ML
P1

MDS (Multidimensional Scaling).

Given N objects s.t. a distance function is defined on them. (In particular, $\mathcal{X}=\{x_1,\cdots,x_N\}\subseteq\mathbb{R}^D$.)

$\boxed{\delta_{ij}}$:= distance b/w object i and object j.

$\boxed{\Delta} := (\delta_{ij})_{N\times N}$ is called the $\boxed{\text{dissimilarity matrix}}$.

Goal: Given Δ. Try to find $\mathcal{Y}=\{y_1,\cdots,y_N\}\subseteq\mathbb{R}^d$

s.t. $\|y_i-y_j\|\approx\delta_{ij}$, $\forall\, i,j=1,\cdots,N$.

Formally, we can achieve this via solving the following optimization problem:

$$\min_{\substack{y_1,\cdots,y_N\\ \in\mathbb{R}^d}}\sum_{i<j}\left(\|y_i-y_j\|-\delta_{ij}\right)^2. \quad (*)$$

Rmk:

① MDS is a dimension reduction method when our input is $\mathcal{X}=\{x_1,\cdots,x_N\}\subseteq\mathbb{R}^D$ and we choose $d\ll D$.

② In general, the input is just the dissimilarity matrix Δ.

③ MDS is a "global" method since it tries to minimize $(*)$, which is the total error made when choosing $\mathcal{Y}$.

④ MDS can be solved using optimization algorithms such as gradient descent.

Isomap (Isometric map)

Algorithm:

Inputs: ① $\mathcal{X}=\{x_1,\cdots,x_N\}\subseteq\mathbb{R}^D$.
② $d\ll D$. ③ $\varepsilon>0$ or $K\in\mathbb{N}$

Output: $\mathcal{Y}=\{y_1,\cdots,y_N\}\subseteq\mathbb{R}^d$.

Procedure:

1° Determine the neighbors of each pt by
(i) ε-neighborhoods, or
(ii) K nearest neighborhoods. (KNN).

2° Construct a neighborhood graph G:
Vertex set: $V=\mathcal{X}=\{x_1,\cdots,x_N\}$.
The edge set E is determined by
(i) $\overline{x_ix_j}\in E \Leftrightarrow \|x_i-x_j\|<\varepsilon$, or
(ii) $\overline{x_ix_j}\in E \Leftrightarrow x_i$ is a KNN of x_j or x_j " " " of x_i.
(iii) $\overline{x_ix_j}\in E$ has weight w_{ij} equal to $\|x_i-x_j\|$.

3° Compute shortest path between any x_i, x_j:
The graph distance b/w $x_i,x_j\in\mathcal{X}$ is defined by $d_G(x_i,x_j):=\min\{L(\gamma)\mid\gamma$ is a path in G from x_i to $x_j\}$
where, if $\gamma=x_{i_1}-x_{i_2}-\cdots-x_{i_K}$,
then $L(\gamma):=\sum_{j=1}^{K-1}w_{i_ji_{j+1}}$.

Rmk:
Commonly, people use Dijkstra's algorithm to compute $d_G(x_i,x_j)$.

4° Denote $\Delta_G(\mathcal{X})=[d_G(x_i,x_j)]$.
Apply MDS on $\Delta_G(\mathcal{X})$ to obtain a low dimensional pt cloud
$\mathcal{Y}=\{y_1,\cdots,y_N\}\subseteq\mathbb{R}^d$ (w/ $\|y_i-y_j\|\approx d_G(x_i,x_j)$ $\forall\, i,j$).