An Introduction to
Multidimensional Scaling and ISOMAP

MUSTAFA HAJIJ

MDS

Let D=[d_{ij}] be an $N \times N$ dissimalrity matrix.

In MDS, we want to find n vectors $x_1, \ldots, x_N$ *in* R^d such that $||x_i - x_j|| \approx d_{ij}$

MDS

Let D=[d_{ij}] be an $N \times N$ dissimalrity matrix.

In MDS, we want to find n vectors $x_1, \dots, x_N$ in R^d such that $||x_i - x_j|| \approx d_{ij}$

- Usually if we choose d to be large enough, we can construct the *vectors* $x_1, \dots, x_N$ with exact solutions : $||x_i - x_j|| = d_{ij}$.

MDS

Let D=[d_{ij}] be an $N \times N$ dissimalrity matrix.

In MDS, we want to find n vectors $x_1, \dots, x_N$ $in\ R^d$ such that $||x_i - x_j|| \approx d_{ij}$

- Usually if we choose d to be large enough, we can construct the *vectors* $x_1, \dots, x_N$ with exact solutions : $||x_i - x_j|| = d_{ij}$.
- In this case the distance d above is the usual Euclidean distance.

MDS

Let D=[d_{ij}] be an $N \times N$ dissimalrity matrix.

In MDS, we want to find n vectors $x_1, \dots, x_N$ in R^d such that $||x_i - x_j|| \approx d_{ij}$

- Usually if we choose d to be large enough, we can construct the *vectors* $x_1, \dots, x_N$ with exact solutions : $||x_i - x_j|| = d_{ij}$.
- In this case the distance d above is the usual Euclidean distance.
- There are cases where the matrix D is valid distance matrix, but still there exists no set of vectors $x_1, \dots, x_N$ in any R^d with perfect $||x_i - x_j|| = d_{ij}$. Such a distance is called non-Euclidean distance.

Classical MDS Algorithm

1-Construct the matrix of squares of the distances $P^{(2)} = [d_{ij}^2]$.

Classical MDS Algorithm

1-Construct the matrix of squares of the distances $P^{(2)} = [d_{ij}^2]$.

2. Apply the double centering: $\mathrm{B} = -\frac{1}{2}JP^{(2)}J$ where $J = I - \frac{1}{n} 11'$, where N is the number of elements.

Classical MDS Algorithm

1-Construct the matrix of squares of the distances $P^{(2)} = [d_{ij}^2]$.

2. Apply the double centering: $\mathrm{B} = -\frac{1}{2} J P^{(2)} J$ where $J = I - \frac{1}{n} 11'$, where N is the number of elements.

3. Extract the largest d positive eigenvalues $\lambda_1 \ldots \lambda_d$ of B and the corresponding m eigenvectors $e_1 \ldots e_d$.

Classical MDS Algorithm

1-Construct the matrix of squares of the distances $P^{(2)} = [d_{ij}^2]$.

2. Apply the double centering: $\mathrm{B} = -\frac{1}{2}JP^{(2)}J$ where $J = I - \frac{1}{n} 11'$, where N is the number of elements.

3. Extract the largest d positive eigenvalues $\lambda_1 \ldots \lambda_d$ of B and the corresponding m eigenvectors $e_1 \ldots e_d$.

4. A d-dimensional MDS coordinates of n objects is derived from the coordinate matrix $X = E_d \Lambda_{\mathrm{d}}^{\frac{1}{2}}$, where E_d is the matrix of d eigenvectors and Λ_d is the diagonal matrix of d eigenvalues of B, respectively

Example

Start with a distance matrix D

$$D = \begin{matrix} 0 & 93 & 82 & 133 \\ 93 & 0 & 52 & 60 \\ 82 & 52 & 0 & 111 \\ 133 & 60 & 111 & 0 \end{matrix}$$

See this for more details

Example

Take the square the elements of D

$$D=\begin{matrix} 0 & 93 & 82 & 133 \\ 93 & 0 & 52 & 60 \\ 82 & 52 & 0 & 111 \\ 133 & 60 & 111 & 0 \end{matrix} \rightarrow \mathbf{P}^{(2)} = \begin{bmatrix} 0 & 8649 & 6724 & 17689 \\ 8649 & 0 & 2704 & 3600 \\ 6724 & 2704 & 0 & 12321 \\ 17689 & 3600 & 12321 & 0 \end{bmatrix}$$

Example

Construct the J matrix

$$D = \begin{matrix} 0 & 93 & 82 & 133 \\ 93 & 0 & 52 & 60 \\ 82 & 52 & 0 & 111 \\ 133 & 60 & 111 & 0 \end{matrix}$$

$$\rightarrow \quad \mathbf{P}^{(2)} = \begin{bmatrix} 0 & 8649 & 6724 & 17689 \\ 8649 & 0 & 2704 & 3600 \\ 6724 & 2704 & 0 & 12321 \\ 17689 & 3600 & 12321 & 0 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - 0.25 \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.75 & -0.25 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}$$

Example

Construct double centering matrix

$$D = \begin{matrix} 0 & 93 & 82 & 133 \\ 93 & 0 & 52 & 60 \\ 82 & 52 & 0 & 111 \\ 133 & 60 & 111 & 0 \end{matrix}$$

$$\mathbf{P}^{(2)} = \begin{bmatrix} 0 & 8649 & 6724 & 17689 \\ 8649 & 0 & 2704 & 3600 \\ 6724 & 2704 & 0 & 12321 \\ 17689 & 3600 & 12321 & 0 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - 0.25 \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.75 & -0.25 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}$$

$$\mathbf{B} = -\tfrac{1}{2}\mathbf{J}\mathbf{P}^{(2)}\mathbf{J} = \begin{bmatrix} 5035.0625 & -1553.0625 & 258.9375 & -3740.938 \\ -1553.0625 & 507.8125 & 5.3125 & 1039.938 \\ 258.9375 & 5.3125 & 2206.8125 & -2471.062 \\ -3740.9375 & 1039.9375 & -2471.0625 & 5172.062 \end{bmatrix}$$

Example

Solve the largest 2 eigenvalues and eigenvector of B

D= $\begin{matrix} 0 & 93 & 82 & 133 \\ 93 & 0 & 52 & 60 \\ 82 & 52 & 0 & 111 \\ 133 & 60 & 111 & 0 \end{matrix}$ → $\mathbf{P}^{(2)} = \begin{bmatrix} 0 & 8649 & 6724 & 17689 \\ 8649 & 0 & 2704 & 3600 \\ 6724 & 2704 & 0 & 12321 \\ 17689 & 3600 & 12321 & 0 \end{bmatrix}$

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - 0.25 \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.75 & -0.25 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}$$

$$\mathbf{B} = -\tfrac{1}{2}\mathbf{J}\mathbf{P}^{(2)}\mathbf{J} = \begin{bmatrix} 5035.0625 & -1553.0625 & 258.9375 & -3740.938 \\ -1553.0625 & 507.8125 & 5.3125 & 1039.938 \\ 258.9375 & 5.3125 & 2206.8125 & -2471.062 \\ -3740.9375 & 1039.9375 & -2471.0625 & 5172.062 \end{bmatrix}$$

$$\lambda_1 = 9724.168,\ \lambda_2 = 3160.986, \quad \mathbf{e_1} = \begin{pmatrix} -0.637 \\ 0.187 \\ -0.253 \\ 0.704 \end{pmatrix}, \ \mathbf{e_2} = \begin{pmatrix} -0.586 \\ 0.214 \\ 0.706 \\ -0.334 \end{pmatrix}$$

Example

Use that to construct the final MDS coordinates.

$$D = \begin{matrix} 0 & 93 & 82 & 133 \\ 93 & 0 & 52 & 60 \\ 82 & 52 & 0 & 111 \\ 133 & 60 & 111 & 0 \end{matrix} \rightarrow \mathbf{P}^{(2)} = \begin{bmatrix} 0 & 8649 & 6724 & 17689 \\ 8649 & 0 & 2704 & 3600 \\ 6724 & 2704 & 0 & 12321 \\ 17689 & 3600 & 12321 & 0 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - 0.25 \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.75 & -0.25 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}$$

$$\mathbf{B} = -\tfrac{1}{2}\mathbf{J}\mathbf{P}^{(2)}\mathbf{J} = \begin{bmatrix} 5035.0625 & -1553.0625 & 258.9375 & -3740.938 \\ -1553.0625 & 507.8125 & 5.3125 & 1039.938 \\ 258.9375 & 5.3125 & 2206.8125 & -2471.062 \\ -3740.9375 & 1039.9375 & -2471.0625 & 5172.062 \end{bmatrix}$$

$$\lambda_1 = 9724.168,\ \lambda_2 = 3160.986, \quad \mathbf{e_1} = \begin{pmatrix} -0.637 \\ 0.187 \\ -0.253 \\ 0.704 \end{pmatrix}, \mathbf{e_2} = \begin{pmatrix} -0.586 \\ 0.214 \\ 0.706 \\ -0.334 \end{pmatrix} \rightarrow \mathbf{X} = \begin{bmatrix} -0.637 & -0.586 \\ 0.187 & 0.214 \\ -0.253 & 0.706 \\ 0.704 & -0.334 \end{bmatrix} \begin{bmatrix} \sqrt{9724.168} & 0 \\ 0 & \sqrt{3160.986} \end{bmatrix} = \begin{bmatrix} -62.831 & -32.97448 \\ 18.403 & 12.02697 \\ -24.960 & 39.71091 \\ 69.388 & -18.76340 \end{bmatrix}$$

Example

Use that to construct the final MDS coordinates.

$$D = \begin{matrix} 0 & 93 & 82 & 133 \\ 93 & 0 & 52 & 60 \\ 82 & 52 & 0 & 111 \\ 133 & 60 & 111 & 0 \end{matrix} \rightarrow \mathbf{P}^{(2)} = \begin{bmatrix} 0 & 8649 & 6724 & 17689 \\ 8649 & 0 & 2704 & 3600 \\ 6724 & 2704 & 0 & 12321 \\ 17689 & 3600 & 12321 & 0 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - 0.25 \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.75 & -0.25 & -0.25 & -0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}$$

$$\mathbf{B} = -\tfrac{1}{2}\mathbf{J}\mathbf{P}^{(2)}\mathbf{J} = \begin{bmatrix} 5035.0625 & -1553.0625 & 258.9375 & -3740.938 \\ -1553.0625 & 507.8125 & 5.3125 & 1039.938 \\ 258.9375 & 5.3125 & 2206.8125 & -2471.062 \\ -3740.9375 & 1039.9375 & -2471.0625 & 5172.062 \end{bmatrix}$$

Coordinates of first point

$$\lambda_1 = 9724.168,\ \lambda_2 = 3160.986, \quad \mathbf{e_1} = \begin{pmatrix} -0.637 \\ 0.187 \\ -0.253 \\ 0.704 \end{pmatrix}, \ \mathbf{e_2} = \begin{pmatrix} -0.586 \\ 0.214 \\ 0.706 \\ -0.334 \end{pmatrix} \rightarrow \mathbf{X} = \begin{bmatrix} -0.637 & -0.586 \\ 0.187 & 0.214 \\ -0.253 & 0.706 \\ 0.704 & -0.334 \end{bmatrix} \begin{bmatrix} \sqrt{9724.168} & 0 \\ 0 & \sqrt{3160.986} \end{bmatrix} = \begin{bmatrix} -62.831 & -32.97448 \\ 18.403 & 12.02697 \\ -24.960 & 39.71091 \\ 69.388 & -18.76340 \end{bmatrix}$$

Stress Majorization

In the classical MDS algorithm the cost function that we are trying to optimize is called the stress function and it is given by :

$$Stress_D(x_1, x_2, \ldots, x_N) = \left(\sum_{i \neq j = 1,\ldots,N} \left(d_{ij} - \|x_i - x_j\|\right)^2 \right)^{1/2}$$

In this function we try to find $x_1, \ldots, x_N$ in a certain dimension d such that Stress($x_1, \ldots, x_N$) is as small as possible

Stress Majorization

The stress function has a more general form as :

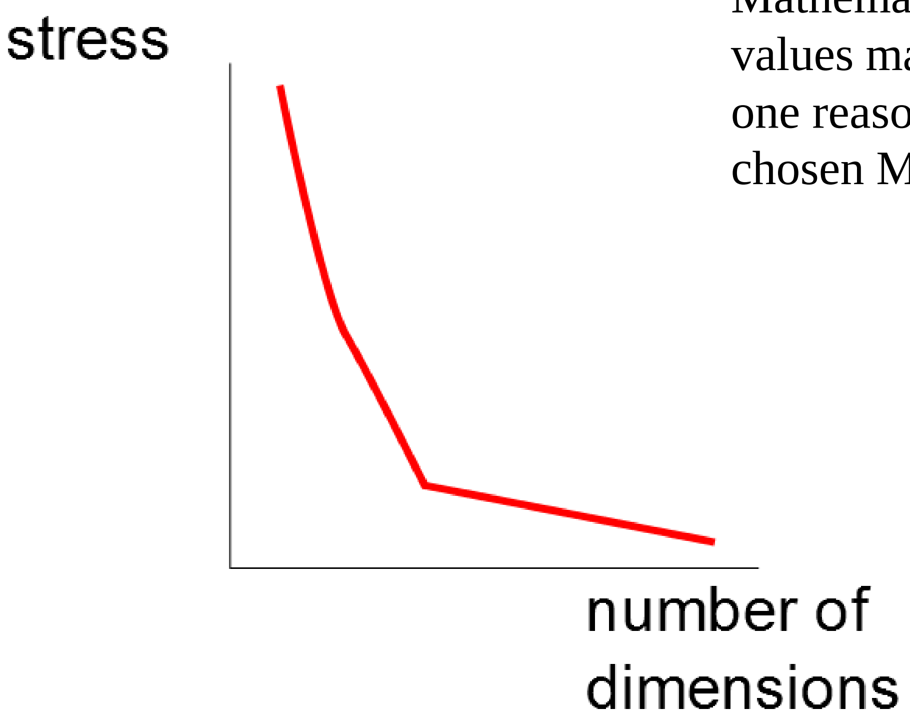
$$\sigma(X) = \sum_{i<j\leq n} w_{ij}(d_{ij}(X) - \delta_{ij})^2$$

Here wij weight between a pair of points (i,j) that represents the confidence in in the similarity between points (i,j) .

δ_{ij} the given distance between the points i,j

Stress Majorization

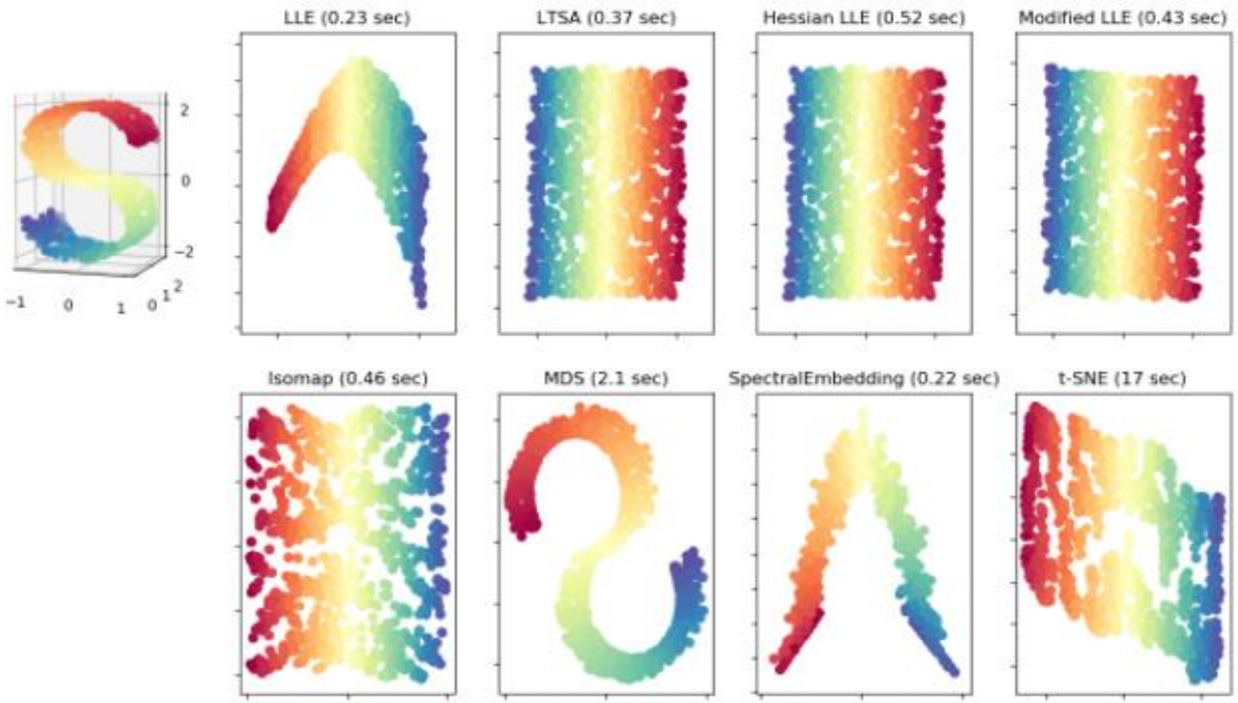
“Pressing” the data into 2 dimensions enables us to visualize the data. However, that comes with a price : high stress function value (which correlates wit distorted representation)



Mathematically non-zero stress values may occur only for only one reason: dimensionality of the chosen MDS projection is too low.

MDS in Sklearn

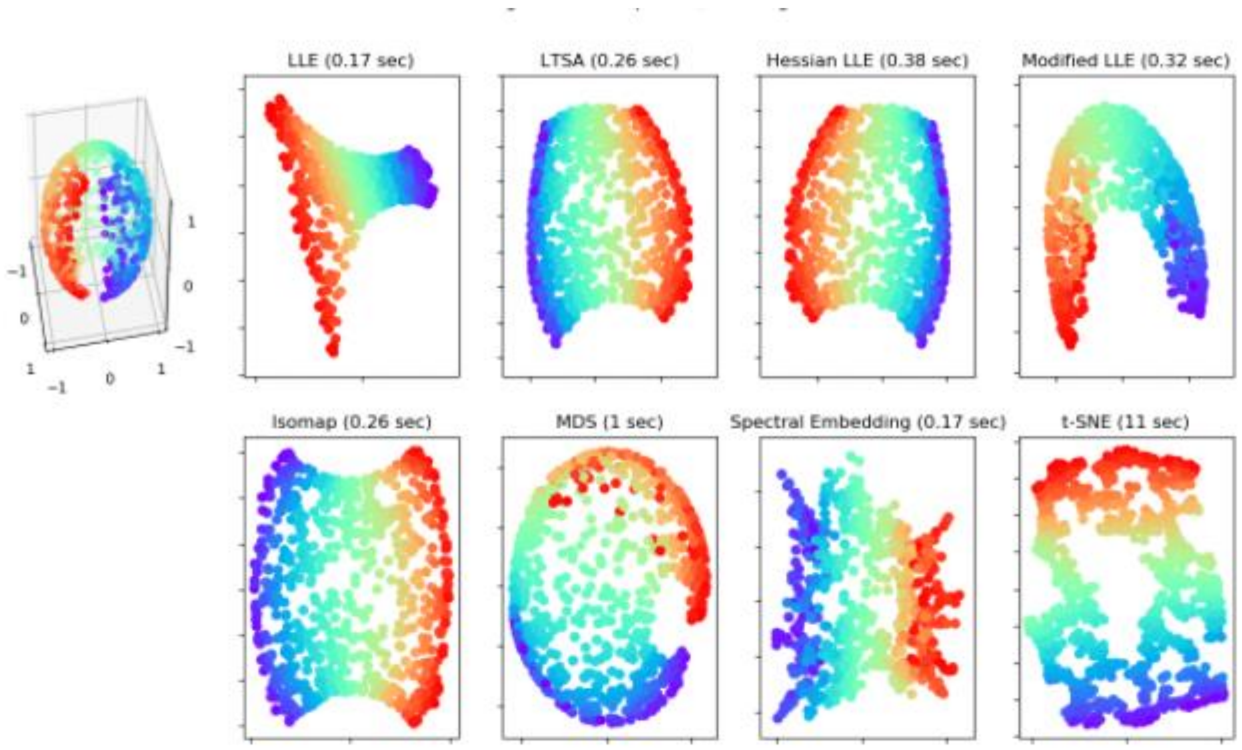
MDS is implemented in [Sklearn]



[Example]

MDS in Sklearn

MDS is implemented in [Sklearn]



[Example]

MDS in Sklearn

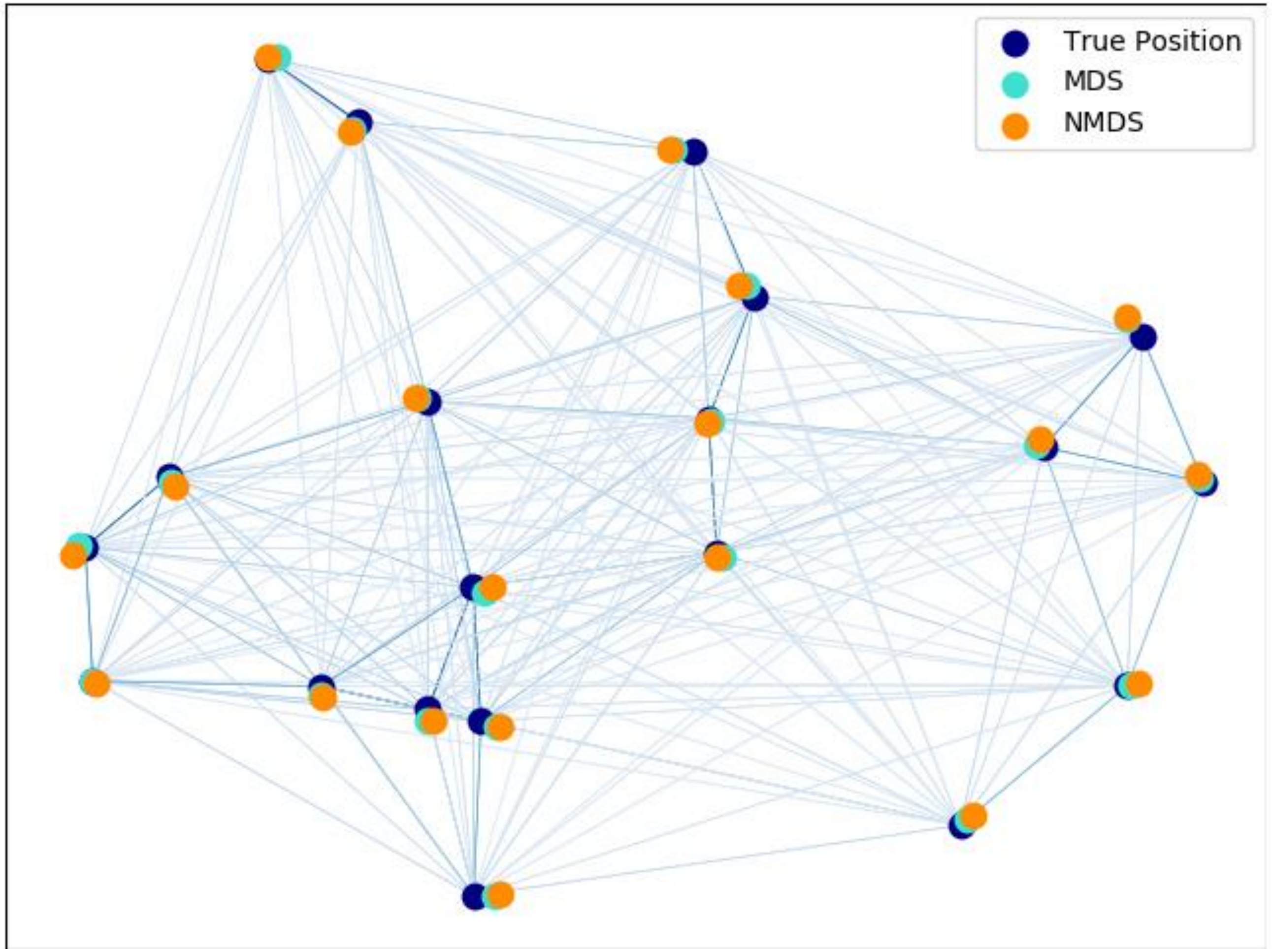
MDS is implemented in [Sklearn]

A selection from the 64-dimensional digits dataset

[Example]

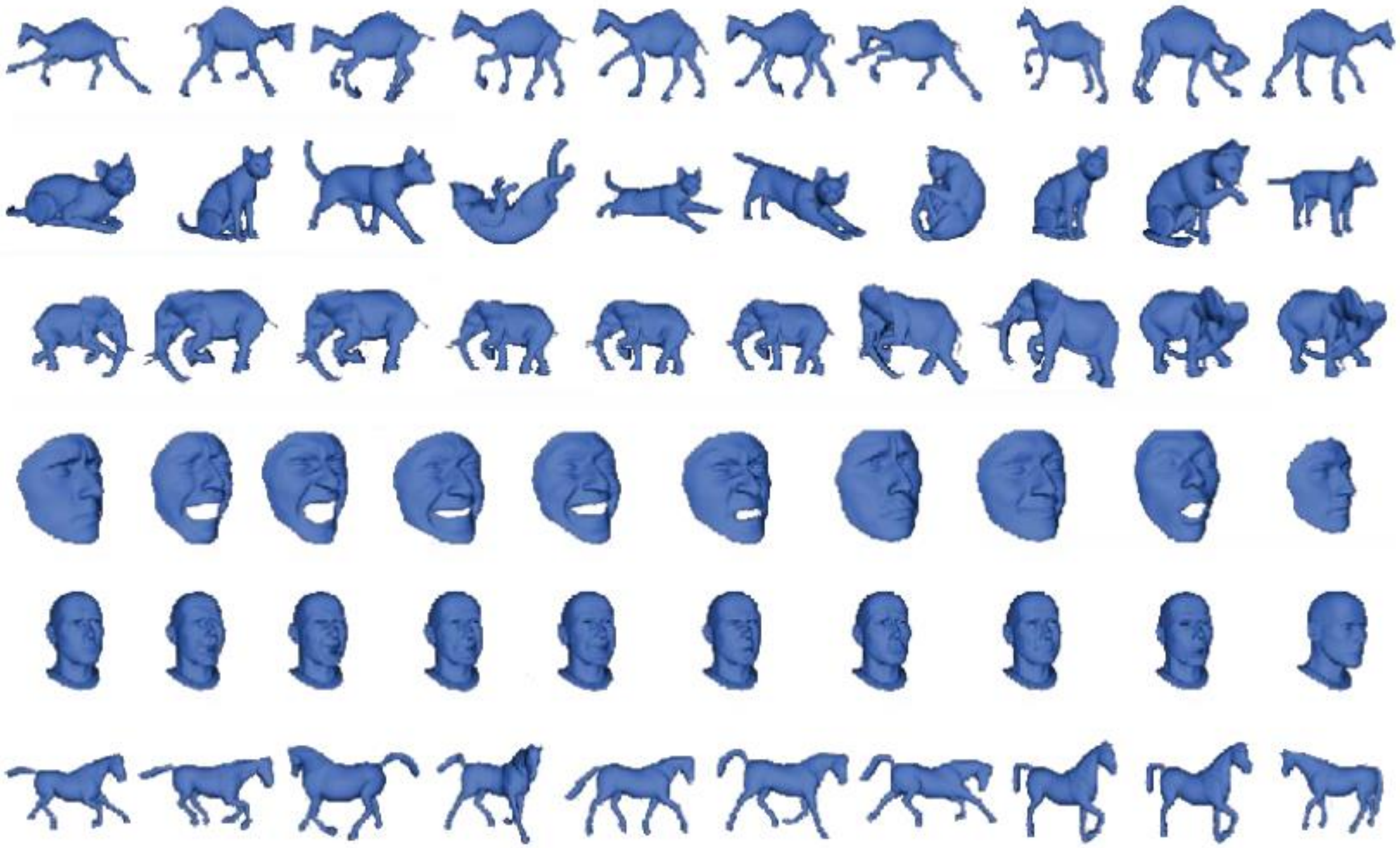
MDS in Sklearn

MDS is implemented in Sklearn

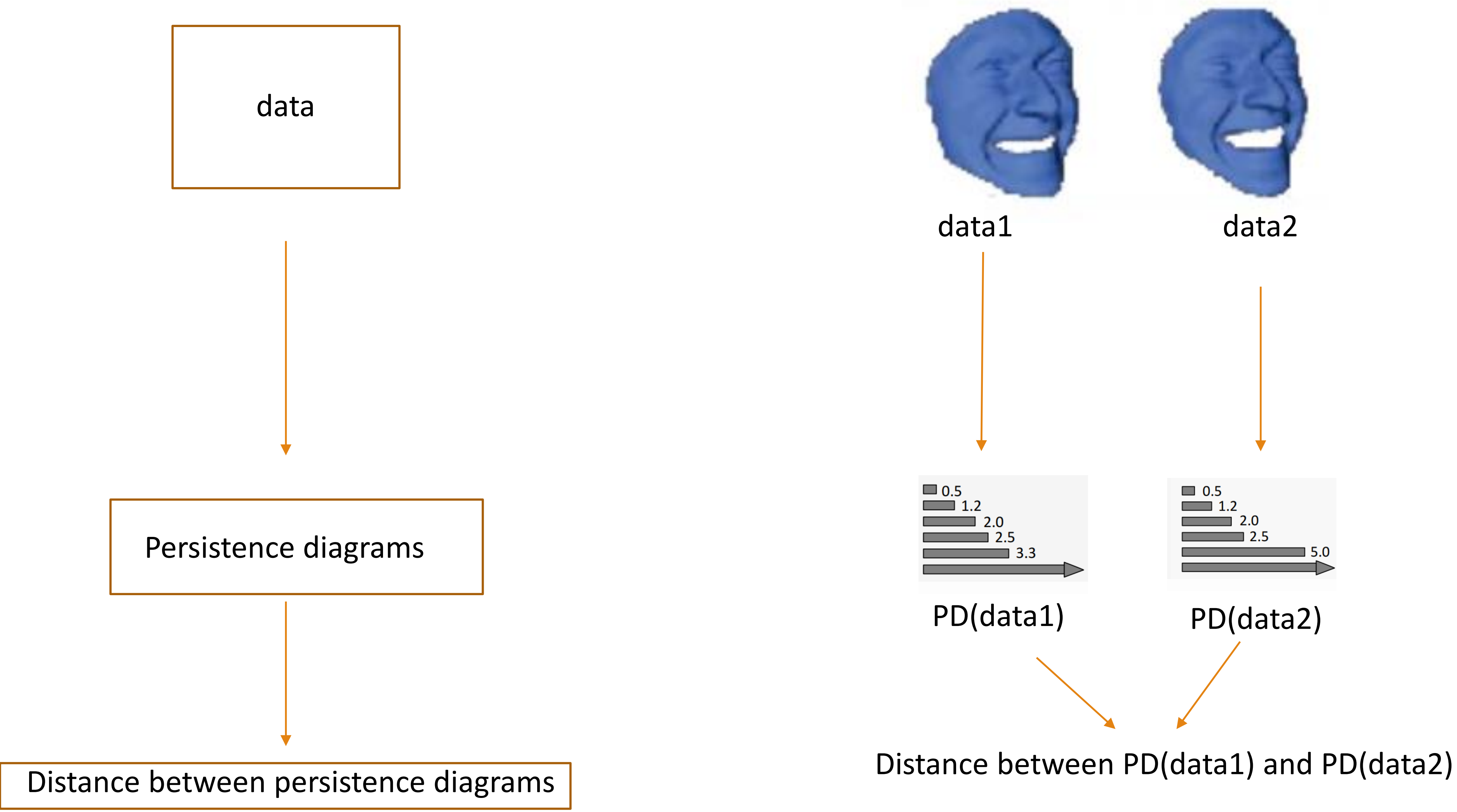


Example

Application



Measuring distance between two persistence diagrams



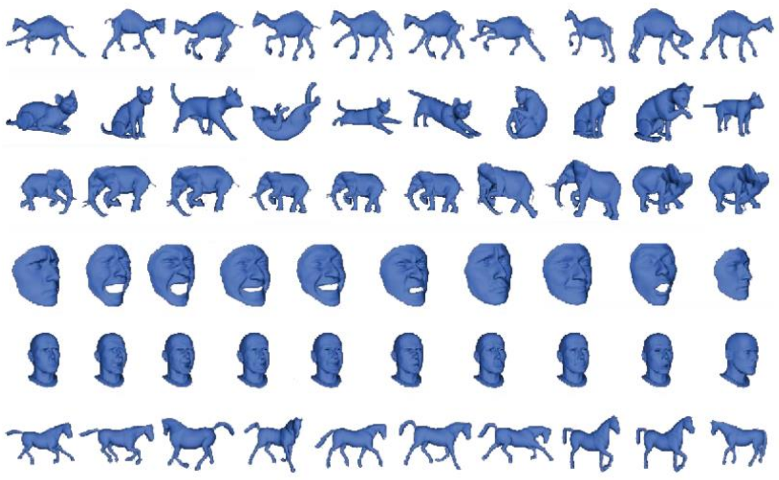
Bottleneck distance between two persistent diagrams

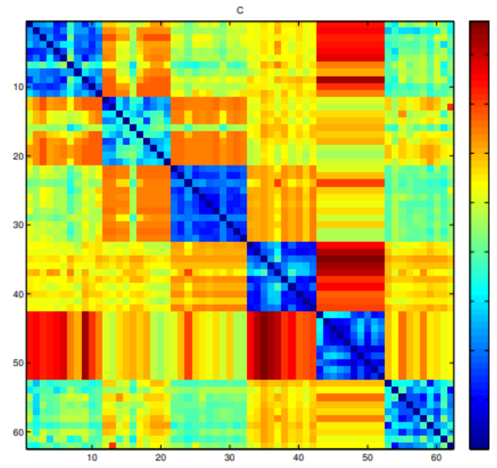
- Given two persistence diagrams X and Y, let η be a bijection between points in the diagram. The following two distances are commonly used in the context of PH to measure the distance between two persistence diagrams:

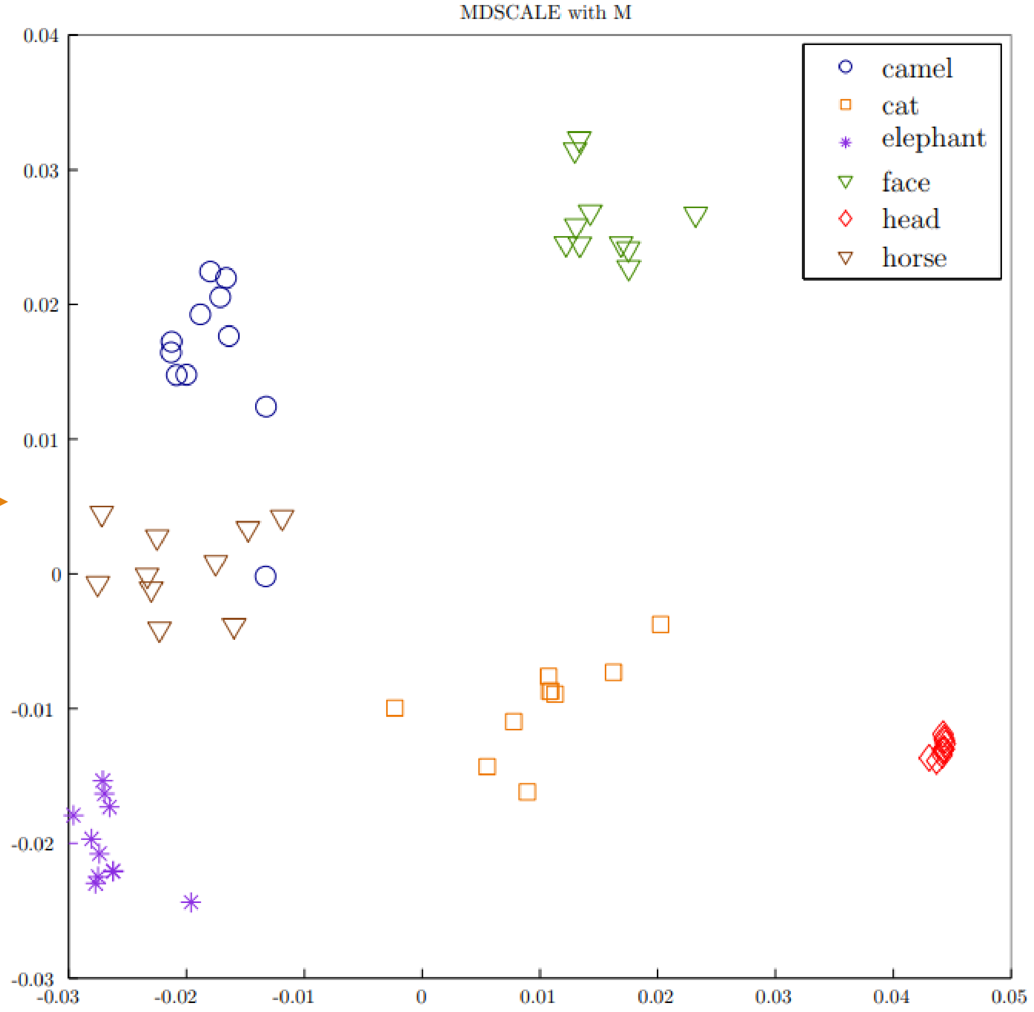
$$W_\infty(X,Y) = \inf_{\eta:X\to Y} \sup_{x\in X} \|x - \eta(x)\|_\infty$$

$$W_q(X,Y) = \left[\inf_{\eta:X\to Y} \Sigma_{x\in X} \|x - \eta(x)\|_\infty^q \right]^{1/q}$$

Bottleneck distance between two persistent diagrams







Input data

Matrix M describes the pair-wise distance between the persistence diagrams of each data element

MDS plot of the matrix M with labels corresponding to each class.

Remarks

- The axes obtained when drawing the MDS coordinates are , in themselves, meaningless.

Remarks

- The axes obtained when drawing the MDS coordinates are , in themselves, meaningless.
- The orientation of the result MDS “picture” is arbitrary.

Remarks

- The axes obtained when drawing the MDS coordinates are , in themselves, meaningless.

- The orientation of the result MDS “picture” is arbitrary.

- When we obtain MDS coordinates that have non-zero stress, we should remember that the distances among the resulting items are distorted representations of the relationships given by the input data. This distortion is greater when the stress is greater.
- That being said, we, in general, can rely on the larger distances as being more accurate than smaller distances.

Non-Matric MDS

- Sometimes, there is no defined metric on points and all we are given is a similarity measure between the points.

Non-Matric MDS

- Sometimes, there is no defined metric on points and all we are given is a similarity measure between the points.

The main idea in non-metric MDS :

- The actual values given to us are not that meaningful
- *Ranking among different points* is important

- Non-metric MDS finds a low-dimensional representation, which respects the ranking of distances as much as possible

Non-Matric MDS

- Recall that in MDS we seek to find an optimal configuration xi that $gives\ d_{ij} \approx d'_{ij} = ||x_i - x_j||$ as close as possible.

Non-Matric MDS

- Recall that in MDS we seek to find an optimal configuration xi that $gives\ d_{ij}\ \approx\ d'_{ij}\ = ||x_i\ -\ x_j||$ as close as possible.
- Relaxing $dij\ \approx\ d'\ ij$ from MDS by allowing$d'\ ij \approx\ f\ (d_{ij})$, for some monotone function f

Monotonic means : $d_{ij} <\ d_{kl}\ \Leftrightarrow\ f\ (dij)\ \leq\ f\ (d_{kl})$

Non-Matric MDS

- Recall that in MDS we seek to find an optimal configuration xi that $gives\ d_{ij} \approx d'_{ij} = ||x_i - x_j||$ as close as possible.

- Relaxing $dij \approx d'\ ij$ from MDS by allowing$d'\ ij \approx f\ (d_{ij})$, for some monotone function f

Monotonic means : $d_{ij} < d_{kl} \Leftrightarrow f\ (dij) \leq f\ (d_{kl})$

Given a dimension d, non-metric MDS seeks to find an optimal configuration $X \subset R^d$ that gives
$f\ (d_{ij}) \approx d\hat{}\ ij = ||x_i - x_j||$ as close as possible.

- $f\ (d_{ij}) = d^*ij$ is only required to preserve the order of d_{ij} ,

i.e., $d_{ij} < d_{kl} \Leftrightarrow f\ (d_{ij}) \leq f\ (d_{kl}) \Leftrightarrow d^*_{ij} \leq d^*_{kl}$

Non-Matric MDS

The stress function for non-metric MDS is given by :

$$Stress = \left(\sum_{i<j} (\hat{d}_{ij} - f(d_{ij}))^2 / \sum d_{ij}^2 \right)^{\frac{1}{2}}$$

Non-metric MDS optimizes over both position of the points of points and f

Non-Matric MDS

The stress function for non-metric MDS is given by :

$$Stress = \left(\sum_{i<j} (\hat{d}_{ij} - f(d_{ij}))^2 / \sum d_{ij}^2 \right)^{\frac{1}{2}}$$

Non-metric MDS optimizes over both position of the points of points and f

Solved numerically using (isotonic regression); we usually use classical MDS as starting initial position.

Non-Matric MDS

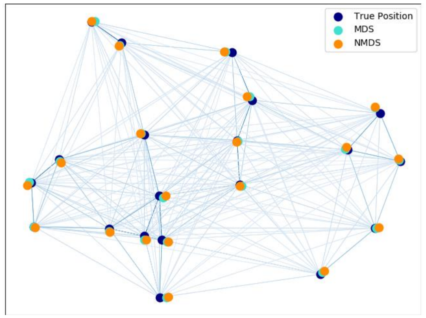
The stress function for non-metric MDS is given by :

$$Stress = \left(\sum_{i<j} (\hat{d}_{ij} - f(d_{ij}))^2 / \sum d_{ij}^2 \right)^{\frac{1}{2}}$$

Non-metric MDS optimizes over both position of the points of points and f

Solved numerically using (isotonic regression); we usually use classical MDS as starting initial position.

Example



ISOMAP

- Isomap extends MDS by utilizing geodesic distances induced by some neighborhood graph.
- There is no essential difference between MDS and ISOMAP algorithm once we find the distance matrix induced by the neighborhood graph.

ISOMAP

- Isomap extends MDS by utilizing geodesic distances induced by some neighborhood graph.
- There is no essential difference between MDS and ISOMAP algorithm once we find the distance matrix induced by the neighborhood graph.

So the steps for ISOMAP on a given data :

1- Construct the neighborhood graph of the data X using one of the neighborhood graphs we studied earlier in the course

ISOMAP

- Isomap extends MDS by utilizing geodesic distances induced by some neighborhood graph.
- There is no essential difference between MDS and ISOMAP algorithm once we find the distance matrix induced by the neighborhood graph.

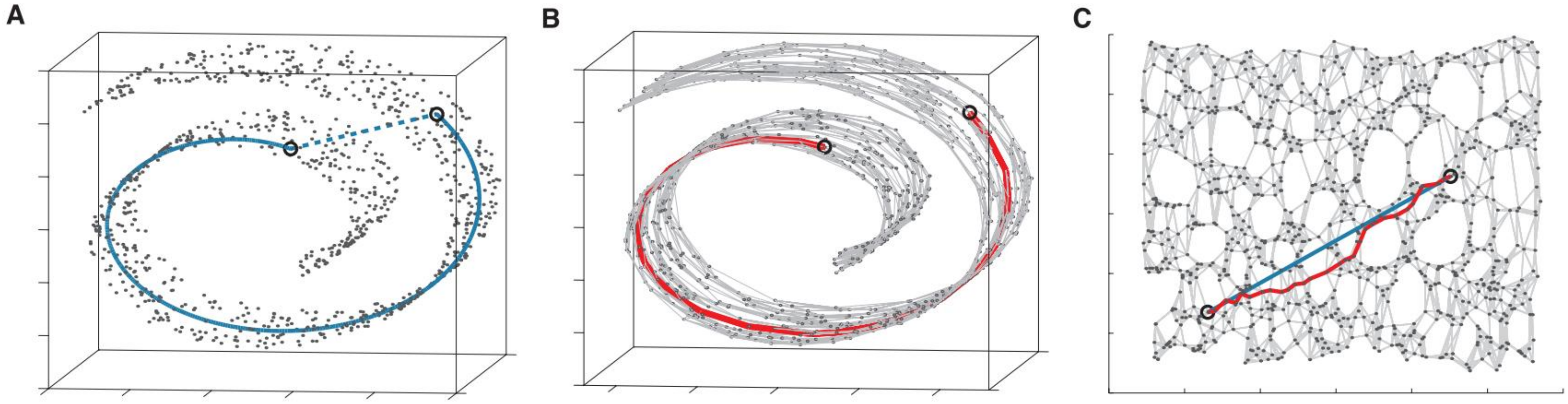
So the steps for ISOMAP on a given data :
1- Construct the neighborhood graph of the data X using one of the neighborhood graphs we studied earlier in the course
2- Use the Dijekstra algorithm or the Floyd–Warshall algorithm to find the distance between nodes on the graph

ISOMAP

- Isomap extends MDS by utilizing geodesic distances induced by some neighborhood graph.
- There is no essential difference between MDS and ISOMAP algorithm once we find the distance matrix induced by the neighborhood graph.

So the steps for ISOMAP on a given data :
1- Construct the neighborhood graph of the data X using one of the neighborhood graphs we studied earlier in the course
2- Use the Dijekstra algorithm or the Floyd–Warshall algorithm to find the distance between nodes on the graph
3- Apply MDS on the distance matrix above and extract the coordinates with the desired dimension

ISOMAP



A- Euclidian distance might not represent the actual distance between the points in the data.
B- We can construct the neighborhood graph of the data and then compute the geodesic distance between the points of the graph
C- Embedding the space we obtained in B into the plane.

Figure reference

Appendix : Floyd–Warshall algorithm

```
let dist be a |V| × |V| array of minimum distances
initialized to infinity

for each edge (u,v)
 dist[u][v] ← w(u,v)    // the weight of the edge (u,v)
for each vertex v
 dist[v][v] ← 0
for k from 1 to |V|
 for i from 1 to |V|
    for j from 1 to |V|
          if dist[i][j] > dist[i][k] + dist[k][j]
              dist[i][j] ← dist[i][k] + dist[k][j]
          end if
```

Floyd algorithm is good to use when we want to compute the distance matrix on a dense graph.
When the graph G is sparse, Dijekstra algorithm is a better choice.