Nonlinear Dimensionality Reduction

Xian-Hong Wang (王先弘)

Department of Mathematics, National Central University
Jhongli District, Taoyuan City 32001, Taiwan

Revised on March 2, 2023

Outline

- Linear dimension reduction
  - Principal component analysis (PCA)
  - Linear discriminant analysis (LDA)
- Non-linear dimension reduction (Manifold Learning)
  - Mutidimensional Scaling (MDA)
  - Isomap
  - Diffusion Maps
  - Laplacian Eigenmaps (LE)
  - Locally linear embedding (LLE)
- Kernel method (KPCA)

Principal component analysis (PCA)

The basic idea of PCA

Given a centered dataset

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_n \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

PCA want to find a direction $\boldsymbol{q}$ which maximizes the spread(variance).
Representing $\boldsymbol{X}$ in terms of its thin-SVD

$$\boldsymbol{X} = \boldsymbol{U}_r \boldsymbol{D}_r \boldsymbol{V}_r^\top,$$

where $\boldsymbol{U}_r \in \mathbb{R}^{n \times r}$, $\boldsymbol{V}_r \in \mathbb{R}^{p \times r}$ are matrices with orthonormal columns,

$$\boldsymbol{U}_r = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_r \end{bmatrix}, \boldsymbol{V}_r = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \dots & \boldsymbol{v}_r \end{bmatrix},$$

and $\boldsymbol{D}_r \in \mathbb{R}^{r \times r}$ is a diagonal matrix with the nonzero singular values as diagonal entries.

Spread

Let $\boldsymbol{q} \in \mathbb{R}^m$ be a unit vector.The projections of the data vectors along the direction $\boldsymbol{q}$ are given by components of the row vector

$$\boldsymbol{Y} = \boldsymbol{q}^\top \boldsymbol{X} = \begin{bmatrix} \boldsymbol{q}^\top \boldsymbol{x}_1 & \boldsymbol{q}^\top \boldsymbol{x}_2 & \cdots & \boldsymbol{q}^\top \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} \in \mathbb{R}^{1\times n}.$$

Define the *spread* of the components as

$$spread(\boldsymbol{Y}) = \Big(\sum_{j=1}^{p} {y_j}^2\Big)^{1/2},$$

where

$$\sum_{j=1}^{N} y_j^2 = \boldsymbol{Y}\boldsymbol{Y}^\top = \boldsymbol{q}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{q} = \|\boldsymbol{X}^\top \boldsymbol{q}\|_2^2.$$

Optimization problem

So the optimization problem is

$$\max\{spread(\boldsymbol{Y})\} = max\{\|\boldsymbol{X}^\top \boldsymbol{q}\| \mid \|\boldsymbol{q}\| = 1\}.$$

Therefore, it follows by the definition of induced matrix norm that

$$\max\{\|\boldsymbol{X}^\top \boldsymbol{q}\| \mid \|\boldsymbol{q}\| = 1\} = \|\boldsymbol{X}^\top\|_2.$$

Furthermore,

$$\max\{spread(\boldsymbol{Y})\} = \sigma_1 = \|\boldsymbol{X}^\top \boldsymbol{u}_1\|_2,$$

where σ_1 is first singular value of $\boldsymbol{X}$, and $\boldsymbol{u}_1$ is first left singular vector corresponding to the $\boldsymbol{u}_1$.
So

$$\boldsymbol{q} = \boldsymbol{u}_1,$$

and the maximum spread equals the largest singular value of the data matrix.

Visualization

Subtract from the data matrix the rank 1 matrix corresponding to the first singular triplet,

$$\widetilde{\boldsymbol{X}} = \boldsymbol{X} - \sigma_1 \boldsymbol{u}_1 (\boldsymbol{v}_1)^\top = \sum_{j=2}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^\top,$$

Define the spread of the components of

$$\widetilde{\boldsymbol{Y}} = q^\top \widetilde{\boldsymbol{X}}.$$

Maximizing the spread with respect to the direction vector q leads to the formula

$$max\{spread(\widetilde{\boldsymbol{Y}})\} = \sigma_2 = \|\widetilde{\boldsymbol{X}}^\top \boldsymbol{u}_2\|,$$

confirming that the second best projection direction is $\boldsymbol{u}_2$.

References

- D. Calvetti and E. Somersalo, *Mathematics of Data Science: A Computational Approach to Clustering and Classification*, SIAM, 2021, pp. 21-32.
- C. Heeyoul and C. Seungjin. Robust kernel Isomap, *ScienceDirect*, 2006, pp. 853-862.

Linear discriminant analysis

LDA

The data considered in this section are assumed to be labled, or annotated, in the sense that each datapoint belongs to one and only one class, and that the class-belonging information is available. More formally, let $\mathscr{D}$ denoted the data set partitioned into k nonintersecting subsets,

$$\mathscr{D} = \mathscr{D}_1 \cup \mathscr{D}_2 \cup \cdots \cup \mathscr{D}_k, \;\; \mathscr{D}_i \cap \mathscr{D}_j = \emptyset \;\; \textit{if } i \neq j,$$

and let

$$I_j = \ell \textit{ equivalent to } x^{(j)} \in \mathscr{D}_\ell, \; 1 \leq j \leq n,$$

define the *annotation vector* $I \in \{1, 2, \ldots, k\}^j$ that contains the labels assigning each data vector to one and only one subset.

LDA

Given a data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, center or noncentered, the components of the data vectors $x^{(j)}$ on a given direction vector $q \in \mathbb{R}^m$, with $\|q\| = 1$, can be collected into a row vector y of length n,

$$y = q^{\mathsf{T}}\boldsymbol{X}.$$

Recalling that the square of the spread of the data in the direction q is the quantity

$$spread(y)^2 = \sum_{j=1}^{n} y_j^2 = yy^{\mathsf{T}},$$

and replacing y by its expression in terms of the data matrix, we obtain

$$spread(y)^2 = q^{\mathsf{T}}\boldsymbol{X}\boldsymbol{X}^{\mathsf{T}}q = q^{\mathsf{T}}\mathsf{S}q.$$

The matrix S,

$$\mathsf{S} = \boldsymbol{X}\boldsymbol{X}^{\mathsf{T}} \in \mathbb{R}^{m\times m},$$

is called the *scatter matrix* of the data $\boldsymbol{X}$.

LDA

Therefore, the direction of maximum (minimum) spread of the data is the vector that maximizes (minimizes) the quadratic expression. The observation motivates us to define a family of scatter matrices to help us optimize the search for projection directions.
Without loss of generality we can reorder the data so that the columns corresponding to points in the same cluster are adjacent, i.e,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \cdots & \boldsymbol{X}_k \end{bmatrix} \in \mathbb{R}^{m \times n},$$

where the columns of $\boldsymbol{X}_\ell$ are the data vectors in cluster $\mathscr{D}_\ell$,

$$\boldsymbol{X}_\ell = \begin{bmatrix} x^{(j_1)} & x^{(j_2)} & \cdots & x^{(j_{n_\ell})} \end{bmatrix} \in \mathbb{R}^{m \times n_\ell},$$

and

$$I_\ell = \begin{bmatrix} j_1 & j_2 & \cdots & j_{n_\ell} \end{bmatrix}.$$

LDA

We have

$$n_1 + n_2 + \cdots + n_k = n.$$

We define the *cluster centroids* as

$$c^{(\ell)} = \frac{1}{n_\ell} \sum_{j \in I_\ell} x^{(j)}, \quad 1 \leq \ell \leq k.$$

The *global centroid* of the data is defined by the formula

$$c = \frac{1}{n} \sum_{j=1}^{n} x^{(j)}.$$

We start by defining the centered data matrices for each cluster,

$$\boldsymbol{X}_{\ell,c} = \begin{bmatrix} x^{(j_1)} - c^{(\ell)} & x^{(j_2)} - c^{(\ell)} & \cdots & x^{(j_{n_\ell})} - c^{(\ell)} \end{bmatrix} \in \mathbb{R}^{m \times n_\ell}, \quad 1 \leq \ell \leq k.$$

LDA

The centered matrices are then collected in the *within-cluster centered data* matrix,

$$\boldsymbol{X}_w = \begin{bmatrix} \boldsymbol{X}_{1,c} & \boldsymbol{X}_{2,c} & \cdots & \boldsymbol{X}_{k,c} \end{bmatrix} \in \mathbb{R}^{m\times n}.$$

Geometrically, $\boldsymbol{X}_w$ provides a representation of the data where each cluster has been separately centered around the origin.
We define the *within-cluster scatter matrix* $\mathsf{S}_w \in \mathbb{R}^{m\times m}$ for the centered data $\boldsymbol{X}_w$ by the formula

$$\mathsf{S}_w = \boldsymbol{X}_w \boldsymbol{X}_w^\top = \sum_{\ell=1}^{k} S_\ell$$

where S_ℓ is the scatter matrix of the ℓth cluster,

$$\mathsf{S}_\ell = \boldsymbol{X}_{\ell,c} \boldsymbol{X}_{\ell,c}^\top = \sum_{j\in I_\ell} (x^{(j)} - c^{(\ell)})(x^{(j)} - c^{(\ell)})^\top \in \mathbb{R}^{m\times m},\ \ 1 \le \ell \le k.$$

LDA

We need measure for how the clusters themselves are scattered. Therefore, we define a centroid approximation of the data matrix $\boldsymbol{X}$ by simply replacing each data vector $x^{(j)}$ by the centroid of its cluster. Denote by $\overline{\boldsymbol{X}}_\ell$ the centroid approximation of $\boldsymbol{X}_\ell$,

$$\overline{\boldsymbol{X}}_\ell = \begin{bmatrix} c^{(\ell)} & c^{(\ell)} & \cdots & c^{(\ell)} \end{bmatrix} \in \mathbb{R}^{m \times n_\ell},$$

which is the rank 1 matrix obtained by replacing each point in the ℓth cluster by the cluster centroid. We collect the centroid approximation matrices for all the clusters into the matrix

$$\overline{\boldsymbol{X}} = \begin{bmatrix} \overline{\boldsymbol{X}}_1 & \overline{\boldsymbol{X}}_2 & \cdots & \overline{\boldsymbol{X}}_k \end{bmatrix} \in \mathbb{R}^{m \times n},$$

whose rank is at most k.

LDA

Next we center the matrix $\overline{X}$ by subtracting the global centroid from each column,

$$\overline{X}_c = \overline{X} - \begin{bmatrix} c & c & \cdots & c \end{bmatrix},$$

and define the *between-cluster scatter matrix* $S_b \in \mathbb{R}^{m \times m}$ by

$$S_b = \overline{X}_c \overline{X}_c^\top,$$

which can be expressed also in the form

$$\begin{aligned} S_b &= \sum_{\ell=1}^{k} \sum_{j=1}^{n_\ell} (c^{(\ell)} - c)(c^{(\ell)} - c)^\top \\ &= \sum_{\ell=1}^{k} n_\ell (c^{(\ell)} - c)(c^{(\ell)} - c)^\top. \end{aligned}$$

We have now introduced all the tools needed to formally state and pursue the objective of the LDA of finding optimal projection directions along which clusters are maximally separated.

LDA

The goal of this section is to find few projection directions in the data space along which the within-cluster spread is as small as possible and between-cluster spread as large as possible.
The objective function guiding the search for vectors determining the directions with the desired properties is the ratio

$$H(q) = \frac{q^{\mathsf{T}} \mathsf{S}_b q}{q^{\mathsf{T}} \mathsf{S}_w q}.$$

We argue that any nonzero vector q that maximizes this ratio is a desired direction.
For the time being, we assume that the matrix S_w is symmetric positive definite. The case of symmetric positive semidefinite S_w will be addressed later.

LDA

We start by observing that since for ever $\alpha > 0$,

$$H(\alpha q) = H(q),$$

In particular, we can scale q so as to guarantee that

$$q^{\mathsf{T}} \mathsf{S}_w q = 1,$$

so that the objective function becomes

$$H(q) = \frac{q^{\mathsf{T}} \mathsf{S}_b q}{q^{\mathsf{T}} \mathsf{S}_w q} = q^{\mathsf{T}} \mathsf{S}_b q.$$

This observation allows us to restate the problem of finding the optimal direction in the following way.

LDA

Since, by assumption, the matrix S_w is symmetric positive definite, it admits the Cholesky factorization,

$$S_w = K^T K,$$

where K is upper triangular and invertible. Hence we can write the constraint as

$$q^T K^T K q = \|Kq\|^2 = 1,$$

or, by defining the vector w of unit length by

$$Kq = w, \quad or\ q = K^{-1} w,$$

and substituting the expression of q in terms of w, the problem can be restated as

$$\arg\max_w w^T K^{-T} S_b K^{-1} w \quad \text{subject to } \|w\|^2 = 1,$$

where $K^{-T} = (K^{-1})^T$.

LDA

Following the method of Lagrange multipliers, we define a Lagrange function

$$L_\lambda(w) = w^\mathsf{T}\mathsf{K}^{-\mathsf{T}}\mathsf{S}_b\mathsf{K}^{-1}w - \lambda(\|w\|^2 - 1),$$

where $\lambda \in \mathbb{R}$ is a Lagrange multiplier, and look for its critical points, i.e., vectors w such that

$$\nabla_w L_\lambda(w) = 2(\mathsf{K}^{-\mathsf{T}}\mathsf{S}_b\mathsf{K}^{-1}w - \lambda w) = 0.$$

Equivalently, the critical points must satisfy

$$\mathsf{K}^{-\mathsf{T}}\mathsf{S}_b\mathsf{K}^{-1}w = \lambda w,$$

that is, w is an eigenvector of unit length of the matrix $\mathsf{K}^{-\mathsf{T}}\mathsf{S}_b\mathsf{K}^{-1}$. Furthermore, since for such eigenvectors,

$$w^\mathsf{T}\mathsf{K}^{-\mathsf{T}}\mathsf{S}_b\mathsf{K}^{-1}w = \lambda\|w\|^2 = \lambda,$$

the maximizer of the objective function is an eigenvector corresponding to the largest eigenvalue of the matrix, denoted by λ_1.

LDA

Having found the vector w, we may solve for q. Observe that from solution, it follows that, by multiplying both sides by the matrix K^{-1}, we have

$$\mathsf{K}^{-1}\mathsf{K}^{-\mathsf{T}}\mathsf{S}_b\mathsf{K}^{-1}w = \lambda\mathsf{K}^{-1}w,$$

or

$$\mathsf{S}_w^{-1}\mathsf{S}_b q = \lambda q,$$

that is, the vectors q are eigenvectors of the nonsymmetric matrix $\mathsf{S}_w^{-1}\mathsf{S}_b$, which has thus been shown to have all real eigenvalues.
If there are more than two clusters in the data, it may be useful to find more than just one projection direction to represent the data. A natural way is to find the eigenvectors $v^{(\ell)}$, $\ell > 1$, set $w^{(\ell)} = v^{(\ell)}$, and define the projection directions as

$$q^{(\ell)} = \mathsf{K}^{-1}w^{(\ell)}.$$

Observe that while the vectors $v^{(\ell)}$ are mutually orthogonal, the vectors $v^{(\ell)}$ are usually not.

LDA

So far, we have assumed that the matrix S_w is symmetric positive definite, and therefore invertible. If the assumption is not valid, the argument above does not work, as S_w will not admit the Cholesky factorization. On the other hand, being symmetric positive semidefinite, the matrix S_w, could be factored as

$$S_w = S_w^{1/2} S_w^{1/2},$$

but since the factors would not be invertible, the argument does not work. To overcome this problem, we replace S_w by a nearly positive definite approximation by setting

$$S_{w,\epsilon} = S_w + \epsilon I_n,$$

where $\epsilon > 0$ is a small scalar. For any $v \in \mathbb{R}^m$, $v \neq 0$, we have

$$v^\top S_{w,\epsilon} v = v^\top S_w v + \epsilon \|v\|^2 \geq \epsilon \|v\|^2 > 0,$$

so the existence of the Cholesky decomposition is warranted.

LDA

To understand under which conditions S_w may not be invertible, assume that $m < n$ and consider the singular value decomposition of the within-cluster centered data,

$$\boldsymbol{X}_w = UDV^\top.$$

If the rank r of the matrix $\boldsymbol{X}_w$ is less than n, the within-cluster centered data are restricted to an r-dimensional subspace of $\mathbb{R}^n$. On the other hand,

$$S_w = \boldsymbol{X}_w \boldsymbol{X}_w^\top = UDD^\top U^\top = UWU^\top,$$

with

$$W = DD^\top = \begin{bmatrix} d_1^2 & & \\ & \ddots & \\ & & d_m^2 \end{bmatrix}.$$

where d_j denotes the jth singular value of $\boldsymbol{X}_w$

LDA

Since the eigenvalues of S_w are d_j^2, the matrix S_w has only r nonzero eigenvalues, and therefore cannot be invertible, and the argument above falls apart. Effectively, the approximation is tantamount to perturbing the within-cluster centered data off the r-dimensional subspace.
Observe that the perturbed scatter matrix can be decomposed as

$$S_w + \epsilon I_m = UWU^\top + \epsilon UU^\top = UW_\epsilon U^\top,$$

where

$$W_\epsilon = \begin{bmatrix} d_1^2 + \epsilon & & \\ & \ddots & \\ & & d_m^2 + \epsilon \end{bmatrix},$$

and since all its eigenvalue are strictly positive, it is invertible.

References

- D. Calvetti and E. Somersalo, *Mathematics of Data Science: A Computational Approach to Clustering and Classification*. SIAM, 2021, pp. 49-62.

Mutidimentional scaling

The basic ideas of MDS

Let $\boldsymbol{X} \in \mathbb{R}^{D\times n}$ be the high-dimensional data matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix},$$

and $\boldsymbol{Y} \in \mathbb{R}^{d\times n}$ be the dimension reduced data

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_n \end{bmatrix}.$$

MDS seeks to preserve that the dissimilarity (inner product) between all pairs of data points.
Set up the distance squared matrix

$$\boldsymbol{D}^2 = [d_{ij}^2]_{n\times n},$$

where $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.

Centering matrix

Definition : The **centering matrix** of size n is defined as

$$\boldsymbol{C}_n = \boldsymbol{I}_n - \frac{1}{n}\boldsymbol{J}_n \in \mathbb{R}^{n\times n},$$

where $\boldsymbol{I}_n$ is the identity matrix of size n and $\boldsymbol{J}_n$ is an $n \times n$ matrix of all 1's.

Properties:

- The eigenvalues of $\boldsymbol{C}_n$ are 0 of multiplicity 1 and 1 of multiplicity $n-1$.
- $\boldsymbol{C}_n$ is idempotent.

Application: Let $\boldsymbol{X} \in \mathbb{R}^{m\times m}$.

- Each column of the product $\boldsymbol{C}_m\boldsymbol{X}$ has a zero mean.
- Each row of the product $\boldsymbol{X}\boldsymbol{C}_m$ has a zero mean.
- Each column and each row of double centering matrix $\boldsymbol{C}_m\boldsymbol{X}\boldsymbol{C}_m$ has a zero mean.

Construct doubly centered squared distance matrix

Assume we centered the high-dimensional data such that

$$\sum_{i=1}^{n} x_i = \mathbf{0}.$$

The distance squared between two observations is

$$d_{ij}^2 = \|x_i - x_j\|_2^2 = \|x_i\|_2^2 + \|x_j\|_2^2 - 2x_i^\top x_j, \tag{1}$$

then we can write down the following formulas

$$\sum_{i=1}^{n} d_{ij}^2 = Tr(\mathbf{X}^\top \mathbf{X}) + n\|\boldsymbol{x}_j\|_2^2 - 2\sum_{i=1}^{n} x_i^\top x_j = Tr(\mathbf{X}^\top \mathbf{X}) + n\|\boldsymbol{x}_j\|_2^2 \tag{2}$$

$$\sum_{j=1}^{n} d_{ij}^2 = n\|\boldsymbol{x}_i\|_2^2 + Tr(\mathbf{X}^\top \mathbf{X}) - 2x_i^\top \sum_{j=1}^{n} x_j = n\|\boldsymbol{x}_i\|_2^2 + Tr(\mathbf{X}^\top \mathbf{X}) \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2 = n\sum_{i=1}^{n} \|x_i\|_2^2 + nTr(\mathbf{X}^\top \mathbf{X}) = 2nTr(\mathbf{X}^\top \mathbf{X}). \tag{4}$$

Construct doubly centered squared distance matrix (cont'd)

So combining the above three formulas $((2)+(3))/n-(3)/n^2$, we can get

$$\frac{1}{n}(\sum_{i=1}^{n} d_{ij}^2+\sum_{i=1}^{n} d_{ij}^2)-\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2=\|x_i\|_2^2+\|x_j\|_2^2. \tag{5}$$

We rewrite the (1)

$$\begin{aligned} x_i^\top x_j &= \frac{1}{2}(\|x_i\|_2^2+\|x_j\|_2^2-d_{ij}^2) \\ &= \frac{1}{2}(\frac{1}{n}(\sum_{i=1}^{n} d_{ij}^2+\sum_{i=1}^{n} d_{ij}^2)-\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2-d_{ij}^2) \\ &= -\frac{1}{2}(d_{ij}-\frac{1}{n}(\sum_{i=1}^{n} d_{ij}^2+\sum_{i=1}^{n} d_{ij}^2)+\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2). \end{aligned}$$

Construct doubly centered squared distance matrix (cont'd)

We define the doubly centered distance matrix (inner product matrix)

$$\boldsymbol{K}(\boldsymbol{D}^2) = [k_{ij}]_{n\times n}$$

where

$$\begin{aligned} k_{ij} &= x_i^\top x_j \\ &= -\frac{1}{2}(d_{ij} - \frac{1}{n}(\sum_{i=1}^{n} d_{ij}^2 + \sum_{i=1}^{n} d_{ij}^2) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2) \\ &= -\frac{1}{2}(\boldsymbol{C}_n\boldsymbol{D}^2\boldsymbol{C}_n)_{ij}. \end{aligned}$$

Hence,

$$\boldsymbol{K}(\boldsymbol{D}^2) = -\frac{1}{2}\boldsymbol{C}_n\boldsymbol{D}^2\boldsymbol{C}_n.$$

Optimization problem

The optimization is

$$\min_{y_1,\cdots,y_n} \sum_{ij} (y_i^\top y_j - k_{ij})^2 \implies \min_{\boldsymbol{Y}} \|\boldsymbol{Y}^\top \boldsymbol{Y} - \boldsymbol{K}(\boldsymbol{D}^2)\|_F^2.$$

Since $\boldsymbol{K}(\boldsymbol{D}^2)$ is a symmetric matrix,

$$\begin{aligned} \boldsymbol{K}(\boldsymbol{D}^2) &= \boldsymbol{U}\Lambda\boldsymbol{U}^\top \\ &= \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1^\top \\ \vdots \\ \boldsymbol{u}_n^\top \end{bmatrix}, \end{aligned}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, and $\boldsymbol{U}$ is orthonormal matrix.

So $\boldsymbol{Y} = \Lambda_d^{\frac{1}{2}} \boldsymbol{U}_d^\top$, where Λ_d is the diagonal matrix of first d eigenvalues and $\boldsymbol{U}_d$ is the matrix of first d eigenvectors of $\boldsymbol{K}(\boldsymbol{D}^2)$.

Classical MDS algorithm

step 1: Identify k nearest neighbors (or ϵ ball neighborhood) of each input data point and construct a neighborhood graph where edge lengths between points in a neighborhood are set as their Euclidean distances.

step 2: Compute Euclidean distances, d_{ij},that are $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.

step 3: Construct a matrix $\boldsymbol{K}(\boldsymbol{D}^2) = -\frac{1}{2}\boldsymbol{C}\boldsymbol{D}^2\boldsymbol{C}$.

step 4: Compute top d eigenvectors of $\boldsymbol{K}(\boldsymbol{D}^2)$ which leads to the eigenvector matrix $\boldsymbol{V} \in \mathbb{R}^{n\times d}$ and the eigenvalue matrix $\Lambda \in \mathbb{R}^{d\times d}$.

step 5: The coordinates of n points in the d-dimensional Euclidean space are given by the column vectors of $\boldsymbol{Y} = \Lambda^{1/2}\boldsymbol{V}^\top$.

References

- I. Borg and P. J.F. Groenen, *Modern Mutidimensional Scaling* (2ed ed.). Springer.
- K. Q. Weinberger and L. K. Saul (2006), Unsupervised Learning of image manifolds by semidefinite programming, *International Journal of Computer Vision*, 70(1), pp. 77-90. doi: 10.1007/s11263-005-4939-z.
- X. Zhen and Z. Liaangliang, The simplified expression of machine learning and multivariate statistical analysis based on the centering matrix, 2021.

Isomap

The basic ideas of Isomap

Isomap seeks to preserve the geodesic distances between all pairs of data points.
Define

$$\boldsymbol{D}_G^2 = [d_G^2(i,j)]_{n\times n},$$

where $d_G(i,j)$ is the geodesic distance between x_i and x_j.
Classical Isomap assumes d_G has euclidean representation, that is, there exists a set $\{z_i\}_{i=1}^n \subset \mathbb{R}^D$ such that

$$\|z_i - z_j\|_2^2 = d_G^2(i,j).$$

Thus, $\boldsymbol{D}_G^2$ can be seen as distance squared matrix.

Optimization problem

Then the algorithm is following the form of MDS:

$$K(D_G^2) = -\frac{1}{2} C_n D_G^2 C_n,$$

and the optimization problem is

$$\min_{Y} \| Y^\top Y - K(D_G^2) \|_F^2.$$

So

$$Y = \Lambda_d^{\frac{1}{2}} U_d^\top,$$

where Λ_d is the diagonal matrix of first d eigenvalues and U_d is the matrix of first d eigenvectors of $K(D_G^2)$.

Additive constant problem

Let d be a measure of dissimilarity ($d_{ii} = 0; d_{ij} = d_{ji} \geq 0$) on a finite set I with n elements.
We consider the case where d^2 has no euclidean representation, that is, W_{d^2} matrix is not positive semidefinite:

$$W_{d^2} = -\frac{1}{2} C D^2 C,$$

with $\boldsymbol{D}^2$ is matrix with terms d_{ij}^2, $\boldsymbol{C}$ is centering matrix

$$\boldsymbol{C} = \boldsymbol{I} - \frac{1}{n} \boldsymbol{j}\boldsymbol{j}^\top,$$

where $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is identity matrix, $\boldsymbol{j} \in \mathbb{R}^n$ is vector with all coordinates equal to 1.
Observe that

$$\boldsymbol{W}_{d^2} = \boldsymbol{C}\boldsymbol{W}_{d^2} = \boldsymbol{W}_{d^2}\boldsymbol{C}, \quad \text{and that } \boldsymbol{W}_{d^2}\boldsymbol{j} = \boldsymbol{0}.$$

Additive constant problem (cont'd)

We solve the "additive constant problem" which is find the smallest c^* such that the dissimilarity measure d_c defined by:

$$d_c(i,j) = \begin{cases} d_{ij} + c & , \text{if} \quad i \neq j \\ 0 & , \text{if} \quad i = j \end{cases}$$

has euclidean representation for all $c \geq c^*$.
Then define:

$$(\boldsymbol{D}_c^2)_{ij} := (d_{ij} + c)^2 = d_{ij}^2 + 2cd_{ij} + c^2 = (\boldsymbol{D}^2)_{ij} + 2c(\boldsymbol{D})_{ij} + c^2(\boldsymbol{J} - \boldsymbol{I})_{ij}.$$

The matrix $\boldsymbol{W}$ associated to d_c define by:

$$\boldsymbol{W}_{d_c} = -\frac{1}{2}\boldsymbol{C}\boldsymbol{D}_c^2\boldsymbol{C} = \boldsymbol{W}_{d^2} + 2c\boldsymbol{W}_d + \frac{c^2}{2}\boldsymbol{C}, \tag{6}$$

where $\boldsymbol{W}_d = -\frac{1}{2}\boldsymbol{C}\boldsymbol{D}\boldsymbol{C}$ with $\boldsymbol{D}$ is matrix with terms d_{ij}.

Additive constant problem (cont'd)

d_c has an euclidean representation if:

$$\boldsymbol{x}^\top \boldsymbol{W}_{d_c} \boldsymbol{x} = \boldsymbol{x}^\top \boldsymbol{W}_{d^2} \boldsymbol{x} + 2c\boldsymbol{x}^\top \boldsymbol{W}_d \boldsymbol{x} + \frac{c^2}{2} \boldsymbol{x}^\top \boldsymbol{C} \boldsymbol{x}$$

is nonnegative for all $\boldsymbol{x}$.

Theorem:

There exists a constant c^* such that the dissimilarity d_c has an euclidean representation for all $c \geq c^*$, and the euclidean representation of d_{c^*} defines a space with at most $(n-2)$ dimensions.

Additive constant problem (cont'd)

Proof:
For a given $\boldsymbol{x}$, $\boldsymbol{x}^\top \boldsymbol{W}_{d_c} \boldsymbol{x}$ is function of c represented by a convex parabola; thus, to any $\boldsymbol{x}$ corresponds a number $\alpha(\boldsymbol{x})$ such that:

$$\boldsymbol{x}^\top \boldsymbol{W}_{d_c} \boldsymbol{x} \geq 0 \quad \text{if} \quad c \geq \alpha(\boldsymbol{x}).$$

Observe that

$$\alpha(\boldsymbol{x}) = \alpha(k\boldsymbol{x}) \quad \text{for any number } k.$$

If the parabola is never negative,

$$\alpha(\boldsymbol{x}) = -\infty.$$

For example,

$$\alpha(\boldsymbol{j}) = -\infty.$$

Additive constant problem (cont'd)

Because d^2 has no euclidean representation, there is at least one $\boldsymbol{x}$ where $\boldsymbol{x}^\top \boldsymbol{W}_{d^2}\boldsymbol{x} < 0$ and for which $\alpha(\boldsymbol{x})$ will be positive; thus, the number:

$$c^* = \sup_{\boldsymbol{x}} \alpha(\boldsymbol{x}) = \alpha(\boldsymbol{x}^*)$$

is positive and such that:

$$\boldsymbol{x}^\top \boldsymbol{W}_{d_c}\boldsymbol{x} \geq 0 \quad \text{for all } \boldsymbol{x} \text{ and all } c \geq c^* \tag{7}$$

$$\boldsymbol{x}^{*\top} \boldsymbol{W}_{d_{c^*}}\boldsymbol{x}^* = 0. \tag{8}$$

The eq.(7) proves that d_c has an euclidean representation for all $c \geq c^*$; The eq.(8) shows that the representation of d_c^* defines a space with at most $(n-2)$ dimensions, since $\boldsymbol{W}_{d_{c^*}}$ has at least two independent eigenvectors ($\boldsymbol{j}$ and $\boldsymbol{x}^*$) associated to the eigenvalue zero.

Calculation of c^*

As a consequence of eq.(7) and eq.(8), c^* and $\boldsymbol{x}^*$ verify

$$\begin{aligned} & \boldsymbol{W}_{d_c^*}\boldsymbol{x}^* = \boldsymbol{0} \\ \Longrightarrow \quad & (\boldsymbol{W}_{d^2} + 2c^*\boldsymbol{W}_d + \frac{c^{*2}}{2}\boldsymbol{C})\boldsymbol{x}^* = \boldsymbol{0} \\ \Longrightarrow \quad & (\boldsymbol{W}_{d^2} + 2c^*\boldsymbol{W}_d + \frac{c^{*2}}{2}\boldsymbol{I})\boldsymbol{C}\boldsymbol{x}^* = \boldsymbol{0}. \end{aligned} \tag{9}$$

Let

$$2\boldsymbol{W}_{d^2}\boldsymbol{x}^* = c^*\boldsymbol{y}, \tag{10}$$

we have the equation from eq.(9) since c^* is strictly positive :

$$\boldsymbol{y} + 4\boldsymbol{W}_d\boldsymbol{x}^* + c^*\boldsymbol{C}\boldsymbol{x}^* = \boldsymbol{0} \tag{11}$$

Calculation of c^* (cont'd)

Equalities eq.(10) and eq.(11) are gathered in the following:

$$\begin{pmatrix} \mathbf{0} & 2\boldsymbol{W}_{d^2} \\ -\boldsymbol{I} & -4\boldsymbol{W}_d \end{pmatrix} \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{C}\boldsymbol{x}^* \end{pmatrix} = c^* \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{C}\boldsymbol{x}^* \end{pmatrix},$$

which proves that c^* is an eigenvalue of the matrix

$$\boldsymbol{B} = \begin{pmatrix} \mathbf{0} & 2\boldsymbol{W}_{d^2} \\ -\boldsymbol{I} & -4\boldsymbol{W}_d \end{pmatrix}.$$

Calculation of c^* (cont'd)

Let a be an eigenvalue of $\boldsymbol{B}$ associated to the eigenvector $\begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{t} \end{pmatrix}$:

$$\begin{cases} 2\boldsymbol{W}_{d^2}\boldsymbol{t} = a\boldsymbol{z} \\ -\boldsymbol{z} - 4\boldsymbol{W}_d\boldsymbol{t} = a\boldsymbol{t}\,' \end{cases}$$

we have

$$(\boldsymbol{W}_{d^2} + 2a\boldsymbol{W}_d + \frac{a^2}{2}\boldsymbol{I})\boldsymbol{t} = \boldsymbol{0}$$

$$\Longrightarrow \quad (\boldsymbol{W}_{d^2} + 2a\boldsymbol{W}_d + \frac{a^2}{2}\boldsymbol{C})\boldsymbol{t} = \quad \boldsymbol{0}$$

$$\Longleftrightarrow \quad \boldsymbol{t}^\top \boldsymbol{W}_{d_a}\boldsymbol{t} = 0.$$

Thus, $\alpha(\boldsymbol{t}) \geq a$, which implies $c^* \geq a$ because of the definition of c^*. The additive constant c^* is thus the largest eigenvalue of matrix $\boldsymbol{B}$.

Classical Isomap algorithm

step 1: Identify k nearest neighbors (or ϵ ball neighborhood) of each input data point and construct a neighborhood graph where edge lengths between points in a neighborhood are set as their Euclidean distances.

step 2: Compute geodesic distances, $d_G(i,j)$, that are associated with the sum of edge weights along shortest paths between all pairs of points.

step 3: Construct a matrix $\boldsymbol{K}(\boldsymbol{D}_G^2) = -\frac{1}{2}\boldsymbol{C}\boldsymbol{D}_G^2\boldsymbol{C}$.

step 4: Compute top d eigenvectors of $\boldsymbol{K}(\boldsymbol{D}_G^2)$ which leads to the eigenvector matrix $\boldsymbol{V} \in \mathbb{R}^{n\times d}$ and the eigenvalue matrix $\Lambda \in \mathbb{R}^{d\times d}$.

step 5: The coordinates of n points in the d-dimensional Euclidean space are given by the column vectors of $\boldsymbol{Y} = \Lambda^{1/2}\boldsymbol{V}^\top$.

Kernel Isomap algorithm

step 1: Identify k nearest neighbors (or ϵ ball neighborhood) of each input data point and construct a neighborhood graph.

step 2: Compute geodesic distances, that are associated with the sum of edge weights along shortest paths between all pairs of points.

step 3: Construct a matrix $\boldsymbol{K}(\boldsymbol{D}^2) = -\frac{1}{2}\boldsymbol{C}\boldsymbol{D}^2\boldsymbol{C}$.

step 4: Compute the largest eigenvalue c^* of the matrix

$$\begin{bmatrix} \mathbf{0} & 2\boldsymbol{K}(\boldsymbol{D}^2) \\ -\boldsymbol{I} & -4\boldsymbol{K}(\boldsymbol{D}) \end{bmatrix} \tag{12}$$

and construct a Mercer kernel matrix $\widetilde{\boldsymbol{K}} = \widetilde{\boldsymbol{K}}(\widetilde{\boldsymbol{D}}^2)$, that is,

$$\widetilde{\boldsymbol{K}} = \boldsymbol{K}(\boldsymbol{D}^2) + 2c\boldsymbol{K}(\boldsymbol{D}) + \frac{1}{2}c^2\boldsymbol{C},$$

where $\widetilde{\boldsymbol{K}}$ is guaranteed to be positive semidefinite for $c \geq c^*$.

Kernel Isomap algorithm (cont'd)

step 5: Compute top d eigenvectors of $\widetilde{\boldsymbol{K}}$ which leads to the eigenvector matrix $\boldsymbol{V} \in \mathbb{R}^{n \times d}$ and the eigenvalue matrix $\Lambda \in \mathbb{R}^{d \times d}$.

step 6: The coordinates of n points in the d-dimensional Euclidean space are given by the column vectors of $\boldsymbol{Y} = \Lambda^{1/2} \boldsymbol{V}^{\top}$.

References

- I. Borg and P. J.F. Groenen. *Modern Mutidimensional Scaling* (2ed ed.). Springer.
- K. Q. Weinberger and L. K. Saul (2006), Unsupervised Learning of image manifolds by semidefinite programming, *International Journal of Computer Vision*, 70(1), pp. 77-90. doi: 10.1007/s11263-005-4939-z.
- X. Zhen and Z. Liaangliang. The simplified expression of machine learning and multivariate statistical analysis based on the centering matrix, 2021.
- C. Heeyoul and C. Seungjin. Robust kernel Isomap, *ScienceDirect*, 2006, pp. 853-862.
- F. Cailliez (1983). The analytical solution of the additive constant problem, *PSYCHOMETRIKA*, 40(2), pp. 305-308.
- A. Saxena, A. Gupta, and A. Mukerjee (2004). Non-linear dimensionality reduction by locally linear Isomaps. *Lecture Notes in Computer Science*(pp. 1038-1043). Research Gate.

References

- J. B. Tenenbaum, V. D. Silva, and J. C. Langford (2000). A global framework for nonlinear dimensionality reduction. *SCIENCE*, 290, pp. 2319-2322.

Diffusion maps

The basic ideas of Diffusion maps

Let $\boldsymbol{X} \in \mathbb{R}^{D\times n}$ be the high-dimensional data matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix},$$

and $\boldsymbol{Y} \in \mathbb{R}^{d\times n}$ be the dimension reduced data

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_n \end{bmatrix}.$$

Diffusion map is construct the reduction data from intrinsic local geometric relationship, i.e,

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\mathrm{D}} \text{ is small } \Longrightarrow \ \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2 \text{ is small,}$$

where $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_D$ is diffusion distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Connectivity

Suppose we take a random walk on our data, jumping between data points in feature space.

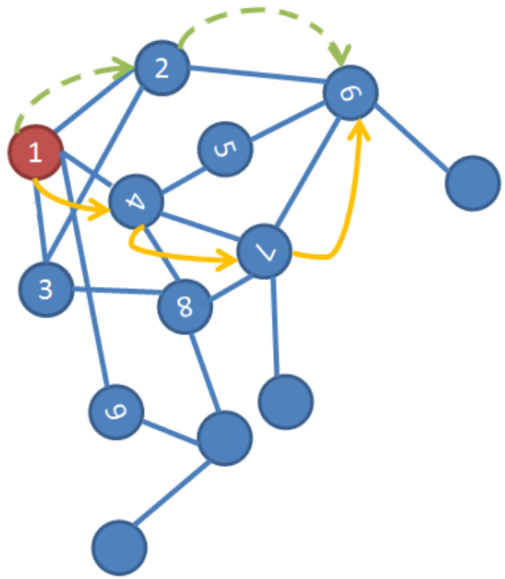


Figure: A random walk on a data set. Each ”jump” has a probability associated with it. The dashed path between nodes 1 and 6 requires two jumps (i.e., two time units) with the probability along the path being p(node 1, node 2) p(node 2, node 6).

Connectivity (cont'd)

The connectivity between two data points, $\boldsymbol{x}$ and $\boldsymbol{y}$, is defined as the probability of jumping from $\boldsymbol{x}$ to $\boldsymbol{y}$ in one step of the random walk, and is

$$\text{connectivity}(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x}, \boldsymbol{y}). \tag{13}$$

It is useful to express this connectivity in terms of a non-normalized likelihood function, k, known as the diffusion kernel:

$$\text{connectivity}(\boldsymbol{x}, \boldsymbol{y}) \propto k(\boldsymbol{x}, \boldsymbol{y}). \tag{14}$$

Connectivity (cont'd)

The diffusion kernel satisfies the following properties:

1. k is symmetric : $k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{y}, \boldsymbol{x})$
2. k is positivity preserving : $k(\boldsymbol{x}, \boldsymbol{y}) \geq 0$

The latter property is specific to the diffusion kernel and allows it to be interpreted as a scaled probability, so that

$$\frac{1}{d_X} \sum_{y \in \boldsymbol{X}} k(\boldsymbol{x}, \boldsymbol{y}) = 1.$$

The relation between the kernel function and the connectivity is then

$$\text{connectivity}(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{d_X} k(\boldsymbol{x}, \boldsymbol{y})$$

with $\frac{1}{d_X}$ the normalization constant.

Connectivity (cont'd)

Define a row-normalized diffusion matrix, $\boldsymbol{P}$, with entries

$$\boldsymbol{P}_{ij} = p(x_i, x_j).$$

By taking powers of the diffusion matrix, we can increase the number of steps taken. For example, take a 2×2 diffusion matrix,

$$\boldsymbol{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}.$$

Each element, P_{ij}, is the probability of jumping between data points i and j. When $\boldsymbol{P}$ is squared, it becomes

$$\boldsymbol{P}^2 = \begin{bmatrix} p_{11}p_{11} + p_{12}p_{21} & p_{12}p_{22} + p_{11}p_{12} \\ p_{21}p_{11} + p_{22}p_{21} & p_{22}p_{22} + p_{21}p_{12} \end{bmatrix}.$$

When making two jumps, these $\boldsymbol{P}_{ij}$ are all the paths from point i to point j. Similarly, $\boldsymbol{P}^t$ sum all paths of length t from point i to point j.

Diffusion Process

With increased values of t (i.e. as the diffusion process), the probability of following a path along the underlying geometric structure of the data set increases.

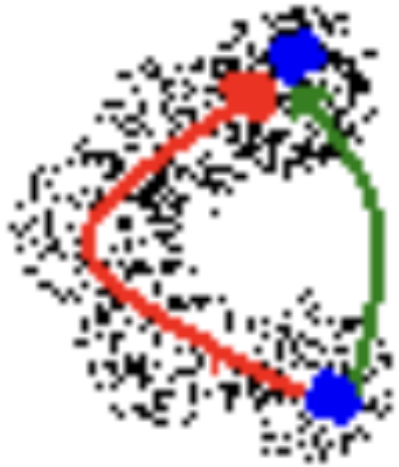


Figure: Paths along the true geometric structure of the data set have high probability.

Diffusion Distance

The diffusion distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined by

$$\begin{aligned} D_t(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 &= \sum_{\boldsymbol{u} \in X} |p_t(\boldsymbol{x}_i, \boldsymbol{u}) - p_t(\boldsymbol{x}_j, \boldsymbol{u})|^2 \\ &= \sum_k |\boldsymbol{P}^t_{ik} - \boldsymbol{P}^t_{jk}|^2 \end{aligned}$$

The diffusion distance is small if there are many high probability paths of length t between two points.

Diffusion map

The diffusion distance in data space simply becomes the Euclidean distance in this new diffusion space. With this in mind, we define and examine the mapping

$$\boldsymbol{y}_i := \begin{bmatrix} p_t(\boldsymbol{x}_i, \boldsymbol{x}_1) \\ p_t(\boldsymbol{x}_i, \boldsymbol{x}_2) \\ \vdots \\ p_t(\boldsymbol{x}_i, \boldsymbol{x}_n) \end{bmatrix} = \boldsymbol{P}_{i*}^{\top}. \tag{15}$$

For this map, the Euclidean distance between y_i and y_j is

$$\begin{aligned} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 &= \sum_{\boldsymbol{u} \in X} |p_t(\boldsymbol{x}_i, \boldsymbol{u}) - p_t(\boldsymbol{x}_j, \boldsymbol{u})|^2 \\ &= \sum_k |\boldsymbol{P}_{ik}^t - \boldsymbol{P}_{jk}^t|^2 = D_t(\boldsymbol{x}_i, \boldsymbol{x}_j)^2. \end{aligned}$$

Note that the dimension of the mapped data is still the sample size n.

Diffusion map (cont'd)

Lemma 1: Suppose $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ is a kernel matrix such that $\boldsymbol{K}[i,j] = k(i,j)$. A diagonal matrix, $\boldsymbol{D}$, normalizes the rows of $\boldsymbol{K}$ to produce a diffusion matrix

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{K}. \tag{16}$$

Then, matrix $\boldsymbol{P}'$, defined as

$$\boldsymbol{P}' = \boldsymbol{D}^{1/2}\boldsymbol{P}\boldsymbol{D}^{-1/2}, \tag{17}$$

1. is symmetric.
2. has the same eigenvalues as $\boldsymbol{P}$.
3. the eigenvectors, $\boldsymbol{x}'_k$ of $\boldsymbol{P}'$ are multiplied by $\boldsymbol{D}^{-1/2}$ and $\boldsymbol{D}^{1/2}$ to get the left $(\boldsymbol{u}^\top \boldsymbol{P} = \lambda \boldsymbol{u}^\top)$ and right eigenvectors $(\boldsymbol{P}\boldsymbol{v} = \lambda \boldsymbol{v})$ of $\boldsymbol{P}$ respectively.

Diffusion map (cont'd)

proof : Substitute (16) into (17) to obtain

$$\boldsymbol{P}' = \boldsymbol{D}^{-1/2}\boldsymbol{K}\boldsymbol{D}^{-1/2}. \tag{18}$$

As $\boldsymbol{K}$ is symmetric, $\boldsymbol{P}'$ will also be symmetric.
We can make $\boldsymbol{P}$ the subject of the equation in (17),

$$\boldsymbol{P} = \boldsymbol{D}^{-1/2}\boldsymbol{P}'\boldsymbol{D}^{1/2}. \tag{19}$$

As $\boldsymbol{P}'$ is symmetric, by spectrum analysis such that

$$\boldsymbol{P}' = \boldsymbol{S}\Lambda\boldsymbol{S}^{\top} \tag{20}$$

where Λ is is a diagonal matrix containing the eigenvalues of $\boldsymbol{P}'$ and $\boldsymbol{S}$ is a matrix with the orthonormal eigenvectors of $\boldsymbol{P}'$ as columns.

Diffusion map (cont'd)

Substituting (20) into (19),

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{D}^{-1/2}\boldsymbol{S}\Lambda\boldsymbol{S}^{\top}\boldsymbol{D}^{1/2} && (21)\\ &= \boldsymbol{D}^{-1/2}\boldsymbol{S}\Lambda\boldsymbol{S}^{-1}\boldsymbol{D}^{1/2} && (22)\\ &= (\boldsymbol{D}^{-1/2}\boldsymbol{S})\Lambda(\boldsymbol{S}^{-1}\boldsymbol{D}^{1/2}) && (23)\\ &= \boldsymbol{Q}\Lambda\boldsymbol{Q}^{-1}, && (24) \end{aligned}$$

where $\boldsymbol{Q} = \boldsymbol{D}^{-1/2}\boldsymbol{S}$. Therefore, the eigenvalues of $\boldsymbol{P}$ and $\boldsymbol{P}'$ are the same.

1. The right eigenvectors of $\boldsymbol{P}$ are the columns of $\boldsymbol{Q}$
2. The left eigenvectors of $\boldsymbol{P}$ are the rows of $\boldsymbol{Q}^{-1}$

Diffusion map (cont'd)

Observe that the eigenvectors of $\boldsymbol{P}$ can be given in terms of the eigenvectors x'_k of $\boldsymbol{P}'$.The right eigenvectors of $\boldsymbol{P}$ are

$$\boldsymbol{v}_k = \boldsymbol{D}^{-1/2}\boldsymbol{x}'_k \tag{25}$$

and the left eigenvectors are

$$\boldsymbol{u}_k = \boldsymbol{D}^{1/2}\boldsymbol{x}'_k. \tag{26}$$

by SVD, we obtain

$$\boldsymbol{P} = \sum_k \lambda_k \boldsymbol{v}_k \boldsymbol{u}_k^\top. \tag{27}$$

Eq.(27) expresses each row of the diffusion matrix in terms of a new basis: $\boldsymbol{u}_k$, the left eigenvectors of the diffusion matrix.

Diffusion map (cont'd)

In this new coordinate system in $\mathbb{R}^n$, a row i of $\boldsymbol{P}$, which is $(\boldsymbol{y}_i)$ is represented by the point

$$M_i = \begin{bmatrix} \lambda_1 v_1[i] \\ \lambda_2 v_2[i] \\ \vdots \\ \lambda_n v_n[i] \end{bmatrix},$$

where $v_n[i]$ is the i-th component of the n-th right eigenvector. Map to the d-dimensional diffusion space at time t, using the d dominant eigenvectors and -values as

$$\boldsymbol{y}_i = \begin{bmatrix} \lambda_1^t v_1[i] \\ \lambda_2^t v_2[i] \\ \vdots \\ \lambda_d^t v_d[i] \end{bmatrix}.$$

References

- J. Porte, B. M. Herbst, W. Hereman, and S. J. Walt. An Introduction to diffusion maps, 2008.

Laplacian Eigenmaps (LE)

The basic ideas of LE

Let X be a dataset

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \in \mathbb{R}^{D\times n},$$

and Y be the dimension reduced data

$$Y = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} \in \mathbb{R}^{d\times n}.$$

LE construct the reduction data from intrinsic local geometric relationship, i.e,

$$\|x_i - x_j\|_2 \text{ is small } \implies \|y_i - y_j\|_2 \text{ is small,}$$

where x_i and x_j are original high dimensional data, and y_i and y_j are lower dimensional data.

Constructing the adjacency graph

1) ϵ-neighborhoods (parameter $\epsilon \in \mathbb{R}$). Nodes i and j are connected by ad edge if $\|x_i - x_j\|^2 < \epsilon$ where the norm is the usual Euclidean norm.

2) k nearest neighbors (parameter $k \in \mathbb{N}$). Node i and j are connected by an edge if i is among k nearest neighbors of j or j is among k nearest neighbors of i. Note that this relation is symmetric.

3) Fully connected.

Choosing the weights

Let $\boldsymbol{W} \in \mathbb{R}^{n \times n}$, we have two variations for weighting the edges:

1) Heat kernel (parameter $t \in \mathbb{R}$). If nodes i and j are connected, put

$$\boldsymbol{W}_{ij} = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{t}};$$

otherwise, put $\boldsymbol{W}_{ij} = 0$.

2) Simple-minded (no parameters ($t = \infty$)).

$$\begin{cases} \boldsymbol{W}_{ij} = 1, & \text{if vertices } i \text{ and } j \text{ are connected by an edge.} \\ \boldsymbol{W}_{ij} = 0, & \text{if vertices } i \text{ and } j \text{ are not connected by an edge.} \end{cases}$$

This simplification avoids the need to choose t.

Optimization problem

This motivates us define the cost function $\Phi(\boldsymbol{Y})$:

$$\begin{aligned}
\Phi(\boldsymbol{Y}) &= \sum_{i,j} \boldsymbol{W}_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 \\
&= \sum_{i,j} \boldsymbol{W}_{ij} (\|\boldsymbol{y}_i\|^2 + \|\boldsymbol{y}_j\|^2 - 2\boldsymbol{y}_i^\top \boldsymbol{y}_j) \\
&= \sum_i \|\boldsymbol{y}_i\|^2 \sum_j \boldsymbol{W}_{ij} + \sum_j \|\boldsymbol{y}_i\|^2 \sum_i \boldsymbol{W}_{ij} - 2 \sum_{ij} \boldsymbol{W}_{ij} \boldsymbol{y}_i^\top \boldsymbol{y}_j \\
&= 2 \sum_i \|\boldsymbol{y}_i\|^2 \sum_j \boldsymbol{W}_{ij} - 2 \sum_{ij} \boldsymbol{W}_{ij} \boldsymbol{y}_i^\top \boldsymbol{y}_j \\
&= 2 \sum_i \|\boldsymbol{y}_i\|^2 d_{ii} - 2 \sum_{ij} \boldsymbol{W}_{ij} \boldsymbol{y}_i^\top \boldsymbol{y}_j,
\end{aligned}$$

where $d_{ii} = \sum_{j=1}^n \boldsymbol{W}_{ij}$.

Optimization problem

$$\begin{aligned}\Phi(\boldsymbol{Y}) &= 2\mathrm{tr}(\boldsymbol{Y}\boldsymbol{D}\boldsymbol{Y}^\top) - 2\mathrm{tr}(\boldsymbol{Y}\boldsymbol{W}\boldsymbol{Y}^\top) \\ &= 2\mathrm{tr}(\boldsymbol{Y}(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{Y}^\top) \\ &= 2\mathrm{tr}(\boldsymbol{Y}\mathscr{L}\boldsymbol{Y}^\top),\end{aligned}$$

where $\boldsymbol{D} = \mathrm{diag}(d_{ii}) \in \mathbb{R}^{n\times n}$, $\mathscr{L} = \boldsymbol{D} - \boldsymbol{W} \in \mathbb{R}^{n\times n}$ is called Laplacian matrix.
So the optimization problem is

$$\min_{\boldsymbol{Y}\in\mathbb{R}^{d\times n}} \mathrm{tr}(\boldsymbol{Y}\mathscr{L}\boldsymbol{Y}^\top) \quad \text{subject to } \boldsymbol{Y}\boldsymbol{D}\mathbf{1}_n = 0\,,\ \boldsymbol{Y}\boldsymbol{D}\boldsymbol{Y}^\top = \boldsymbol{I}_{d\times d},$$

where $\mathbf{1}_n \in \mathbb{R}^n$ is a vector with all 1's and $\boldsymbol{I}$ is identity matrix.

Optimization problem

Define lagrange function

$$L_{\lambda,\Lambda}(\boldsymbol{Y}) = \mathrm{tr}(\boldsymbol{Y}\mathscr{L}\boldsymbol{Y}^\top) + \lambda^\top \boldsymbol{Y}\boldsymbol{D}\mathbf{1}_n + \mathrm{tr}(\Lambda(\boldsymbol{I} - \boldsymbol{Y}\boldsymbol{D}\boldsymbol{Y}^\top)),$$

where $\lambda \in \mathbb{R}^d$ and $\Lambda \in \mathbb{R}^{d\times d}$, diagonal matrix, are lagrange multipliers.
Find

$$\frac{\partial L}{\partial \boldsymbol{Y}^\top} = 2\mathscr{L}\boldsymbol{Y}^\top + \boldsymbol{D}\mathbf{1}_n\lambda^\top - 2\boldsymbol{D}\boldsymbol{Y}^\top\Lambda = 0,$$

we obtained

$$\begin{aligned} \lambda &= \mathbf{0} \\ \mathscr{L}\boldsymbol{Y}^\top &= \boldsymbol{D}\boldsymbol{Y}^\top\Lambda. \end{aligned}$$

Hence

$$\boldsymbol{D}^{-1}\mathscr{L}\boldsymbol{Y}^\top = \boldsymbol{Y}^\top\Lambda.$$

Eigenvalue problem

From above results, we get a eigenvalue problem of $D^{-1}\mathscr{L}$.
Observing that

$$\begin{aligned}
\mathscr{L}\mathbb{1}_N &= (D - W)\mathbf{1}_n \\
&= D\mathbf{1}_n - W\mathbf{1}_n \\
&= \begin{bmatrix} d_{11} \\ d_{22} \\ \vdots \\ d_{nn} \end{bmatrix} - \begin{bmatrix} \sum_j W_{1j} \\ \sum_j W_{2j} \\ \vdots \\ \sum_j W_{nj} \end{bmatrix} \\
&= \mathbf{0}_n,
\end{aligned}$$

the $\mathbf{1}_n$ is eigenvector corresponding to 0 eigenvalue conflicts with first constraint, so we select the eigenvector corresponding to the second smallest eigenvalue to the $(d+1)$-th smallest eigenvalue.

References

- M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering, 2001.

Locally linearly embedding (LLE)

The basic ideas of LLE

Let $\boldsymbol{X}$ be a dataset

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix} \in \mathbb{R}^{D \times n},$$

and $\boldsymbol{Y}$ be the dimension reduced data

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_n \end{bmatrix} \in \mathbb{R}^{d \times n}.$$

LLE construct the dimension reduction data from intrinsic local linear relationship. For example,

$$\boldsymbol{x}_1 = c_1 \boldsymbol{x}_2 + c_2 \boldsymbol{x}_3 + c_3 \boldsymbol{x}_4,$$

where c_1, c_2, and c_3 are called weight coefficients. After using LLE to reduce dimension,

$$\boldsymbol{y}_1 \approx c_1 \boldsymbol{y}_2 + c_2 \boldsymbol{y}_3 + c_3 \boldsymbol{y}_4.$$

LLE

We choose $k \geq d$, and use K-nearest neighbor(KNN) algorithm to find k-nearest neighbor of each x_j.
Denoting the neighbor matrices of x_j by

$$N_j = knn(x_j, k) = \begin{bmatrix} x_{j_1} & \cdots & x_{j_k} \end{bmatrix} \in \mathbb{R}^{D \times k}, \quad 1 \leq j \leq n,$$

where $j_i \in \{1, 2, \ldots, n\}$.
Then we also define coefficient matrix $\boldsymbol{W} \in \mathbb{R}^{k \times n}$

$$\boldsymbol{W} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix},$$

where weights $w_j \in \mathbb{R}^k$ is coefficient vector, summarizing the contribution of the x_{j_i} to the jth reconstruction. That is,

$$x_j = N_j w_j = \begin{bmatrix} x_{j_1} & x_{j_2} & \cdots & x_{j_k} \end{bmatrix} w_j, \quad 1 \leq j \leq n.$$

LLE algorithm

Step 1 . Find the coefficient matrix, i.e, set $\boldsymbol{w}_j$ to be a solution of the problems

$$\min_{\boldsymbol{w}_j \in \mathbb{R}^k} \sum_{j=1}^{n} \|\boldsymbol{x}_j - \sum_{i=1}^{k} \boldsymbol{W}_{ij} \boldsymbol{x}_{j_i}\|_2^2 \quad \text{subject to } \sum_{i=1}^{k} \boldsymbol{W}_{ij} = 1\,, \quad 1 \le j \le n. \quad (P1)$$

Step 2 . With $\boldsymbol{w}_j$ given from Step 1, set $\boldsymbol{Y}$ to be a solution of the problem

$$\min_{\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathbb{R}^d} \sum_{j=1}^{n} \|\boldsymbol{y}_j - \sum_{i=1}^{k} \boldsymbol{W}_{ij} \boldsymbol{y}_{j_i}\|_2^2 \text{ subject to } \sum_{j=1}^{n} \boldsymbol{y}_j = \boldsymbol{0} \text{and } \boldsymbol{Y}\boldsymbol{Y}^\top = n \times \boldsymbol{I}_d, \quad (P2)$$

where $\boldsymbol{I}_d$ is $d \times d$ the identity matrix.

Optimizing the weights

Start by writing the latter using matrix notation,

$$\begin{aligned}
\sum_{j=1}^{n} \|x_j - \sum_{i=1}^{k} W_{ij} x_{j_i}\|^2 &= \sum_{j=1}^{n} \|1 \times x_j - \sum_{i=1}^{k} W_{ij} x_{j_i}\|^2 \\
&= \sum_{j=1}^{n} \| \sum_{i=1}^{k} W_{ij} x_j - \sum_{i=1}^{k} W_{ij} x_{j_i}\|^2 \\
&= \sum_{j=1}^{n} \| \sum_{i=1}^{k} W_{ij} (x_j - x_{j_i})\|^2. \\
&= \sum_{j=1}^{n} \|(\widetilde{X}_j - N_j) w_j\|^2,
\end{aligned}$$

where $\widetilde{X}_j = \begin{bmatrix} x_j & \cdots & x_j \end{bmatrix} \in \mathbb{R}^{D \times k}$.

Optimizing the weights

It follows that

$$\begin{aligned}\sum_{j=1}^{n} \|(\widetilde{\boldsymbol{X}}_j - N_j)\boldsymbol{w}_j\|^2 &= \sum_{j=1}^{n} \boldsymbol{w}_j^\top (\widetilde{\boldsymbol{X}}_j - N_j)^\top (\widetilde{\boldsymbol{X}}_j - N_j)\boldsymbol{w}_j \\ &= \sum_{j=1}^{n} \boldsymbol{w}_j^\top \boldsymbol{S}_j \boldsymbol{w}_j, \end{aligned} \tag{28}$$

where $\boldsymbol{S}_j = (\widetilde{\boldsymbol{X}}_j - N_j)^\top (\widetilde{\boldsymbol{X}}_j - N_j) \in \mathbb{R}^{k\times k}$ are *local covariance matrix*. And the constraint

$$\sum_{i=1}^{k} \boldsymbol{W}_{ij} = \mathbf{1}_k^\top \boldsymbol{w}_j = 1,$$

where $\mathbf{1}_k \in \mathbb{R}^k$ is vector with all components are 1.

Optimizing the weights

Hence, the problems (Q1) can be restated componentwise as

$$\min_{\boldsymbol{w}_j} \ \boldsymbol{w}_j^\top \boldsymbol{S}_j \boldsymbol{w}_j \quad \text{subject to} \quad \mathbf{1}_k^\top \boldsymbol{w}_j = 1\,, \quad 1 \le j \le n.$$

Define a Lagrange function

$$L_\lambda(\boldsymbol{w}_j) = \boldsymbol{w}_j^\top \boldsymbol{S}_j \boldsymbol{w}_j + \lambda(\mathbf{1}_k^\top \boldsymbol{w}_j - 1),$$

where $\lambda \in \mathbb{R}$ is a Lagrange multiplier.
And look for its critical points, i.e, vector $\boldsymbol{w}_j$ such that

$$\nabla_{\boldsymbol{w}_j} L_\lambda(\boldsymbol{w}_j) = 2\boldsymbol{S}_j \boldsymbol{w}_j + \lambda \mathbf{1}_k = 0.$$

We have

$$\boldsymbol{w}_j = -\frac{\lambda \boldsymbol{S}_j^{-1} \mathbf{1}_k}{2}. \tag{29}$$

Optimizing the weights

Then focusing on constraint,

$$\mathbf{1}_k^\top \boldsymbol{w}_j = -\frac{\lambda \mathbf{1}_k^\top \boldsymbol{S}_j^{-1} \mathbf{1}_k}{2} = 1,$$

that gives

$$\lambda = -\frac{2}{\mathbf{1}_k^\top \boldsymbol{S}_j^{-1} \mathbf{1}_k}.$$

Hence, subtituting the expression of λ in terms of w_j in (29),

$$\boldsymbol{w}_j = \frac{\boldsymbol{S}_j^{-1} \mathbf{1}_k}{\mathbf{1}_k^\top \boldsymbol{S}_j^{-1} \mathbf{1}_k},$$

where the denominator is sum of entries of $\boldsymbol{S}_j^{-1}$, and numerator is sum of column vectors of $\boldsymbol{S}_j^{-1}$. It has a unique solution if the *local covariance matrix* $\boldsymbol{S}_j$ is invertible.

Optimizing the weights

On the other hand, if S_j is singular, it may have multiple solutions. In this situation it is suggested that we add a small regularization term ϵI to S_j, and set

$$w_j = \frac{(S_j + \epsilon I)^{-1} \mathbf{1}_k}{\mathbf{1}_k^\top (S_j + \epsilon I)^{-1} \mathbf{1}_k}.$$

Mapping to lower dimensional $\mathbb{R}^d$

We begin by introducing a sparse matrix $\widetilde{W} \in \mathbb{R}^{n\times n}$,

$$\widetilde{W}_{j_i,j} = \begin{cases} W_{ij} & \text{, if } x_{j_i} \text{ is column of } N_j \\ \mathbf{0} & \text{, if } x_{j_i} \text{ is not column of } N_j \end{cases}, \quad \text{for } 1 \le i \le k \text{ and } 1 \le j \le n.$$

That is $\widetilde{W}_{ij} = 0$ if x_i does not belong to the set of neighbors of x_j.
Observing that

$$\sum_{i=1}^{k} W_{ij} y_{j_i} = Y \widetilde{W}_j,$$

so we can rewrite the cost function,

$$\sum_{j=1}^{N} \|y_j - \sum_{i=1}^{k} W_{ij} y_{j_i}\|^2 = \sum_{j=1}^{N} \|Y(I_j - \widetilde{W}_j)\|^2,$$

where I_j is jth column of identity matrix $I \in \mathbb{R}^{d\times d}$.

Mapping to lower dimensional $\mathbb{R}^d$

It follows that,

$$\begin{aligned}\sum_{j=1}^{N} \|\boldsymbol{Y}(\boldsymbol{I}_j - \widetilde{\boldsymbol{W}}_j)\|^2 &= \operatorname{tr}(\boldsymbol{Y}(\boldsymbol{I} - \widetilde{\boldsymbol{W}})(\boldsymbol{I} - \widetilde{\boldsymbol{W}})^\top \boldsymbol{Y}^\top) \\ &= \operatorname{tr}(\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top)\end{aligned}$$

where

$$\boldsymbol{M} = (\boldsymbol{I} - \widetilde{\boldsymbol{W}})(\boldsymbol{I} - \widetilde{\boldsymbol{W}})^\top \in \mathbb{R}^{n\times n}.$$

Hence, the optimization problem (Q2) can be restate as,

$$\min_{\boldsymbol{Y}} \operatorname{tr}(\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top) \quad \text{subject to } \boldsymbol{Y}\boldsymbol{Y}^\top = N \times \boldsymbol{I} \text{ and } \sum_{j=1}^{n} y_j = \boldsymbol{0}.$$

Mapping to lower dimensional $\mathbb{R}^d$

We define a Lagrange function

$$L_{\lambda,\Lambda}(\boldsymbol{Y}) = \operatorname{tr}(\boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^\top) - \operatorname{tr}(\Lambda(\boldsymbol{Y}\boldsymbol{Y}^\top - n\boldsymbol{I})) - \lambda^\top \boldsymbol{Y}\mathbf{1}_n,$$

where $\Lambda \in \mathbb{R}^{d\times d}$ is a diagonal matrix and $\lambda \in \mathbb{R}^d$. And look for its critical points, i.e, matrix $\boldsymbol{Y}$ such that

$$\frac{\partial L}{\partial \boldsymbol{Y}} = 2\boldsymbol{M}\boldsymbol{Y}^\top - 2\boldsymbol{Y}^\top\Lambda - \mathbf{1}_n\lambda^\top = 0.$$

We get,

$$\lambda = \mathbf{0} \quad \text{and } \boldsymbol{M}\boldsymbol{Y}^\top = \boldsymbol{Y}^\top\Lambda.$$

It can be seen that $\boldsymbol{Y}^\top$ is actually a matrix composed of eigenvectors of $\boldsymbol{M}$. In order to reduce the data to d dimension, we only need to take the eigenvector corresponding to the second smallest eigenvalue to the $(d+1)$-th smallest eigenvalue.

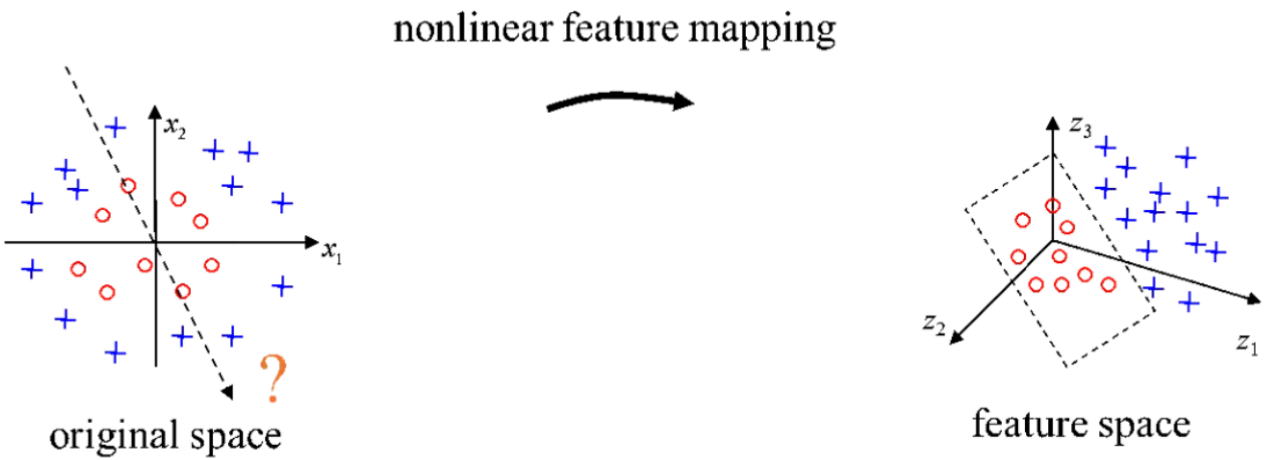
References

- S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding,*SCIENCE*, 290, pp. 2323-2325.
- S. T. Roweis and L. K. Saul. An introduction to locally linear embedding, 2001.
- L. Liren. Avoiding unwanted results in locally linear embedding: A new understanding of regularization, 2021.

Kernel method

Kernel method (Kernel trick)

The role of Kernel trick in machine learning is to hope that the data after an nonlinear projection can be more separated in a higher-dimensional space when different types of data cannot be separated by linear classifiers in the original space.



Kernel function

Definition : For all x_i and x_j in the input space $\mathcal{X}$, certain functions $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be expressed as an inner product in other space $\mathcal{V}$. The function k is often referred to as a *kernel* or a *kernel function*. That is,

$$k(x_i, x_j) = < \phi(x_i), \phi(x_j) >,$$

where $\phi : \mathcal{X} \to \mathcal{V}$ is a nonlinear mapping.
Mercer's condition (Discrete analog) : A matrix $K \in \mathbb{R}^{n \times n}$, $K_{ij} = k(x_i, x_j)$, which satisfies for all vectors $g \in \mathbb{R}^n$, the property,

$$< g, Kg >= g^\top Kg \geq 0.$$

Mercer's theorem : An implicitly defined function ϕ exists whenever the space $\mathcal{X}$ can be equipped with a suitable measure ensuring the kernel function k satisfies Mercer's condition.

Commonly used kernel functions

1. Linear kernel :
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = < \boldsymbol{x}_i, \boldsymbol{x}_j > .$$
2. Polynomial kernel :
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (< \boldsymbol{x}_i, \boldsymbol{x}_j > + c)^d,$$
where $d \in \mathbb{Z}^+$.
3. Gaussian Radial Basis Function kernel (RBF) :
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}},$$
where $\sigma \in \mathbb{R} - \{0\}$.

Kernel-based PCA

Let Φ be a data matrix in the feature space,

$$\Phi = \begin{bmatrix} \phi(x_1) & \phi(x_2) & \cdots & \phi(x_n) \end{bmatrix},$$

where $\phi(\cdot)$ is a nonlinear mapping onto feature space.
Define the covariance matrix

$$\boldsymbol{C} = \frac{1}{n}(\Phi \boldsymbol{H})(\Phi \boldsymbol{H})^\top, \tag{30}$$

where

$$\boldsymbol{H} = \boldsymbol{I}_n - \frac{1}{n}\boldsymbol{J}_n \in \mathbb{R}^{n \times n},$$

is centering matrix, $\boldsymbol{I}_n$ is identity matrix of size n and $\boldsymbol{J}_n$ is an $n \times n$ matrix of all 1's.

Eigenvectors of $\widetilde{C}$

Define the Mercer kernel matrix (using linear kernel here)

$$\boldsymbol{K} = (\Phi \boldsymbol{H})^\top (\Phi \boldsymbol{H}) = \boldsymbol{H}\Phi^\top \Phi \boldsymbol{H} \in \mathbb{R}^{n \times n}$$

The eigendecomposition of $\boldsymbol{K}$ is of the form

$$\boldsymbol{K} = \boldsymbol{V}\Lambda\boldsymbol{V}^\top, \tag{31}$$

where $\boldsymbol{V} = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}$ is orthonormal and $\Lambda = diag(\lambda_i)$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.
Pre-multiplying both sides of Eq.(31) by $\Phi\boldsymbol{H}$ leads to

$$(\Phi \boldsymbol{H})\boldsymbol{K}\boldsymbol{V} = (\Phi \boldsymbol{H})\boldsymbol{V}\Lambda. \tag{32}$$

Then, observing $\boldsymbol{H}$ is symmetric, we have

$$\Phi \boldsymbol{H}\boldsymbol{H}\Phi^\top \Phi \boldsymbol{H}\boldsymbol{V} = \Phi \boldsymbol{H}\boldsymbol{V}\Lambda. \tag{33}$$

Eigenvectors of $\widetilde{C}$ (cont'd)

Thus, Eq.(33) can be written as

$$\Phi HH\Phi^\top \Phi HV = nC\Phi HV = \Phi HV\Lambda \tag{34}$$

$$nCU = U\Lambda, \tag{35}$$

where $U = \Phi HV$ is the eigenvector matrix of C. That is,

$$U = \Phi H \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} \tag{36}$$

$$= \begin{bmatrix} \Phi Hv_1 & \cdots & \Phi Hv_n \end{bmatrix} \tag{37}$$

$$= \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \tag{38}$$

where $u_i = \Phi Hv_i$.
Hence, i-th best projection direction (maximizing the variance) in feature space, u_i, the eigenvector of nC, is found.

Coordinates in the feature space

Normalized the $\boldsymbol{u}_i$

$$\begin{aligned}
\tilde{\boldsymbol{u}}_i &= \frac{1}{\|\Phi \boldsymbol{H} \boldsymbol{v}_i\|} \Phi \boldsymbol{H} \boldsymbol{v}_i = \frac{1}{\sqrt{\boldsymbol{v}_i^\top \boldsymbol{H} \Phi^\top \Phi \boldsymbol{H} \boldsymbol{v}_i}} \Phi \boldsymbol{H} \boldsymbol{v}_i \\
&= \frac{1}{\sqrt{\boldsymbol{v}_i^\top \boldsymbol{H} \Phi^\top \Phi \boldsymbol{H} \boldsymbol{v}_i}} \Phi \boldsymbol{H} \boldsymbol{v}_i = \frac{1}{\sqrt{\boldsymbol{v}_i^\top \boldsymbol{K} \boldsymbol{v}_i}} \Phi \boldsymbol{H} \boldsymbol{v}_i \\
&= \frac{1}{\sqrt{\boldsymbol{v}_i^\top \lambda_i \boldsymbol{v}_i}} \Phi \boldsymbol{H} \boldsymbol{v}_i = \frac{1}{\sqrt{\lambda_i}} \Phi \boldsymbol{H} \boldsymbol{v}_i,
\end{aligned}$$

where λ_i is the corresponding value of $\boldsymbol{v}_i$.

Coordinates in the feature space (cont'd)

Centering Φ and projecting onto the unit vector $\tilde{u}_i$ leads to the eigenvector v_i scaled by $\lambda^{1/2}$, i.e.,

$$\begin{aligned}
< \Phi \boldsymbol{H}, \tilde{u}_i > &= (\Phi \boldsymbol{H})^\top \tilde{u}_i \\
&= (\Phi \boldsymbol{H})^\top \frac{1}{\sqrt{\lambda_i}} \Phi \boldsymbol{H} v_i \\
&= \frac{1}{\sqrt{\lambda_i}} \boldsymbol{H} \Phi^\top \Phi \boldsymbol{H} v_i \\
&= \frac{1}{\sqrt{\lambda_i}} \boldsymbol{K} v_i \\
&= \frac{1}{\sqrt{\lambda_i}} \lambda_i v_i \\
&= \lambda_i^{1/2} v_i.
\end{aligned}$$

References

- X. Zhen and Z. Liaangliang. The simplified expression of machine learning and multivariate statistical analysis based on the centering matrix, 2021.
- C. Heeyoul and C. Seungjin. Robust kernel Isomap, *ScienceDirect*, 2006, pp. 853-862.