. And for each pair of points i and j compute the shortest path distance from i to j on the graph G using the Floyd-Warshall algorithm. The function requires input arguments distance matrix D and n_neighbors (neighborhood area to search shortest distance). It returns a shortest distance matrix D1.
iso_mds is a function designed specifically to perform MDS in the last step of Isomap algorithm. It needs input 'data' data matrix and 'k' number of intend dimensions. It will return a data matrix with lower dimensions, i.e. low_dim_data.
my_isomap is the dimension reduction process using Isomap algorithm with input original data matrix and number of embedding dimension k, returning the low dimensional data matrix.

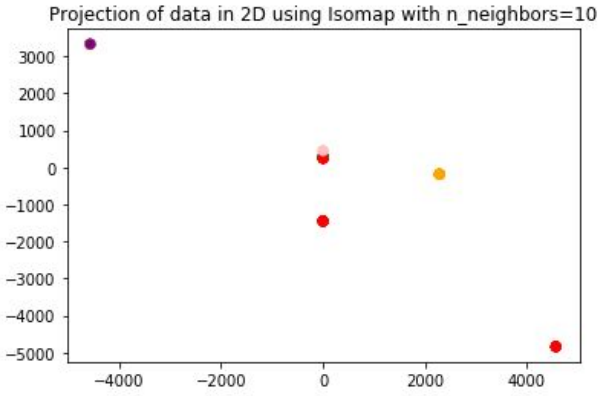
3. Results

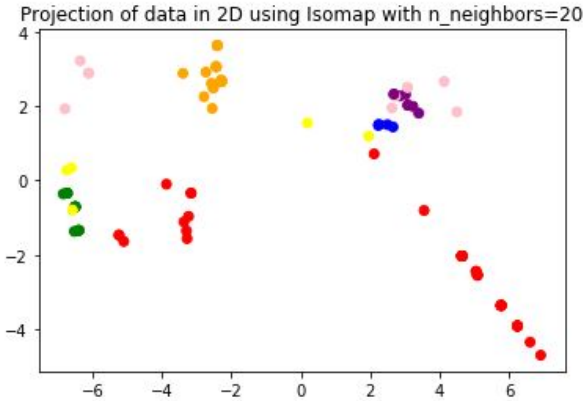
We plot the 2D projection of data points after applying PCA, MDS and Isomap dimension reduction. With Isomap technique, we plot several cases with different choice of parameter n_neighbors.

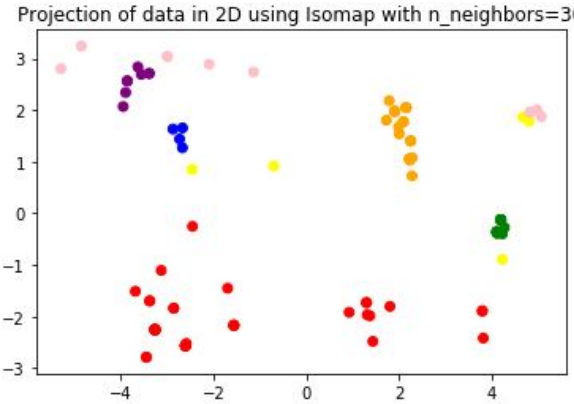

Projection of data in 2D using PCA with colored categories

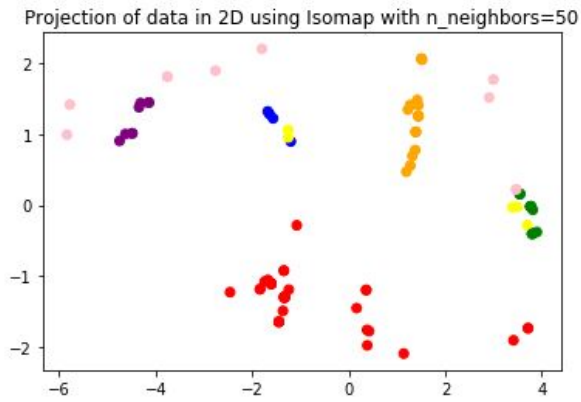

Projection of data in 2D using MDS with colored categories

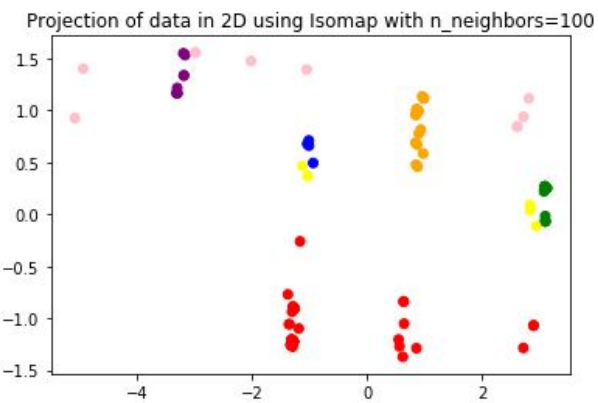

Projection of data in 2D using Isomap with n_neighbors=1


Projection of data in 2D using Isomap with n_neighbors=10


Projection of data in 2D using Isomap with n_neighbors=20


Projection of data in 2D using Isomap with n_neighbors=30


Projection of data in 2D using Isomap with n_neighbors=50


Projection of data in 2D using Isomap with n_neighbors=100

4. Observation and Conclusion

From the above, we can see that the PCA plot and MDS plot are exactly the same except their projection directions. This verifies that PCA is equivalent to classical MDS.

In observing the plots of Isomap method with different parameter n_neighbors, we can see that when n_neighbors is close to 10, the high dimensional data will be projected into high coincide 2D data under Isomap techniques. When n_neighbors is small and tends to 1, the data is not separated well by Isomap. And when n_neighbors gets larger, the data will be separated and grouped better. Specifically notice that when n_neighbors equals 100 in this case, the Isomap result is equivalent to classical MDS and PCA results since for each single point in the dataset, it can maximally possess 100 neighbor points.

Therefore, we conclude that with an unknown high dimension manifold, Isomap method provides the most comprehensive dimension reduction results requiring a careful choice on parameter n_neighbors, while PCA and classical MDS are preferable if the dataset is simple and linear separable.