

Dimensionality Reduction

Part 1: MDS

Ng Yen Kaow

Dimensionality Reduction

□ Linear methods

- **PCA** (Principal Component Analysis)
- **cMDS** (Classical Multidimensional Scaling)

□ Non-linear methods

- **metric-MDS**
- **Kernel classical MDS**
- **Isomap**
- **LLE** (Locally Linear Embedding)
- **t-SNE** (t-distributed Stochastic Neighbor Embedding)
- **UMAP** (Uniform Manifold Approximation and Projection)

Multidimensional Scaling (MDS)

- Classical MDS (cMDS)
 - Reconstruct coordinates from Euclidean distance matrix
- Metric MDS (mMDS)
 - Redefined cMDS problem with loss function defined on any metric
- Non-metric MDS (nMDS)
 - When only an ordering on the distances is known
- Kernel classical MDS

Classical MDS (cMDS)

- Reconstruct a set of points given their Euclidean distances
- Given $n \times n$ distance matrix $D = (d_{ij})$, reconstruct coordinates $x_1, \dots, x_n \in \mathbb{R}^m$ with $\|x_i - x_j\| = d_{ij}$
 - The solution $X = [x_1 \ \dots \ x_n]^T \in \mathbb{R}^{n \times m}$ is not unique due to infinitely many translations, rotations, and reflections
 - A centered solution $X = (x_{ij})$ (i.e. $\forall k, 1 \leq k \leq m, \sum_i x_{ik} = 0$) can be found using cMDS
 - Note that solution is still not unique

cMDS idea

- Given $D = (d_{ij})$, first note that Euclidean distance d_{ij} is related to $X = (x_{ij})$ through
$$(d_{ij})^2 = (x_i - x_j)^T (x_i - x_j) = x_i^T x_i + x_j^T x_j - 2x_i^T x_j$$
- On the other hand, for X where $\forall k, 1 \leq k \leq m, \sum_i x_{ik} = 0$, we can show that

$$A = -2XX^T$$

where $A = (d_{ij}^2)$

- Then it suffices that we compute $-A/2$ to obtain XX^T
- Finally, since XX^T can be decomposed to recover X

cMDS algorithm

- **Step 1.** Compute matrix CAC

Given $D = (d_{ij})$, computed CAC

where

$$A = \left(-\frac{1}{2} d_{ij}^2 \right)$$

$$C = I - \frac{1}{n} \mathbf{1}^T \mathbf{1}$$

- CAC simultaneously centers the rows and columns of the squared distance matrix A (double centering)
- It can be shown that $CAC = XX^T$ for centered X (proof in later slides)
 - $\Rightarrow CAC$ is positive semi-definite (proof later)
 - $\Rightarrow CAC$ decompose to non-negative values

cMDS algorithm

- **Step 2.** Decompose CAC into orthonormal basis

Method 1: Eigendecompose CAC into $Q\Lambda Q^T$

- Then, X can be computed as $Q\Lambda^{1/2}$

- $Q\Lambda Q^T = Q\Lambda^{1/2}\Lambda^{1/2}Q^T = Q\Lambda^{1/2}(Q\Lambda^{1/2})^T = XX^T$

Method 2: Decompose CAC directly into XX^T using Cholesky factorization

- Only works if CAC is positive definite
- CAC is (positive semi-definite and) positive definite iff all x_i are linearly independent
 - Cholesky in numpy/scipy will not execute unless the input is positive definite
 - Use one that works (e.g. pyre) or write your own

cMDS algorithm

- **Step 3.** Choose from the decomposed basis

Both methods face the problem that the output matrix is not of dimension $n \times m$

- Eigendecomposition $Q \in \mathbb{R}^{n \times n}$
- Cholesky factorization $L \in \mathbb{R}^{n \times n}$

- If $n < m$ (fewer datapoints than features)
 - No problem in embedding the extra features since n points can fit on an $(n - 1)$ -D plane
 - Naturally suited for dimensionality reduction purpose if use all $n - 1$ eigenvectors
 - If need fewer than $(n - 1)$ -D space, see later slides

cMDS algorithm

- **Step 3.** Choose from the decomposed basis

Both methods face the problem that the output matrix is not of dimension $n \times m$

- Eigendecomposition $Q \in \mathbb{R}^{n \times n}$
- Cholesky factorization $L \in \mathbb{R}^{n \times n}$

- If $n > m$ (more datapoints than features)

Problem 1

- CAC is not positive definite since there are insufficient features for linear independence
 - Bad news for Cholesky factorization

cMDS algorithm

- **Step 3.** Choose from the decomposed basis

Both methods face the problem that the output matrix is not of dimension $n \times m$

- Eigendecomposition $Q \in \mathbb{R}^{n \times n}$
- Cholesky factorization $L \in \mathbb{R}^{n \times n}$

- If $n > m$ (more datapoints than features)

Problem 2

- Need to deduce m (or fewer) vectors
 - In an ideal eigendecomposition there will be $\text{rank}(XX^T) (\leq m)$ non-negative eigenvalues and $n - \text{rank}(XX^T)$ zero eigenvalues
 - But eigendecomposition usually not ideal with zero eigenvalues, often resulting in complex numbers

cMDS algorithm

- **Step 3.** Choose from the decomposed basis
 - Many implementations will output negative eigenvalues, so extra care is needed
 - For eigendecomposition
 - Remove the eigenpairs with small, negative, or complex eigenvalues, forming Q_1 and Λ_1
 - Choose the set of eigenvalues S from Λ_1 such that $\frac{\sum_{\lambda' \in S} \lambda'}{\sum_{\lambda \in \Lambda_1} \lambda}$ is sufficiently large
 - Finally, compute $Q_1 \Lambda_1^{1/2}$ and retain only those in S
 - For Cholesky factorization
 - Choose the vectors with the largest norms

(Proof) $XX^T = CAC$ for centered X

- Will expand XX^T and CAC and show equivalence
- Given $X \in \mathbb{R}^{m \times n}$, denote XX^T as B , then we can write the Euclidean distance between x_i and x_j as

$$d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij} \quad (1)$$

- If $\forall k, \sum_i x_{ik} = 0$ (X is centered), then

$$\sum_{j=1}^n b_{ij} = \sum_{j=1}^n \sum_{k=1}^m x_{ik} x_{jk} = \sum_{k=1}^m x_{ik} \left(\sum_{j=1}^n x_{jk} \right) = 0$$

Denote $\text{tr}(B) = \sum_{i=1}^n b_{ii}$, $\because \sum_{j=1}^n b_{ij} = 0$, (1) \Rightarrow

$$\left. \begin{aligned} \sum_{i=1}^n d_{ij}^2 &= \sum_{i=1}^n b_{ii} + \sum_{i=1}^n b_{jj} = \text{tr}(B) + nb_{jj} \\ \sum_{j=1}^n d_{ij}^2 &= \sum_{j=1}^n b_{ii} + \sum_{j=1}^n b_{jj} = nb_{ii} + \text{tr}(B) \\ \sum_{i,j=1}^n d_{ij}^2 &= \sum_{i,j=1}^n b_{ii} + \sum_{i,j=1}^n b_{jj} = 2n\text{tr}(B) \end{aligned} \right\} (2)$$

(Proof) $XX^T = CAC$ for centered X

Rewrite (1) as $b_{ij} = \frac{1}{2} (b_{ii} + b_{jj} - d_{ij}^2)$, then (1)+(2)

$$\begin{aligned} b_{ij} &= \frac{1}{2} \left(\frac{1}{n} \left(\sum_{i=1}^n d_{ij}^2 - \text{tr}(B) \right) + \sum_{j=1}^n d_{ij}^2 - \text{tr}(B) \right) - d_{ij}^2 \\ &= \frac{1}{2} \left(\frac{1}{n} \left(\sum_{i=1}^n d_{ij}^2 + \frac{1}{n} \sum_{j=1}^n d_{ij}^2 - \frac{1}{n} \sum_{i,j=1}^n d_{ij}^2 \right) - d_{ij}^2 \right) \end{aligned}$$

Done, but for notation simplicity let $a_{ij} = -\frac{1}{2} d_{ij}^2$, then

$$b_{ij} = -\frac{1}{n} \sum_{i=1}^n a_{ij} - \frac{1}{n} \sum_{j=1}^n a_{ij} + \frac{1}{n^2} \sum_{i,j=1}^n a_{ij} + a_{ij}$$

Further make things easy to see with

$$a_{i\blacksquare} = \frac{1}{n} \sum_{i=1}^n a_{ij}, \quad a_{\blacksquare j} = \frac{1}{n} \sum_{j=1}^n a_{ij}, \quad a_{\blacksquare\blacksquare} = \frac{1}{n^2} \sum_{i,j=1}^n a_{ij}$$

$$\Rightarrow b_{ij} = a_{ij} - a_{i\blacksquare} - a_{\blacksquare j} + a_{\blacksquare\blacksquare}$$

(Proof) $XX^T = CAC$ for centered X

- Now expand CAC into terms consisting of a_{ij}

Given $A = (a_{ij})$, observe that

$$\left. \begin{aligned} [1 \ 1 \ \dots \ 1]A &= n(a_{i\blacksquare}) \\ A[1 \ 1 \ \dots \ 1] &= n(a_{\blacksquare j}) \\ [1 \ 1 \ \dots \ 1]A[1 \ 1 \ \dots \ 1] &= n^2(a_{\blacksquare\blacksquare}) \end{aligned} \right\} (3)$$

On the other hand,

$$\begin{aligned} CAC &= \left(I - \frac{1}{n}J\right) A \left(I - \frac{1}{n}J\right) \\ &= A - \frac{1}{n}JA - \frac{1}{n}AJ + \frac{1}{n^2}JAJ \end{aligned} \quad (4)$$

Finally, (3)+(4) gives

$$(CAC)_{ij} = a_{ij} - a_{i\blacksquare} - a_{\blacksquare j} + a_{\blacksquare\blacksquare} = b_{ij}$$

(Proof) CAC is PSD

- Follows immediately from the fact that $CAC = XX^T$, an inner product
 - An inner product $B = XX^T$ of any matrix X (centered or not) is called a Gram matrix, or Gramian
 - Gramians are known to be positive semi-definite (see proof in other slides)

Choose between XX^T or $X^T X$

- XX^T and $X^T X$ have equivalent eigenpairs
- For any matrices A and B , AB and BA have the same non-zero eigenvalues
 - Let $\lambda \neq 0$ be a eigenvalue for AB with eigenvector v
 - Then $ABv = \lambda v \Rightarrow BABv = \lambda Bv$
 $\Rightarrow (BA)(Bv) = \lambda(Bv)$
 $\Rightarrow \lambda$ is a eigenvalue of BA with eigenvector (Bv)
- One can choose the smaller between XX^T and $X^T X$ to perform MDS
- This also means that MDS and PCA computes the same eigendecomposition

Generalization of cMDS

- Can classical MDS be generalized to other metrics?

- The relation between coordinates X and distances A through $XX^T = CAC$ is central to cMDS

- In showing $XX^T = CAC$, we used the relation

$$(d_{ij})^2 = (x_i - x_j)^T (x_i - x_j)$$

(or, in terms of the Gramian, $d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}$)

- This equation describes the Pythagorean principle for Euclidean distance

- $XX^T = CAC$ does not hold for other metrics

Metric MDS (mMDS)

- Given distance matrix $(\delta_{ij})_{n \times n}$ and weights $(w_{ij})_{n \times n}$, find $X = [x_1 \quad \dots \quad x_n]^T$ where $x_i \in \mathbb{R}^r$, which minimizes

$$\text{stress}(X) = \sum_{i,j,i < j} w_{ij} (d(x_i, x_j) - \delta_{ij})^2$$

where $d(x_i, x_j)$ denotes the distance between x_i and x_j

- The weights w_{ij} allow removing entries where δ_{ij} is not available

SMACOF Algorithm for mMDS

□ Minimize $\text{stress}(X)$ through majorization

□
$$\text{stress}(X) = \sum_{i,j,i < j} w_{ij} (d(x_i, x_j) - \delta_{ij})^2$$
$$= \sum w_{ij} d^2(x_i, x_j) + \sum w_{ij} \delta_{ij}^2 - 2 \sum w_{ij} \delta_{ij} d(x_i, x_j)$$

Since

- $\sum w_{ij} \delta_{ij}^2$ is constant, C
- $\sum w_{ij} d^2(x_i, x_j)$ is quadratic, $\text{tr}(X' V X)$
- $\sum w_{ij} \delta_{ij} d(x_i, x_j) = \text{tr}(X' B(X) X) \geq \text{tr}(X') B(Z) Z$
where $B(Z) = (b_{ij})$ for

$$b_{ij} = \begin{cases} -\frac{w_{ij} \delta_{ij}}{d(x_i, x_j)} & \text{if } d(x_i, x_j) \neq 0 \text{ and } i \neq j \\ 0 & \text{if } d(x_i, x_j) = 0 \text{ and } i \neq j \end{cases}$$
$$b_{ii} = -\sum_{j=1, j \neq i}^n b_{ij}$$

SMACOF Algorithm for mMDS

- $\text{stress}(X) = C + \text{tr}(X'VX) - 2\text{tr}(X'B(X)X)$
which is bounded above by
 $C + \text{tr}(X'VX) - 2\text{tr}(X'B(Z)Z) = \tau(X, Z)$
- Majorization iteratively updates X^k at the k^{th} iteration to $\min_X \tau(X, X^{k-1})$
 - $\text{stress}(X)$ will decrease monotonically
 - Stops iteration when $\text{stress}(X^k) - \text{stress}(X^{k-1})$ is below a given threshold
- Proofs for the majorization method requires too much details to provide here

Sammon mapping

- A special case of $\text{stress}(X)$ where weights are inversely proportional to distance δ_{ij}
 - Emphasize accuracy on small δ_{ij} distances

- Given distance matrix $(\delta_{ij})_{n \times n}$, find $X = [x_1 \ \dots \ x_n]^T$ where $x_i \in \mathbb{R}^r$, which minimizes

$$\text{stress}(X) = \frac{1}{\sum_{i,j,i < j} \delta_{ij}} \sum_{i,j,i < j} \frac{(d(x_i, x_j) - \delta_{ij})^2}{\delta_{ij}}$$

where $d(x_i, x_j)$ denotes the distance between x_i and x_j

- The simpler $\text{stress}(X)$ allows a gradient descent optimization

nMDS vs cMDS

□ Similarity vs dissimilarity

- cMDS attempts to recover XX^T , a measure of the similarity between x_i and x_j
- nMDS attempts to recover distances $d(x_i, x_j)$, a measure of the dissimilarity between x_i and x_j

□ Linear vs non-linear

- cMDS attempts to recover XX^T , a linear kernel
- nMDS, for instance Sammon mapping, can be considered as recovering a non-linear distance measure with an inverse $(1/\delta)$ factor

□ Closed-form vs iterative method

- cMDS is solved through a closed-form solution
- nMDS can only be approximated iteratively using gradient descent or majorization

Kernel Classical MDS