ENS Lyon January 2010

A geometric (and partial) introduction to Dimensionality Reduction

F. Chazal
Geometrica Group
INRIA Saclay

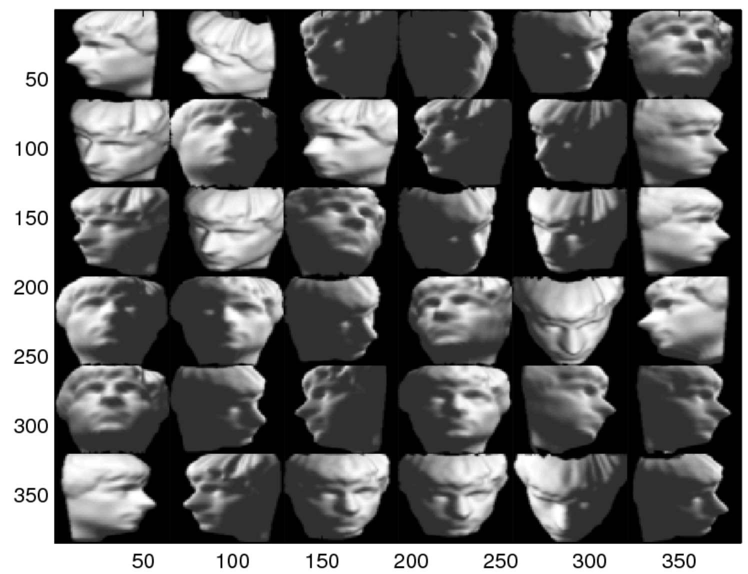
To download these slides:
http://geometrica.saclay.inria.fr/team/Fred.Chazal/Teaching/Dim_Reduction.pdf

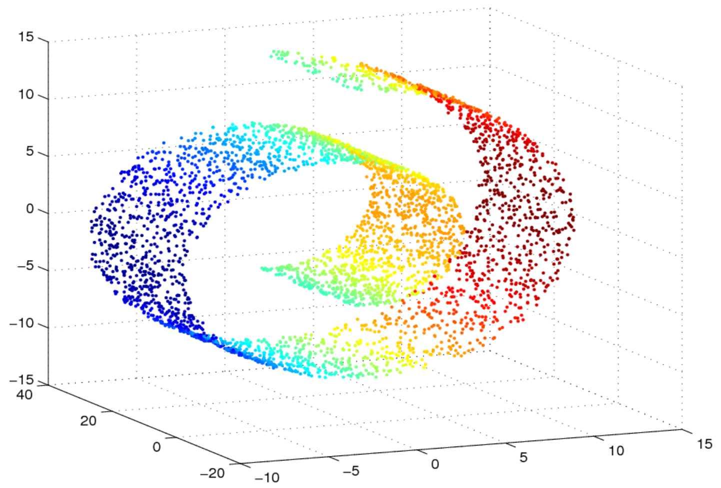
If you have any question:
frederic.chazal@inria.fr

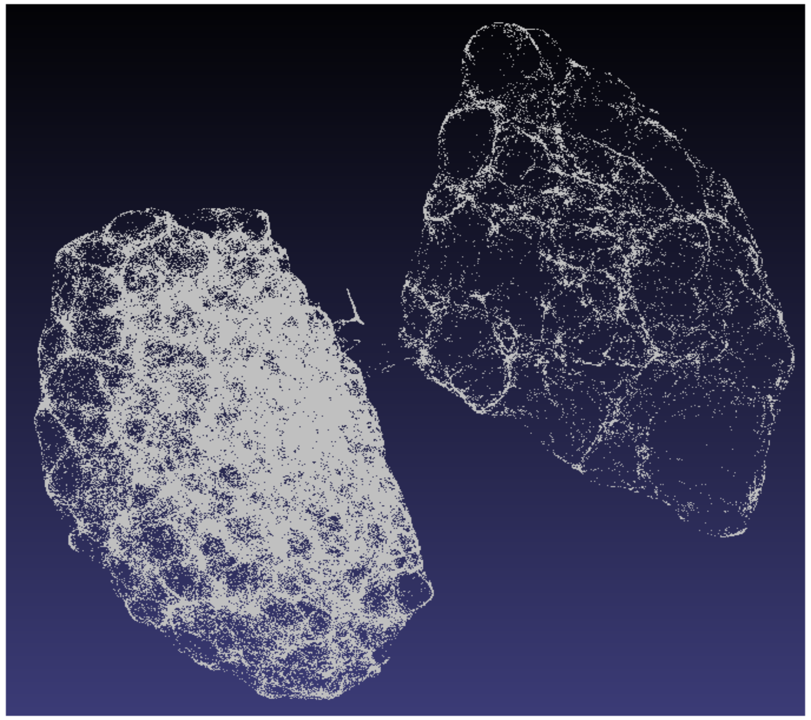
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche
SACLAY - ÎLE-DE-FRANCE

GEOMETRICA

Introduction



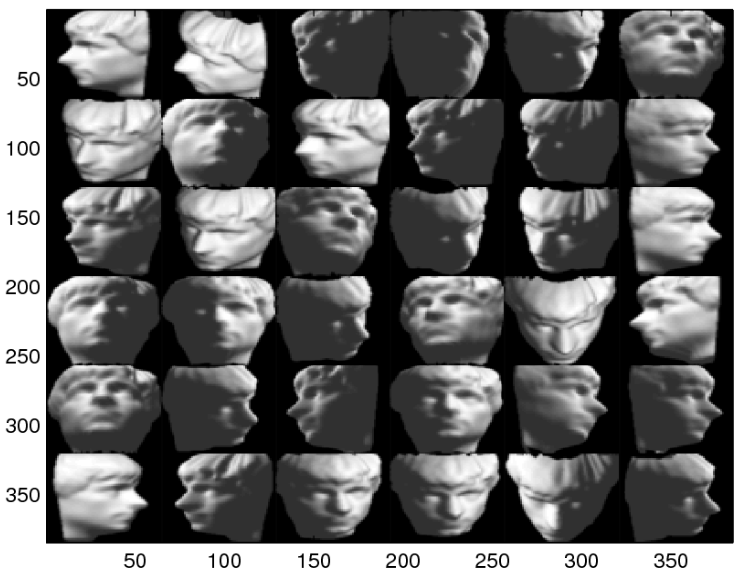


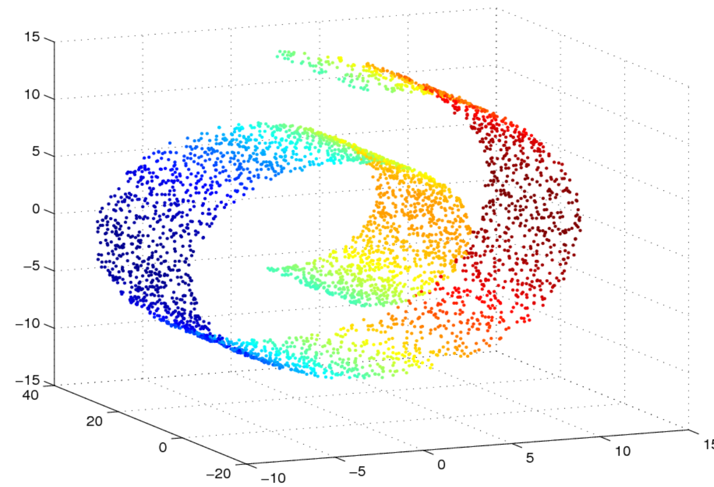


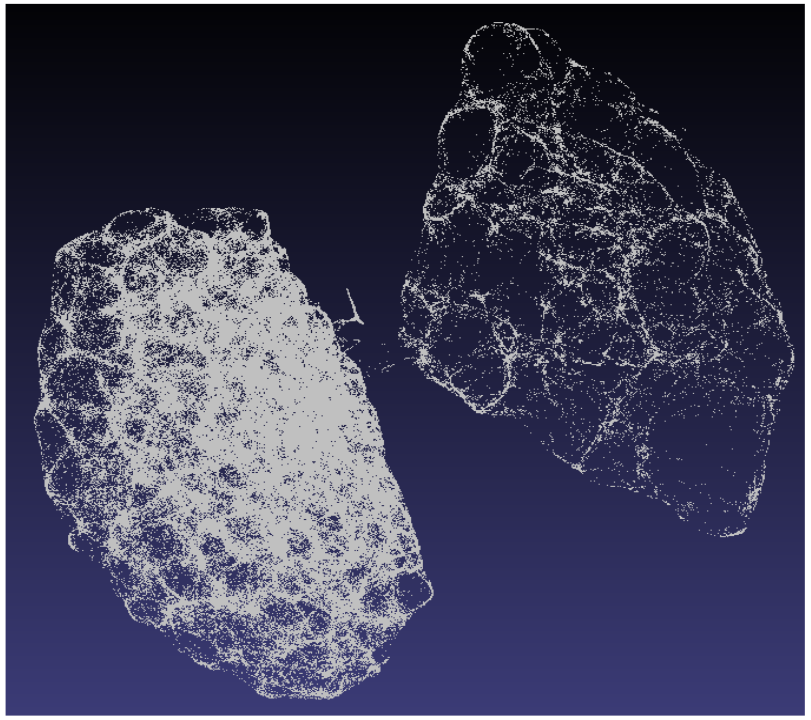
- More and more available data represented by point clouds in high dimensional spaces:
  - measurement and data storage capacities are growing very fast,
  - e.g. images databases, astronomic data,...

- Data often depends upon a small numbers of “independant” parameters (e.g. number of degrees of freedom of an observed system):
  - data sampled around low dimensional shapes (manifolds).
  - underlying manifolds may be highly non linear.

Introduction







- Need to analyze and visualize these data.
- Dimensionality reduction methods intend to embedded the data in low dimensional spaces while preserving as well as possible (some of) their geometric properties. $\Rightarrow$ many different approaches that gave rise to a huge literature in the last decade...
- In this talk:
  - a very incomplete and partial introduction to dimensionality reduction,
  - a focus on a small set of geometric-motivated methods (trying to avoid as most as possible technical details).

Preliminaries and notations

The following notations and assumptions are used all along the talk.

- Data: $X = \{x_1, x_2, \cdots, x_N\} \subset \mathbb{R}^D$ a finite point cloud with mean vector

$$\overline{x} = \sum_{n=1}^{N} x_i \in \mathbb{R}^D$$

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{pmatrix}$$

- N: number of data points
- D: ambient dimension
- Underlying/latent manifold: $M \subset \mathbb{R}^D$ is a d-dimensional submanifold of $\mathbb{R}^D$. The points of X are assumed to be sampled on or around M.

Preliminaries and notations

Different “equivalent” points of view:

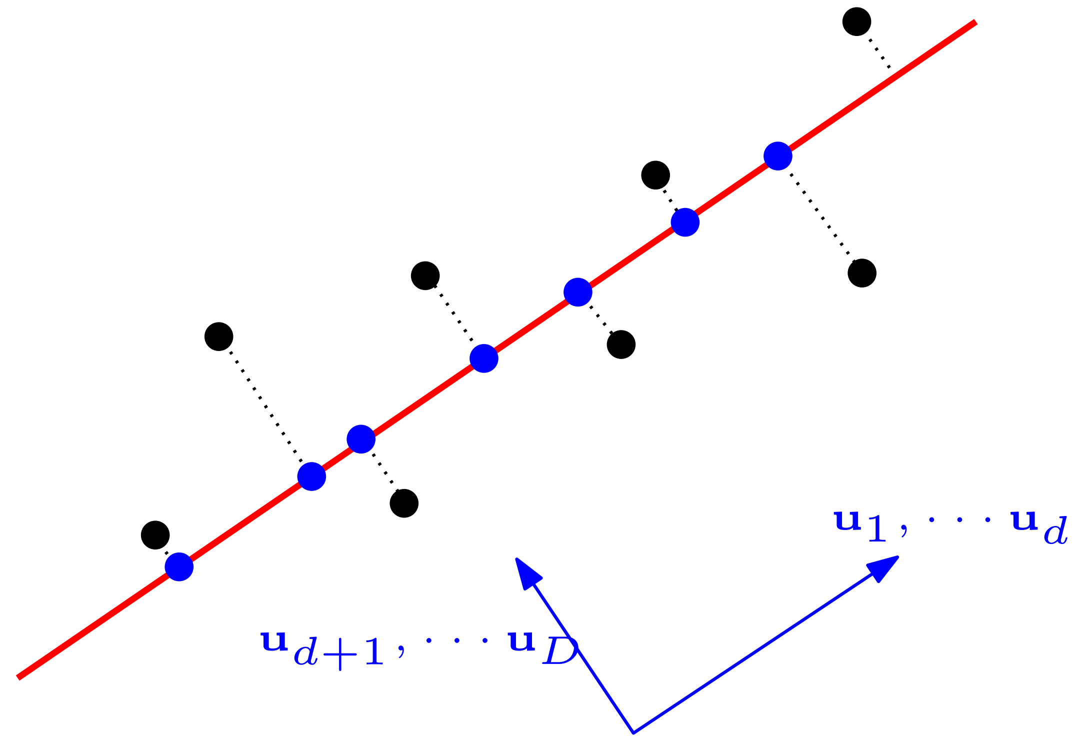
1. $M \subset \mathbb{R}^D$ is a submanifold and one intends to find an embedding Y of X in some low dimensional space such that the “geometry” of Y is as similar as possible as the one of M in some sense,

2. $M = f(N)$ where N is some d-dimensional manifold (the latent manifold - in general N is expected to be an open subset of $\mathbb{R}^d$) and $f : N \to \mathbb{R}^D$ an embedding with some specified properties (isometry, conformal,...). One then intends to find Y such that $X = f(Y)$. The coordinates of Y are known as the latent variables.

3. In some statistical/probabilistic approaches (not considered in this talk): $X = f(Y) + \varepsilon(Y)$ where ε is some noise model.

PCA

Find the d-dimensional subspace of $\mathbb{R}^D$ that best approximates X in a least square sense (and then project X on this subspace)



Let V be a d-dimensional subspace of $\mathbb{R}^D$ and let $\mathbf{u}_1, \cdots \mathbf{u}_D$ be an orthonormal basis s.t. $\mathbf{u}_1, \cdots \mathbf{u}_d$ is a basis of $\vec{V}$.

Approximate each point x_n by $\tilde{x}_n = \sum_{i=1}^{d} \alpha_{ni} \mathbf{u}_i + \sum_{i=d+1}^{D} b_i \mathbf{u}_i$

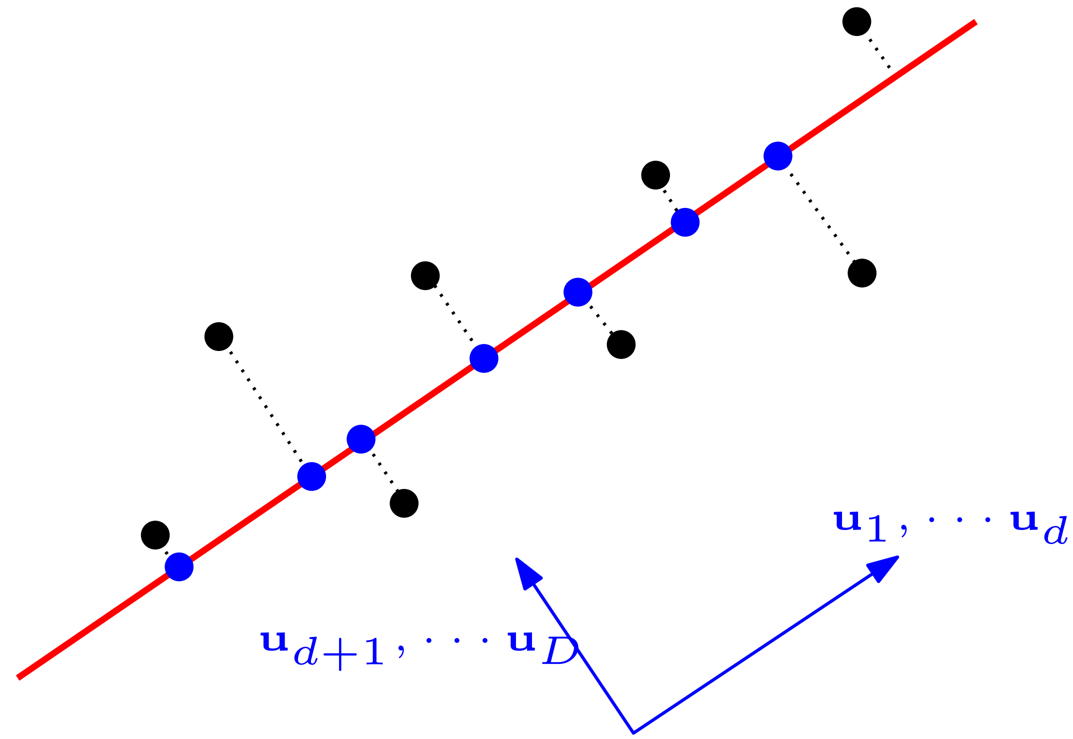
Independent of n

Minimize $E = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|^2$

PCA

$$\tilde{x}_n = \sum_{i=1}^{d} \alpha_{ni} \mathbf{u}_i + \sum_{i=d+1}^{D} b_i \mathbf{u}_i$$

$$E = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|^2$$



Minimizing E with respect to α_{ni} and b_i leads to

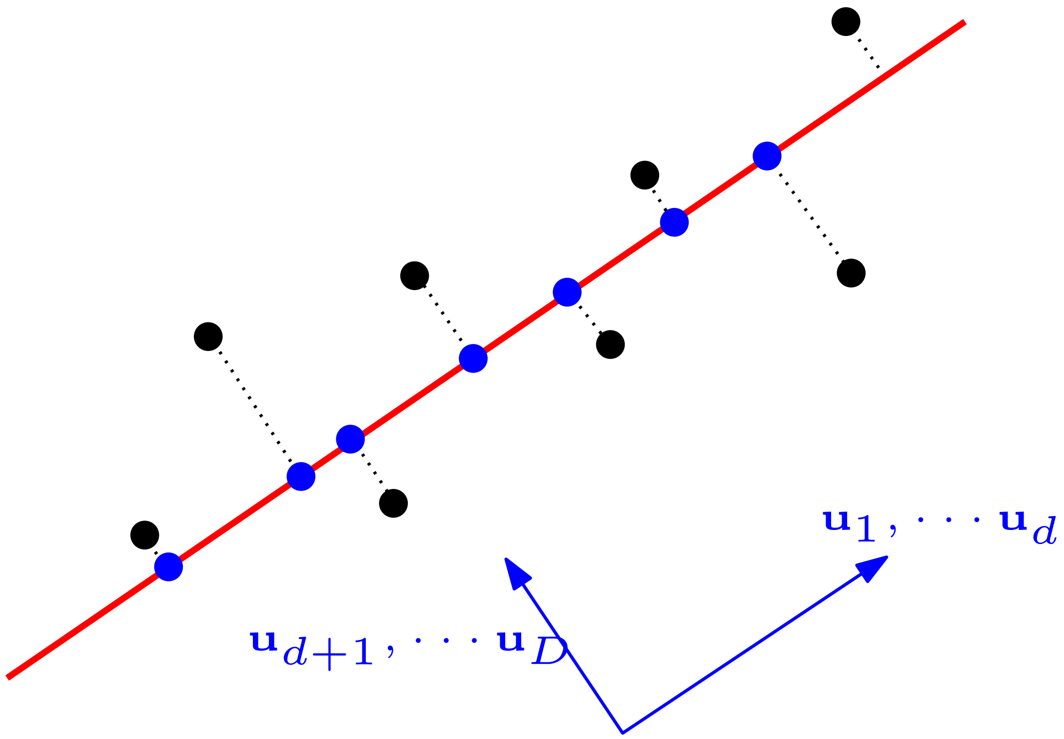
$$x_n - \tilde{x}_n = \sum_{i=d+1}^{D} \{(x_n - \overline{x})^T \mathbf{u}_i\} \mathbf{u}_i$$

$\Rightarrow$ Given $\vec{V}$ the best affine subspace V is the one passing through $\overline{x}$ and $\tilde{x}_n$ is the orthogonal projection on V.

PCA

$$E = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|^2$$

$$x_n - \tilde{x}_n = \sum_{i=d+1}^{D} \{(x_n - \overline{x})^T \mathbf{u}_i\} \mathbf{u}_i$$



Now E only depends on $\mathbf{u}_i$:

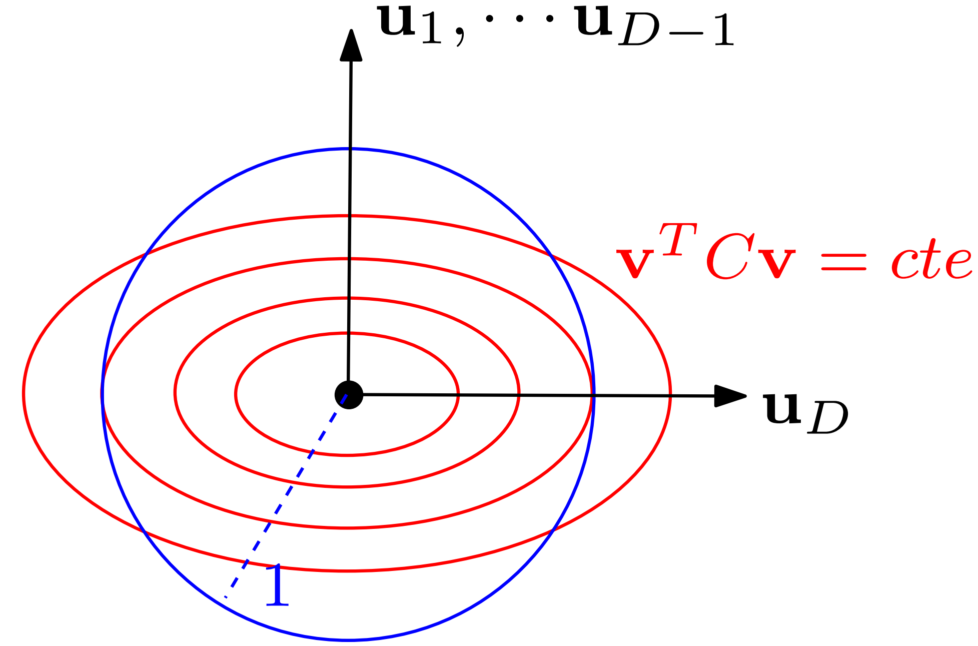
$$E = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=d+1}^{D} (x_n^T \mathbf{u}_i - \overline{x}^T \mathbf{u}_i)^2 = \sum_{i=d+1}^{D} \mathbf{u}_i^T C \mathbf{u}_i$$

where

$$C = \frac{1}{N} \sum_{n=1}^{N} (x_n - \overline{x})(x_n - \overline{x})^T$$ is the covariance matrix of X

PCA

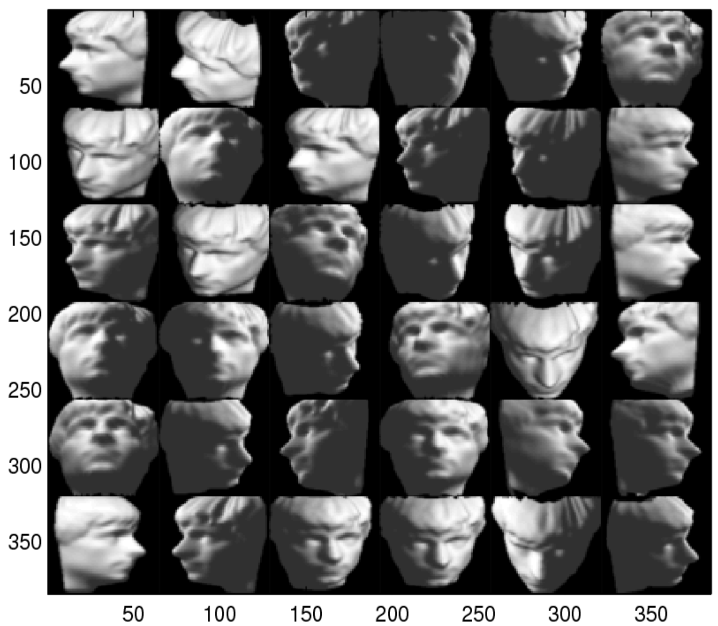
Find the d-dimensional subspace of $\mathbb{R}^D$ that best approximates X in a least square sense (and then project X on this subspace)

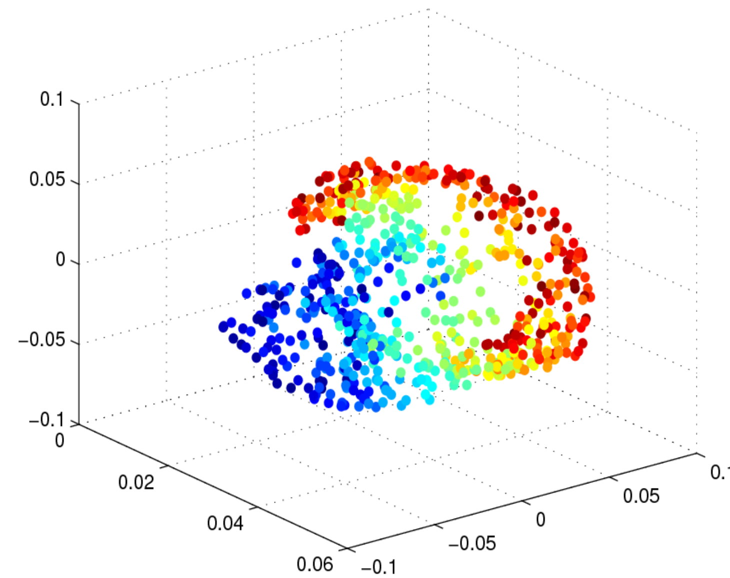


Solution: the space spanned by the d eigenvectors corresponding to the d largest eigenvalues of the covariance matrix

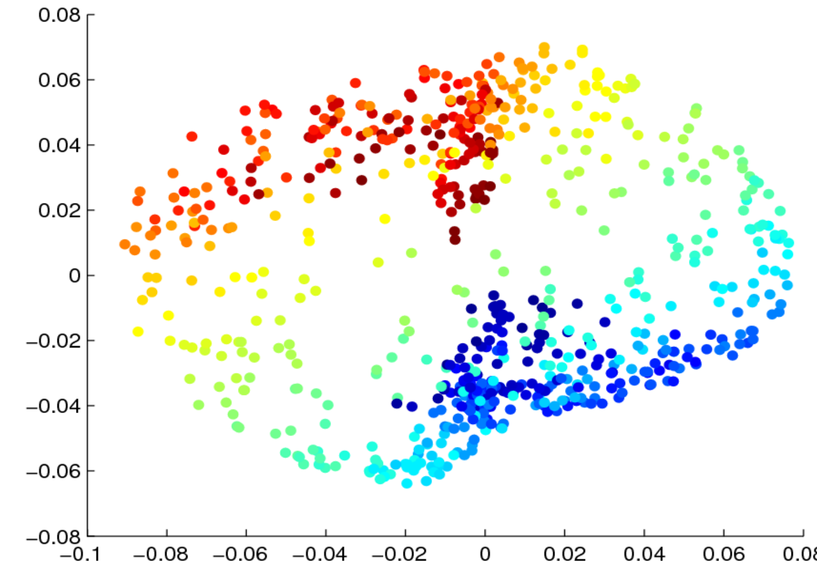
$$C = \frac{1}{N} \sum_{n=1}^{N} (x_n - \overline{x})(x_n - \overline{x})^T$$

PCA: example





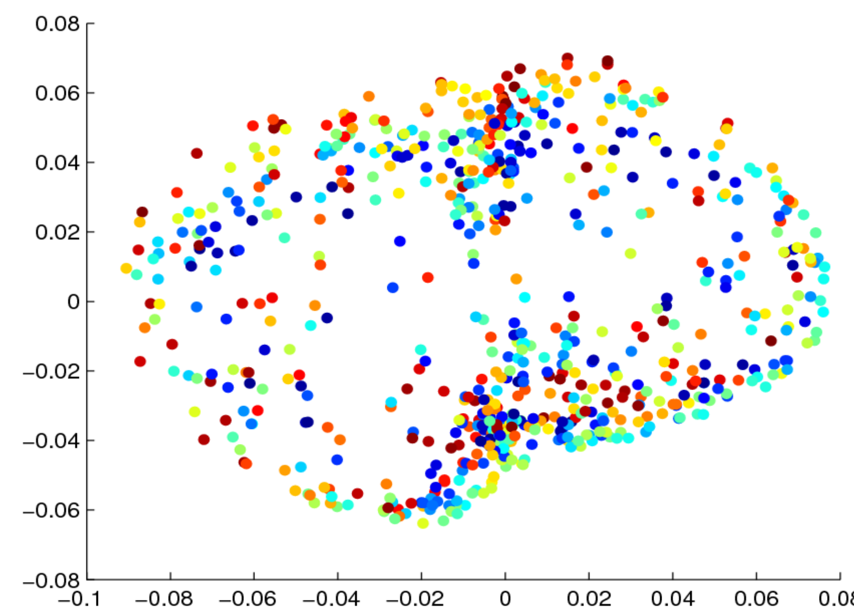
3D-proj: light



2D-proj: pose 1

Dimension: $64 * 64 = 4096.$

$N = 698$



2D-proj: pose 2

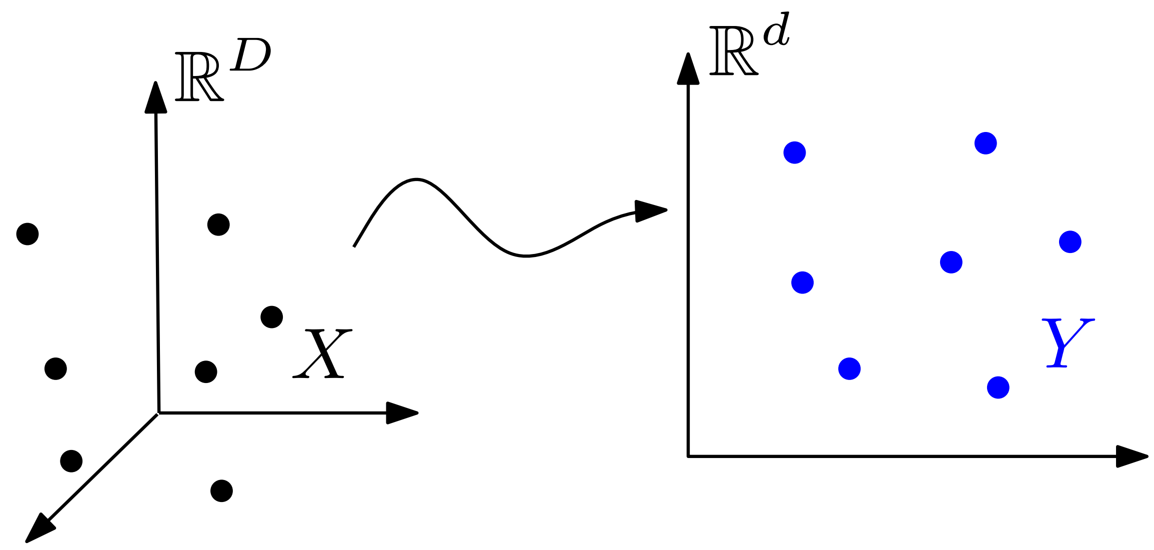
3 free parameters:

- left-right pose,
- up-down pose,
- light pose.

Multidimensional Scaling (MDS)

Find a low dimensional "projection" $Y \subset \mathbb{R}^d$ of the data X such as to preserve, as closely as possible, the pairwise distances between data points.

Without loss of gen. we assume that $\overline{x} = 0$ (and $\overline{y} = 0$).



- The $N \times N$ matrix of squared pairwise distance: $D = D_X = (\|x_i - x_j\|^2)$
- Relationship between D and the *Gram matrix*: $G = G_X = (x_i^T x_j) = XX^T$:

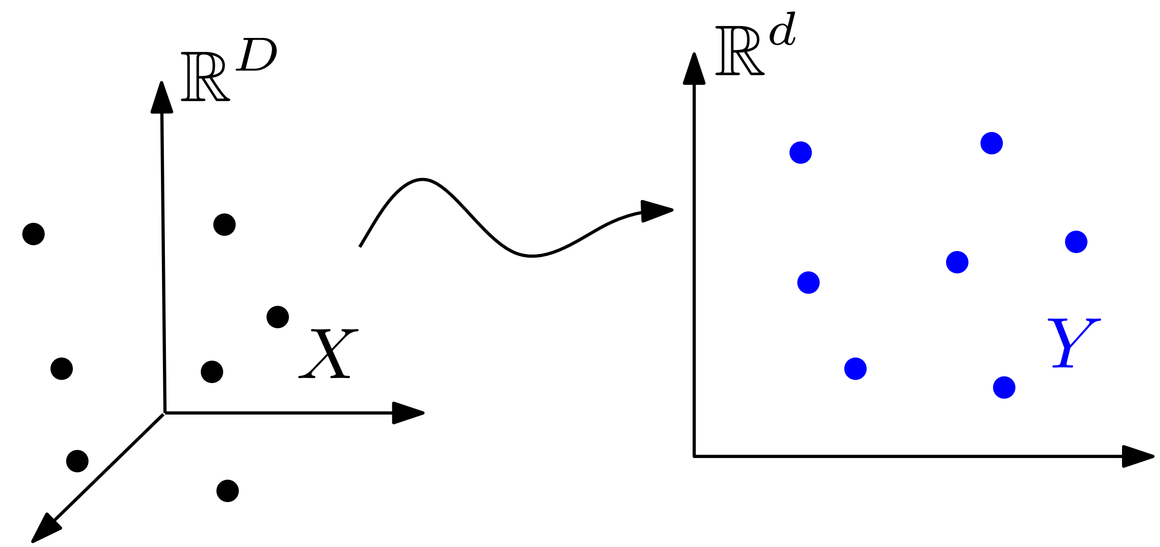
$$G = -\frac{1}{2} JDJ \quad \text{where} \quad J = Id_N - \frac{1}{N}\mathbf{1}\mathbf{1}^T = (\delta_{ij} - \frac{1}{N})$$

- **Goal:** Find $Y = \{y_1, \cdots y_N\} \subset \mathbb{R}^d$ minimizing

$$\rho(D_X, D_Y) = \|G_X - G_Y\|_2^2 = \|\frac{1}{2}J(D_X - D_Y)J\|_2^2$$

Multidimensional Scaling (MDS)

Find a low dimensional "projection" $Y \subset \mathbb{R}^d$ of the data X such as to preserve, as closely as possible, the pairwise distances between data points.



Without loss of gen. we assume that $\overline{x} = 0$ (and $\overline{y} = 0$).

Solution:

- Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0$ be the eigenvalues of G_X and $\{\mathbf{v}_1, \cdots \mathbf{v}_N\} \subset \mathbb{R}^N$ an orthonormal eigenbasis.
- $Y \subset \mathbb{R}^d$ minimizing $\rho(D_X, D_Y)$ is given by the columns of the $d \times N$ matrix

$$Y = \begin{pmatrix} \sqrt{\lambda_1}\mathbf{v}_1^T \\ \sqrt{\lambda_2}\mathbf{v}_2^T \\ \vdots \\ \sqrt{\lambda_d}\mathbf{v}_d^T \end{pmatrix}$$

Multidimensional Scaling (MDS)

Justification:

$$\begin{aligned}
\min_Y \|G_X - G_Y\|_2^2 &= \min_Y \|XX^T - YY^T\|^2 \\
&= \min_Y \sum_{i=1}^{N}\sum_{j=1}^{N} (x_i^T x_j - y_i^T y_j)^2 \\
&= \min_Y Tr((XX^T - YY^T)^2)
\end{aligned}$$

XX^T and YY^T are semidefinite positive: $XX^T = V\Lambda V^T$ and $YY^T = W\Lambda' W^T$ where
- $VV^T = WW^T = Id_N$,
- $\Lambda = Diag(\lambda_1, \cdots, \lambda_N)$ is diagonal with $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_N$,
- $\Lambda' = Diag(\lambda'_1, \cdots \lambda'_d, 0, \cdots, 0)$ is diagonal with $\lambda'_1 \geq \cdots \geq \lambda'_d \geq 0$ because $Y \subset \mathbb{R}^d$.

$$\begin{aligned}
\min_Y Tr((XX^T - YY^T)^2) &= \min_{W,\Lambda'} Tr(\Lambda - V^T W \Lambda' W^T V)^2 \quad \text{(use } Tr(AB) = Tr(BA)\text{)} \\
&= \min_{Q,\Lambda'} Tr(\Lambda - Q\Lambda' Q^T)^2 \ \text{ with } \quad Q = V^T W \\
&= \min_{Q,\Lambda'} Tr(\Lambda^2) + Tr(Q\Lambda' Q^T Q \Lambda' Q^T) - 2Tr(\Lambda Q \Lambda' Q^T)
\end{aligned}$$

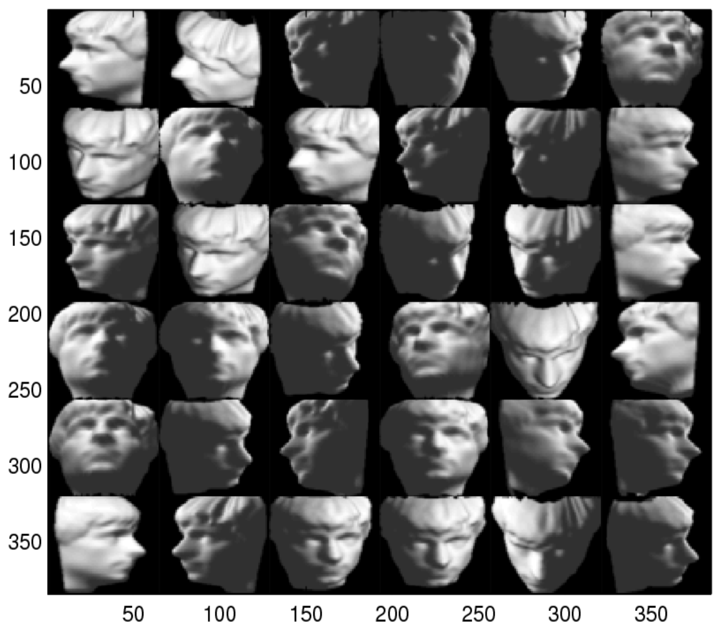
Multidimensional Scaling (MDS)

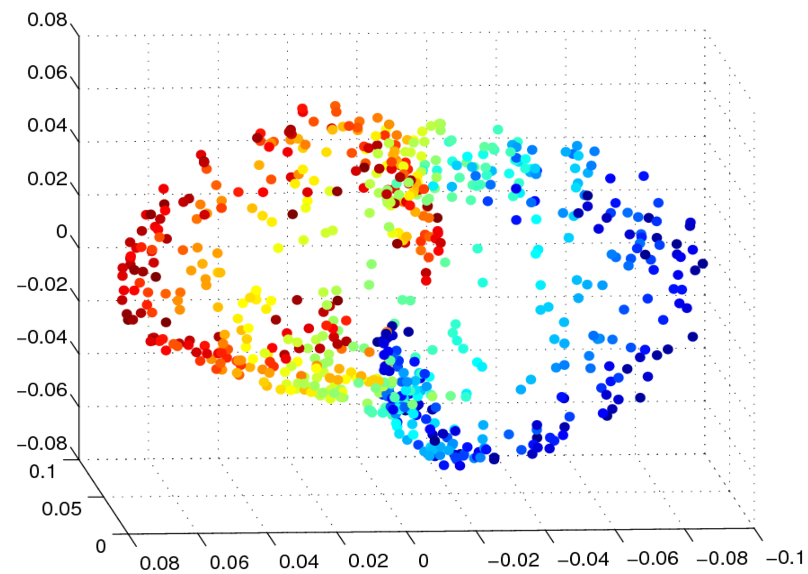
Justification:

$$
\begin{aligned}
\min_{Y} Tr((XX^T - YY^T)^2) &= \min_{W,\Lambda'} Tr(\Lambda - V^T W \Lambda' W^T V)^2 \quad \text{(use } Tr(AB) = Tr(BA)\text{)} \\
&= \min_{Q,\Lambda'} Tr(\Lambda - Q\Lambda' Q^T)^2 \quad \text{with} \quad Q = V^T W \\
&= \min_{Q,\Lambda'} Tr(\Lambda^2) + Tr(Q\Lambda' Q^T Q \Lambda' Q^T) - 2Tr(\Lambda Q \Lambda' Q^T) \\
&= \min_{\Lambda'} Tr(\Lambda^2 + \Lambda'^2 - 2\Lambda\Lambda') \\
&= \min_{\Lambda'} Tr(\Lambda - \Lambda')^2
\end{aligned}
$$

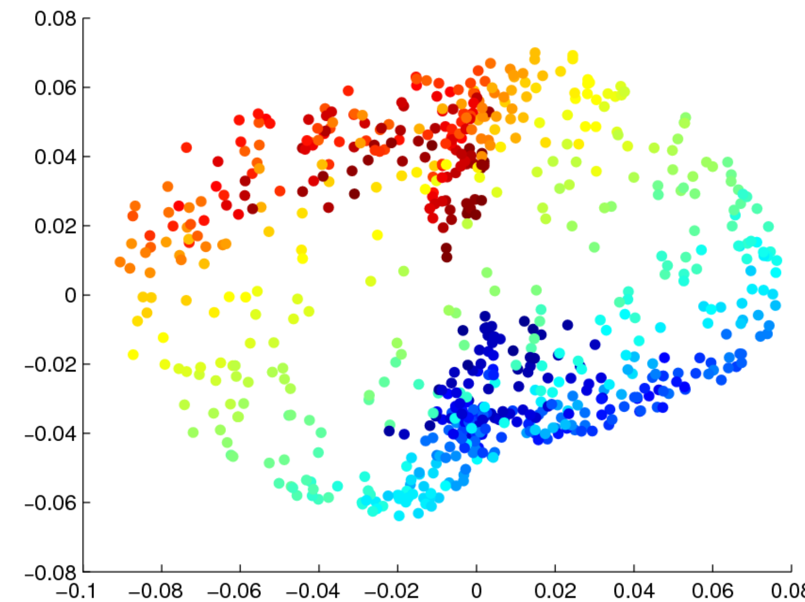
- The minimum is thus obtain for $\Lambda' = Diag(\lambda_1, \cdots, \lambda_d, 0, \cdots, 0)$ and one can choose $Q = V^T W = Id_N$ ($\Rightarrow W = V$).
- Since $YY^T = W\Lambda' W^T$, one has $Y = W\Lambda'^{\frac{1}{2}} = V\Lambda'^{\frac{1}{2}}$.

MDS: example





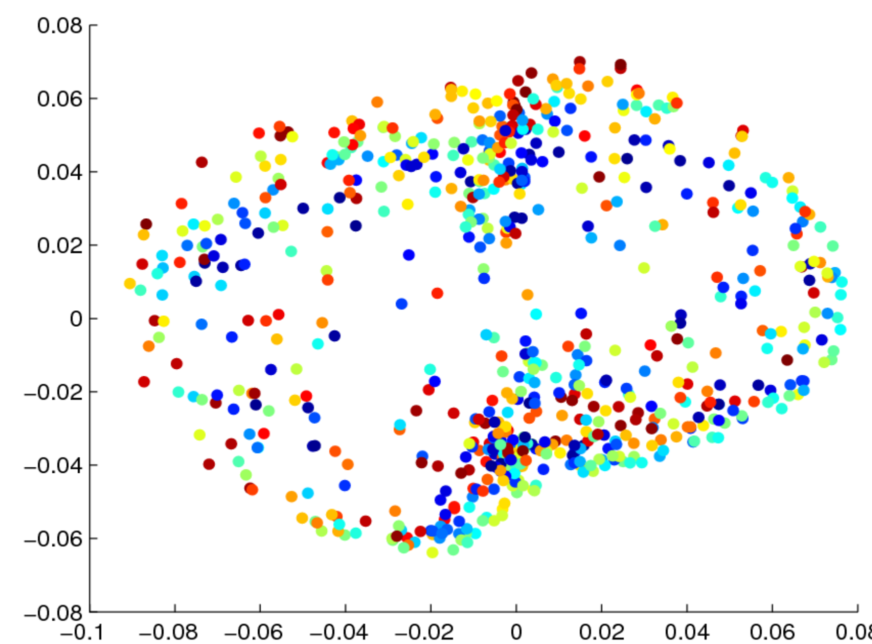
3D-proj: light



2D-proj: pose 1

Dimension: $64 * 64 = 4096$.
$N = 698$

3 free parameters:
- left-right pose,
- up-down pose,
- light pose.



2D-proj: pose 2

MDS: remarks

- Let $v \in \mathbb{R}^D$ be an eigenvector of C with eigenvalue λ. One has

$$GXv = XX^T Xv = XCv = \lambda Xv$$

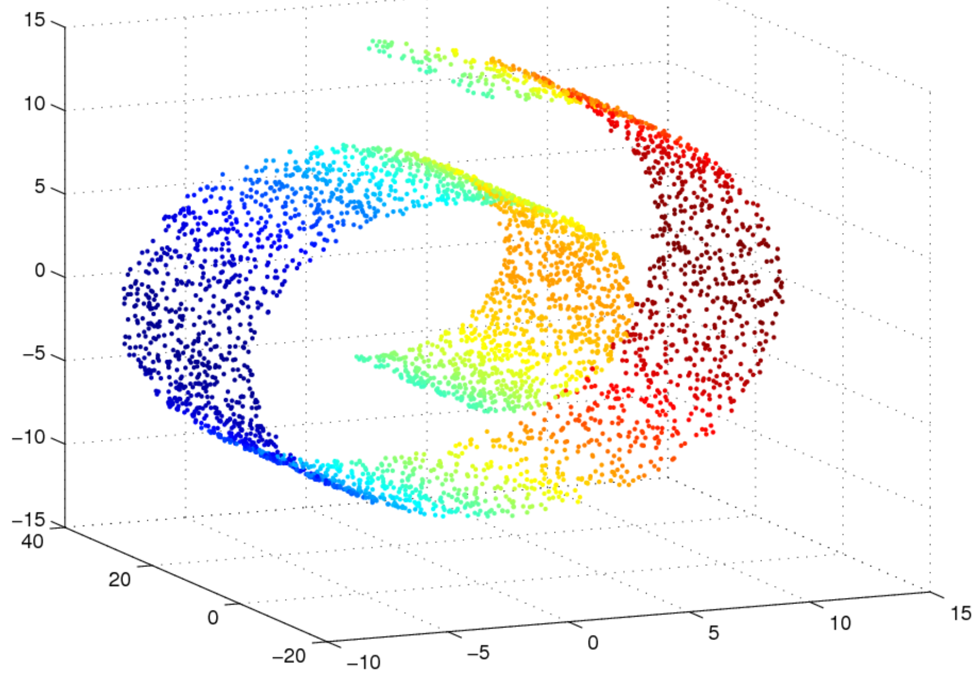
  so $Xv \in \mathbb{R}^N$ is an eigenvector of G with eigenvalue λ. Equivalently if $w \in \mathbb{R}^N$ is an eigenvector of G with eigenvalue μ, $X^T w \in \mathbb{R}^D$ is an eigenvector of C with eigenvalue μ.

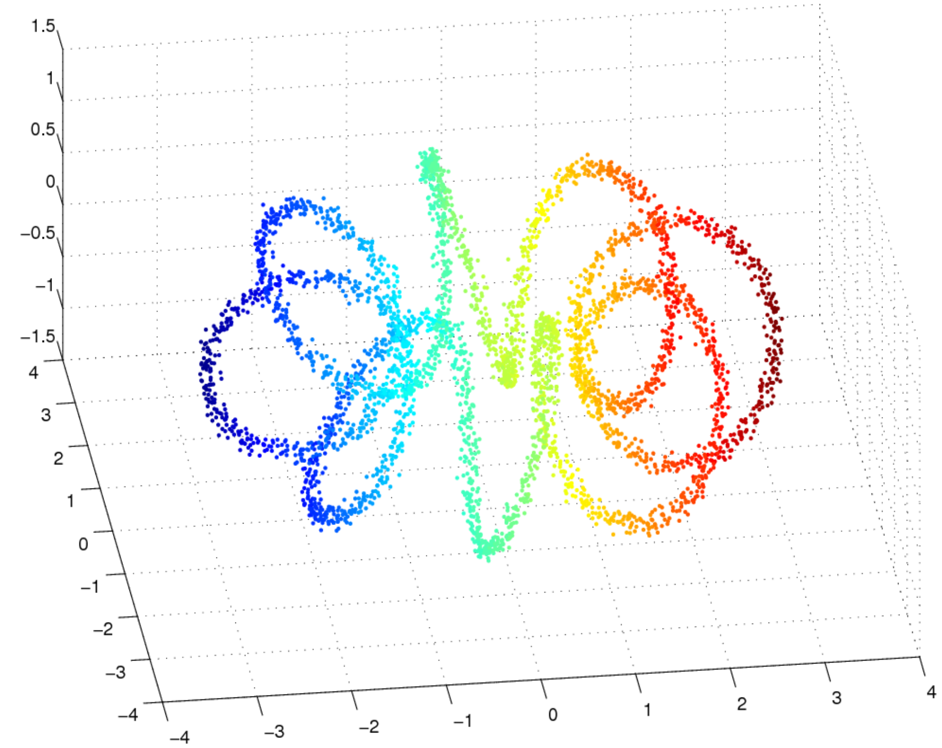
- IMPORTANT: MDS does not require the knowledge of the coordinates of the points of X. If only the matrix D of the pairwise squared distances between the data points is known, one can still apply MDS by first "double centering" D: $G = -\frac{1}{2} JDJ$.

- IMPORTANT: If D is not obtained from a point cloud $X \subset \mathbb{R}^D$, one can still apply MDS but G may have negative eigenvalues (indeed negative eigenvalues signify that D is non Euclidean). The d-dimensional embedding Y_{MDS} given by MDS is the one that have the Gram matrix that best approximates $G = -\frac{1}{2} JDJ$ (Eckart and Young '36): for any $Y \subset \mathbb{R}^d$,

$$\|Y_{MDS} Y_{MDS}^T - G\| \leq \|YY^T - G\|$$

Turning non linear

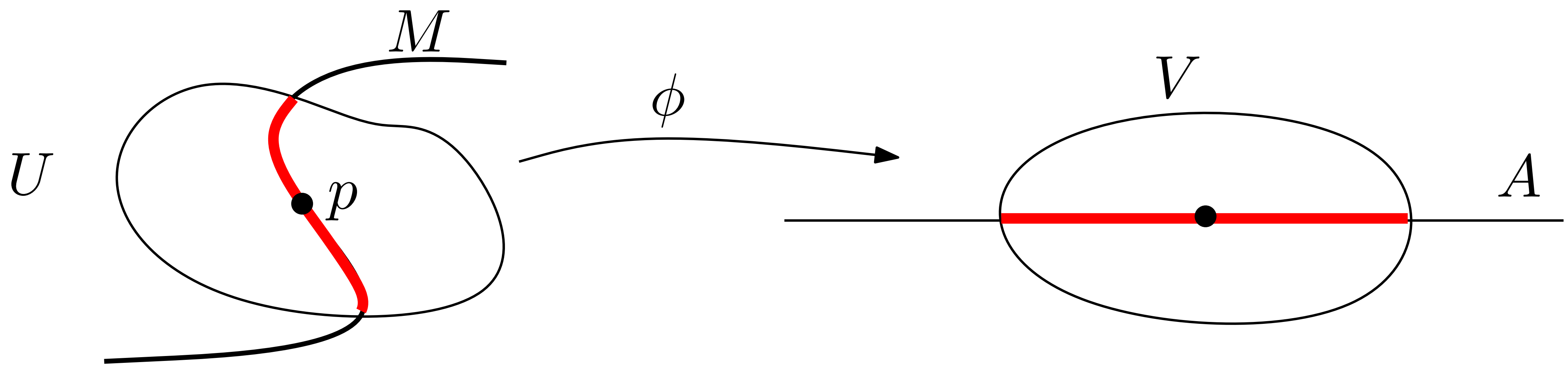




- (Classical) PCA and MDS become inefficient when the data is located around highly non linear manifolds.

- From now on, we assume that the observed data lie on or are close to a d-dimensional submanifold $M \subset \mathbb{R}^D$.

Turning non linear

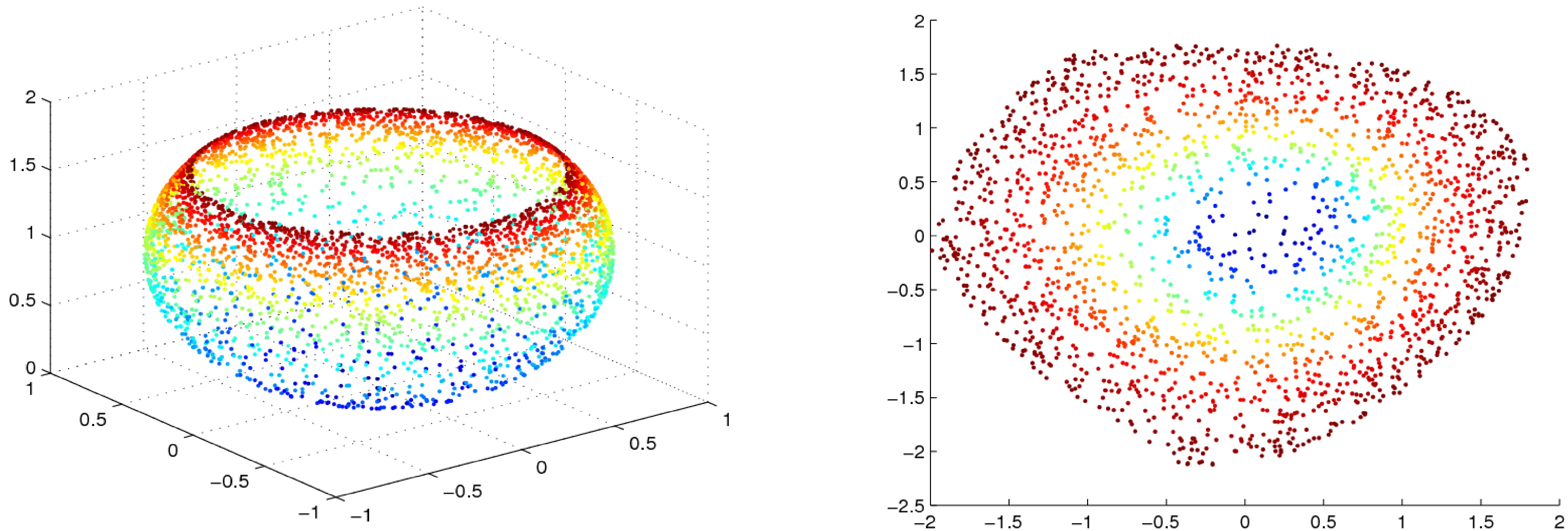


A subset $M \subset \mathbb{R}^D$ is a submanifold of dimension d (of class $\mathcal{C}^k$) if for any $p \in M$, there exist a neighborhood U of p in $\mathbb{R}^D$, a diffeomorphism ϕ (of classe $\mathcal{C}^k$ between U and an open set V and an affine subspace A in $\mathbb{R}^D$ such that

$$\phi(U \cap M) = A \cap V$$

- (Classical) PCA and MDS become inefficient when the data is located around highly non linear manifolds.
- From now on, we assume that the observed data lie on or are close to a d-dimensional submanifold $M \subset \mathbb{R}^D$.

Locally Linear Embedding (L. Saul and S Roweis '00)



- Preservation of the local geometry of the data: LLE intends to find an embedding of the data $X \subset \mathbb{R}^D$ such that
  - nearby points remain nearby in the target low dimensional space,
  - nearby points remain similarly co-located in the target low dimensional space.

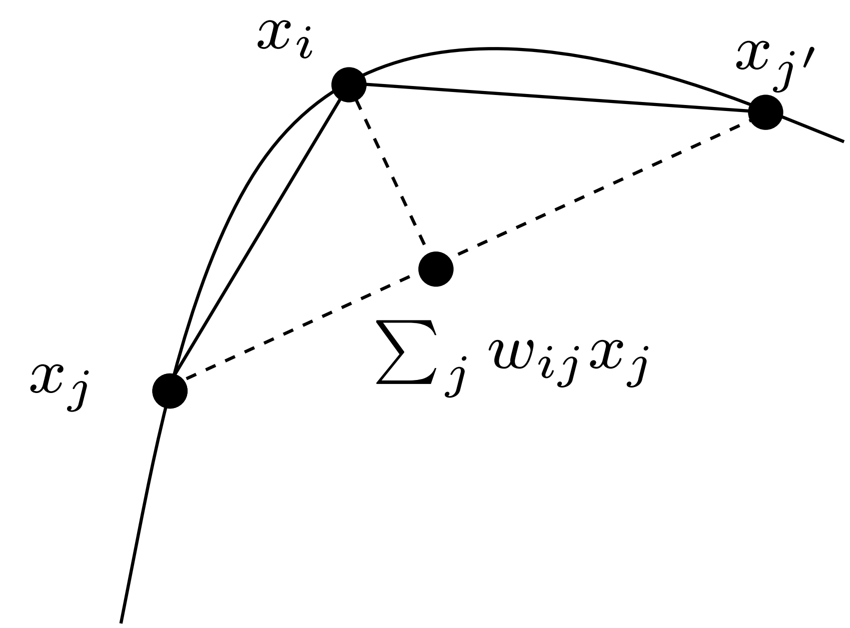
Locally Linear Embedding (L. Saul and S Roweis '00)

1. Build a neighborhood graph $\mathcal{G}$ with vertex set X.

2. Compute weights w_{ij} that best reconstruct each data point x_i from its neighbors by minimizing the cost function

$$E(W) = \sum_i \|x_i - \sum_j w_{ij} x_j\|^2$$

with the constraints that $w_{ij} = 0$ if x_i and x_j are not connected in $\mathcal{G}$ and that $\sum_j w_{ij} = 1$.



3. Compute the vectors y_i minimizing the quadratic cost

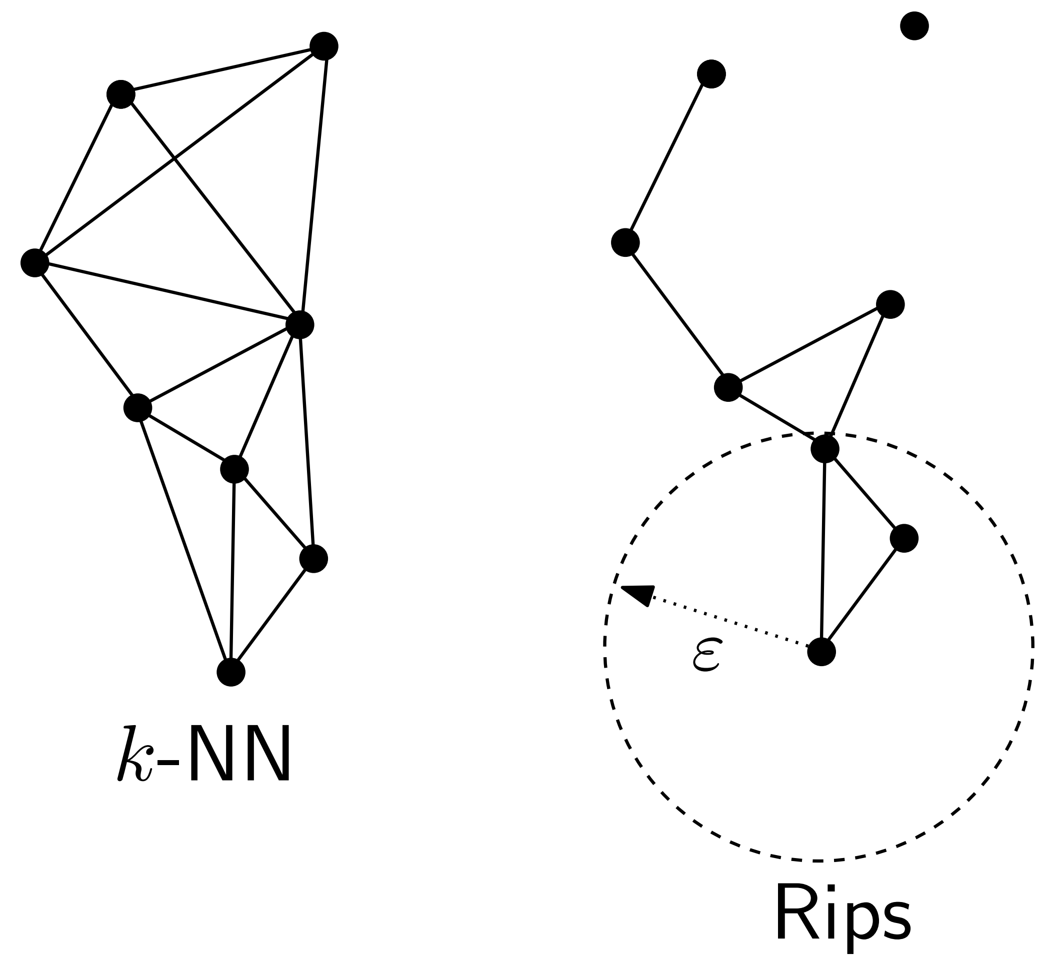
$$\Phi(Y) = \sum_i \|y_i - \sum_j w_{ij} y_j\|^2$$

Locally Linear Embedding (L. Saul and S Roweis '00)

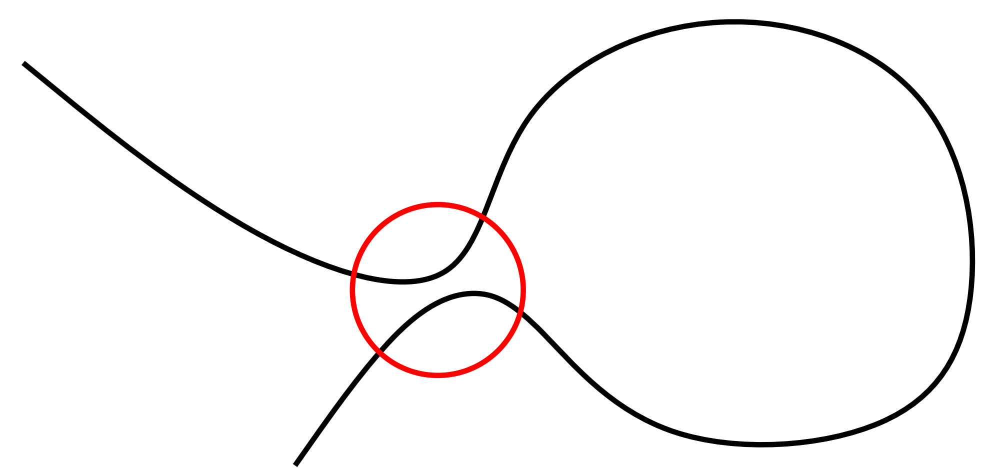
1. Build a neighborhood graph $\mathcal{G}$ with vertex set X.

- k-**NN graph** (depends on an integer parameter k): $x_i x_j$ is an edge of $\mathcal{G}$ iff x_j is one of the k nearest neighbours of x_i (and vice-versa).

- **Rips graph** (depends on a real parameter $\varepsilon > 0$): $x_i x_j$ is an edge of $\mathcal{G}$ iff $d(x_i, x_j) \leq \varepsilon$.



Warning: The choice of the neighborhood graph may be critical!



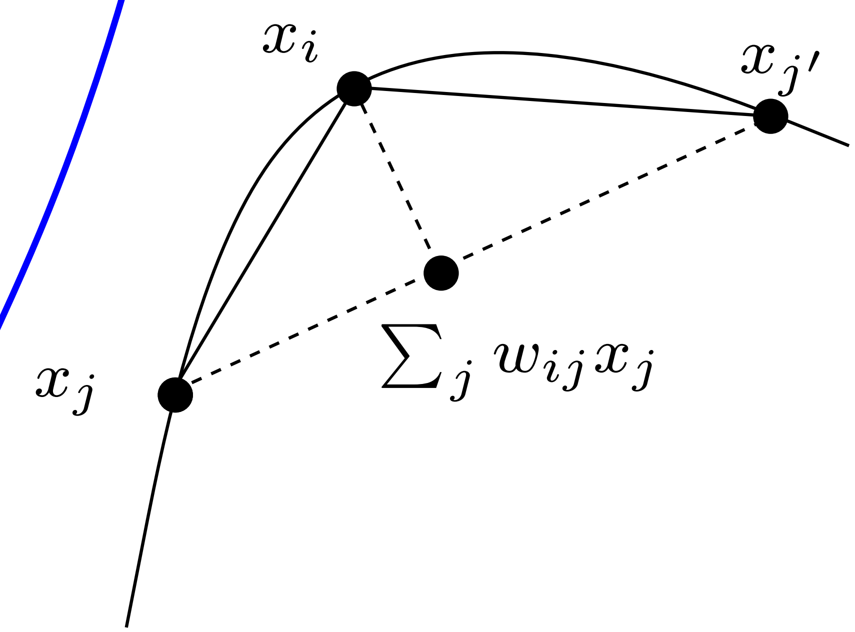
Locally Linear Embedding (L. Saul and S Roweis '00)

Invariance under scaling, rotation and translation

2. Compute weights w_{ij} that best reconstruct each data point x_i from its neighbors by minimizing the cost function

$$E(W) = \sum_i \|x_i - \sum_j w_{ij} x_j\|^2$$

with the constraints that $w_{ij} = 0$ if x_i and x_j are not connected in $\mathcal{G}$ and that $\sum_j w_{ij} = 1$.



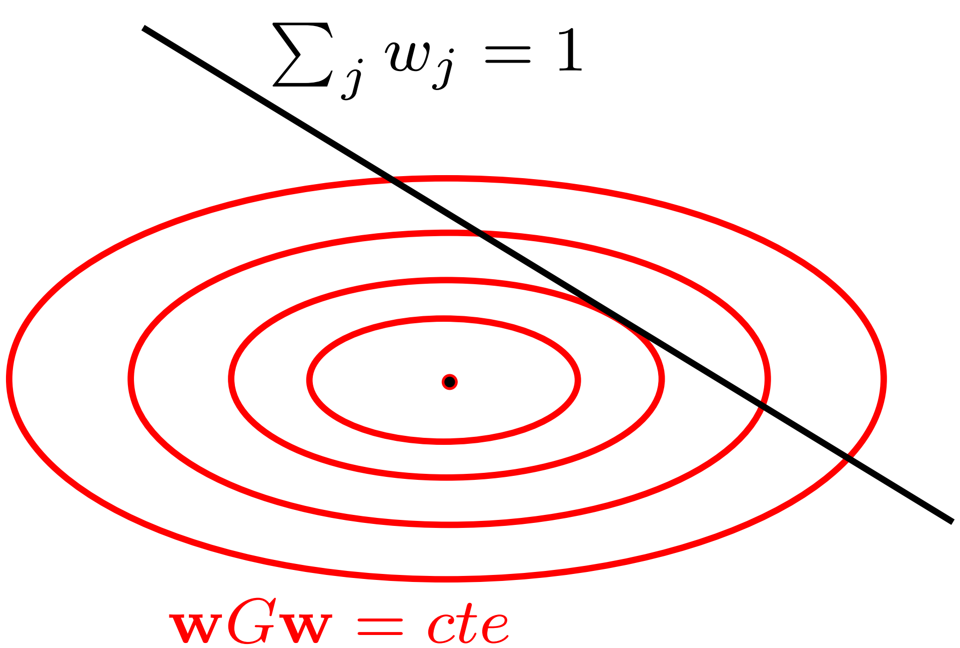
N quadratic minimizations (with contraints), each involving the local Gram matrix $G^i = (G^i_{jk}) = ((x_i - x_j)^T (x_i - x_k))$

Locally Linear Embedding (L. Saul and S Roweis '00)

Let $x \in X$, let x_j be its neighbors in $\mathcal{G}$ and let $w_j = w_{ij}$:

$$\varepsilon = \|x - \sum_{j \in N_{\mathcal{G}}(x)} w_j x_j\|^2 = \|\sum_j w_j (x - x_j)\|^2 = \sum_{j,k} w_j w_k G_{jk}$$

where $G = (G_{jk}) = ((x - x_j)^T (x - x_k))$ is the "local" Gram matrix.



G being semipositive definite, the minimization of ε admits a closed form solution:

- Solve the linear system $G\mathbf{w} = (1, 1, \cdots, 1)^T$
- Rescale the w_j such that they sum to 1.

Warning: if G is singular or nearly singular (e.g. if the number of neighbors is greater than D), it may need to be regularized by adding a small multiple of the identity matrix ($\Rightarrow$ penalize large weigths).

Locally Linear Embedding (L. Saul and S Roweis '00)

constraints:

- remove translational degree of freedom: $\overline{y} = \sum_i y_i = 0$
- remove rotational degree of freedom: $\frac{1}{N} \sum_i y_i y_i^T = Id_d$

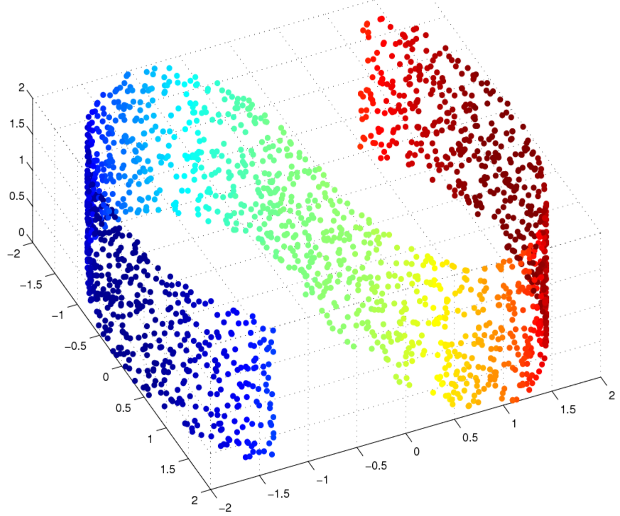
Solution:

- $M = (I - W)^T (I - W)$ with $W = (w_{ij})$
- compute the $(d+1)$ eigenvectors $\mathbf{v}_0, \cdots \mathbf{v}_d$ of M corresponding to the $(d+1)$ smallest eigenvalues $\lambda_0 \leq \cdots \leq \lambda_d$ and discard $\mathbf{v}_0$.
- the y_i are given by the lines of the matrix $(\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_d)$.

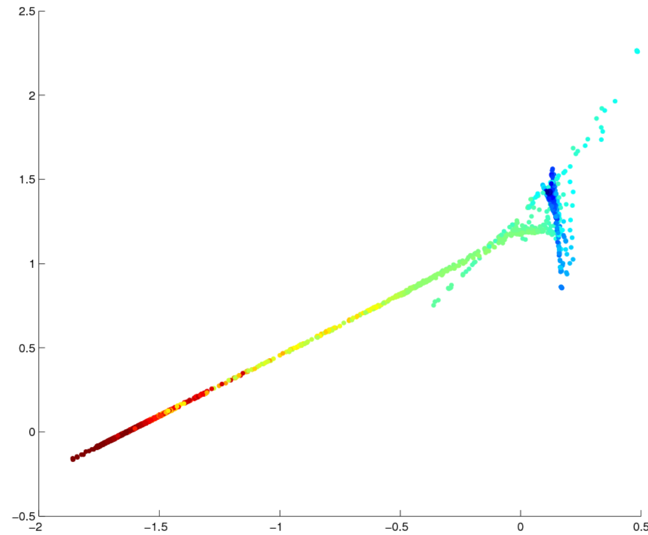
3. Compute the vectors y_i minimizing the quadratic cost

$$\Phi(Y) = \sum_i \|y_i - \sum_j w_{ij} y_j\|^2$$

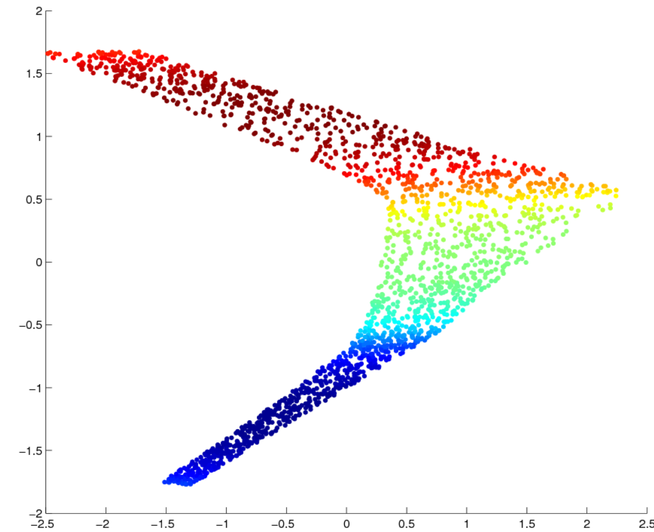
LLE: examples



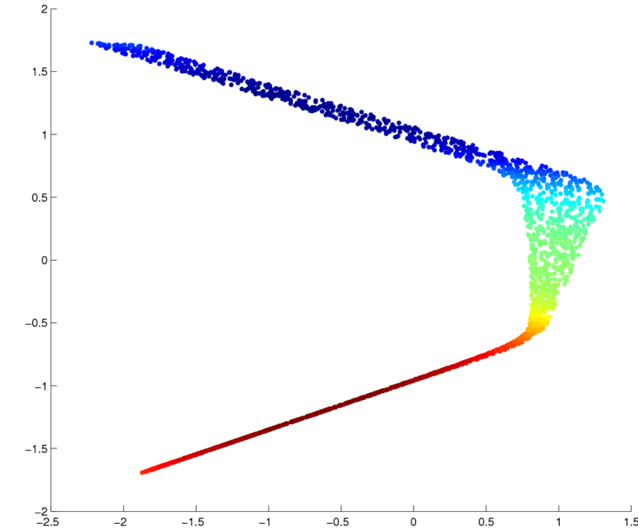
S



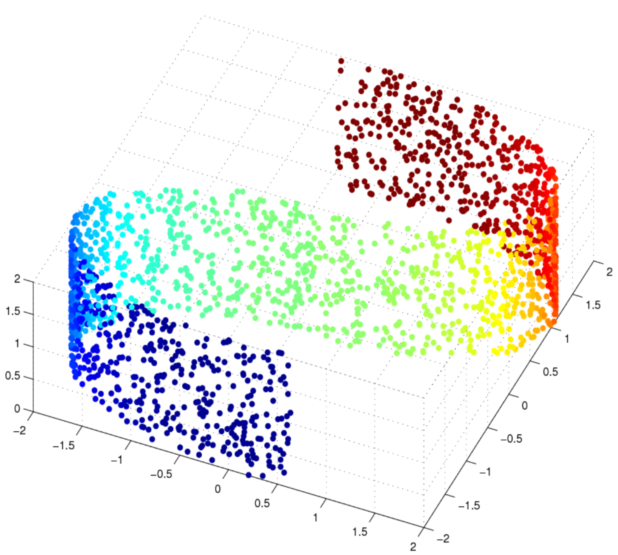
5-NN



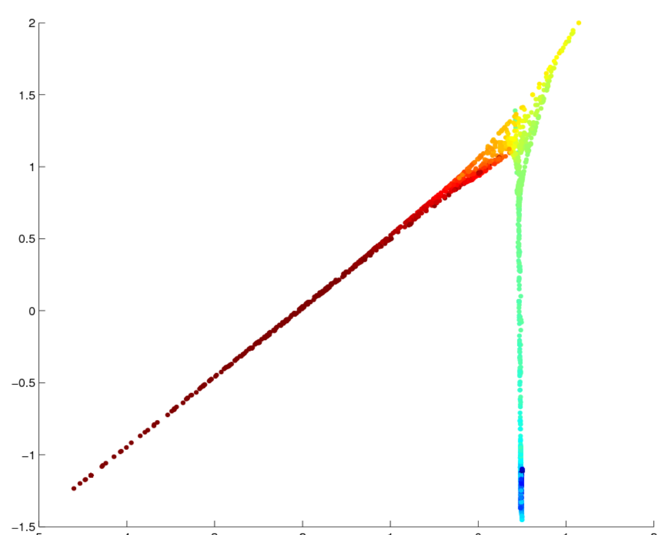
12-NN



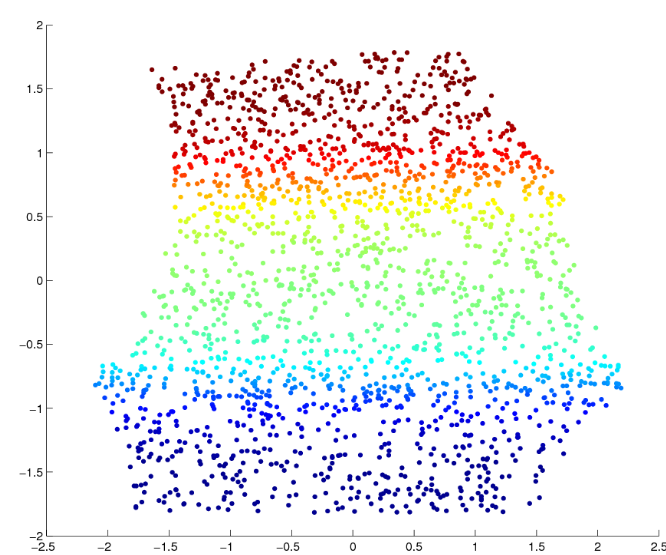
30-NN



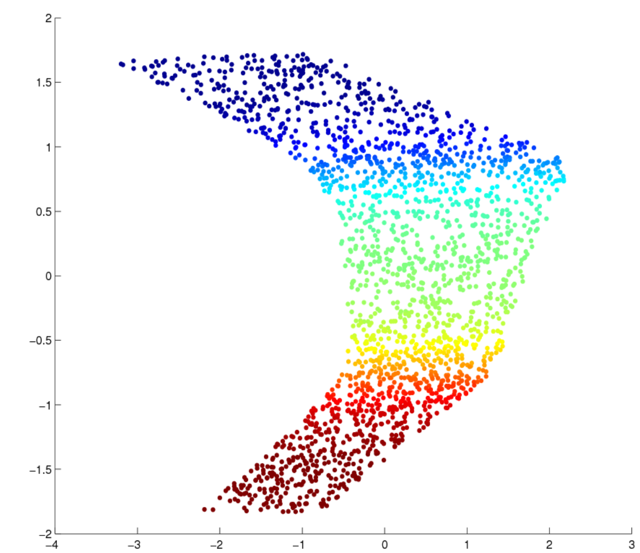
S'



5-NN

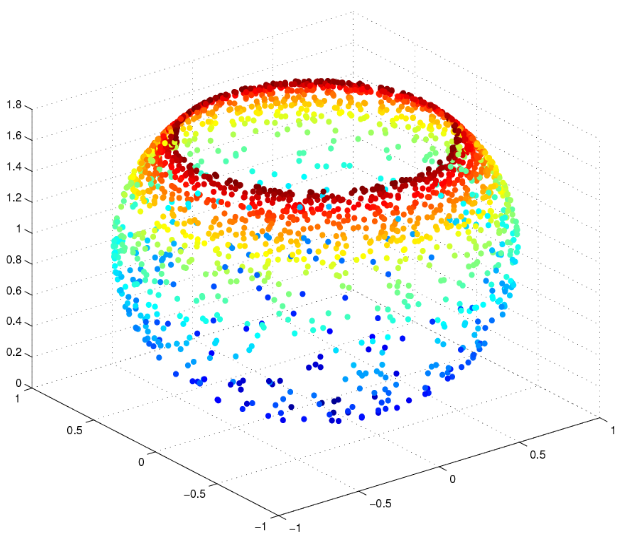


12-NN

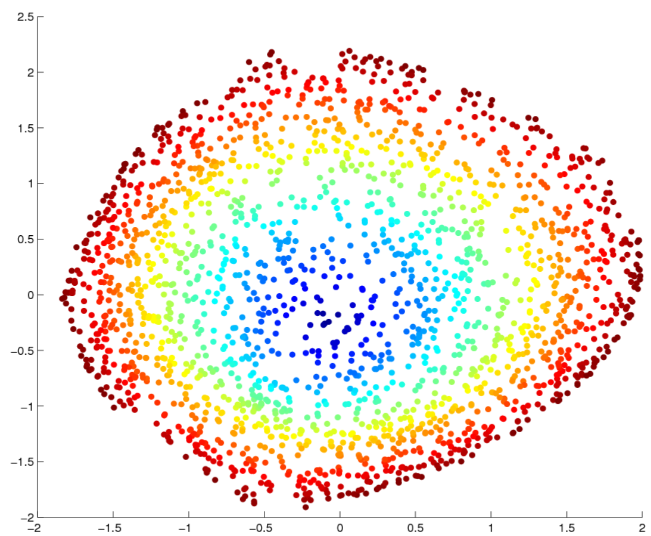


30-NN

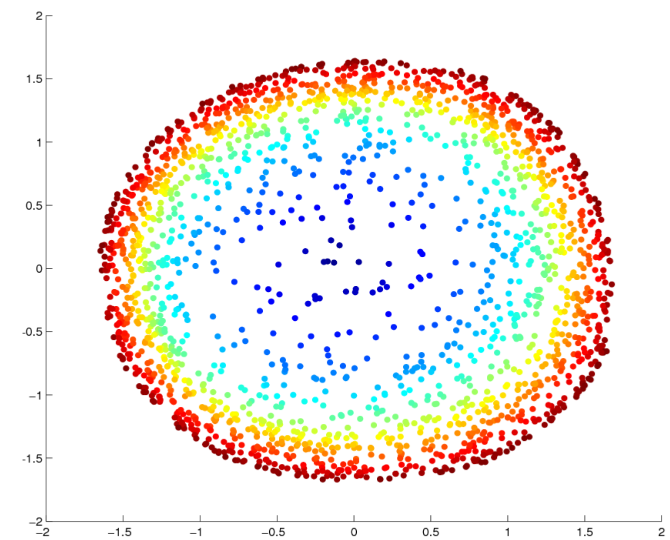
LLE: examples



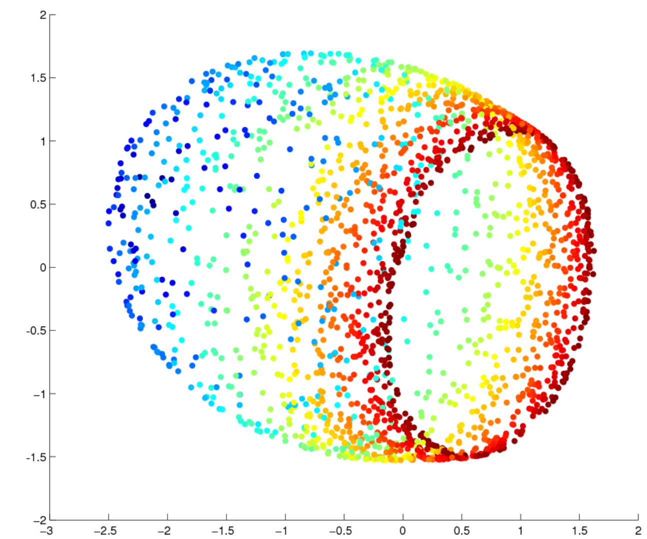
Fishbowl



12-NN

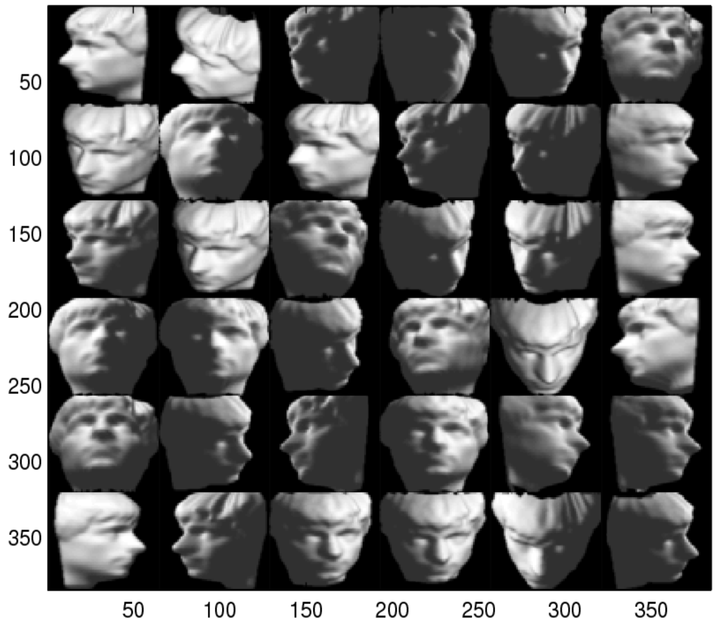


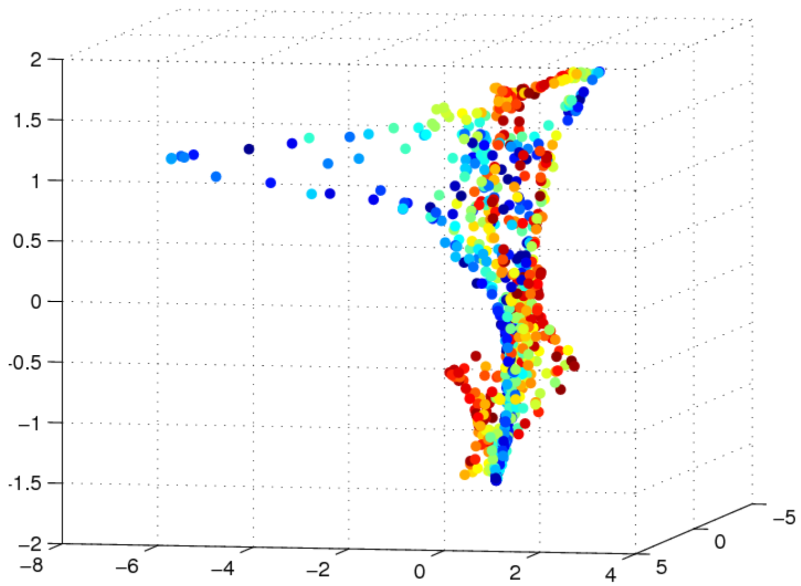
30-NN



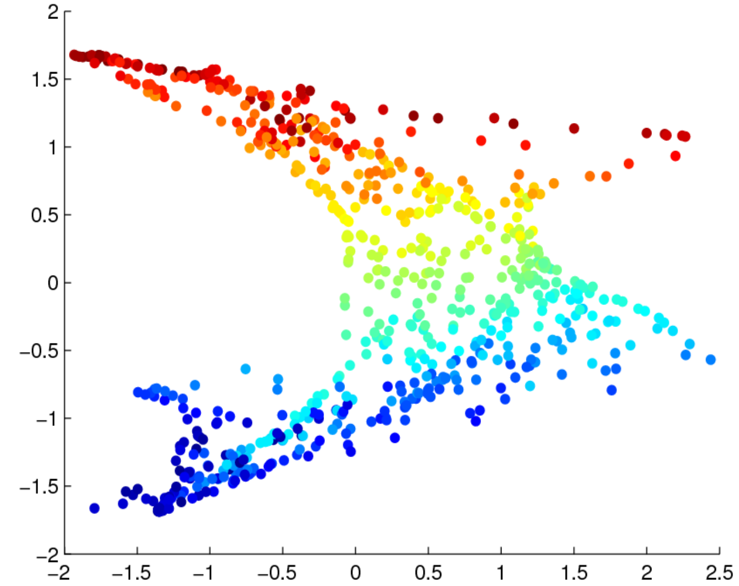
80-NN

LLE: example





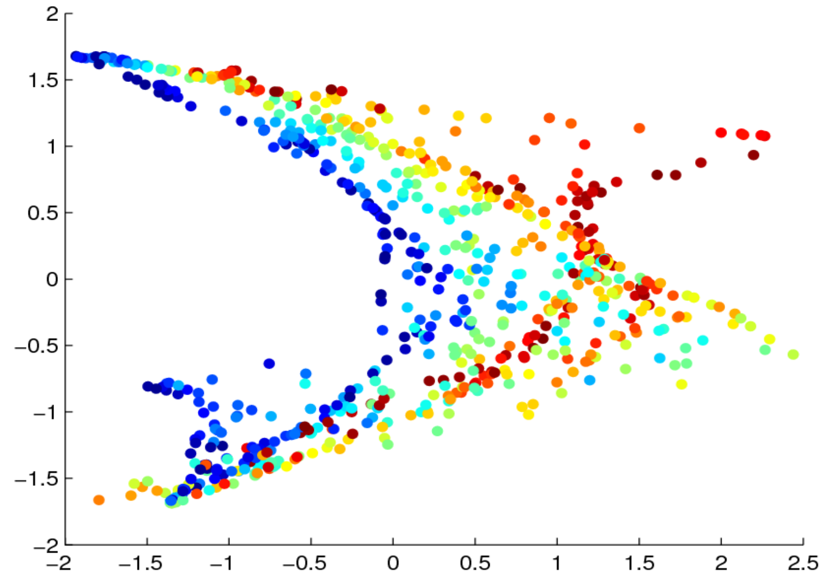
3D-proj: light



2D-proj: pose 1

$k = 6$ NN

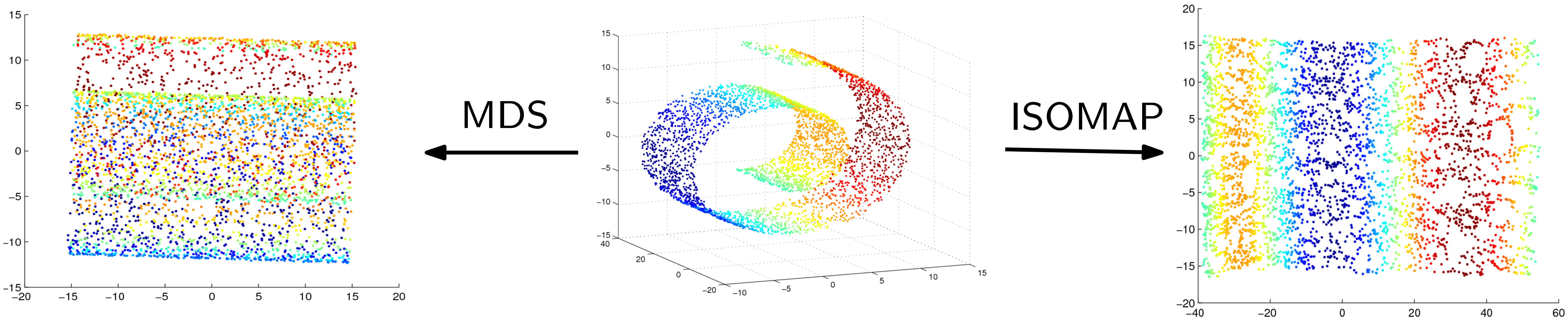
Dimension: $64 * 64 = 4096$.
$N = 698$



2D-proj: pose 2

3 free parameters:
- left-right pose,
- up-down pose,
- light pose.

ISOMAP (de Silva, Tenenbaum, Langford '00)



Variant of MDS where the matrix of Euclidean distances between data points is replaced by the matrix of the geodesic distances between data points.

Algorithm:

1. Build a neighborhood graph $\mathcal{G}$ with vertex set X such that the *geodesic* distances on $\mathcal{G}$ approximates the *geodesic* distances on M.

2. Build the matrix $D_{\mathcal{G}} = (d^2_{\mathcal{G}}(x_i, x_j))$ of the pairwise squared distances in $\mathcal{G}$.

3. Apply MDS to $D_{\mathcal{G}}$.

Geodesic distance approximation

The geodesic distance between x_i and x_j

$$d_M(x_i, x_j) = \inf\{l(\gamma)|\gamma : [0,1] \to M, \gamma(0) = x_i, \gamma(1) = x_j\}$$

in the manifold M is approximated by the length $d_{\mathcal{G}}(x_i, x_j)$ of the shortest path between x_i and x_j in $\mathcal{G}$.

for all $y \in M$ there exists $x \in X$ s.t. $d_M(x, y) < \delta$.

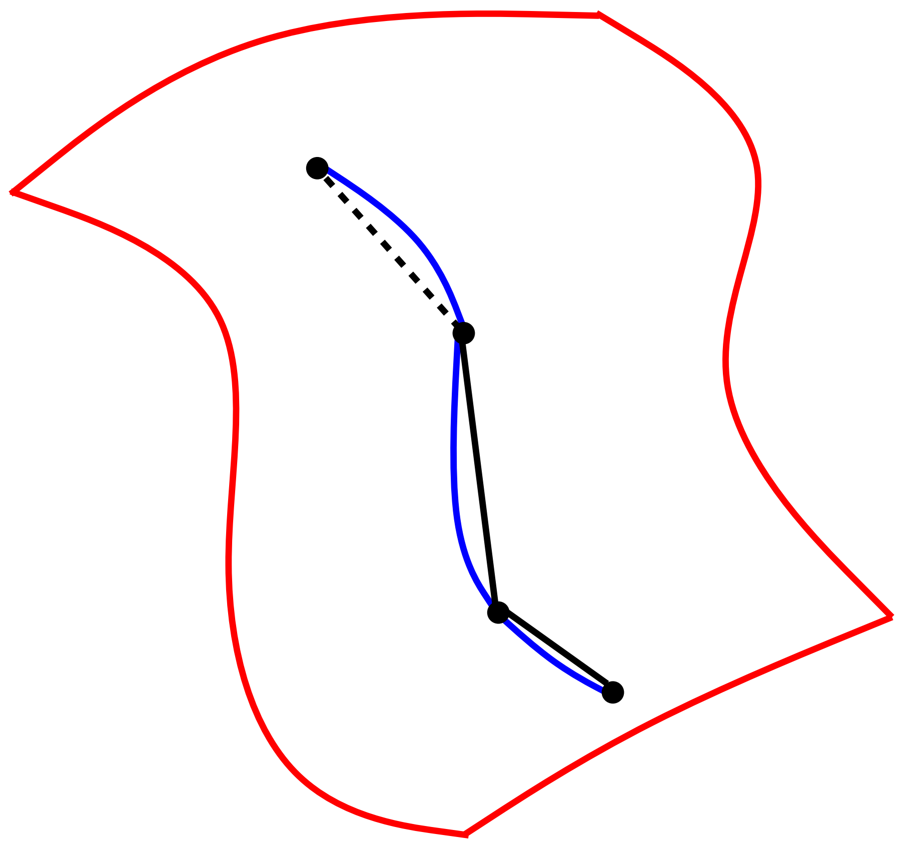
Theorem: [Bernstein & al'00] Let $\lambda > 0$. For some small enough $\delta, \varepsilon > 0$ ($\delta < \varepsilon$), if X is a δ-sample of M and if $\mathcal{G}$ is such that $(d(x_i, x_j) < \varepsilon \Leftrightarrow (x_i x_j)$ *is an edge of* $\mathcal{G})$ then for all x_i, x_j

$$1 - \lambda < \frac{d_{\mathcal{G}}(x_i, x_j)}{d_M(x_i, x_j)} < 1 + \lambda$$

Geodesic distance approximation

Sketch of proof:

Let $d_S(x, x') = \min_P \sum_{j=0}^{p-1} d_M(x_j, x_{j+1})$ where $P = (x_{i_0} = x, x_{i_1}, \cdots x_{i_p} = x') \subset X$.

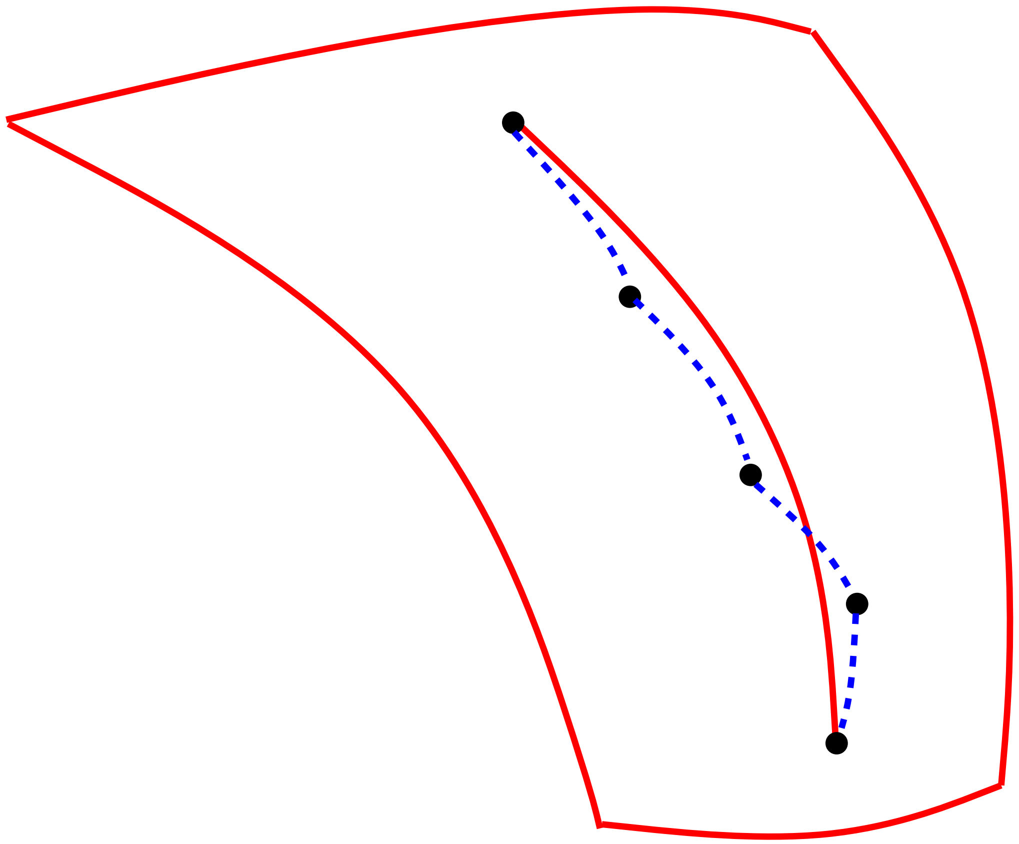


- From a distance function property (Federer):

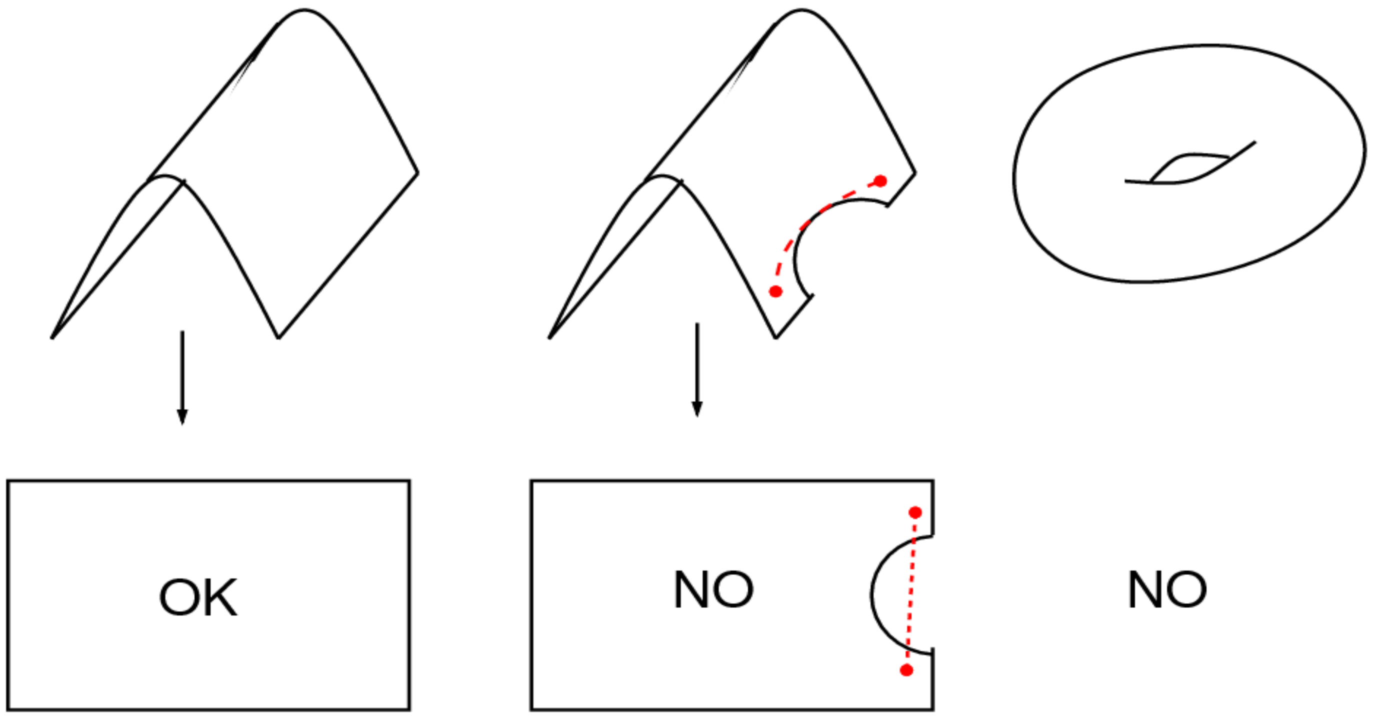
$$\frac{R - \varepsilon}{R} d_S \leq d_{\mathcal{G}} \leq d_S, \;\; R = reach(M)$$

- Using that X is a δ-sample of M one "approximately" gets

$$d_M \leq d_S \leq \frac{\varepsilon}{\varepsilon - 2\delta} d_M$$



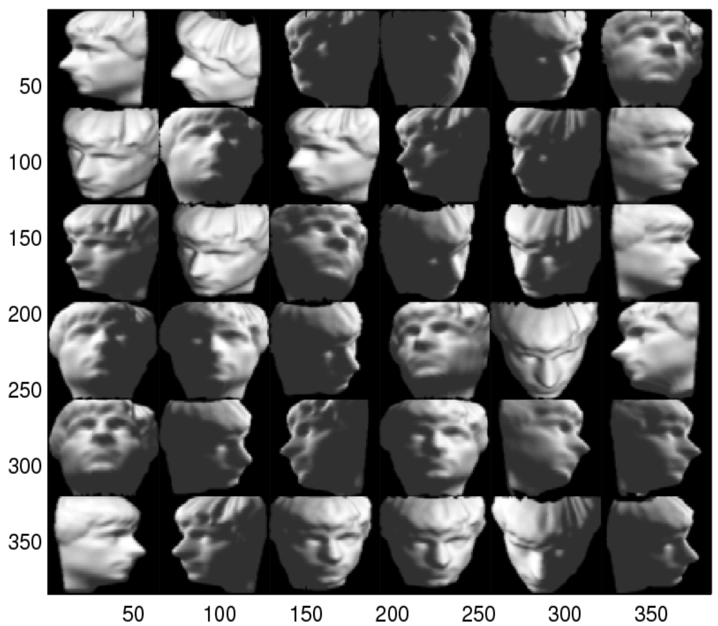
Theoretical guarantees of ISOMAP



ISOMAP intends to map X into $Y \subset \mathbb{R}^d$ in such a way that the pairewise geodesic distances in X are as close as possible to the pairwise euclidean distances in Y

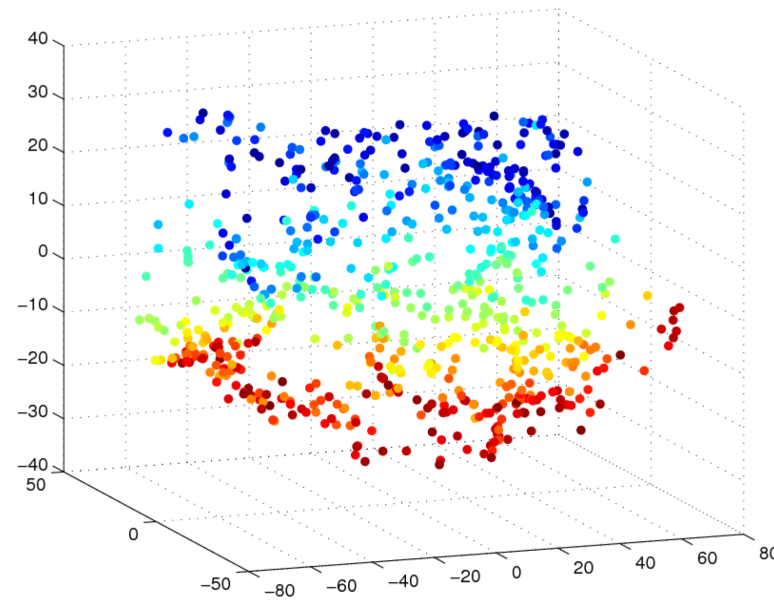
$\Longrightarrow M$ has to be isometric to a convex open subset of $\mathbb{R}^d$, i.e. there exists a convex open Ω domain in $\mathbb{R}^d$ and an embedding $f : \Omega \to \mathbb{R}^d$ s.t. $f(\Omega) = M$ and for all $y, y' \in \Omega$, $d_M(f(y), f(y')) = \|x - x'\|$.

ISOMAP: examples

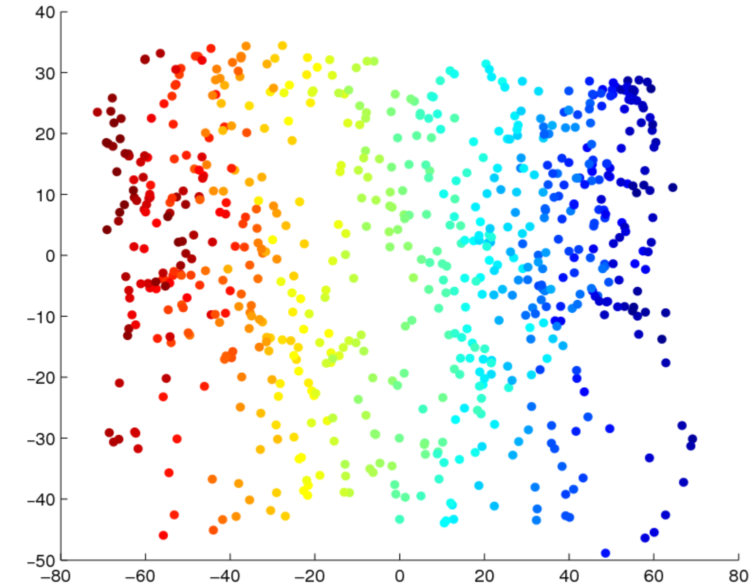


Dimension: $64 * 64 = 4096$.
$N = 698$

3 free parameters:
- left-right pose,
- up-down pose,
- light pose.

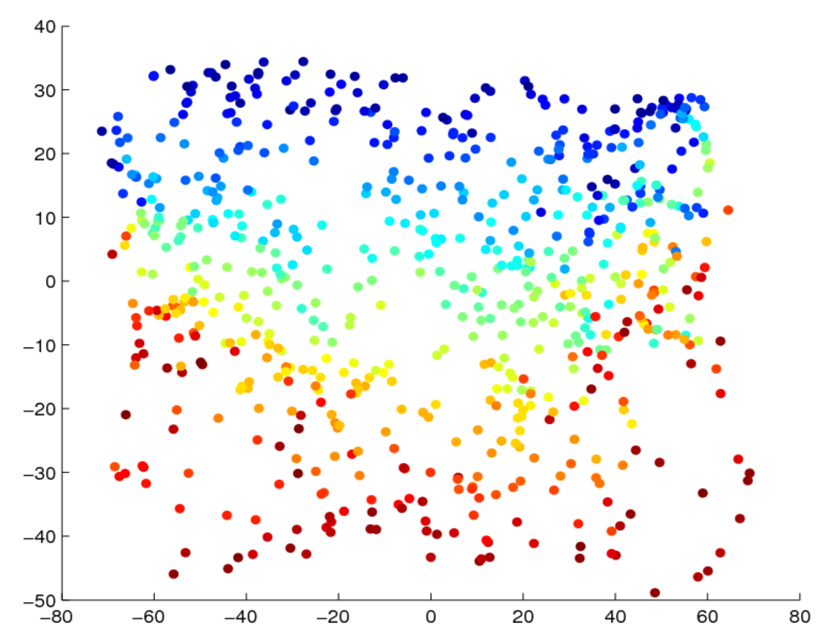


3D-proj: light



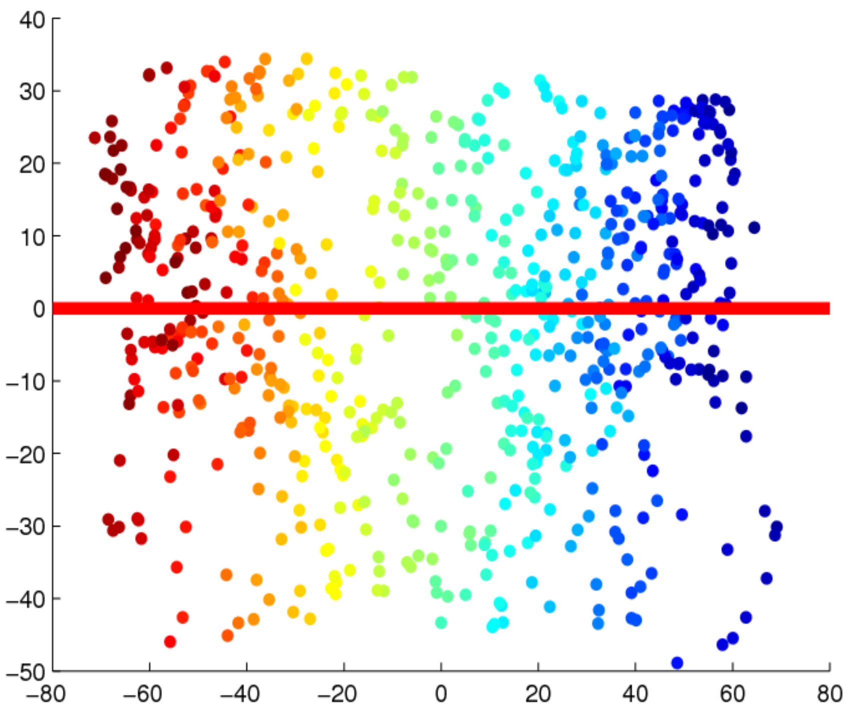
2D-proj: pose 1

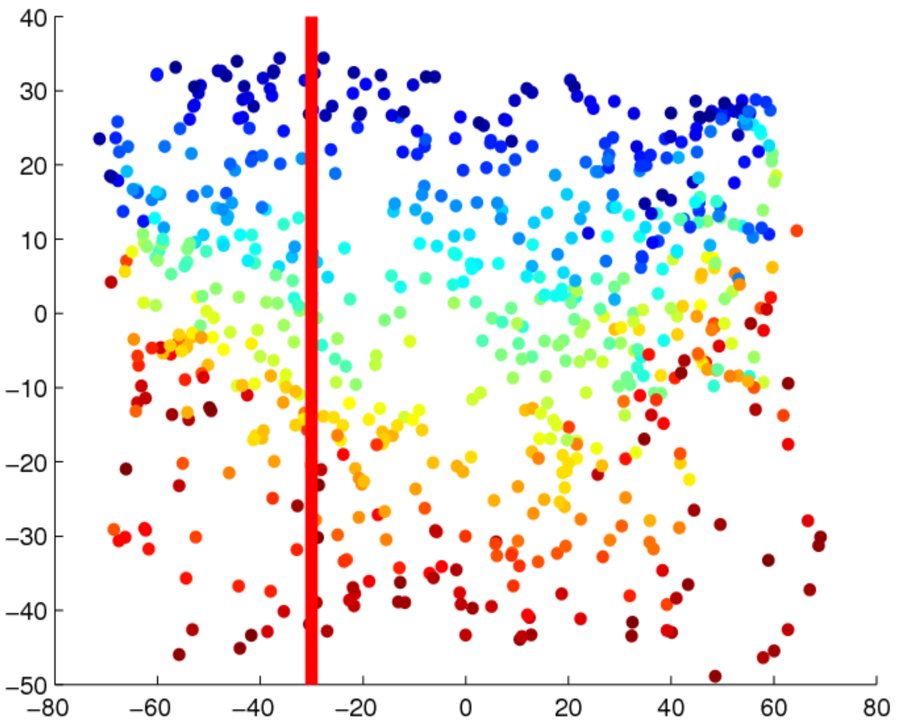
$k = 6$ NN

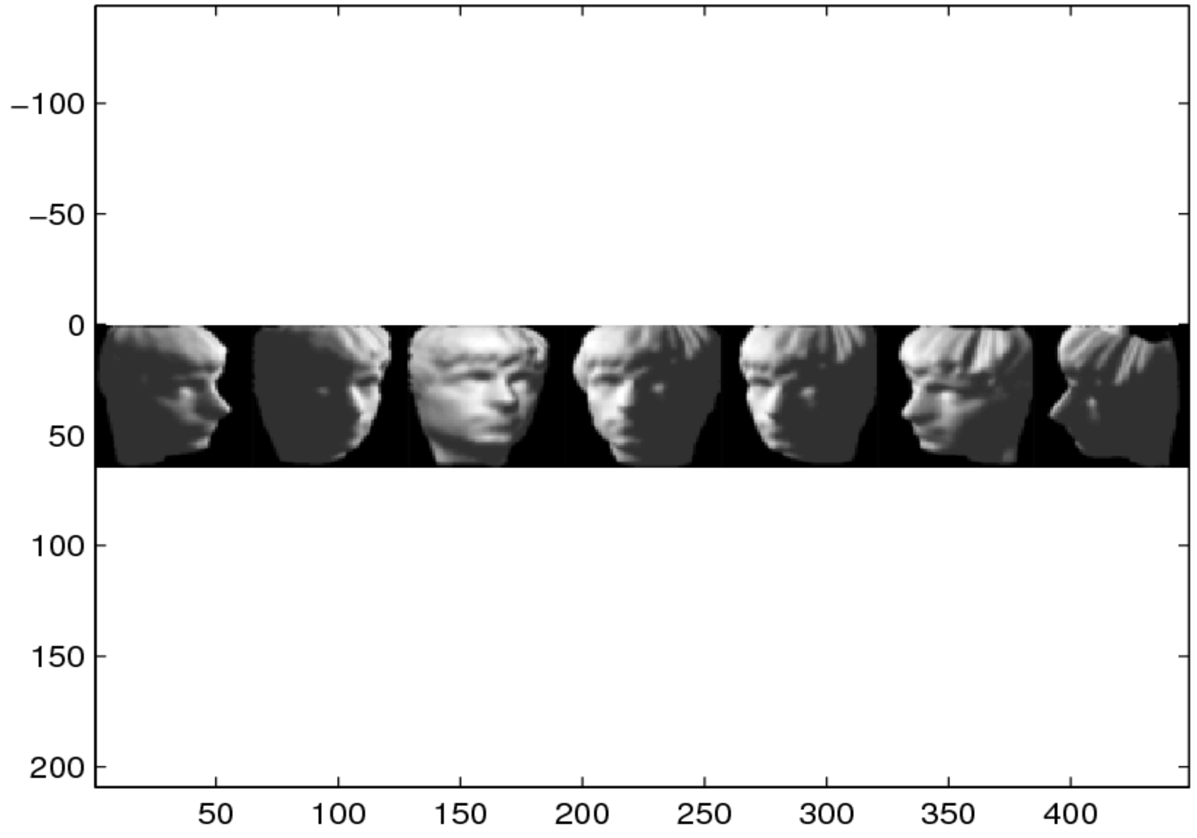


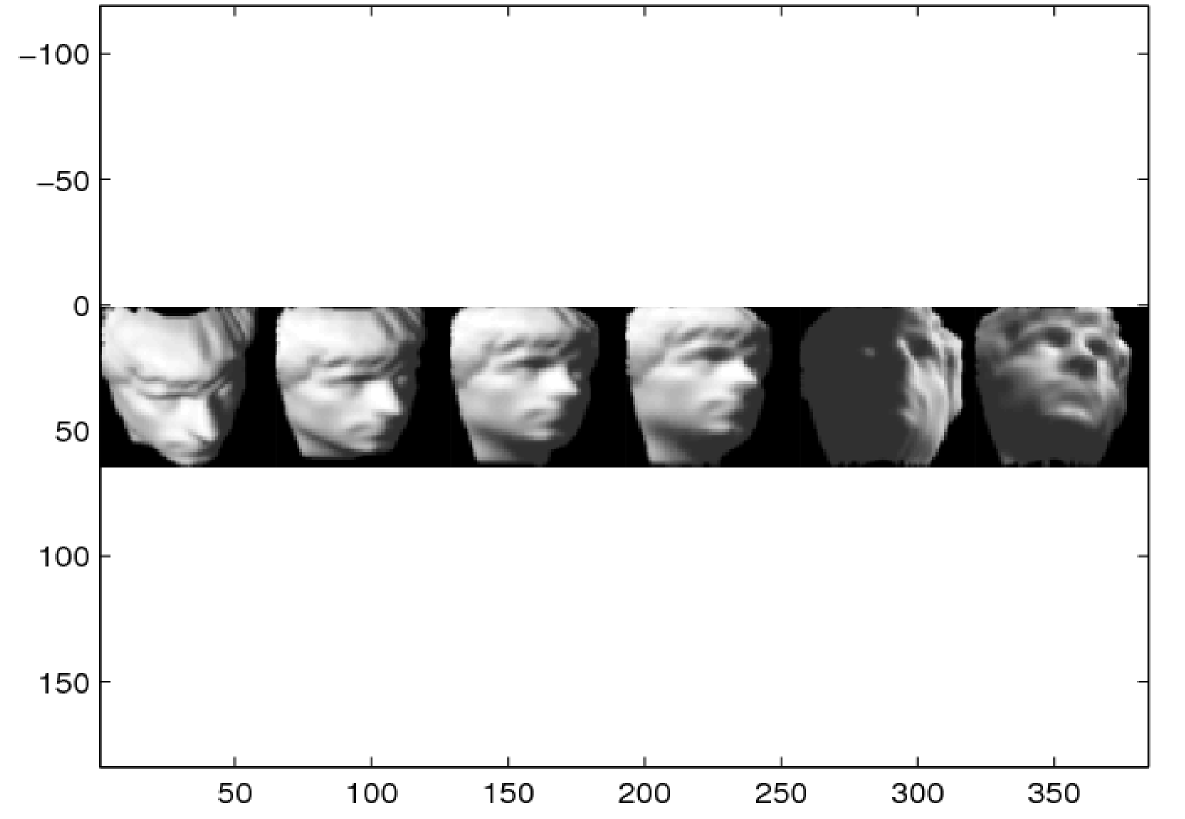
2D-proj: pose 2

ISOMAP: examples









ISOMAP: remarks

Advantages:

- intend to preserve the "intrinsic metric" of the data.
- come with geometric guarantees

Drawbacks:

- ISOMAP is a global method: as in MDS, if the size of the data is very large, the computations of the eignevalues/eigenvectors of $G = -0.5JDJ$ is an issue.
  $\Longrightarrow$ Landmark ISOMAP
- Assuming that $M \subset \mathbb{R}^D$ is isometric to a convex open set of $\mathbb{R}^d$ is rather restrictive.
  $\Longrightarrow$ Conformal ISOMAP
  $\Longrightarrow$ Hessian eigenmaps (HLLE)

Landmark ISOMAP (de Silva, Tenebaum)

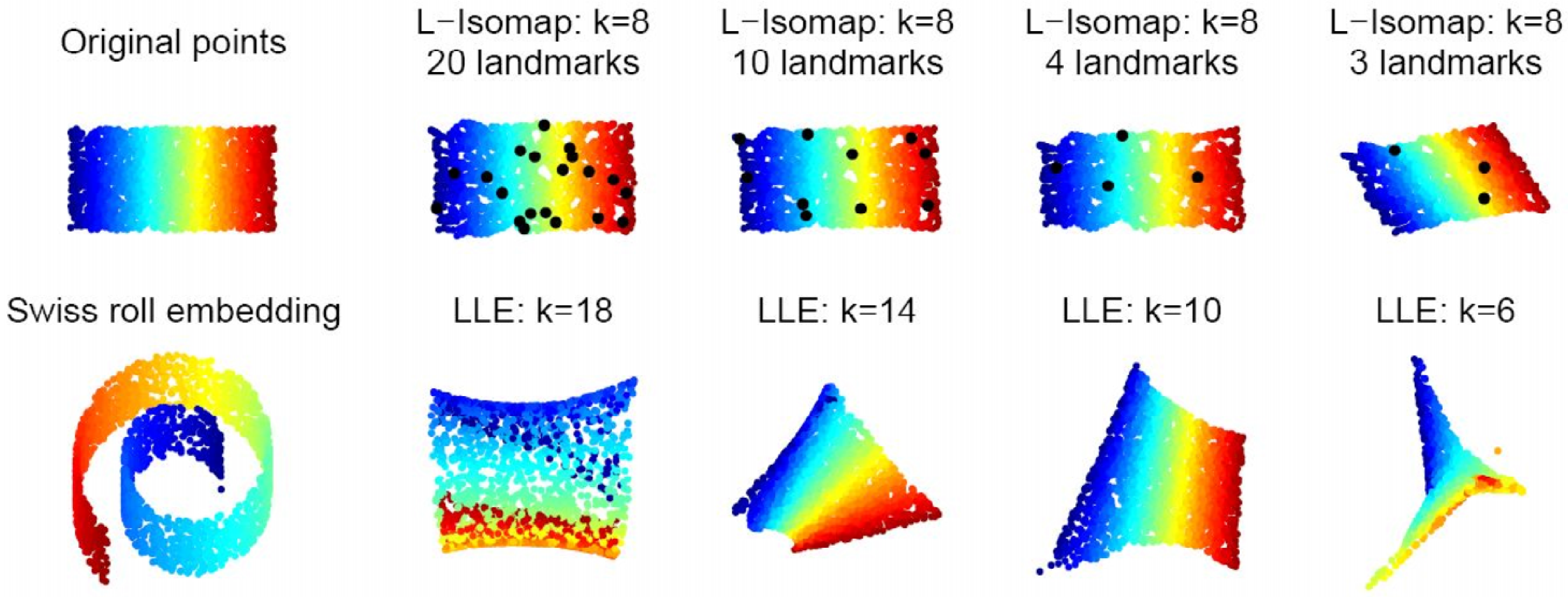
Select $n > d$ landmarks among the data points and compute the $n \times N$ matrix $D_{n,N}$ of the squared distances from each data point to the landmarks.

Replace classical MDS by a Landmark-MDS:

- Compute the matrix D_n of the squared distances between the landmarks and $G_n = -\frac{1}{2} J D_n J$.
- The embedding of the landmarks in $\mathbb{R}^d$ is given by (classical) MDS, i.e. by the $n \times d$ matrix $Y_n^T = (\sqrt{\lambda_1}\mathbf{v}_1 \sqrt{\lambda_2}\mathbf{v}_2 \cdots \sqrt{\lambda_d}\mathbf{v}_d)$ where λ_i and $\mathbf{v}_i$ are the largest eigenvalues/vectors of G_n.
- Embed the remaining points in the following way: for $x \in X$, let D_x be the vector of the distances between x and the n landmarks and let $\overline{D}_n$ be the vector of the mean of the columns of D_n. Then x is sent to

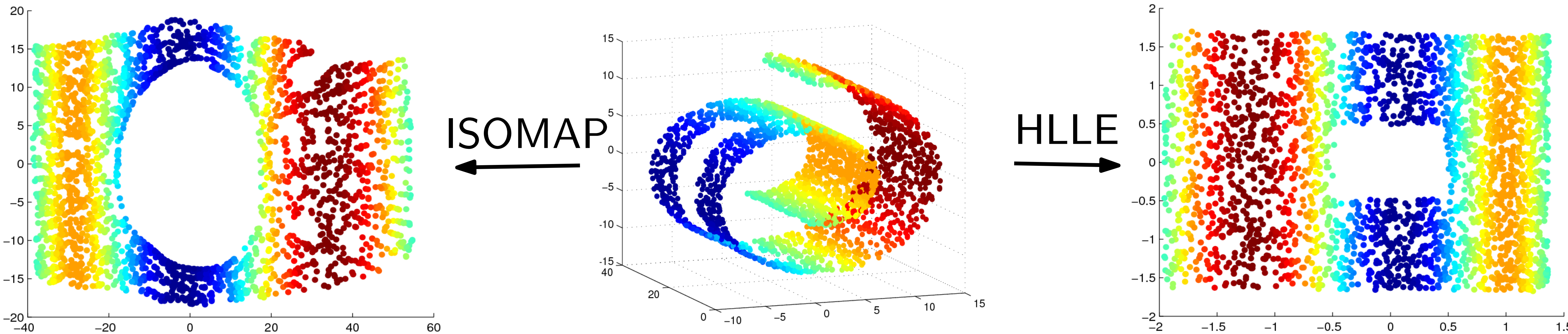
$$y = \frac{1}{2} L^{\#}(\overline{D}_n - D_x) \text{ where } L^{\#} = \begin{pmatrix} \mathbf{v}_1^T / \sqrt{\lambda_1} \\ \mathbf{v}_2^T / \sqrt{\lambda_2} \\ \vdots \\ \mathbf{v}_d^T / \sqrt{\lambda_d} \end{pmatrix}$$

Landmark ISOMAP: example



Results from V. de Silva, J.B. Tenenbaum, NIPS 15, 2003

Hessian eigenmaps (HLLE) (D. Donoho, C. Grimes '03)



A "proven" method for isometric embeddings of open sets of euclidean spaces:

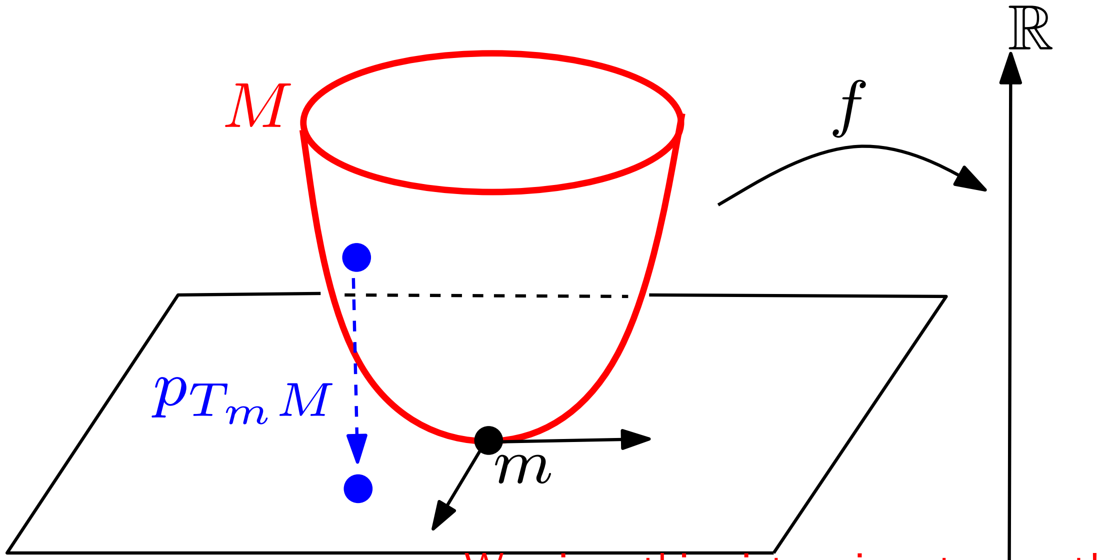
- $M = \psi(\Omega)$, $\psi : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$ isometry and Ω does not need to be convex....

- Rely on a (nice) property of a Hessian operator defined on the space of $\mathcal{C}^2$ functions on M.

- ... but it involves the estimation of 2^{nd} order differential quantities.

HLLE

$\Omega \subset \mathbb{R}^d$ be an open connected set and let $\psi : \Omega \to M$ be a smooth locally isometric embedding.



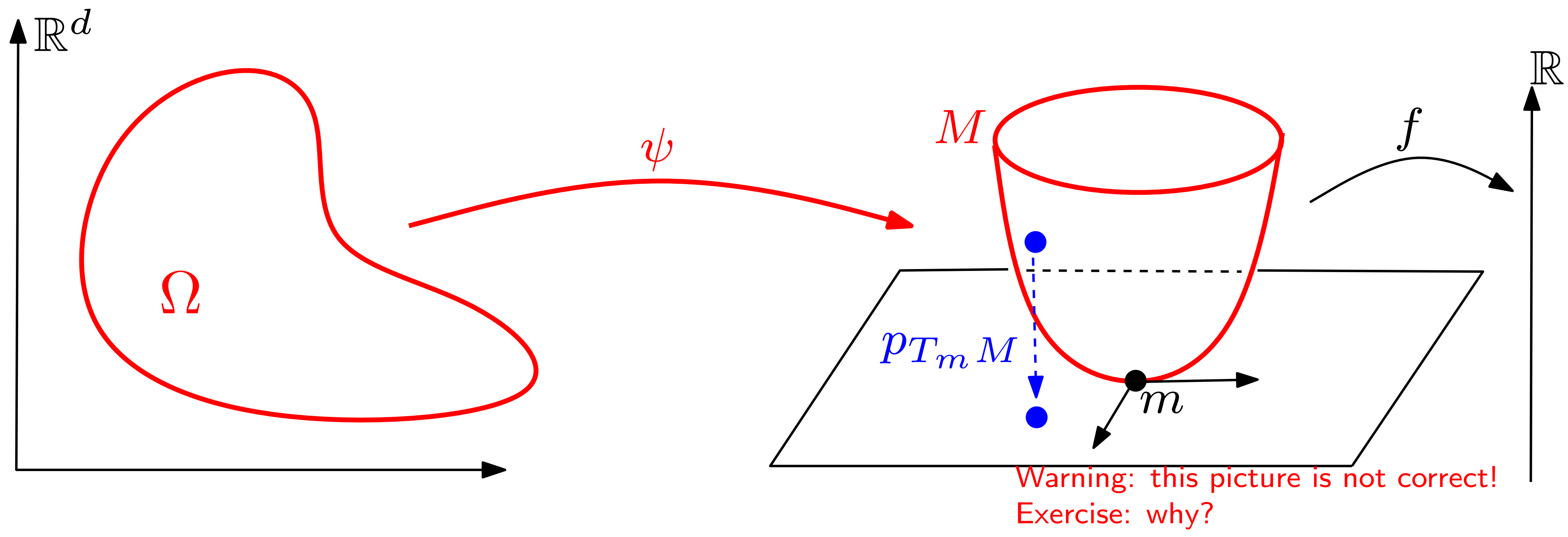
- Let $m \in M$, let $(x_1, \cdots, x_d)$ be an orthonormal coordinate system on $T_m M$. The projection $p_{T_m M}$ of M on $T_m M$ is well- defined on a neighborhood of m in M. For any $f \in \mathcal{C}^2(M, \mathbb{R})$, the Hessian of f at m in tangent coordinates is defined by

$$(H_f^{tan}(m))_{ij} = \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_j} f(p_{T_m M}^{-1}(x))|_{x=0}$$

- Consider the quadratic form on $\mathcal{C}^2(M, \mathbb{R})$ defined by

$$\mathcal{H}(f) = \int_M \|H_f^{tan}(m)\|^2 dm$$

HLLE

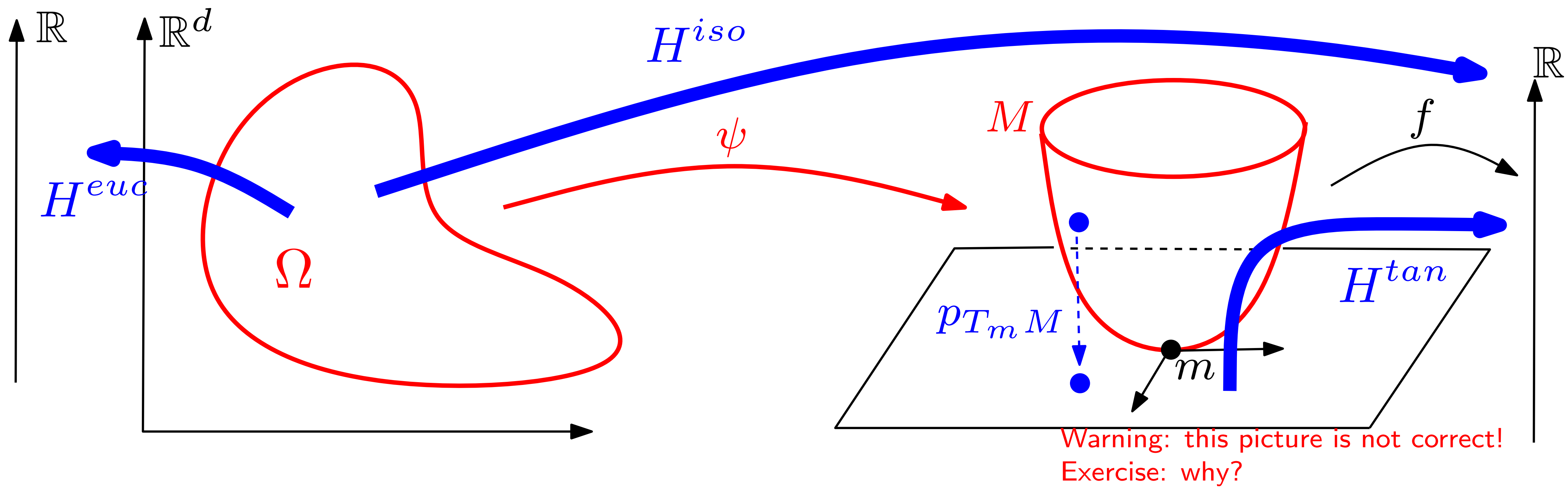


Theorem [Donoho et al. '03]: Assume that $M = \psi(\Omega)$ where $\Omega \subset \mathbb{R}^d$ is an open connected set and ψ is a locally isometric embedding of Ω. Then the null-space of the quadratic form

$$\mathcal{H}(f) = \int_M \|H_f^{tan}(m)\|^2 dm$$

is $(d+1)$-dimensional and generated by the constant functions and the d original isometric coordinates $pr_i \circ \psi^{-1}$ where $pr_i : \mathbb{R}^d \to \mathbb{R}$ is the linear projection on the i^{th} coordinate in $\mathbb{R}^d$.

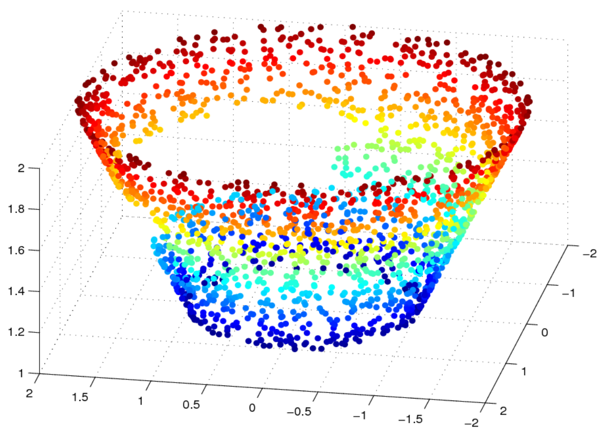
HLLE



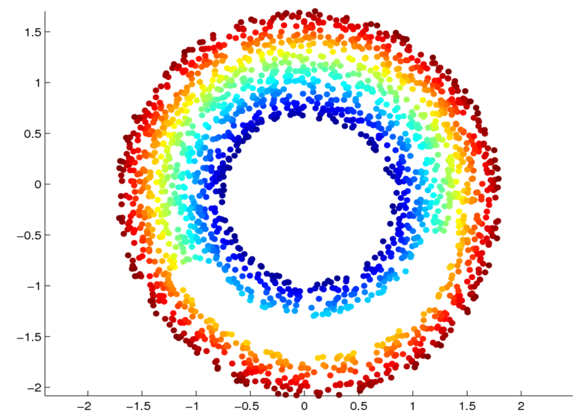
sketch of proof:

- $H_f^{iso} = H_{f\circ\psi}^{euc} \Rightarrow \mathcal{H}^{iso}(f) = \mathcal{H}^{euc}(f \circ \psi)$, $\forall f \in \mathcal{C}^2(M, \mathbb{R})$.
- The null-space of $\mathcal{H}^{euc}$ is the $(d+1)$-dimensional space of affine functions on $\mathbb{R}^d$.
- $H_f^{tan}(m) = H_f^{iso}(m)$:
  - let $v \in T_m M$ and let $\gamma_v : [0, \varepsilon) \longrightarrow M$ a unit speed geodesic s.t. $\gamma_v(0) = m$ and $\gamma_v'(0) = v$. Then $(f \circ \gamma_v)''(0) = v^T H_f^{iso}(m) v$.
  - let $\delta_v : [0, \varepsilon) \longrightarrow M$ defined by $\delta_v(t) = p_{T_m M}^{-1}(tv)$. Then $(f \circ \delta_v)''(0) = v^T H_f^{tan}(m) v$.
  - the accelerations of γ_v and δ_v at 0 are normal to $T_m M \Rightarrow |\gamma_v(t) - \delta_v(t)| = o(t^2) \Rightarrow (f \circ \gamma_v)''(0) = (f \circ \delta_v)''(0)$.

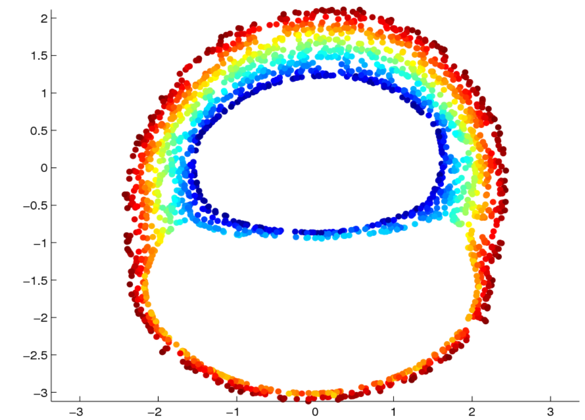
HLLE: examples



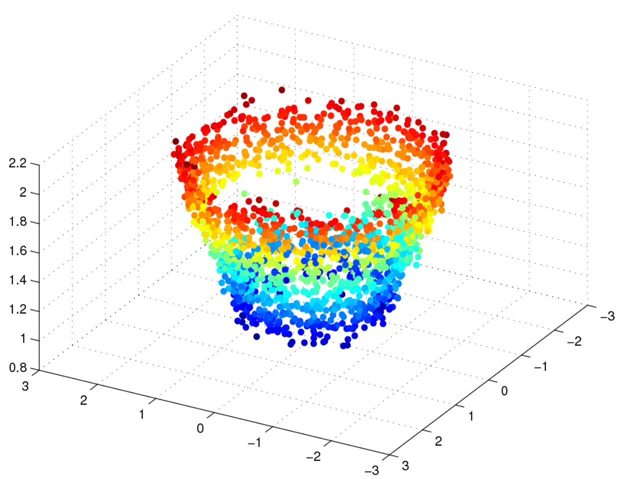
Cone



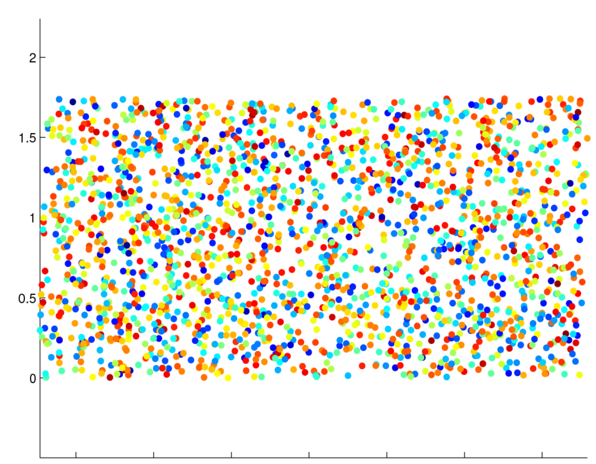
HLLE $(k = 12)$



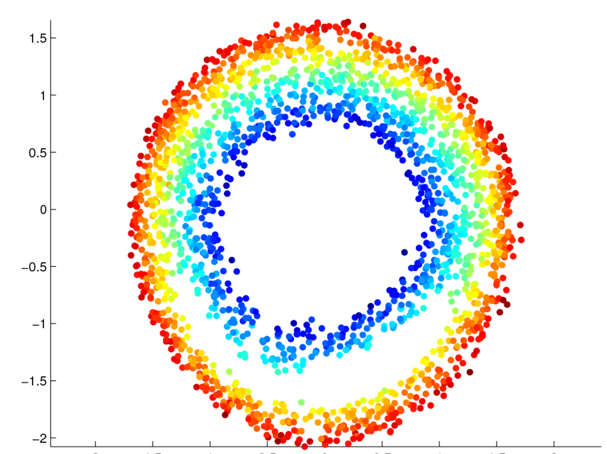
ISOMAP
$(k = 12)$



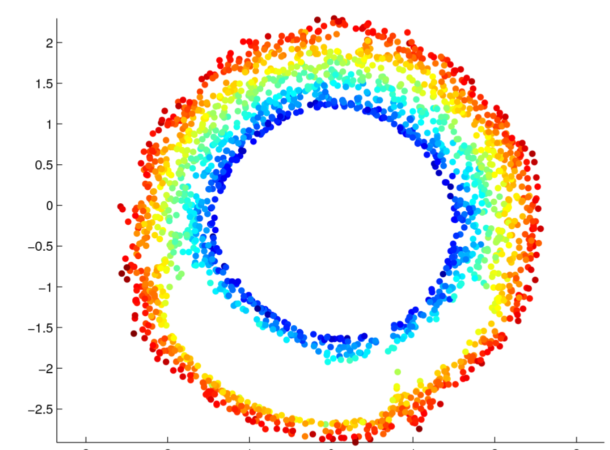
Noisy cone



HLLE $(k = 12)$

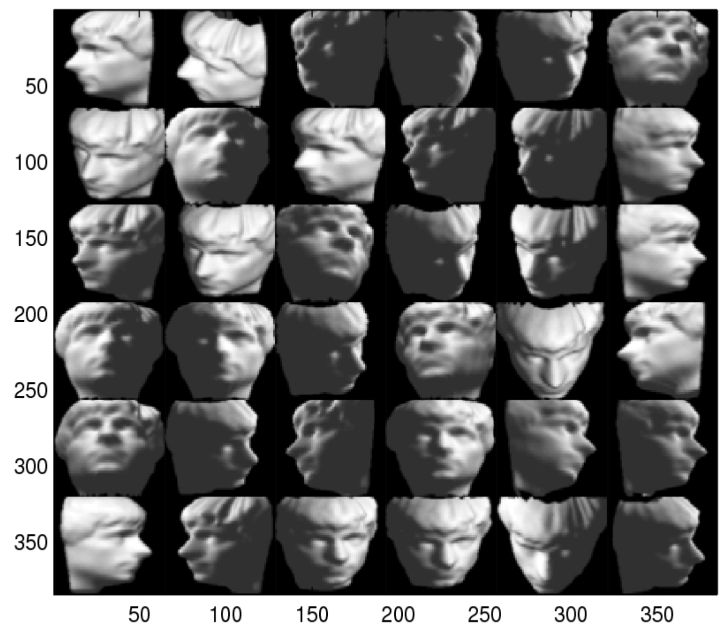


HLLE $(k = 20)$



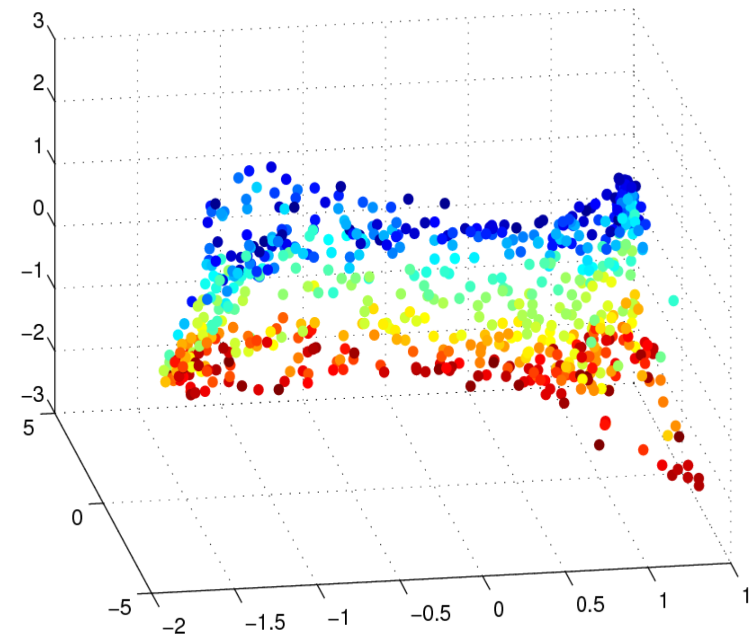
ISOMAP
$(k = 20)$

HLLE: examples

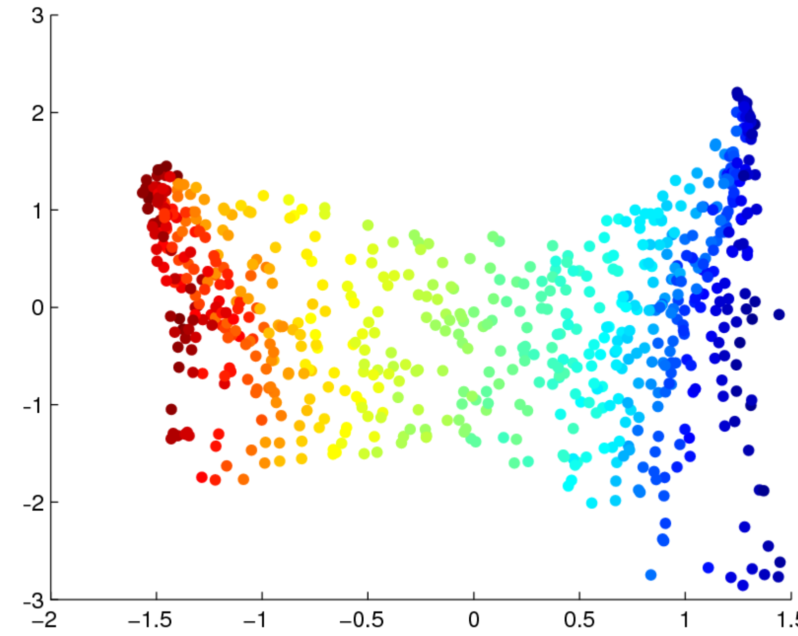


Dimension: $64 * 64 = 4096$.
$N = 698$

3 free parameters:
- left-right pose,
- up-down pose,
- light pose.

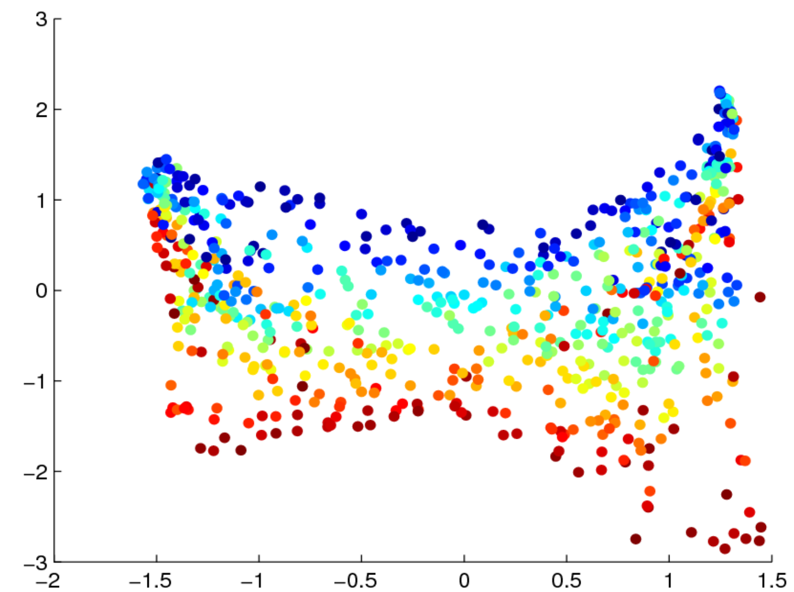


3D-proj: light



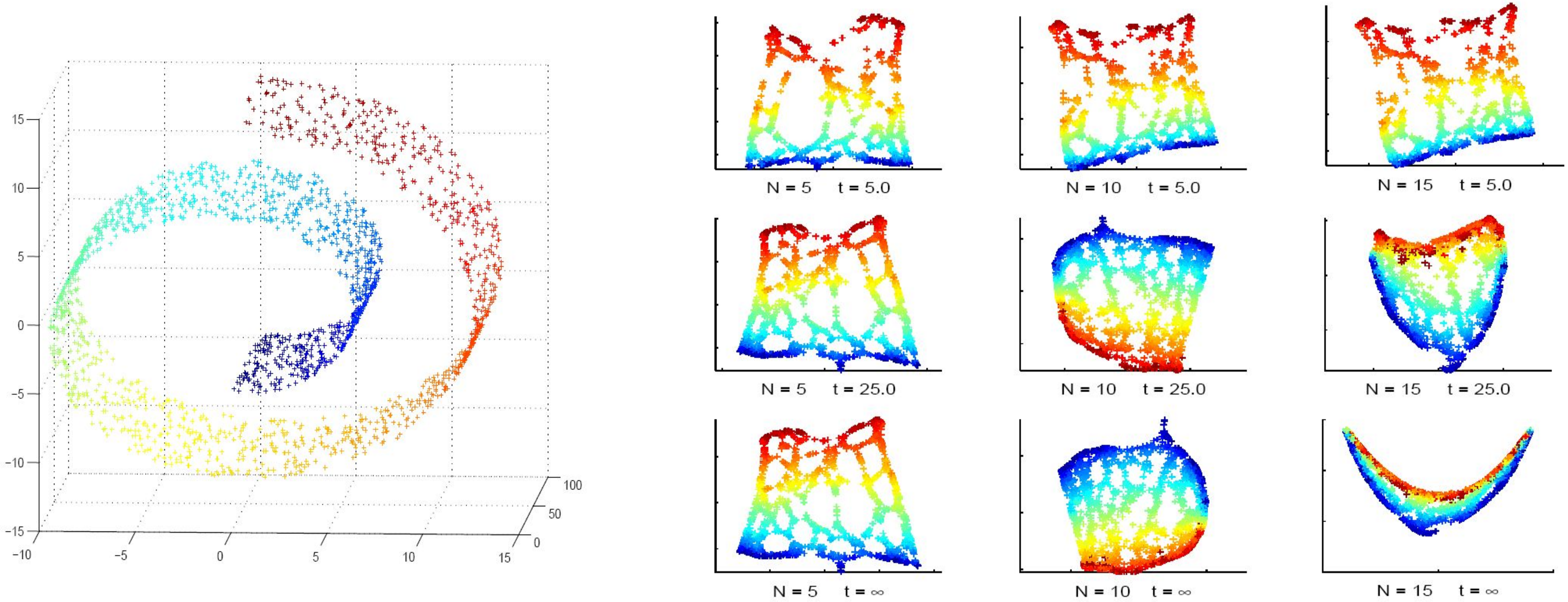
2D-proj: pose 1

$k = 12$ NN



2D-proj: pose 2

Laplacian eigenmaps (M. Belkin, P. Niyogi '02)



[from Belkin et al, Neural Computation, 2003; 15 (6):1373-1396]

- Laplacian eigenmaps intend to embed the data X in a d-dimensional in such a way that close/similar points in X remain close in the low dimensional space.
- Analogy with harmonic analysis on the underlying manifold.

Laplacian eigenmaps

Overview of the method:

1. Build a neighborhood graph $\mathcal{G}$ (e.g. k-NN or Rips).

2. Assign weights w_{ij} to the edges of $\mathcal{G}$ representing the "similarity" between the nodes:

   - Heat kernel: if $(x_i x_j)$ is an edge of $\mathcal{G}$ then

     $$w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}}$$

     $w_{ij} = 0$ otherwise.

   - Simple-minded ($t = +\infty$): $w_{ij} = 1$ if $(x_i x_j)$ is an edge of $\mathcal{G}$; $w_{ij} = 0$ otherwise.

3. Find $Y = \{y_1, \cdots y_N\} \subset \mathbb{R}^d$ that minimizes

   $$E = \sum_{i,j} \|y_i - y_j\|^2 w_{ij}$$

Laplacian eigenmaps

1-**dimensional case:** find $\mathbf{y}^T = (y_1, \cdots y_N)$ of X in $\mathbb{R}$ that minimize

$$E = \sum_{i,j} (y_i - y_j)^2 w_{ij}$$ (with somme additional constraints - see below)

Heavy penalty if close points in X are mapped far away

$$E = \sum_{i,j} (y_i^2 + y_j^2 - 2y_i y_j) w_{ij} = \sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2 \sum_{i,j} y_i y_j w_{ij} = 2\mathbf{y}^T L \mathbf{y}$$

with $L = D - W$ positive semidefinite.

- $\Rightarrow$ add a constraint to remove a scaling factor (and avoid obvious solution): $\mathbf{y}^T D \mathbf{y} = 1$ (use D rather than Id to reflect the respective importance of the vertices in $\mathcal{G}$).
- $\Rightarrow$ $\mathbf{y}$ minimizing E is given by the smallest non zero eigenvalue solution to the generalized eigenvalue problem $L\mathbf{y} = \lambda D \mathbf{y}$ (note that the eigen-function corresponding to the eigenvalue 0 is the constant function $(1, \cdots, 1)$ mapping all the data points on a single point - corresponding constraint: $\mathbf{y}^T D (1, \cdots, 1)^T = 0$).

Laplacian eigenmaps

General case - Minimization of E:

$$E = \sum_{i,j} \|y_i - y_j\|^2 w_{ij} = ?$$

Laplacian eigenmaps

General case - Minimization of E:

$$E = \sum_{i,j} \|y_i - y_j\|^2 w_{ij} = Tr(Y^T L Y)$$

where D is diagonal with $D_{ii} = \sum_j w_{ij}$ and $L = D - W$ is the matrix of the *Laplacian operator* on $\mathcal{G}$.
Let $\mathbf{f}_0, \cdots, \mathbf{f}_d$ be the solutions of the generalized eigenvector problem

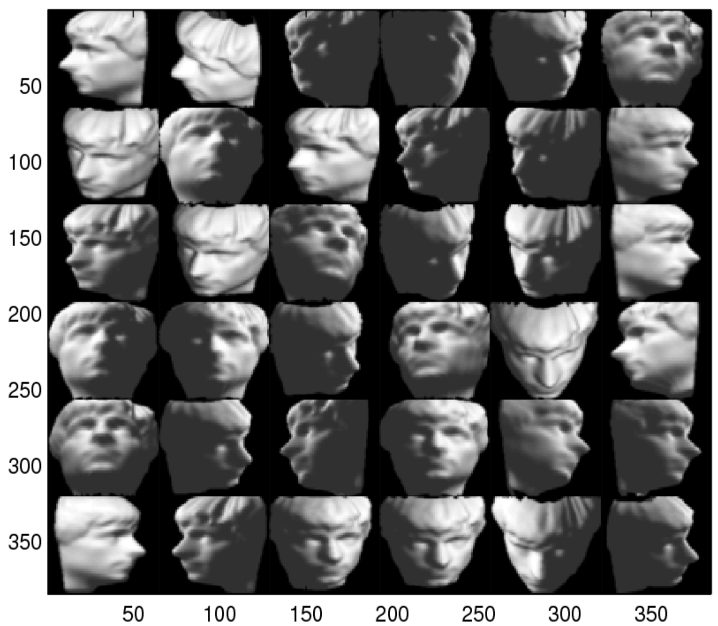
$$L\mathbf{f} = \lambda D\mathbf{f}$$

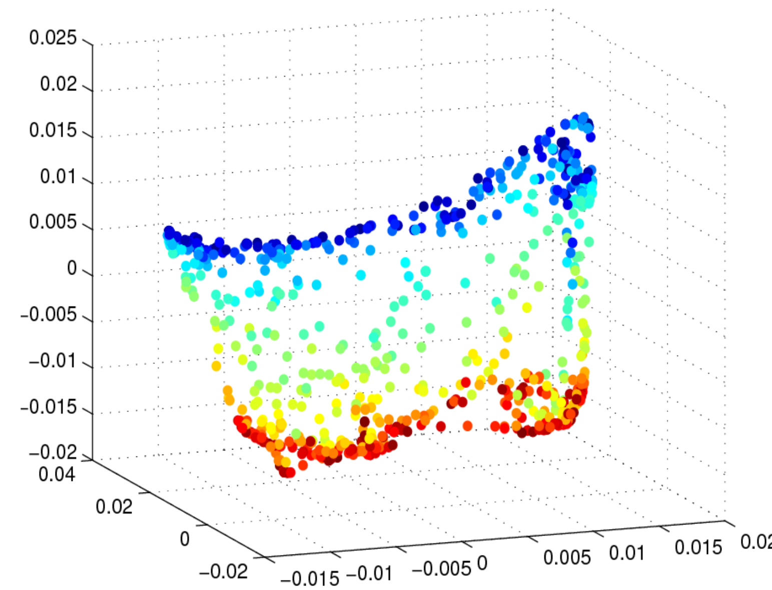
ordered according to increasing eigenvalues:

$$L\mathbf{f}_0 = \lambda_0 D\mathbf{f}_0, \cdots L\mathbf{f}_d = \lambda_d \mathbf{f}_d, \quad 0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_d$$

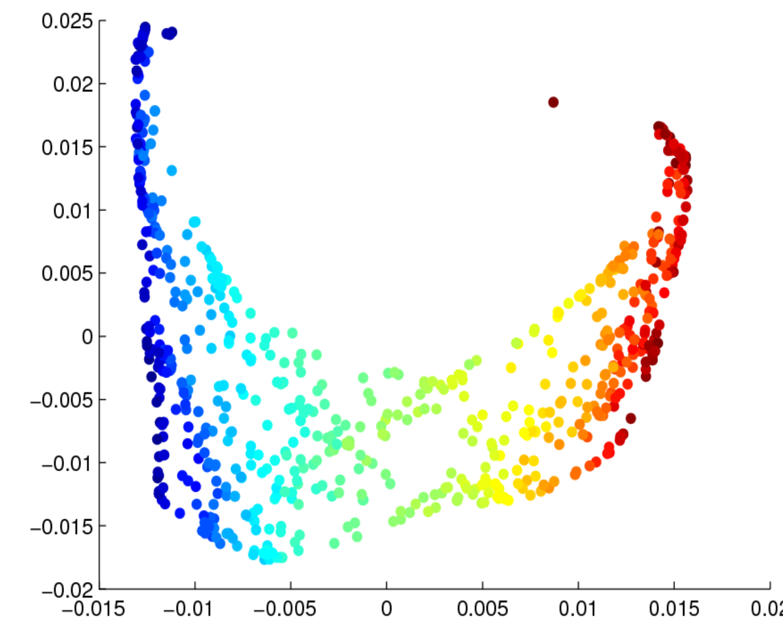
The embedding $y_i \in \mathbb{R}^d$ of x_i is given by $y_i = (\mathbf{f}_1(x_i), \cdots, \mathbf{f}_d(x_i))$ (Note that $\mathbf{f}_0$ corresponding to the eigenvalue 0 is discarded).

Laplacian eigenmaps: examples





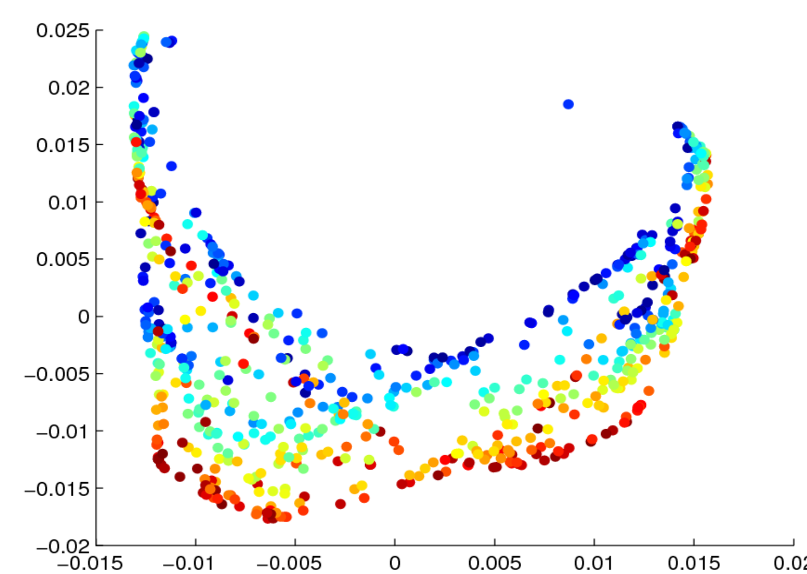
3D-proj: light



2D-proj: pose 1

Dimension: $64 * 64 = 4096$.
$N = 698$

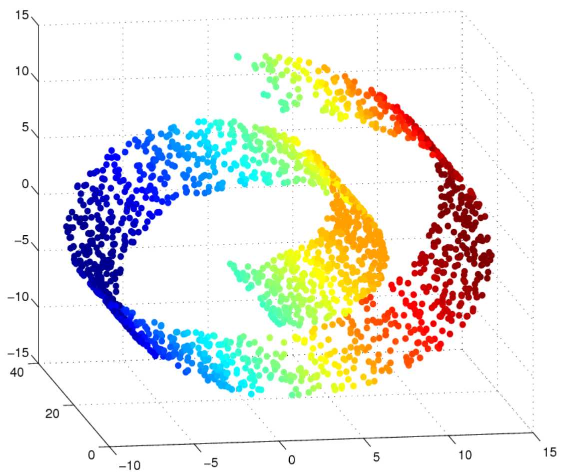
$k = 12$ NN, $t = 1$



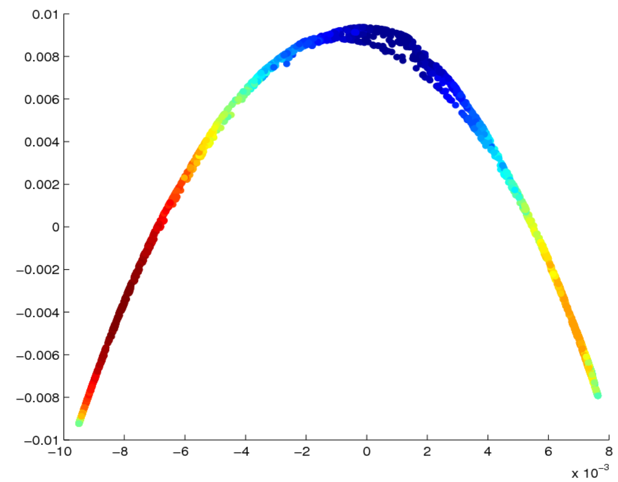
2D-proj: pose 2

3 free parameters:
- left-right pose,
- up-down pose,
- light pose.

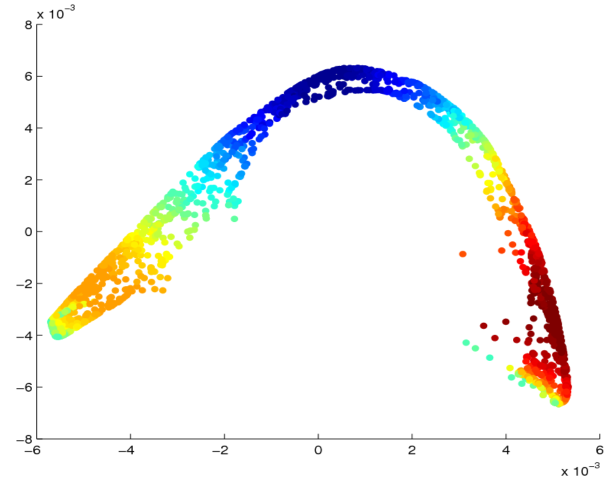
Laplacian eigenmaps: examples



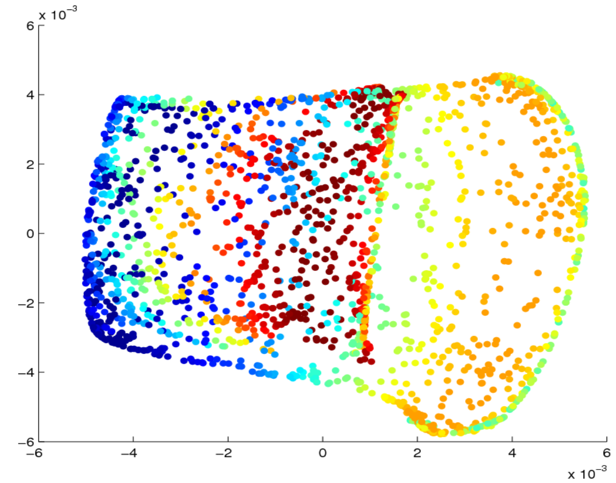
Swiss Roll



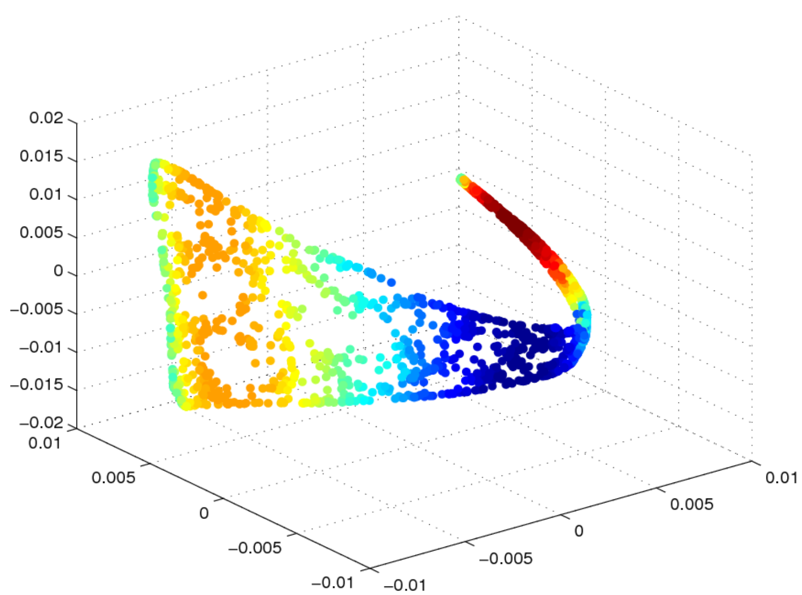
$k = 12$



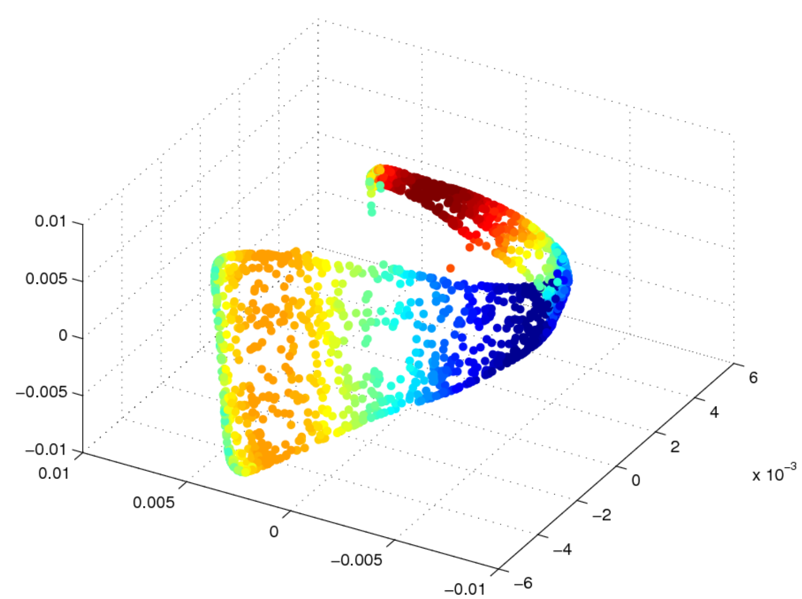
$k = 30$



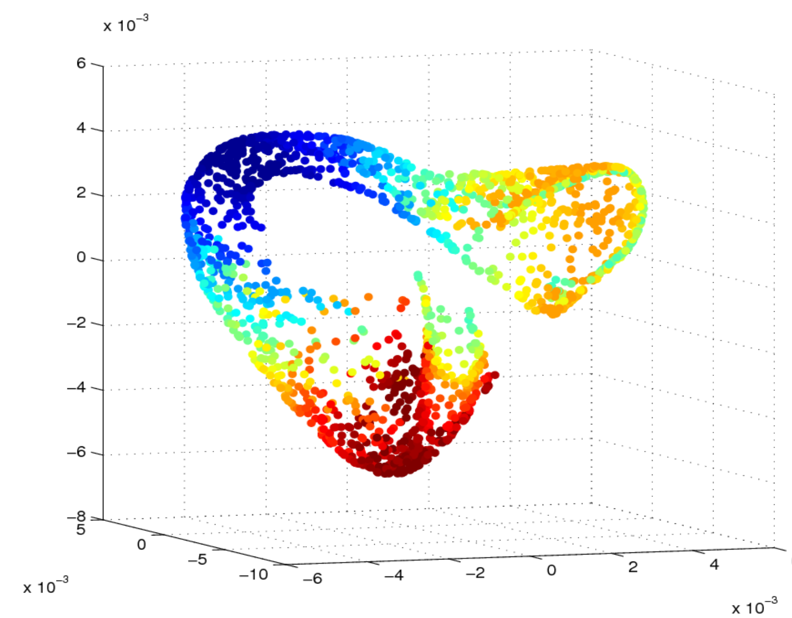
$k = 50$



$k = 12$



$k = 30$



$k = 50$

Analogy with the Laplace-Beltrami operator

Problem: Let M be a compact Riemannian d-manifold. Find the "best" map $f : M \to \mathbb{R}$ such that the points that are close together on M are mapped close together on $\mathbb{R}$.

Assuming that f is smooth, the way how close points are mapped far away by f is given by $\|\nabla f\|$. So the problem can be stated as find

$$argmin_{\{\|f\|_{L^2(M)}=1\}} \int_M \|\nabla f(m)\|^2 dm$$

Stokes' formula: for any vector field $\mathbf{X}$ on M, $\int_M \langle \mathbf{X}, \nabla f \rangle = -\int_M div(\mathbf{X}) f$

$$\Rightarrow \quad \int_M \|\nabla f(m)\|^2 = -\int_M div(\nabla f) f = \int_M \mathcal{L}(f) f$$

Laplace-Beltrami operator on M: $\mathcal{L}f := -div\nabla(f)$.

The solution is then given by the eigenfunction f_1 corresponding to the first non zero eigenvalue of $\mathcal{L}$.

Belkin, Niyogi'08: the analogy can be turned into a convergence result...

Choice of the weights

- Heat flow: $f : M \subset \mathbb{R}^D \to \mathbb{R}$ initial heat distribution, $u(x,t)$ heat distribution at time t ($u(x,0) = f(x)$).

- Heat equation: $(\frac{\partial}{\partial t} + \mathcal{L})u = 0$ has solution given by $u(x,t) = \int_M H_t(x,y)f(y)$, H_t being the heat kernel.

- $$\mathcal{L}f(x) = -\mathcal{L}u(x,0) = -\left(\frac{\partial}{\partial t}\int_M H_t(x,y)f(y)\right)_{t=0}$$

- for x, y close and t small,

$$H_t(x,y) \approx \frac{1}{(4\pi t)^{\frac{m}{2}}} e^{-\frac{\|x-y\|^2}{4t}} \quad \text{and} \quad \lim_{t\to 0}\int_M H_t(x,y)f(y) = f(x)$$

- Therefore, for t small,

$$\mathcal{L}f(x) \approx \frac{1}{t}\left(f(x) - \frac{1}{(4\pi t)^{\frac{m}{2}}}\int_M e^{-\frac{\|x-y\|^2}{4t}} f(y)dy\right)$$

Choice of the weights

- Therefore, for t small,

$$\mathcal{L}f(x) \approx \frac{1}{t}\left(f(x) - \frac{1}{(4\pi t)^{\frac{m}{2}}}\int_M e^{-\frac{\|x-y\|^2}{4t}} f(y)dy\right)$$

- For $x_i \in X$,

$$\mathcal{L}f(x_i) \approx \frac{1}{t}\left(f(x_i) - \frac{1}{N}(4\pi t)^{\frac{m}{2}} \sum_{j,\|x_i-x_j\|<\varepsilon} e^{-\frac{\|x_i-x_j\|^2}{4t}} f(x_j)\right)$$

- note that $\mathcal{L}c^{te} = 0 \Rightarrow (\frac{1}{N}(4\pi t)^{\frac{m}{2}})^{-1} = \sum_{j,\|x_i-x_j\|<\varepsilon} e^{-\frac{\|x_i-x_j\|^2}{4t}}$ and $\frac{1}{t}$ does not affect the eigendecomposition of the discrete laplacian.

$\Rightarrow$ Choice of the weights: $w_{ij} = e^{-\frac{\|x_i-x_j\|^2}{4t}}$ if $\|x_i - x_j\| < \varepsilon$; $w_{ij} = 0$ otherwise.

Diffusion maps (R. Coifman, S. Lafon, A. Lee, M. Maggioni,... '05)

Input: $X \subset \mathbb{R}^D$ and a weight function $w(x_i, x_j) = w_{ij}$ such that the matrix $W = (w_{ij})$ is symmetric and semi-definite positive.

- Let $d_i = \sum_j w_{ij}$ and let $p_{ij} = p(x_i, x_j) = \frac{w_{ij}}{d_i}$ = probability for a random walker on X to make a step from x_i to x_j (note that $\sum_j p_{ij} = 1$). The iterates $P^t = (p_t(x_i, x_j))$ of $P = (p_{ij})$ can be seen as the probabilities of going from x_i to x_j in t time steps.

- Diffusion operator:

$$Pf(x_i) = \sum_{j=1}^{N} p_{ij} f(x_j)$$

  It can be seen as an operator acting on the probability distributions $\mu^T = (\mu(x_1), \cdots \mu(x_N))$ on X

$$\mu^T P(x_j) = \sum_{i=1}^{N} \mu(x_i) p_{ij} \text{ with unique stationary dist. } \mu_0(x_i) = \frac{d_i}{\sum_k d_k}$$

Diffusion maps (R. Coifman, S. Lafon, A. Lee, M. Maggioni,... '05)

- The unique stationary distribution $\mu_0(x_i) = \frac{d_i}{\sum_k d_k}$ satisfies

$$\mu_0(x_i)p_{ij} = \mu_0(x_j)p_{ji}$$

- **Idea:** for a fixed time t, define a metric such that two points x_i, x_j are close if the conditional probability distributions $p_t(x_i, .)$ and $p_t(x_j, .)$ are close.

- Diffusion distance:

$$D_t^2(x_i, x_j) = \|p_t(x_i, .) - p_t(x_j, .)\|^2_{\frac{1}{\mu_0}} = \sum_k \frac{(p_t(x_i, x_k) - p_t(x_j, x_k))^2}{\mu_0(x_k)}$$

$\rightarrow$ Close connection with the spectral theory of the random walk.

- Left and right eigenvectors of P: $1 = |\lambda_0| \geq |\lambda_1| \geq \cdots \geq \lambda_{N-1}$

$$\mu_j^T P = \lambda_j \mu_j^T \quad \text{and} \quad P f_j = \lambda_j f_j \text{ with } f_j = \frac{\mu_j}{\mu_0}$$

Diffusion maps (R. Coifman, S. Lafon, A. Lee, M. Maggioni,... '05)

- Choose normalized μ_j, f_j: $\|\mu_j\|^2_{\frac{1}{\mu_0}} = 1$ and $\|f_j\|^2_{\mu_0} = \sum_k f_j(x_k)^2 \mu_0(x_k) = 1$.
- Biorthogonal decomposition of P^t:

$$p_t(x_i, x_j) = \sum_k \lambda_k^t f_k(x_i) \mu_k(x_j)$$

- This implies

$$D_t^2(x_i, x_j) = \sum_{k=1}^{N} \lambda_k^{2t} (f_k(x_i) - f_k(x_j))^2$$

Note that since $f_0 \equiv 1$, it does not enter into the sum.

- The diffusion distance is then approximated by

$$D_t^2(x_i, x_j) \approx \sum_{k=1}^{d} \lambda_k^{2t} (f_k(x_i) - f_k(x_j))^2$$

Diffusion maps (R. Coifman, S. Lafon, A. Lee, M. Maggioni,... '05)

- The diffusion distance is then approximated by

$$D_t^2(x_i, x_j) \approx \sum_{k=1}^{d} \lambda_k^{2t} (f_k(x_i) - f_k(x_j))^2$$

- Embedding of the data in $\mathbb{R}^d$:

$$x_i \mapsto y_i = (\lambda_1^t f_1(x_i), \cdots \lambda_d^t f_d(x_i))$$

The (approximated) diffusion metric becomes the euclidean metric between the data points in $\mathbb{R}^d$.

A few references

- LLE:
  - L. K. Saul, S. T. Roweis, "Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds", Journal of Machine Learning Research 4 (2000), 119-155.
- ISOMAP:
  - J.B. Tenenbaum, V. de Silva, J. C. Langford, "A global Geometric Framework for Nonlinear Dimensionality Reduction", Science 290: 2319-2323, 2000.
  - V. de Silva, J. B. Tenenbaum, "Global versus Local Methods in Nonlinear Dimensionality Reduction", Advances in Neural Information Processing Systems 15, MIT Press, 2003.
- HLLE:
  - D. L. Donoho, C. Grimes, "Heissian Eigenmaps: Locally Linear Embedding Techniques for High-dimensional Data", Proceedings of the National Academy of Sciences 100, 10, 5591-5596.

A few references

- Laplacian Eigenmaps:
  - M. Belkin, P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction for Data Representation", Neural Computation 15, 6, 1373–1396, 2003.
- Diffusion maps:
  - R. R. Coifman and S. Lafon and A. Lee and M. Maggioni and B. Nadler and F. Warner and S. Zucker, "Geometric Diffusions as a tool for Harmonic Analysis and structure definition of data: Diffusion maps", Proc. of Nat. Acad. Sci. 102, 7426–7431, 2005.

Two related geometric problems of fundamental importance

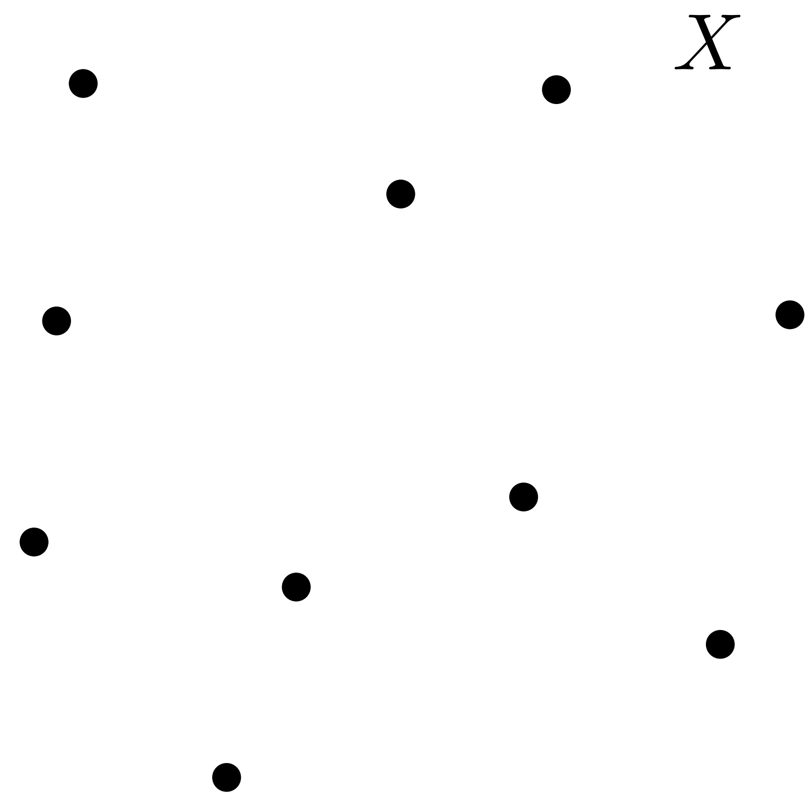
- **Nearest neighbors search:**
  - most of the previously presented methods rely on the construction of a neighborhood graph $\rightarrow$ being able to compute nearest neighbors is mandatory!
  - a fundamental problem for many applications.

- **Landmark selection/downsampling (quantization):**
  - when the size of the data becomes too large, it becomes necessary to subsample.
  - landmark selection must "preserve" the geometric structure of the data.

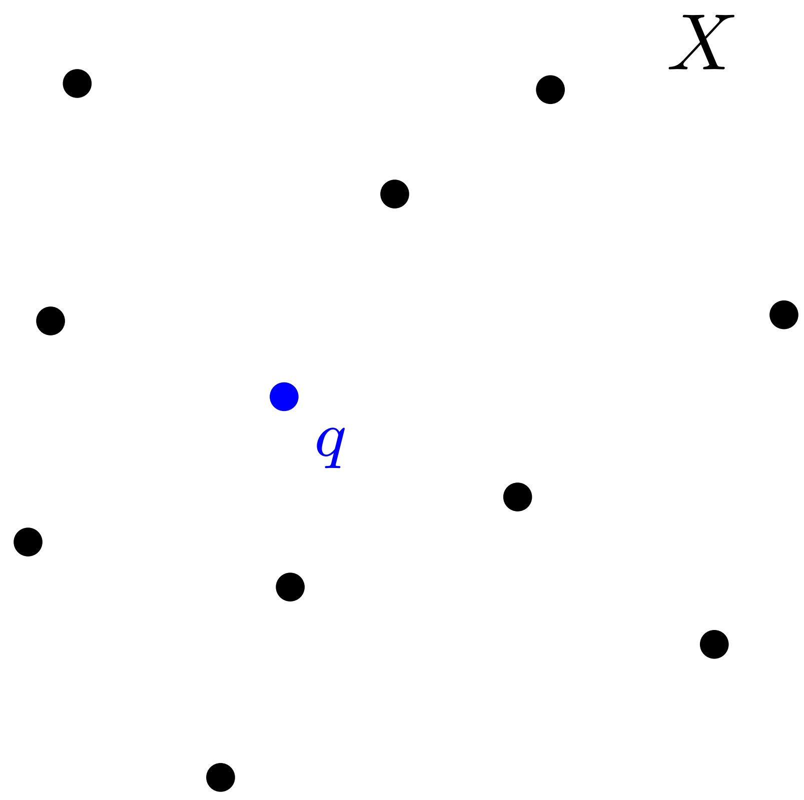
Widely studied problems (huge litterature)!
$\Rightarrow$ Many theoretical and practical results. Here we just quickly give a few hints on the subject and on existing methods.

The $(k-)$NN problem

Input: A point cloud X



The $(k-)$NN problem



Input: A point cloud X

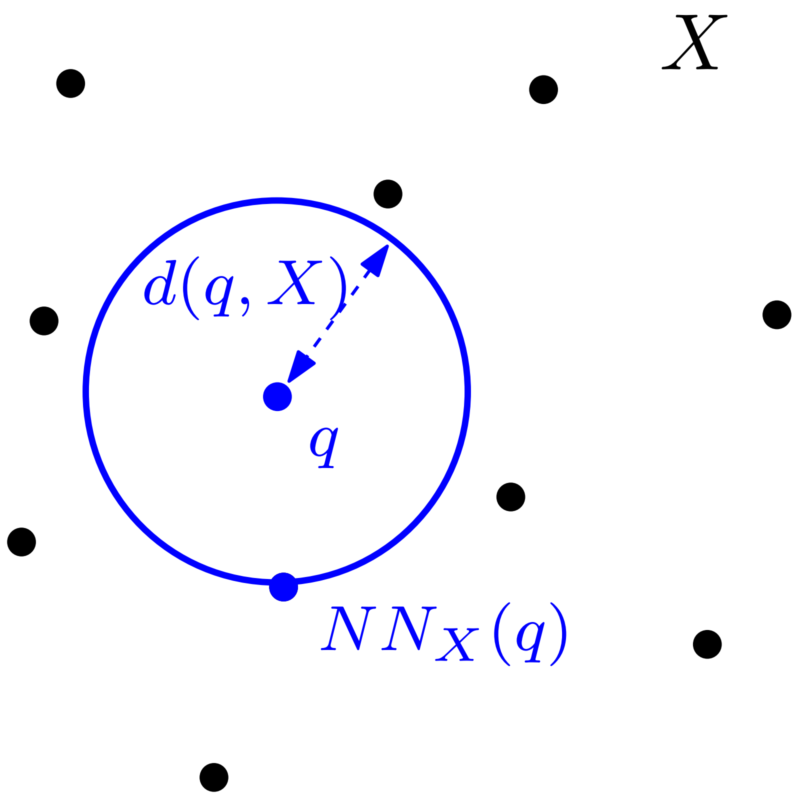
Query input: A point q

The $(k-)$NN problem

Input: A point cloud X

Query input: A point q

Goal: Find the nearest neighbor of q in X.

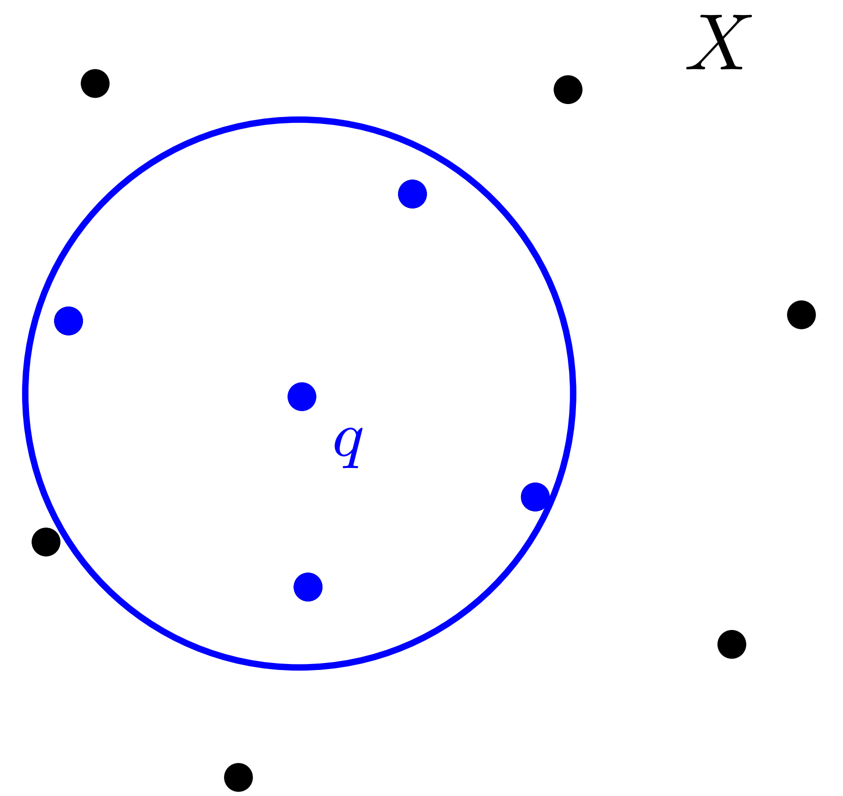


The $(k-)$NN problem

Input: A point cloud X

Query input: A point q

Goal: Find the nearest neighbor of q in X.

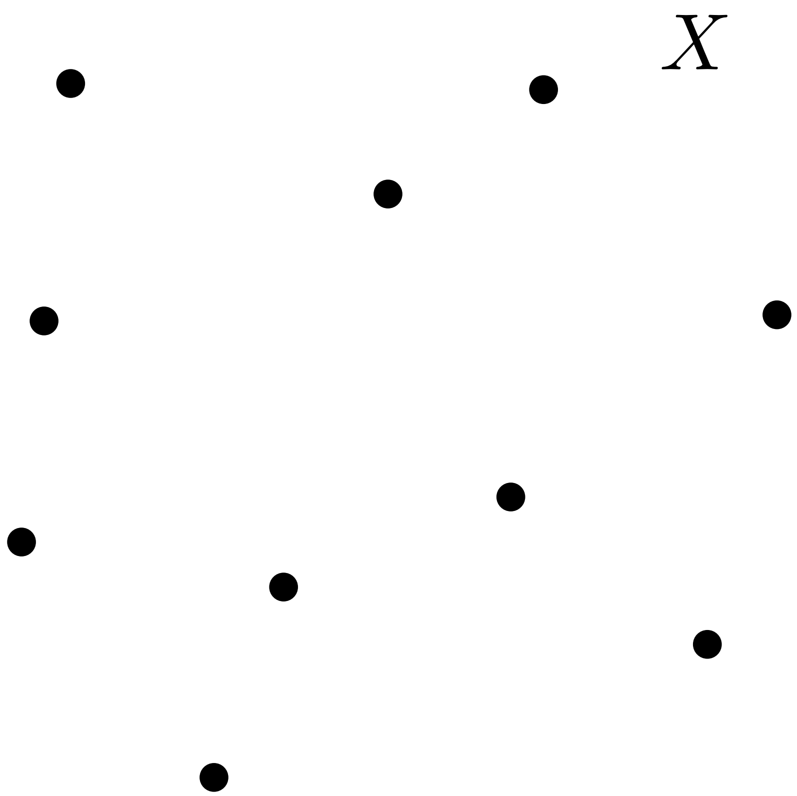


Variants:

- given an integer k, find the first k nearest neighbors of q in X.
- given $r > 0$, find the points of X at distance at most r from q.

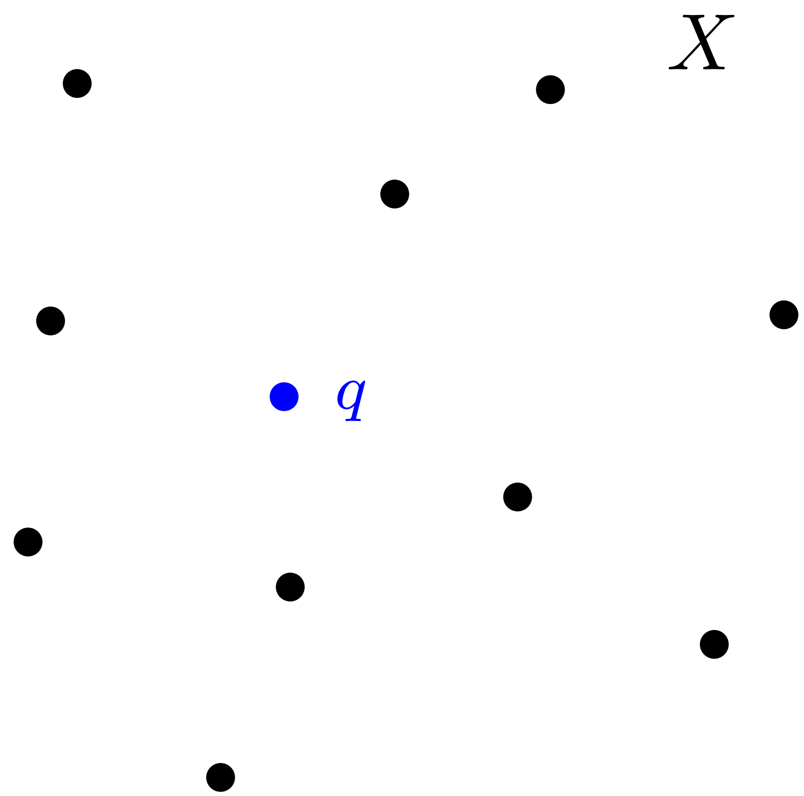
The ANN problem

Input: A point cloud X, a positive real $\varepsilon > 0$



The ANN problem

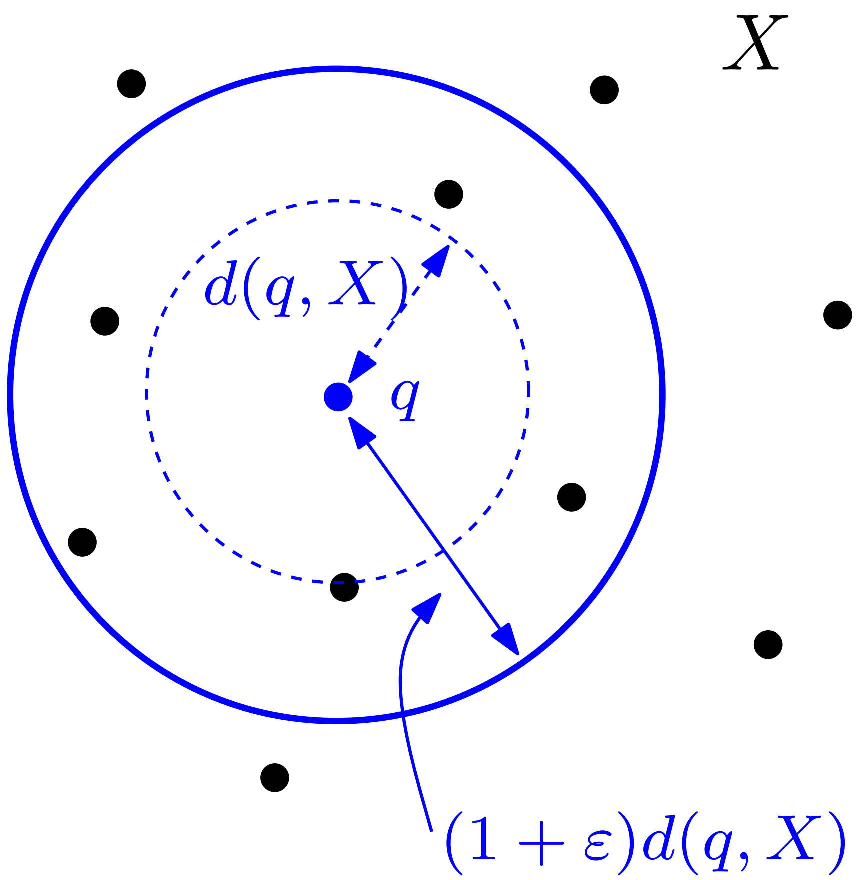
Input: A point cloud X, a positive real $\varepsilon > 0$
Query input: A point q



The ANN problem

Input: A point cloud X, a positive real $\varepsilon > 0$
Query input: A point q

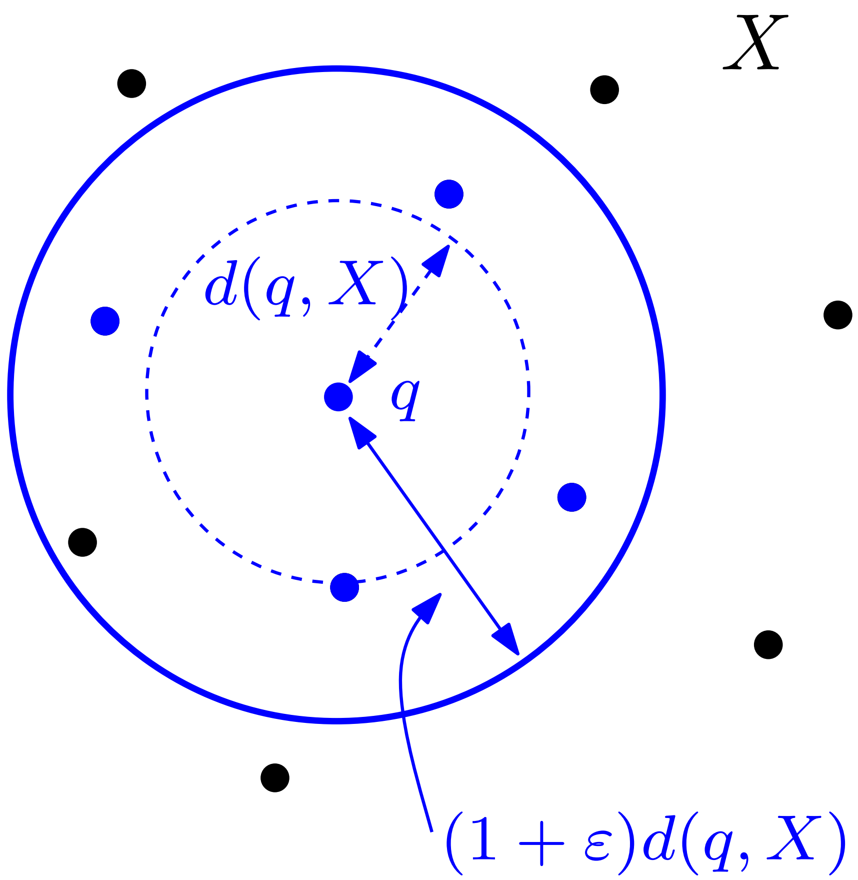
Goal: Find a point $x \in X$ s.t. $d(q,x) \leq (1+\varepsilon)d(q,X)$.



The ANN problem

Input: A point cloud X, a positive real $\varepsilon > 0$
Query input: A point q

Goal: Find a point $x \in X$ s.t. $d(q,x) \leq (1+\varepsilon)d(q,X)$.



Same variant with k approximate nearest neighbors: find $x_1, \cdots x_k \in X$ s.t. for any $i = 1, \cdots k$, $d(q,x_i) \leq d(q, x^k(q))$ where $x^k(q)$ is the k^{th} nearest neighbor of q.

The (A)NN problem

Many approaches and variants:

- the naive algorithm (brute force): linear size, linear query time
- kd-trees: linear size, $O(\log n)$ query time (under some restrictive conditions
- fat cells)
- BBD trees: linear size, $O((\frac{d}{\varepsilon})^d \log n)$ query time
- Voronoï diagrams: $n^{O(d)}$ size, $O(\log n)$ query time
- etc....

The (A)NN problem

Many approaches and variants:

- the naive algorithm (brute force): linear size, linear query time
- kd-trees: linear size, $O(\log n)$ query time (under some restrictive conditions
- fat cells)
- BBD trees: linear size, $O((\frac{d}{\varepsilon})^d \log n)$ query time
- Voronoï diagrams: $n^{O(d)}$ size, $O(\log n)$ query time
- etc....

The curse of dimensionality: every data structure for NN-search in sub-linear time has either exponential size or exponential query time (in d!)

The (A)NN problem

Many approaches and variants:

- the naive algorithm (brute force): linear size, linear query time
- kd-trees: linear size, $O(\log n)$ query time (under some restrictive conditions
- fat cells)
- BBD trees: linear size, $O((\frac{d}{\varepsilon})^d \log n)$ query time
- Voronoï diagrams: $n^{O(d)}$ size, $O(\log n)$ query time
- etc....

The curse of dimensionality: every data structure for NN-search in sub-linear time has either exponential size or exponential query time (in d!)

Non longer true if one consideres the ANN problem!

The (A)NN problem

Many approaches and variants:

- the naive algorithm (brute force): linear size, linear query time
- kd-trees: linear size, $O(\log n)$ query time (under some restrictive conditions
- fat cells)
- BBD trees: linear size, $O((\frac{d}{\varepsilon})^d \log n)$ query time
- Voronoï diagrams: $n^{O(d)}$ size, $O(\log n)$ query time
- etc....

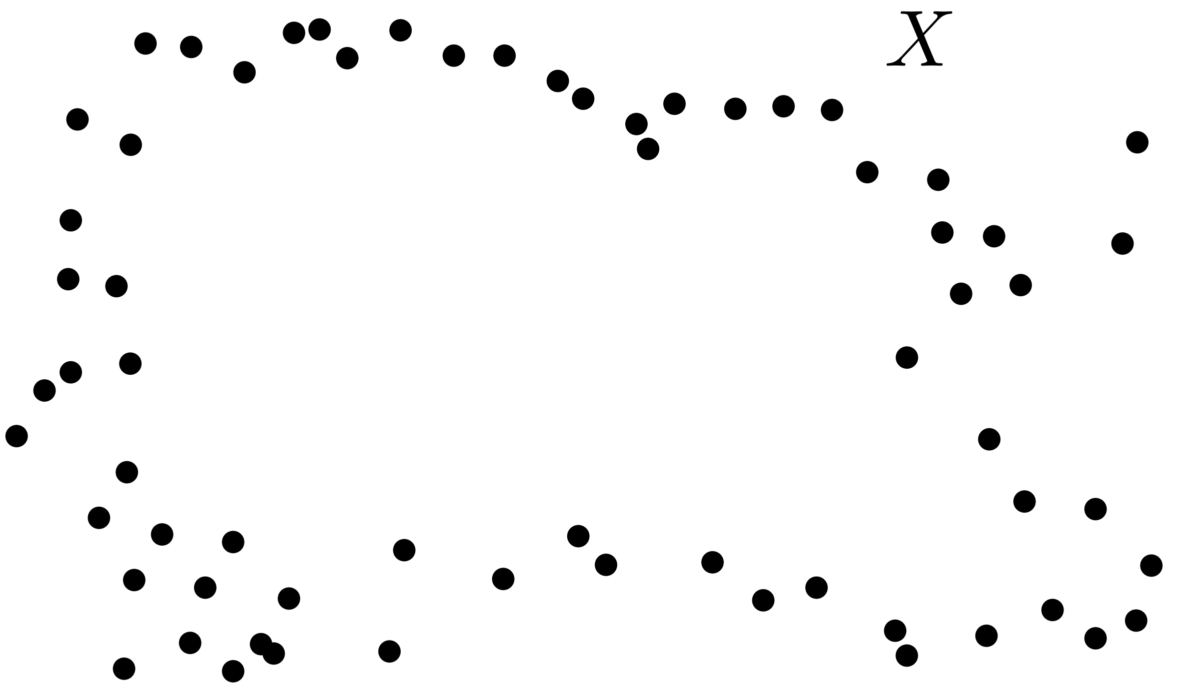
The curse of dimensionality: every data structure for NN-search in sub-linear time has either exponential size or exponential query time (in d!)

Non longer true if one consideres the ANN problem!

In the following of this course ANN or NN search will be considered as a "black box". In practice, you can use for example the ANN library developed by D. Mount : http://www.cs.umd.edu/mount/ANN

Landmark selection

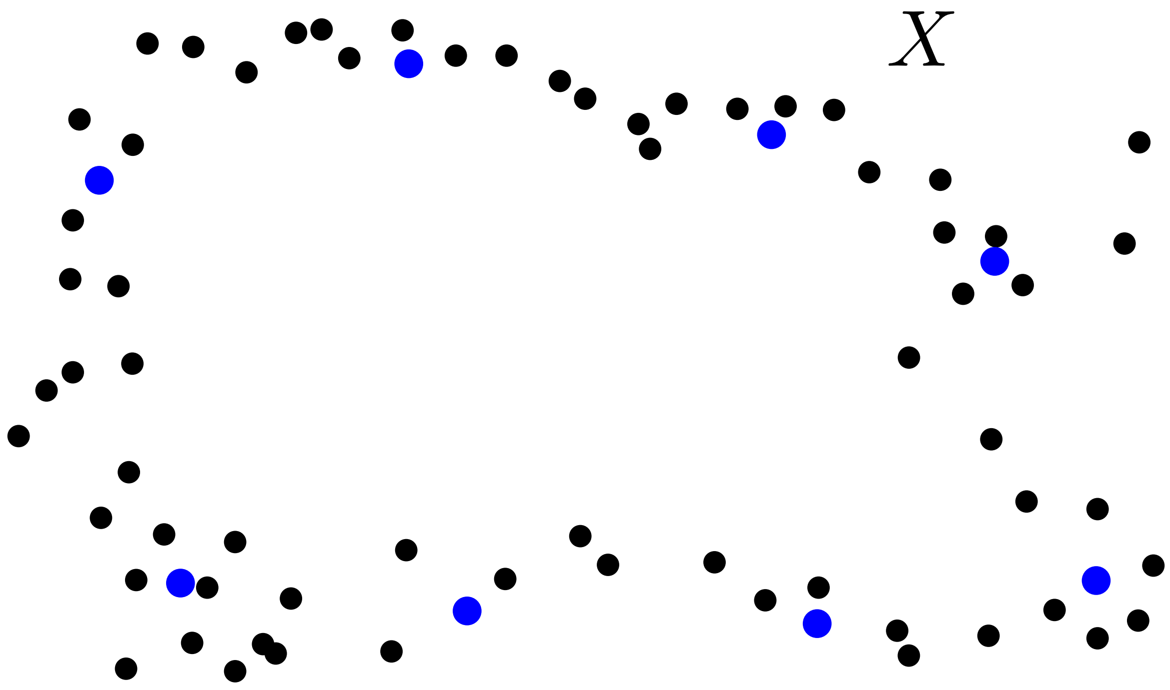
Input: A (large) point cloud X



Landmark selection

Input: A (large) point cloud X

Goal: given an integer $k < N$, select a set L of k points that samples X "as best as possible".

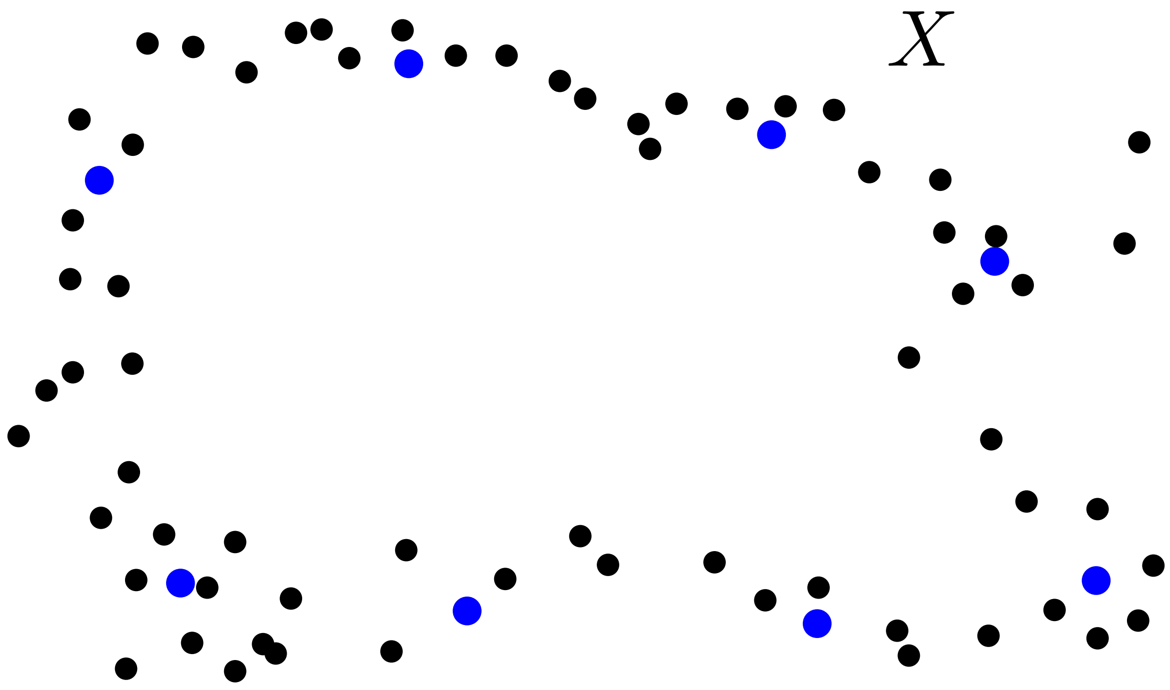


Landmark selection

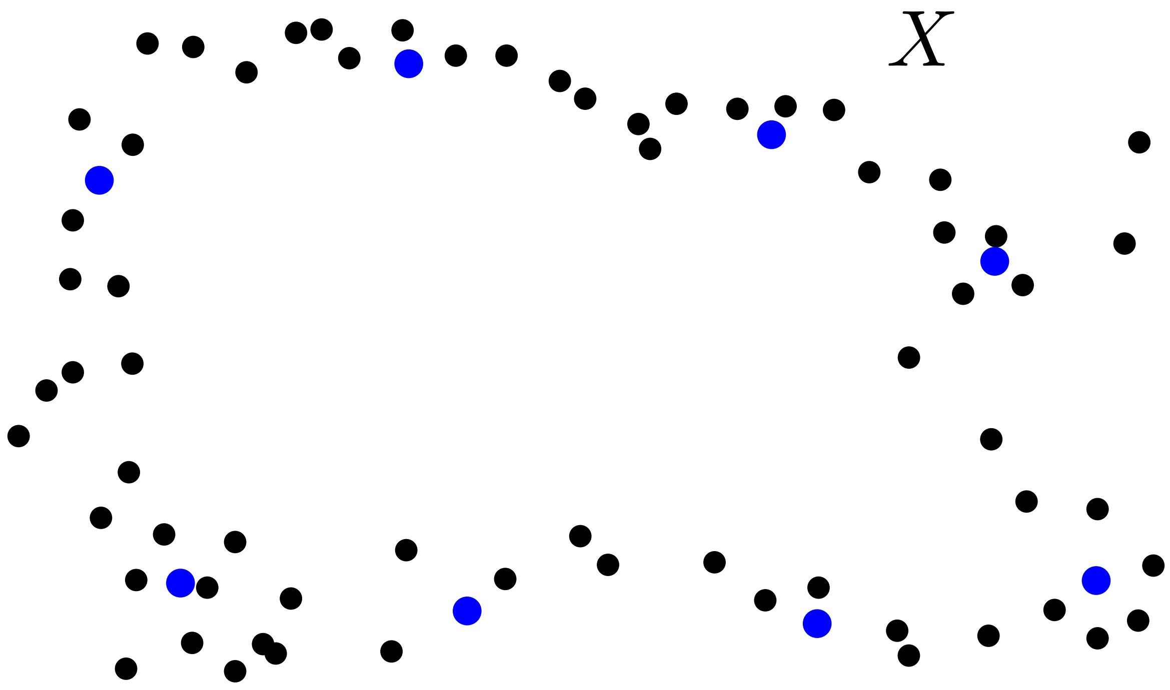
Input: A (large) point cloud X

Goal: given an integer $k < N$, select a set L of k points that samples X "as best as possible".

Not canonically defined!



Landmark selection



Input: A (large) point cloud X

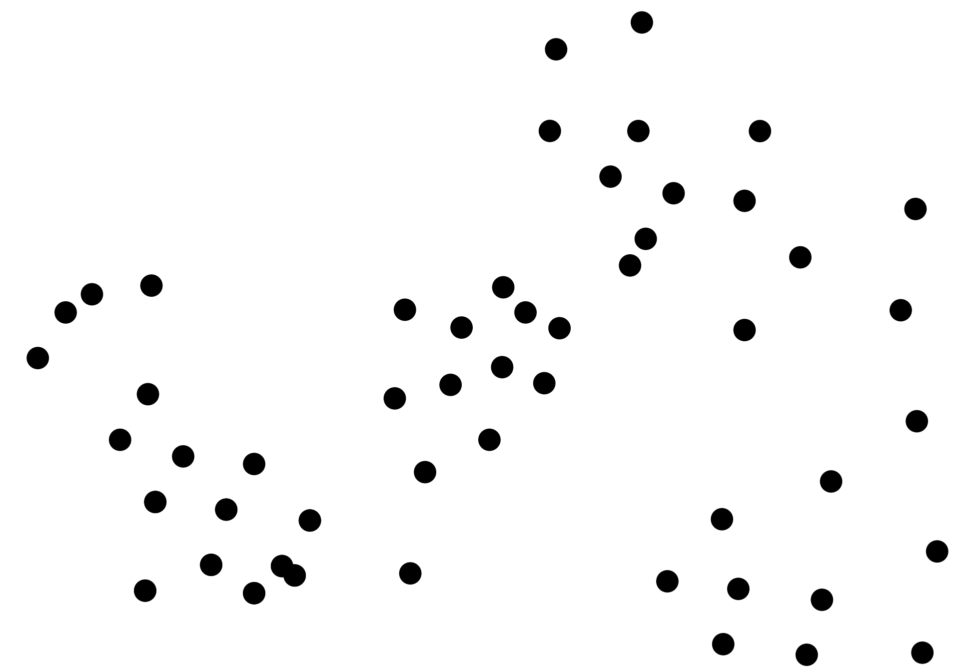
Goal: given an integer $k < N$, select a set L of k points that samples X "as best as possible".

Not canonically defined!

Many methods:

- random sampling in X
- k-means algorithm and variants
- furthest point sampling
- etc ...

the k-means algorithm

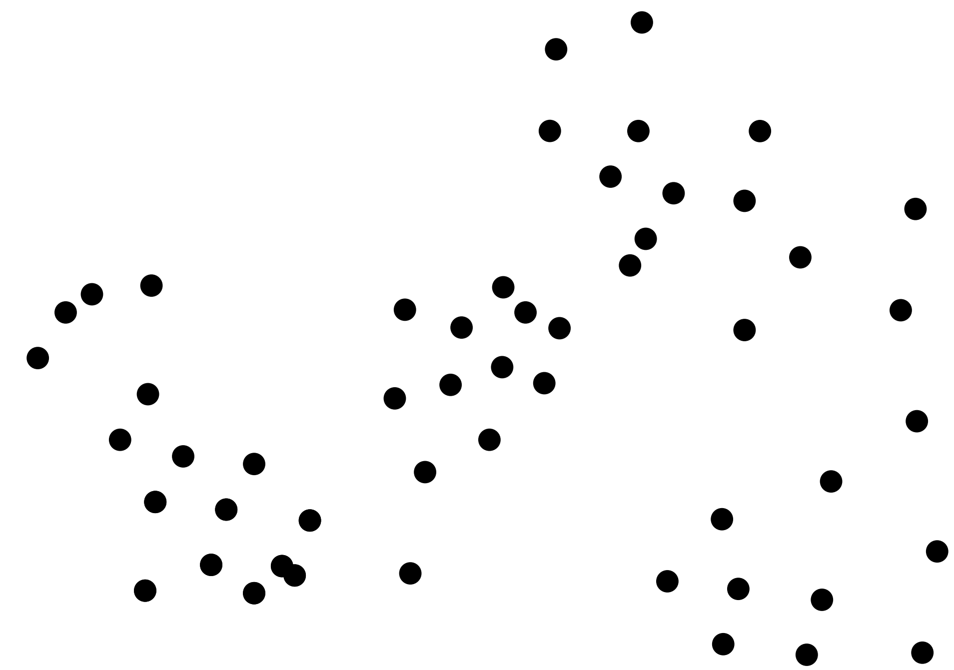


Input: A (large) set of N points X and an integer $k < N$.

Goal: Find a set of k points $L = \{y_1, \cdots y_k\}$ that minimizes

$$E = \sum_{i=1}^{N} d(x_i, L)^2$$

the k-means algorithm



Input: A (large) set of N points X and an integer $k < N$.

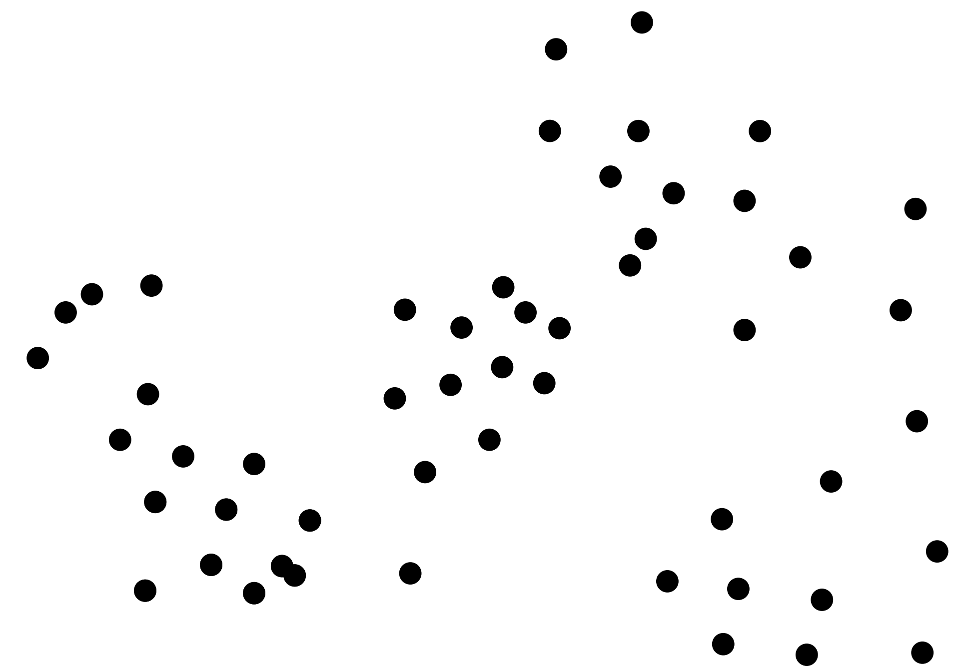
Goal: Find a set of k points $L = \{y_1, \cdots y_k\}$ that minimizes

$$E = \sum_{i=1}^{N} d(x_i, L)^2$$

 This is a NP-hard problem!

The Lloyd's algorithm: a very simple local search algorithm $\rightarrow$ local minimum.

the k-means algorithm



The Lloyd's algorithm

- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat

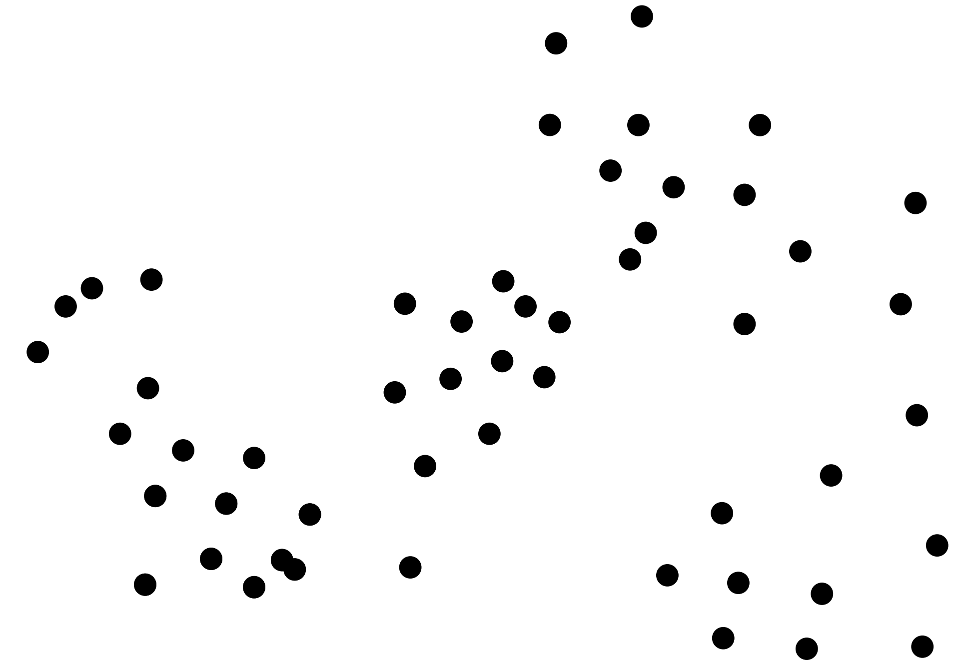
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;

- Until convergence

the k-means algorithm



The Lloyd's algorithm

- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat

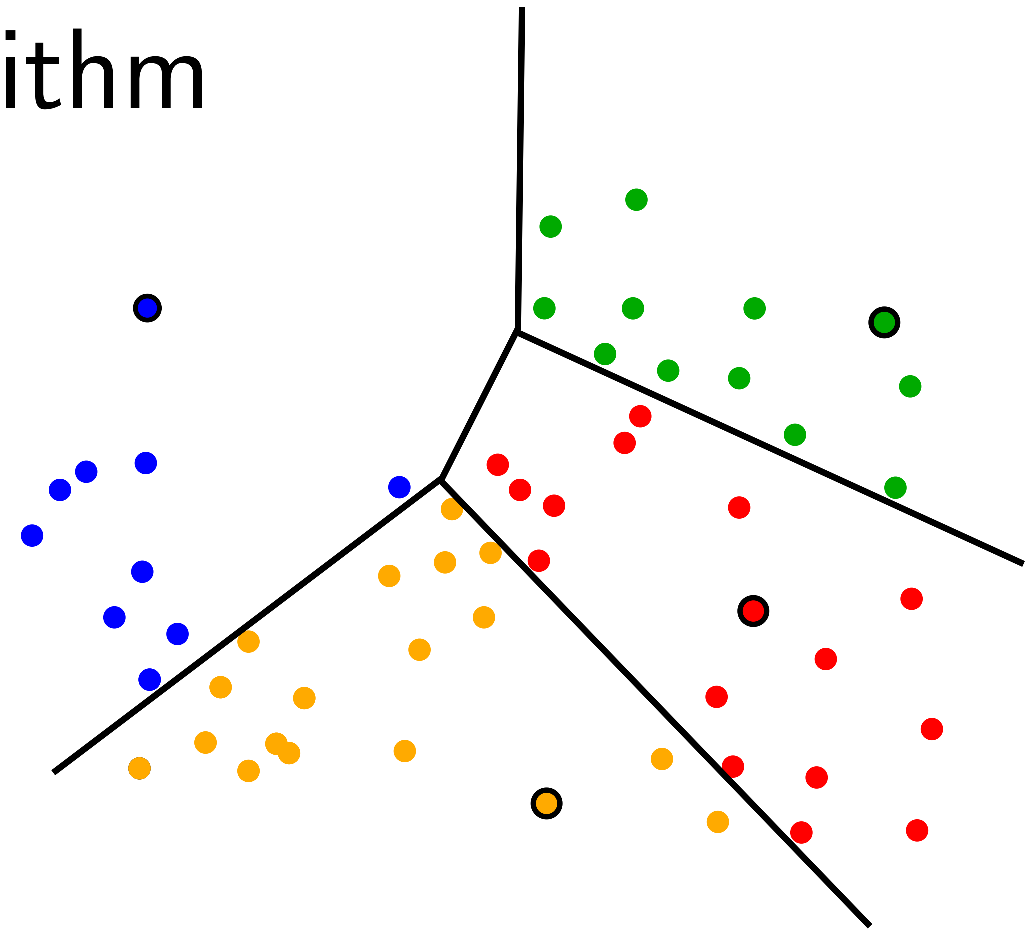
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;

- Until convergence

the k-means algorithm



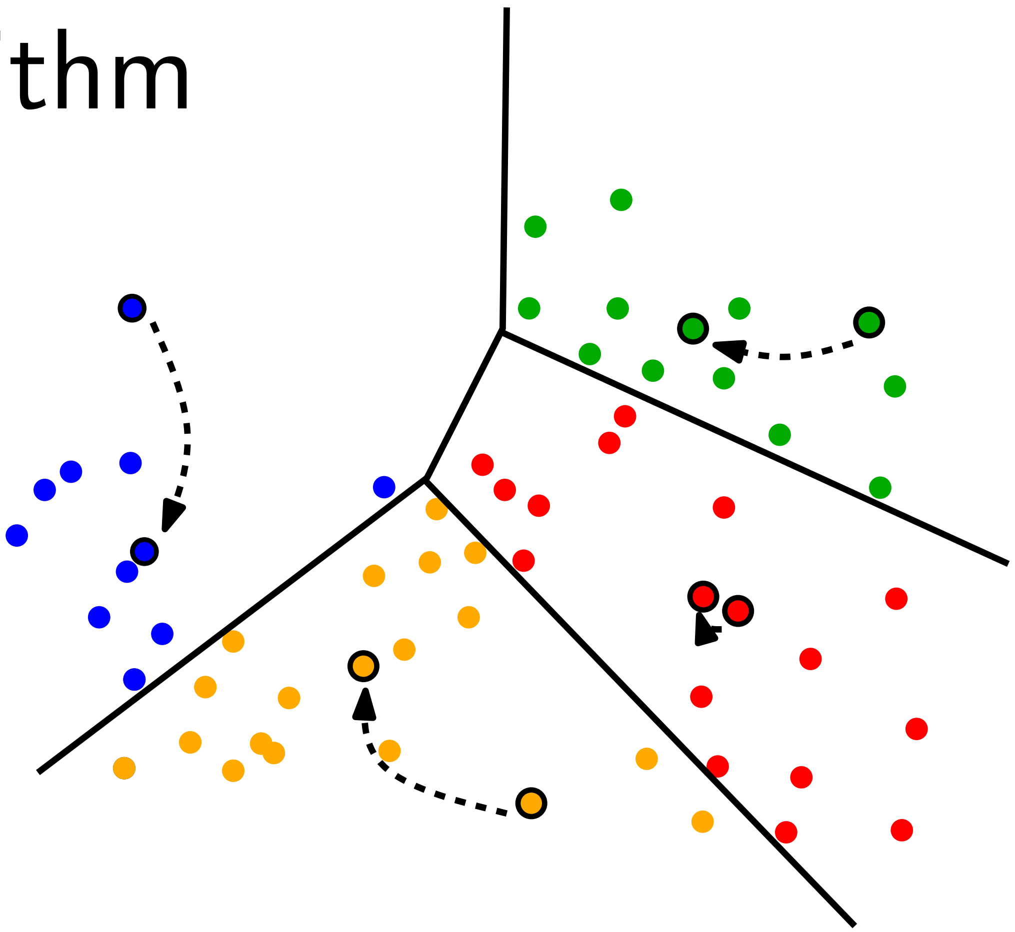
The Lloyd's algorithm

- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

the k-means algorithm



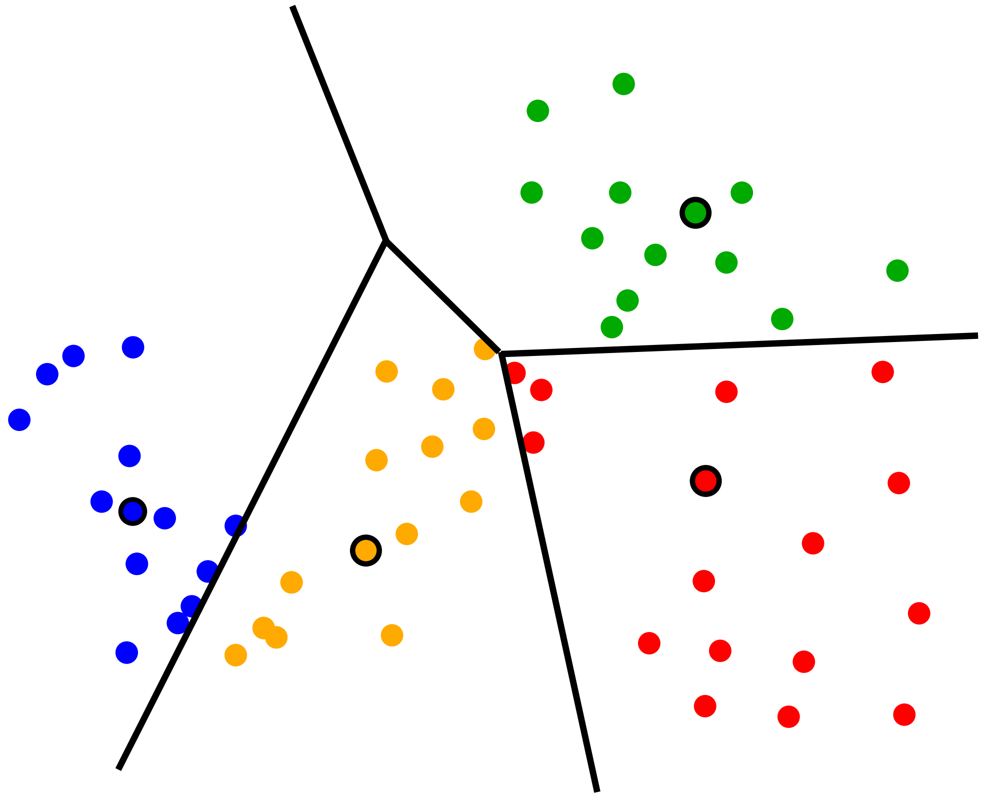
The Lloyd's algorithm

- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

the k-means algorithm



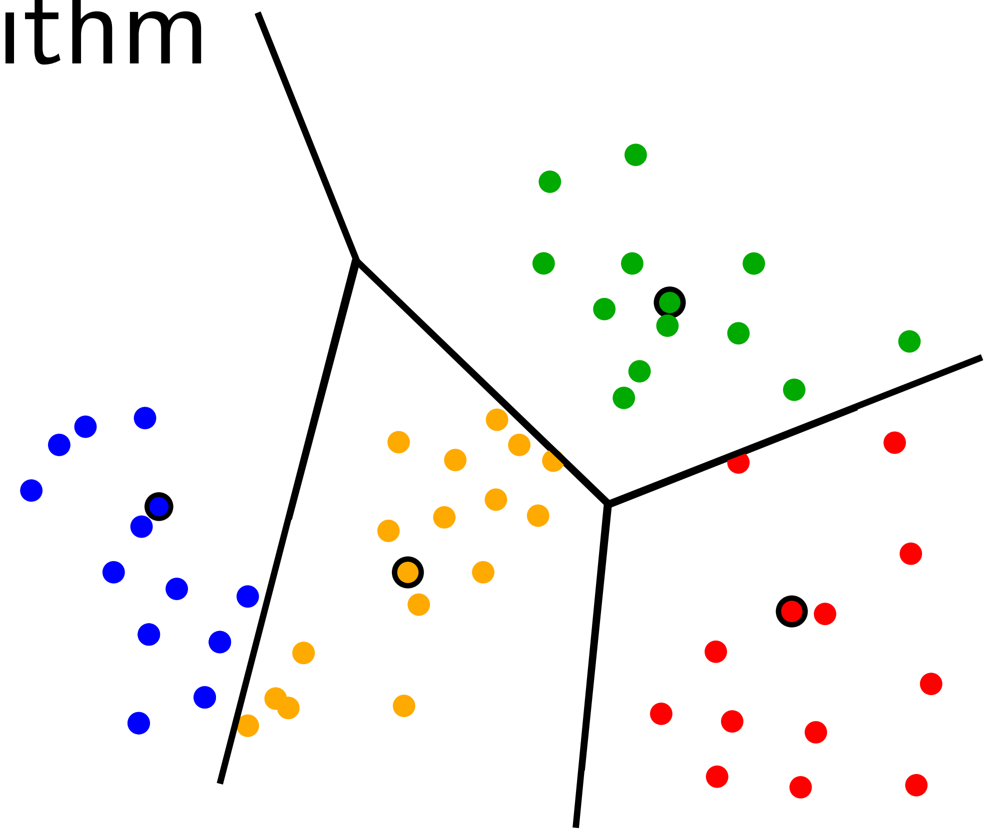
The Lloyd's algorithm

- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

the k-means algorithm



The Lloyd's algorithm

- Select $L^1 = \{y_1^1, \cdots y_k^1\}$ initial "seeds";
- $i = 1$;
- Repeat
  - For $(j = 1; j \leq k; j++)$ $S_j^i = \{x \in X : d(x, y_j^i) = d(x, L^i)\}$;
  - For $(j = 1; j \leq k; j++)$

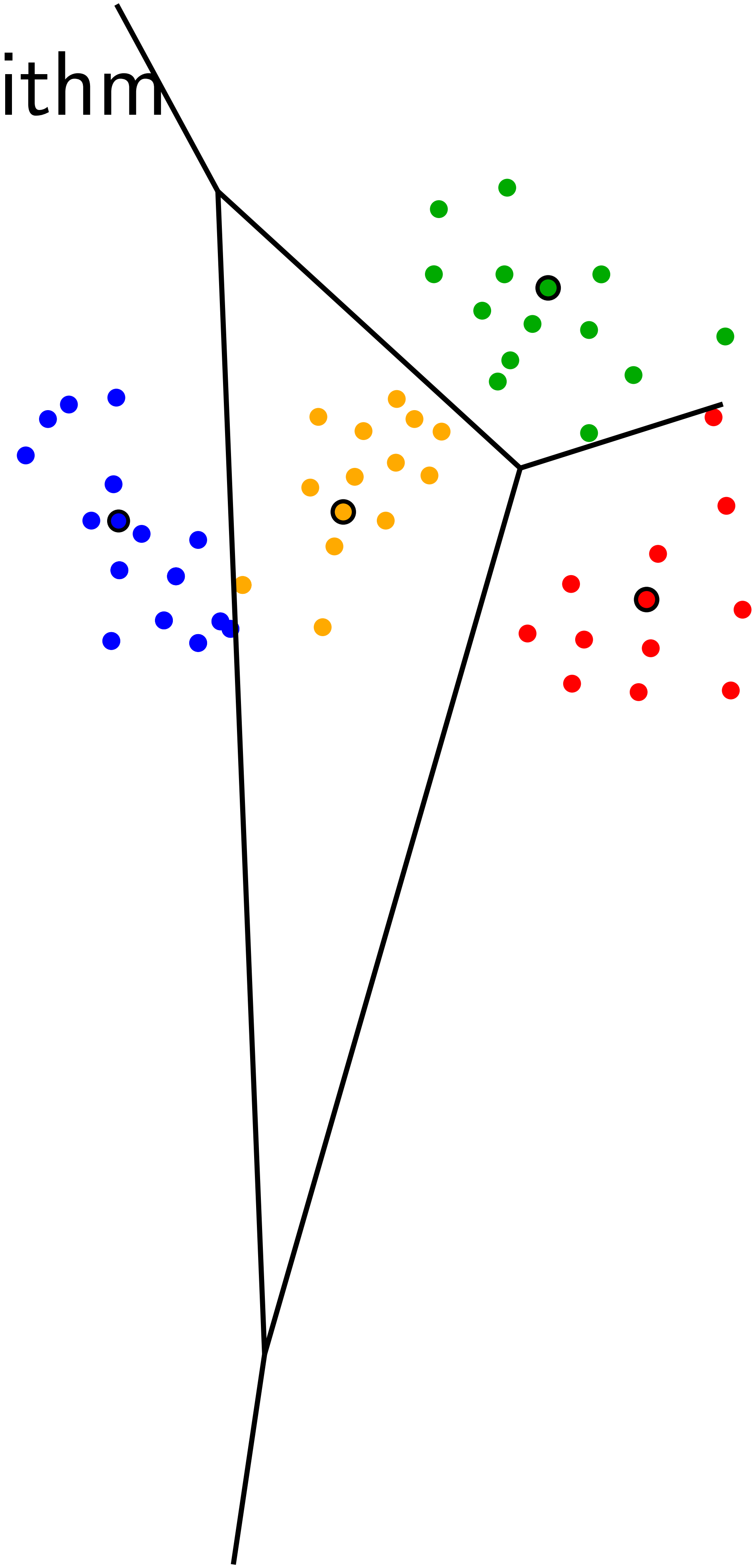
$$y_j^{i+1} = \frac{1}{|S_j^i|} \sum_{x \in S_j^i} x$$

  - $i++$;
- Until convergence

the k-means algorithm

Warnings:

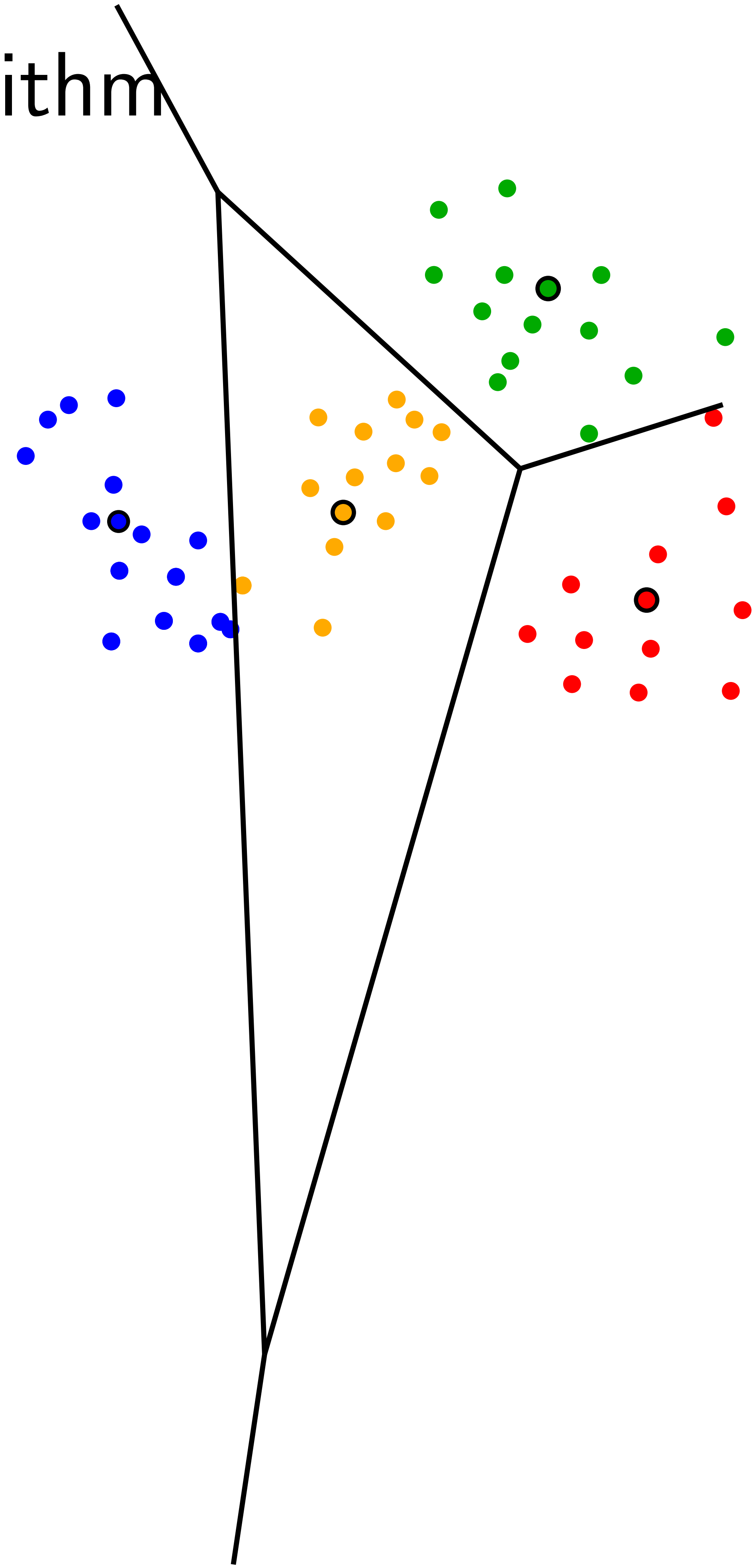
- Lloyd's algorithm does not ensure convergence to a global minimum!
- The speed of convergence is not guaranteed.
- the choice of the initial seeds may be critical (but there exists some strategies $\rightarrow$ k-means++).



the k-means algorithm

Warnings:

- Lloyd's algorithm does not ensure convergence to a global minimum!
- The speed of convergence is not guaranteed.
- the choice of the initial seeds may be critical (but there exists some strategies $\rightarrow$ k-means$++$).

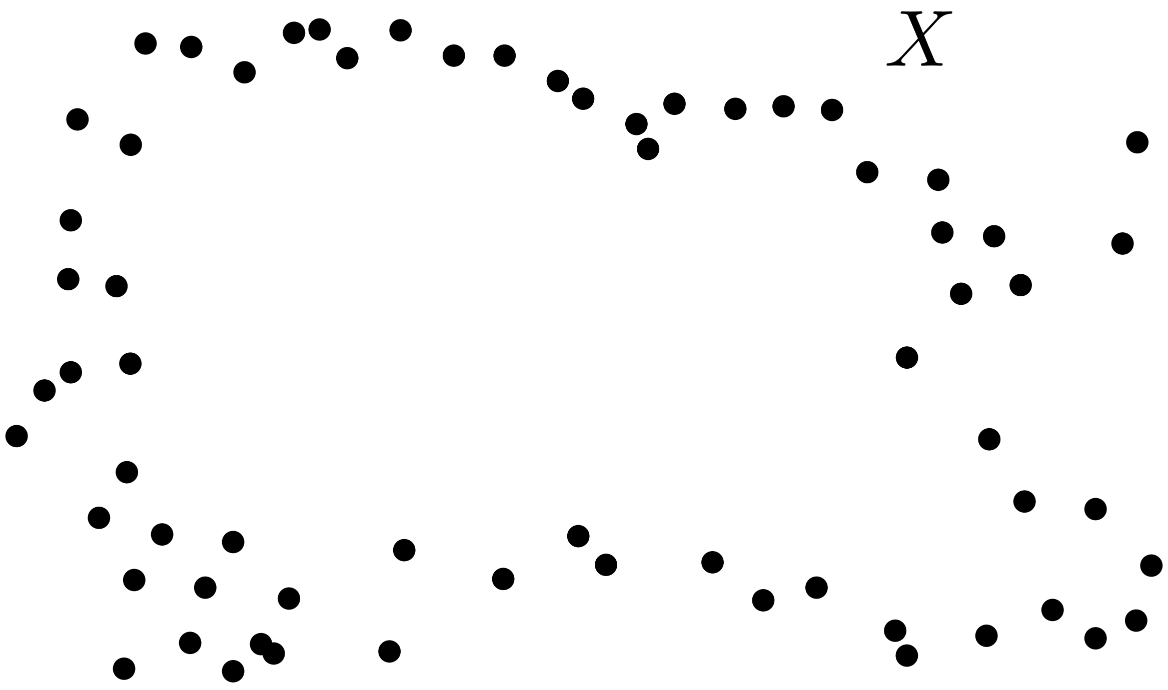


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $= \{$ y$_1\}$;
- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \mathrm{argmax}_{x \in X} d(x, L)$;
  - $L = L \cup \{y_i\}$

$\}$

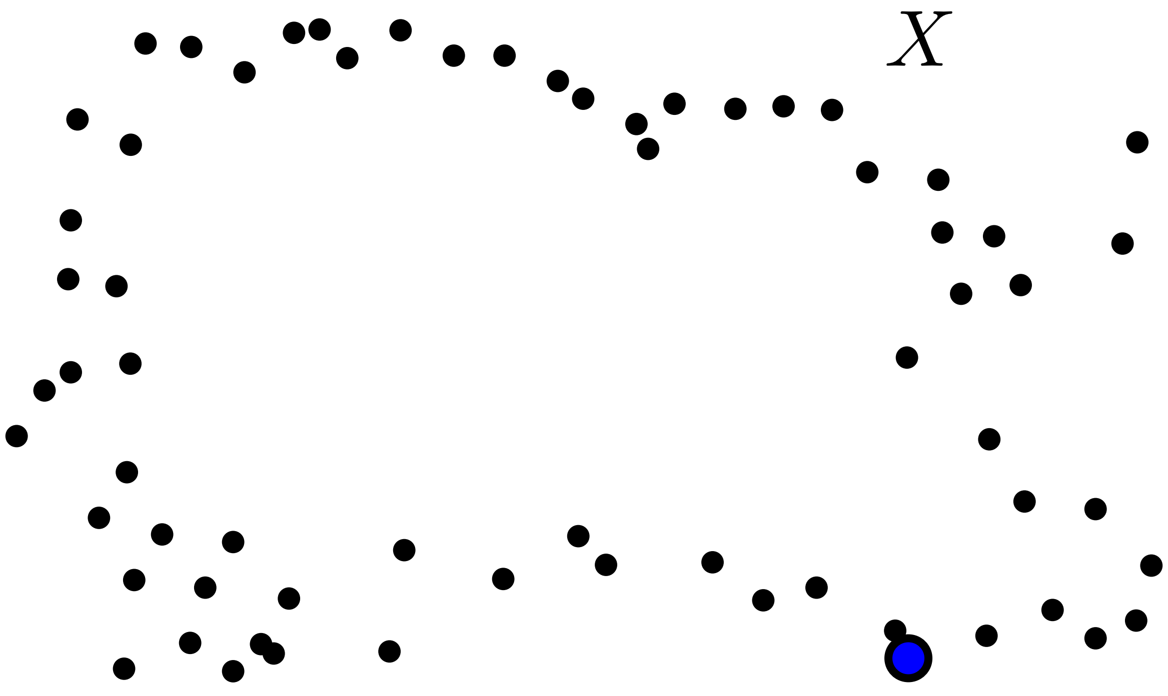


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $=$ { y$_1$};
- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \mathrm{argmax}_{x \in X} d(x, L)$;
  - $L = L \cup \{y_i\}$

}

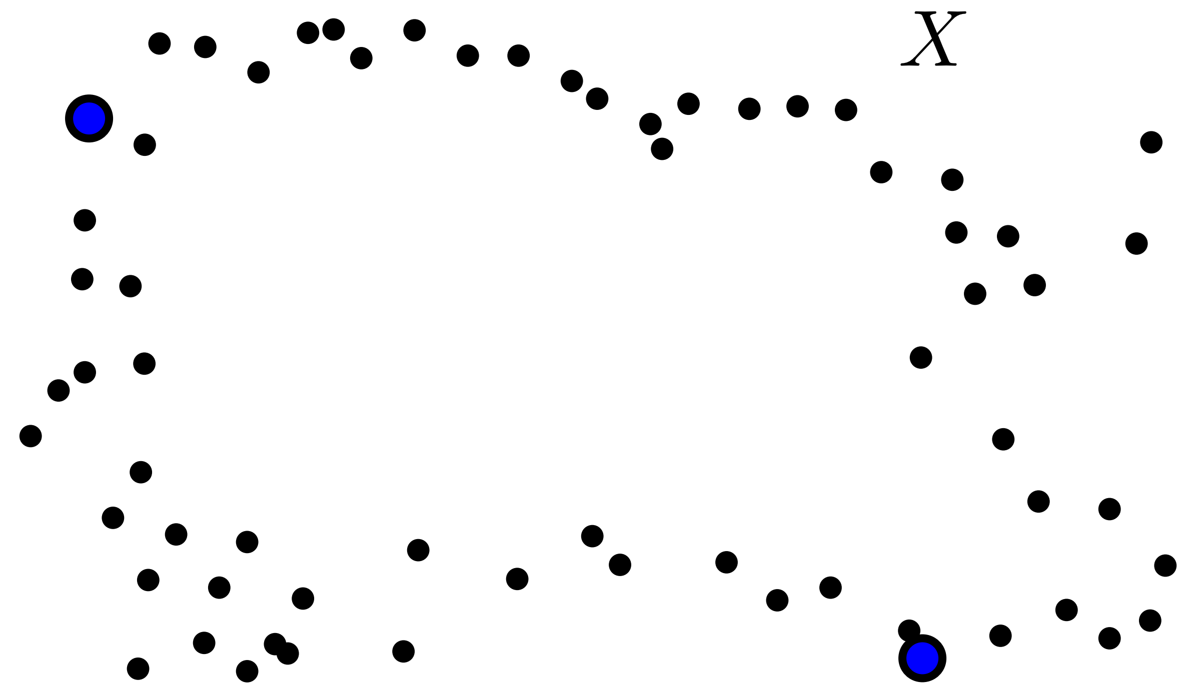


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $=$ { y_1};
- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \mathrm{argmax}_{x \in X} d(x, L)$;
  - $L = L \cup \{y_i\}$

}

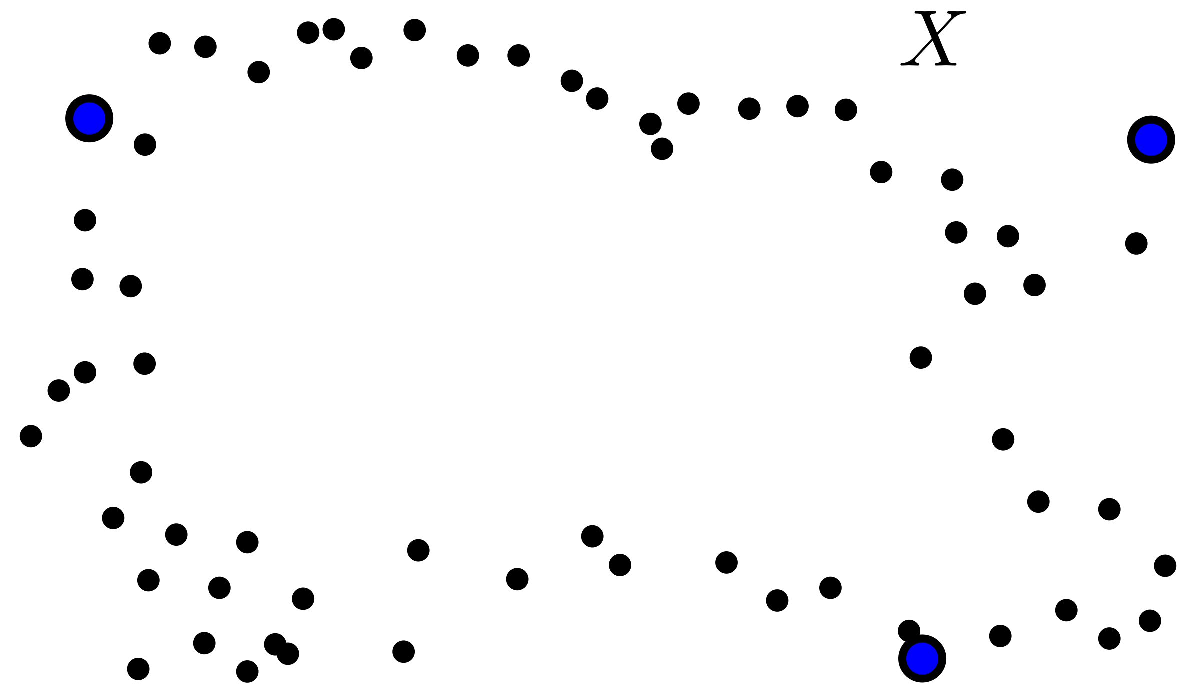


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $= \{$ y$_1\}$;
- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \text{argmax}_{x \in X} d(x, L)$;
  - $L = L \cup \{y_i\}$

$\}$



The furthest point strategy

- select randomly $y_1 \in$X;

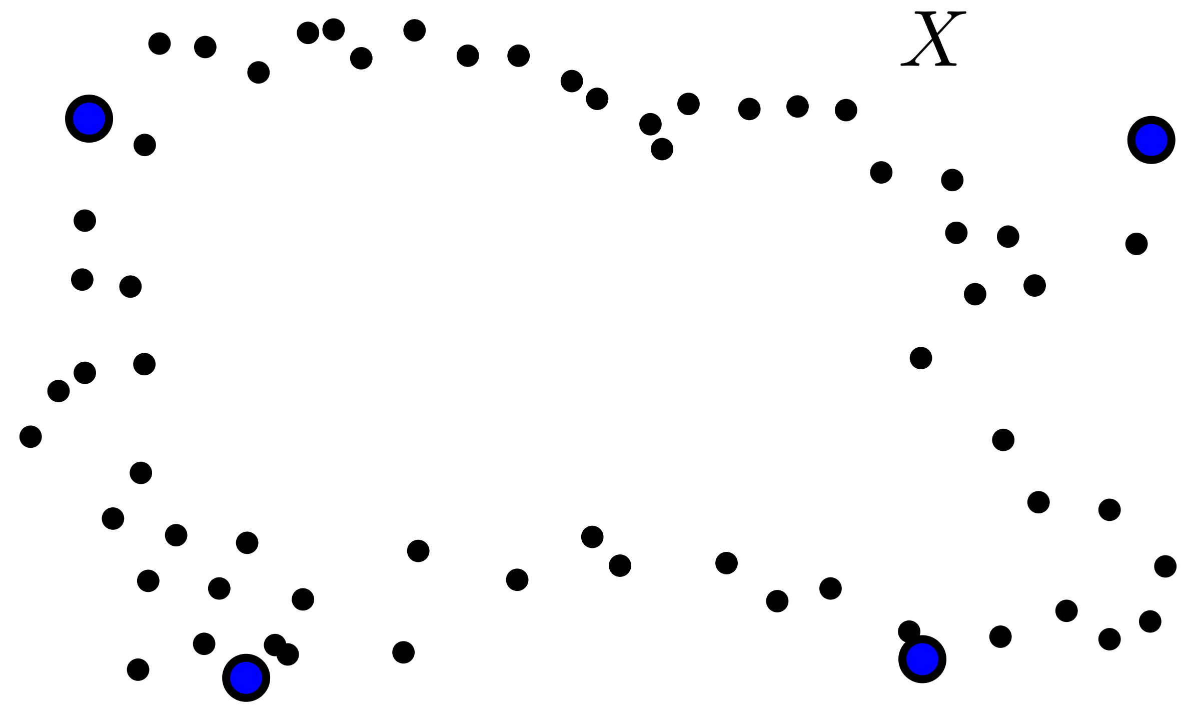
$-$L $=$ { y$_1$};

- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \mathrm{argmax}_{x \in X} d(x, L)$;

  - $L = L \cup \{y_i\}$

}

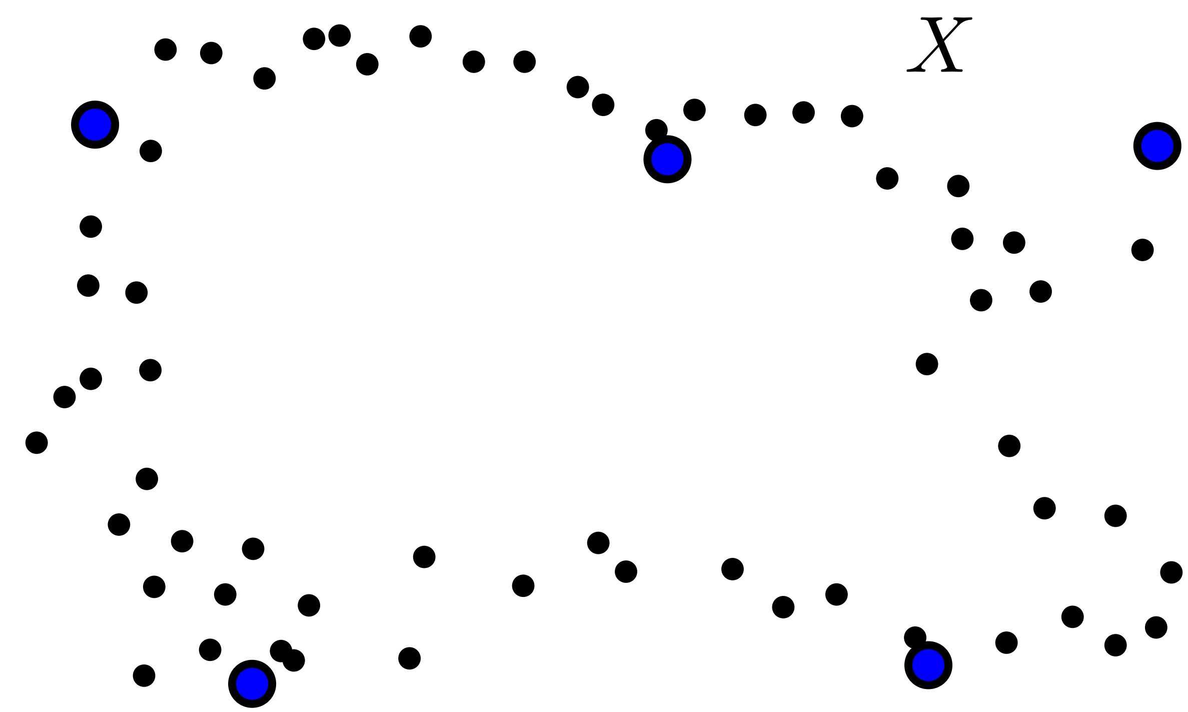


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $= \{$ y$_1\}$;
- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \mathrm{argmax}_{x \in X} d(x, L)$;
  - $L = L \cup \{y_i\}$

$\}$

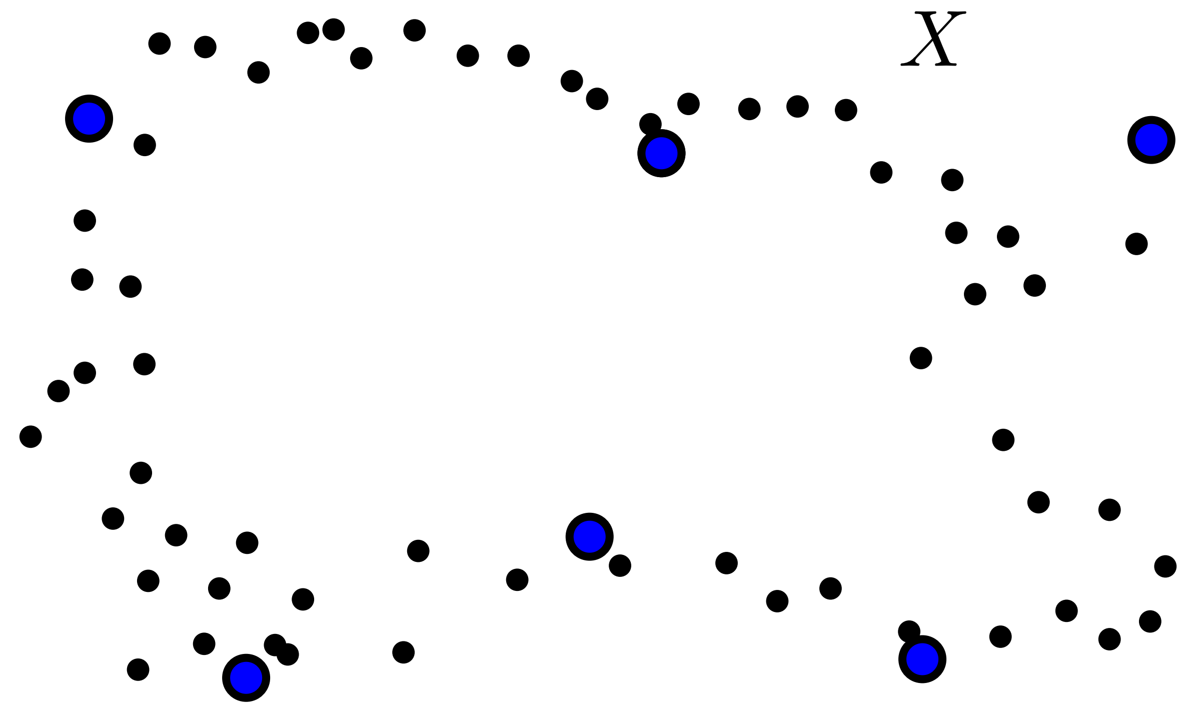


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $=$ { y$_1$};
- for $(i = 2; i \leq k; i++)\{$

- $y_i = \text{argmax}_{x \in X} d(x, L)$;
- $L = L \cup \{y_i\}$

}

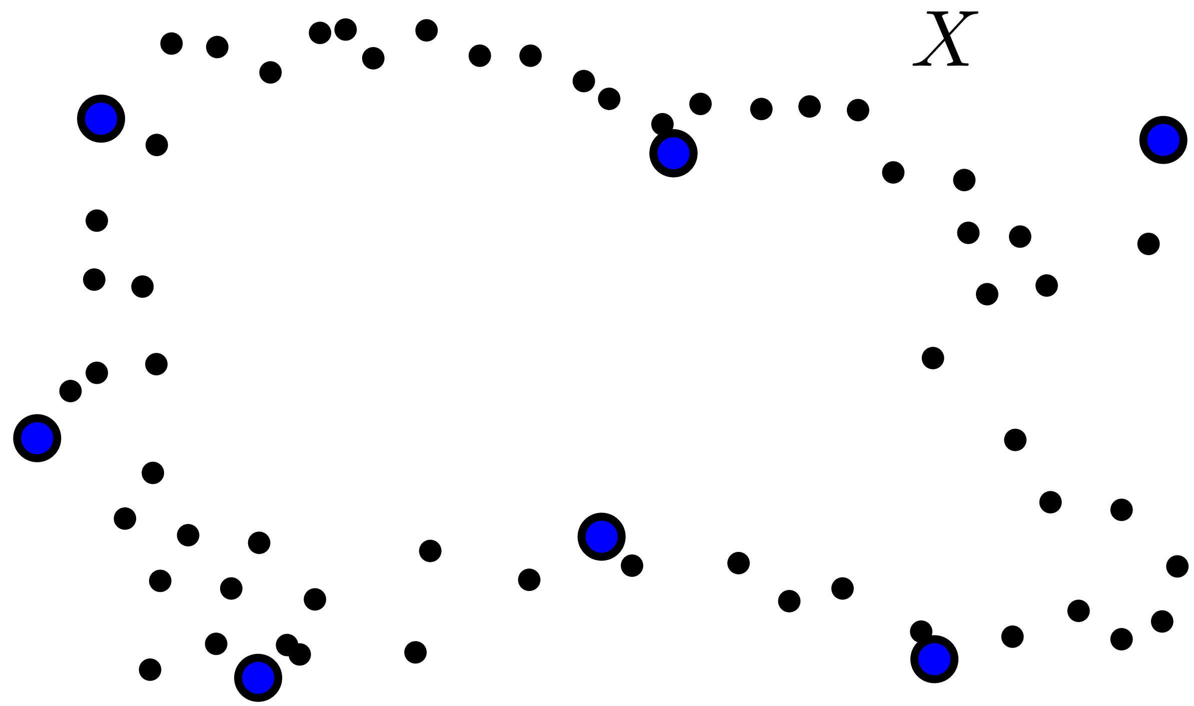


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $=$ { y$_1$};
- for $(i = 2; i \leq k; i++)\{$

- $y_i = \operatorname{argmax}_{x \in X} d(x, L)$;
- $L = L \cup \{y_i\}$

$\}$



The furthest point strategy

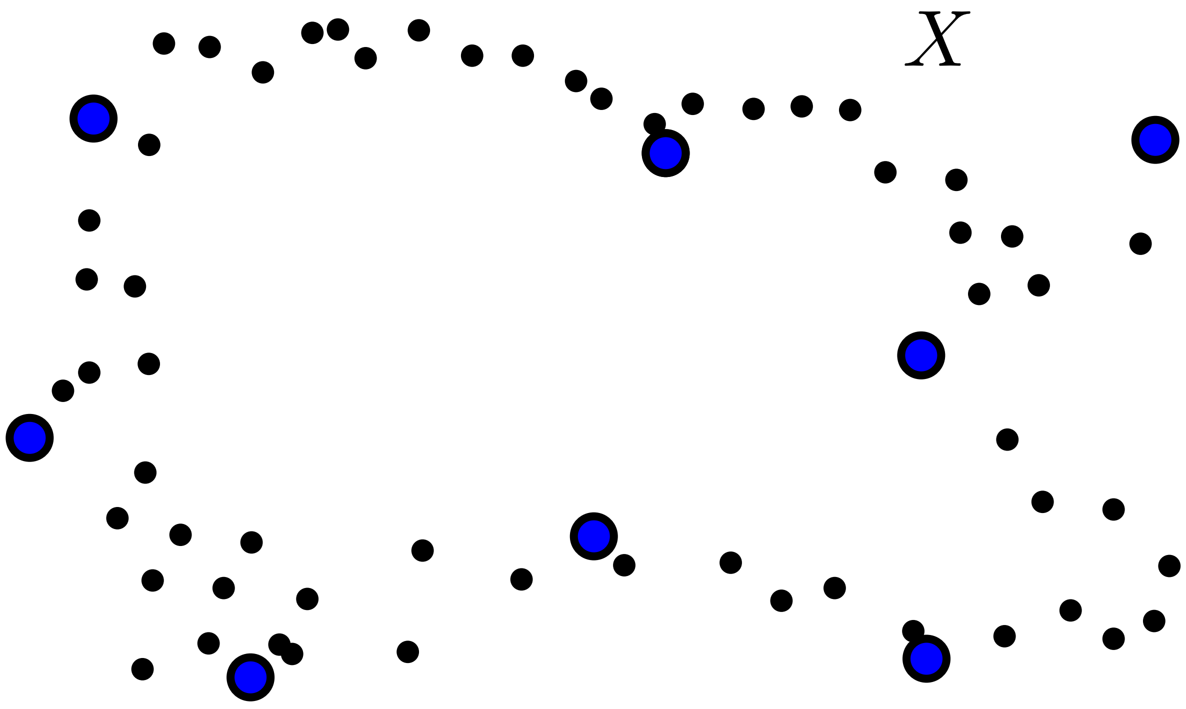
- select randomly $y_1 \in$X;

$-$L $= \{$ y$_1\}$;

- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \mathrm{argmax}_{x \in X} d(x, L)$;
  - $L = L \cup \{y_i\}$

$\}$

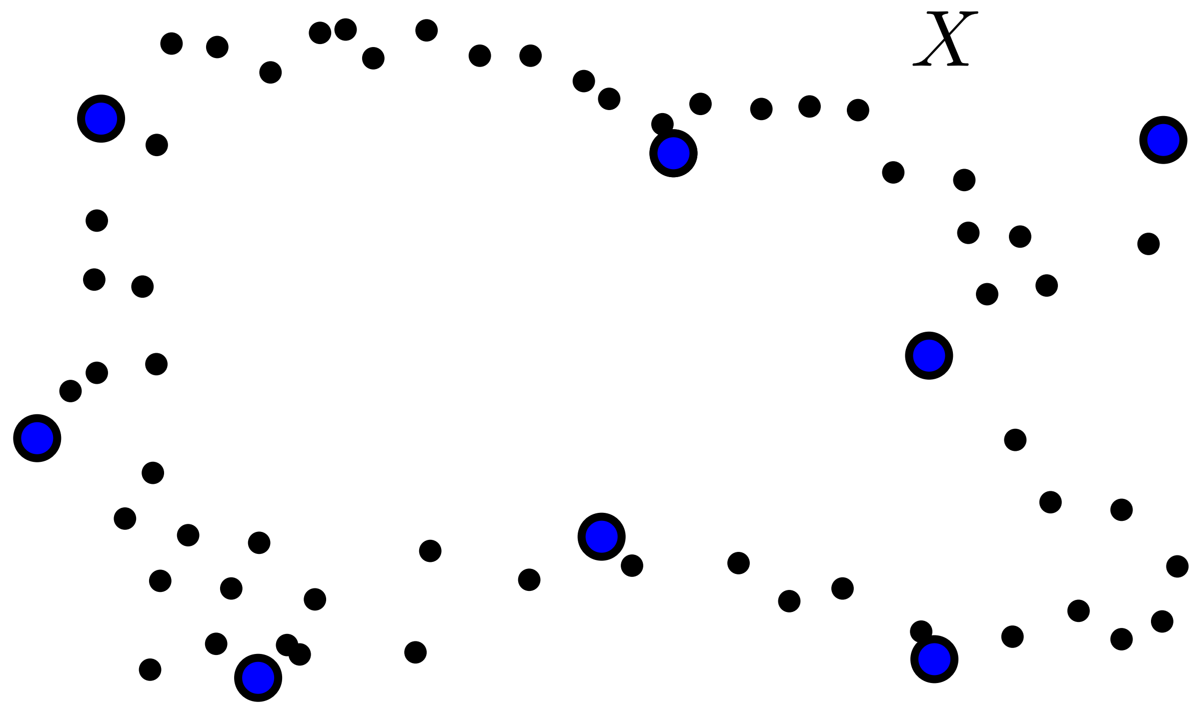


The furthest point strategy

- select randomly $y_1 \in$X;
$-$L $= \{$ y$_1\}$;
- for $(i = 2; i \leq k; i++)\{$

  - $y_i = \mathrm{argmax}_{x \in X} d(x, L)$;
  - $L = L \cup \{y_i\}$

$\}$

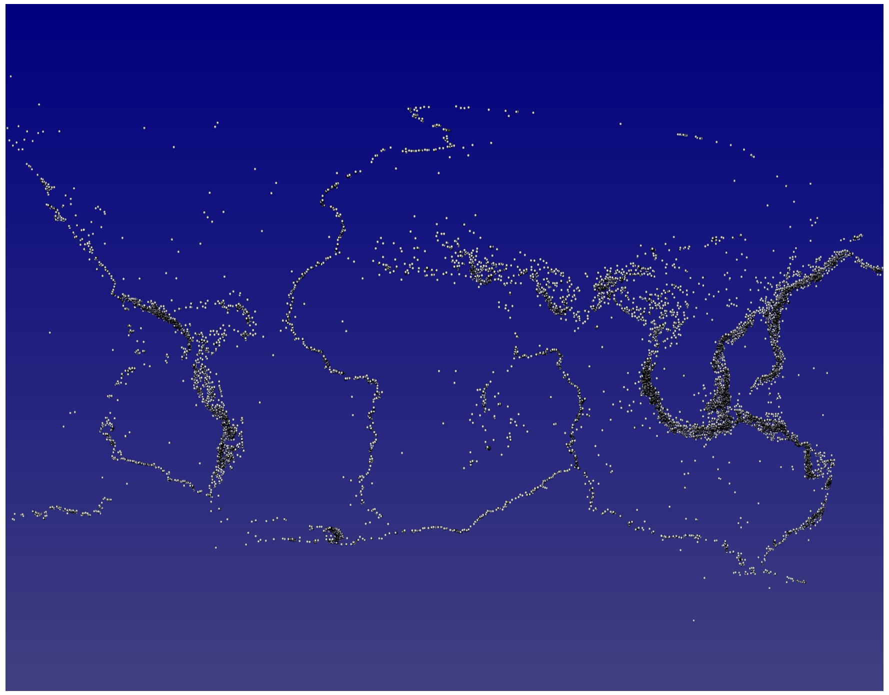


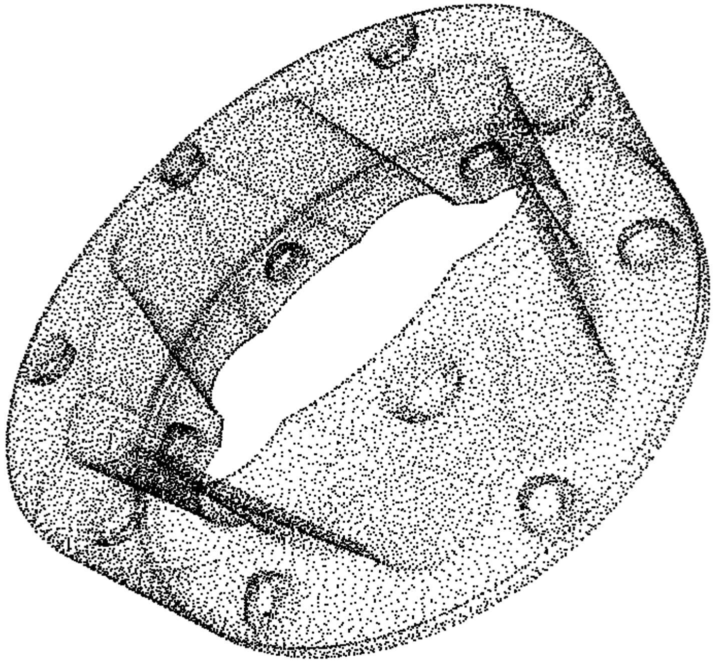
Warning:

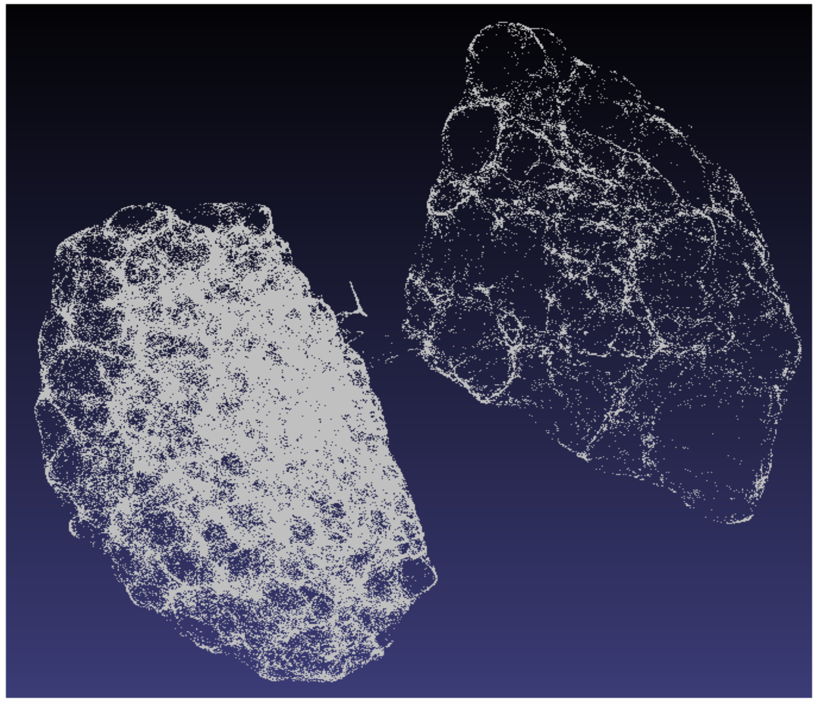
- Sensitive to outliers!

Perspectives: topological and geometric data analysis

Previous methods usually comes with theoretical guarantees when M is a smooth manifold with trivial topology.

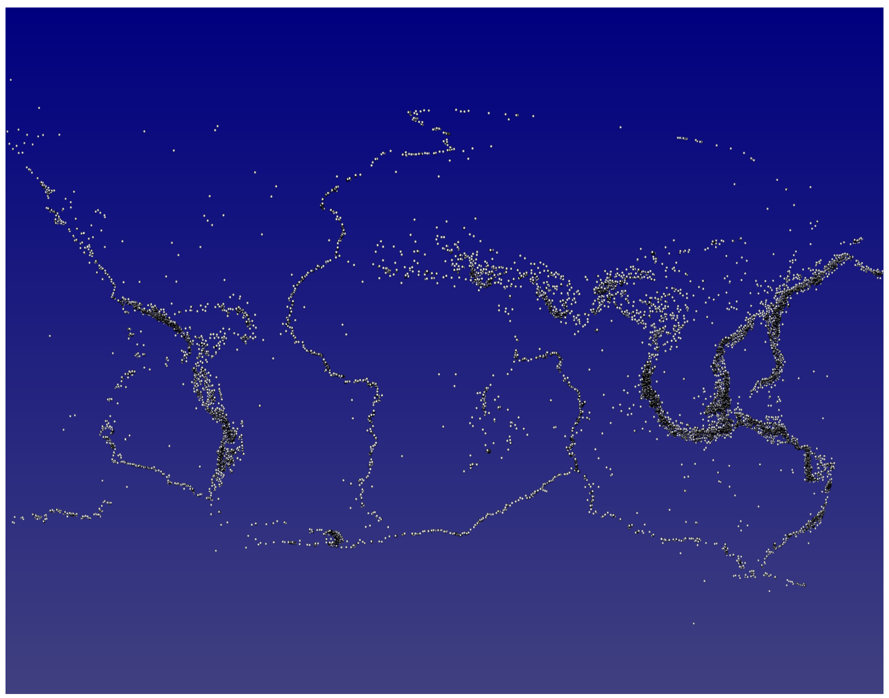


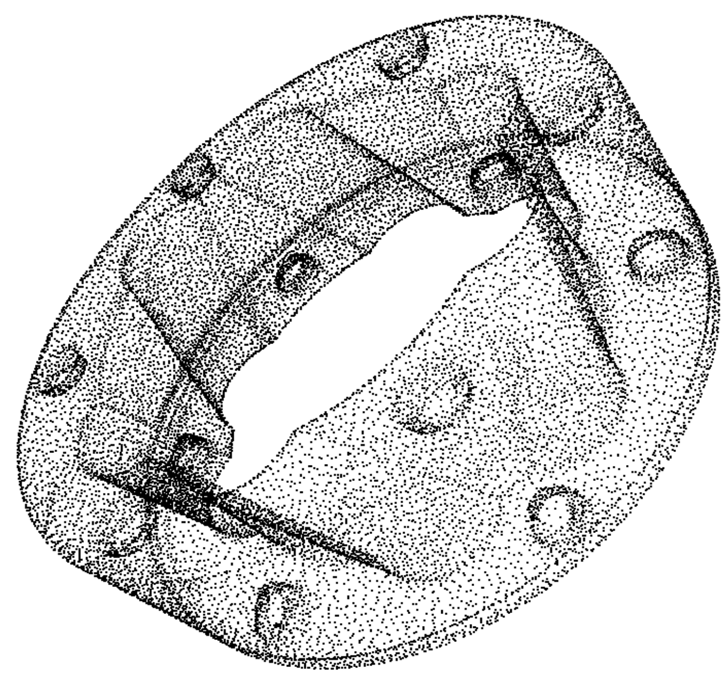


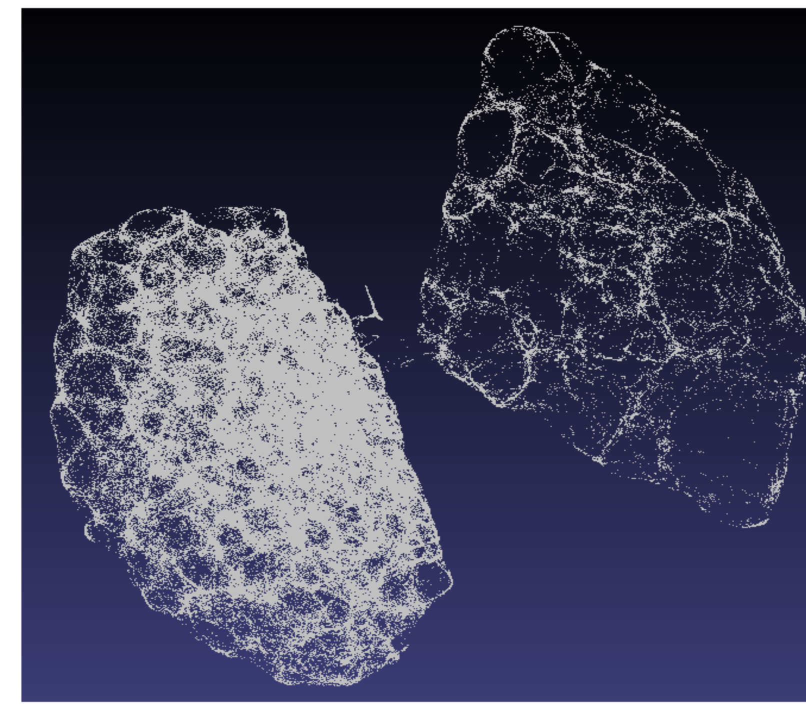


Perspectives: topological and geometric data analysis

Previous methods usually comes with theoretical guarantees when M is a smooth manifold with trivial topology.







- Geometric structures underlying data sets may carry complex topology/geometry.
- What is the relevant topology/geometry of a point cloud data set?

A recent and fastly growing fields based upon topology and geometric measure tools