MTTTS17
Dimensionality Reduction and Visualization

Spring 2019
Jaakko Peltonen

Lecture 8: Nonlinear dimensionality reduction, part 3

Brief Recap

Faithfully?

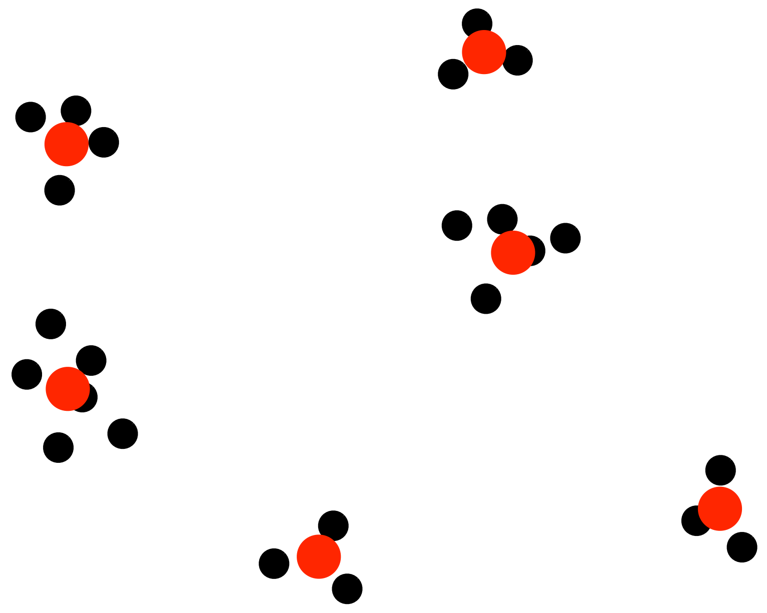
- Good Precision: Points that are close in the “reduced” space are close in the original space
- Good Recall: Points that are close in the original space are close in the “reduced” space
- In general, impossible to get both

Vector Quantization

- minimizes the distortion

$$E = \frac{1}{NV} \sum_{i=1}^{N} \|x_i - v(x_i)\|^2$$

- with N the number of points (samples), V the number of centroids (units), x the samples, v the centroids



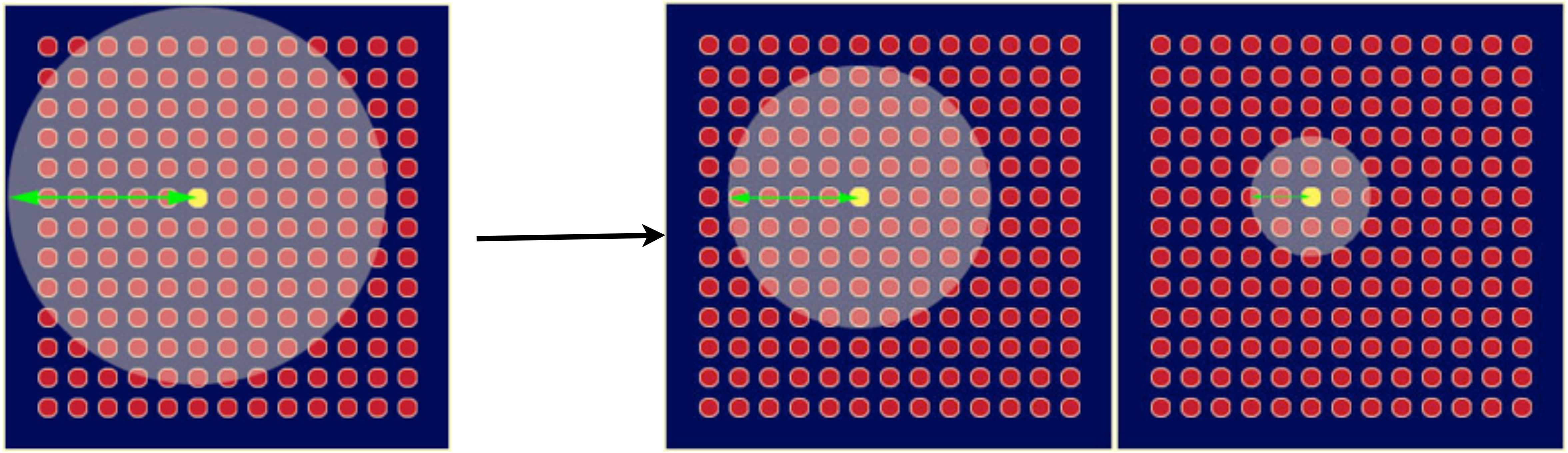
SOM

- Random initialization or PCA initialization
- Iterations

$$u(t+1) = u(t) + h_{u,v}(t)\alpha(t)(x(t)-u(t))$$

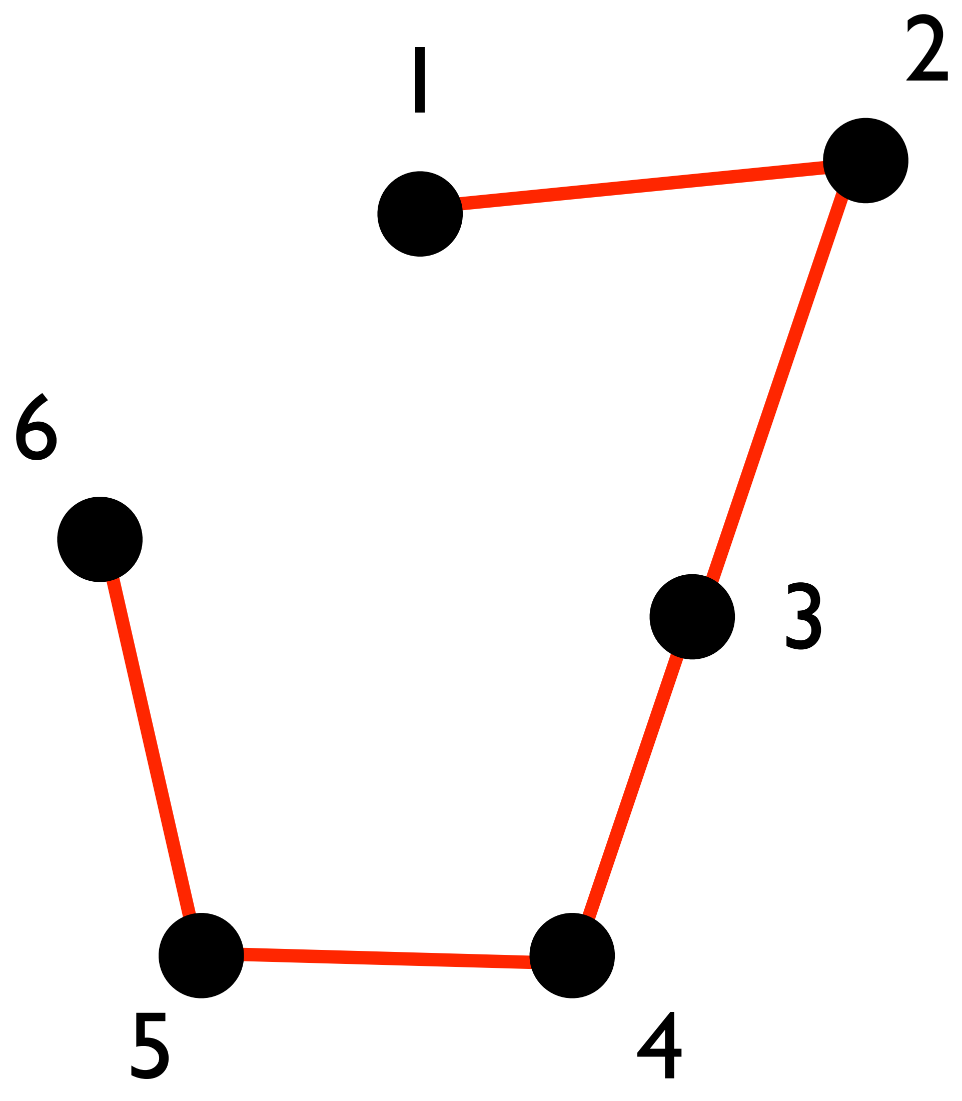
$$h_{u,v}(t)=\exp\left(\frac{-d_{grid}(u,v)}{\sigma(t)}\right) \quad \sigma(t)=\sigma_0\exp\left(\frac{-t}{\lambda}\right)$$

- Example:



Laplacian eigenmap

```
>> L

L =

    -1     1     0     0     0     0
     1    -2     1     0     0     0
     0     1    -2     1     0     0
     0     0     1    -2     1     0
     0     0     0     1    -2     1
     0     0     0     0     1    -1

>> [V,D]=eig(L)

V =

    0.1494    0.2887    0.4082   -0.5000    0.5577    0.4082
   -0.4082   -0.5774   -0.4082    0.0000    0.4082    0.4082
    0.5577    0.2887   -0.4082    0.5000    0.1494    0.4082
   -0.5577    0.2887    0.4082    0.5000   -0.1494    0.4082
    0.4082   -0.5774    0.4082    0.0000   -0.4082    0.4082
   -0.1494    0.2887   -0.4082   -0.5000   -0.5577    0.4082


D =

   -3.7321         0         0         0         0         0
         0   -3.0000         0         0         0         0
         0         0   -2.0000         0         0         0
         0         0         0   -1.0000         0         0
         0         0         0         0   -0.2679         0
         0         0         0         0         0   -0.0000
```



Curvilinear component analysis (CCA)

- Demartines, Hérault, 1997.
- *Curvilinear component analysis* (CCA) is like (absolute) MDS, except that only short distances are taken into account.
- More formally, the cost function reads

$$\sigma_r = \sum_{i<j} \left(d(x_i, x_j) - d(y_i, y_j)\right)^2 F(d(y_i, y_j), \lambda_y)$$

where *F(d,*λ_y*)* equals unity, if *d*<λ_y, and zero otherwise; and *d* denotes the Euclidean distance of points in the original space (*x*) and in the projection (*y*), respectively. (Actually, *F(d,*λ_y*)*, could be any monotonically decreasing function in *d*.)

Mathematical Framework

A General View of Dimensionality Reduction

Assume we have

- a finite set of data points $\mathbf{X} = (\mathbf{x}^i \in \mathbb{R}^N | i = 1 \dots n)$
- corresponding projections $\mathbf{\Xi} = (\xi^{\mathbf{i}} \in \mathbb{R}^M, i = 1 \dots n)$
- a sequence of tuples of data points and their projections

$$\mathbf{X\Xi} = ((\mathbf{x}^1, \xi^1), \dots (\mathbf{x}^n, \xi^n))$$

- denote the set of all finite subsequences of a given set A by S(A) (for example $A = \mathbb{R}^N$)
- length of a given sequence $\mathbf{X} = (\mathbf{x}^1, \dots, \mathbf{x}^n)$: $|\mathbf{X}| = n$

For all methods the coefficients ξ^i are determined based on the same general principle using the same basic ingredients

General Principle of Dimensionality Reduction

Determine projections ξ^i by minimizing an error measure between the characteristics derived from the original training set **X** and corresponding characteristics of its projections:

- function $\mathrm{char}_{\mathcal{X}} : S(\mathbb{R}^N) \times \mathbb{R}^N \to S(\mathbb{R})$ maps data sequence **X** and point **x** in $\mathbb{R}^N$ to some characteristics of the point **x** that we want to preserve. Usually $|\mathrm{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x})| = |\mathbf{X}|$.

  For example in MDS, the characteristics of a point **x** are the vector of squared distances from **x** to all other points in **X**.

- function $\mathrm{char}_{\mathcal{E}} : S(\mathbb{R}^M \times \mathbb{R}^N) \times (\mathbb{R}^M \times \mathbb{R}^N) \to S(\mathbb{R})$ maps a subset $\mathbf{X\Xi}$ of points and their projections and a given tuple of a point and its projection. Usually $|\mathrm{char}_{\mathcal{E}}(\mathbf{X\Xi}, (\mathbf{x}, \xi))| = |\mathbf{X\Xi}|$

  In MDS, the characteristics of a projected point ξ^i are the vector of squared distances from ξ^i to all other projected points. (In this case the characteristics are computed from projected data only.)

General Principle of Dimensionality Reduction

- error measure $S(\mathbb{R}) \times S(\mathbb{R}) \to \mathbb{R}$ (difference of characteristics)

  In MDS the error for the i:th data point is the sum of squared differences between the low-dimensional and high-dimensional characteristics (low-dimensional and high-dimensional squared distances).

- DR: given finite sequence $\mathbf{X} \in S(\mathbb{R}^N)$ determine the projections ξ^i for every $\mathbf{x}^i$ such that the costs are minimal:

$$\mathrm{costs}(\mathbf{X\Xi}) := \sum_{\mathbf{x}^i \in \mathbf{X}} \mathrm{error}(\mathrm{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}^i), \mathrm{char}_{\mathcal{E}}(\mathbf{X\Xi}, (\mathbf{x}^i, \xi^i)))$$

- possible constraints: $S(\mathbb{R}^M \times \mathbb{R}^N) \to \mathbb{R}$ imposed on ξ^i to guarantee uniqueness (optimized simultaneously with the costs)

General Framework

$$\begin{tikzcd}
\mathbf{x} \in \mathbb{R}^N \arrow[r] \arrow[d, "\text{minimize costs}(\mathbf{X}\boldsymbol{\Xi})"', "f:\mathbb{R}^N \to \mathbb{R}^M"] & \text{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}) \arrow[d, "\text{error}(\text{char}_{\mathcal{X}}, \text{char}_{\mathcal{E}})"] \\
\boldsymbol{\xi} \in \mathbb{R}^M \arrow[r] & \text{char}_{\mathcal{E}}(\mathbf{X}\boldsymbol{\Xi}, (\mathbf{x}, \boldsymbol{\xi}))
\end{tikzcd}$$

- implicit mapping: $f : \mathbb{R}^N \to \mathbb{R}^M, \mathbf{x}^i \to \boldsymbol{\xi}^i$ optimize the representatives $\boldsymbol{\xi}^i$ directly
- explicit mapping function $f_W : \mathbb{R}^N \to \mathbb{R}^M, \mathbf{x}^i \to f_W(\mathbf{x}^i) = \boldsymbol{\xi}^i$ optimize parameters W of the mapping function, direct out-of-sample extension for a new sample $f_W(\widehat{\mathbf{x}}) = \widehat{\boldsymbol{\xi}}$
- Additional information can be part of the framework:
  - label information y^i for samples $\mathbf{x}^i$ (supervised methods)
  - optimization constraints

General Principle of Dimensionality Reduction

Methods might differ

- in the definition of the characteristics
- in the way the error is measured
- in implicit or explicit computation of the characteristics
- in the optimization (analytical or numerical)
- in the way possible constraints are included
  (objective and constraints might be contradictory)

More methods:
Isomap,
Locally Linear Embedding,
Maximum Variance Unfolding

Isomap

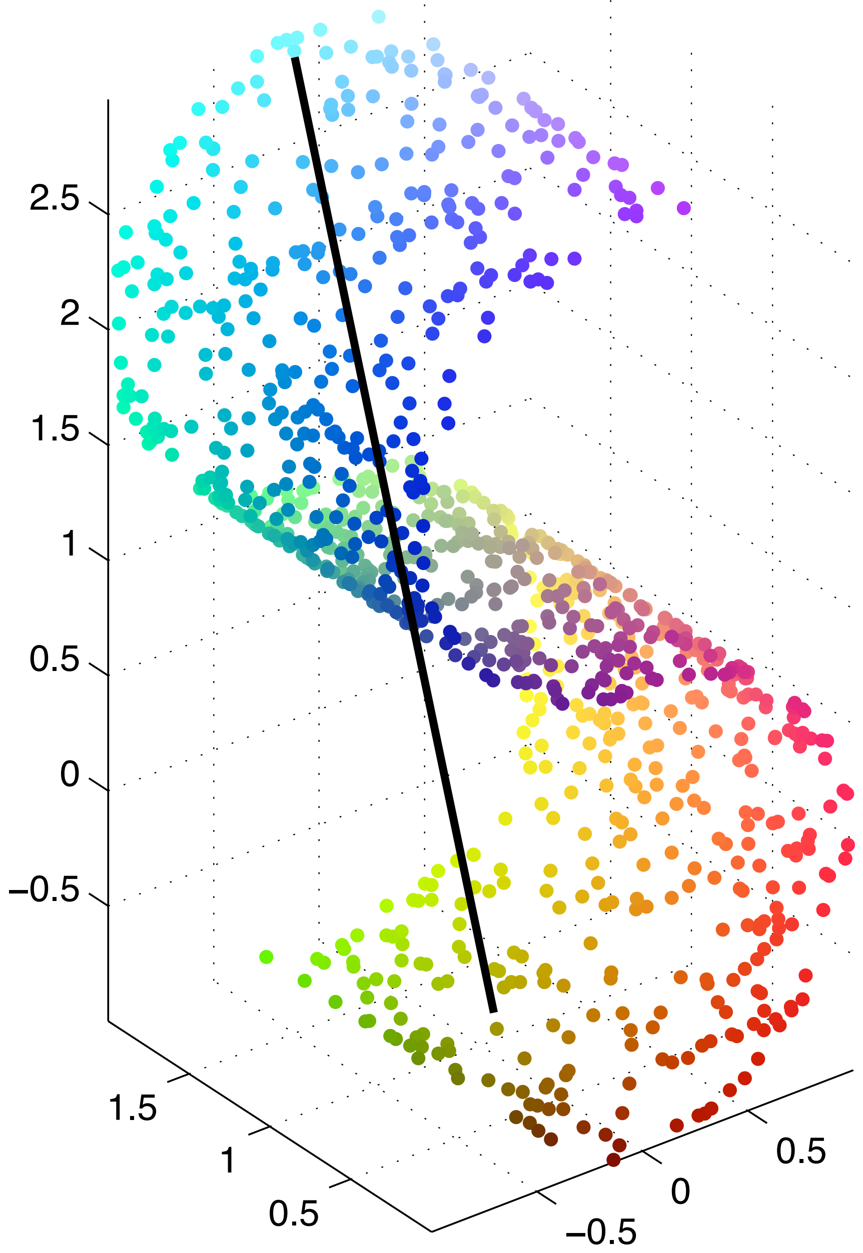
Multidimensional scaling methods tried to preserve all squared distances —> preservation of largest distances had biggest effect on the cost

Methods like Sammon’s mapping and Curvilinear Component Analysis tried to focus more on accurate preservation of small distances in the original space (Sammon) or accurateness of small on-screen distances (Curvilinear Component Analysis).

These methods essentially partly sacrifice preservation of large distances in favour of preserving small ones.
What if we want to try to also preserve large distances?

Isomap

If the data lies along a manifold embedded in a high-dimensional space, long Euclidean distances might not follow the manifold. Therefore directly preserving long Euclidean distances would not "unfold" the manifold.

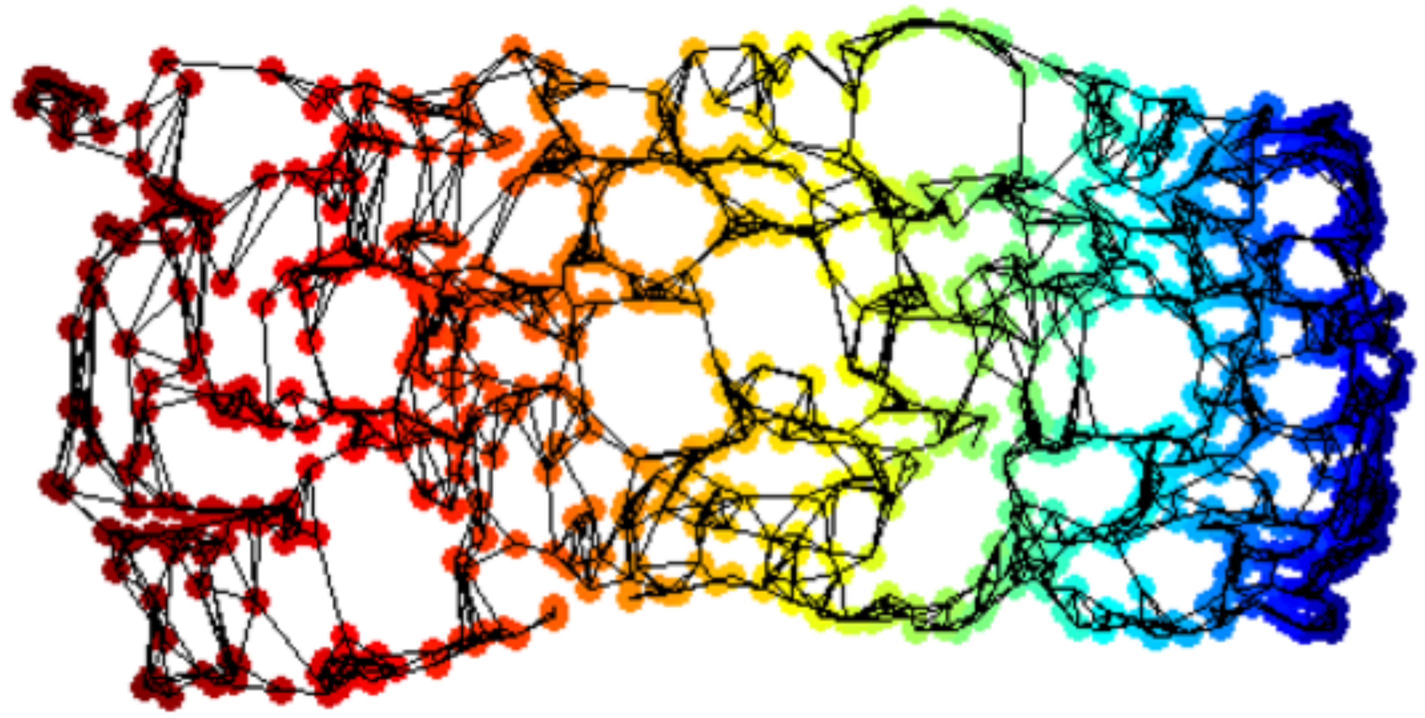


Euclidean distance between two points corresponds to the length of the line connecting the points. The line usually does not follow the manifold of the data.

Isomap

Euclidean distance in the original space might not be appropriate ---> replace by *geodesic distance* (Joshua B. Tenenbaum, Vin de Silva, and John C. Langford, “A Global Geometric Framework for Nonlinear Dimensionality Reduction”, *Science*, 2000)

Approximate geodesic distance as shortest distance along a *neighbourhood graph* of the data.

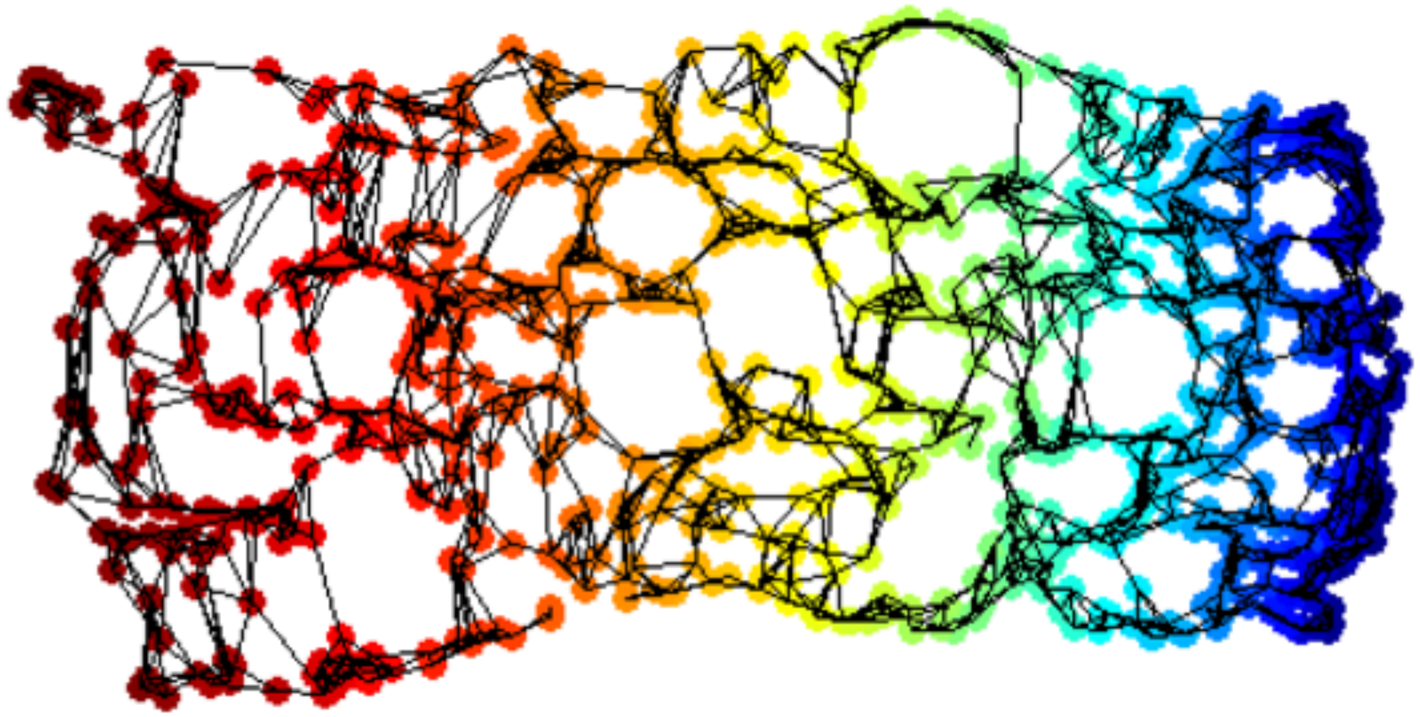


Isomap

Construct neighborhood graph (k neighborhood or ϵ-balls) and compute shortest path length along the graph between all pairs of points:

- first compute distance to the neighbors of each point, and write this as a distance matrix (set distance to non-neighbors to infinity).
- Then use Dijkstra's algorithm to find shortest distance along the graph from a point to all other points.

This defines pairwise affinities (distances), and therefore defines the characteristics $\mathrm{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x})$ of the data to be preserved.

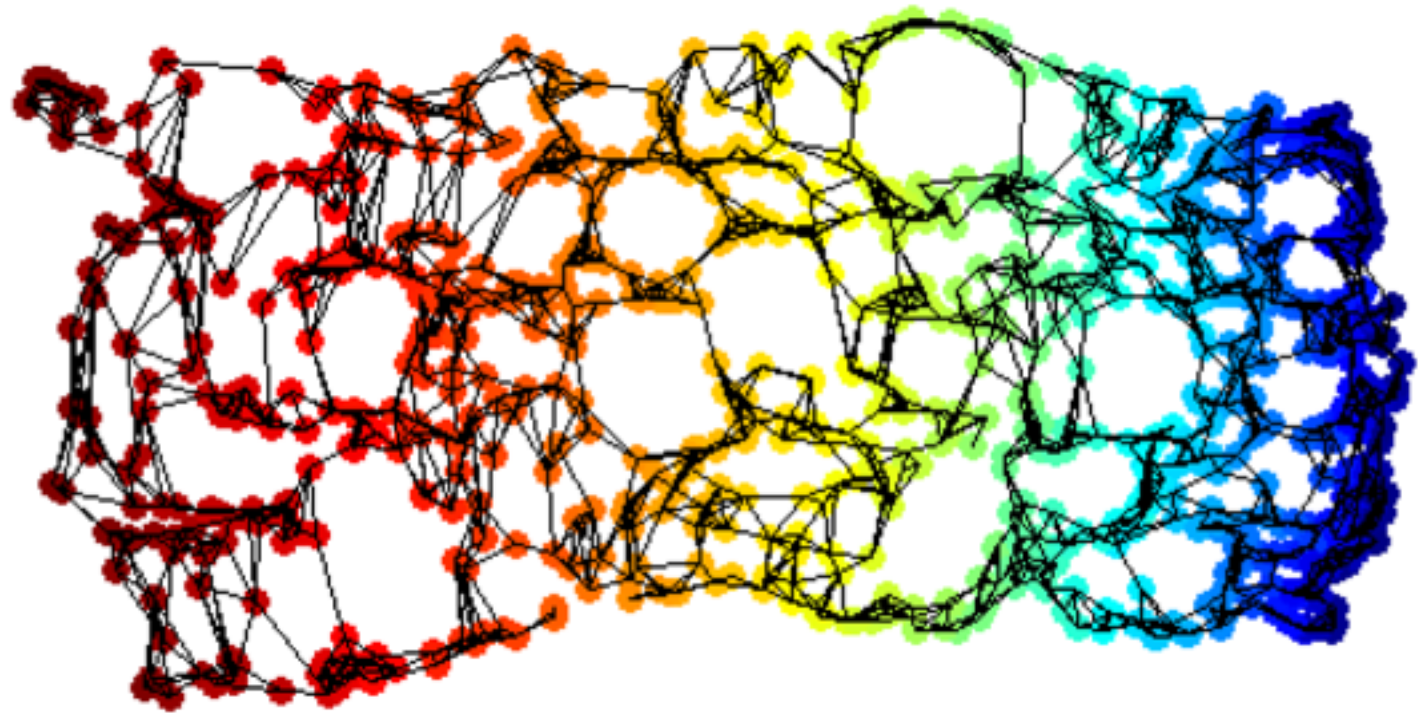


Isomap

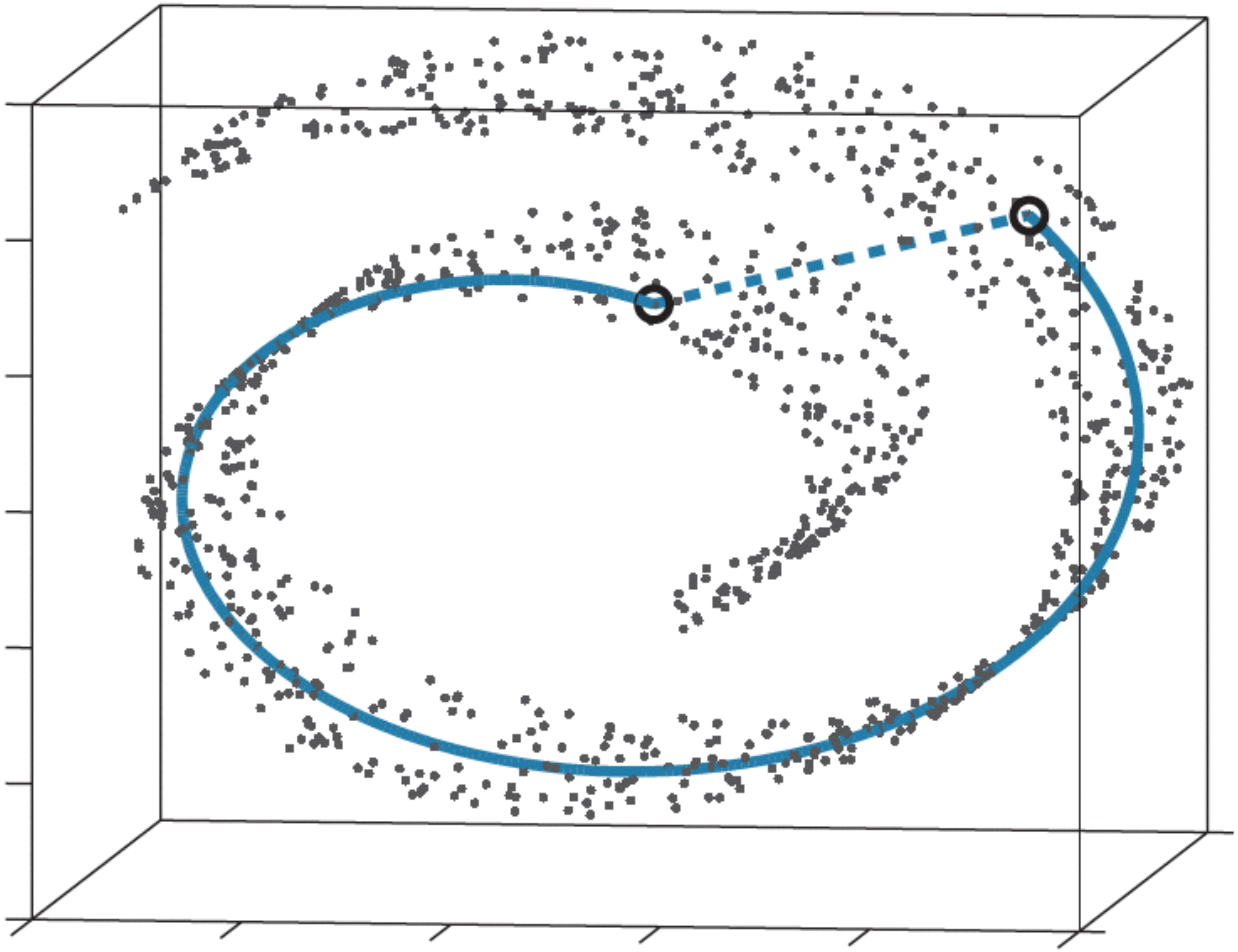
Afterwards follow standard MDS procedure: find low-dimensional coordinates whose Euclidean squared distances best approximate the above-defined squared shortest-path distances.

Standard MDS code can be used, the difference is only in how the original distances are computed.

(In brief: center the distance matrix - subtract mean of rows from each row, subtract mean of columns from each column, subtract mean of elements from each element. Compute eigenvectors of the centered matrix, eigenvectors corresponding to largest eigenvalues give the reduced coordinates.)

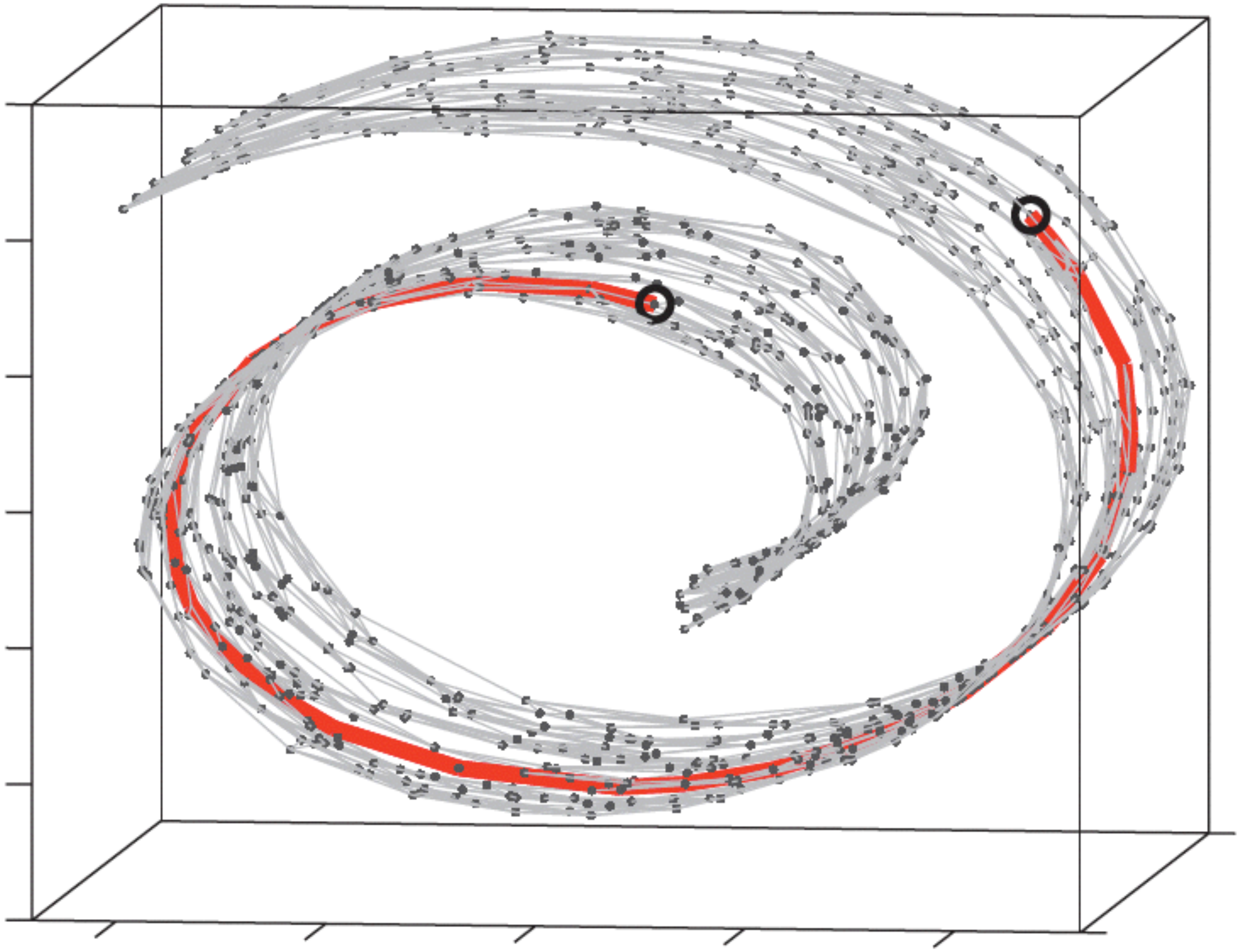


Isomap



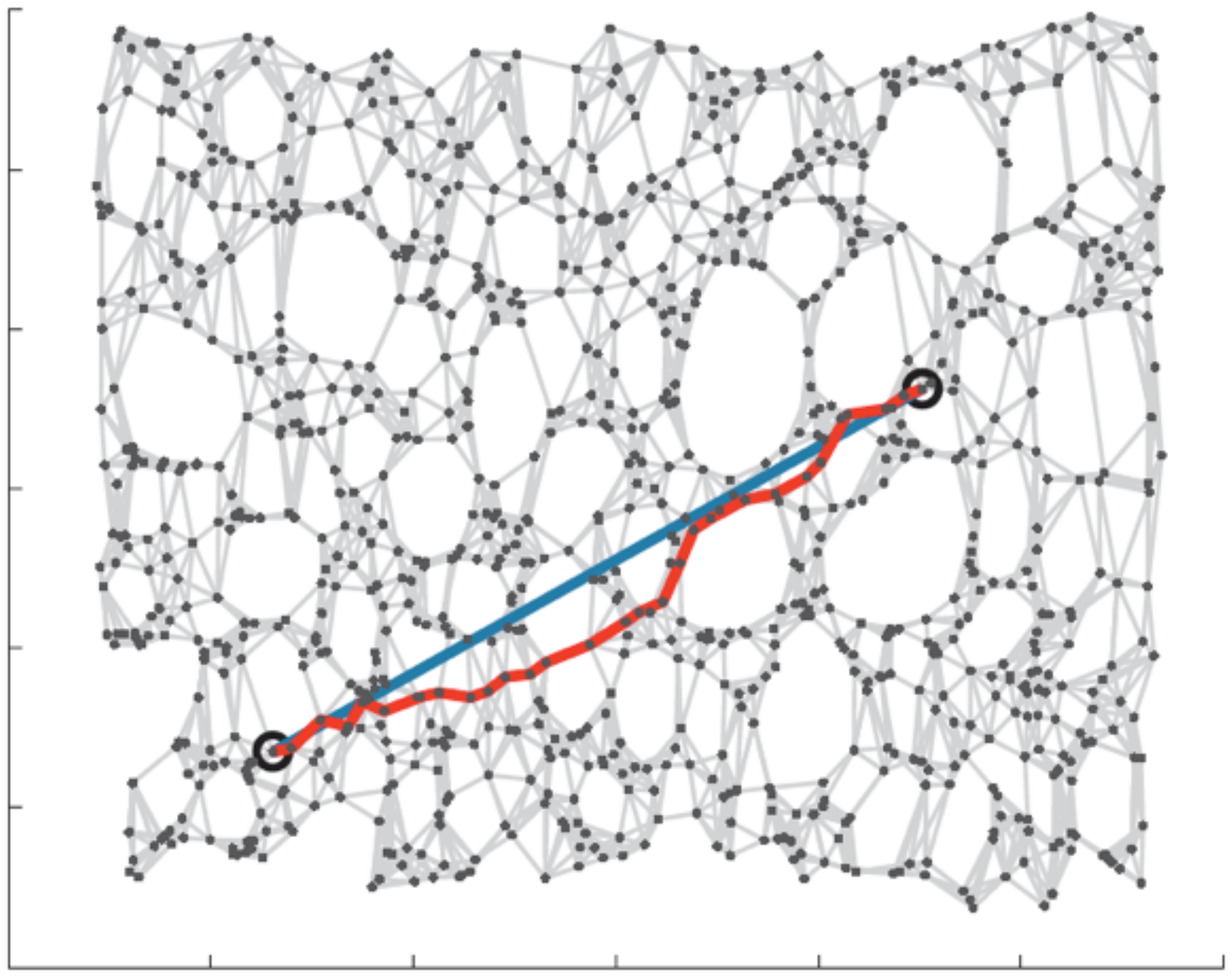
Swiss roll example: Euclidean distance can “jump” across the manifold, while the “ideal” distance goes along the manifold.

Isomap



Neighborhood graph (with N=1000 data points and K=7 neighbors connected to each point), geodesic approximated by shortest path along the graph.

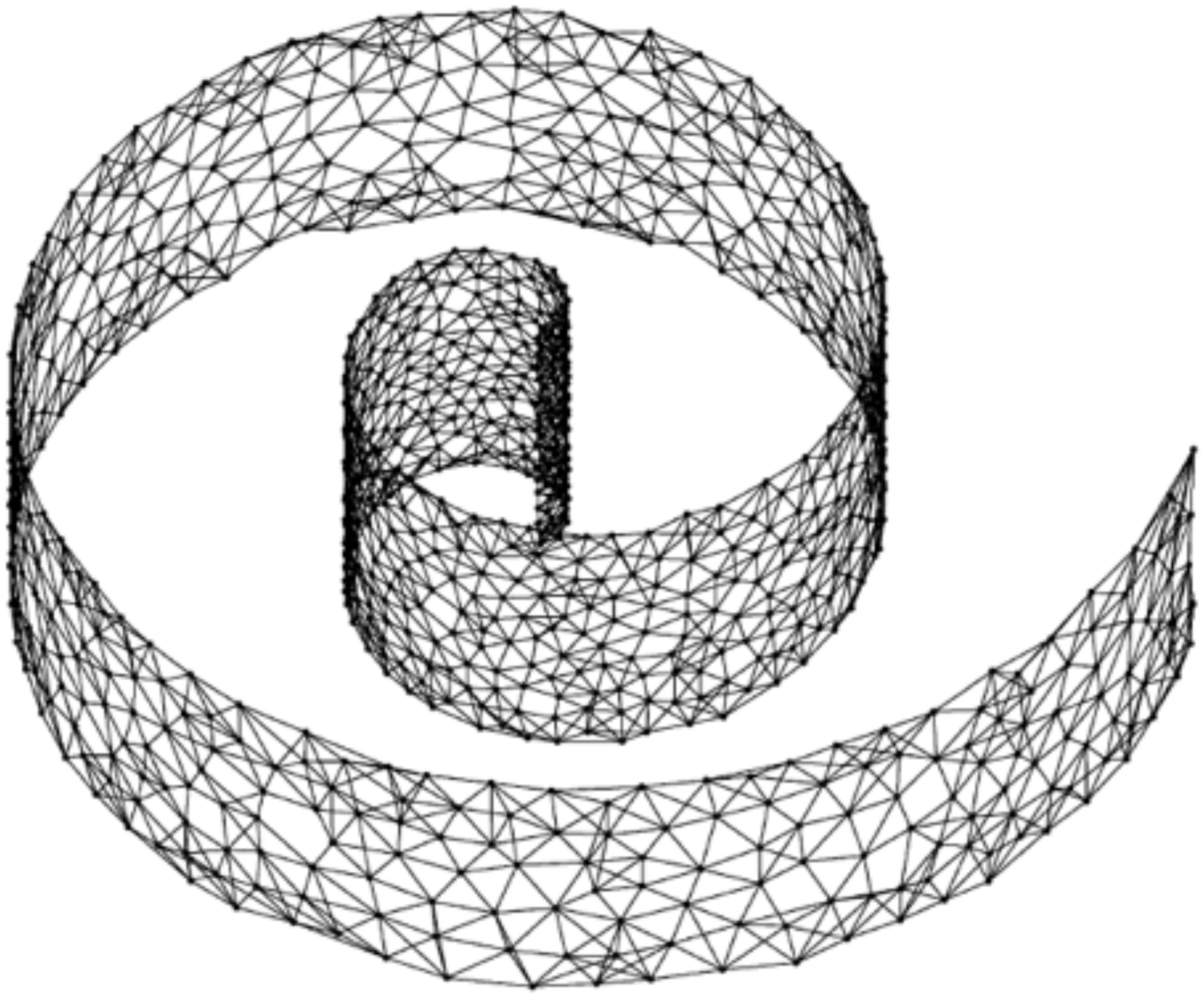
Isomap



Two-dimensional embedding computed by MDS, to preserve the resulting approximate squared geodesic distances, as squared Euclidean distances on the display.

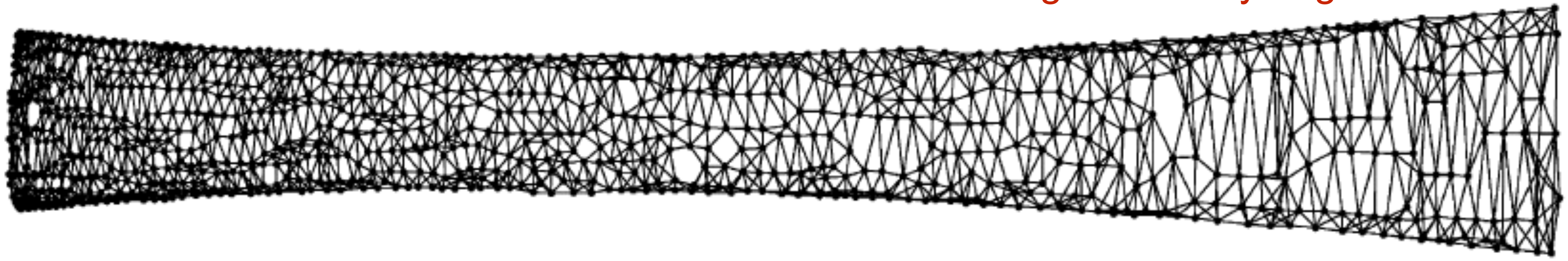
Note: geodesics are approximated directly by the Euclidean (straight-line) low-dim. distances, not by shortest paths along the graph.

Isomap



Downsides of Isomap: distances along the graph are only approximations of the true geodesic distances. They overestimate the distances, especially large distances, when data is sparse (less “waypoints” to choose from—>suboptimal path).

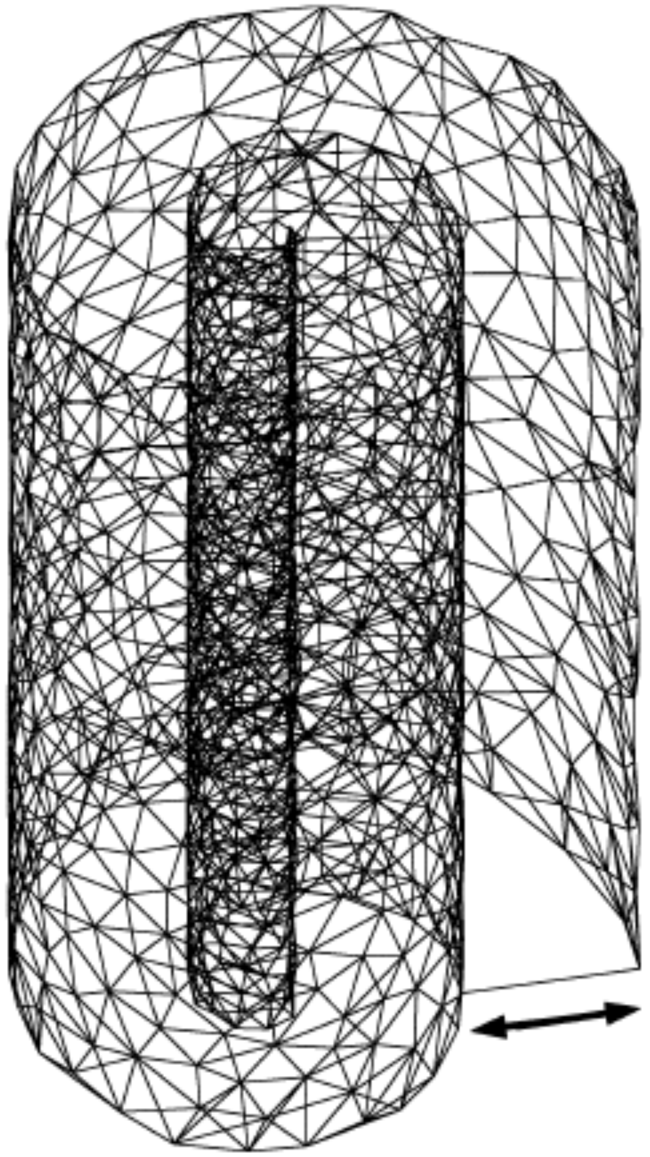
Overestimation of distances causes overstretching of faraway edges

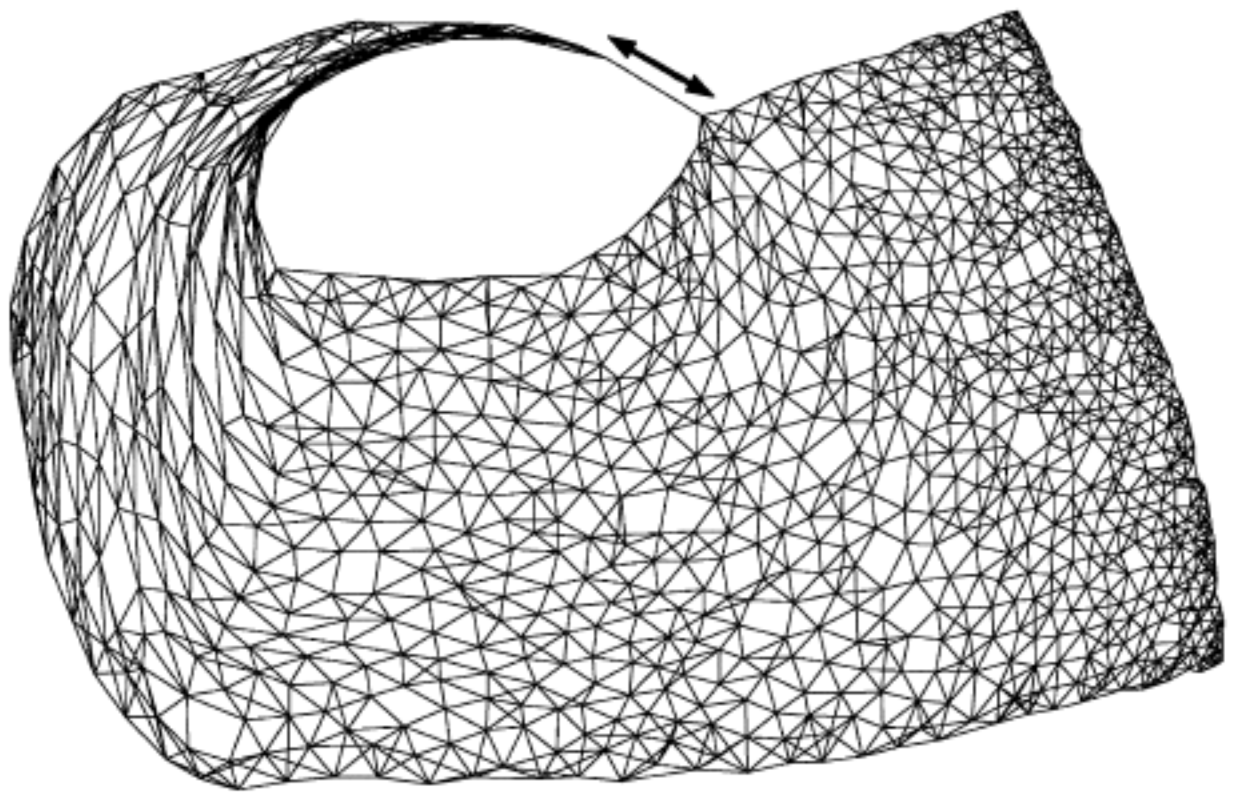


From: John Aldo Lee, Amaury Lendasse , Michel Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. Neurocomputing 57 (2004) 49 – 76.

Isomap

Downsides of Isomap: poorly constructed neighbourhood graphs can distort the approximation of geodesic distances (especially longer distances), and distort the resulting embedding.





From: John Aldo Lee, Amaury Lendasse , Michel Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. Neurocomputing 57 (2004) 49 – 76.

Curvilinear distance analysis (CDA)

J.A. Lee, A. Lendasse, N. Donckers, M. Verleysen, “A robust nonlinear projection method”, in: M. Verleysen (Ed.), Proceedings of ESANN’2000, Eighth European Symposium on ArtiHcial Neural Networks, D-Facto Publications, Bruges, Belgium, 2000, pp. 13–20.

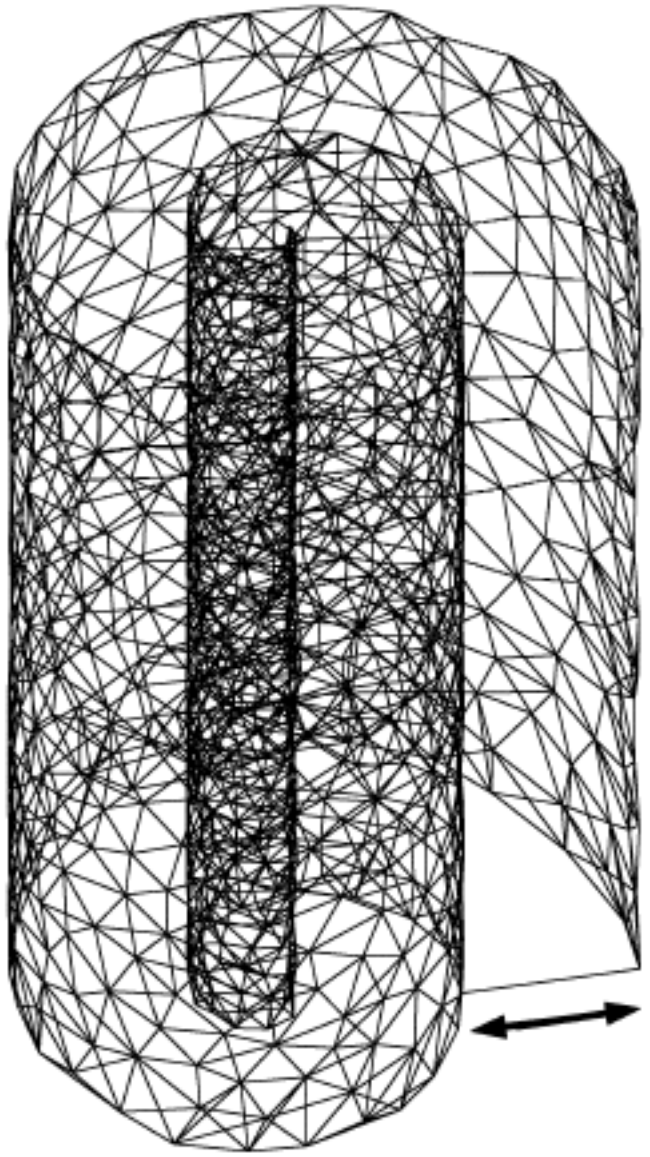
- *Curvilinear component analysis* (CCA) is like (absolute) MDS, but only short distances are taken into account.
- More formally, the cost function reads

$$\sigma_r = \sum_{i<j} \left(d(x_i, x_j) - d(y_i, y_j)\right)^2 F(d(y_i, y_j), \lambda_y)$$

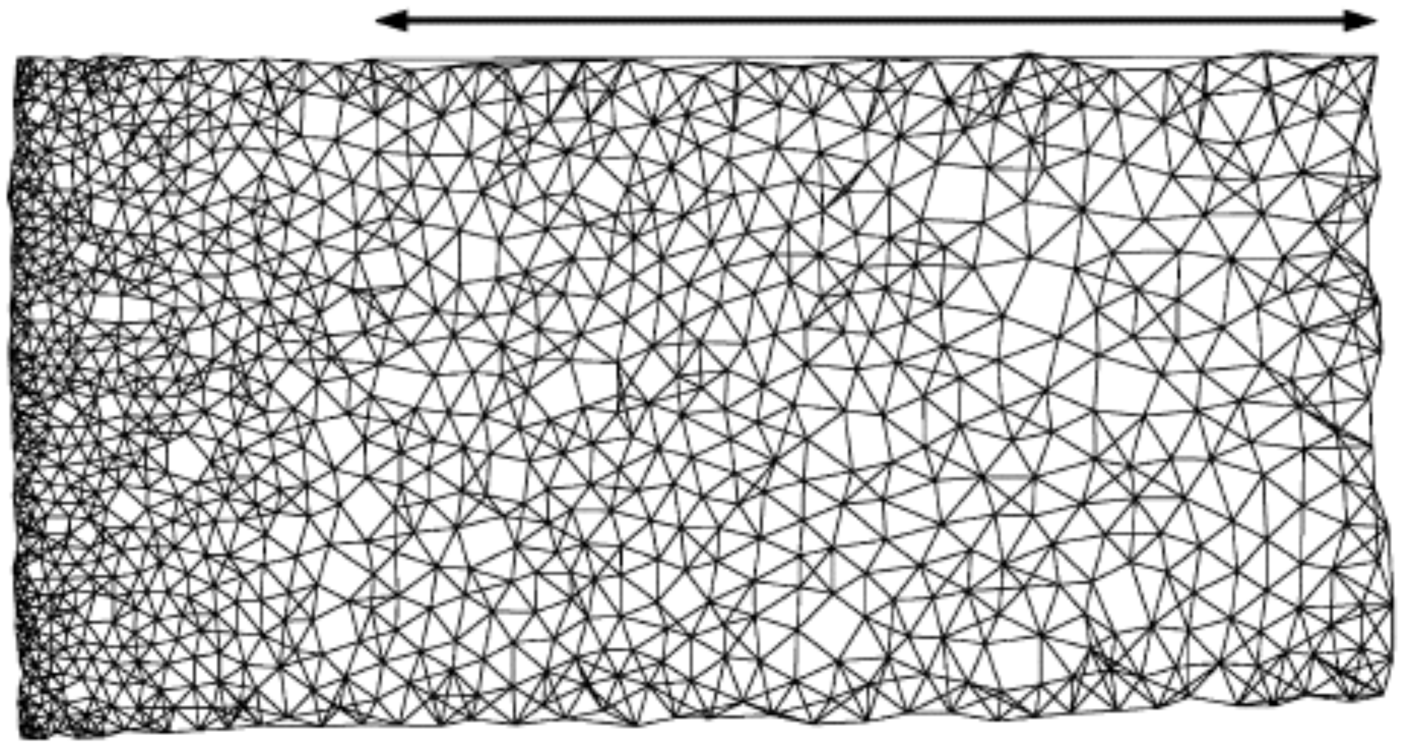
where $F(d,\lambda_y)$ equals unity, if $d<\lambda_y$, and zero otherwise; and *d* denotes the Euclidean distance of points in the original space (*x*) and in the projection (*y*), respectively. (Actually, $F(d,\lambda_y)$, could be any monotonically decreasing function in *d*.)

- Curvilinear distance analysis (CDA) is the same, except the $d(x_i,x_j)$ are computed as distances along a neighbourhood graph, just like in Isomap!

Curvilinear Distance Analysis

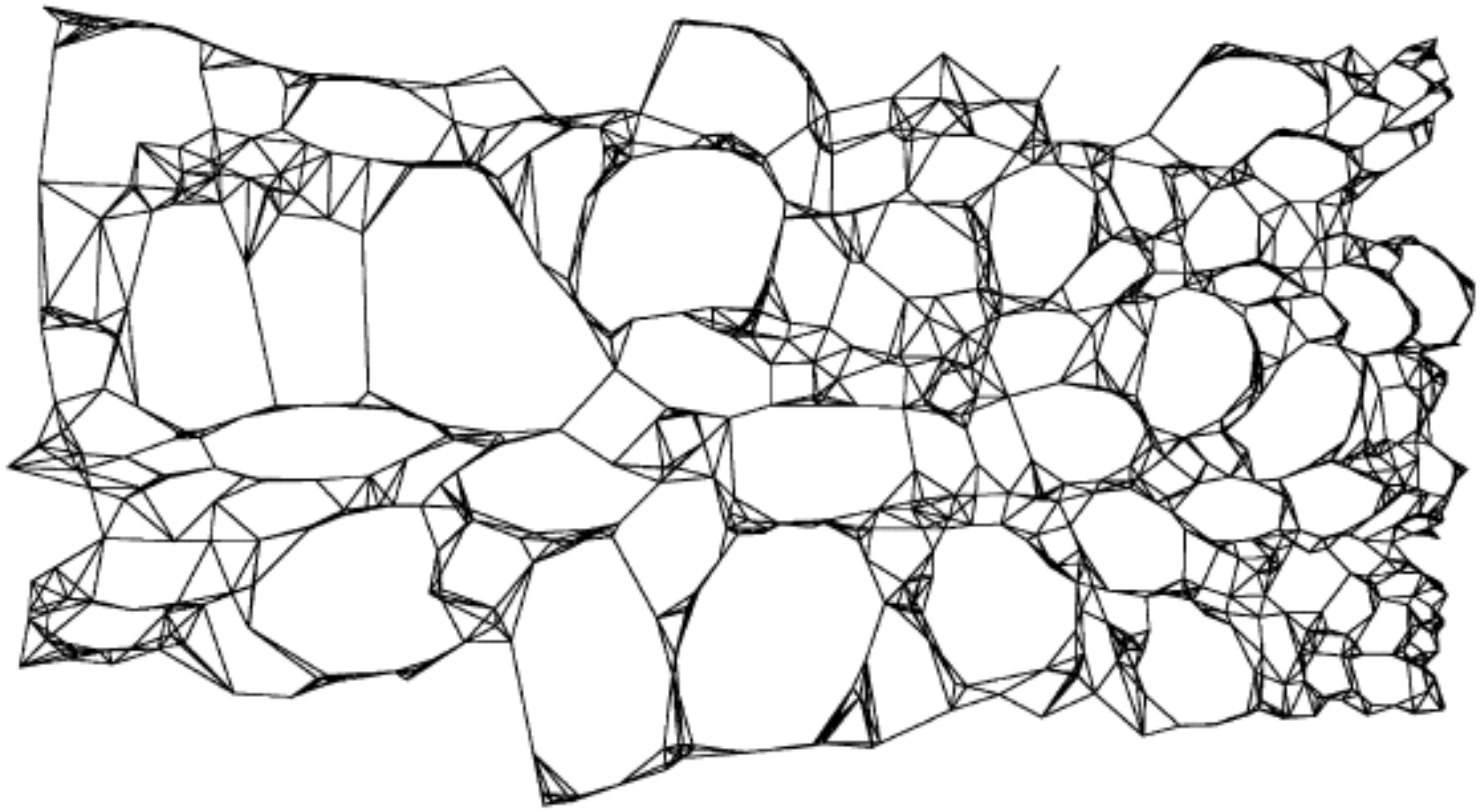


In CDA poorly approximated longer distances don't distort as much as in Isomap, since the embedding concentrated on small distances.



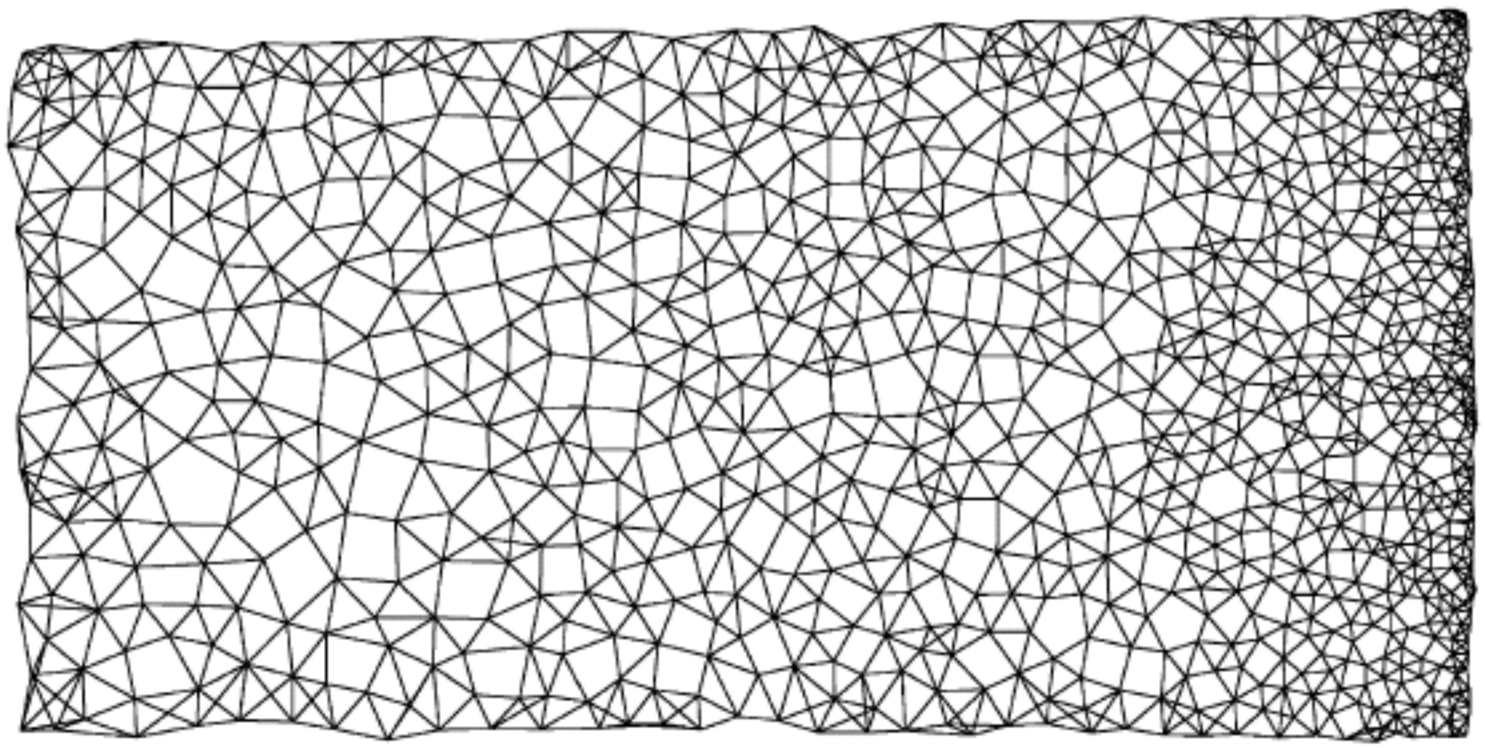
From: John Aldo Lee, Amaury Lendasse , Michel Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. Neurocomputing 57 (2004) 49 – 76.

Curvilinear distance analysis (CDA)



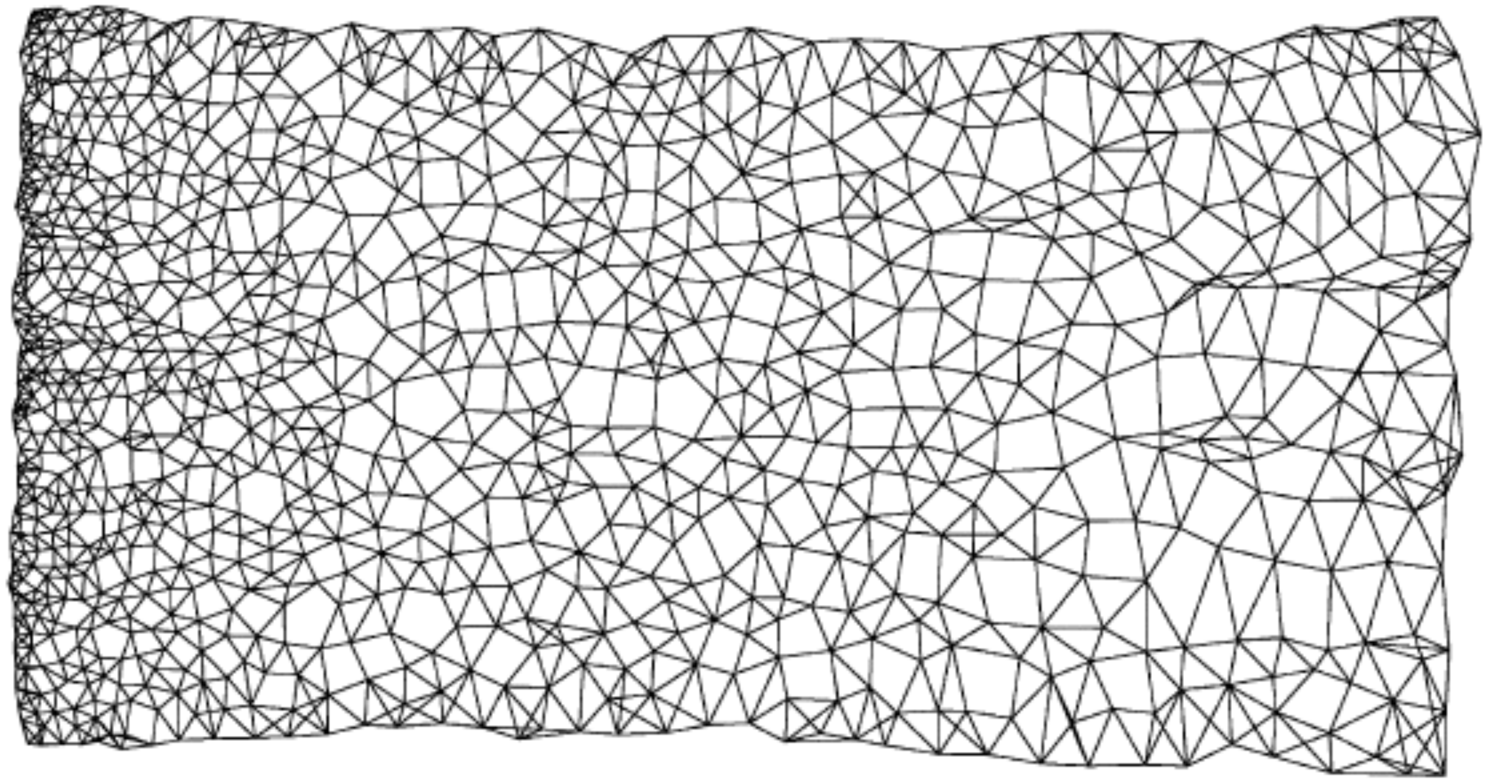
Some implementations of Isomap use a random sampling of a subset of points, to speed up computation. This can greatly distort distances (top).

If one must use a subset, it is better to create the subset by vector quantisation methods, as is done in Curvilinear Distance Analysis (bottom).

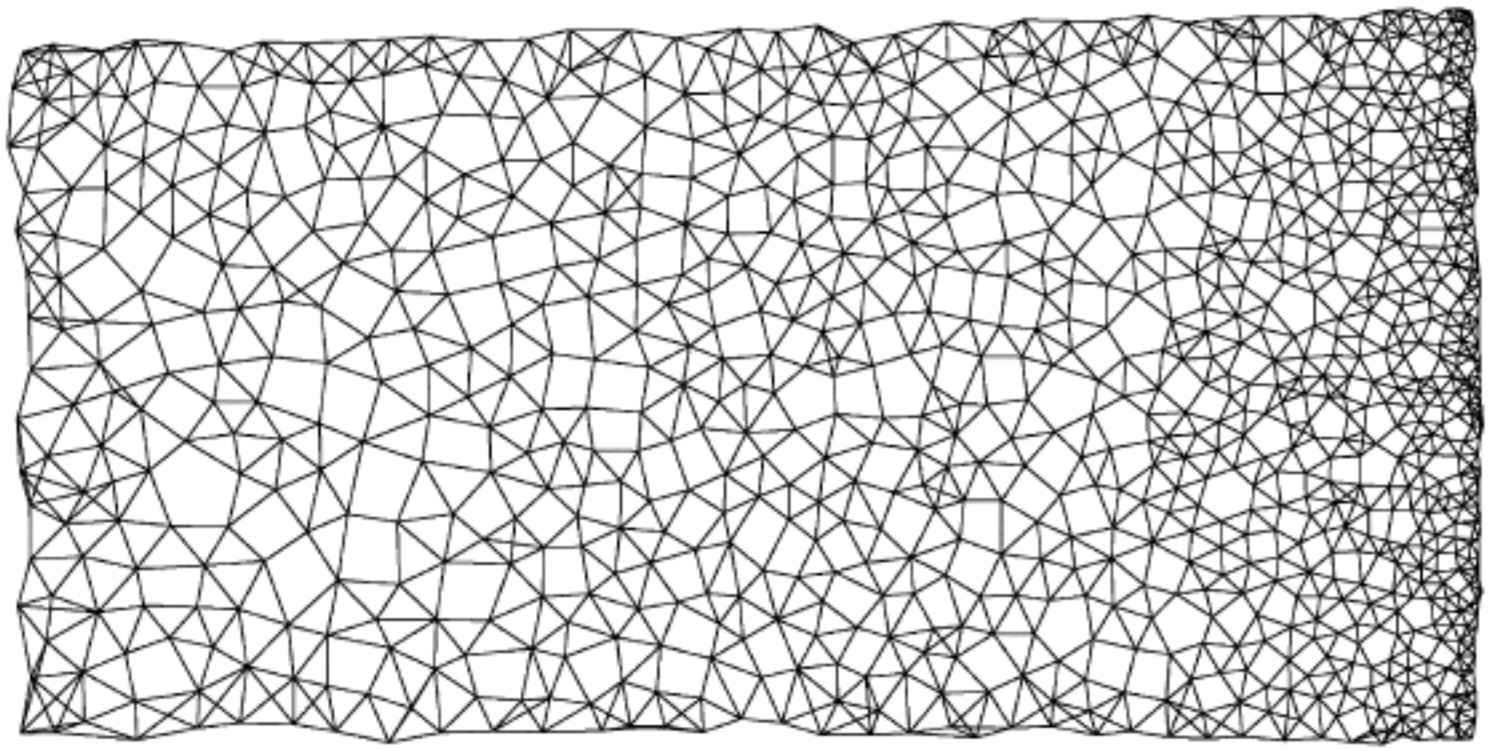


From: John Aldo Lee, Amaury Lendasse, Michel Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. Neurocomputing 57 (2004) 49 – 76.

Curvilinear distance analysis (CDA)



Same data set, but now Isomap uses vector quantisation to create the subset (top). Now the quality is closer to that of Curvilinear Distance Analysis (bottom).



From: John Aldo Lee, Amaury Lendasse , Michel Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. Neurocomputing 57 (2004) 49 – 76.

Curvilinear distance analysis (CDA)

In principle approximated geodesic distances could similarly be used as inputs in any method that is based on distances (for example in Sammon’s mapping).

Autoencoders

Instead of preserving distances etc., why not try to directly reconstruct positions of original data points?

Recall that PCA could be seen as a linear dimensionality reduction method that minimised a squared-distance *reconstruction error* from reconstructed positions of data to the original positions. Can we do something similar with nonlinear projections?

Autoencoders (autoencoder networks) are a simple nonlinear extension of the concept: try to find the best low-dimensional nonlinear mapping, such that the original data point coordinates can be reconstructed as well as possible from the low-dimensional coordinates by another mapping.

Autoencoders

$$f_W : \mathbb{R}^N \to \mathbb{R}^M, \ \mathbf{x}^i \to f_W(\mathbf{x}^i) = \boldsymbol{\xi}^i$$

Highdim—>lowdim mapping, parameters W

$$g_V : \mathbb{R}^M \to \mathbb{R}^N, \ \boldsymbol{\xi}^i \to g_V(\mathbf{x}^i) = \hat{\mathbf{x}}^i$$

Lowdim—>highdim mapping, parameters V

$$\text{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}^i) = \mathbf{x}^i$$

Characteristics of an original point are simply its coordinates

$$\text{char}_{\mathcal{E}}(\mathbf{X}\boldsymbol{\Xi}, (\mathbf{x}^i, \boldsymbol{\xi}^i)) = g_V(\boldsymbol{\xi}^i)$$

Characteristics of a low-dimensional point are the high-dimensional coordinates reconstructed from it

$$\begin{aligned}&\text{error}(\text{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}^i), \text{char}_{\mathcal{E}}(\mathbf{X}\boldsymbol{\Xi}, (\mathbf{x}^i, \boldsymbol{\xi}^i)))\\ &\quad = ||\mathbf{x}^i - g_V(\boldsymbol{\xi}^i)||^2 = ||\mathbf{x}^i - g_V(f_W(\mathbf{x}^i))||^2\end{aligned}$$

Error = reconstruction error. Minimize over data points with respect to V and W.

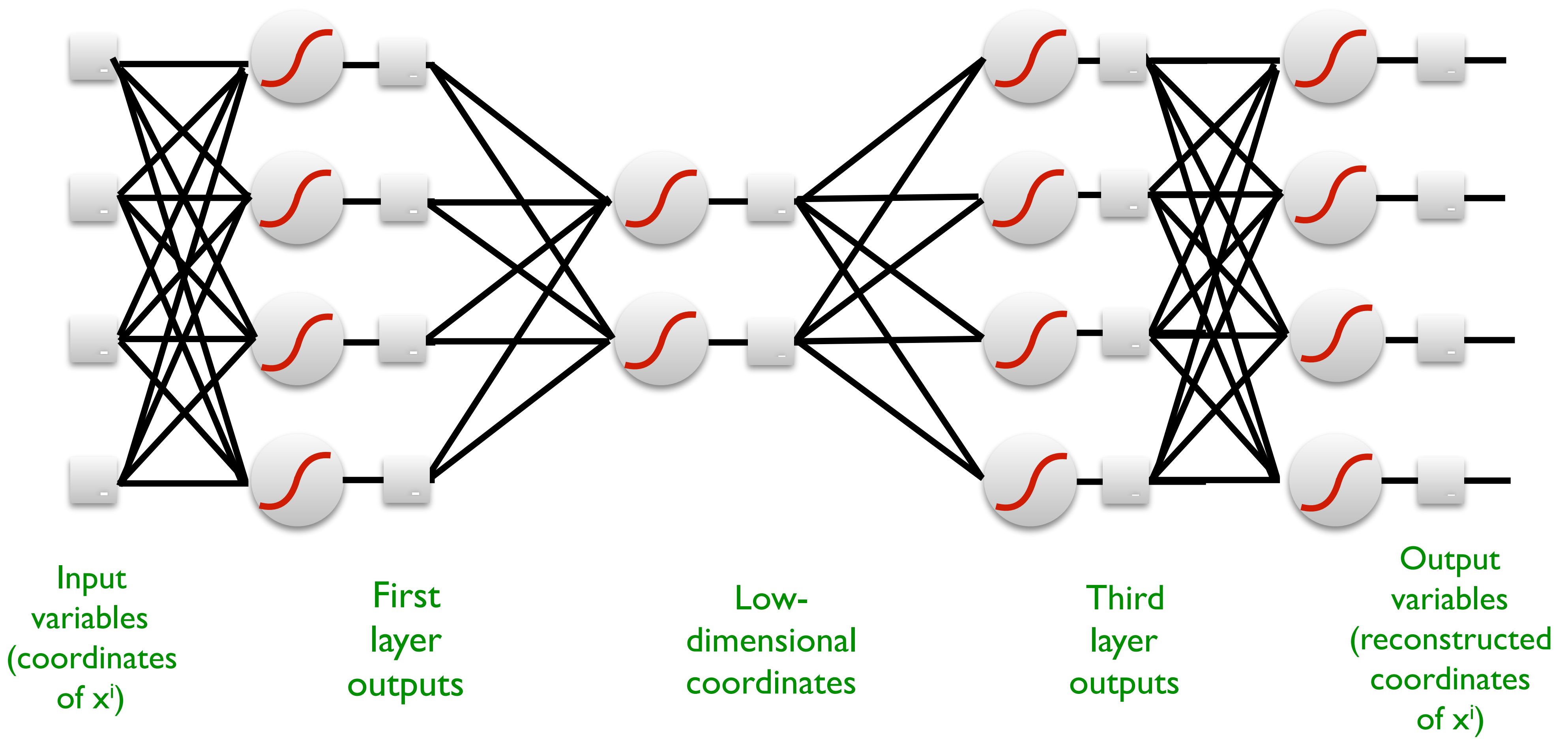
Autoencoders

Autoencoders could be created using any parametric highdim—>lowdim mapping f and parametric lowdim—>highdim mapping g.

Both mappings can be realised as one multilayer neural network. Each neuron (circle) computes a weighted sum of its inputs, followed by a nonlinear transformation of the sum (e.g. a logistic sigmoid).

$$s = \sum_j w_j x_j, \quad y = \phi(s) = 1/(1 + \exp(-s))$$

Parameters of the mappings – weights of the weighted sums. Can be learned by gradient descent to minimise the reconstruction error.


Input variables (coordinates of x^i)
First layer outputs
Low-dimensional coordinates
Third layer outputs
Output variables (reconstructed coordinates of x^i)

Locally Linear Embedding

LLE (Sam Roweis & Lawrence Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding”, *Science*, 2000): a method that does not aim at coordinate reconstruction, but uses coordinate reconstruction as part of the method.

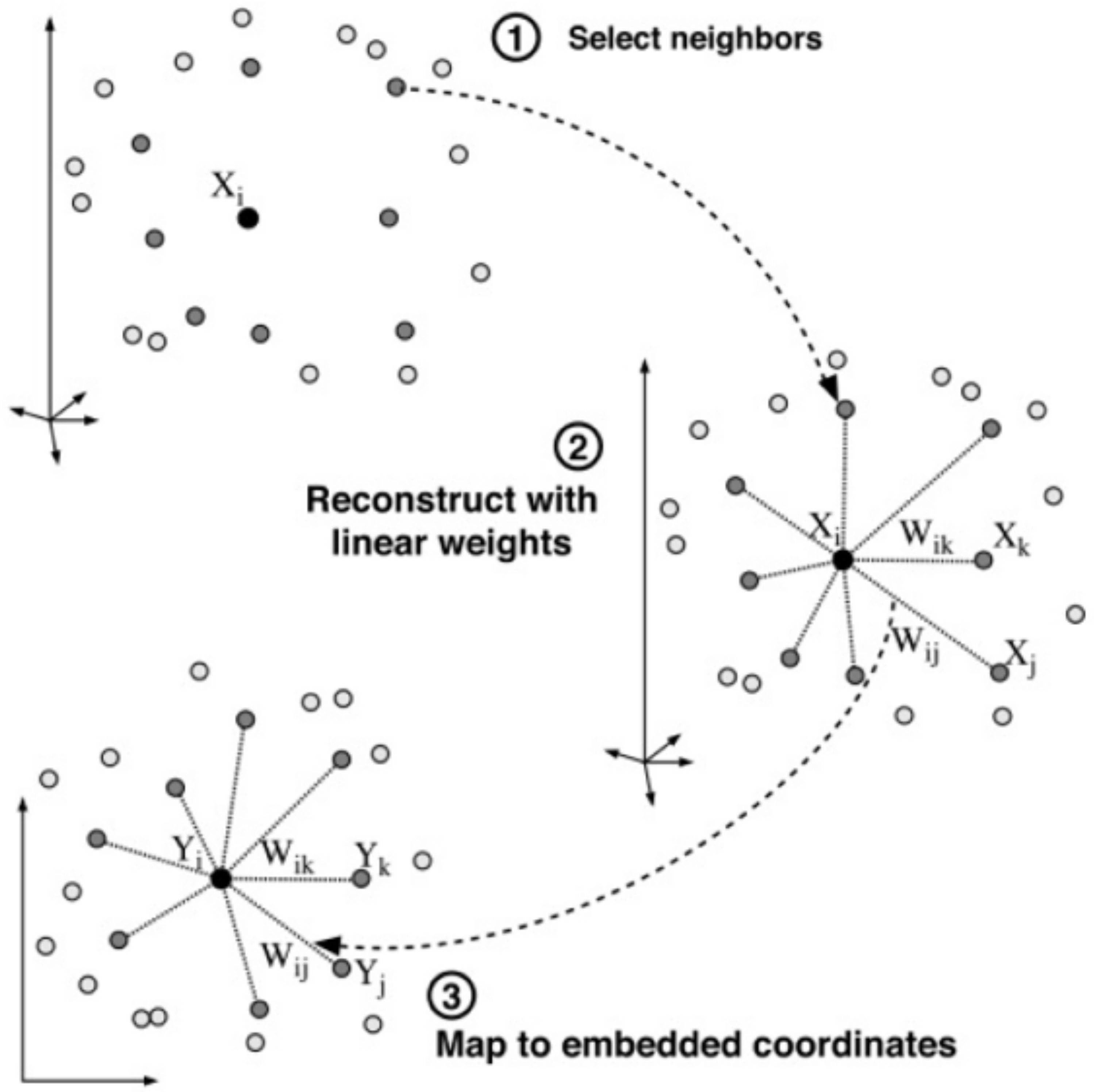
Idea: if the manifold of the data is *locally linear*, then each data point lies on the linear subspace spanned by its neighbors (or at least close to the subspace).

If that is true, then the position of each point can be reconstructed as a linear combination of the positions of its neighbors.

Try to preserve the same linear combination in the lower-dimensional space!

Locally Linear Embedding

LLE (Roweis&Saul 2000): preservation of local topologies by reconstruction of data points by linear combination of its neighbors



Locally Linear Embedding

LLE (Roweis&Saul 2000): preservation of local topologies by reconstruction of data points by linear combination of its neighbors

Let $\mathcal{N}(\mathbf{x})$ be the neighbors of **x**, find reconstruction weights by

$$\mathrm{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}) = \arg\min_{\mathbf{w}_i} \{(\mathbf{x} - \sum_{\mathbf{x}^j \in \mathcal{N}(\mathbf{x})} w_{ij}\mathbf{x}^j)^2 | \sum_i w_{ij} = 1\}$$

the constraint ensures rotation and translation invariance

Preserve local linear relationships by minimizing mean squared error:

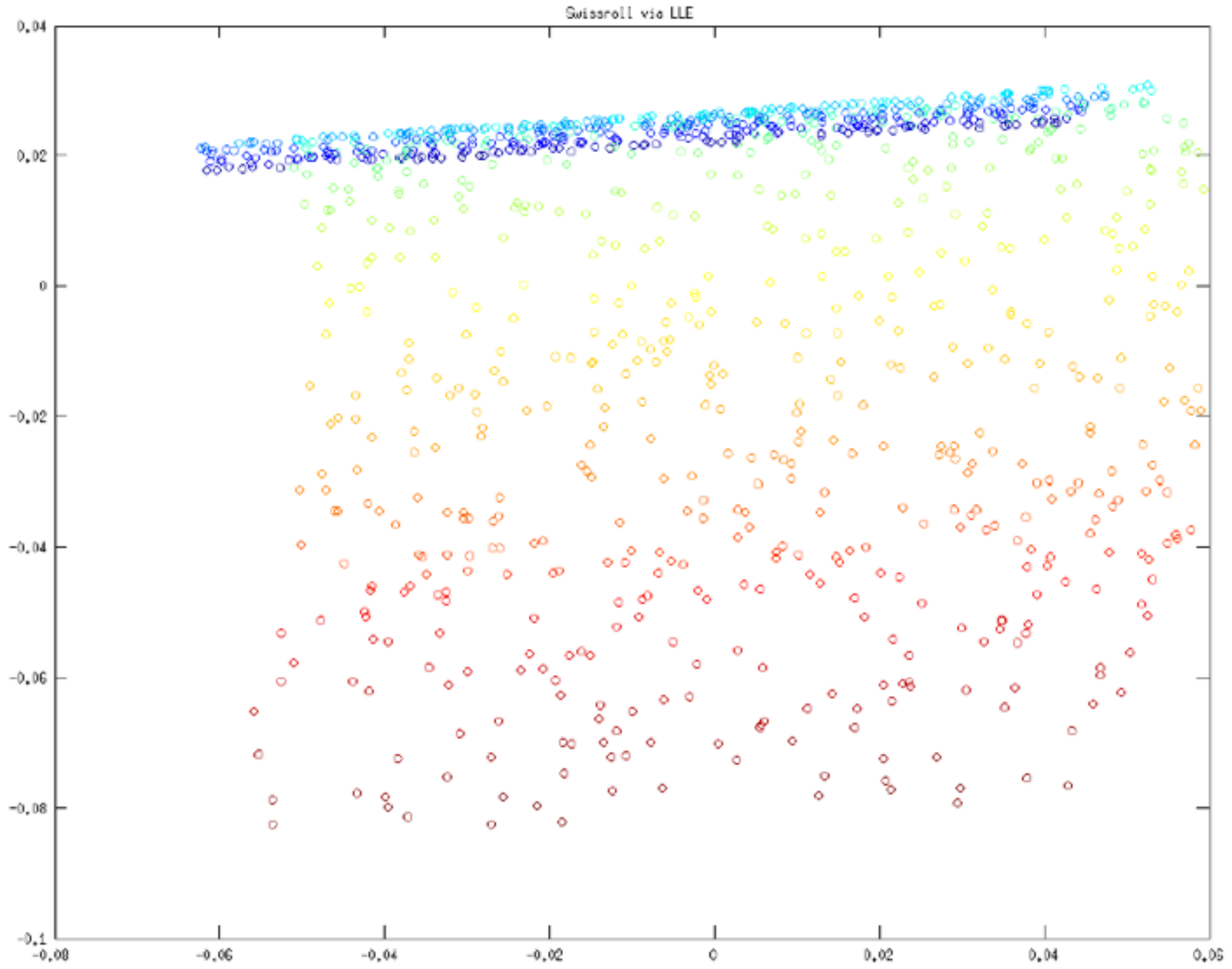
$$\mathrm{char}_{\mathcal{E}}(\mathbf{X\Xi}, (\mathbf{x}, \xi)) = (\xi - \sum_{\xi^j \in \mathcal{N}(\xi)} w_{ij}\xi^j)^2$$

The weights are the same fixed weights used in the original space.

Use constraint of centered coordinates $\sum_i \xi^i = 0$ and unit covariance $\mathbf{\Xi}^\top \mathbf{\Xi} = \mathbf{I}$, to avoid trivial optimum where all coordinates are zero. Normalization leads to a unique optimum found by solving a generalized eigenvalue problem.

Locally Linear Embedding

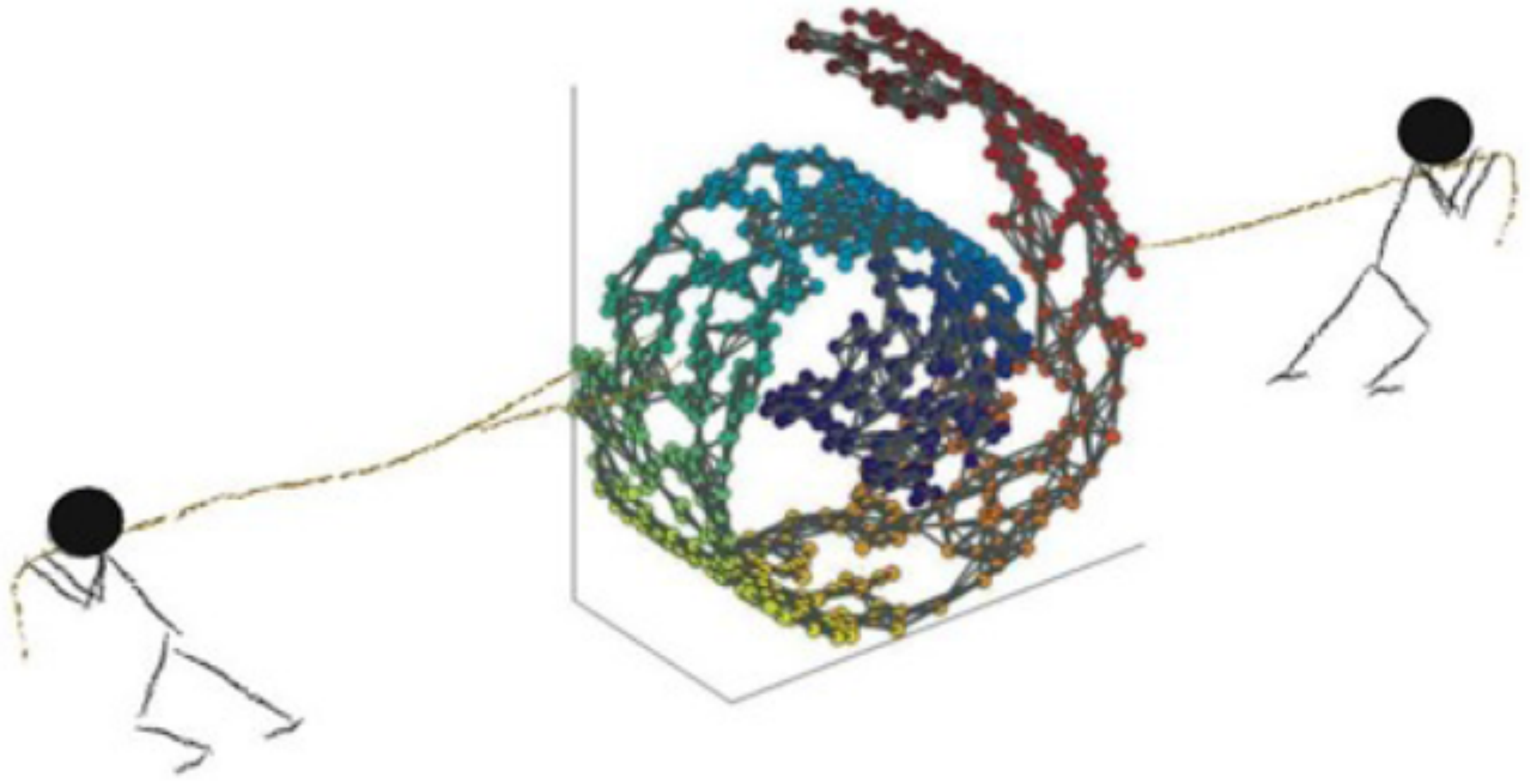
LLE (Roweis&Saul 2000): preservation of local topologies by reconstruction of data points by linear combination of its neighbors



Maximum Variance Unfolding (MVU)

Saul, L. K.; Weinberger, K. Q.; Ham, J. H.; Sha, F.; and Lee, D. D. 2006. Spectral methods for dimensionality reduction. In Semisupervised Learning. MIT Press.
Sun, J.; Boyd, S.; Xiao, L.; and Diaconis, P. 2006. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. To appear in SIAM Review.
Kilian Q. Weinberger and Lawrence K. Saul. An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In proceedings of AAAI 2006.

MVU is also based on a neighborhood graph. Projections are determined by maximizing the variance in the projection $\sum_{ij} d_{\mathcal{E}}(\xi^i, \xi^j)$, while preserving the distances of neighboring points



Maximum Variance Unfolding (MVU)

Saul, L. K.; Weinberger, K. Q.; Ham, J. H.; Sha, F.; and Lee, D. D. 2006. Spectral methods for dimensionality reduction. In Semisupervised Learning. MIT Press.
Sun, J.; Boyd, S.; Xiao, L.; and Diaconis, P. 2006. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. To appear in SIAM Review.
Kilian Q. Weinberger and Lawrence K. Saul. An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In proceedings of AAAI 2006.

MVU is also based on a neighborhood graph. Projections are determined by maximizing the variance in the projection $\sum_{ij} d_{\mathcal{E}}(\xi^i, \xi^j)$, while preserving the distances of neighboring points

Maximize $\sum_{ij} \|\vec{y}_i - \vec{y}_j\|^2$ **subject to:**
(1) $\|\vec{y}_i - \vec{y}_j\|^2 = \|\vec{x}_i - \vec{x}_j\|^2$ **for all** (i, j) **with** $\eta_{ij} = 1$.
(2) $\sum_i \vec{y}_i = 0$

From: Kilian Q. Weinberger and Lawrence K. Saul. An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In proceedings of AAAI 2006.

Maximum Variance Unfolding

MVU (Weinberger&Saul 2006) is also based on a neighborhood graph. Projections are determined by maximizing the variance in the projection $\sum_{ij} d_{\mathcal{E}}(\xi^i, \xi^j)$ while preserving the distances of neighboring points

$$\mathrm{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}) = (1_{\mathbf{x}^1 \in \mathcal{N}(\mathbf{x})}(\mathbf{x} - \mathbf{x}^1)^2, \ldots, 1_{\mathbf{x}^n \in \mathcal{N}(\mathbf{x})}(\mathbf{x} - \mathbf{x}^n)^2)$$

$$\mathrm{char}_{\mathcal{E}}(\mathbf{X\Xi}, (\mathbf{x}, \xi)) = (1_{\mathbf{x}^1 \in \mathcal{N}(\mathbf{x})}(\xi - \xi^1)^2, \ldots, 1_{\mathbf{x}^n \in \mathcal{N}(\mathbf{x})}(\xi - \xi^n)^2)$$

with the constraint $d_{\mathcal{E}}(\xi^i, \xi^j) = d_{\mathcal{X}}(\mathbf{x}^i, \mathbf{x}^j)$ if $\mathbf{x}^j \in \mathcal{N}(\mathbf{x}^i)$ and centered $\mathbf{\Xi}$ optimization can be written based on the Gram matrix $K_{ij} = \xi^i \xi^j$

And the solution can be found by a semidefinite program (SDP) $\max(\mathrm{Tr}(K))$ subject to $K \succeq 0$, the projections are given by eigenvalue decomposition similar to MDS

Maximum Variance Unfolding

And the solution can be found by a semidefinite program (SDP) $\max(\mathrm{Tr}(K))$ subject to $K \succeq 0$, the projections are given by eigenvalue decomposition similar to MDS

$$K_{ij} = \vec{y}_i \cdot \vec{y}_j$$

Maximize trace(K) subject to:

(1) $K_{ii} - 2K_{ij} + K_{jj} = \|\vec{x}_i - \vec{x}_j\|^2$ **for all** (i, j) **with** $\eta_{ij} = 1$.

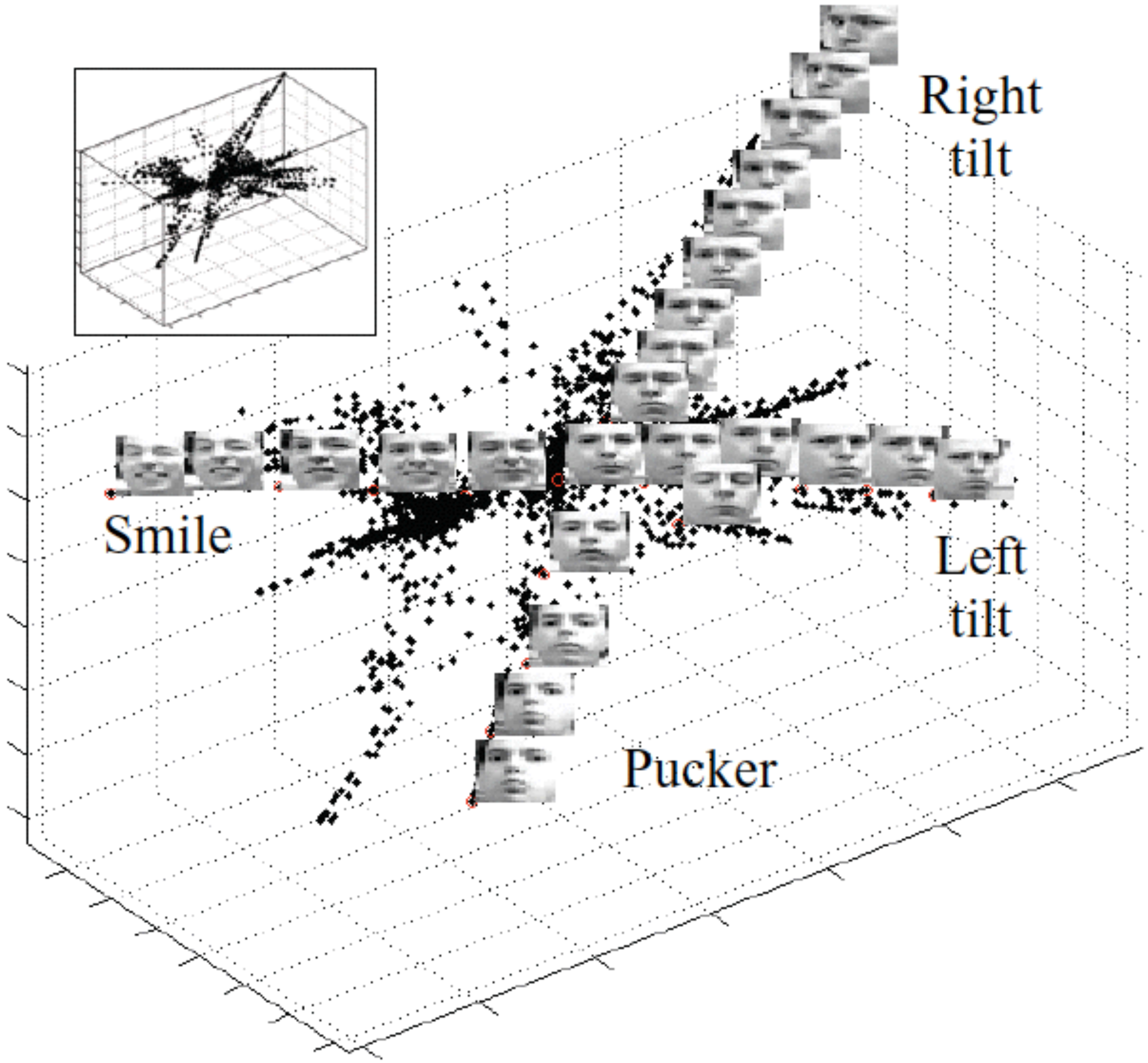
(2) $\Sigma_{ij} K_{ij} = 0$.

(3) $K \succeq 0$.

From: Kilian Q. Weinberger and Lawrence K. Saul. An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In proceedings of AAAI 2006.

Maximum Variance Unfolding

MVU on a data set of facial poses and expressions of one person



From: Kilian Q. Weinberger and Lawrence K. Saul. An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In proceedings of AAAI 2006.