Dimensionality reduction

November 2, 2021

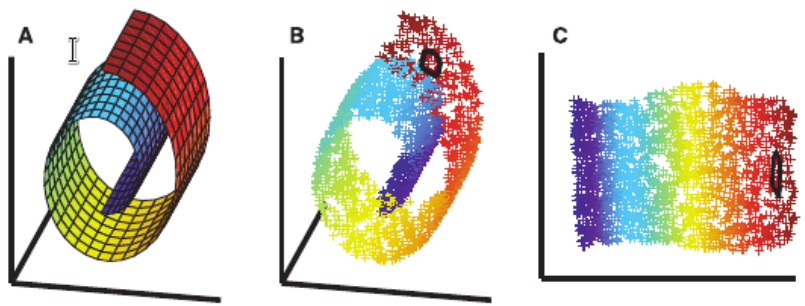
Frederic.Cazals@inria.fr

Dimensionality reduction

Dimensionality reduction

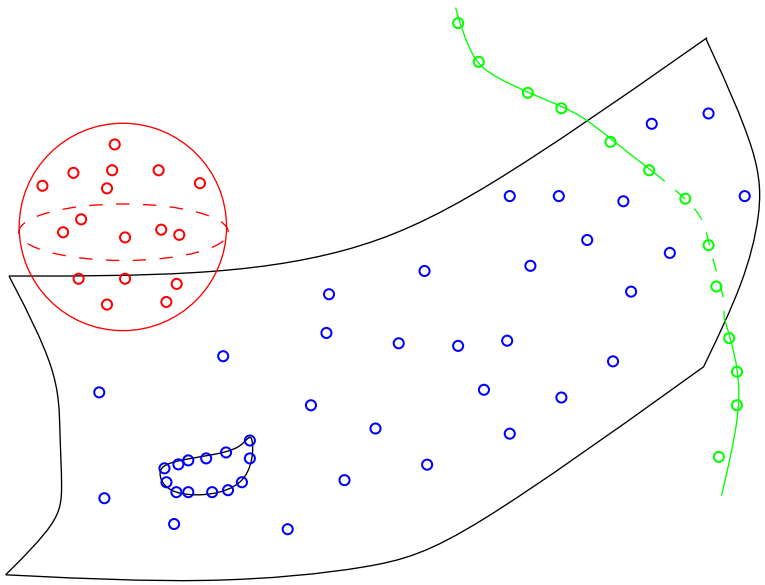
Dimensionality reduction?

- The problem:
  - Given a point cloud $\{x_i\} \in \mathbb{R}^D$
  - There exists a latent model for the data at hand: discover it and map the points into $\mathbb{R}^d$, with $d < D$
- Example: mapping point of the swiss roll from $\mathbb{R}^3$ into $\mathbb{R}^2$



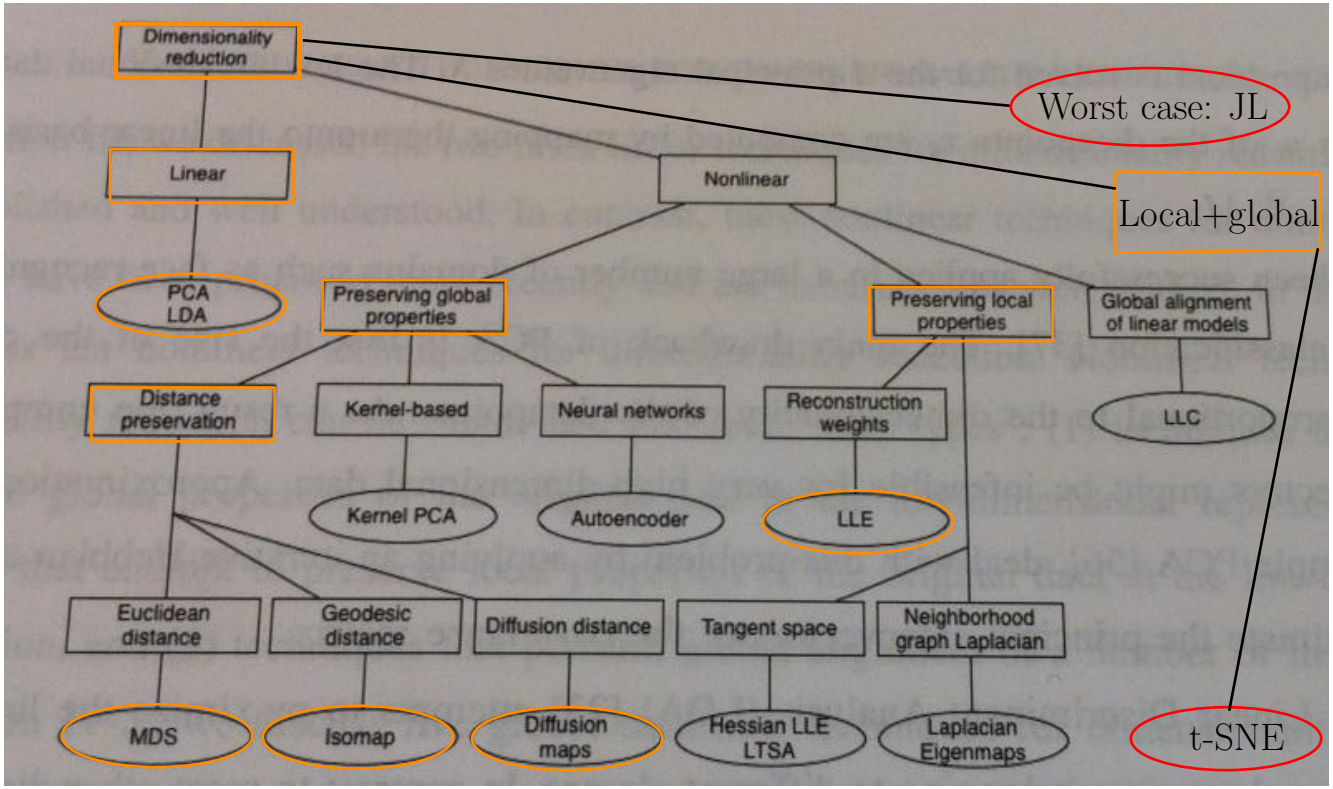
Before getting started: selected questions

- Data and their intrinsic dimension: see lecture #2
- Difficult questions:
  - Underlying geometric model: linear vs non linear, manifold vs stratified space
  - Target dimension: input or output?
  - Criterion optimized: local, global, mix of the two
  - Number of dimensions vs number of samples



A stratified space: pieces of dimension 1, 2, 3

Taxonomy of dimensionality reduction methods



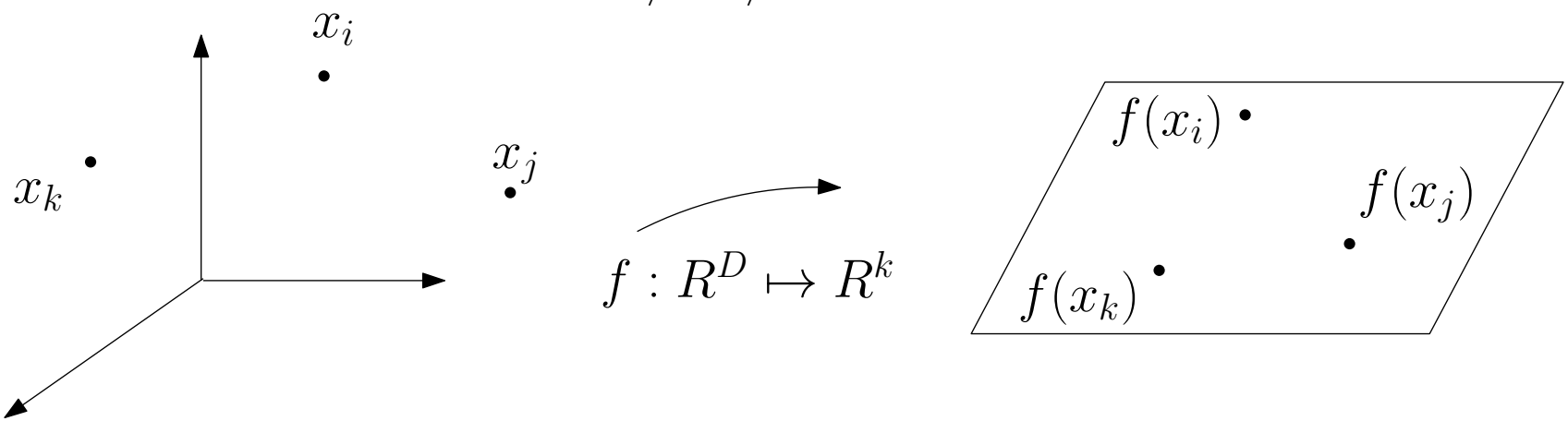
▷Ref: van der Maaten et al, Dimensionality reduction: a comparative review, 2009

▷Ref: J.A. Lee and M. Verleysen, Nonlinear dimensionality reduction, Springer, 2007

Dimensionality reduction

Johnson-Lindenstrauss lemma: no distance distorsion with high probability

- Embedding dimension k: $k = \frac{4 \ln n}{\varepsilon^2/2 - \varepsilon^3/3}$



$$\forall (i,j) : 1 - \varepsilon \leq \frac{||f(x_i) - f(x_j)||}{||x_i - x_j||} \leq 1 + \varepsilon$$

- Theory: see lecture #2

Dimensionality reduction

Data model in matrix form: notations

- Selected matrices.
  - Matrix 1_n: a $n \times 1$ column vector of n ones.
  - Matrix H_n: the following matrix of ones

$$H = 1_n {1_n}^{\mathsf{T}} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \vdots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \tag{1}$$

- Notations. We consider a $n \times d$ matrix whose
  - rows: individuals, $i = 1, \ldots, n$
  - columns: features, $j = 1, \ldots, d$

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{pmatrix} = \begin{pmatrix} \vdots & c_j & \vdots \end{pmatrix} = \begin{pmatrix} \cdots \\ x_i \\ \cdots \end{pmatrix} \tag{2}$$

The centroid

- Centroid – aka center of mass. The centroid or center of mass

$$\mu = \frac{1}{n}\sum_i X_i = \begin{pmatrix} \mu_i & \dots & \mu_d \end{pmatrix} \text{ with } \mu_j = \frac{1}{n}\sum_{i=1,\dots,n} x_{i,j} \tag{3}$$

One has the equivalently form:

$$\mu = \frac{1}{n} 1_n{}^{\top} X. \tag{4}$$

The data centered matrix is defined by,

$$X - \mu = \left(x_{i,j} - \mu_{j\cdot}\right) \tag{5}$$

or equivalently in matrix form

$$X - \mu = X - 1_n \mu = X - \frac{1}{n} H_n X. \tag{6}$$

- One property of the centroid. One has:

Lemma 1. Consider a point set $X_1, \dots, X_n$, and a point x. Its centroid μ minimizes the sum of squared distances to all points.

Proof.
Expand $\sum_i \|X_i - X\|^2 = \sum_i \|X_i - \mu + \mu - X\|^2$

Intermezzo: k-means (k-means++) and variants

- k-means: uses the center of mass, aka centroid
- Using the sum of squared distances to data points:
  - k-means: the center of a cluster is its centroid.
  - k-medoids: the center of a cluster must be a data point.
  - k-medians: the center of a cluster is a geometric median of the points – requires a notion of median in d-dimensions (e.g. based on depth)
- Using the sum of distances to data points:
  - point minimizing the sum of distances: the Fermat–Weber point.
  - sample point minimizing the sum of distances.
- Nb: in general, difficult (NP-hard) optimization problems

The Covariance Matrix

The covariance[1] of two features is defined by

$$\mathrm{Cov}(C_j, C_k) = \frac{1}{n-1} \sum_{i=1,\ldots,n} (x_{ij} - \mu_i)(x_{ik} - \mu_k). \tag{7}$$

Arranging these into a matrix yields the $d \times d$ covariance matrix:

$$C = \frac{1}{n-1}(X - \mu)^{\mathsf{T}}(X - \mu). \tag{8}$$

Lemma 2. One has

$$C = \frac{1}{n-1} X^{\mathsf{T}} X - \mu^{\mathsf{T}} \mu. \tag{9}$$

[1] Note the division by $n-1$ and not n: this is the so-called Bessel correction, which aims at ensuring that the estimator has no bias; this is related to the fact that in computing the variance, there are $n-1$ independent residuals $x_i - \overline{X}$, since all residuals add up to 0.

The Gram matrix

The Gram matrix is the $n \times n$ matrix defined by

$$G = XX^{\mathsf{T}} = \left(g_{i,j}\right), \text{ with } g_{i,j} = < X_i, X_j > = X_i {X_j}^{\mathsf{T}}. \tag{10}$$

As we shall see below, it is convenient to work with the Gram matrix of the centered data.

$$G^{\star} = (X - \mu)(X - \mu)^{\mathsf{T}}. \tag{11}$$

Squared Distance Matrix and Gram matrices

- Squared distance matrix D: the $n \times n$ matrix defined by

$$D = \left(d_{i,j}^2\right), \text{ with } d_{i,j}^2 = \left\|X_i - X_j\right\|^2 = g_{i,i} + g_{j,j} - 2g_{i,j}. \tag{12}$$

- For centered data:

Lemma 3. For centered data, the Gram matrix and the squared distance matrix satisfy:

$$G = -\frac{1}{2} KDK, \text{ with } K_{ij} = \delta_{ij} - \frac{1}{n}. \tag{13}$$

- General case:

Lemma 4. The Gram matrix of the centered data and the squared distance matrix satisfy:

$$G^\star = -\frac{1}{2}\left(D - \frac{1}{n}DH - \frac{1}{n}HD + \frac{1}{n^2}HDH.\right) \tag{14}$$

Dimensionality reduction

Matrices and matrix norms

Definition 5. Let A be a $m \times n$ real matrix. A matrix norm is a function $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ such that the following three properties hold:

- (i) $f(A) \geq 0$,
- (ii) $f(A + B) \leq f(A) + f(B)$,
- (iii) $f(\alpha A) = \mid \alpha \mid f(A), \alpha \in \mathbb{R}$.

▷ The Frobenius norm.

$$\|A\|_F = \sqrt{\sum_{i,j} \mid a_{ij} \mid^2}. \tag{15}$$

Definition 6. (p-norms) Defined from the p-norms in the vector spaces associated with the linear map encoded by matrix A:

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p = 1} \|Ax\|_p . \tag{16}$$

Definition 7. (Subordinate norm) Consider (i) $A \in \mathbb{R}^{m \times n}$, (ii) $\|\cdot\|_\alpha$ a norm on $\mathbb{R}^n$, (iii) $\|\cdot\|_\beta$ a norm on $\mathbb{R}^m$ and define the *subordinate* norm

$$\|A\|_{\alpha,\beta} = \sup_{x \neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha}. \tag{17}$$

The Singular Value Decomposition

Definition 9. An SVD for a $m \times n$ real valued matrix A is a decomposition

$$A_{m\times n} = V_{m\times m} S_{m\times n} {U_{n\times n}}^{\mathsf{T}} \tag{19}$$

With

$$UU^{\mathsf{T}} = \mathrm{I}_n, i.e., U \text{ orthogonal matrix}, \tag{20}$$

$$VV^{\mathsf{T}} = \mathrm{I}_m, i.e., V \text{ orthogonal matrix}. \tag{21}$$

▷ Properties: one has [1, Thm. 13.6]:

- Matrix S is diagonal; its entries are the so-called singular values.
- The columns of U are the eigenvectors of $A^{\mathsf{T}}A$.
- The columns of V are the eigenvectors of AA^{T}.
- If the singular values are distinct, the SVD is unique–up to the same permutation of the columns of U, V and S.

From Eq. 19, one gets

$$a_{ij} = \sum_{k=1,\ldots,n} \sigma_{kk} v_{ik} u_{jk}. \tag{22}$$

▷ Rmk. The singular values are unchanged upon transposing matrix A:

$$A_{m\times n} = V_{m\times m} S_{m\times n} {U_{n\times n}}^{\mathsf{T}} \tag{23}$$

$$A_{n\times m} = U_{n\times n} S_{n\times m} {V_{m\times m}}^{\mathsf{T}} \tag{24}$$

The Covariance Matrix – again

Recall that an orthogonal matrix P is a matrix such that $PP^{\mathsf{T}} = \mathrm{I}$. Recall also the following spectral theorem [1, Chapter 12]:

Theorem 10. For every $d \times d$ real symmetric matrix A, there is an orthogonal matrix P and a diagonal matrix $D = diag(\lambda_i), i = 1, \ldots, d, \lambda_i \in \mathbb{R}$ such that

$$A = PDP^{\mathsf{T}}. \tag{25}$$

Let us now process the covariance matrix with the SVD:

$$C = \frac{(X-\mu)^{\mathsf{T}}}{\sqrt{n-1}} \frac{X-\mu}{\sqrt{n-1}}. \tag{26}$$

Plugging the following SVD $\frac{X-\mu}{\sqrt{n-1}} = VSU^{\mathsf{T}}$ into the previous equation yields:

$$C = US^{\mathsf{T}}SU^{\mathsf{T}}. \tag{27}$$

On the other hand, from the spectral Thm:

$$C = PDP^{\mathsf{T}}. \tag{28}$$

Comparing both:

- **The squared singular values are the eigenvalues of C.**
- **The columns of U are the eigenvectors of C.**

SVD and matrix approximation

Main refs: [2, 1]

Theorem 11. Let A be an $m \times n$ matrix of rank r, and let $A = VSU^{\mathsf{T}}$ be an SVD for A. Denote $\sigma_1 \geq \cdots \geq \sigma_p$ the singular of A, with $p = \min(m, n)$, and let u_i and v_i the columns of U and V, respectively.
The best rank $k < r$ approximation of A, in the $\|\cdot\|_2$ sense, is given by

$$A_k = \sum_{i=1,\ldots,k} \sigma_i v_i u_i^{\mathsf{T}} = V \mathrm{diag}(\sigma_1, \ldots, \sigma_k) U^{\mathsf{T}}. \tag{29}$$

and one has $\|A - A_k\|_2 = \sigma_{k+1}$.

Theorem 12. (Rayleigh-Ritz) Let A be a symmetric $d \times d$ matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$, and let $(u_1, \ldots, u_d)$ be any orthonormal basis of eigenvectors of A, where u_i is associated with λ_i. Then

$$\max_{x \neq 0} \frac{x^{\mathsf{T}} A x}{x^{\mathsf{T}} x} = \lambda_1, \tag{30}$$

and this maximum is attained for $x = u_1$. Also, working in the complementary space

$$\max_{x \neq 0, x \in \{u_1, \ldots, u_k\}^{\perp}} \frac{x^{\mathsf{T}} A x}{x^{\mathsf{T}} x} = \lambda_{k+1}, \tag{31}$$

with the maximum attained for $x = u_{k+1}$, where $1 \leq k \leq d - 1$.

Dimensionality reduction

Principal components analysis: rationale

- Projecting points on a vector: $(X - \mu)v$ is a $(n \times d) \times (d \times 1) = (n \times 1)$ vector
- PCA: main idea
  - find an orthonormal basis $\{v_h\}$,
  - such that the variance of the $Y_h = (X - \mu)v_h$ is maximized,

We formalize as follows:

Definition 13. Let X be a $n \times d$ data matrix, and μ the associated center of mass. Consider a family $\{v_h\}, 1 \leq h \leq k \leq d$ of mutually orthogonal unit vectors from S^{d-1}. Define the associated centered points

$$Y_h = (X - \mu)v_h. \tag{32}$$

These centered points define *principal components* provided that the following conditions are met:

- $\mathrm{Var}[Y_h]$ is maximized.
- $\mathrm{Cov}(Y_h, Y_{h+1}) = 0$.

PCA: main theorem

Theorem 14. Consider an SVD decomposition of the centered data matrix, i.e. $X - \mu = VSU^{\mathsf{T}}$, and let $\sigma_1 \geq \cdots \geq \sigma_d$ the associated singular values.
The principal components of X are the centered points

$$Y_h = (X - \mu)u_h, \tag{33}$$

with $\{u_k\}$ the eigenvectors of U (ie the columns of U), and one has

$$\mathrm{Var}\,[Y_h] = \frac{\sigma_h^2}{n-1}. \tag{34}$$

Algorithm 1 Algorithm for PCA.
Alternative to last step: diagonalize the covariance matrix.

Compute the centered data matrix $(X - \mu)$
Compute its SVD $(X - \mu) = VSU^{\mathsf{T}}$
Compute the centered points $E = (X - \mu)U$
Possibly compute a lower dimensional embedding $R = (X - \mu)U\mathrm{I}_{d\times k}$
{//Dimension-wise: $(n \times d)(d \times d)(d \times k)$}

PCA: two steps of the proof

• Variance and covariance of two centered points.
Consider a centered point along a unit direction v: $Y = (X - \mu)v \in \mathbb{R}^d$. The variance of Y satisfies:

$$\mathrm{Var}\,[Y] = \frac{1}{n-1}((X-\mu)v)^{\mathsf{T}}(X-\mu)v = \frac{1}{n-1}v^{\mathsf{T}}(X-\mu)^{\mathsf{T}}(X-\mu)v. \tag{35}$$

Likewise, the covariance of two centered points along unit directions v and w $Y_h = (X - \mu)v$ and $Y' = (X - \mu)w$ satisfy

$$\mathrm{Cov}(Y, Y') = \frac{1}{n-1}v^{\mathsf{T}}(X-\mu)^{\mathsf{T}}(X-\mu)w. \tag{36}$$

• First principal directions. Maximizing the variance of Eq. (35) is equivalent to maximizing

$$v^{\mathsf{T}}\frac{1}{n-1}(X-\mu)^{\mathsf{T}}(X-\mu)v. \tag{37}$$

By the Rayleigh-Ritz Thm (Thm. 12): max eigenvalue of $\frac{1}{n-1}(X-\mu)^{\mathsf{T}}(X-\mu)$, namely $\sigma_1^2/(n-1)$. Using the associated eigenvector u_1, we get the first reduced point

$$Y_1 = (X - \mu)u_1. \tag{38}$$

• Remaining principal directions. One uses the second part of the Rayleigh-Ritz theorem, observing also that the column vectors of U are mutually orthogonal.

PCA: practical matters

- Some guidelines:
  - In choosing : always report the residual variance on the principal directions discarded
  - In case the point cloud does not have homogeneous dimension: also perform local PCA – cf the *local covariance dimension* seen in Lecture #2

Dimensionality reduction

Multi-dimensional scaling (MDS): rationale

▷ MDS: find coordinates from distance matrix or Gram matrix; reduce dimensionality

$$G^* = (X - \mu)(X - \mu)^\mathsf{T} \tag{39}$$

$$= (VSU^\mathsf{T})(VSU^\mathsf{T})^\mathsf{T} = VS^2V^\mathsf{T} = (VS)(VS)^\mathsf{T}. \tag{40}$$

▷ Associated embedding: the so-called **realizing coordinates** for the centered data:

$$\hat{X}_c = VS \tag{41}$$

▷ MDS and approximation: upon sorting the eigenvalues (or singular values) of G^*, consider the matrix defined from the first k rows of V, that is:

$$G_k^* = \sum_{i=1,\ldots,k} \sigma_i v_i v_i^\mathsf{T} = V\mathrm{diag}(\sigma_1, \ldots, \sigma_k)V^\mathsf{T}. \tag{42}$$

This matrix is the best approximation for the matrix 2-norm of G^* (by Thm 11), and

$$\|G - G_k\|_2 = \sigma_{k+1}. \tag{43}$$

Gram and PCA yield identical embeddings

- Gram, *realizing coordinates*: using the SVD of $X - \mu$ yields

$$G^* = (X - \mu)(X - \mu)^\mathsf{T} = (VSU^\mathsf{T})(VSU^\mathsf{T})^\mathsf{T} = VS^2V^\mathsf{T} = (VS)(VS)^\mathsf{T}. \quad (44)$$

whence the realizing coordinates

$$\hat{X}_c = VS \quad (45)$$

- PCA, centered points: also using $X - \mu = VSU^\mathsf{T}$:

$$Y = (X - \mu)U = VSU^\mathsf{T}U = VS. \quad (46)$$

Python code: PCA with eigen decomposition

```
def pca_with_eigen_decomposition(X):
  n, d = X.shape

  # check X centered
  assert np.allclose(X.mean(axis=0), np.zeros(d))

  # Covariance matrix
  C = np.dot(X.T, X) / (n-1)

  # Eigen decomp.
  eigen_vals, eigen_vecs = np.linalg.eig(C)

  X_pca = np.dot(X, eigen_vecs) # project onto PC space
  return X_pca
```

Python code: PCA with SVD

```
def pca_with_svd(X):
    n, d = X.shape

    # Compute full SVD
    U, Sigma, Vh = np.linalg.svd(X,
        full_matrices=False,
        compute_uv=True)

    # Transform X with SVD components
    X_svd = np.dot(U, np.diag(Sigma))
    return X_svd
```

Python code: MDS

```
def mds(X):
    n, d = X.shape

    # Gram matrix and Eigen decomposition
    G = np.dot(X, X.T)
    eigen_vals, eigen_vecs = np.linalg.eig(G)

    # Embedding
    Y = np.dot(eigen_vecs, np.diag( np.sqrt(eigen_vals)))
    return Y
```

Dimensionality reduction

ISOMAP: rationale

▷ ISOMAP: distance MDS with *geodesic distances*

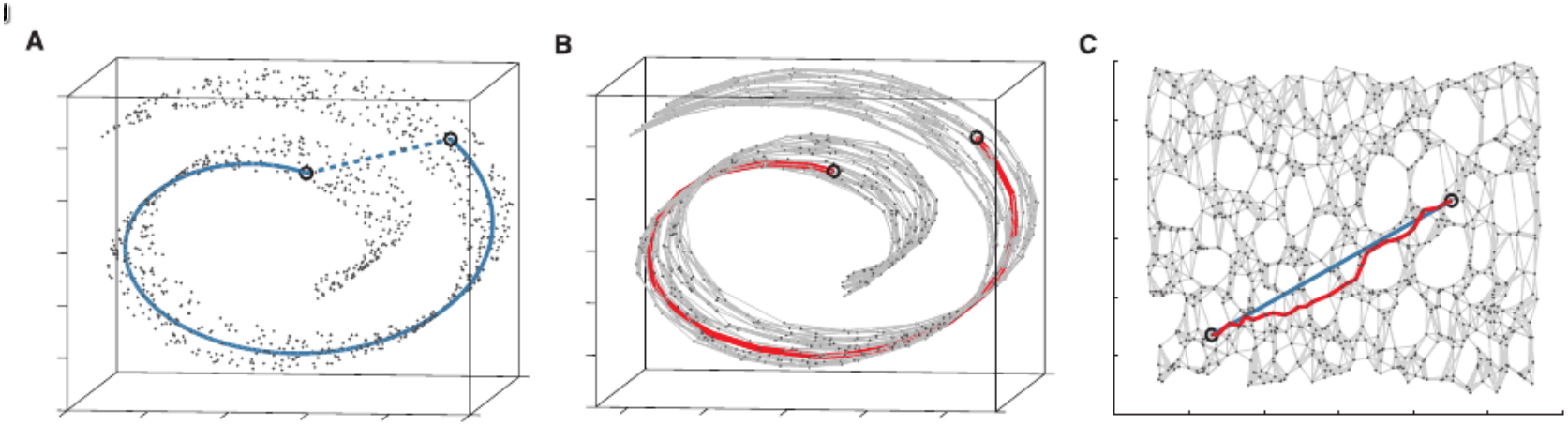


Figure: ISOMAP: **illustration** From [3].

- Nearest neighbors: avoid short-cut across the ambient space
- Sensitivity to the parameter ε controlling neighborhoods
- Nb: geodesic on the swiss roll $\Leftrightarrow$ line in the plane

ISOMAP: algorithm

Algorithm 2 Algorithm ISOMAP, from [3].

Compute a nearest neighbor graph on the data – connect points i, j such that $d_{ij} \leq \varepsilon$
Compute the matrix D of geodesic distances between all pairs of points – Floyd's algorithm
Compute the Gram matrix of centered data $G^\star$ from the squared distance matrix
Apply MDS

ISOMAP: data centering

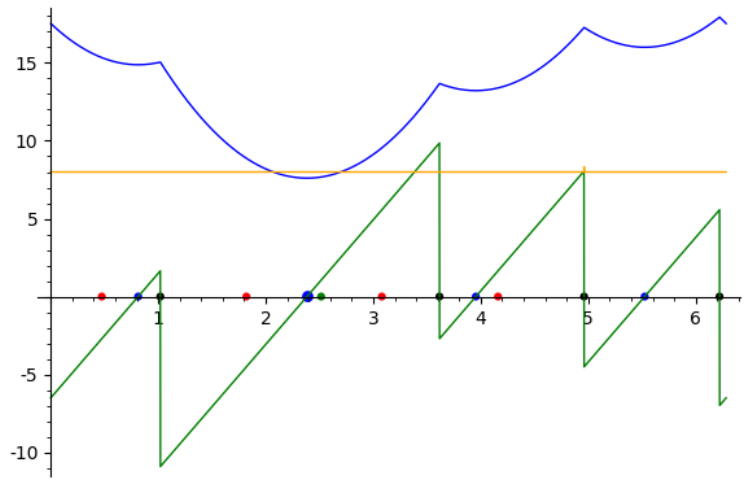
- Sampling on S^1: consider n angles $\Theta_0 = \{\theta_i\}_{i=1,\dots,n}$.
- Distance function:

$$F_p(\theta) = \sum_{i=1,\dots,n} w_i f_i(\theta), \text{ with } f_i(\theta) = d^p(X(\theta), X(\theta_i)). \tag{47}$$

- Center of mass on the unit circle: for $p = 2$, consider the min. of the function:

$$\theta^* = \arg \min_{\theta \in [0,2\pi)} F_p(\theta). \tag{48}$$

- Fréchet mean of four points on S^1:

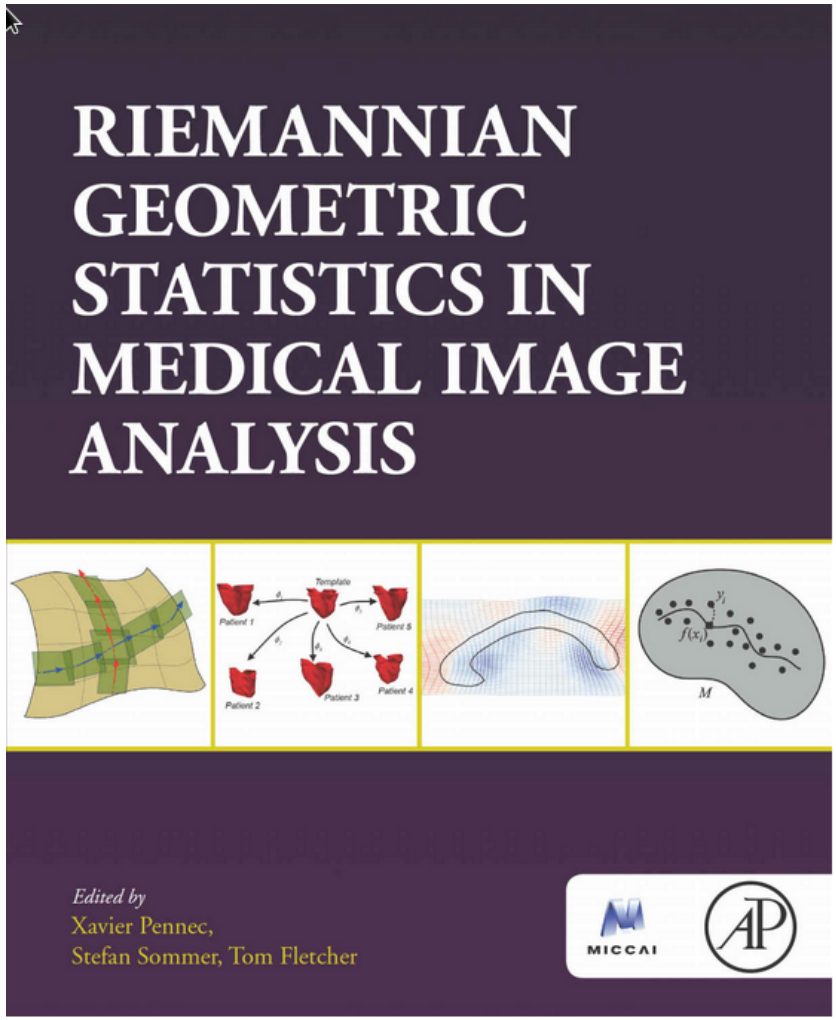


- Functions: blue: function F_2; green: derivative F_2'; orange: second derivative F_2''
- Points: red bullets: data points; black bullets: antipodal points; blue bullets: local minima of the function; large blue bullet: Fréchet mean θ^*; green bullet: circular mean.

- Thm.: computing the Fréchet mean is decidable (due to Lindemann's theorem on the transcendence of π) and has $\tilde{O}(n \log n)$ complexity.

- Ref: Fréchet mean on the unit circle, O'Donnell - Cazals, 2021

ISOMAP and PCA in Riemannian geometry



▷Ref: Riemannian Geometric Statistics in Medical Image Analysis; Pennec, Sommer, Fletcher, 2019

Dimensionality reduction

Locally Linear Embedding

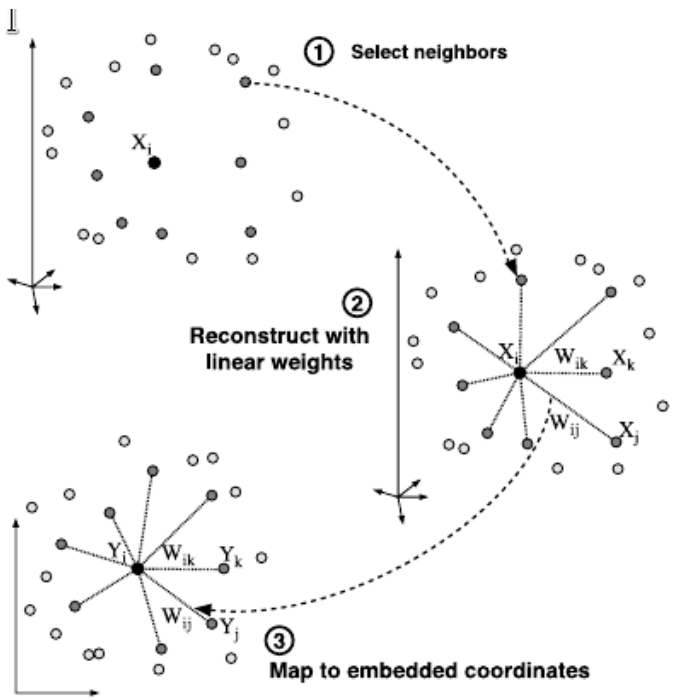
- (NNG) Compute a nearest neighbor graph in $\mathbb{R}^D$
- (Local reconstruction) Compute weights to locally reconstruct x_i from its neighbors, Eq. 49
- (Embedding) Use the weights to find a mapping into $\mathbb{R}^d$, minimizing Eq. 50

▷ Local reconstruction in $\mathbb{R}^D$:

$$\varepsilon(W) = \sum_i \left\| x_i - \sum_j w_{ij} x_j \right\|^2 \quad (49)$$

▷ Embedding into $\mathbb{R}^d$:

$$\sum_i \left\| y_i - \sum_j w_{ij} y_j \right\|^2 \quad (50)$$



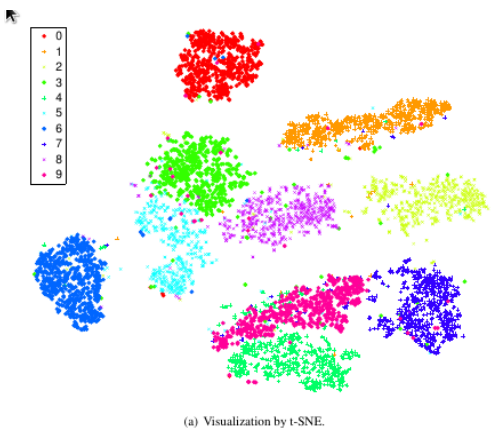
▷Ref: Nonlinear Dimensionality Reduction by LLE, Science, 2020 [4]

Dimensionality reduction

SNE : motivation

- Two algorithms:
  - `SNE` : Stochastic Neighbor Embedding
  - `t-SNE` : t-SNE, with t from Student-t
- Overview:
  - Input: point set $\{x_i\} \in \mathbb{R}^D$
  - Output: point set $\{y_i\} \in \mathbb{R}^d$, with $d = 2$ or $d = 3$ (visualization)
  - Rationale: conserve pairwise distances: $D_{ij}^2 \sim d_{ij}^2$
  - Howto: convert the point clouds into probability distributions (whence Stochastic), using Euclidean distances – one point becomes one distribution:

$$x_i \leftrightarrow P_i = \{p_{i|j}\}; y_i \leftrightarrow Q_i = \{q_{i|j}\}$$



(a) Visualization by t-SNE.

SNE : pairwise distances versus proba. distributions

- In $\mathbb{R}^D$: proba. distribution $P_i = \{p_{i|j}\}$ for x_i: with $D_{ij} = \|x_i - x_j\|$

$$p_{i|j} = \frac{\exp(-D_{ij}^2/2\sigma_i)}{\sum_{k\neq i}\exp(-D_{ik}^2/2\sigma_i)} \tag{51}$$

- In $\mathbb{R}^d$: proba. distribution $Q_i = \{q_{i|j}\}$ for y_i: with $d_{ij} = \|y_i - y_j\|$

$$q_{i|j} = \frac{\exp(-d_{ij}^2)}{\sum_{k\neq i}\exp(-d_{ik}^2)} (\text{Nb, bandwidth:} 1/\sqrt{2}) \tag{52}$$

- Comparing the two distributions $P = \{P_i\}$ and $Q = \{Q_i\}$: via Kullback-Leibler D.:

$$\text{Cost } C = D_{\mathrm{KL}}(P\|Q) = \sum_i D_{\mathrm{KL}}(P_i\|Q_i) = \sum_{ij} p_{i|j}\log\frac{p_{i|j}}{q_{i|j}} \tag{53}$$

- Remarks:
  - Lack of symmetry for $p_{i|j}$ and $q_{i|j}$
  - In KL: small $q_{i|j}$ for large $p_{i|j}$: large penalty

SNE : choice of the bandwidth σ_i via perplexity

- Recall the def.:

$$p_{i|j} = \frac{\exp(-D_{ij}^2/2\sigma_i)}{\sum_{k \neq i} \exp(-D_{ik}^2/2\sigma_i)} \tag{54}$$

- Entropy for P_i:
  - $H(P_i) = -\sum_j p_{i|j} \log_2 p_{i|j}$
  - Nb: $\sigma_i \nearrow \Rightarrow H(P_i) \nearrow$ since conditional probas are more uniform
- Perplexity for P_i associated with p_i:
  - $\text{Perp}(P_i) = 2^{H(P_i)}$
  - Intuition: effective number of neighbors
- Observation: SNE is robust in changes of the perplexity (values in the range 5..50), which makes the choice of σ_i relatively easy

SNE : global optimization – cost function

- Cost: (non convex functional) $C = D_{\mathrm{KL}}(P||Q)$
- Gradient of the cost wrt the projected points y_i:

$$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_j - y_i) \tag{55}$$

- Nb for colored terms: mismatches ... since we aim at $D_{ij} \sim d_{ij}$ and $P_i \sim Q_i$
- Initiation of the solution: $\mathcal{Y}^{(0)} = \{y_1^{(0)}, \ldots, y_n^{(0)}\}$
  - n points drawn from an isotropic Gaussian centered at the origin (plus some Gaussian noise, at least at early stages)
- Iterative solution via gradient descent:

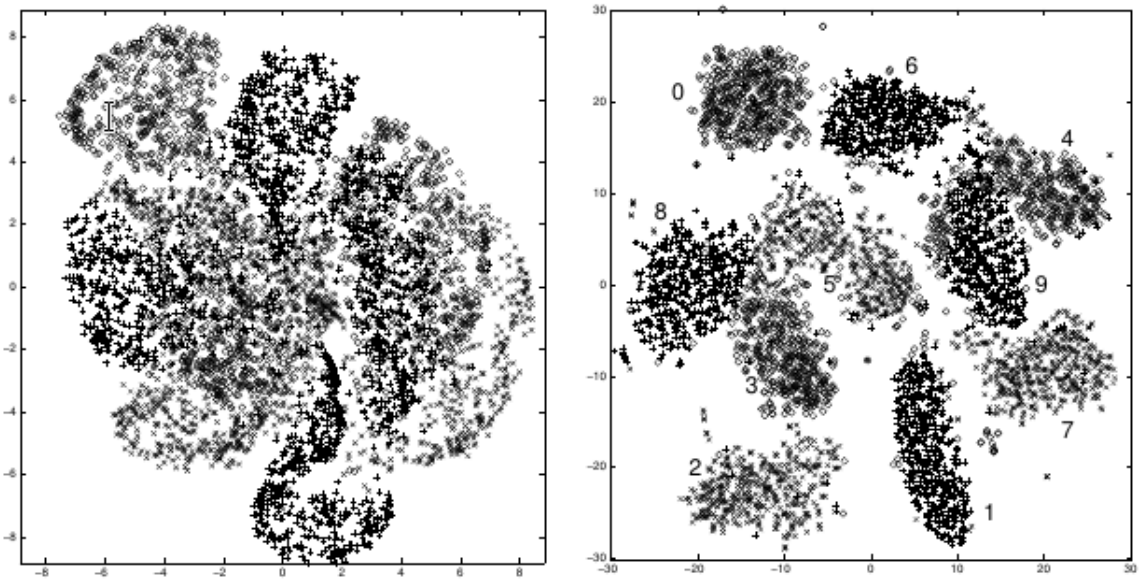
$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t)} + \eta\frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}) \tag{56}$$

with

- η: learning rate
- $\alpha(t)$: momentum at iteration t

From SNE to t-SNE

- Cost function in SNE : hard to optimize (non convex, complex gradient)
- SNE suffers from the so-called *crowding problem:* consider two shells (region between two balls) centered at $x_i \in \mathbb{R}^D$: in projecting from $\mathbb{R}^D$ to $\mathbb{R}^d$ (with $d = 2, 3$), there not enough space to accommodate all points



Digits projected into 2D **(Left)** SNE **(Right)** pre-t-SNE
Nb: note the scales on the two axis

Symmetric SNE

- Natural choice in $\mathbb{R}^d$: proba. distribution $Q_i = \{q_{ij}\}$:

$$q_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq l} \exp(-d_{kl}^2)} \text{ (Nb) quadratic \# terms} \tag{57}$$

- Natural choice in $\mathbb{R}^D$: proba. distribution $P_i = \{p_{ij}\}$:

$$p_{ij} = \frac{\exp(-D_{ij}^2/2\sigma)}{\sum_{k \neq l} \exp(-D_{kl}^2/2\sigma)} \text{ (Nb) quadratic \# terms} \tag{58}$$

- However: the latter is not good enough: for an outlier x_i, p_{ij} very small
- In $\mathbb{R}^D$: proba. distribution $P_i = \{p_{ij}\}$:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}. \tag{59}$$

Guarantee: $\sum_j p_{ij} > 1/2n$.

- Nb: still requires the choice of bandwidths σ – cf perplexity

New cost and its gradient

- Cost:

$$\text{Cost } C = D_{\mathrm{KL}}(P\|Q) = \sum_i D_{\mathrm{KL}}(P_i\|Q_i) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{60}$$

- Associated gradient:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_j - y_i) \tag{61}$$

Symmetry is not enough: mismatched tails for mismatched dimensions

▷ Crowding effect: volumes in high dim and low dim are not consistent

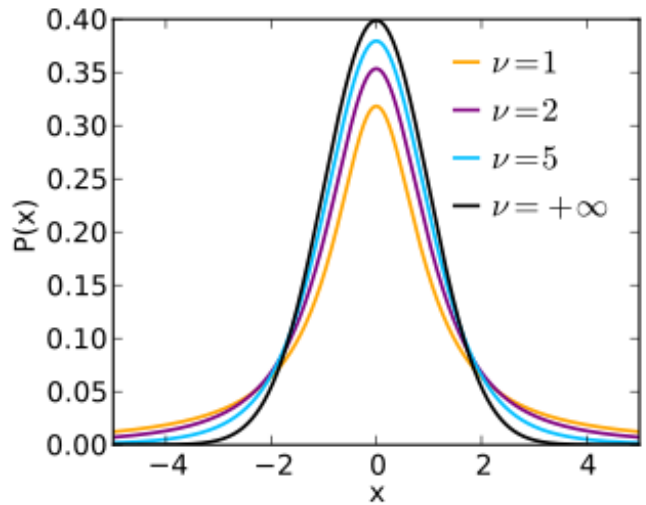
▷ Geometry to proba. distributions: using Gaussian functions to convert distances into distributions both for $\mathbb{R}^D$ and $\mathbb{R}^d$ maintains the problem

▷ Solution: use

- $\mathbb{R}^D$: Gaussian weights (light tail)
- $\mathbb{R}^d$: Student-t weights (heavy tail)

▷ Student-t with 1 d.o.f.

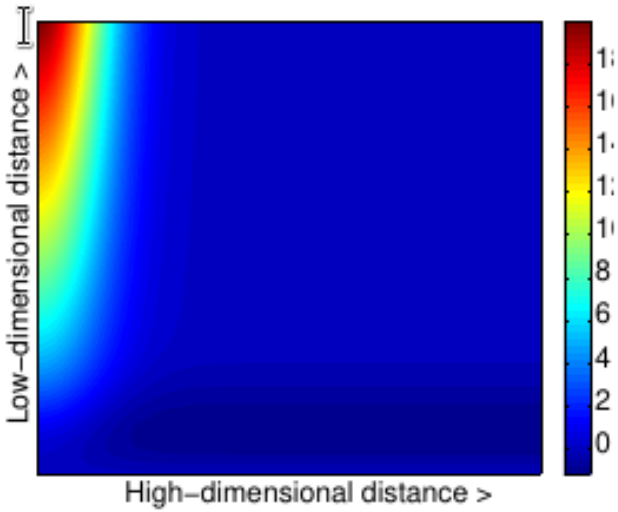
$$q_{ij} = \frac{(1 + \|y_i - y_j\|)^{-1}}{\sum_{k \neq l}(1 + \|y_i - y_j\|^{-1})} \quad (62)$$



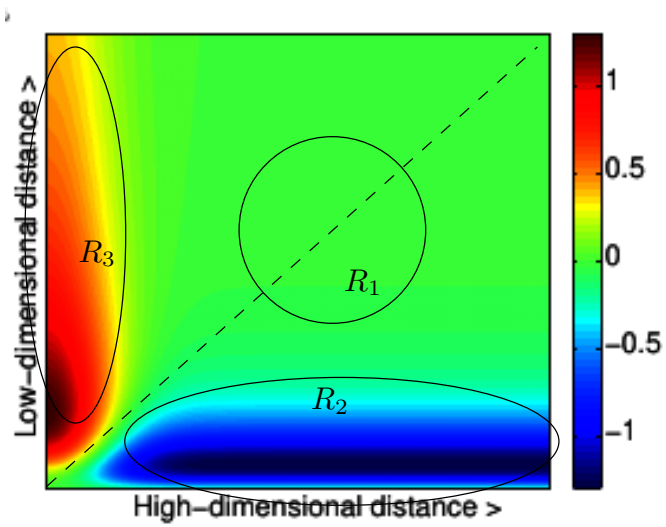
▷ Student-t, wikipedia:
`https://en.wikipedia.org/wiki/Student%27s_t-distribution`

SNE and t-SNE : comparison of gradients

▷ Gradients as a function of the pairwise distances $D_{ij} \times d_{ij}$: red:attraction (positive); blue: repulsion (negative)



SNE



t-SNE

- R_2: t-SNE repels points distant in $\mathbb{R}^D$ but close in $\mathbb{R}^d$. Much more specific than SNE .
- R_3: t-SNE attract points close in $\mathbb{R}^D$ but far apart in $\mathbb{R}^d$. More homogeneous than SNE .
- R_1: t-SNE relatively neutral for all other pairs, which is not the case of SNE .

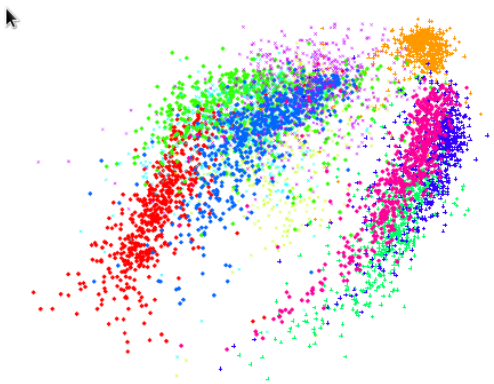
t-SNE : algorithm

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=*1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

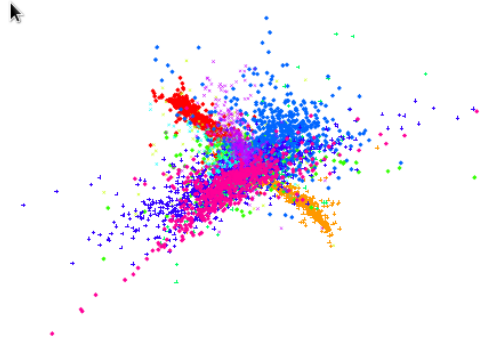
Results on the MNIST Database

- MNIST dataset: total of 60,000 + 10,000 handwritten digits; `http://yann.lecun.com/exdb/mnist/`
- 6000 random handwritten digits



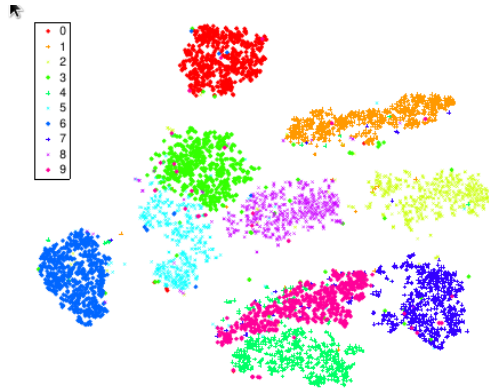
(a) Visualization by Isomap.

ISOMAP



(b) Visualization by LLE.

LLE



(a) Visualization by t-SNE.

t-SNE

Discussion and comparison to contenders

- Pros:
  - vs PCA: `t-SNE` is non linear.
  - vs MDS: MDS favors long distances; here, short and long distances on equal footing.
  - vs Isomap: no short-circuiting problem; similarly to MDS, Isomap favors long (geodesic) distances.
  - vs LLE: LLE preserves the covariance matrix (in low dim); can be achieved by collapsing points + outliers. does not happen in `t-SNE` .
- Cons:
  - Visualization – no quantitative assessment on the dimension. Also, what if $d > 3$?
  - Quadratic cost – accelerations needed
  - Local linearity assumption used: Euclidean distances used in weights
  - Cost C is non convex; parameter tuning involved (η, $\alpha(t)$)

▷Ref: Visualizing data using t-SNE, van der Maaten and Hinto, 2008 [5]
▷Ref: Accelerating t-SNE using tree-based algorithms, van der Maaten, 2014 [6]

Dimensionality reduction

Graph Laplacians, random walks on graphs, diffusion maps: a primer

Frederic.Cazals@inria.fr

Graph and associated point set

▷ Weighted graph: we consider a weighted graph G whose nodes are index from $1, \ldots, n$. The set of edges defined nodes which are connected that is $i \sim j$. The weights are ≥ 0 and symmetric, that is

$$w_{ij} \geq 0, w_{ij} = w_{ji}; W = (w_{ij})_{i,j=1,\ldots,n}. \tag{63}$$

▷ Geometric realization:

- nodes are associated to a point set $\{x_i\}_{i=1,\ldots,n}$,
- weights are typically given by a kernel, e.g. a Gaussian kernel – for some $\varepsilon > 0$:

$$k(x_i, x_j) = exp\big(-\frac{\|x_i - x_j\|^2}{\varepsilon}\big). \tag{64}$$

Laplacian and normalized Laplacian

- Graph: node degree and volume

$$d_i = \sum_j w_{ij}, \mathrm{Vol}(G) = \sum_i d_i; D = \mathrm{Diag}(\{d_i\}). \tag{65}$$

- Laplacian matrix:

$$L = D - W = \begin{cases} d_i - w_{ii} & \text{diagonal term} \\ -w_{ij} & \text{off diagonal and } i \sim j \\ 0 & \text{off diagonal and } i \not\sim j. \end{cases} \tag{66}$$

- Normalized Laplacian matrix – a symmetric matrix:

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = \begin{cases} 1 - \frac{w_{ii}}{d_i} & \text{diagonal term} \\ -\frac{w_{ij}}{\sqrt{d_i d_j}} & \text{off diagonal and} i \sim j \\ 0 & \text{off diagonal and} i \not\sim j. \end{cases} \tag{67}$$

Random walk on a graph

▷ Random walk on G: modeled as a Markov process x_t

$$\mathbb{P}\left[x_{t+1} = j \mid x_t = i\right]. \tag{68}$$

▷ Transitions: defined by the matrix

$$p_{ij} = \frac{w_{ij}}{d_i}, P = (p_{ij})_{i,j=1,\ldots,n} = D^{-1}W. \tag{69}$$

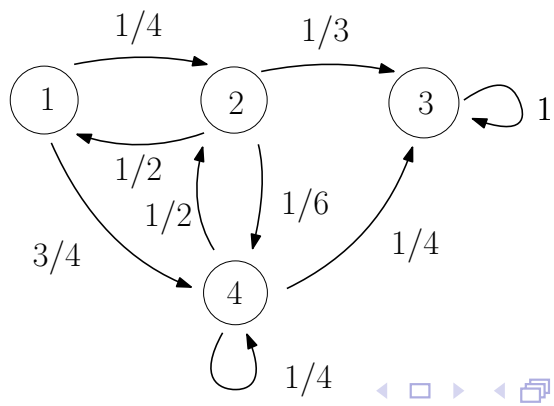
▷ Matrix form: consider a row vector f of probabilities to be on the n vertices of the graph. Upon applying one step of the random walk, the new occupancy probabilities are given by

$$f\ P. \tag{70}$$

▷ Trivial observations:

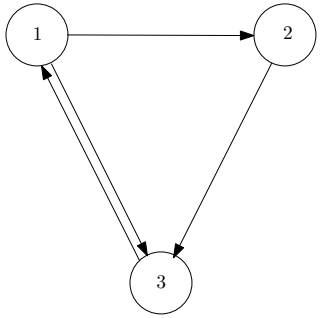
- Matrix P is row stochastic that is ${1_n}^{\mathsf{T}} P = 1_n$.
- Matrix P is not symmetric unless the graph is regular that is $d_i = \text{constant}$

▷ Graph and associated stochastic matrix P



$$P = \begin{bmatrix} 0 & 1/4 & 0 & 3/4 \\ 1/2 & 0 & 1/3 & 1/6 \\ 0 & 0 & 1 & 0 \\ 0 & 1/2 & 1/4 & 1/4 \end{bmatrix}$$

Intermezzo: Google page rank



Three web pages

Sergei Brin - Larry Page

"PageRank can be thought of as a model of user behavior. We assume there is a "random surfer" who is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank. And, the d damping factor is the probability at each page the "random surfer" will get bored and request another random page. One important variation is to only add the damping factor d to a single page, or a group of pages. This allows for personalization and can make it nearly impossible to deliberately mislead the system in order to get a higher ranking."

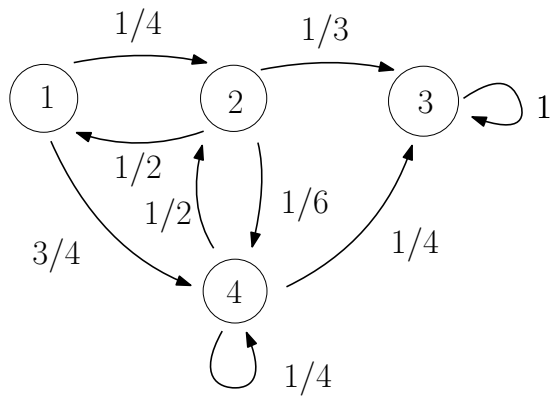
▷ The Google matrix:

$$G = dA + (1-d)E$$
$$= d \begin{pmatrix} 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + (1-d) \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$

`http://infolab.stanford.edu/~backrub/google.html`

Random walk on a graph

Graph and associated stochastic matrix P



$$P = \begin{bmatrix} 0 & 1/4 & 0 & 3/4 \\ 1/2 & 0 & 1/3 & 1/6 \\ 0 & 0 & 1 & 0 \\ 0 & 1/2 & 1/4 & 1/4 \end{bmatrix}$$

▷ Random walk on G: modeled as a Markov process x_t

$$\mathbb{P}\left[x_{t+1} = j \mid x_t = i\right]. \tag{71}$$

▷ Transitions: defined by the row-stochastic matrix P

▷ Iterating P: applying one step of the random walk to occupancy vector f yields the new occupancy probabilities $f\ P$.

▷ Stationary distribution, def: invariant occupancy probabilities i.e. $\pi P = \pi$.

▷ Thm. A Markov chain which is irreducible and aperiodic convergences to its stationary distribution.

Random walk: stationary distribution

▷ Stationary distribution, def: vector of occupancy probabilities that remain unchanged upon applying P, that is

$$\pi P = \pi. \tag{72}$$

▷ Stationary distribution, matrix form: π is given by the column vector

$$\pi = (\frac{d_i}{\mathrm{Vol}(G)})_{i=1,\dots,n} = \frac{1}{\mathrm{Vol}(G)} D\ 1_n. \tag{73}$$

Indeed, one has

$$\pi^{\mathsf{T}} P = \frac{1}{\mathrm{Vol}(G)} 1_n{}^{\mathsf{T}} D^{\mathsf{T}} D^{-1} W \tag{74}$$

$$= \frac{1}{\mathrm{Vol}(G)} (\cdots \sum_i w_{ij} \dots) \qquad = \frac{1}{\mathrm{Vol}(G)} (\cdots \sum_i w_{ji} \dots) = \pi^{\mathsf{T}}. \tag{75}$$

Transition matrix: a symmetric version

- Difficulty: P is not symmetric unless the graph is regular.
- Bringing it into a symmetric form:

$$P = D^{-1}W = D^{-1/2}(D^{-1/2}WD^{-1/2})D^{1/2} \tag{76}$$
$$= D^{-1}(D - L) \tag{77}$$
$$= D^{-1}(D - D^{1/2}\mathcal{L}D^{1/2}) \tag{78}$$
$$= \mathsf{I} - D^{-1/2}\mathcal{L}D^{1/2} = D^{-1/2}(\mathsf{I} - \mathcal{L})D^{1/2}. \tag{79}$$

- Key expression: P expressed via the symmetric matrix P_s

$$P = D^{-1/2}P_sD^{1/2}, \text{ with } P_s = D^{-1/2}WD^{-1/2} = \mathsf{I} - \mathcal{L}. \tag{80}$$

- Eigenwork. Matrix P_s being symmetric, it can be diagonalized is an orthonormal basis $V = \{v_j\}$:

$$P_s = V\Lambda V^{\mathsf{T}}, \tag{81}$$

from which we get

$$P = D^{-1/2}P_sD^{1/2} = (D^{-1/2}V)\Lambda(V^{\mathsf{T}}D^{1/2}) = (D^{-1/2}V)\Lambda(D^{1/2}V)^{\mathsf{T}} \equiv \Psi\Lambda\Phi^{\mathsf{T}}, \tag{82}$$

with

$$\Psi = D^{-1/2}V = (\psi_1, \ldots, \psi_n), \text{ and } \Phi = D^{1/2}V = (\phi_1, \ldots, \phi_n). \tag{83}$$

Random walk: the spectral expansion

- Random walk iteration: matrix form after t steps

$$P^t = D^{-1/2} P_s^t D^{1/2} \tag{84}$$

But

$$P_s^t = \sum_i \lambda_i^t v_i v_i^{\mathsf{T}} \tag{85}$$

Whence

$$P^t = \sum_i \lambda_i D^{-1/2} v_i v_i^{\mathsf{T}} D^{1/2} = \sum_i \lambda_i D^{-1/2} v_i (D^{1/2} v_i)^{\mathsf{T}} \tag{86}$$

$$= \sum_i \lambda_i^t \psi_i \phi_i^{\mathsf{T}}. \tag{87}$$

- Application to a connected graph – cf the Perron–Frobenius theorem:

Theorem 15. For a connected graph G, the

$$\lambda_0 = 1 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-1} = 0. \tag{88}$$

Moreover, the stationary distribution π is the eigenvector ϕ_0.

In this case, the matrix P_s^t of Eq. (85) is the $n \times n$ matrix defined by

$$P_s^t = 1_n \phi_0^{\mathsf{T}} + \sum_{j \geq 1} \lambda_j^t \psi_j \phi_j^{\mathsf{T}}. \tag{89}$$

Diffusion maps: definition and probability distribution

▷ *k*-dimensional approximation. due to the decay of eivenvalues, focusing on the top k ones yields an embedding of points in dimension k. One defines:

Definition 16. The order $1 \leq k \leq n-1$ diffusion map is defined via the embedding of point x_j in the space of the first k eigenvectors:

$$\Psi_t(j) = (\lambda_1^t \psi_1[j], \lambda_2^t \psi_2[j], \ldots, \lambda_k^t \psi_k[j]) \quad \text{(nb: j-th coord.)} \tag{90}$$

▷ Diffusion map: probability distribution For any two points of the graph, identified by their indices i and j: proba. to move from i to j in t steps:

$$p_t(i,j) \tag{91}$$

From Eq. (89), one gets

$$p_t(i,j) = \phi_0[j] \; + \; \sum_{j \geq 1} \lambda_j^t \psi_j[i] \; \phi_j[j]. \tag{92}$$

Diffusion maps: diffusion distance

- Similarity between two points: comparing their probability distributions

$$D_t^2(i_0, i_1) = \sum_j (p_t(i_0, j) - p_t(i_1, j))^2 \frac{1}{\phi_0[j]}. \tag{93}$$

The following holds:

Theorem 17. The diffusion distance between two points is equal to the Euclidean distance in the diffusion map space, that is

$$D_t^2(i_0, i_1) = \|\Psi_t(i_0) - \Psi_t(i_1)\|^2. \tag{94}$$

References

▷ Refs: Graphs, Laplacians, random walks: [7, 8, 9]; Diffusion maps: [10, 11, 12]; Spectral clustering: [13]

J. Gallier.

Geometric methods and applications: for computer science and engineering, volume 38.
Springer Science & Business Media, 2011.

G. Golub and C.F. Van Loan.

Matrix computations, volume 3.
JHU Press, 2012.

J.B. Tenenbaum, V. Silva, and J.C. Langford.

A global geometric framework for nonlinear dimensionality reduction.
Science, 290(5500):2319, 2000.

Sam T Roweis and Lawrence K Saul.

Nonlinear dimensionality reduction by locally linear embedding.
science, 290(5500):2323–2326, 2000.

L. van der Maaten and G. Hinton.

Visualizing data using t-SNE.
Journal of Machine Learning Research, 9(Nov):2579–2605, 2008.

L. van Der Maaten.

Accelerating t-SNE using tree-based algorithms.
Journal of machine learning research, 15(1):3221–3245, 2014.

L. Lovász.

Random walks on graphs: A survey.
Combinatorics, Paul erdos is eighty, 2(1):1–46, 1993.

F.R.K. Chung.

Spectral graph theory.


J. Gallier.
Geometric methods and applications: for computer science and engineering, volume 38.
Springer Science & Business Media, 2011.


G. Golub and C.F. Van Loan.
Matrix computations, volume 3.
JHU Press, 2012.


J.B. Tenenbaum, V. Silva, and J.C. Langford.
A global geometric framework for nonlinear dimensionality reduction.
Science, 290(5500):2319, 2000.


Sam T Roweis and Lawrence K Saul.
Nonlinear dimensionality reduction by locally linear embedding.
science, 290(5500):2323–2326, 2000.


L. van der Maaten and G. Hinton.
Visualizing data using t-SNE.
Journal of Machine Learning Research, 9(Nov):2579–2605, 2008.


L. van Der Maaten.
Accelerating t-SNE using tree-based algorithms.
Journal of machine learning research, 15(1):3221–3245, 2014.


L. Lovász.
Random walks on graphs: A survey.
Combinatorics, Paul erdos is eighty, 2(1):1–46, 1993.


F.R.K. Chung.

Spectral graph theory.
Number 92. American Mathematical Soc., 1997.

D. Aldous and J-A. Fill.
Reversible Markov chains and random walks on graphs, 2002.
Unfinished monograph, recompiled 2014, available at
http://www.stat.berkeley.edu/$\sim$aldous/RWG/book.html.

R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker.
Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps.
PNAS, 102(21):7426–7431, 2005.

S. Lafon and A.B. Lee.
Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning and data set parameterization.
IEEE Trans. on Pattern Analysis and Machine Intelligence, 28(9):1393–1403, 2006.

B. Nadler, S. Lafon, R. Coifman, and I.G. Kevrekidis.
Diffusion maps, spectral clustering and reaction coordinates of dynamical systems.
Applied and Computational Harmonic Analysis, 21(1):113–127, 2006.

U. Von Luxburg.
A tutorial on spectral clustering.
Statistics and Computing, 17(4):395–416, 2007.