Chap 6 Classical Multidimensional Scaling.

New ~ Week 0

6.1 Intro to Multidim. Scaling.

Input: a similarity matrix or an array of similarity matrices.

Output: a low dimensional data set in some Euclidean space.

MDS = multidimensional scaling.

MDS $\begin{cases} \text{qualitative (nonmetric MDS)} \\ \text{quantitative (metric MDS)} \end{cases}$.

By number of similarity matrices,

(1) classical MDS.
(one similarity matrix in an unweighted model).

(2) replicated MDS
(several .. matrices .. ..)

(3) weighted MDS
(.. .. .. .. a weighted ..)
(i.e. different similarity matrices have different importance.)

6.2 Euclidean Metric and Gram Matrices.

In classical MDS (CMDS), the similarity matrix is assumed to come from data pts in an Euclidean space and distances b/w them.

Goal: Find "configuration" $\mathcal{Y} \subseteq \mathbb{R}^d$ s.t. the Euclidean distance matrix of $\mathcal{Y}$ best approximates the given similarities.

Def

$\mathcal{X} = \{x_1, \cdots, x_n\} \subseteq \mathbb{R}^D$. ($d_2 :=$ the Euclidean distance).

The Euclidean distance matrix on $\mathcal{X}$ is

$$\boxed{D} := [d_2(x_i, x_j)]_{i,j=1}^n .$$

The Euclidean square-distance matrix on $\mathcal{X}$ is

$$\boxed{S} := [d_2^2(x_i, x_j)]_{i,j=1}^n .$$

Rmk: D and S above are symmetric and invariant of shift and rotation on $\mathcal{X}$.

Notation:

For $a \in \mathbb{R}^D$, $\boxed{\mathcal{X}_a} := \{x + a \mid x \in \mathcal{X}\}$.

For rotation R (orthogonal w/ det 1),

$$\boxed{\mathcal{X}_R} := \{Rx \mid x \in \mathcal{X}\}.$$

Def D: $n \times n$ symmetric matrix.

D: Euclidean distance matrix if

$\exists$ integer $m > 0$, $\mathcal{Z} = \{z_1, \cdots, z_n\} \subseteq \mathbb{R}^m$ s.t.

$$D = [d_2(z_i, z_j)]_{i,j=1}^n .$$

Such $\mathcal{Z}$ is called a configurative point set (or a configuration) of D.

Q:

How to determine whether a given symmetric matrix is a Euclidean distance matrix?

Def $\mathcal{X} = \{x_1, \cdots, x_n\} \subseteq \mathbb{R}^D$.

The Gram matrix on the data set $\mathcal{X}$ is

$$\boxed{G} = [\langle x_i, x_j \rangle]_{i,j=1}^n .$$

Rmk:

① $X := \begin{pmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{pmatrix}$, the $D \times n$ data matrix

Then $G = X'X$. Thus, G: psd. (positive semidef.)

② $D_{ij} = \sqrt{G_{ii} + G_{jj} - 2G_{ij}}$ ⊛

i.e. $S = \delta(G)\mathbf{1}^T + \mathbf{1}\delta(G)^T - 2G$.
where $\delta(G)$: diagonal elements of G
$\mathbf{1} = [1, 1, \cdots, 1]^T$.
i.e. we can recover S (so D) via G.

Def $\mathcal{X} = \{x_1, \cdots, x_n\} \subseteq \mathbb{R}^D$, a data set.

$\boxed{\bar{x}} := \frac{1}{n}\sum_i x_i$

$\boxed{\hat{\mathcal{X}}} := \mathcal{X}_{-\bar{x}} = \{\underbrace{x_i - \bar{x}}_{\hat{x}_i} \mid i = 1, \cdots, n\}$, called a centered data set.

$\boxed{\hat{X}} := \begin{bmatrix} | & & | \\ \hat{x}_1 & \cdots & \hat{x}_n \\ | & & | \end{bmatrix}$, called the centered data matrix.

$\boxed{G^c} = [\langle \hat{x}_i, \hat{x}_j \rangle]_{i,j=1}^n = \hat{X}'\hat{X}$, called the centering Gram matrix of $\mathcal{X}$.

In general, for $a \in \mathbb{R}^D$, the a-shifted Gram matrix of $\mathcal{X}$ is

$$\boxed{G^a} = [\langle x_i - a, x_j - a \rangle]_{i,j=1}^n .$$

Rmk: As before, $D_{ij} = \sqrt{G^c_{ii} + G^c_{jj} - 2G^c_{ij}}$.

Def

$\boxed{\mathbb{1}} := [1,1,\cdots,1]' \in \mathbb{R}^n$.

$\boxed{E} := \mathbb{1}\mathbb{1}'$, the $n\times n$ matrix w/ all entries $=1$.

$\boxed{H} := I - \frac{1}{n}E$, called the **n-centralizing matrix**.

(If n is understood, omit "n-").

Lemma 6.1

H has the following properties:

(1) $H^2 = H$.

(2) $\mathbb{1}'H = H\mathbb{1} = 0$.

(3) $\mathcal{X}$: centered data set $\Leftrightarrow XH = X$.

(4) A psd C is a centering Gram matrix $\Leftrightarrow HCH = C$.

⟨Pf⟩

(1), (2) : direct computation.

(3) : note $\mathcal{X}$: centered $\Leftrightarrow X\mathbb{1} = 0$. (*)

(4) : ($\Rightarrow$) by (3). ($\Leftarrow$) use (*). #

Lemma 6.2

X : data matrix, G : Gram matrix of X.

Then

XH : the centered data matrix of X, and

$HGH = G^c$, the centering Gram matrix of X.

⟨Pf⟩

XH: by def. HGH: by def. #

Def

A: symmetric $n\times n$ matrix (not necessarily psd).

The **centering matrix** of A, denoted $\boxed{A^c}$, is

$A^c = HAH$. Rmk: A symmetric S is centering $\Leftrightarrow HSH = S$.

Thm 6.1 (*) $\mathcal{X}$: data set.

$S :=$ Euclidean square-distance matrix of $\mathcal{X}$.

$G^c :=$ centering Gram matrix of $\mathcal{X}$.

Then $G^c = -\frac{1}{2}S^c \left(= -\frac{1}{2}HSH\right)$.

(we can find G^c via S).

⟨Pf⟩

Straightforward computations. #

The following gives an answer for how to determine whether a matrix is Euclidean distance:

Thm 6.2

A : symmetric matrix. Then

(1) A : a Gram matrix $\Leftrightarrow A$: psd.

(2) A : centering Gram $\Leftrightarrow A$: centering psd.

(3) A : Euclidean square-distance matrix

$\Leftrightarrow -\frac{1}{2}A^c$: centering psd, and $A = \delta(G)\mathbb{1}^T + \mathbb{1}\delta(G)^T - 2G$,

where $G = -\frac{1}{2}A^c$.

6.3 Description of Classical Multidimensional Scaling.

Idea: (of metric multidimensional scaling (MMDS)).

$D = [d_{ij}]_{i,j=1}^n$ distance matrix of n object.

$d > 0$: an integer.

The goal is to find $\mathcal{Y} = \{y_1,\cdots,y_n\} \subseteq \mathbb{R}^d$ s.t.

$$d_Y(y_i,y_j) \approx d_{ij},\ \forall i,j. \quad (*)$$

In practice, closeness of (*) is measured by a reasonable "lost function".

Lemma 6.3

$D = [d_{ij}]_{i,j=1}^n$, Euclidean distance matrix.

$S = [d_{ij}^2]$: its square-distance " .

$G^c := -\frac{1}{2}S^c$ (i.e. its centering Gram matrix).

$r := \operatorname{rank}(G^c)$.

Then $\exists$ r-dim. centered data set $\mathcal{X} = \{x_1,\cdots,x_n\} \subseteq \mathbb{R}^r$

s.t. $d_2(x_i,x_j) = d_{ij},\ \forall i,j$.

Def

$\boxed{r} = \operatorname{rank}(G^c)$ in Lemma 6.3 is called the **intrinsic configuration dimension** of D.

$\mathcal{X}$ in L 6.3 is called an **exact configurat'n** of D.

Rmk:

The lost funct'n of CMDS is set to be

$$\boxed{\eta(\mathcal{Y})} = \sum_{i,j=1}^n \left(d_{ij}^2 - d_2^2(y_i,y_j)\right),\ \text{s.t. } \mathcal{Y} = T(\mathcal{X}),\quad \begin{matrix}\mathbb{R}^r\\ \cup\end{matrix}$$

where T : o.g. project'n from $\mathbb{R}^r$ to some d-subsp. S_d.

$\mathcal{X}$: an exact configurat'n of D.

i.e. we are solving the minimization problem:

$$\boxed{\underset{Y\in\mathcal{M}_{d,n}}{\arg\min}\ \eta(\mathcal{Y}),\ \text{where } Y = [y_1 \cdots y_n],\ \mathcal{Y} = \{y_1,\cdots,y_n\}}$$

Lemma 6.4

$\mathcal{Z}=\{z_1,\cdots,z_n\}\subseteq\mathbb{R}^r$, a given data set.

$S_Z := [s_{ij}]$, where $s_{ij}=d_2^2(z_i,z_j)$.

$G_Z^c :=$ its centering Gram matrix.

Then $$tr(G_Z^c)=\frac{1}{2n}\sum_{i=1}^n\sum_{j=1}^n s_{ij}.$$

<Pf>

By Thm 6.1, $G_Z^c=\frac{-1}{2}S_Z^c=\frac{-1}{2}HS_ZH$.

$$\Rightarrow tr(G_Z^c)=tr(\tfrac{-1}{2}HS_ZH)=tr(\tfrac{-1}{2}H^2S_Z)=\tfrac{-1}{2}tr(HS_Z)$$
$$=\tfrac{-1}{2}tr((I-\tfrac{1}{n}E)S_Z)=\tfrac{-1}{2}[\underbrace{tr(S_Z)}_{=0}-\tfrac{1}{n}tr(ES_Z)]$$
$$=\frac{1}{2n}tr(ES_Z)=\frac{1}{2n}\sum_{i=1}^n\sum_{j=1}^n s_{ij}. \quad \#$$

Lemma 6.5

$D_Z := [d_2(z_i,z_j)]_{i,j=1}^n$, $\hat{Z} := [\hat{z}_1,\cdots,\hat{z}_n]$.

Then $$\|\hat{Z}\|_F=\frac{1}{\sqrt{2n}}\|D_Z\|_F. \quad \left(\text{Recall: } \|A\|_F:=\sqrt{tr(A^*A)}.\right)$$

Frobenius norm.

<Pf>

$$\|D_Z\|_F^2=\sum_{i=1}^n\sum_{j=1}^n s_{ij}\overset{\text{by L6.4}}{=}2n\,tr(G_Z^c)=2n\cdot tr(\hat{Z}'\cdot\hat{Z})$$
$$=2n\|\hat{Z}\|_F^2. \text{ Thus, } \|\hat{Z}\|_F=\frac{1}{\sqrt{2n}}\|D_Z\|_F. \quad \#$$

Thm 6.3

$\mathcal{X}$: the centered configurat'n of D in Lemma 6.3.

SVD of X : $X=V\Sigma_r U'$, where (r: rank of X).

$\Sigma_r=diag(\sigma_1,\cdots,\sigma_r)$, $r\times r$, $\sigma_1\geq\cdots\geq\sigma_r>0$.

$V=[v_1,\cdots,v_r]$, $r\times r$, orthogonal.

U, $n\times r$, w/ orthonormal colums.

For $d\leq r$, $V_d := [v_1,\cdots,v_d]$.

$Y := V_d'X$.

Then Y is a sol'n to the minimizat'n problem stated before Lemma 6.4 w/

$$\eta(\mathcal{Y})=\sum_{i=d+1}^r\sigma_i^2.$$

Rmk:

① Although the motivat'ns of PCA and CMDS are different, when the input similarity matrix of CMDS is an Euclidean distance matrix, they are essentially identical.

② CMDS can be restated in terms of weight matrices:

The input similarity matrix is the weight matrix of n objects, say W.

The goal of CMDS is, under a given d, try to find $\mathcal{Y}\subseteq\mathbb{R}^d$, $\mathcal{Y}=\{y_1,\cdots,y_n\}$ s.t.

$\|W-W(\mathcal{Y})\|_F^2$ is minimized. (this is exactly the lost function)

6.4 CMDS Algorithm.

Input: a similarity matrix D.

Step 1 Create centering Gram matrix.

$G := \frac{-1}{2}H(D_\circ^2)H$, where $D_\circ^2$: pwise square of D. (called the Hadamard square of D)

Step 2 Make spectral decomp. of G.

$r := rank(G)$.

$G=U\Lambda U'$, spec. decomp. of G, where

$U=[u_1,\cdots,u_r]$, $\Lambda=diag(\lambda_1,\cdots,\lambda_r)$, $\lambda_1\geq\cdots\geq\lambda_r>0$.

Step 3 Find configurat'n.

$U_d := [u_1,\cdots,u_d]$

$\Sigma_d := diag(\sqrt{\lambda_1},\cdots,\sqrt{\lambda_d})$

Then, the configurat'n is $Y=\Sigma_d U_d'$.

Def

$A=[a_{ij}]$, $B=[b_{ij}]$, matrices of the same dim.

The **Hadamard product** of A and B, denoted $\boxed{A\boxtimes B}$, is $[a_{ij}b_{ij}]$. (also called **pwise product**)

Chap 7 Random Projection. New~ Week 1

Problems of PCA : < 7.1 Intro.

PCA tries to minimize "global distortion" but may not preserve local separation of the data.

The following concept deals w/ local separat'n.

7.1.1 Lipschitz Embedding.

Def $f:\mathcal{X}\subseteq\mathbb{R}^D\to\mathbb{R}^k$ is called a **Lipschitz embedding** or **Lipschitz mapping** if $\exists$ const. $A,B>0$

s.t. $A\|u-v\|^2\leq\|f(u)-f(v)\|^2\leq B\|u-v\|^2$, $\forall\, u,v\in\mathcal{X}$.

Rmk:
For general applicatn, we may use norms other than Euclidean norms.

7.1.2 J-L Embeddings.

Def $\mathcal{X} = \{x_1, \cdots, x_n\} \subseteq \mathbb{R}^D$.
$\mathcal{R} = \{r_1, \cdots, r_K\} \subseteq \mathbb{R}^D$, K random vectors.
$X := [x_1' \cdots x_n']$, $D \times n$.
$R := [r_1' \cdots r_K']$, $D \times K$.
$Y := R'X = [y_1' \cdots y_n']$, $K \times n$
The map $f: \mathcal{X} \to \mathcal{Y}$, $x_i \mapsto R' x_i$, is called a $\boxed{\text{random projection}}$.

Rmk:
① Random proj. proves a very useful tool in the construction of Lipschitz mappings.
② The feasibility of rand. proj. to construct Lip. map is due to a result by Johnson and Lindenstrauss. Thus, a Lip. map constructed as so is often called $\boxed{\text{JL-embedding}}$.
③ It is a kind of Monte-Carlo method.

7.2 Random Proj. Algo.

7.2.1 Rand. Matrix and Rand. Proj.

Notation: r: r.v.
① $\boxed{r \sim N(0,1)}$ if r: standard normal.
② $\boxed{r \sim U(\{-1,1\})}$ if r: uniform on $\{-1,1\}$.
③ $\boxed{r \sim U(\{\pm\sqrt{3}, 0\})}$ if $r := \begin{cases} \sqrt{3} & \text{w/ prob. } 1/6 \\ 0 & \text{"} \quad \text{"} \quad 2/3 \\ -\sqrt{3} & \text{"} \quad \text{"} \quad 1/6 \end{cases}$
We shall call r.v. distributed as above of $\boxed{\text{Type-1}}$, $\boxed{\text{Type-2}}$, and $\boxed{\text{Type 3}}$, resp.

Rmk:
All of the three above have zero mean and unit variance.

Def
A $\boxed{\text{random matrix}}$ is a matrix whose entries are iid r.v. Thus we can talk about random matrix of Type-1, Type-2, and Type-3.

Def
$R = [r_{ij}]$, a $K \times D$ random matrix of a certain type.
A $\boxed{\text{normalized random projection}}$ is a map $\mathbb{R}^D \to \mathbb{R}^K$ w/ $u \mapsto \frac{1}{\sqrt{K}} Ru$.

7.2.2 Rand. Proj. Algo.

2 Steps:
(1) Random matrix creation.
(2) Matrix multiplication.

7.3 Justification.

7.3.1 Johnson and Lindenstrauss Lemma.

$\boxed{\text{Lemma 7.1 (JL)}}$.
The statement is logically weird. !!!

$\boxed{\text{Lemma 7.2}}$
$R = [r_{ij}]$: $K \times D$ rand. matrix whose entries are iid w/ zero mean and unit var.
$f: \mathbb{R}^D \to \mathbb{R}^K$, $a \mapsto \frac{1}{\sqrt{K}} Ra$.
Then (a: fixed) (1) $f(a)$: rand. vector in $\mathbb{R}^K$.
(2) all components of $f(a)$ are iid w/ 0 mean and var. $\frac{1}{K}\|a\|^2$.
Thus, $E(\|f(a)\|^2) = \|a\|^2$.

$\boxed{\text{Cor 7.1}}$
f: as above.
$a \in \mathbb{R}^D$: unit.
Then $\sqrt{K} f(a)$: K-dim. rand. vector w/ iid entries w/ zero mean and unit var.
i.e. For $K \times D$ rand. matrix R w/ iid entries w/ zero mean and unit var, and unit vector a,
Ra is a rand. vector whose entries are iid w/ zero mean and unit var.

7.3.2 Rand. Proj. based on Gaussian Dist.

$\boxed{\text{Thm 7.1 (Dasgupta and Gupta)}}$ ⊛
$0 < \varepsilon < 1$, n: positive integer.
K: positive integer such that $K \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \ln n$.
Then, $\forall\, \mathcal{X} \subseteq \mathbb{R}^D$, $|\mathcal{X}| = n$, $\exists$ linear $f: \mathbb{R}^D \to \mathbb{R}^K$
s.t. $(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2$, $\forall\, u, v \in \mathcal{X}$.

Def

An algorithm has $\boxed{\text{randomized polynomial time}}$ if

(1) it runs in poly. time in input size and

(2) NO $\to$ NO, and

YES $\to$ YES w/ positive (const.) prob.

$\boxed{\text{Lemma 7.3}}$ ($k \le D$)

R: $k \times D$ rand. matrix of Gaussian type. (Type-1).

$a \in \mathbb{R}^D$: unit vector.

$y := Ra$, $\beta > 1$.

Then $\Pr[\|y\|^2 \le k/\beta] < \exp\left(\frac{k}{2}(1 - \frac{1}{\beta} - \ln\beta)\right)$, and

$\Pr[\|y\|^2 \ge k\beta] < \exp\left(\frac{k}{2}(1 - \beta + \ln\beta)\right)$.

Rmk: (*) (Cor 7.2)

Using Lemma 7.3, we can prove:

Under conditions of Thm 7.1, $f(a) := \frac{1}{\sqrt{k}} Ra$, then for $u \ne v$ in $\mathcal{X}$,

$$\Pr\left(\frac{\|f(u)-f(v)\|^2}{\|u-v\|^2} \notin [1-\varepsilon, 1+\varepsilon]\right) < \frac{2}{n^2}.$$

Denote the event $\left(\frac{\|f(u)-f(v)\|^2}{\|u-v\|^2} \notin [1-\varepsilon, 1+\varepsilon]\right)$ by $A_{u,v}$.

Then

$$P\left((1-\varepsilon)\|u-v\|^2 \le \|f(u)-f(v)\|^2 \le (1+\varepsilon)\|u-v\|^2, \forall u,v\right)$$

$$= P\left(\bigcap_{u \ne v} A_{u,v}^c\right) = 1 - P\left(\bigcup_{u\ne v} A_{u,v}\right) > 1 - \frac{n(n-1)}{2}\cdot\frac{2}{n^2}$$

Thus, Thm 7.1 is proved and we $= 1/n > 0$.

also have:

Thm 7.1 (Continued.)

The mapping f in Thm 7.1 can be found in randomized polynomial time.

7.3.3 Rand. Proj. based on Type 2 and Type 3.

Thm 7.2 (Achlioptas).

Conditions as in Thm 7.1.

$\mathcal{X} \subseteq \mathbb{R}^D$, $|\mathcal{X}| = n$, $X = [x_1' \cdots x_n']$

R: $k \times D$ rand matrix of Type 2 or Type 3.

$f: \mathbb{R}^D \to \mathbb{R}^k$, $u \mapsto \frac{1}{\sqrt{k}} Ru$.

Then f satisfies the following Lipschitz condit'n:

$$(1-\varepsilon)\|u-v\|^2 \le \|f(u)-f(v)\|^2 \le (1+\varepsilon)\|u-v\|^2, \forall u,v \in \mathcal{X},$$

w/ prob. at least $1 - 1/n^\beta$, where $\beta = \frac{k(\varepsilon^2/4 - \varepsilon^3/6)}{\ln n} - 2 > 0$.

<Pf>

Use conditions to conclude:

$$\Pr\left(\frac{\|f(u)-f(v)\|^2}{\|u-v\|^2} \notin [1-\varepsilon, 1+\varepsilon]\right) < \frac{2}{n^{2+\beta}}. \quad (\#)$$

Part III Nonlinear Dimensionality Reduction.

Chap 8 Isomaps.

8.1 Isomap Embeddings.

8.1.1 Description of Isomaps.

Motivation: use geodesic metric instead of Euclidean metric.

Method:

Measure dissimilarity using geodesic metric, which is approximated by graph distance.

Def

A map is called an isometric map if it preserves distances.

Rmk:

① "Isomap" is an abbreviation of isometric mapping.

② This method was introduced by de Silva, etal.

Mathematical description:

Assume $\mathcal{X} = \{x_1, \cdots, x_n\} \subseteq \mathbb{R}^D$ lies on a d-dim. Riemannian mfd $M \subseteq \mathbb{R}^D$, w/ $d \ll D$.

$d_M :=$ geodesic metric on M.

Then $\exists$ chart $f: M \to \mathbb{R}^d$ s.t.

$$d_2(f(x), f(z)) = d_M(x,z), \forall x,z \in M.$$

In particular, geodesic distances on $\mathcal{X}$ is preserved by f.

8.1.2 Geodesic metric on discrete set.

Motivation:

Though geodesic metric is infeasible due to unknown formula of M, we can use graph distance to app. it.

Def

$\mathcal{X} = \{x_1, \cdots, x_n\}$: a data set in $\mathbb{R}^D$.

$G = [\mathcal{X}, E]$, a graph indicating the nbd system on $\mathcal{X}$

Define $d_G: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called the graph distance, by (i) if $(x,y) \in E$, then $d_G(x,y) := d_2(x,y)$.

(ii) if $(x,y) \notin E$, for a path $\gamma = (x_0, x_1, \cdots, x_{s+1})$ w/ $x_0 = x$, $x_{s+1} = y$, the path distance

$d_\gamma(x,y) := d_2(x_0, x_1) + \cdots + d_2(x_s, x_{s+1})$.

$d_G(x,y) := \min\{d_\gamma(x,y) \mid \gamma : \text{path from } x \text{ to } y\}$.

Rmk:
When the data set is dense enough on M, d_G app. d_M under certain conditions.

8.1.3 Isomap kernel and its Const. Shift.

$D_G := [d_G(x_i, x_j)]$, where $\mathcal{X} = \{x_1, \cdots, x_n\}$.

Note that d_G is a true metric on $\mathcal{X}$.

case1 $\mathcal{X}$ is dense enough on M so that D_G app.

$D_M = [d_M(x_i, x_j)]$ very well.

Then, for a preassigned target dimension k, we may apply CMDS on D_G to find $\mathcal{Y} = \{y_1, \cdots, y_n\} \subseteq \mathbb{R}^k$ w/ $[d_2(y_i, y_j)] \approx D_G \approx D_M$.

More precisely,

$S := D_G \boxtimes D_G = [d_G^2(x_i, x_j)]$.

By Thm 6.1, $G^c = -\frac{1}{2} S^c = -\frac{1}{2} HSH$ is the centered Gram matrix.

Applying PCA, we can obtain $\mathcal{Y}$. (depending on the preassigned k)

G^c is called the Isomap kernel.

case2 $\mathcal{X}$ is not dense enough on M.

Issue: D_G not app. D_M well.

$\Rightarrow$ G^c may not be positive semi-definite.

$\Rightarrow$ CMDS not applicable.

Simple Sol'n: add a $\tau > 0$ as a compensation.

$$d_\tau(x_i, x_j) := \begin{cases} d_G(x_i, x_j) + \tau, & i \neq j. \\ 0, & i = j. \end{cases} \quad \left(\begin{array}{c}\text{constant-shift} \\ \text{technique}\end{array}\right).$$

Thus, $S_\tau := [d_\tau^2(x_i, x_j)]$ and the centered Gram is $G_\tau^c := -\frac{1}{2} S_\tau^c$.

G_τ^c is called a constant-shifted Isomap kernel.

claim: $\exists\, \tau > 0$ s.t. G_τ^c psd.

Thm 8.1 (Cailliez).

The minimal $\tau = \tau^*$ making G_τ^c psd is the largest eigenvalue of

$$B = \begin{pmatrix} 0 & -S^c \\ -I & 2D^c \end{pmatrix}.$$

8.2 Isomap Algorithm.

8.2.1 Algorithm Description

Step1 Neighborhood Definition.

$\mathcal{X} = \{x_1, \cdots, x_n\}$: a pt set in $\mathbb{R}^d$.

Use either k-neighborhood or ε-neighborhood to form a neighborhood system on $\mathcal{X}$.

The method is called k-Isomap or ε-Isomap resp.

Step2 Graph distance computation.

Denote the graph created via the nbd system by $G = [\mathcal{X}, E]$.

case1 ε-nbd.

Compute the graph distance matrix.

case2 k-nbd.

Since the graph is not simple, to ensure the matrix to be symmetric, (1) either define

$d_G(x_i, x_j) = \min\{d_G(i,j), d_G(j,i)\}$, or

(2) modify G to be undirected.

Step3 DR kernel construction.

$S_G := [d_G^2(i,j)]$.

$G^c := -\frac{1}{2} H S_G H = -\frac{1}{2} S_G^c$.

case1 G^c: psd, done.

case2 G^c: not psd. Use $G_{\tau^*}^c$ instead, where

τ^* = maximum eigenvalue of $\begin{pmatrix} 0 & -S_G^c \\ -I & 2D_G^c \end{pmatrix}$.

Step4 CMDS.

8.3 Dijkstra's algorithm.

For dense graph, use Floyd's algorithm.

For sparse " , " Dijkstra's " .

See P160 - P169.

Chap 9 Maximum Variance Unfolding.

Motivation:

Preserving both local distances and angles.

AKA:

(1) $\boxed{\text{MVU}}$ ($\boxed{\text{maximum variance unfolding}}$)

(2) $\boxed{\text{SDE}}$ ($\boxed{\text{semidefinite embedding}}$)

(reason: involving $\boxed{\text{semidefinite programming}}$ ($\boxed{\text{SDP}}$)).

Efficiency Modification:

$\boxed{\text{landmark MVU}}$ ($\boxed{\text{LMVU}}$).

9.1 MVU Method Description.

9.1.1 Description of MVU Method.

$\mathcal{X} \subseteq \mathbb{R}^D$, lies on a convex d-mfd M.

h: an isometric chart on (the whole) M.

$\mathcal{Y} := h(\mathcal{X}) \subseteq \mathbb{R}^d$, which is the target.

Suppose we have built a nbd system on $\mathcal{X}$.

For $x_i \in \mathcal{X}$, let P_i be the nbd of x_i.

Let $Q_i := h(P_i)$. $y_i := h(x_i)$. $N(i) := \{j \mid x_j \in P_i\}$.

Idea: $\|x_j - x_k\| \approx \|y_j - y_k\|$, $\forall\, j,k \in N(i)$.

Thus we want to obtain:

$$\mathcal{Y} = \underset{\mathcal{Y} \subseteq \mathbb{R}^d}{\arg\min} \sum_{i=1}^{n} \sum_{j\sim k} \left(\|x_k - x_j\|^2 - \|y_k - y_j\|^2\right)^2, \text{ where}$$

$j \sim k$, $\forall\, j,k \in N(i)$, including i.

Issue:

The above optimizatn problem is nonconvex; hard to solve directly.

Sol'n:

Consider "another similar optimizatn problem" solvable by SDP.

Let $Y := (y_1 \cdots y_n)$, where $\mathcal{Y} = \{y_1, \cdots, y_n\}$.

$K := Y^T Y$, the Gram matrix.

WLOG, we may assume $\mathcal{Y}$ is centered.

i.e. $y_1 + \cdots + y_n = 0$. i.e. $\boxed{\sum_{i,j=1}^{n} K_{ij} = 0.}$ ($*_1$) centering constraint.

$S_{kj} := \|y_j - y_k\|^2$.

Then

$$\boxed{S_{kj} = K_{kk} + K_{jj} - 2K_{kj},\ \forall\, j,k \in N(i),\ 1 \le i \le n.} \quad (*_2)$$

local geometry constraint

(we may assume $k<j$ to eliminate redundancy)

Note that ($*_2$) is a constraint on local geometry.

($*_1$) " " convenient constraint.

MVU, as the name suggests, has the following constraint:

$$\boxed{\mathcal{Y} = \underset{\substack{\mathcal{Y} \subseteq \mathbb{R}^d \\ \text{w/}(*_1),(*_2)}}{\arg\max} \sum_{i,j=1}^{n} \|y_i - y_j\|^2 \quad (*_3)}$$

pull $\mathcal{Y}$ as far apart as possible.

We want everything to be stated in terms of K, so we make the following deduction:

By Thm 6.1, $G_Y^c = -\frac{1}{2} S_Y^c = -\frac{1}{2} H S_Y H$.

Note that $K = G_Y^c$ (due to ($*_1$)).

$$\Rightarrow \operatorname{tr}(K) = \operatorname{tr}(G_Y^c) = \frac{-1}{2} \operatorname{tr}(H S_Y H)$$

$$\underset{(H^2=H)}{=} \frac{-1}{2} \operatorname{tr}(S_Y H) = \frac{-1}{2} \operatorname{tr}\left(S_Y \left(I - \frac{1}{n} E\right)\right)$$

$$\underset{(\operatorname{tr}(S_Y)=0)}{=} \frac{1}{2n} \operatorname{tr}(S_Y E) = \frac{1}{2n} \sum_{i,j=1}^{n} \|y_i - y_j\|^2.$$

Thus, ($*_3$) can be rephrased as

$$\mathcal{Y} = \underset{\substack{\mathcal{Y} \subseteq \mathbb{R}^d \\ \text{w/}(*_1)(*_2)}}{\arg\max} \operatorname{tr}(K).$$

In summary, we obtain the kernel K via

$$\boxed{\begin{aligned} &\text{maximize } \operatorname{tr}(K),\ K: n\times n \text{ psd (i.e. } K \succeq 0). \\ &\text{s.t. } \sum_{i,j=1}^{n} K_{ij} = 0, \\ &K_{kk} + K_{jj} - 2K_{kj} = S_{kj},\ k,j \in N(i),\ 1 \le i \le n. \end{aligned}}$$

Rmk:

① The input can either be (i) $\mathcal{X}$ or (ii) D_X (equivalently, S_X).

② ($*_2$) requires the neighbor relations to be complete. i.e. for each i, x_j and x_k are nb, $\forall\, j,k \in N(i)$. We can achieve this by locally completing the graph.

9.1.2 MVU Algorithm.

Assume $\mathcal{X} \subseteq \mathbb{R}^D$ or a local dist. matrix D is given.

$\boxed{\text{Step 1.}}$ Nbd definition. (~~若~~ when only $\mathcal{X}$ is given).
k-nbd is commonly used. $A :=$ adjacency matrix.

$\boxed{\text{Step 2}}$. Compute S or G^c.

$\boxed{\text{Step 3.}}$ Constraints Generating.

If $A(k,j)=1$, set the constraint

$$K_{kk}+K_{jj}-2K_{kj} = S_{kj} \ (\text{or } G_{kk}+G_{jj}-2G_{kj}).$$

Also, we need to add in the centering constraint

$$\sum_{k,j=1}^{n} K_{kj} = 0.$$

$\boxed{\text{Step 4}}$. Build MVU kernel using SDP.

Apply SDP to obtain K.

$\boxed{\text{Step 5.}}$ Spectral Decomp. of K.

Choose a d and find out the d-leading eigenvec. of K.

9.2 SDP.

Def

For $n \times n$ sym. matrices A and B, define their $\boxed{\text{inner product}}$ $\boxed{A \cdot B}$ by $A \cdot B = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}B_{ij}$.

Given C, A_i, $i=1,\cdots,m$, symmetric $n \times n$.

$b_i \in \mathbb{R}$, $i=1,\cdots,m$.

The following optimization problem is called a $\boxed{\text{semidefinite programming.}}$ ($\boxed{\text{SDP}}$).

$$\boxed{\begin{aligned} \text{minimize} \quad & C \cdot X \\ \text{s.t.} \quad & A_i \cdot X = b_i, \ i=1,\cdots,m. \\ & X \succeq 0. \end{aligned}}$$

Def

The set of $n \times n$ sym. matrices satisfying the constraints is called the $\boxed{\text{feasible set}}$.

$\boxed{\text{Prop}}$

An SDP has a sol'n if the feasible set $\neq \emptyset$.

Rmk:

MVU can be solved by SDP.