MTTS1

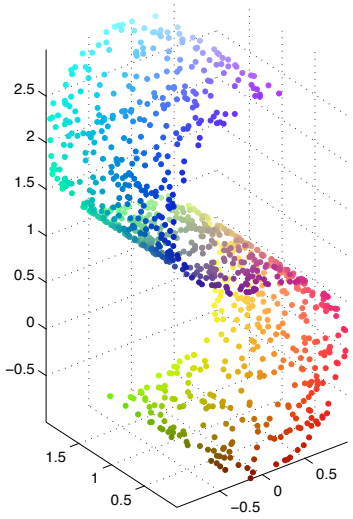
Dimensionality Reduction and Visualization

Spring 2014

Jaakko Peltonen

Dimension reduction methods

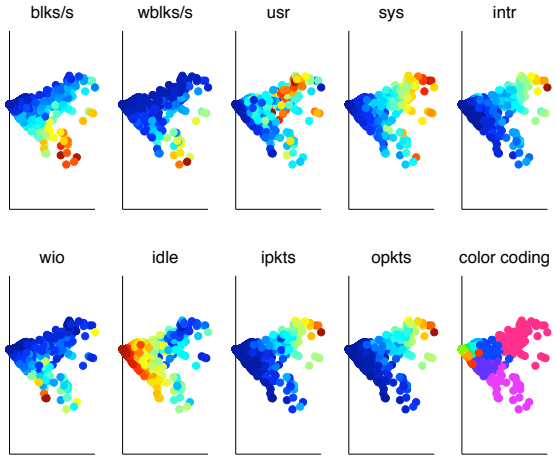
- What to do if... I have to analyze multidimensional data???
- Solution: Dimensionality Reduction from 17 dimensions to 1D, 2D or 3D
- Problem: How to project the nodes *faithfully* into low-dimensional space (1D-3D)?



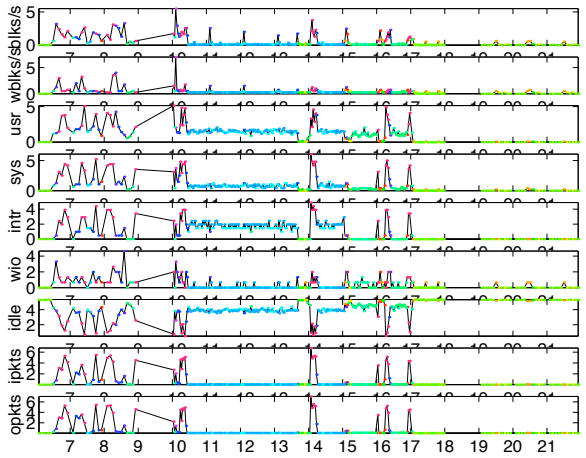
Original 3D data

2D projection by Curvilinear Component Analysis

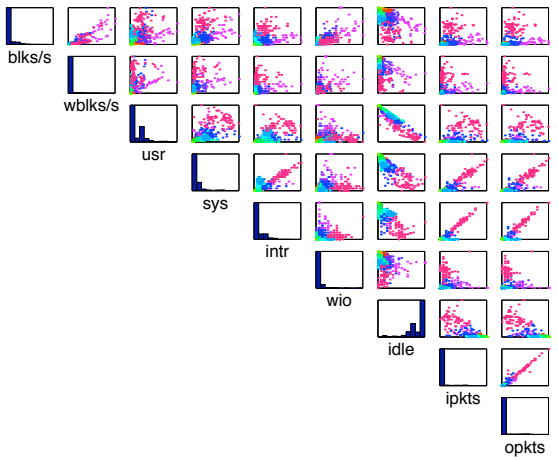
Visualization of a data set



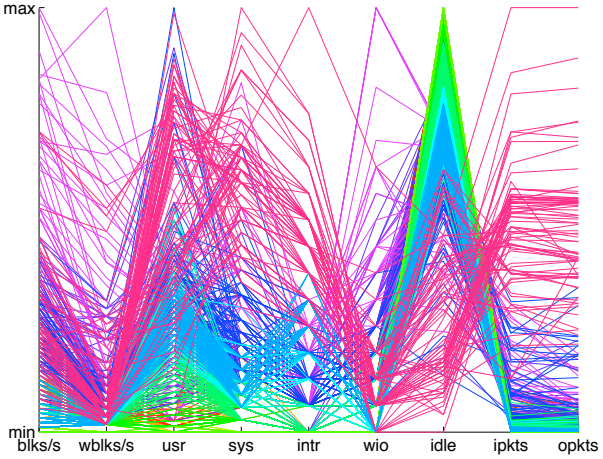
(a) Component planes



(b) Time-series



(c) Scatterplot matrix



(d) Parallel axis

From Juha Vesanto's doctoral thesis (2002), page 38.

Dimensionality Reduction

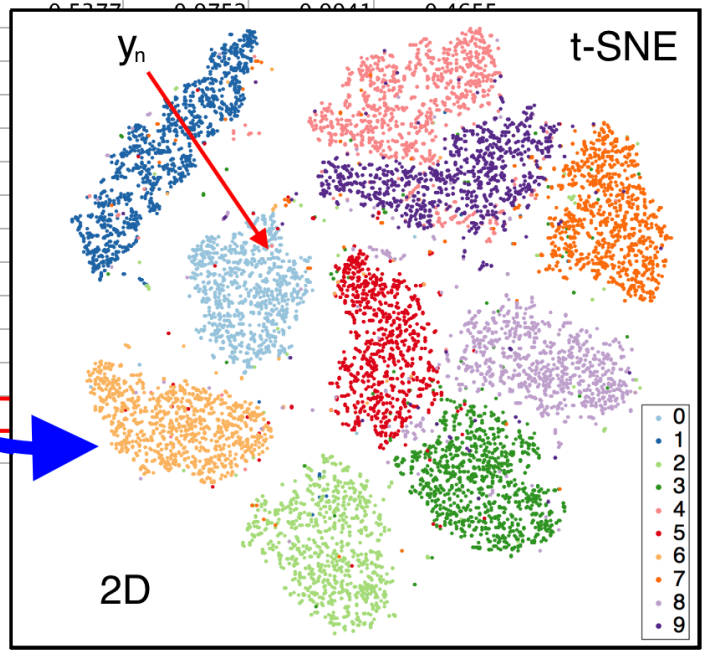
MNIST

	1	2	3	4	5	6	7	8	9	10
1	-0.3719	-1.5272	0.5759	-0.5991	0.9791	-0.5112	-0.2705	0.2778	0.3959	-1.1959
2	1.1788	0.6853	-0.4852	-0.3301	-0.1806	-0.7734	-0.4423	-0.3600	-0.6339	-0.0988
3	-1.8661	0.6259	1.2910	-1.2056	-1.7147	-1.3259	0.5882	0.4728	-0.1525	-0.1700
4	-0.5520	-1.5008	0.5909	0.2450	-0.7648	-0.6206	-0.6121	-1.1448	-0.0695	-0.6950
5	-0.5772	-2.0357	0.1287	1.5396	-0.3895	0.3938				
6	-0.7983	0.2106	0.8516	0.1278	1.1448	-0.8832				
7	-2.7710	0.9666	0.2987	0.3813	-0.0296	0.3866				
8	0.6923	-1.3235	1.0212	-0.2151	0.0815	-0.1155				
9	-3.6301	0.7912	-1.0352	-0.3903	-0.7005	0.1987				
10	0.6871	0.3195	0.3887	-1.8636	0.0575	0.0401				
11	-1.1250	-0.4799	-1.3353	-0.4977	2.2862	0.7745				
12	-1.4807	-0.1793	-2.1890	-0.3219	0.2592	-0.6512				
13	-1.0811	-1.8521	1.3248	-0.3865	-1.2451	0.1368				
14	-0.8135	1.6910	0.5258	-0.7722	0.1577	1.0025				
15	0.7151	0.2077	-0.8843	0.3581	-0.4365	0.5765				
16	0.7337	-1.1998	-0.0254	0.3646	0.1651	-0.1967				
17	0.7133	-0.9934	-0.6849	1.3934	0.1603	-0.0562				
18	-0.3193	1.2468	-0.9969	1.5263	0.6650	-0.8921				
19	0.3098	-0.5230	-0.7743	-1.5109	0.3518	0.7477				

196 features

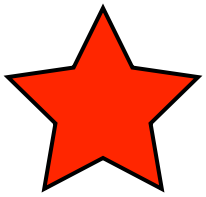
x_n

N points



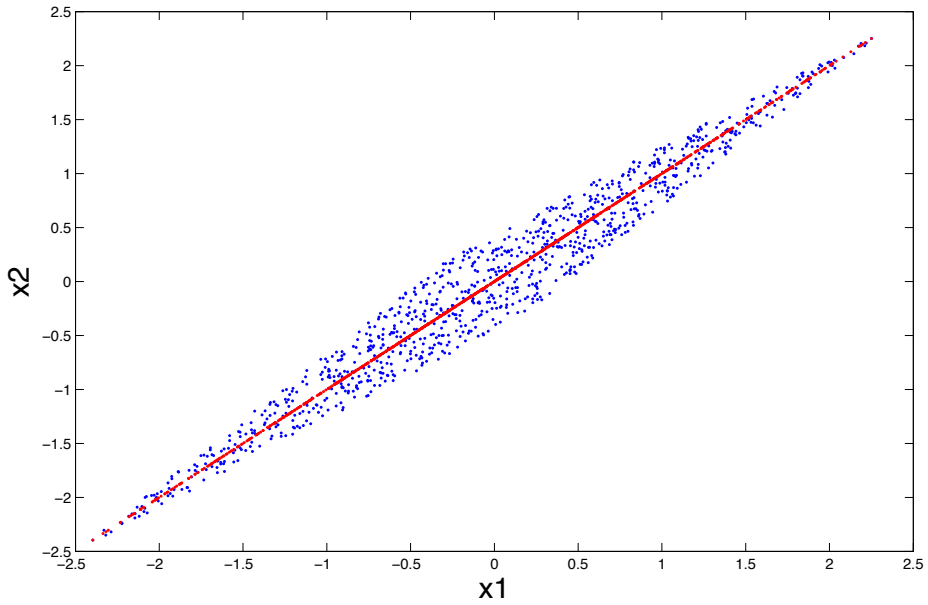
Scatter-plot of a high D data $\rightarrow$ 2D or 3D
t-SNE is the most commonly used method

Principal component analysis (PCA)



- PCA is stable, there are no additional parameters, and it is guaranteed always to converge to the same optima.
- Hence, PCA is usually the first dimension reduction method to try (if it doesn't work, then try something more fancy)

Principal component analysis (PCA)



Covariance matrix (after subtracting mean from data)

$$C = X^T X = \begin{bmatrix} 999.00 & 979.84 \\ 979.84 & 999.00 \end{bmatrix}$$

Eigenvector matrix (projection to principal components)

$$V = \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

Eigenvalues (variance along principal components)

$$[\lambda_1 \lambda_2] = \begin{bmatrix} 1978.8 & 19.2 \end{bmatrix}$$

Proportion of variance explained

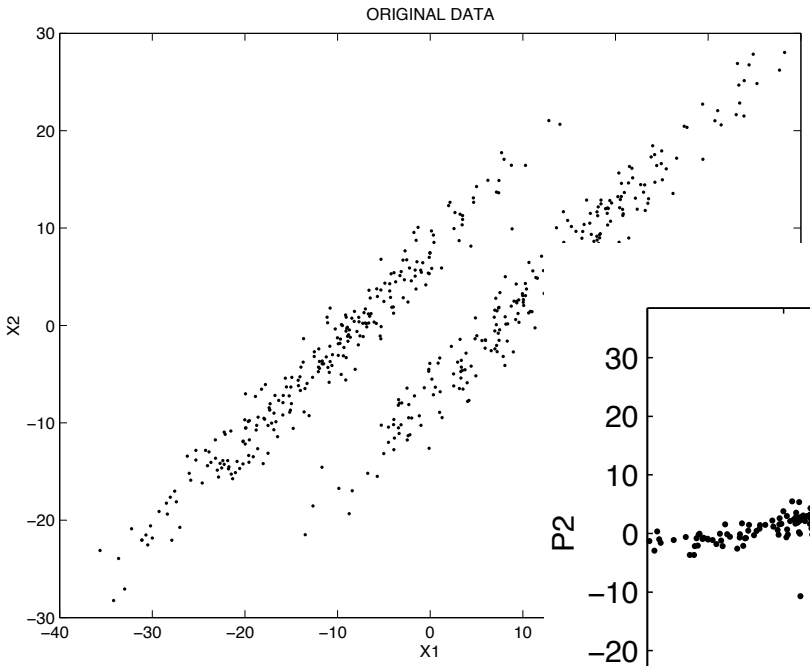
$$TVE = \frac{\lambda_1}{\sum_{i=1}^{d} \lambda_i} = \begin{bmatrix} 0.9904 \end{bmatrix}$$

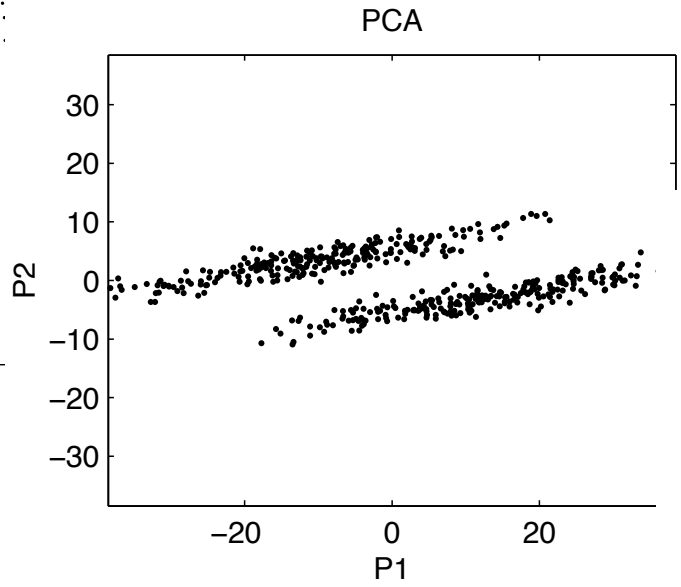
Reconstruction error

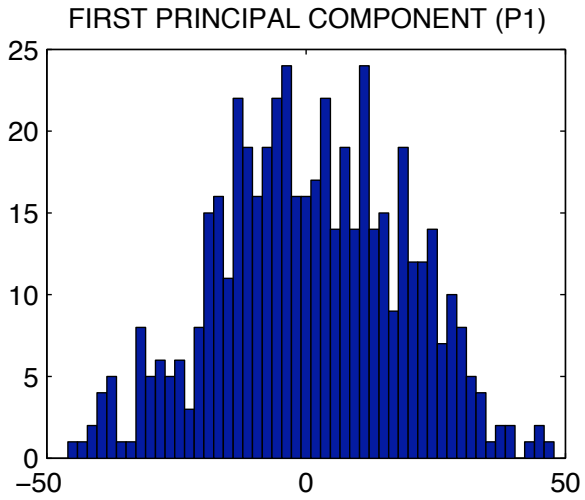
$$E = \frac{\sum_{i=1}^{d} var[(\hat{X}^i - X^i)]}{d} = \begin{bmatrix} 0.0096 \end{bmatrix}$$

Two clusters

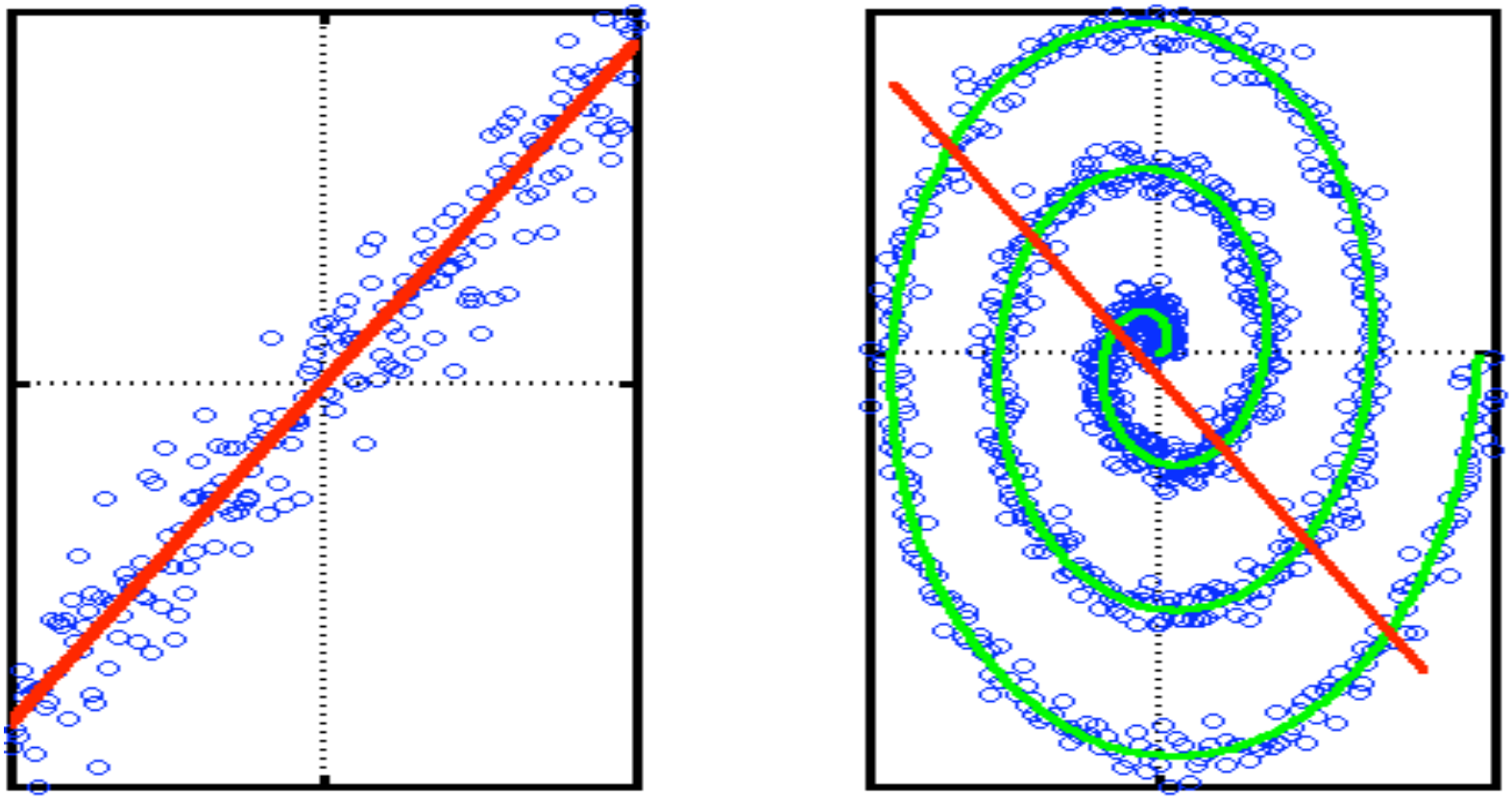
- The PCA fails to separate the clusters (you don't see cluster structure from the 1D visualization, lower right)





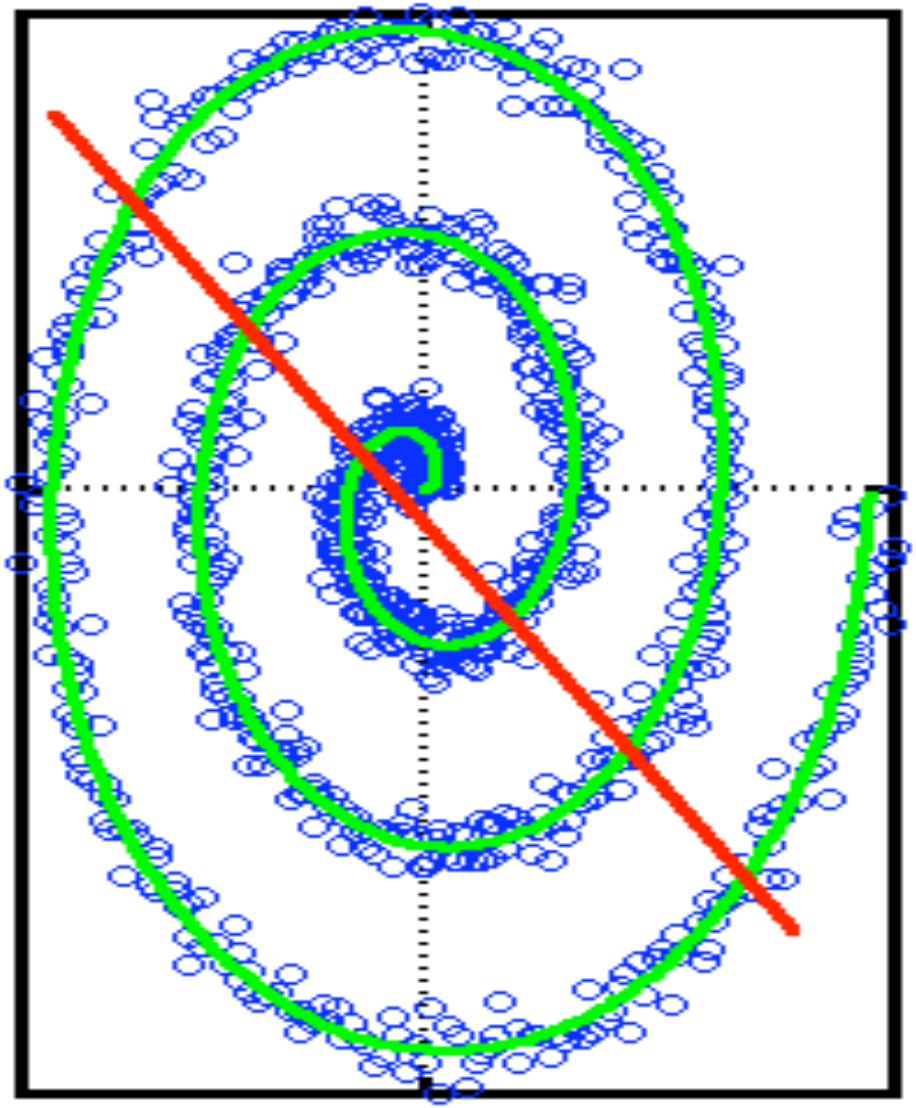


Nonlinear data



The first principal component is given by the red line. The green line on the right gives the "correct" non-linear dimension (which PCA is of course unable to find).

Manifolds



- Left, PCA mapping would not find the "correct" 1D manifold, shown in green, because they try to preserve global features.
- Often, preserving local features, like trustworthiness, continuity or conformity - or manifold structure - is more important than global properties.

Outlook

- (un-)supervised Dimensionality reduction and visualization
- Generalized framework for dimensionality reduction
- Quality assessment

Nonlinear dimensionality reduction methods

Introduction

- Represent high-dimensional data by low-dimensional counterparts preserving as much information as possible
- Ill-posed problem: which data is relevant to the user? (dependent on the specific data domain and situation at hand)
- Huge variety of methods proposed with different properties

Dimension reduction methods

- There are several methods, with different optimization goals and complexities
- We will go through some of them (most not in any detail, last ones not at all):
  - **Principal component analysis (PCA)** - "simple" linear method that tries to preserve global distances
  - **Multidimensional scaling (MDS)** - tries to preserve global distances
  - *Sammon's projection* - a variation of the MDS, pays more attention to short distances
  - *Isometric mapping of data manifolds (ISOMAP)* - a graph-based method (of the MDS spirit)
  - *Curvilinear component analysis (CCA)* - MDS-like method that tries to preserve distances in small neighborhoods
  - *Maximum variance unfolding* - maximizes variance with the constraint that the short distances are preserved (an exercise in semidefinite programming)
  - *Self-organizing map (SOM)* - a flexible and scalable method that tries a surface that passes through all data points (originally developed at HUT)
  - *Independent component analysis (ICA)* - a fast linear method, suitable for some applications
  - Nerv (NEighbor Retrieval Visualizer, originally developed at HUT)

Different methods, different properties

- **Spectral techniques:** the rely on the spectrum of the neighborhood graph of the data, preserving important properties of it.
  - Example methods: Locally Linear Embedding (LLE), Isomap, Laplacian Eigenmaps
  - usually unique algebraic solution of the objective
  - in order to make the cost functions unimodal and to make algebraic solution of objective possible, the methods are based on very simple affinity functions

Different methods, different properties

- **Non-parametric methods:** they usually do not find a general mapping function from a high-dimensional space to a lower-dimensional space, instead they find a mapping a finite data set

- They can use more complicated affinities between data points, but it comes with higher computational costs

- Additional modeling/optimization and computational effort must be done for *out-of-sample extension* (for mapping new data points that were not int he training set)

Different methods, different properties

- **Explicit mapping functions:** some methods explicitly learn (infer) a (non-)linear mapping function
  - linear functions: Principal Component Analysis, Linear Discriminant Analysis
  - nonlinear functions: autoencoder networks, locally linear coordination

Different methods, different properties

- **Supervised techniques:** use ”ground truth” information provided by a teacher (oracle) to learn the mapping or mapping function
  - Linear Discriminant Analysis, Partial Least Squares regression, adaptive metrics
  - non-linear extensions by kernels

Dimensionality Reduction Setting

Assume we have

- a high-dimensional data space $\mathcal{X}$
- a data set from the data space: $\mathbf{x}^i \in \mathbb{R}^N,\ i = 1 \ldots n$

We want to find

- an output space (embedding space) $\mathcal{E}$
- low-dimensional representatives $\boldsymbol{\xi}^i \in \mathbb{R}^M$ for the data set in the embedding space

General aim of dimensionality reduction: find a **mapping**

$$f : \mathbb{R}^N \to \mathbb{R}^M$$

such that the **interesting properties** of the data distribution in $\mathcal{X}$ are **preserved** as well as possible also in $\mathcal{E}$

Multidimensional scaling (MDS)

- *Multidimensional scaling (MDS)* is a dimension reduction method that tries to preserve a measure of similarity (or dissimilarity or distance) between pairs of data points
- MDS has roots in the field of psychology (one consequence: lots of conventional notation)
- MDS can be used as
  - an exploratory visualization technique to find the structure of the data; and
  - a tool to test hypothesis.

Color similarities

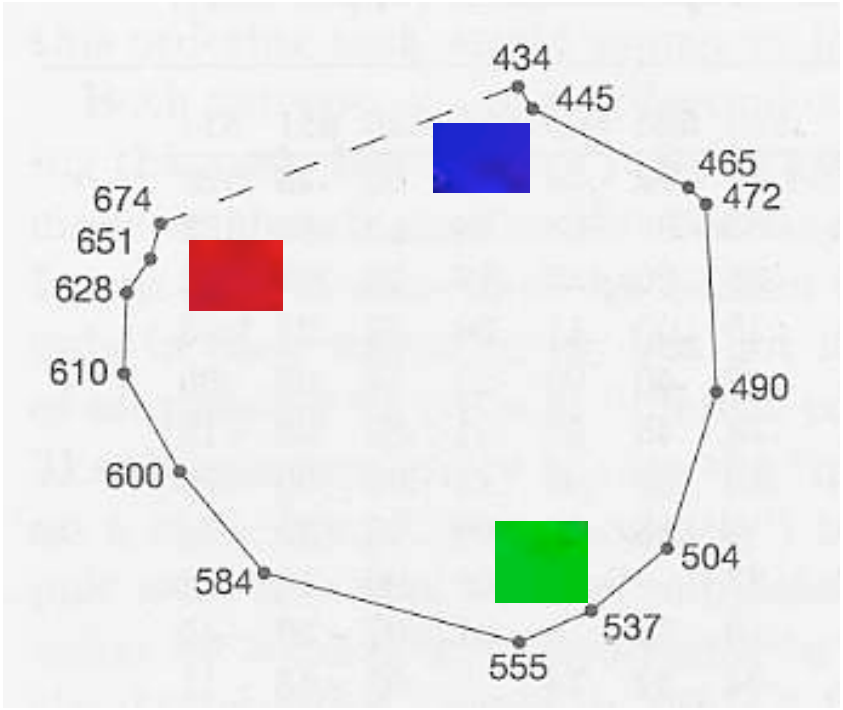
- Psychological test in 1950's: how is the similarity of colors perceived?
- Pairs of 14 colors were rated by 31 people. Ratings were averaged.

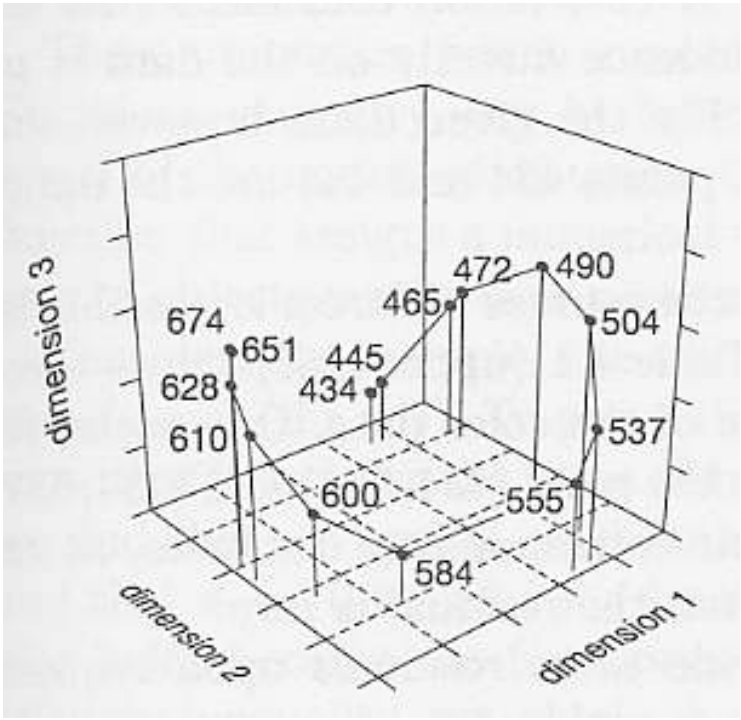
nm	434	445	465	472	490	504	537	555	584	600	610	628	651	674
434	–	.14	.17	.38	.22	-.73	-1.07	-1.21	-.62	-.06	.42	.38	.28	.26
445	.86	–	.25	.11	-.05	-.75	-1.09	-.68	-.35	-.04	.44	.65	.55	.53
465	.42	.50	–	.08	-.32	-.57	-.47	-.06	.00	-.32	.17	.12	.91	.82
472	.42	.44	.81	–	.12	-.36	-.26	.15	.00	-.11	.00	.33	.23	1.03
490	.18	.22	.47	.54	–	-.07	.08	.48	.40	.00	.22	.17	.07	.00
504	.06	.09	.17	.25	.61	–	.31	.28	.45	.68	.01	.00	.00	-.15
537	.07	.07	.10	.10	.31	.62	–	.13	.35	.09	.31	.00	.00	-.75
555	.04	.07	.08	.09	.26	.45	.73	–	-.05	.17	-.09	-.22	-.32	-.34
584	.02	.02	.02	.02	.07	.14	.22	.33	–	-.05	-.01	-.06	-.16	-.18
600	.07	.04	.01	.01	.02	.08	.14	.19	.58	–	.21	.07	-.39	-.40
610	.09	.07	.02	.00	.02	.02	.05	.04	.37	.74	–	-.08	-.13	-.11
628	.12	.11	.01	.01	.01	.02	.02	.03	.27	.50	.76	–	-.03	-.16
651	.13	.13	.05	.02	.02	.02	.02	.02	.20	.41	.62	.85	–	-.11
674	.16	.14	.03	.04	.00	.01	.00	.02	.23	.28	.55	.68	.76	–

Similarities of colors with different wavelengths (lower half, Ekman 1954) and residuals of 1D MDS representation (upper half) [B 4.1].

Color similarities

- The 14 colors where then projected by MDS (trying to preserve similarities) into 2D and 3D representations. The 2D representation shows that the red-violet (wavelength 434 nm) is perceived quite similar to blue-violet (wavelength 674 nm)





Ordinal MDS representations for color proximities in 2D and 3D [B 4.1, 4.3]

Multidimensional scaling (MDS)

- More formally, an MDS algorithm is given the original distances p_{ij} (called *proximities)* between data points *i* and *j*
- MDS algorithm then tries to find a low-dimensional (usually 2-3D) representation *X* for the points (*X* is just used to denote the Euclidean coordinates of the projected data points)
- More formally, MDS tries to find representation *X* that minimizes the error function (called *stress*, by convention)

$$\sigma_r = \sum_{i<j} \left(f(p_{ij}) - d_{ij}(X)\right)^2$$

where $d_{ij}(X)$ is the Euclidean distance between the data points *i* and *j* in representation *X*; and *f* is a function that defines the MDS model (next slide).

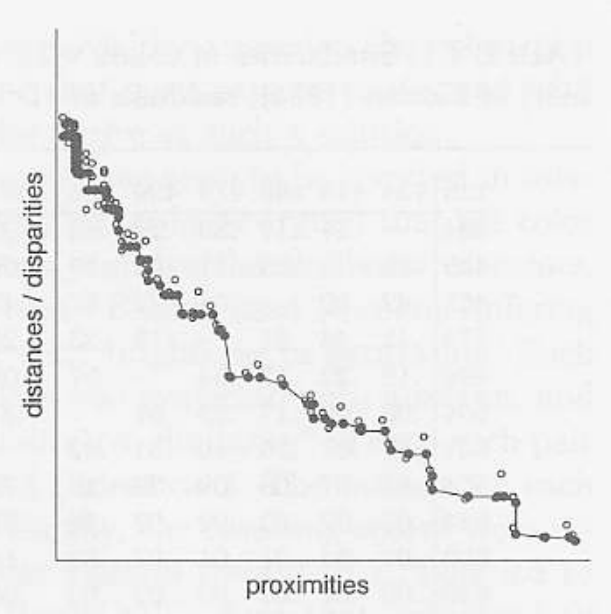
Multidimensional scaling (MDS)

$$\sigma_r = \sum_{i<j} \left(f(p_{ij}) - d_{ij}(X)\right)^2$$

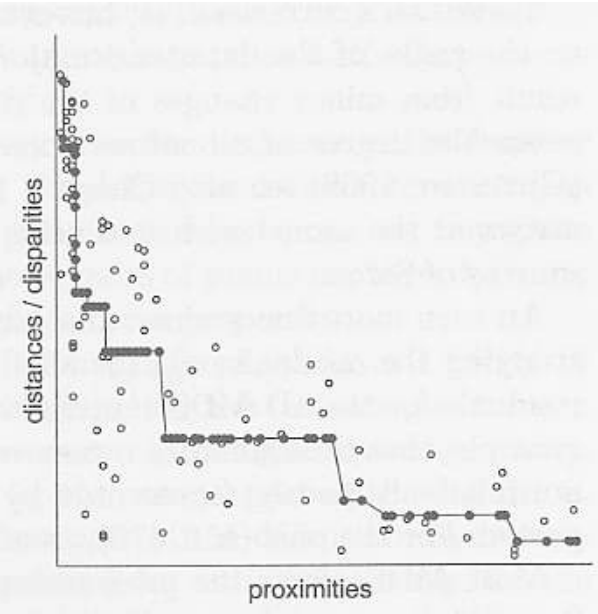
- The choice of *f* defines the MDS model. For example:
  - $f(p_{ij})=p_{ij}$ - absolute MDS (linear model, = PCA)
  - $f(p_{ij})=b\ p_{ij}$ - ratio MDS (linear model)
  - $f(p_{ij})=a+b\ p_{ij}$ - interval MDS (linear model)
  - $f(p_{ij})=a+b \log p_{ij}$ - useful in psychology
  - $f(p_{ij})$ is any monotonically increasing function (*ordinal* or *nonmetric MDS*) - this would be the most important special case of MDS
- The parameters of *f* (like *a* and *b* above) are optimized at the same time as the representation *X* (the details of the optimization algorithms is outside the scope of this course)
- It is conventional to denote the "transformed proximities", or "approximate distances", by d-hats, $\hat{d}_{ij} = f(p_{ij})$.

Shepard diagram

- There are two classical visualizations of MDS: *Shepard diagram* (shows the goodness of fit) and *Scree plot* (shows optimal dimensionality of the data)
- *Shepard diagram* shows the distances d_{ij} (white circles) and disparities $f(p_{ij})$ (filled circles) as a function of proximities p_{ij}.

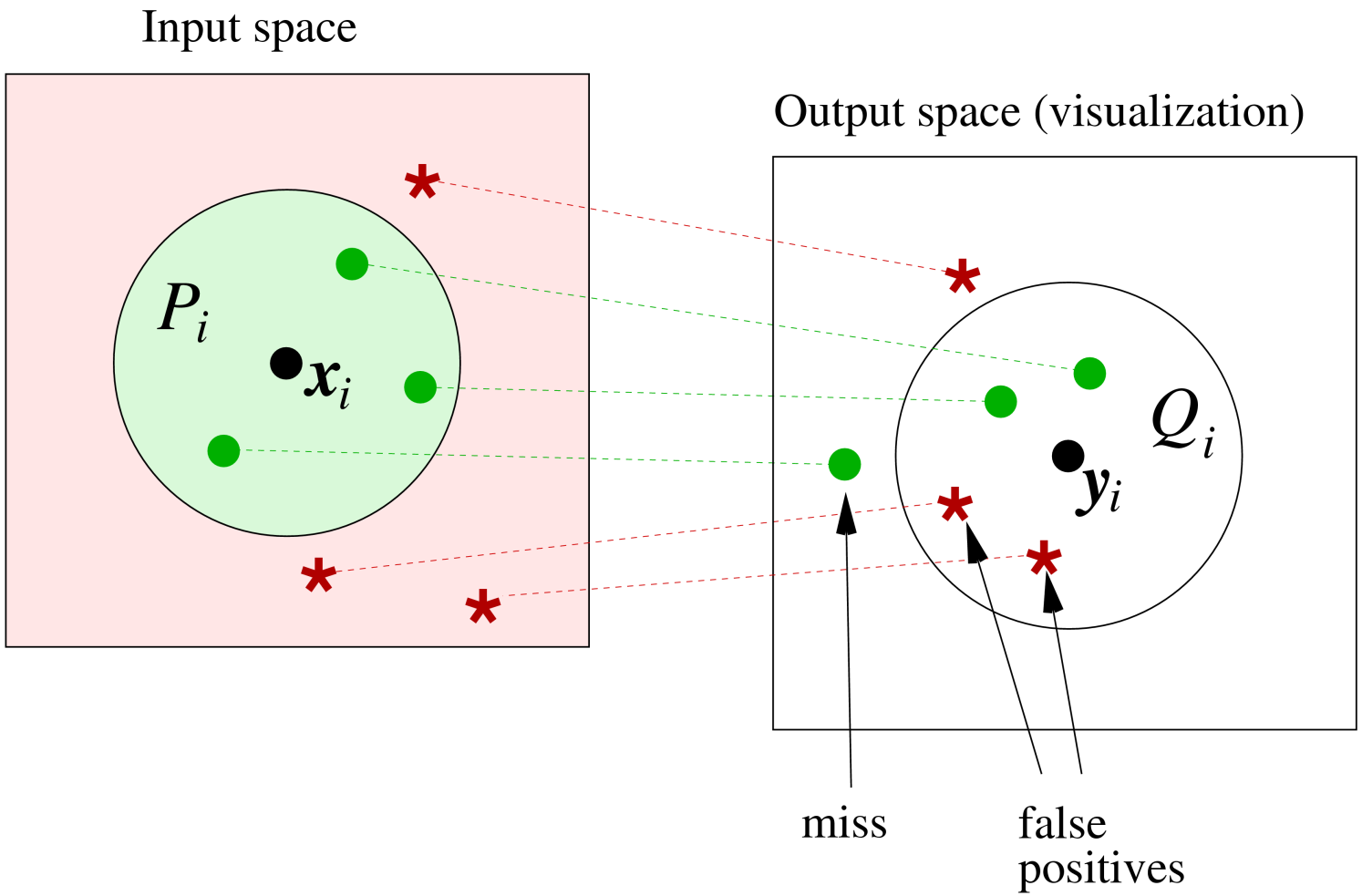


2D MDS



1D MDS

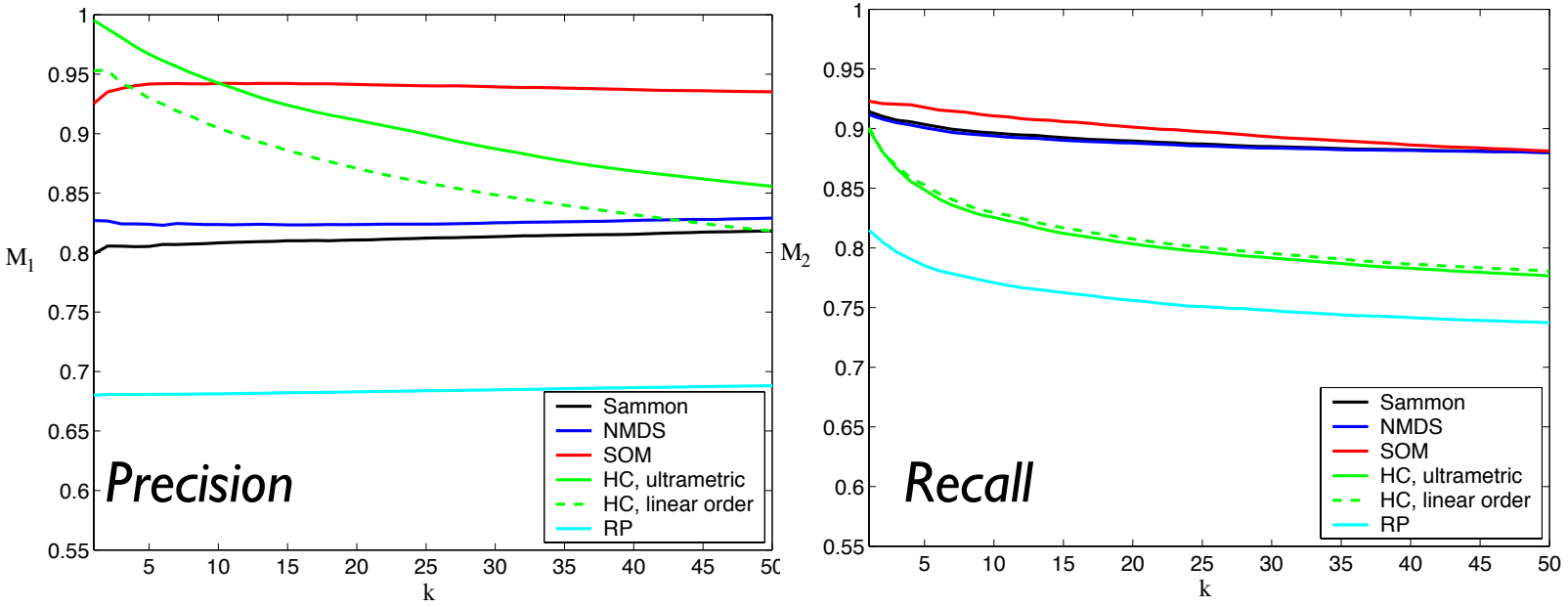
Shepard diagrams of 2D and 1D MDS projections of the color data.



Minimize errors for best information retrieval.

Performance of MDS

- MDS is tries to preserve the large distances at the expense of small ones, hence, it can "collapse" some small distances on the expense of preserving large distances
- A projection is *trustworthy (precision)* if *k* closest neighbors of a sample on the projection are also close by in the original space. A projection *preserves the original neighborhoods* (*recall*) if all *k* closest neighbors of a sample in the original space are also close by in the projection.



Figures are from Kaski, Nikkilä, Oja, Venna, Törönen, Castrén, *Trustworthiness and metrics in visualizing similarity of gene expression*, BMC Bioinformatics 2003, 4:48.

Precision and recall as a function of the neighborhood size *k* for a yeast data set. Non-metric (ordinal) MDS (NMDS) is shown in blue. Larger precision and recall is better.

Performance of MDS

- Relatively better recall, worse precision
- MDS algorithms typically have running times of the order $O(N^2)$, where N is the number of data items.
- This is not very good: $N=1,000$ data items are ok, but $N=1,000,000$ is getting very slow.
- Some solutions: use landmark points (i.e., use MDS only on a subset of data points and place the remaining points according to those, use MDS on cluster centroids etc.), use some other algorithm or modification of MDS.
- MDS is not guaranteed to find the global optimum of the stress (cost) function, nor it is guaranteed to converge to the same solution at each run (many of the MDS algorithms are quite good and reliable, though)

Performance of MDS

- Relatively better recall, worse precision
- MDS algorithms typically have running times of the order $O(N^2)$, where N is the number of data items.
- This is not very good: $N=1,000$ data items are ok, but $N=1,000,000$ is getting very slow.
- Some solutions: use landmark points (i.e., use MDS only on a subset of data points and place the remaining points according to those, use MDS on cluster centroids etc.), use some other algorithm or modification of MDS.
- MDS is not guaranteed to find the global optimum of the stress (cost) function, nor it is guaranteed to converge to the same solution at each run (many of the MDS algorithms are quite good and reliable, though)

Performance of MDS

- Relatively better recall, worse precision
- MDS algorithms typically have running times of the order $O(N^2)$, where N is the number of data items.
- This is not very good: $N=1,000$ data items are ok, but $N=1,000,000$ is getting very slow.
- Some solutions: use landmark points (i.e., use MDS only on a subset of data points and place the remaining points according to those, use MDS on cluster centroids etc.), use some other algorithm or modification of MDS.
- MDS is not guaranteed to find the global optimum of the stress (cost) function, nor it is guaranteed to converge to the same solution at each run (many of the MDS algorithms are quite good and reliable, though)

Classical Metric Multidimensional Scaling

First major steps by Young and Householder (1938):
Classical metric MDS is defined as linear generative model

$$\mathbf{x} = W\xi \text{ with } W \in \mathbb{R}^{N\times M} \text{ and } W^\top W = \mathbf{I}_M$$

where the observed data **X** and the latent variables are assumed to be centered.
Pairwise affinities given by scalar products $S_{ij} = \langle \mathbf{x}^i, \mathbf{x}^j \rangle$

Gram matrix:
$$S = [S_{ij}]_{1\le i,j\le n} = \mathbf{X}^\top \mathbf{X} = (W\mathbf{\Xi})^\top (W\mathbf{\Xi})$$
$$= \mathbf{\Xi}^\top W^\top W \mathbf{\Xi} = \mathbf{\Xi}^\top \mathbf{\Xi}$$

Find solution by eigenvalue decomposition of Gram matrix S:

$$S = U\Lambda U^\top = (\Lambda^{1/2}U^\top)^\top (\Lambda^{1/2}U^\top)$$

U = n x n orthonormal matrix
Λ= diagonal eigenvalue matrix

$\rightarrow \hat{\mathbf{\Xi}} = \mathbf{I}_{M\times n}\Lambda^{1/2}U^\top$ eigenvalues sorted in descending order

Classical Metric MDS and PCA

- PCA needs data coordinates **X**, not needed in metric MDS
- PCA decomposes covariance, which is proportional to $\mathbf{X}\mathbf{X}^\top$

$$\widehat{C_{XX}} \propto \mathbf{X}\mathbf{X}^\top = V\Lambda_{\mathrm{PCA}}V^\top \rightarrow \hat{\Xi}_{\mathrm{PCA}} = \mathbf{I}_{M\times n}V^\top\mathbf{X}$$

- metric MDS decomposes Gram matrix

$$S = \mathbf{X}^\top\mathbf{X} = U\Lambda_{\mathrm{MDS}}U^\top \rightarrow \hat{\Xi}_{\mathrm{MDS}} = \mathbf{I}_{M\times n}\Lambda_{\mathrm{MDS}}^{1/2}U^\top$$

- It is easy to show that

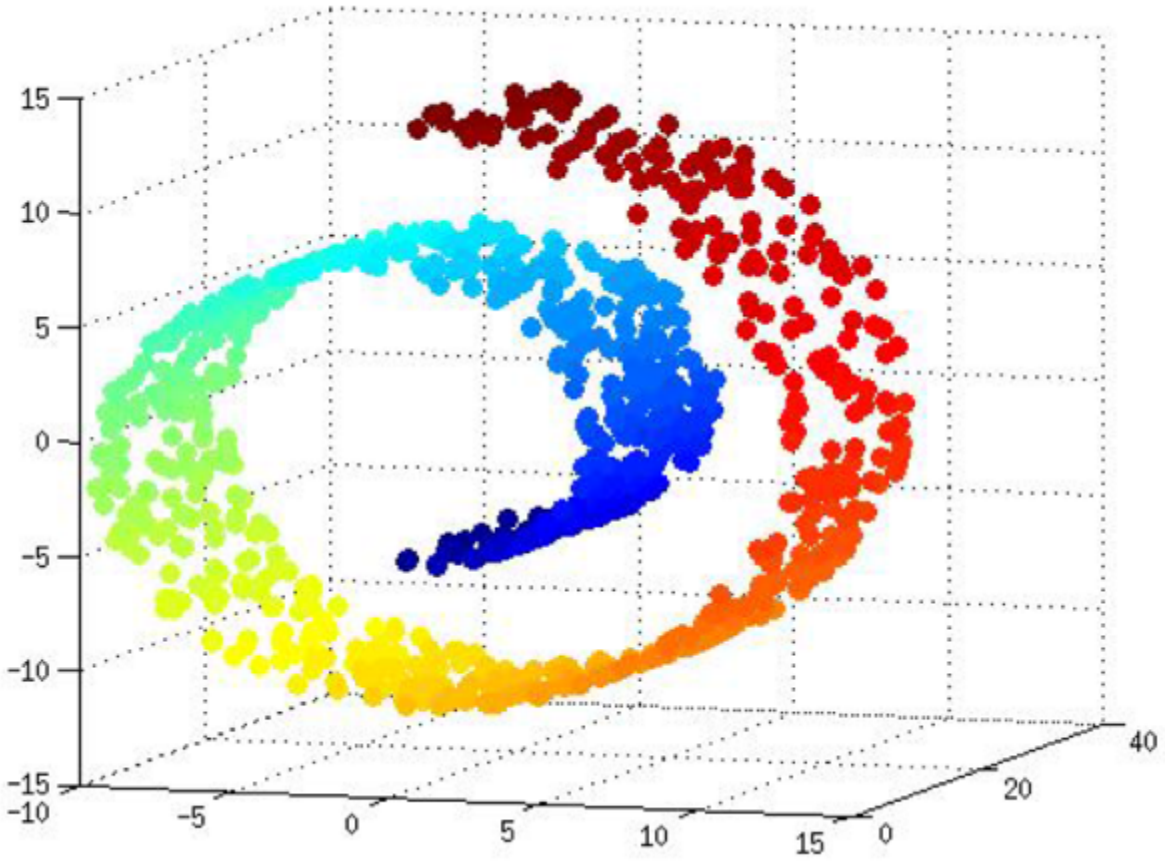
$$\hat{\Xi}_{\mathrm{PCA}} = \hat{\Xi}_{\mathrm{MDS}}$$
$$\mathbf{I}_{M\times n}V^\top\mathbf{X} = \mathbf{I}_{M\times n}\Lambda_{\mathrm{MDS}}^{1/2}U^\top$$

- To do so, replace **X** by its singular value decomposition

$$\mathbf{X} = V\Sigma U^\top$$

Example Data

swiss roll: 1000 three-dimensional samples



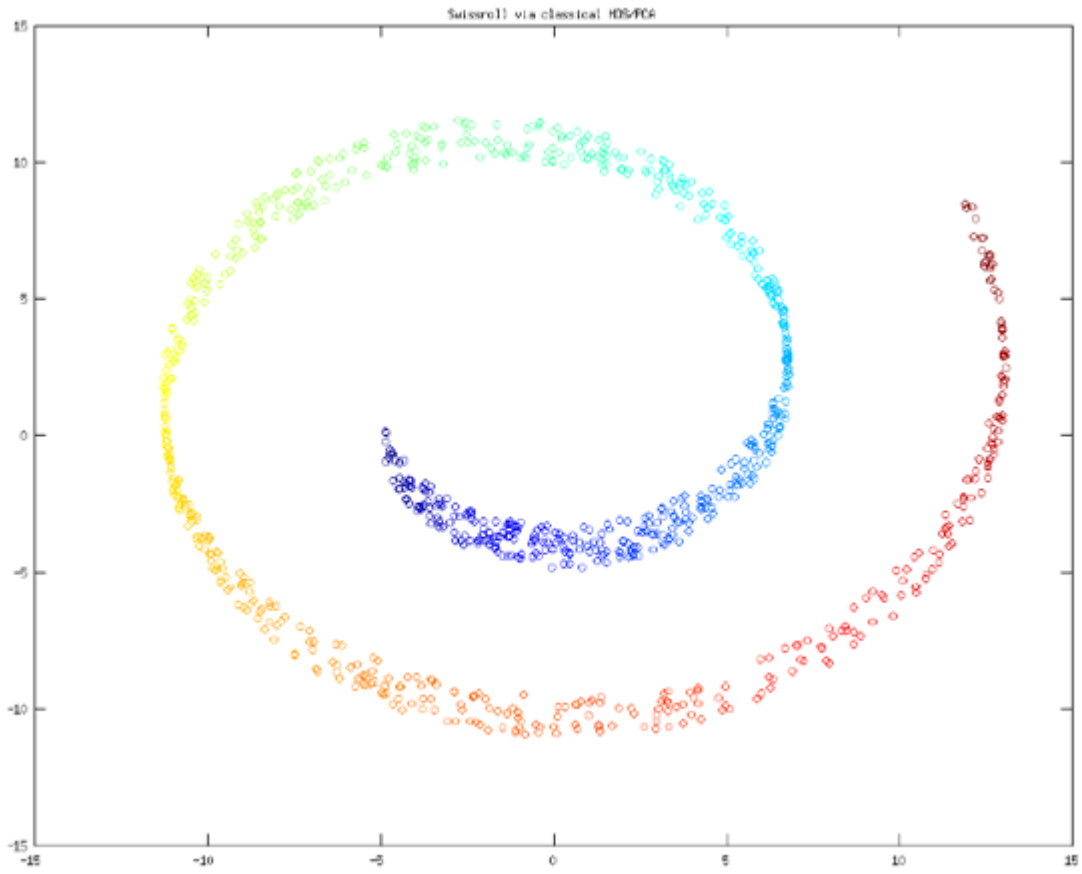
MDS

Pairwise Euclidean distances:

$$\mathrm{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}) = (d_{\mathcal{X}}(\mathbf{x}^1, \mathbf{x}), \ldots, d_{\mathcal{X}}(\mathbf{x}^n, \mathbf{x}))$$

$$\mathrm{char}_{\mathcal{E}}(\mathbf{X}\boldsymbol{\Xi}, (\mathbf{x}, \boldsymbol{\xi})) = (d_{\mathcal{E}}(\boldsymbol{\xi}^1, \boldsymbol{\xi}), \ldots, d_{\mathcal{E}}(\boldsymbol{\xi}^n, \boldsymbol{\xi}))$$

error = stress (least squared error (LSE))



Some MDS Variants

- **Nonmetric MDS** (Shepard 1962) and (Kruskal 1964) focuses on rank information ranks of closenesses between points instead of the specific interpoint distances:
- Proximities can be transformed to distances by $f(\delta(\mathbf{x}^i, \mathbf{x}^j)) \approx d_{\mathcal{X}}(\mathbf{x}^i, \mathbf{x}^j)$ with a monotonic transformation f

- Optimization done by minimizing

$$E_{\text{nMDS}} = \sqrt{\frac{1}{a}\sum_{ij} w_{ij}(d_{\mathcal{X}}(\mathbf{x}^i, \mathbf{x}^j) - d_{\mathcal{E}}(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j))^2}$$

where the normalizing constant is $a = \sum_{i,j}^{n} w_{ij} d_{\mathcal{X}}(\mathbf{x}^i, \mathbf{x}^j)$

Sammon Mapping

$$\sigma_r = \sum_{i<j} \frac{(p_{ij} - d_{ij}(X))^2}{p_{ij}}$$

- It is considered a non-linear approach as the projection cannot be represented as a linear combination of the original variables as possible in techniques such as principal component analysis.
- The minimization can be performed either by gradient descent. The number of iterations need to be experimentally determined and convergent solutions are not always guaranteed. Many implementations prefer to use the first Principal Components as a starting configuration.
- The Sammon mapping increases the importance of small distances and decreases the importance of large distances → nonlinear mapping

Sammon Mapping

$$\sigma_r = \sum_{i<j} \frac{(p_{ij} - d_{ij}(X))^2}{p_{ij}}$$

- It is considered a non-linear approach as the projection cannot be represented as a linear combination of the original variables as possible in techniques such as principal component analysis.
- The minimization can be performed either by gradient descent. The number of iterations need to be experimentally determined and convergent solutions are not always guaranteed. Many implementations prefer to use the first Principal Components as a starting configuration.
- The Sammon mapping increases the importance of small distances and decreases the importance of large distances → nonlinear mapping

Sammon Mapping

$$\sigma_r = \sum_{i<j} \frac{(p_{ij} - d_{ij}(X))^2}{p_{ij}}$$

- It is considered a non-linear approach as the projection cannot be represented as a linear combination of the original variables as possible in techniques such as principal component analysis.
- The minimization can be performed either by gradient descent. The number of iterations need to be experimentally determined and convergent solutions are not always guaranteed. Many implementations prefer to use the first Principal Components as a starting configuration.
- The Sammon mapping increases the importance of small distances and decreases the importance of large distances → nonlinear mapping

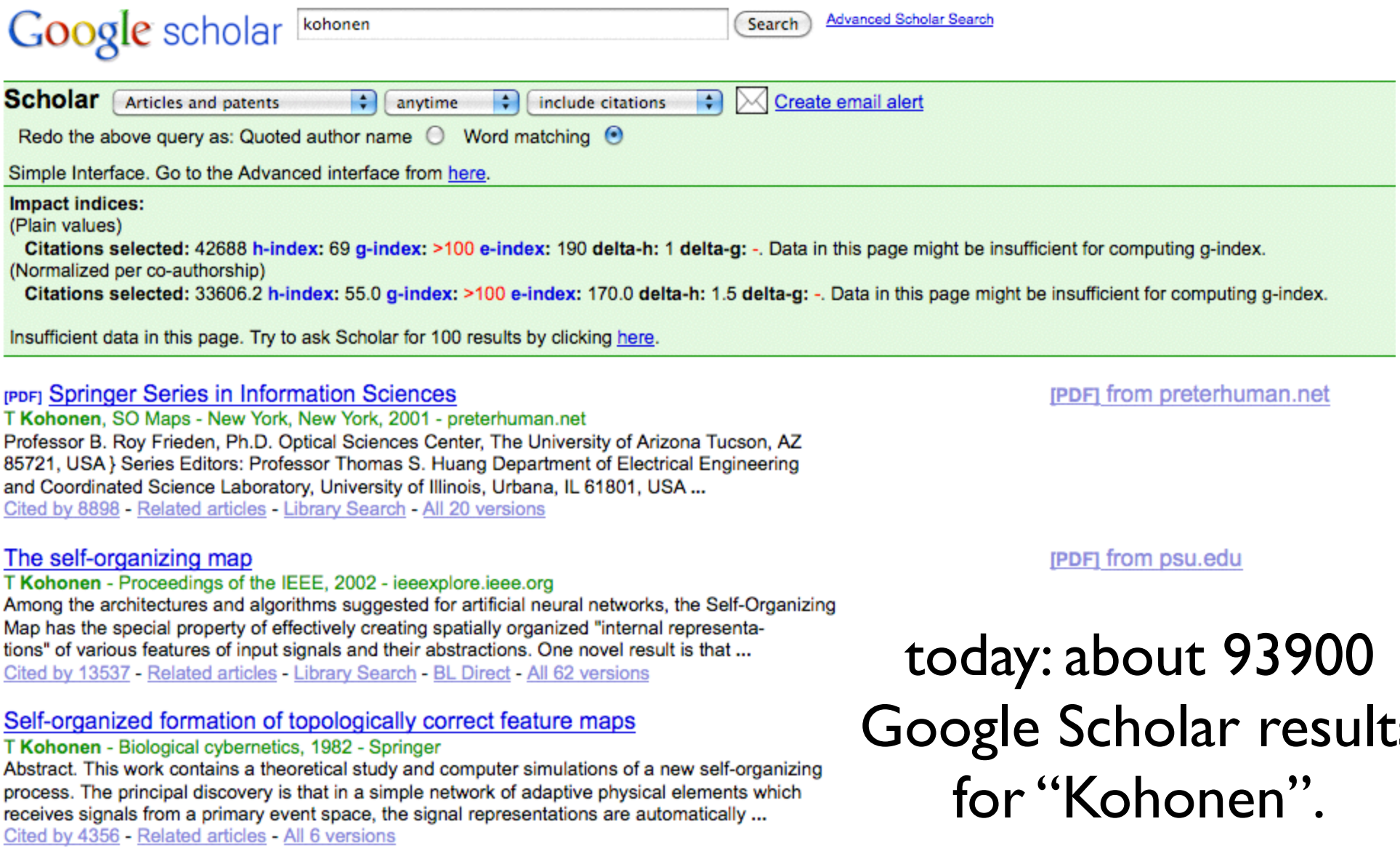
Self-Organizing Maps

Teuvo Kohonen

Dr. Eng., Emeritus Professor of the Academy of Finland; Academician

His research areas are the theory of self-organization, associative memories, neural networks, and pattern recognition, in which he has published over 300 research papers and four monography books. His fifth book is on digital computers. His more recent work is expounded in the ***third, extended edition (2001)*** of his book **Self-Organizing Maps**.

Self-Organizing Maps



today: about 93900 Google Scholar results for "Kohonen".

Self-Organizing Maps

Google self-organizing map

Search About 4,460,000 results

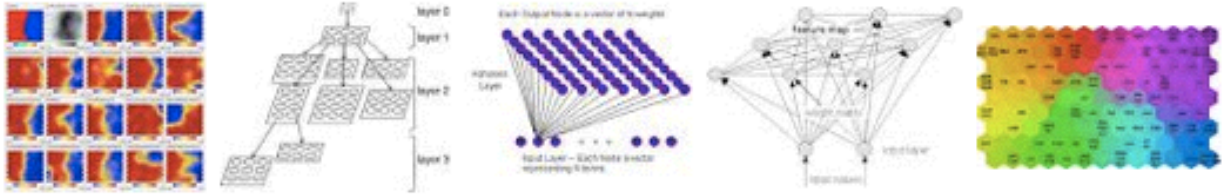
Web
Images
Maps
Books
More

Any time
Past hour
Past 24 hours
Past week
Past month
Past year

All results
Verbatim

Self-organizing map - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Self-organizing_map** - Cached - Similar
A **self-organizing map** (**SOM**) or self-organizing feature map (SOFM) is a type of artificial neural network (ANN) that is trained using unsupervised learning to ...
Learning algorithm - Interpretation - Alternatives - See also

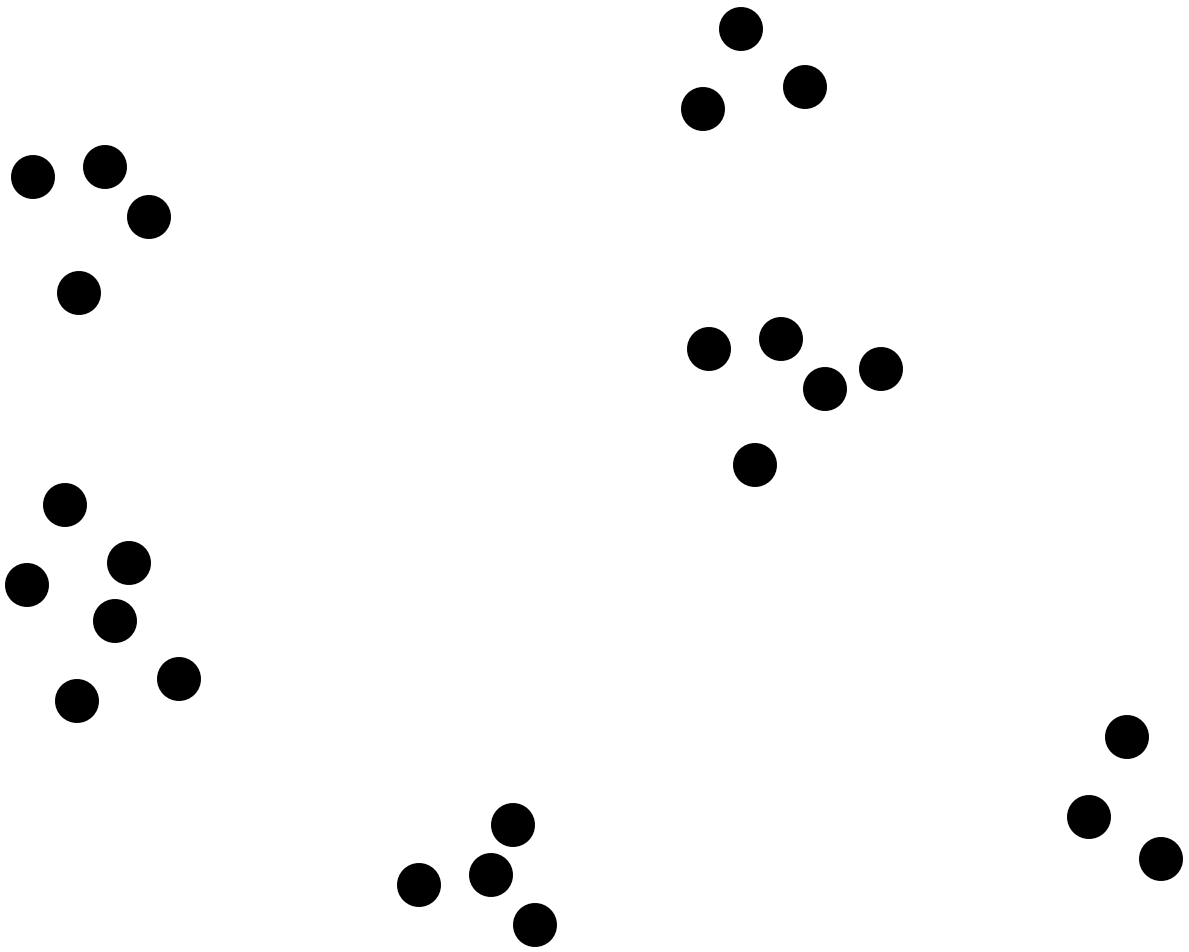
Images for **self-organizing map**



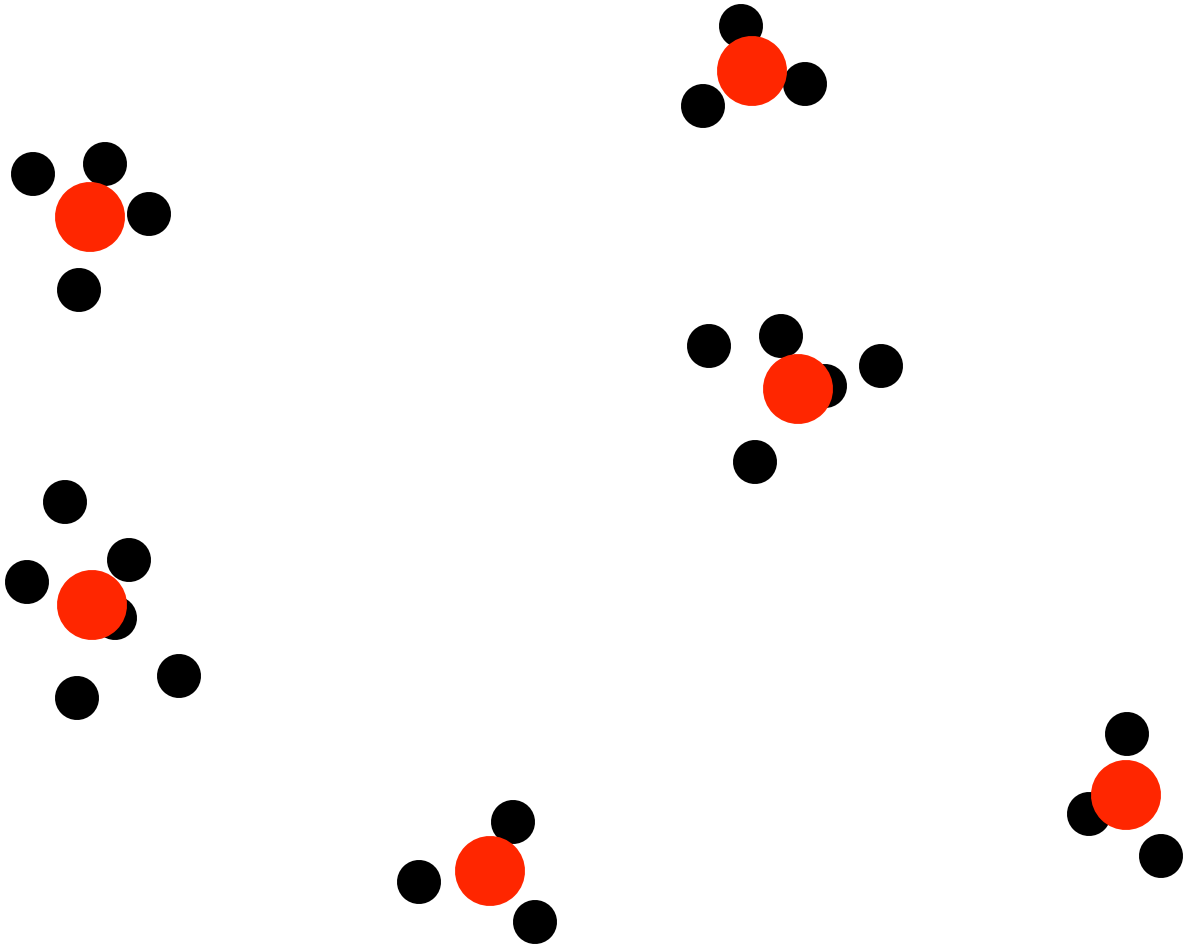
Self-Organizing Maps
davis.wpi.edu/~matt/courses/soms/ - Cached - Similar
23 Mar 1999 ... **Self-organizing maps** (SOMs) are a data visualization technique invented by Professor Teuvo Kohonen which reduce the dimensions of data ...

Self-Organizing Map (SOM)
users.ics.aalto.fi/jhollmen/dippa/node9.html - Cached
The **Self-Organizing Map** is one of the most popular neural network models. It belongs to the category of competitive learning networks. The **Self-Organizing Map** ...

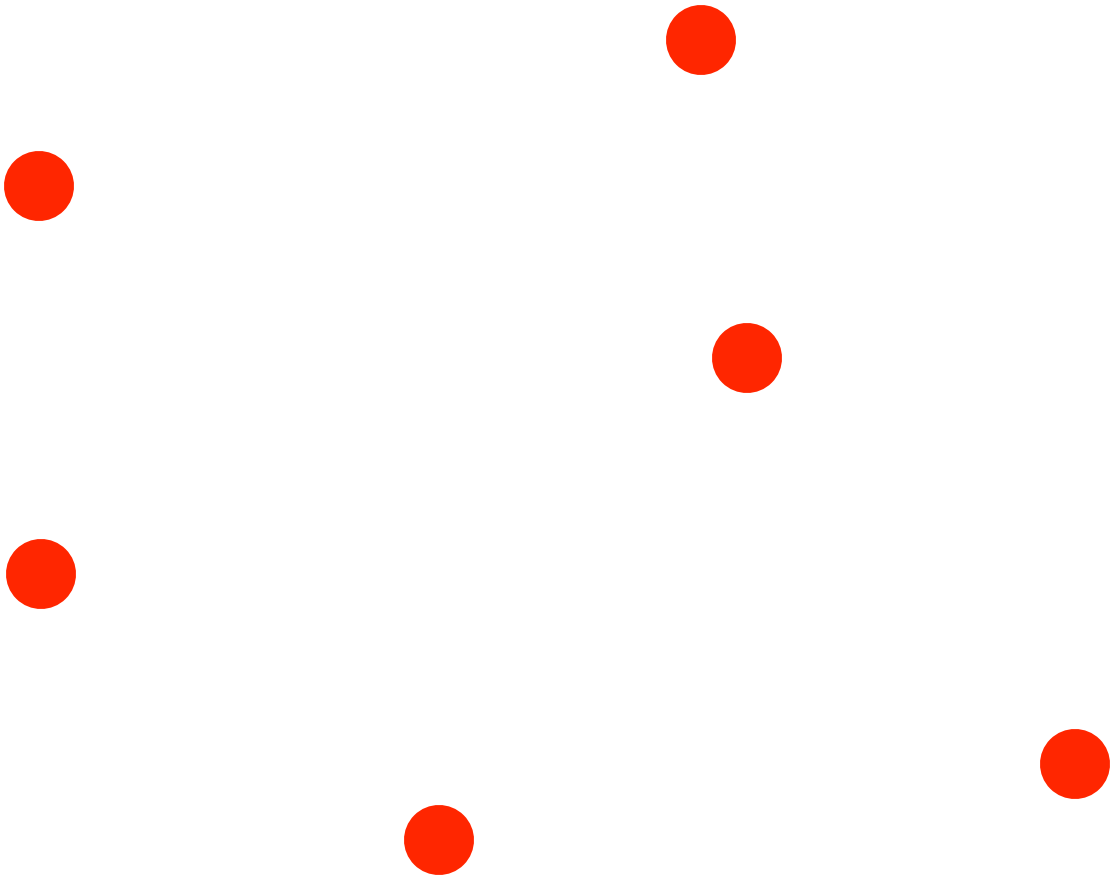
Vector Quantization



Vector Quantization



Vector Quantization



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

Vector Quantization

- One of the 1000 algorithms:
  - random initialization among the samples
  - iterations

$$v(x(t)) = v(x(t)) + \alpha(t)(x(t) - v(x(t)))$$

with v(x(t)) the closest centroid of x(t)
and Robbins-Monro conditions

$$\int_0^\infty \alpha(t)dt = \infty \qquad \int_0^\infty \alpha(t)^2 dt < \infty$$

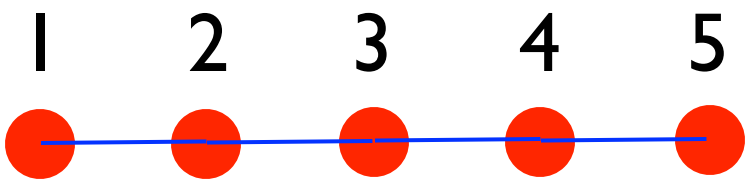
Vector Quantization

- Matlab examples

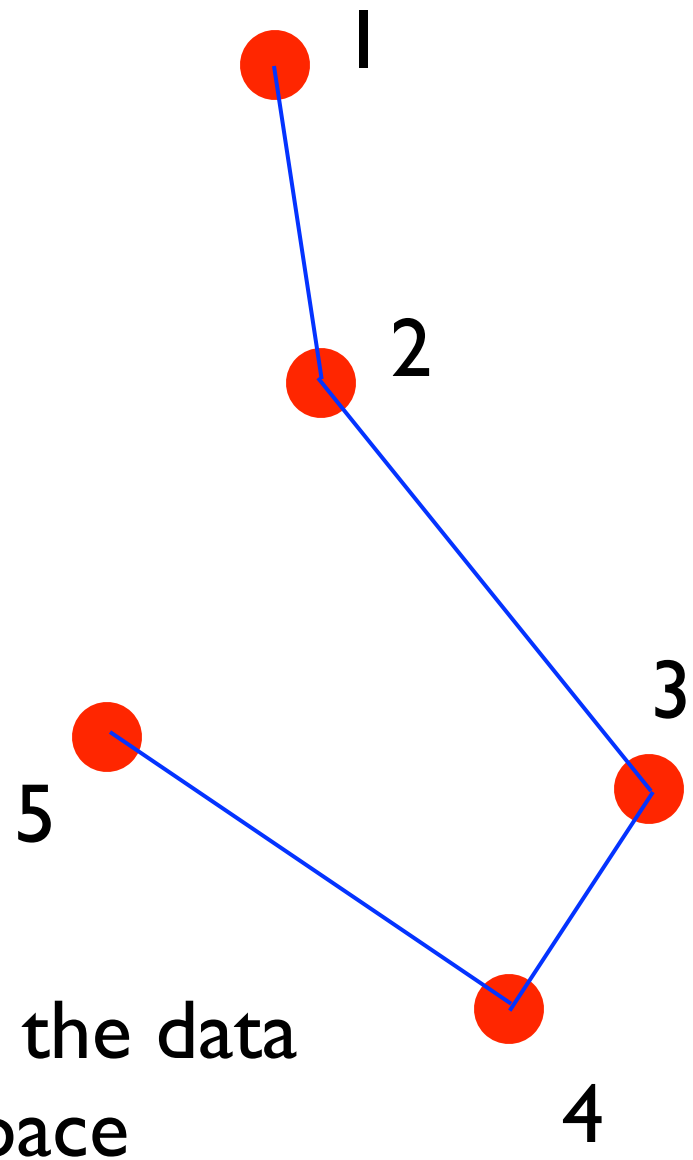
Self-Organizing Maps

- like stretching a sheet of plastic by pulling on certain points on it

SOM



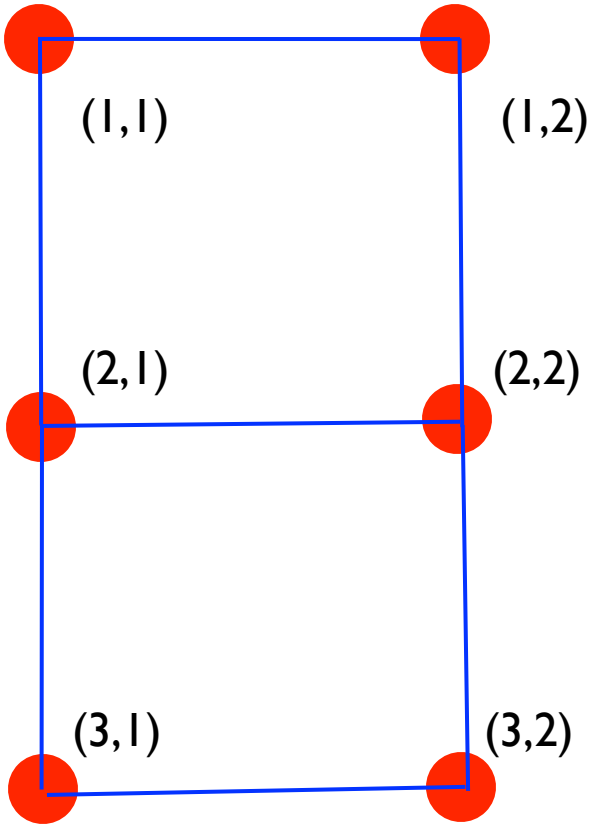
Structure of the map (definition)



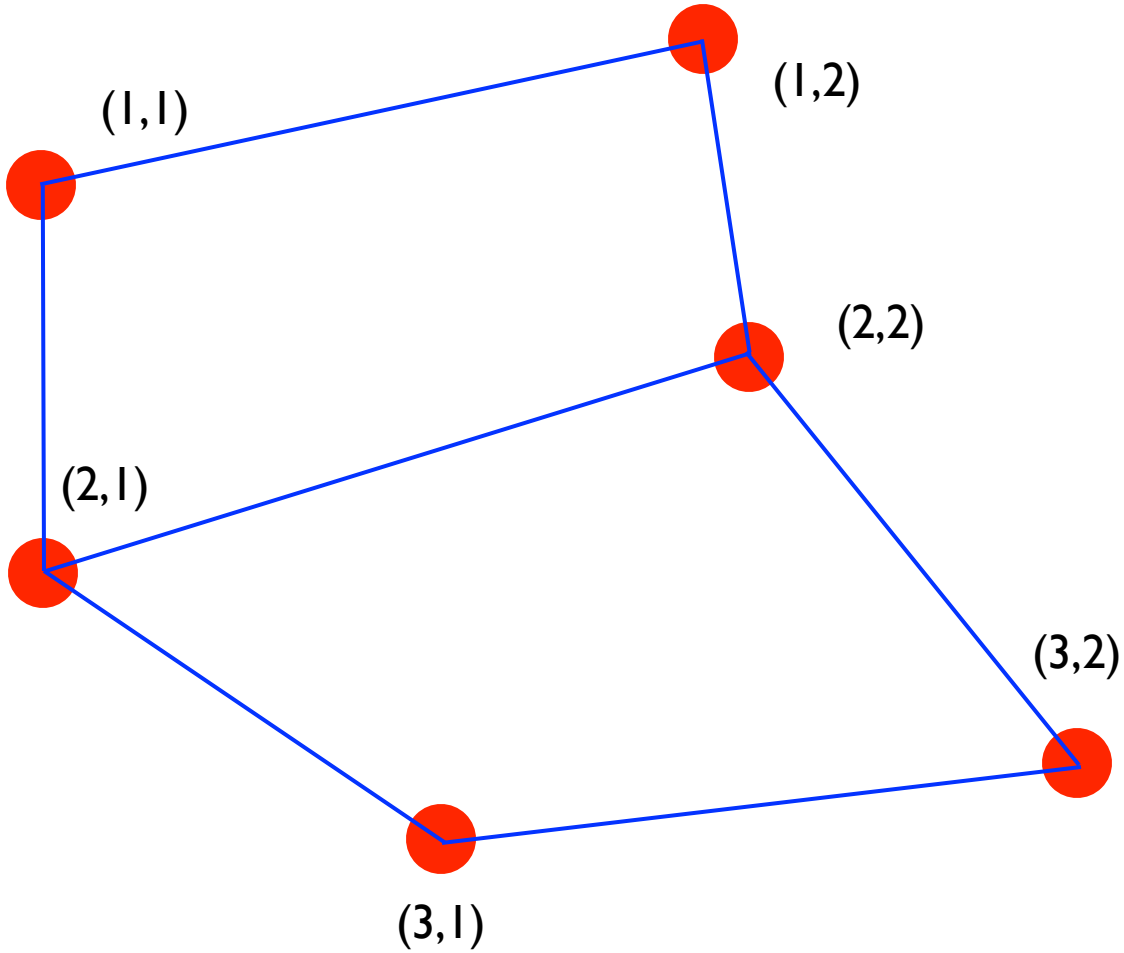
Map in the data space

SOM

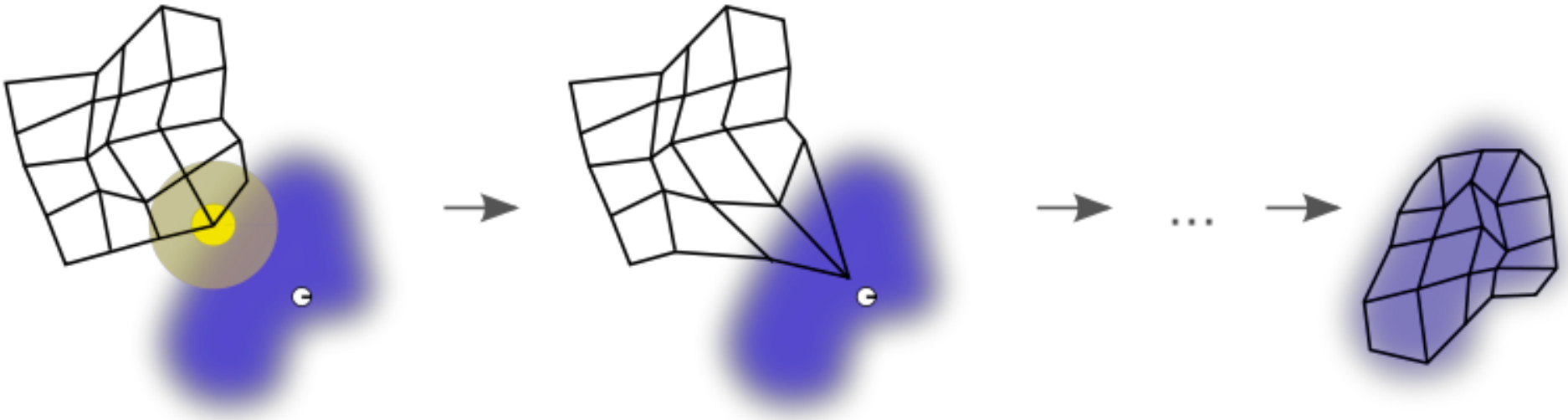
Structure of the map (definition)



Map in the data space



SOM



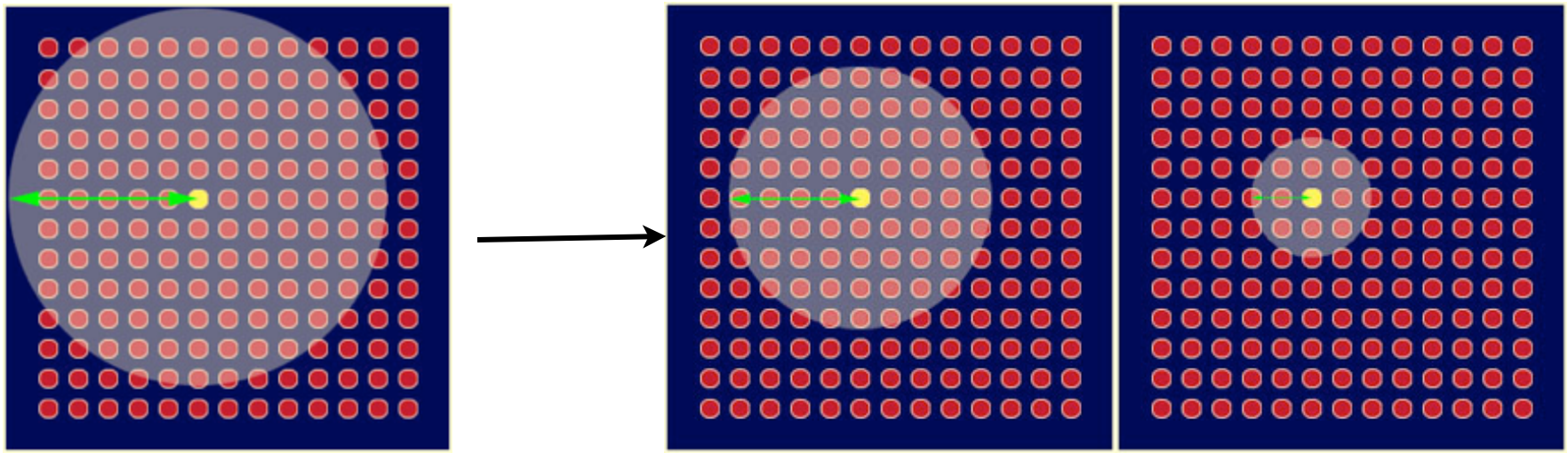
SOM

- 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:

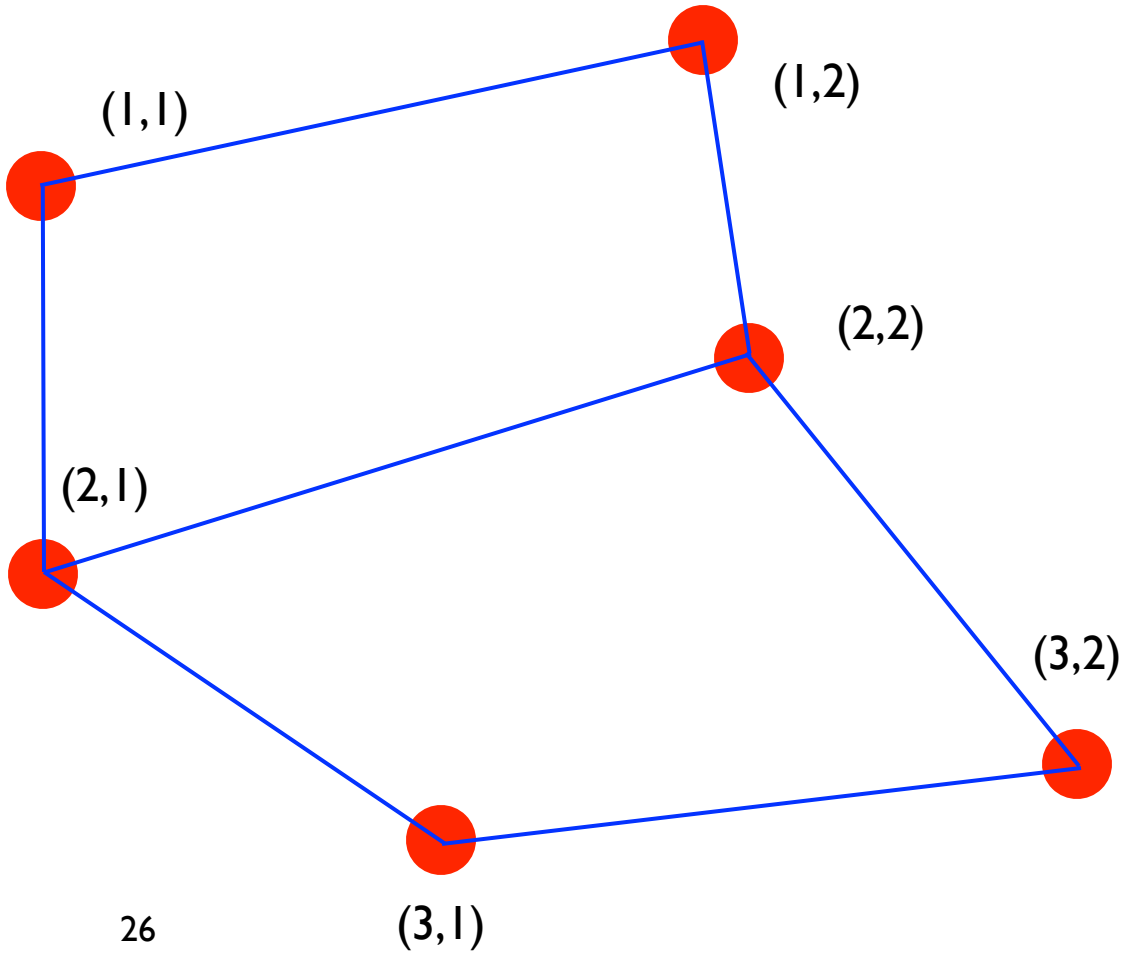


SOM

- Matlab Examples

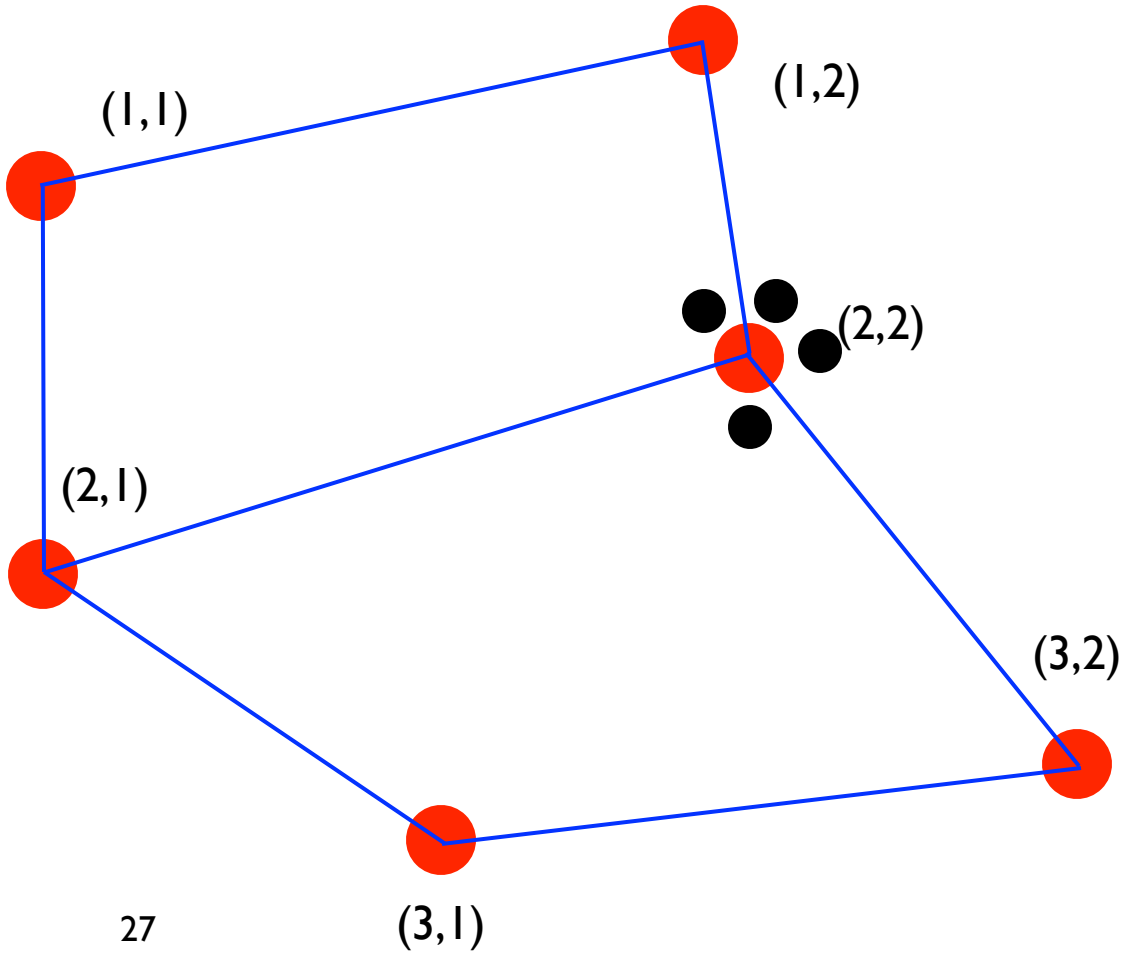
SOM

- And then? What about visualization?



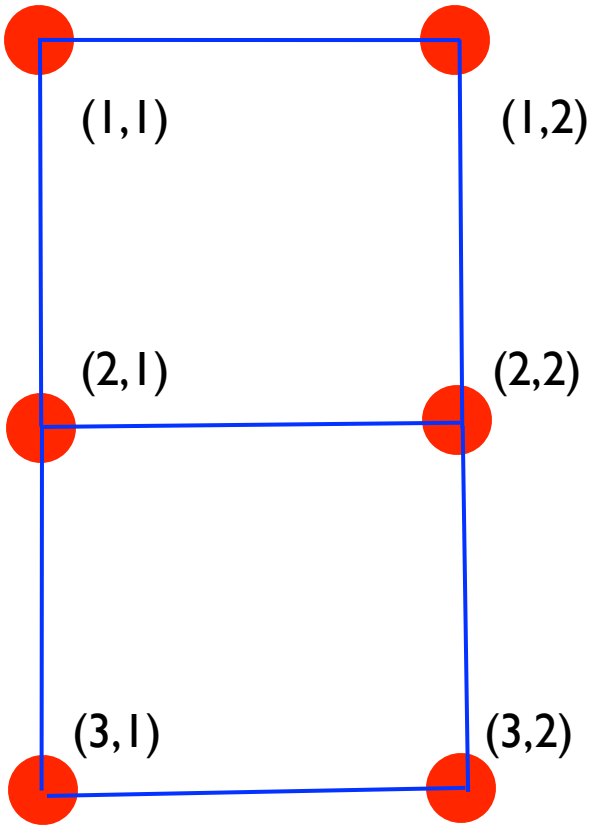
SOM

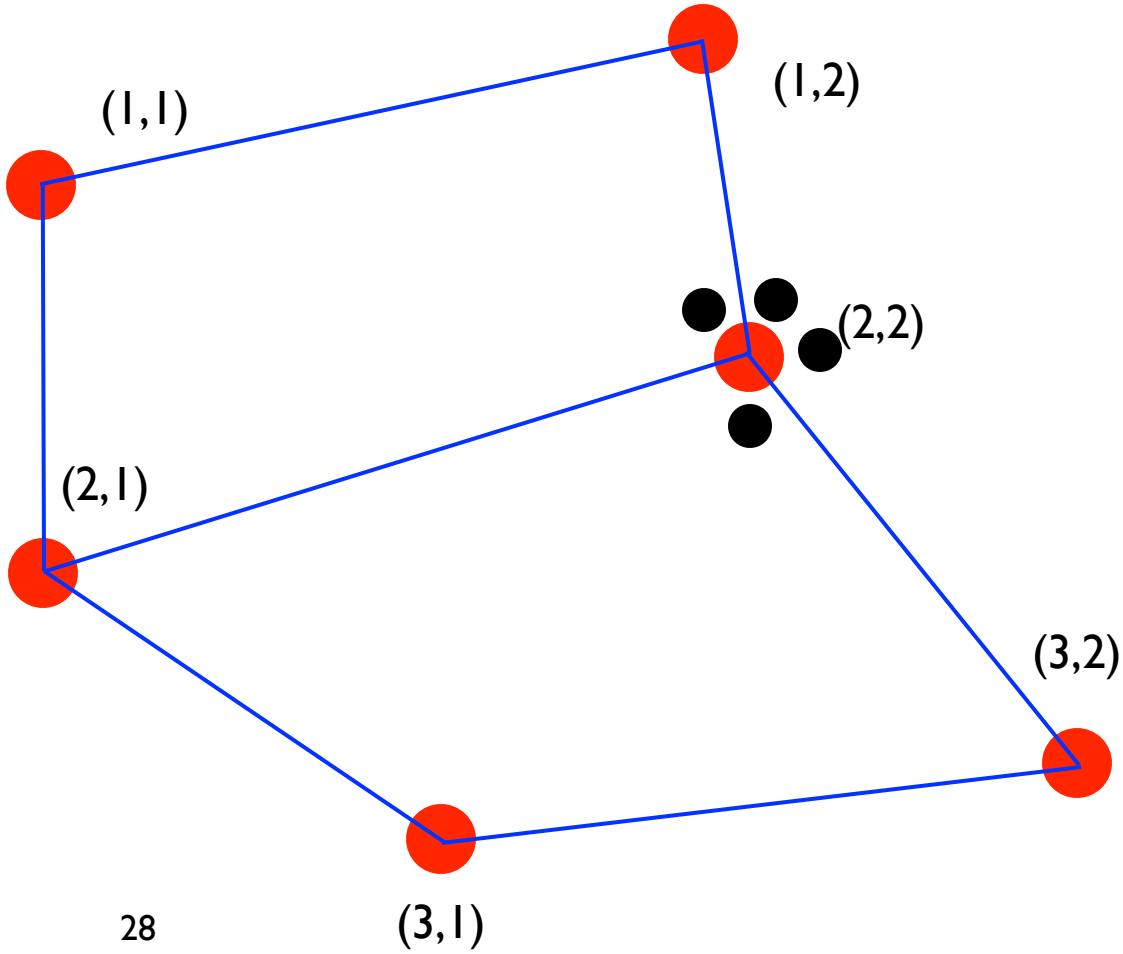
- And then? What about visualization?



SOM

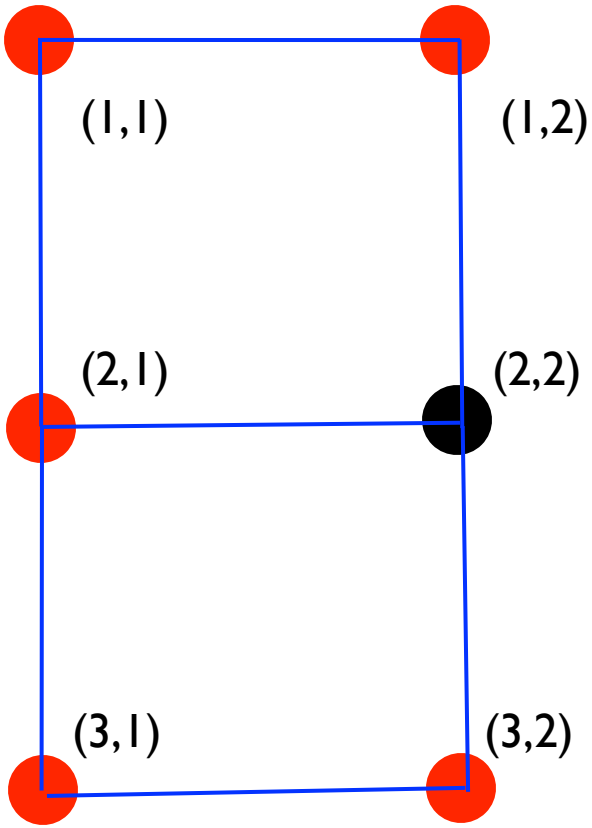
- And then? What about visualization?

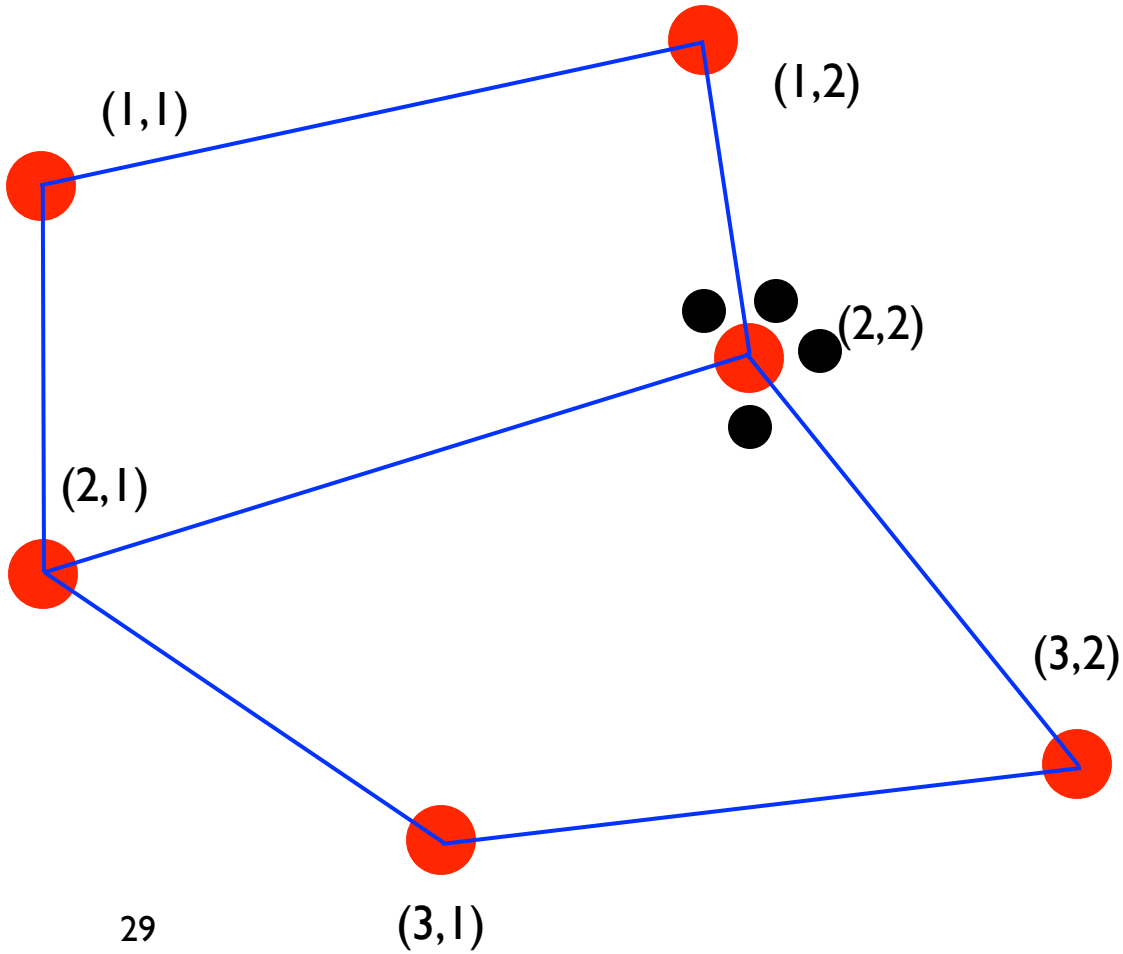




SOM

- And then? What about visualization?

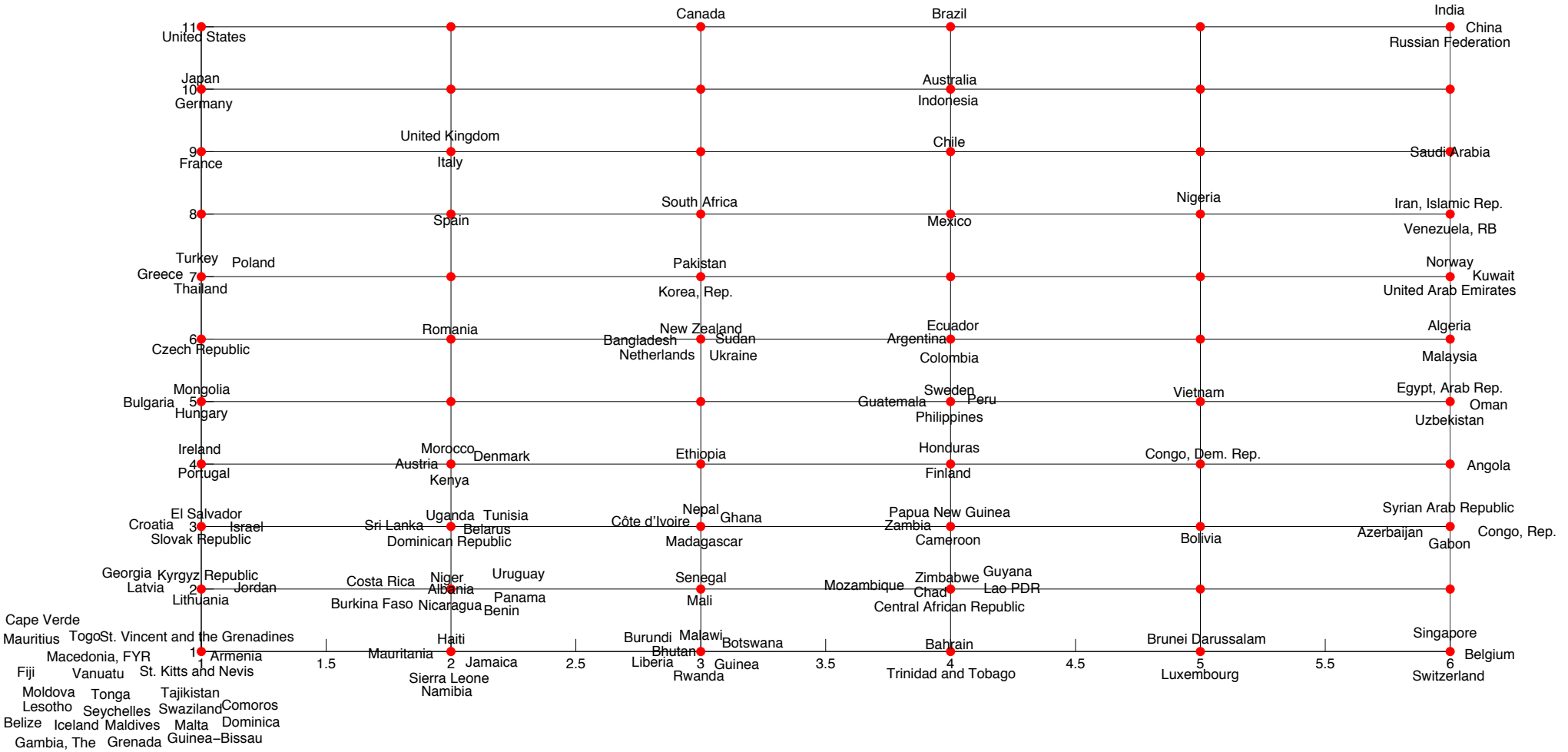




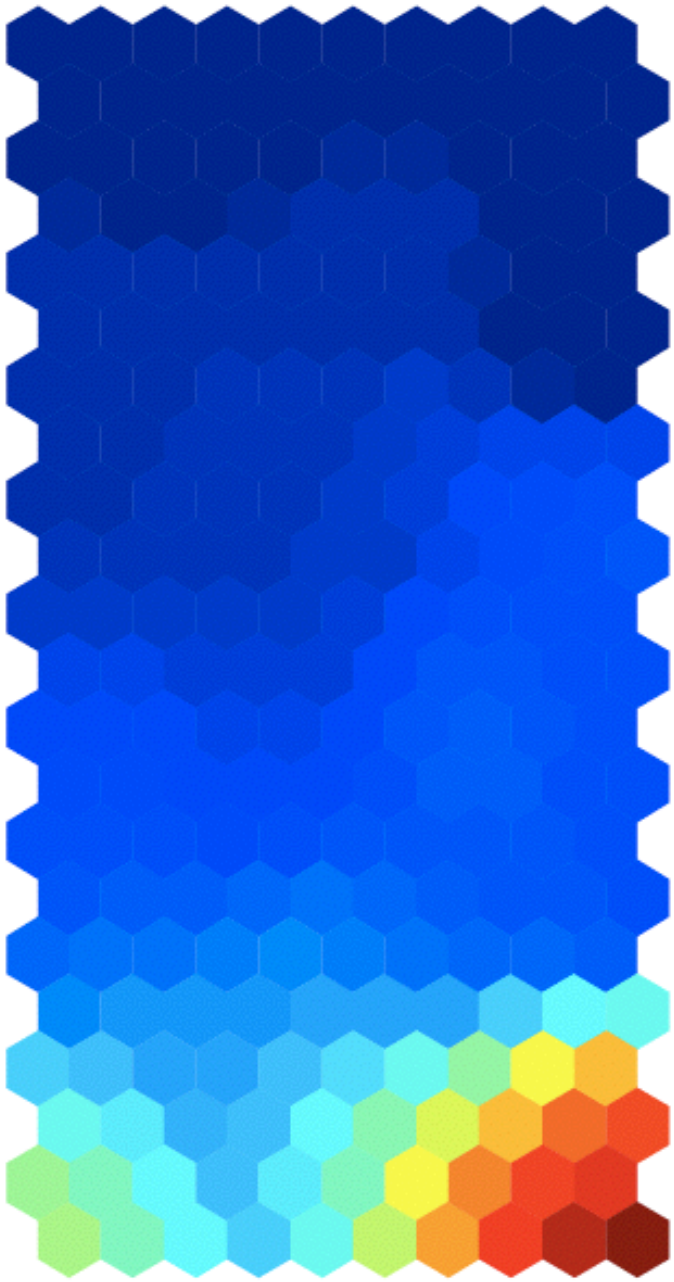
SOM

- Wealth Data Example: Excel + Matlab

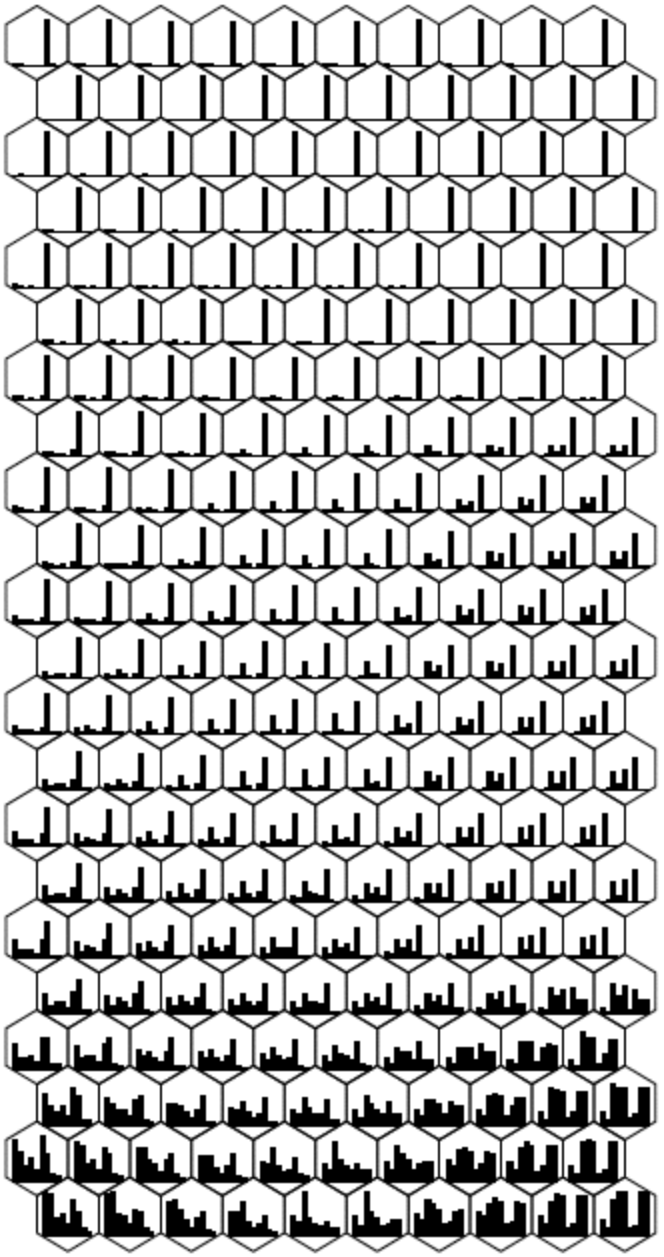
SOM


11
United States
Canada
Brazil
India
China
Russian Federation
10
Japan
Germany
Australia
Indonesia
9
France
United Kingdom
Italy
Chile
Saudi Arabia
8
Spain
South Africa
Mexico
Nigeria
Iran, Islamic Rep.
Venezuela, RB
7
Turkey
Greece
Poland
Thailand
Pakistan
Korea, Rep.
Norway
Kuwait
United Arab Emirates
6
Czech Republic
Romania
New Zealand
Bangladesh
Netherlands
Sudan
Ukraine
Ecuador
Argentina
Colombia
Algeria
Malaysia
5
Bulgaria
Mongolia
Hungary
Sweden
Guatemala
Peru
Philippines
Vietnam
Egypt, Arab Rep.
Oman
Uzbekistan
4
Ireland
Portugal
Morocco
Austria
Denmark
Kenya
Ethiopia
Honduras
Finland
Congo, Dem. Rep.
Angola
3
Croatia
El Salvador
Israel
Slovak Republic
Uganda
Tunisia
Sri Lanka
Belarus
Dominican Republic
Nepal
Côte d'Ivoire
Ghana
Madagascar
Papua New Guinea
Zambia
Cameroon
Bolivia
Syrian Arab Republic
Azerbaijan
Congo, Rep.
Gabon
2
Georgia
Latvia
Kyrgyz Republic
Jordan
Lithuania
Costa Rica
Niger
Albania
Burkina Faso
Nicaragua
Uruguay
Panama
Benin
Senegal
Mali
Mozambique
Zimbabwe
Chad
Central African Republic
Guyana
Lao PDR
1
Cape Verde
Mauritius
Togo
St. Vincent and the Grenadines
Macedonia, FYR
Armenia
Fiji
Vanuatu
St. Kitts and Nevis
Moldova
Tonga
Tajikistan
Lesotho
Seychelles
Swaziland
Comoros
Belize
Iceland
Maldives
Malta
Dominica
Gambia, The
Grenada
Guinea–Bissau
Haiti
Mauritania
Jamaica
Sierra Leone
Namibia
Burundi
Malawi
Botswana
Bhutan
Liberia
Guinea
Rwanda
Bahrain
Trinidad and Tobago
Brunei Darussalam
Luxembourg
Singapore
Belgium
Switzerland
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6

SOM



Values of the prototype vectors
shown on the SOM grid.
Left: one of the feature values.
Right: many of the feature values.



From Juha Vesanto's
doctoral thesis, 2002

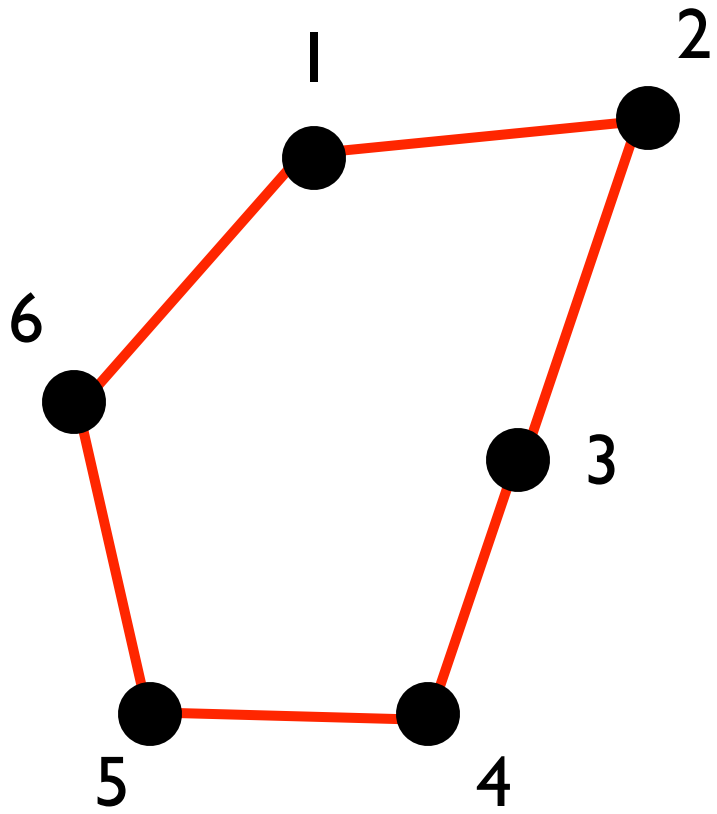
Laplacian eigenmap

- Laplacian eigenmap is a spectral method, like PCA.
- construct *k*-nearest neighbors graph. Assign $W_{ij}=1$, if *i* and *j* are neighbors, otherwise assign $W_{ij}=0$. Define diagonal matrix *D*, $D_{ii}=\sum_j W_{ij}$, and graph Laplacian, $L=D-W$.
- The embedding of data points is given by the eigenvectors of *L*, corresponding to the *d* smallest non-zero eigenvalues.

Laplacian eigenmap

N=6

k=2



octave:20> L
L =

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

octave:21> [V,D] = eig(L)
V =

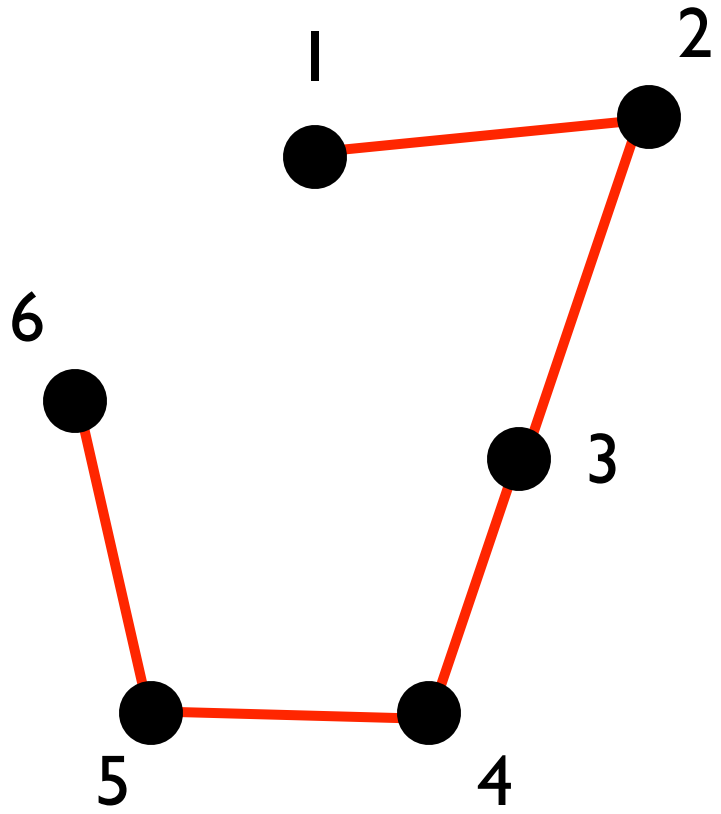
-0.408248	0.458032	-0.351483	0.551189	-0.171826	0.408248
0.408248	-0.533409	-0.220926	0.424400	0.391430	0.408248
-0.408248	0.075377	0.572409	-0.126788	0.563257	0.408248
0.408248	0.458032	-0.351483	-0.551189	0.171826	0.408248
-0.408248	-0.533409	-0.220926	-0.424400	-0.391430	0.408248
0.408248	0.075377	0.572409	0.126788	-0.563257	0.408248

D =

-4.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.00000	-3.00000	0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	-3.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	-1.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	-1.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

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
```



Laplacian eigenmap

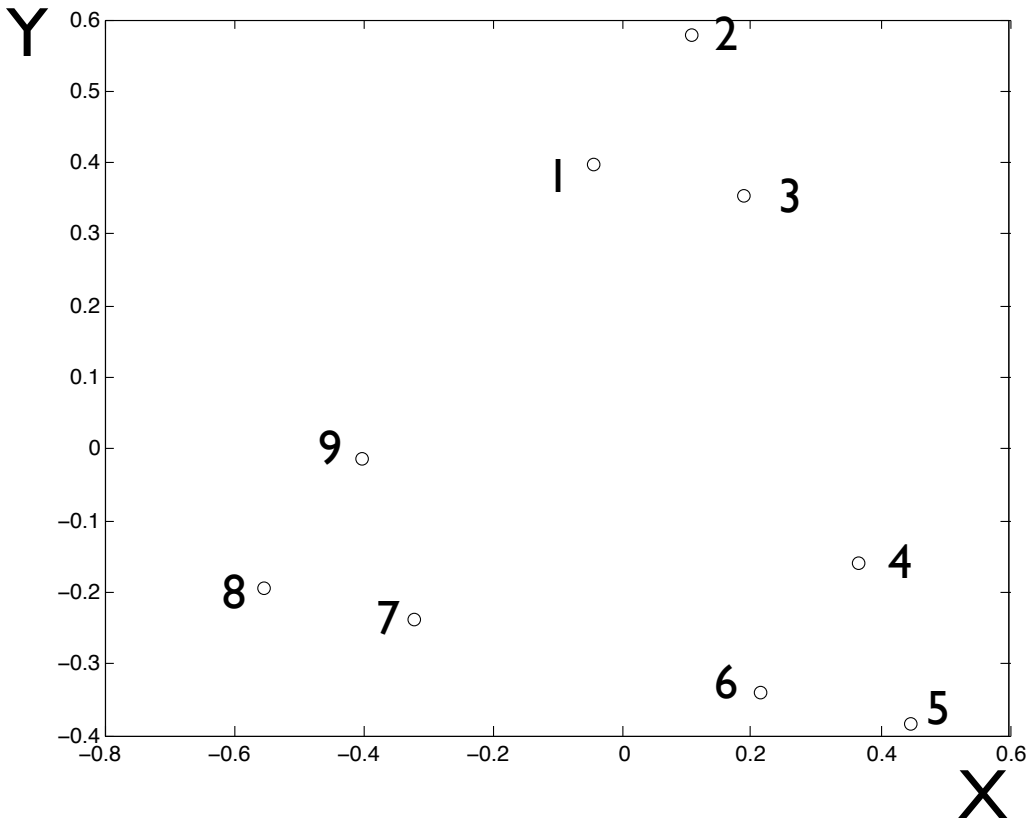
```
>> L

L =

    -3     1     1     0     0     0     0     0     1
     1    -2     1     0     0     0     0     0     0
     1     1    -3     1     0     0     0     0     0
     0     0     1    -3     1     1     0     0     0
     0     0     0     1    -2     1     0     0     0
     0     0     0     1     1    -3     1     0     0
     0     0     0     0     0     1    -3     1     1
     0     0     0     0     0     0     1    -2     1
     1     0     0     0     0     0     1     1    -3

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

V =
```

Y X

-0.4082	0.5014	-0.1803	-0.3099	0.3433	-0.2527	0.3982	-0.0451	0.3333
-0.0000	-0.1232	0.2895	0.5573	0.1121	0.3482	0.5775	0.1093	0.3333
0.4082	-0.2177	-0.4863	-0.2475	-0.4554	-0.0955	0.3542	0.1874	0.3333
-0.4082	-0.0946	0.5243	-0.2475	-0.4554	-0.0955	-0.1601	0.3674	0.3333
-0.0000	-0.1891	-0.2514	0.3798	0.3669	-0.4069	-0.3834	0.4455	0.3333
0.4082	0.5300	0.0546	-0.1323	0.0885	0.5024	-0.3394	0.2130	0.3333
-0.4082	-0.4068	-0.3441	-0.1323	0.0885	0.5024	-0.2381	-0.3223	0.3333
0.0000	0.3123	-0.0380	0.4422	-0.4319	-0.2497	-0.1941	-0.5548	0.3333
0.4082	-0.3123	0.4317	-0.3099	0.3433	-0.2527	-0.0148	-0.4005	0.3333

D =

-5.0000	0	0	0	0	0	0	0	0
0	-4.3028	0	0	0	0	0	0	0
0	0	-4.3028	0	0	0	0	0	0
0	0	0	-3.0000	0	0	0	0	0
0	0	0	0	-3.0000	0	0	0	0
0	0	0	0	0	-3.0000	0	0	0
0	0	0	0	0	0	-0.6972	0	0
0	0	0	0	0	0	0	-0.6972	0
0	0	0	0	0	0	0	0	-0.0000



Curvilinear component analysis (CCA)

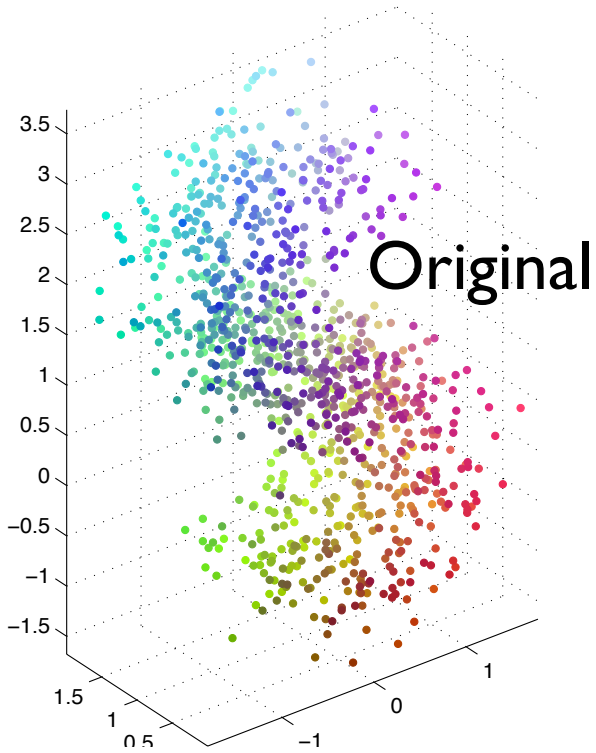
- Demartines, Hérault, 1997.
- *Curvilinear component analysis* (CCA) is like (absolute) MDS, except that only short distances are taken into account.
- More formally, the cost function reads

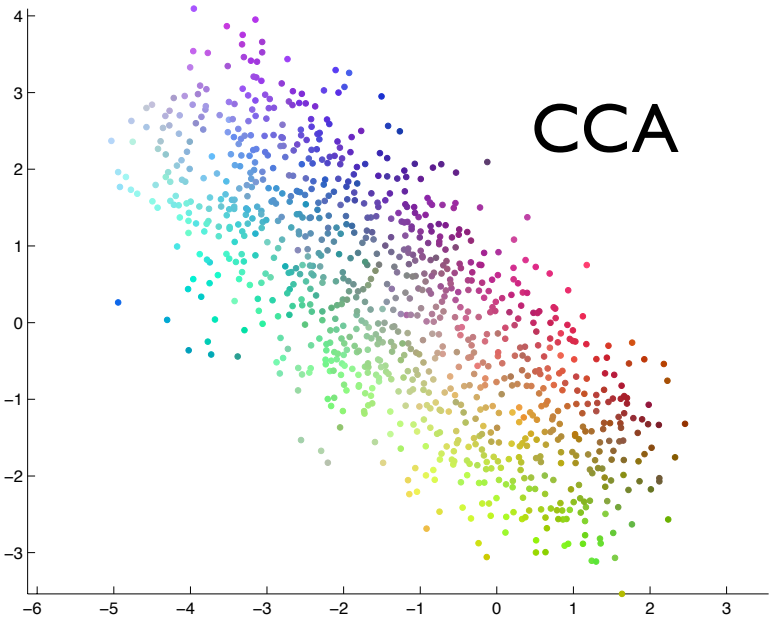
$$\sigma_r = \sum_{i<j} \left(d(x_i, x_j) - d(y_i, y_j)\right)^2 F(d(y_i, y_j), \lambda_y)$$

where $F(d,\lambda_y)$ 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 component analysis (CCA)

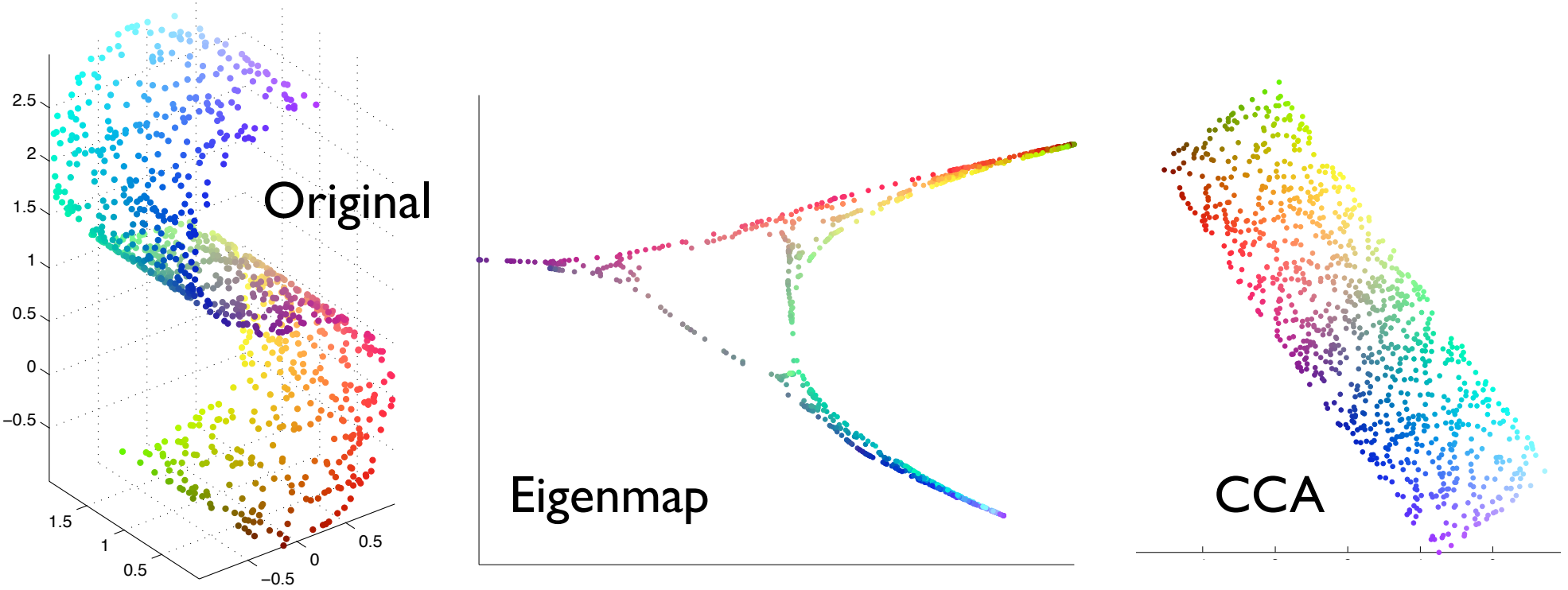
- CCA performs generally well in terms of precision; it appears to be quite robust.
- Notice outliers at right: they are result of small neighborhood.





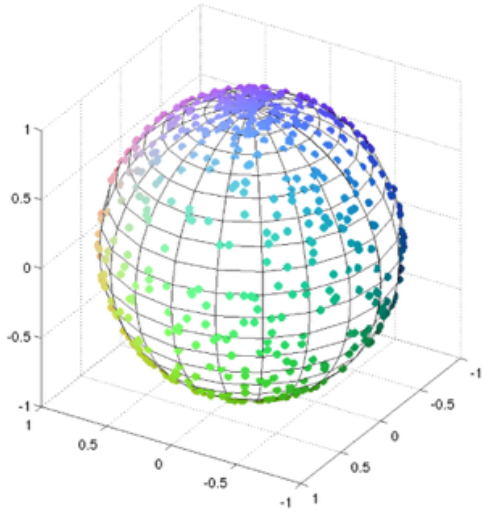
Figures by Jarkko Venna

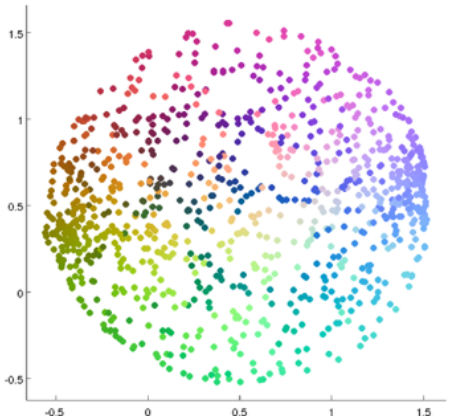
Laplacian eigenmap



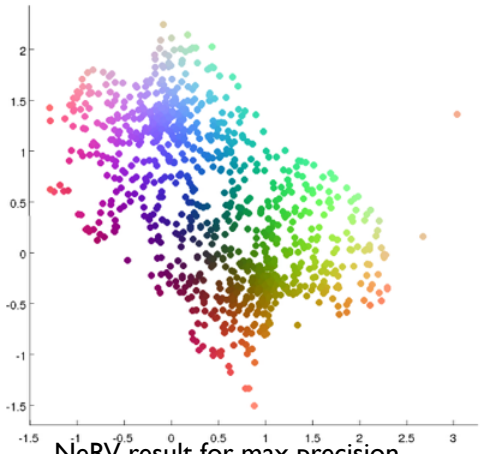
Methods that directly use precision and recall?

- We interpreted some methods by precision and recall. Most methods don't optimize these, they use various cost functions.
- The Neighbor Retrieval Visualizer (Venna & Kaski 2007; Venna, Peltonen, Nybo, Aidos, Kaski 2010) directly maximizes a user-specified tradeoff of precision & recall





NeRV, result for max recall



NeRV, result for max precision

- Stochastic Neighbor Embedding (Hinton&Roweis 2002) can also be interpreted like this: maximizes recall
- Details later on the Neighbor Embedding lectures

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

Isomap

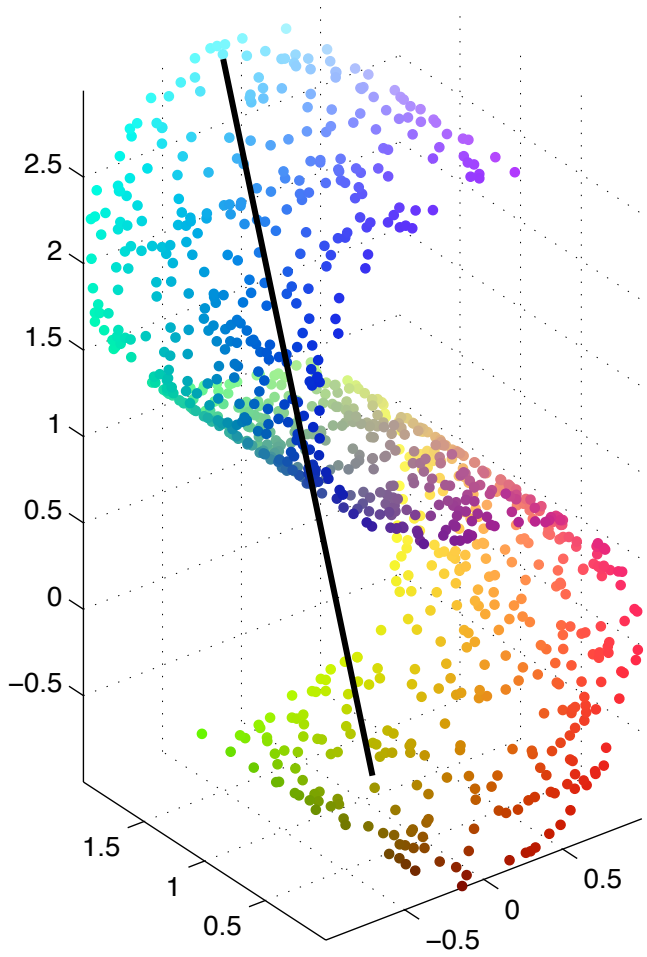
Multidimensional scaling methods tried to preserve all squared distances —> preservation of largest distances had biggest effect on the cost

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.

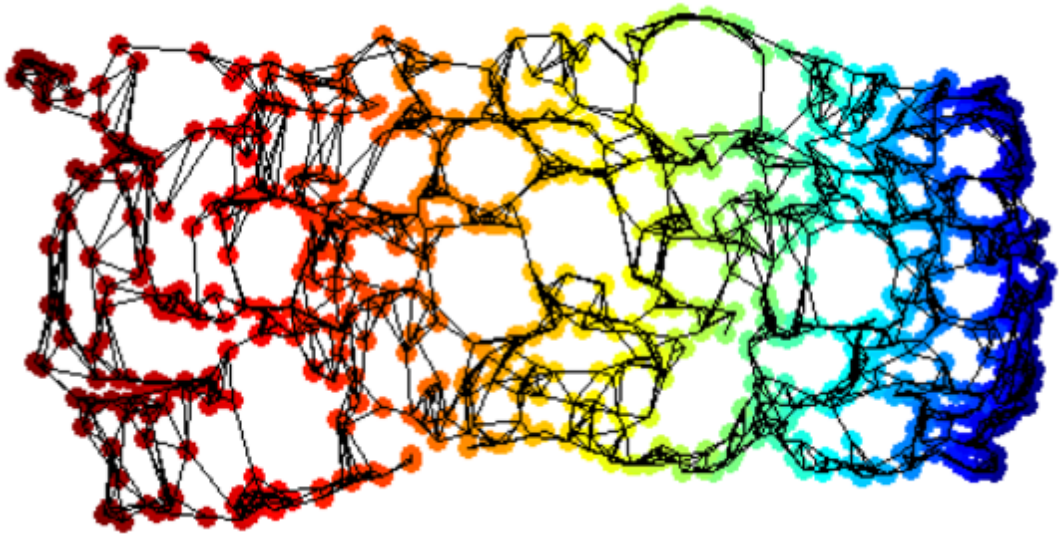


Euclidean distance between two points corresponds to the length of the line connecting the points. The line usually does not follow the manifold of the data.

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.

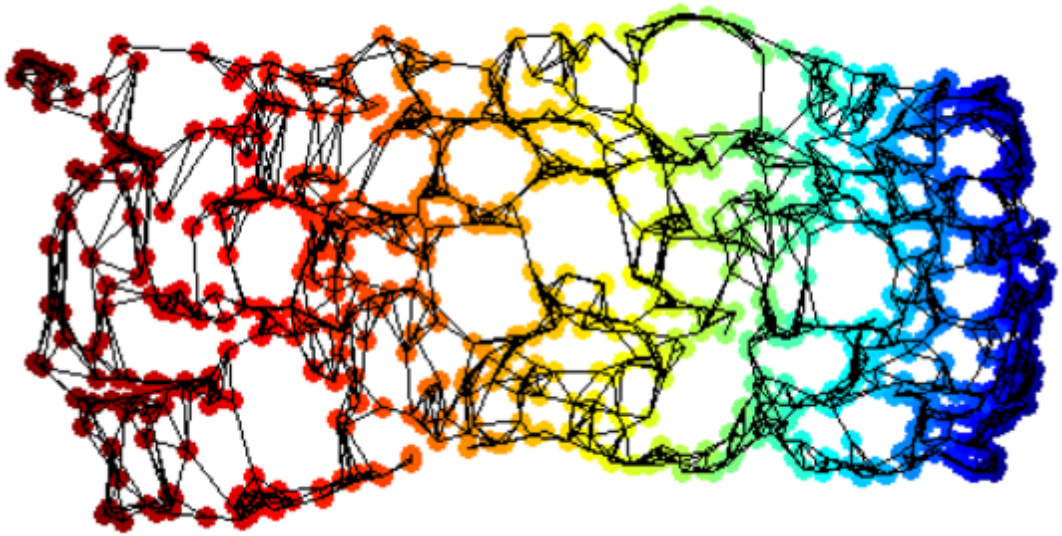


Isomap

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.

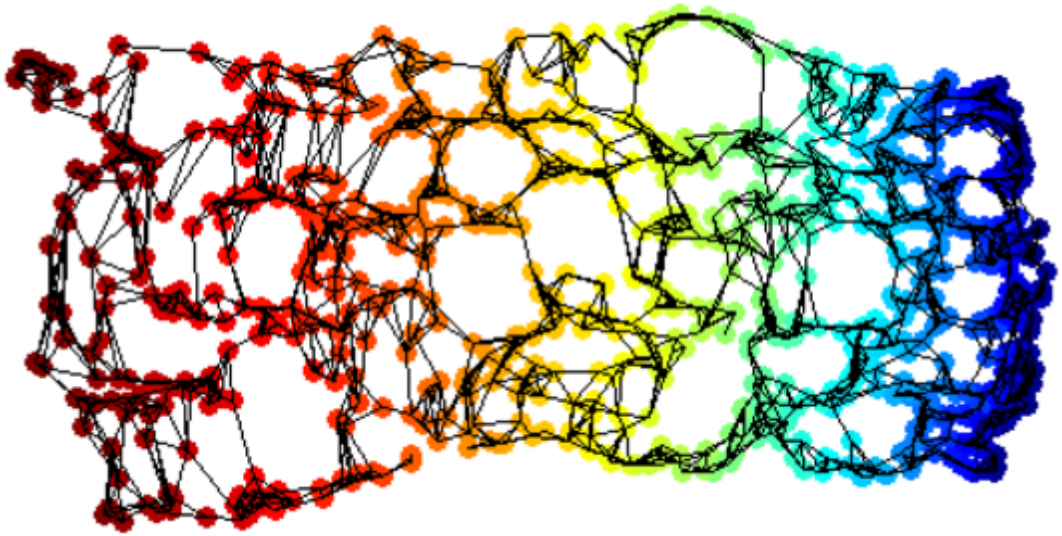


Isomap

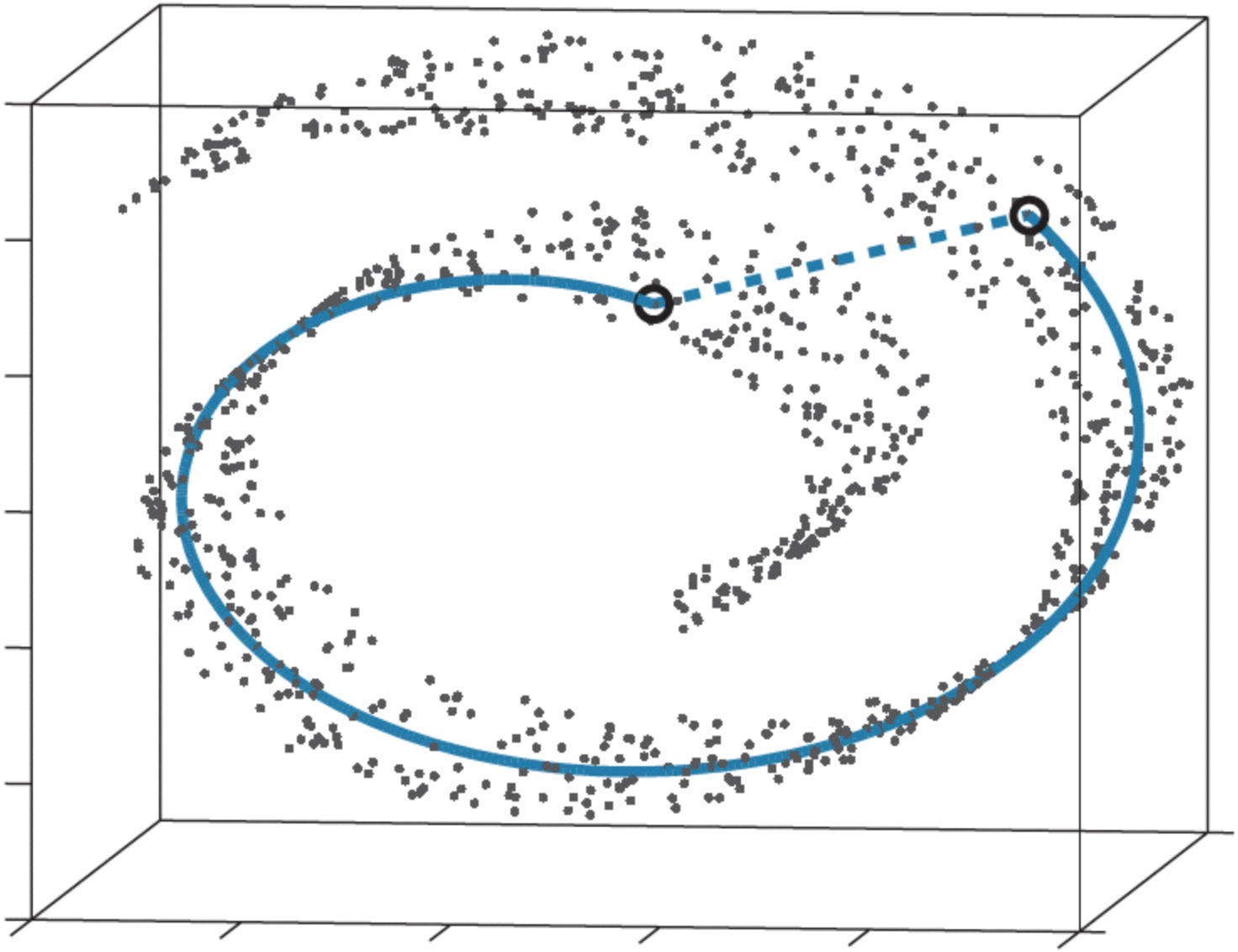
Afterwards follow standard MDS procedure: find low-dimensional coordinates whose Euclidean squared distances best approximate the above-defined squared shortest-path distances.

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.)

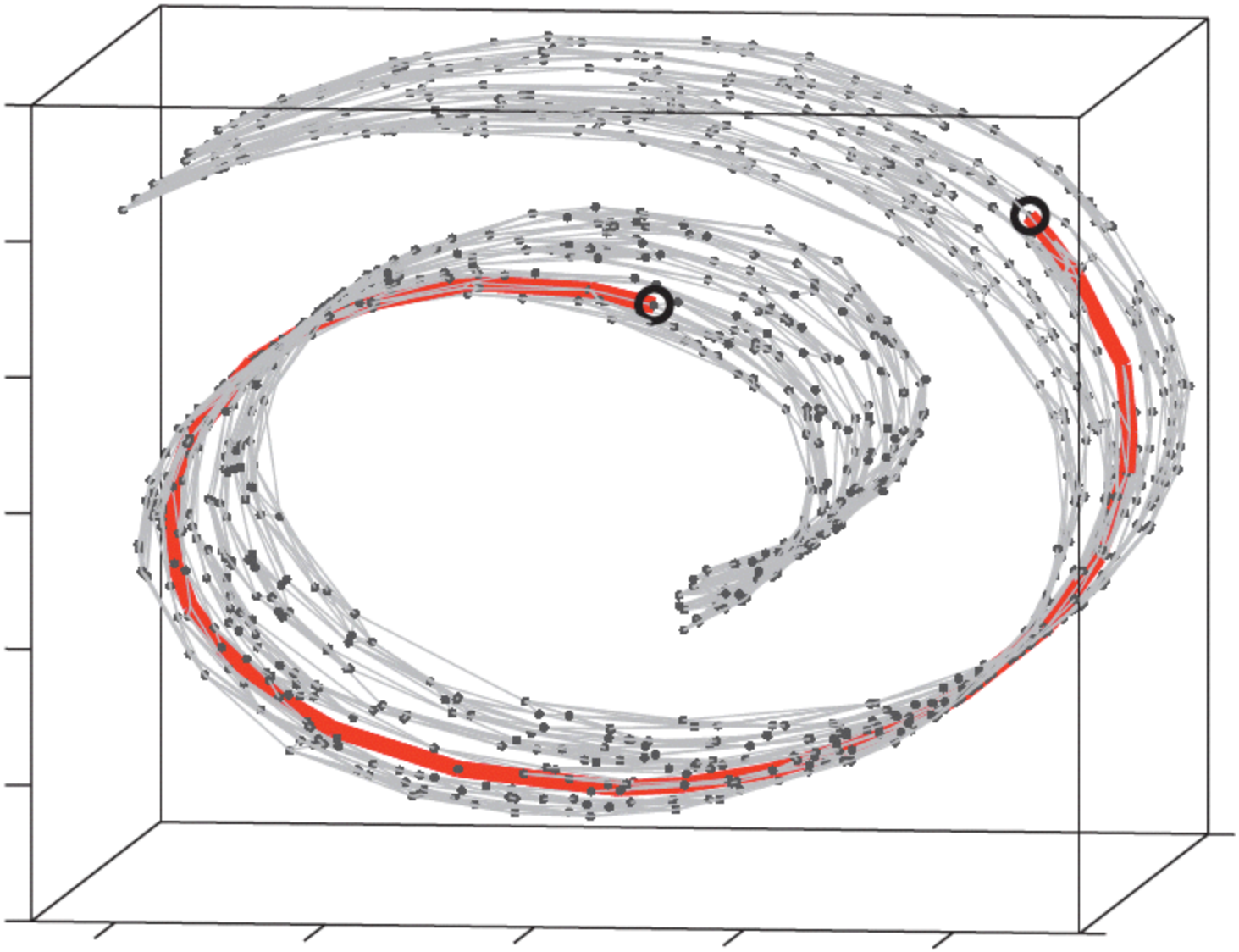


Isomap



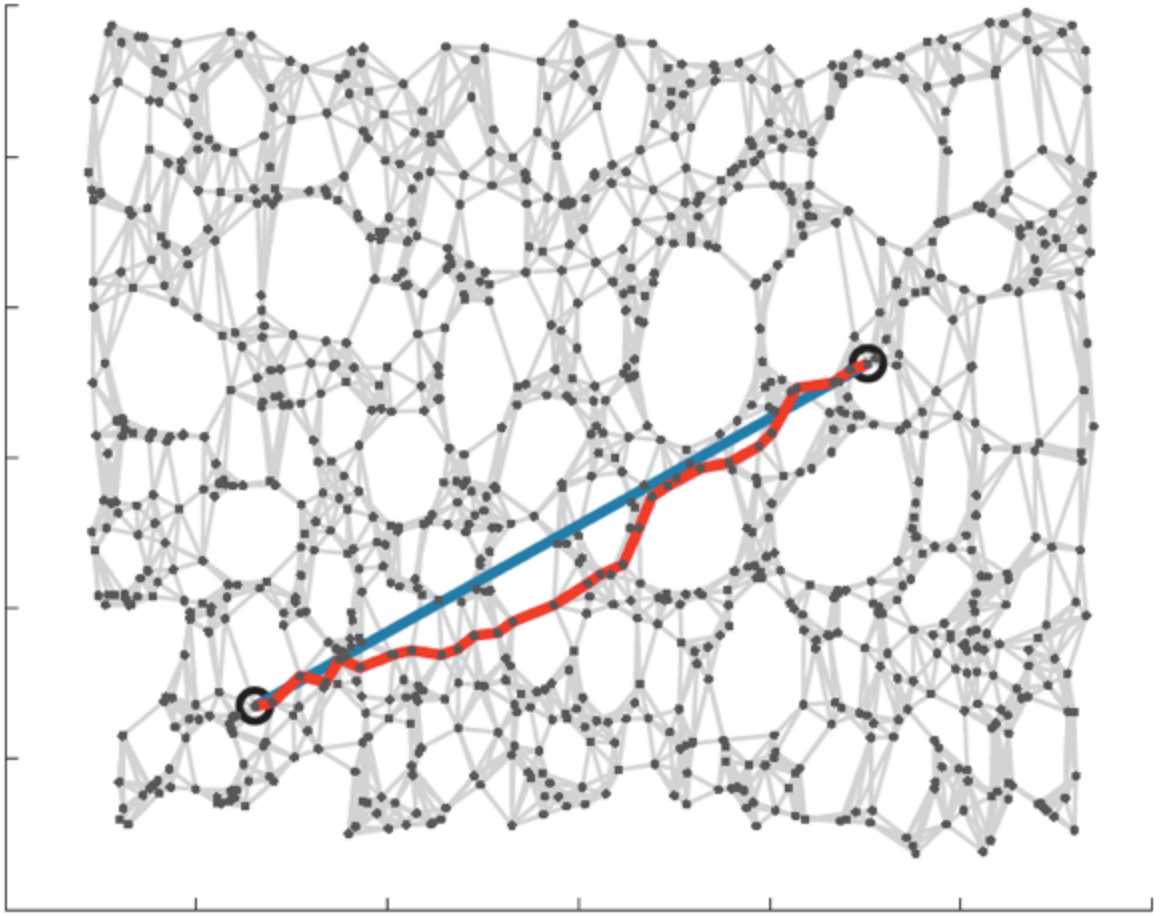
Swiss roll example: Euclidean distance can “jump” across the manifold, while the “ideal” distance goes along the manifold.

Isomap



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

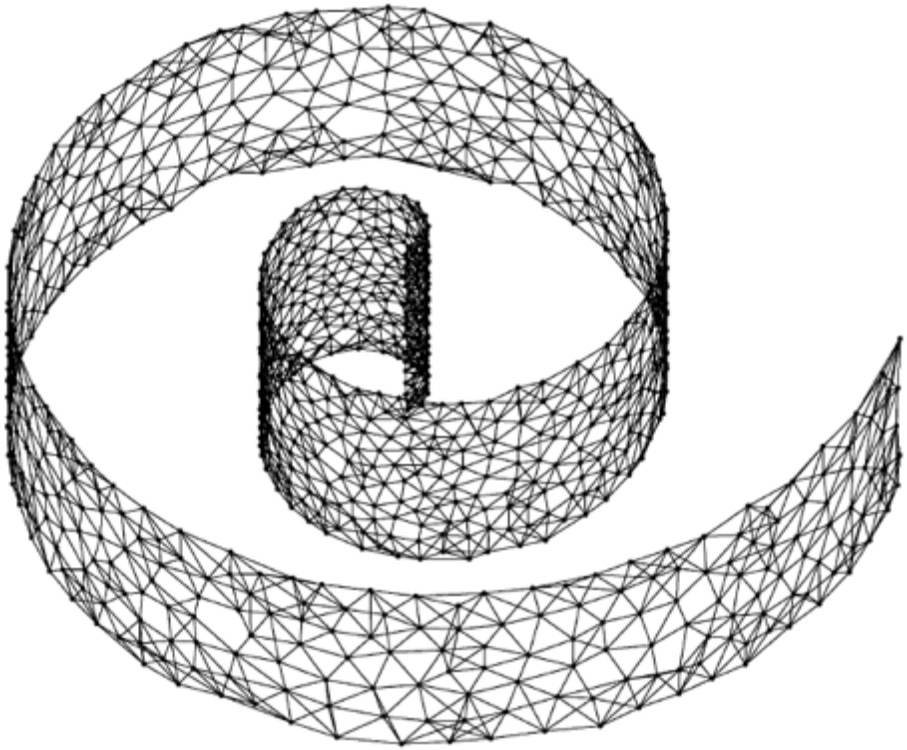
Isomap



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

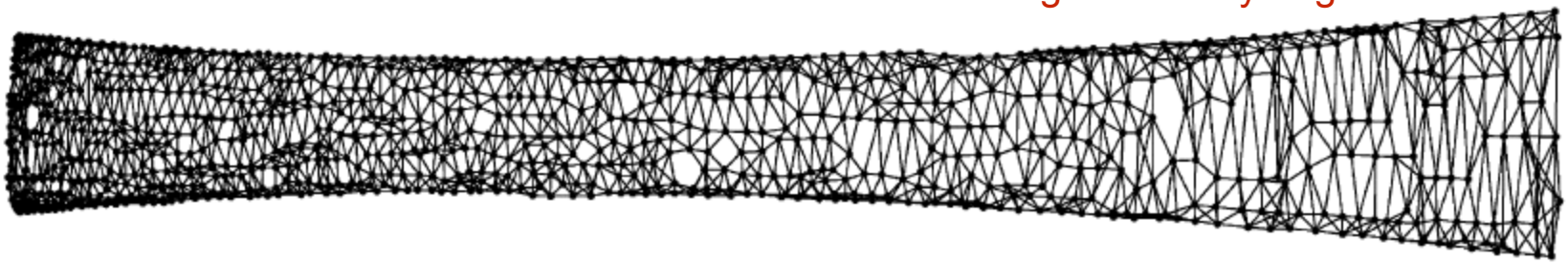
Note: geodesics are approximated directly by the Euclidean (straight-line) low-dim. distances, not by shortest paths along the graph.

Isomap



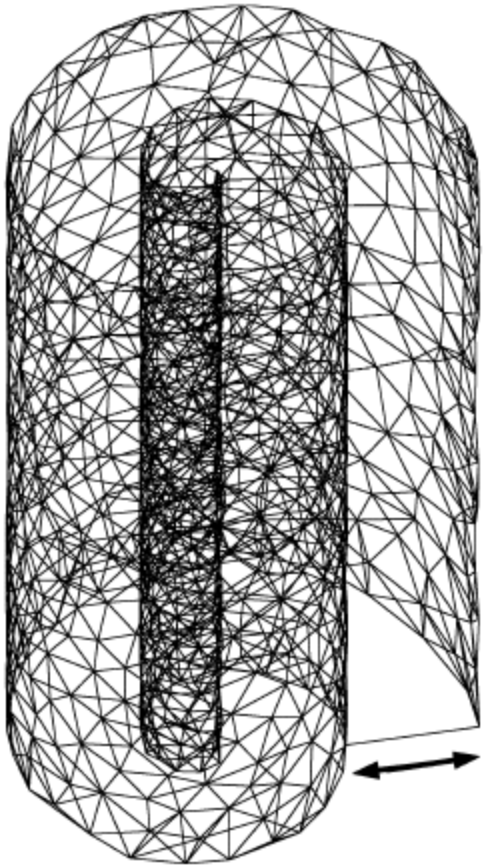
Downsides of Isomap: distances along the graph are only approximations of the true geodesic distances. They overestimate the distances, especially large distances, when data is sparse (less “waypoints” to choose from—>suboptimal path).

Overestimation of distances causes overstretching of faraway edges

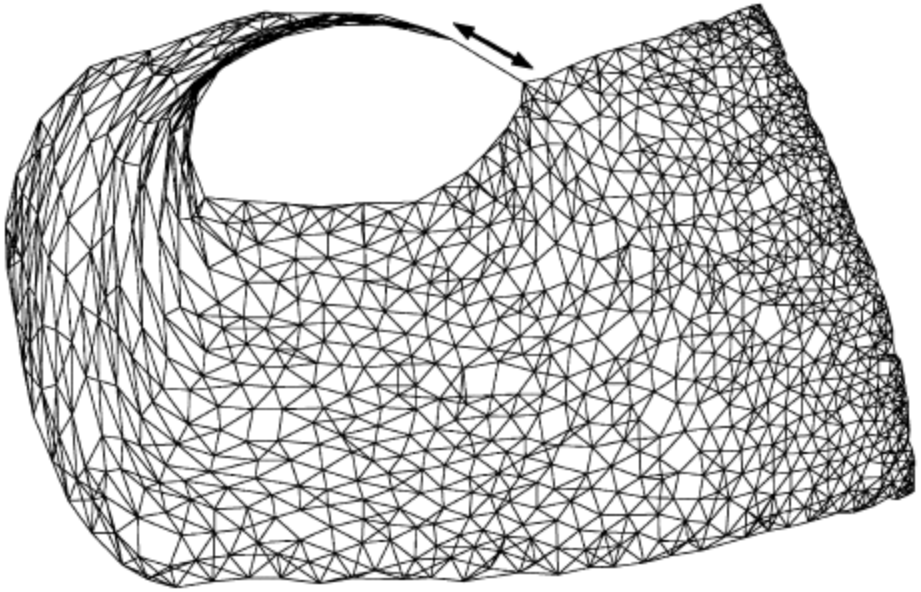


From: John Aldo Lee, Amaury Lendasse , Michel Verleysen. Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. Neurocomputing 57 (2004) 49 – 76.

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.

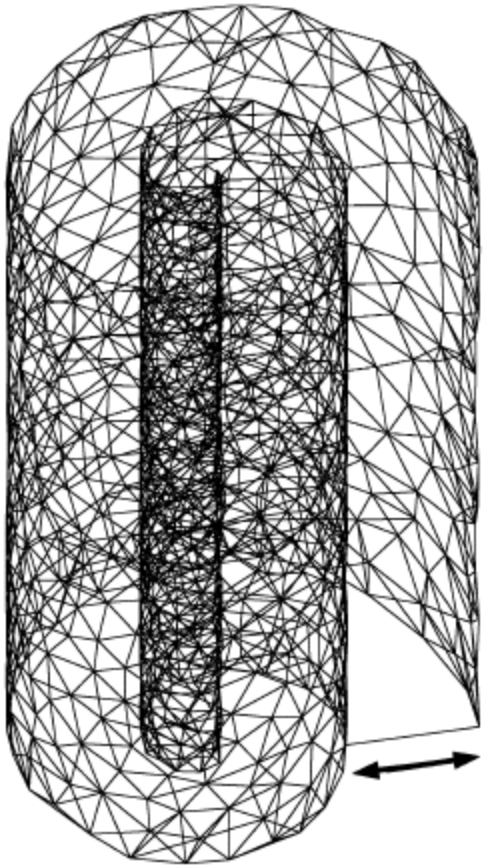
- *Curvilinear component analysis* (CCA) is like (absolute) MDS, but only short distances are taken into account.
- More formally, the cost function reads

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

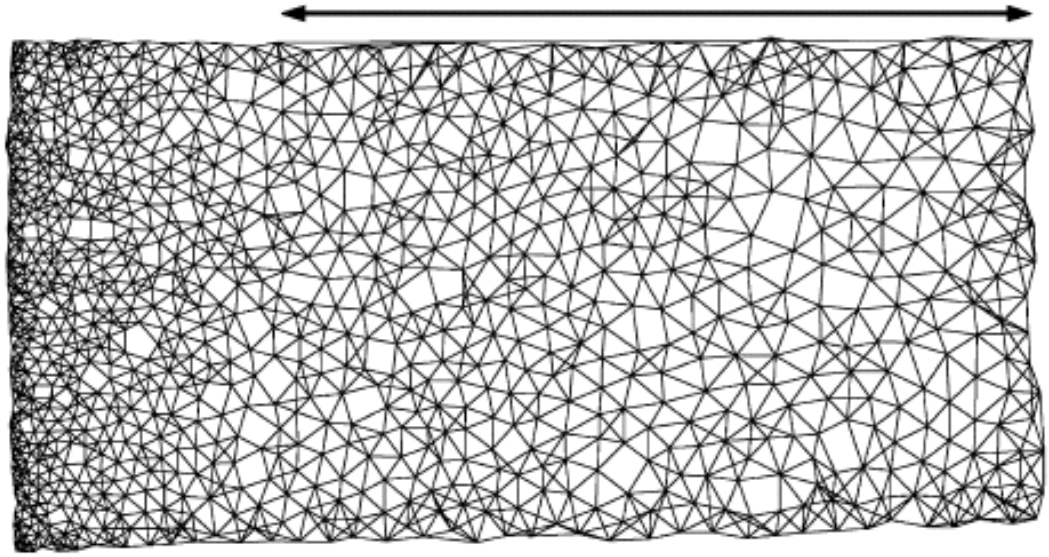
where $F(d,\lambda_y)$ equals unity, if $d<\lambda_y$, and zero otherwise; and d denotes the Euclidean distance of points in the original space (x) and in the projection (y), respectively. (Actually, $F(d,\lambda_y)$, could be any monotonically decreasing function in d.)

- 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

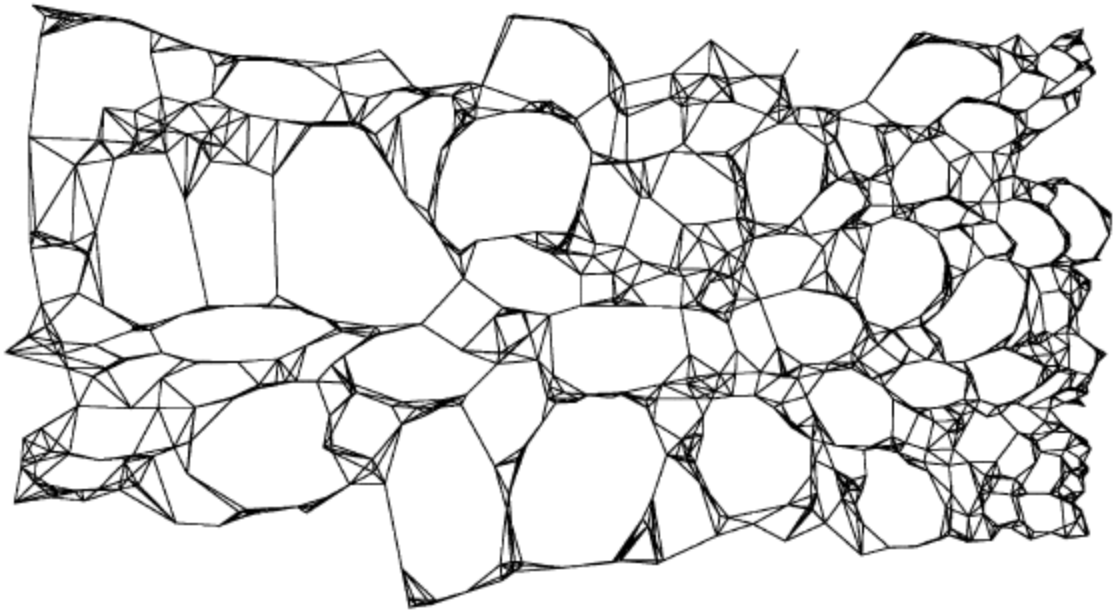


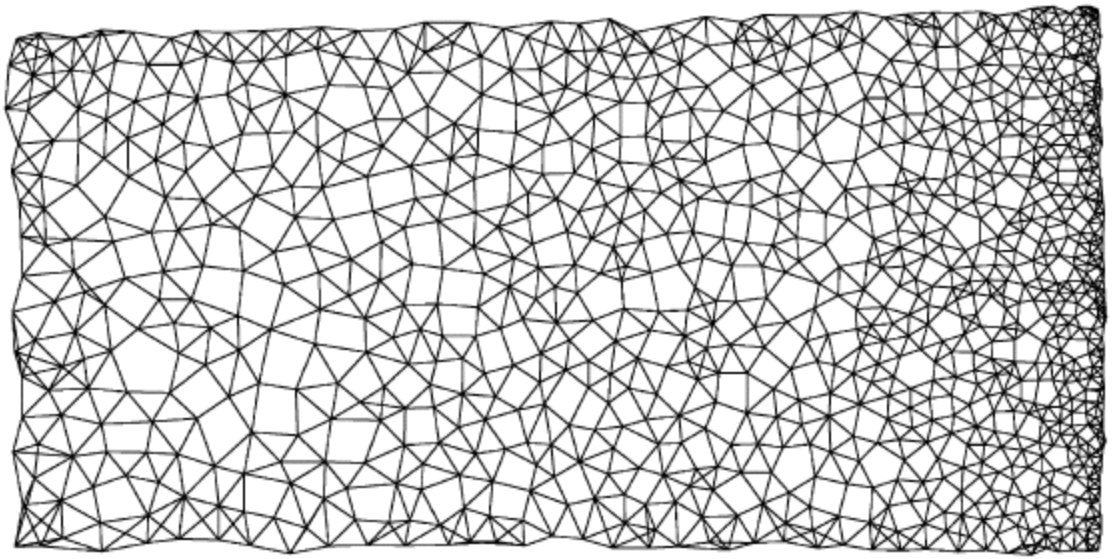
In CDA poorly approximated longer distances don't distort as much as in Isomap, since the embedding concentrated on small distances.



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)



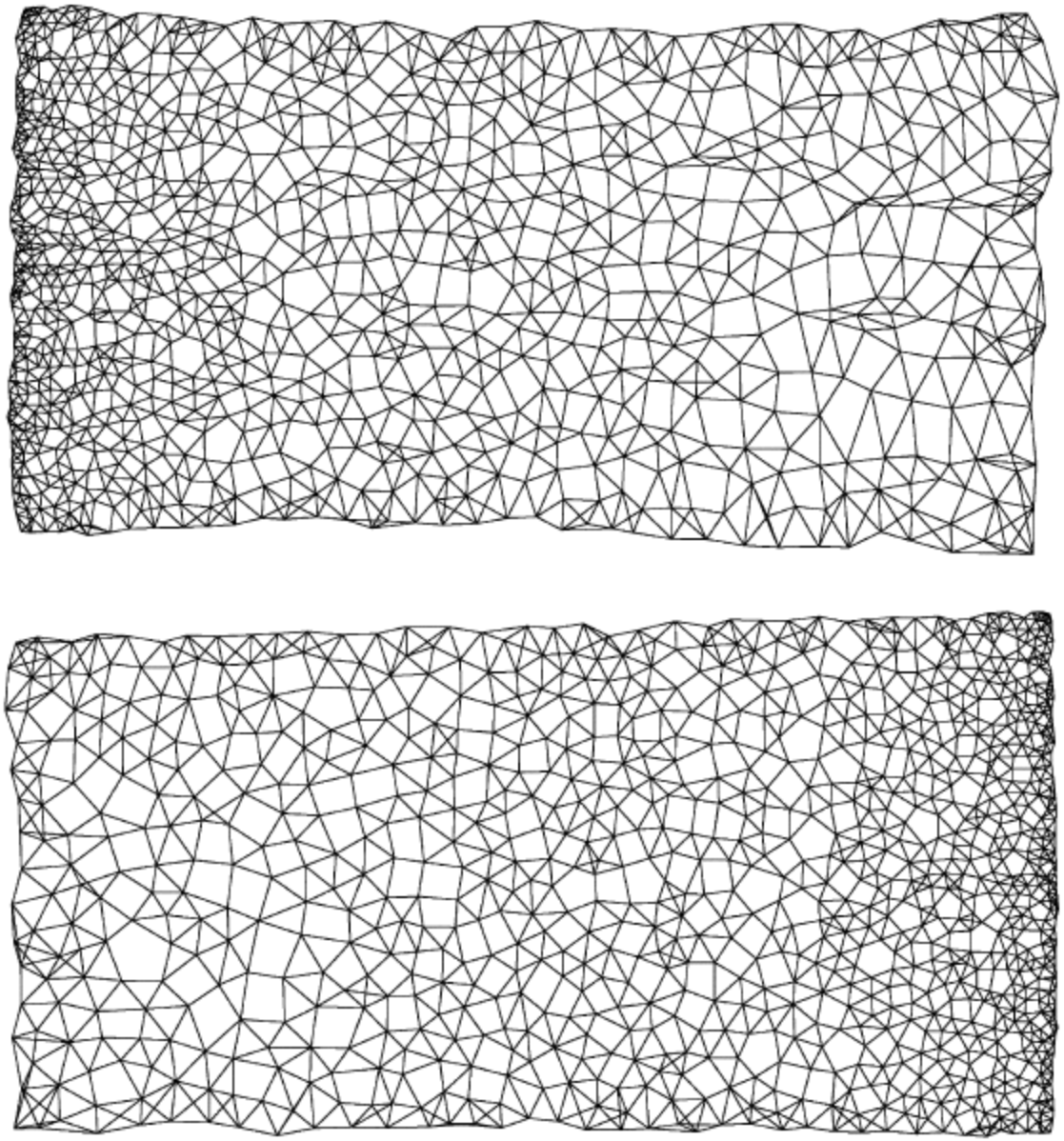


Some implementations of Isomap use a random sampling of a subset of points, to speed up computation. This can greatly distort distances (top).

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

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

Characteristics of an original point are simply its coordinates

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

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

$$\mathrm{error}(\mathrm{char}_{\mathcal{X}}(\mathbf{X}, \mathbf{x}^i), \mathrm{char}_{\mathcal{E}}(\mathbf{X}\boldsymbol{\Xi}, (\mathbf{x}^i, \boldsymbol{\xi}^i)))$$

$$= ||\mathbf{x}^i - g_V(\boldsymbol{\xi}^i)||^2 = ||\mathbf{x}^i - g_V(f_W(\mathbf{x}^i))||^2$$

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

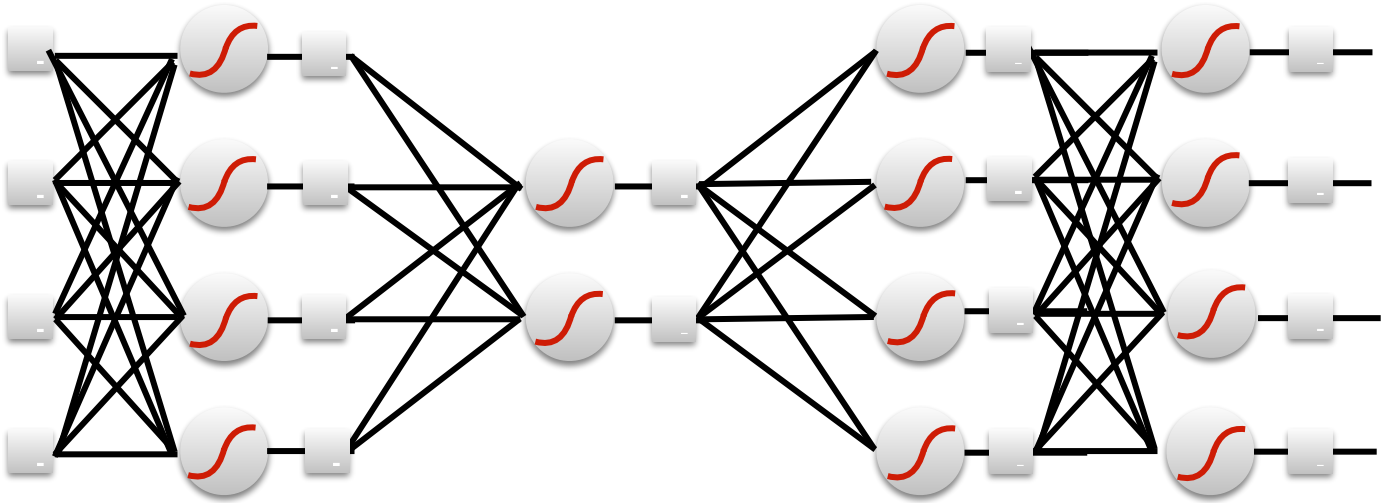
Autoencoders

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.



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.

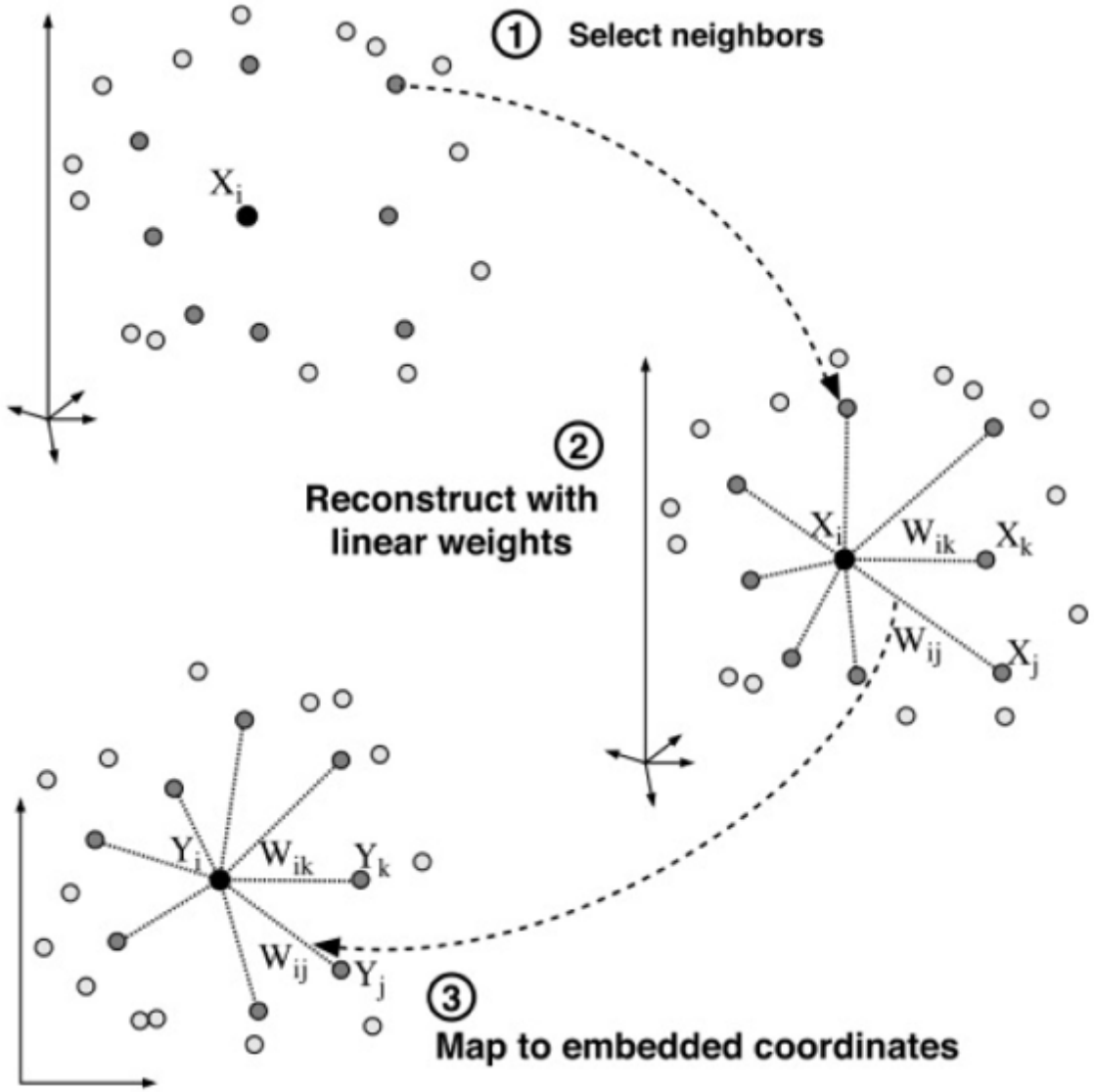
Idea: if the manifold of the data is *locally linear*, then each data point lies on the linear subspace spanned by its neighbors (or at least close to the subspace).

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:

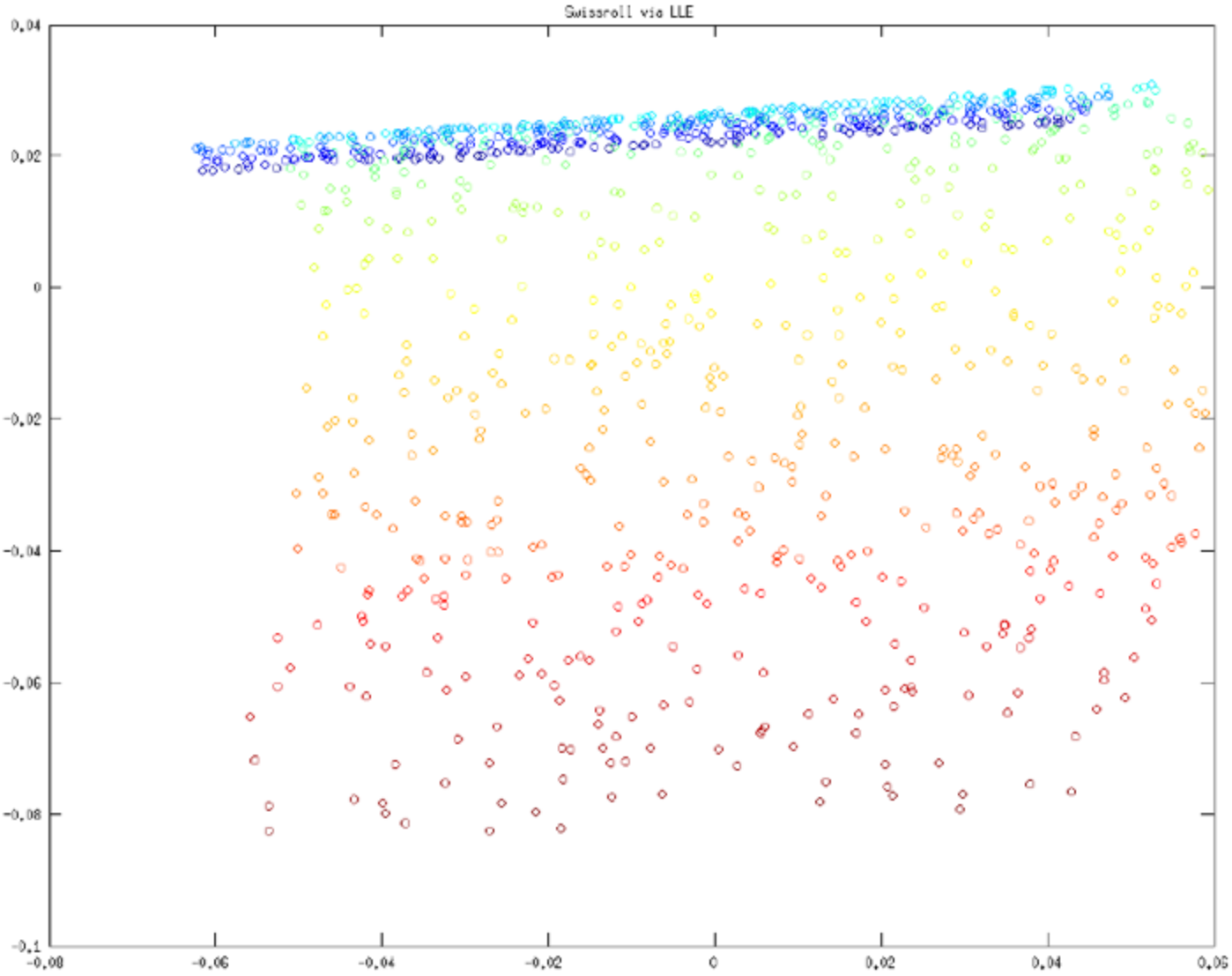
$$\mathrm{char}_{\mathcal{E}}(\mathbf{X}\Xi, (\mathbf{x}, \xi)) = (\xi - \sum_{\xi^j \in \mathcal{N}(\xi)} w_{ij}\xi^j)^2$$

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 $\Xi^\top \Xi = \mathbf{I}$, 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

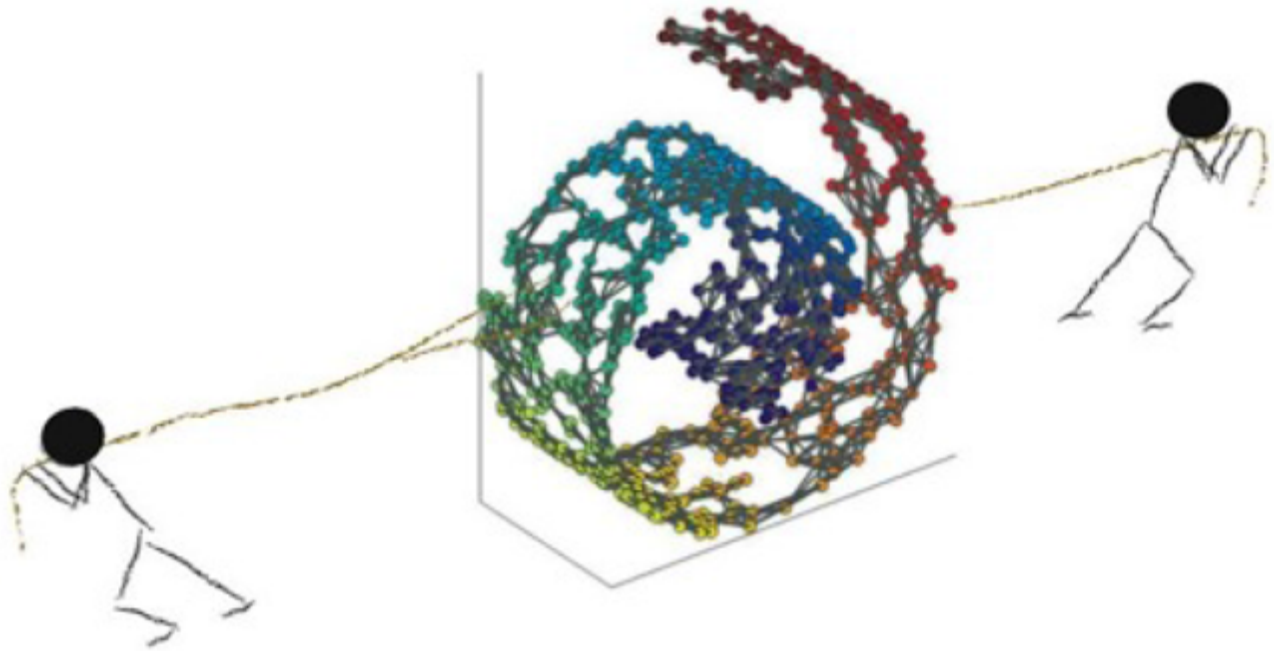
LLE (Roweis&Saul 2000): preservation of local topologies by reconstruction of data points by linear combination of its neighbors



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 $\sum_{ij} d_{\varepsilon}(\xi^i, \xi^j)$, 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 $\sum_{ij} d_{\mathcal{E}}(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j)$ while preserving the distances of neighboring points

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

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

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

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** $\eta_{ij} = 1$.

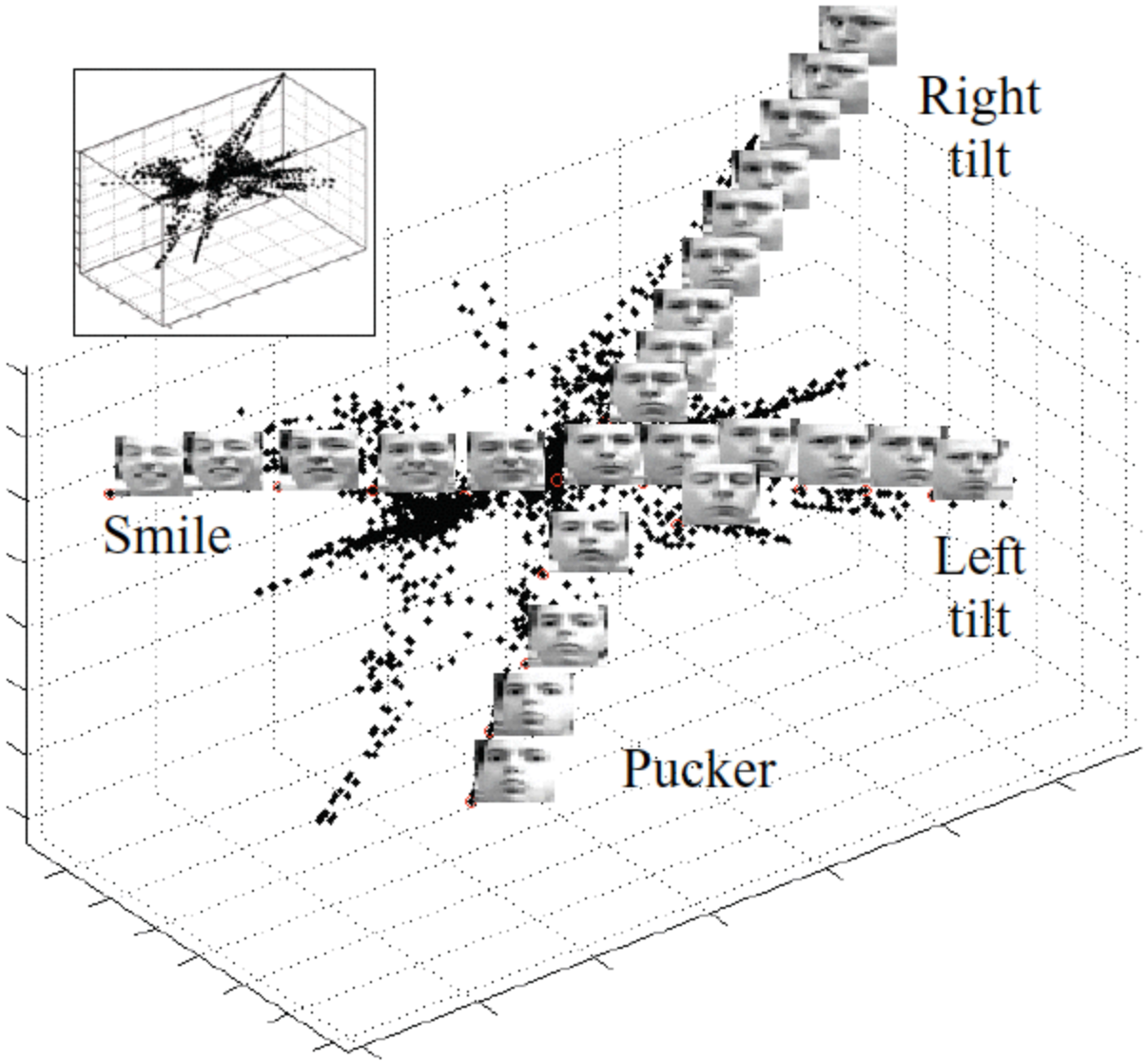
(2) $\Sigma_{ij} K_{ij} = 0$.

(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.

Neighbor embedding

Distance Preservation

Most dimensionality reduction methods discussed so far have been based on **preservation of distances**.

$$E_{\mathrm{mMDS}} = \frac{1}{a} \sum_{ij} w_{ij} \left(d_{\mathcal{X}}(\mathbf{x}^i, \mathbf{x}^j) - d_{\mathcal{E}}(\boldsymbol{\xi}^i, \boldsymbol{\xi}^j) \right)^2$$

Metric MDS stress function

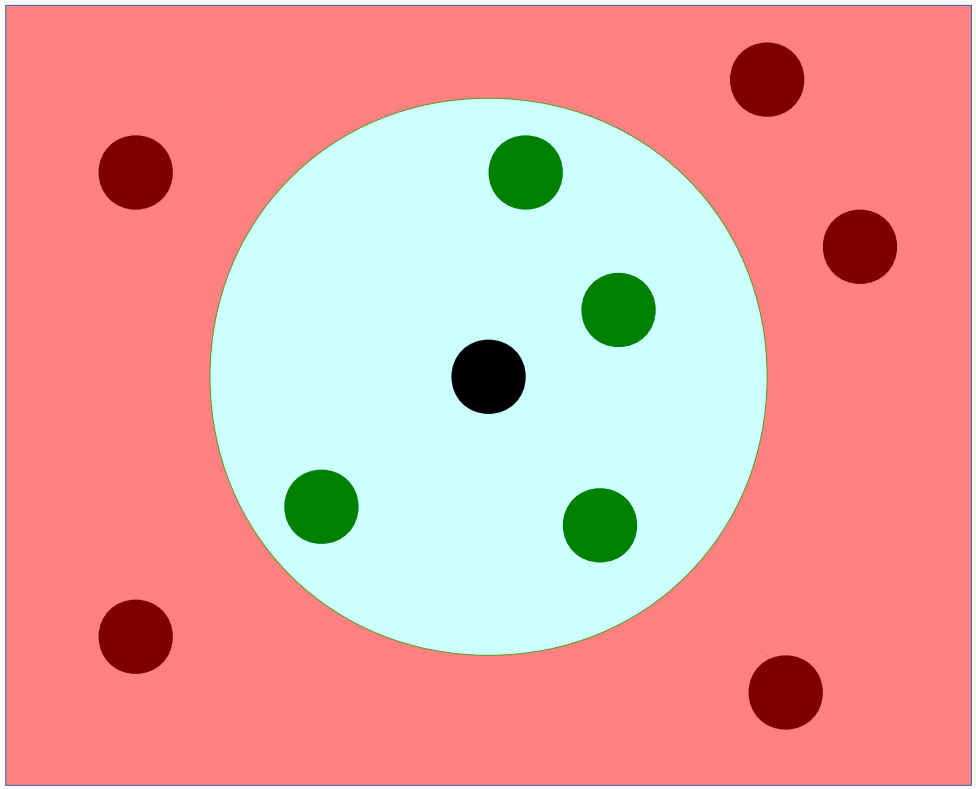
Other variants have modified which distances are most important to preserve: for example Sammon's mapping considers small distances the most important to preserve. But the aim is still to preserve distances.

Are distances the important thing to the analyst, if the aim is information visualization?

Alternative Idea: Preserve Neighborhoods

Neighbors are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.

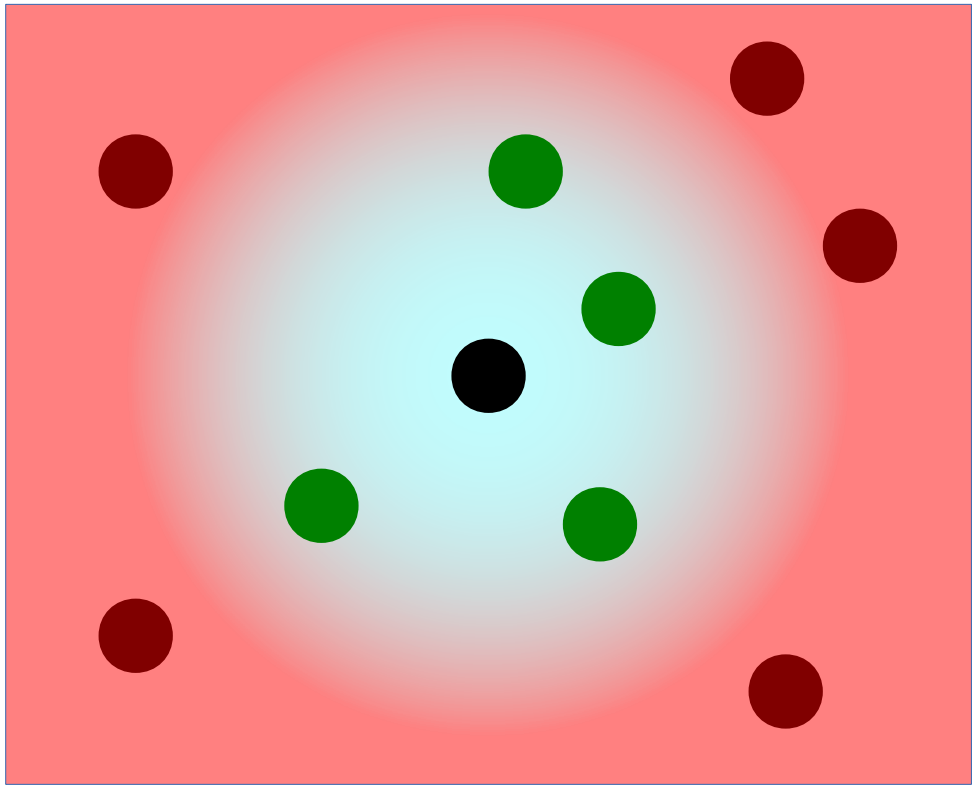


Hard neighborhood - each point is a neighbor or a non-neighbor

Alternative Idea: Preserve Neighborhoods

Neighbors are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.

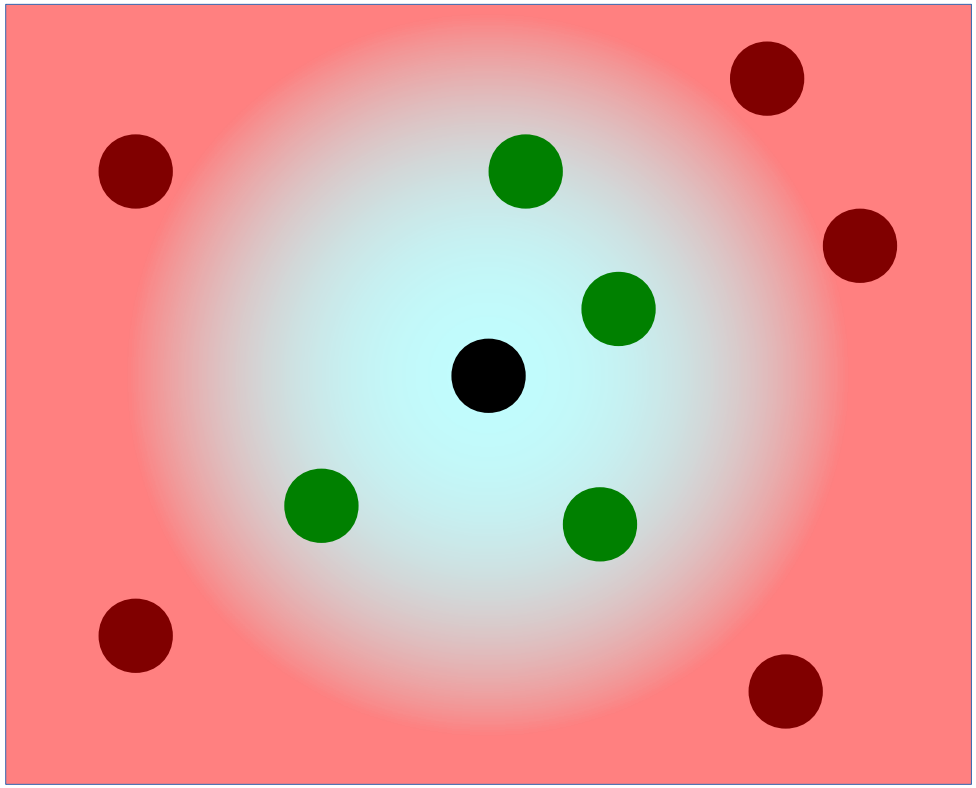


Soft neighborhood - each point is a neighbor with some weight and a non-neighbor with some weight

Alternative Idea: Preserve Neighborhoods

Neighbors are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.



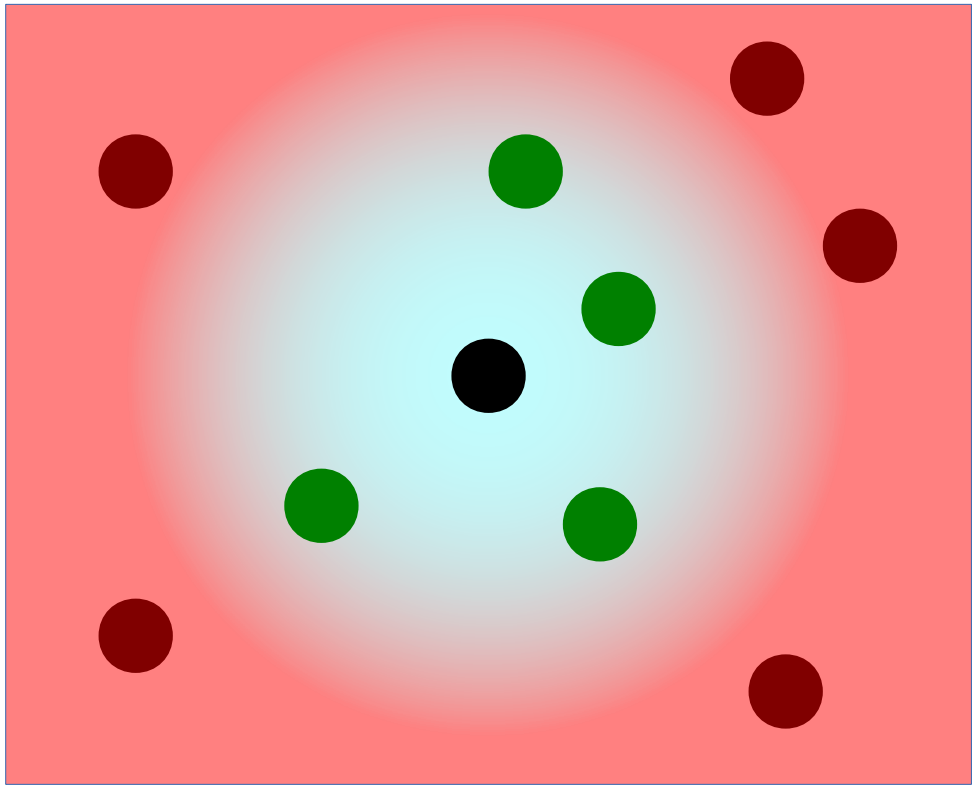
Soft neighborhood - each point is a neighbor with some weight and a non-neighbor with some weight

Some images and equations on the following slides are from the SNE paper (Roweis & Hinton '02).

Alternative Idea: Preserve Neighborhoods

Neighbors are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.



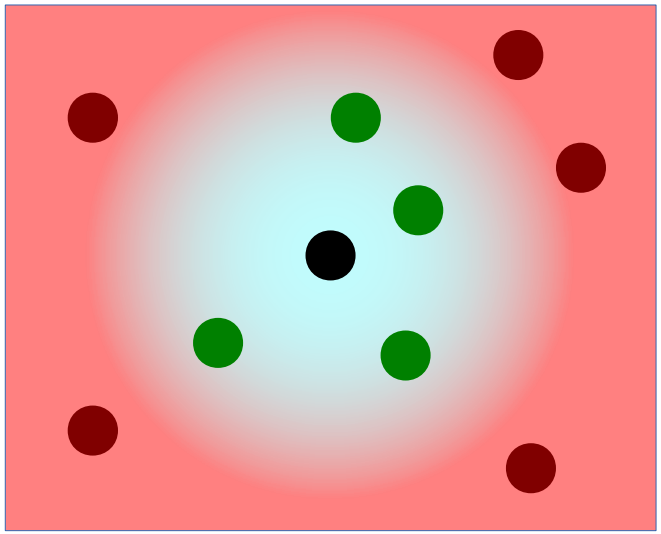
Probabilistic neighborhood

$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}$$

(probability to be picked as a neighbor **in input space**. Unlike in SOM, sums to 1)

Alternative Idea: Preserve Neighborhoods

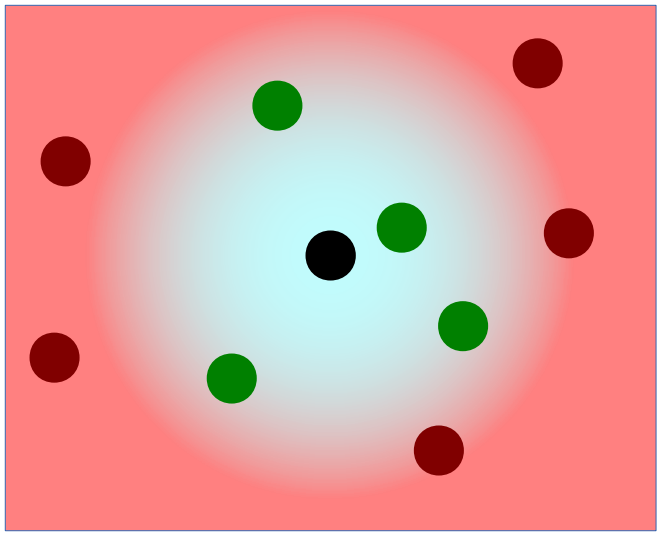
In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.



Probabilistic input neighborhood

$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}$$

(probability to be picked as a neighbor)



Probabilistic output neighborhood

$$q_{ij} = \frac{\exp(-||\mathbf{y}_i - \mathbf{y}_j||^2)}{\sum_{k \neq i} \exp(-||\mathbf{y}_i - \mathbf{y}_k||^2)}$$

(probability based on display coords.)

Stochastic Neighbor Embedding

Two probability distributions over a set of items can be compared by the **Kullback-Leibler (KL) divergence** = relative entropy = amount of surprise when encountering items from the 1st distribution when items were expected to come from the 2nd .

Use KL divergence to compare neighborhoods between the input and the output!

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} = \sum_i KL(P_i || Q_i)$$

KL divergence is nonnegative, and zero if and only if the distributions are equal. The value of the divergence sum depends on output coordinates, and can be minimized with respect to them. This is **Stochastic Neighbor Embedding**.

Stochastic Neighbor Embedding, details

How to set the size of the input neighborhood?
- controlled by a scale parameter in the input distance

$$d_{ij}^2 = \frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2}$$

scale parameter

- a scale parameter is hard to set without knowing a lot about the original data features
- it's beneficial to set the scale parameter differently for each individual point: in a sparsely populated region, neighborhood probability should maybe decrease less rapidly?

Idea: set an **"effective number of neighbors"**. In a uniform distribution over k neighbors, the entropy is log(k). Find the scale parameter value (by binary search in some interval) so that the entropy of p_{ij} becomes log(k) for a desired value of k.

This value becomes different around each point.

Stochastic Neighbor Embedding, details

Adjusting the output coordinates is done by gradient descent to minimize the sum of KL divergences. Start from a random initial output configuration, then iteratively take steps along the gradient.

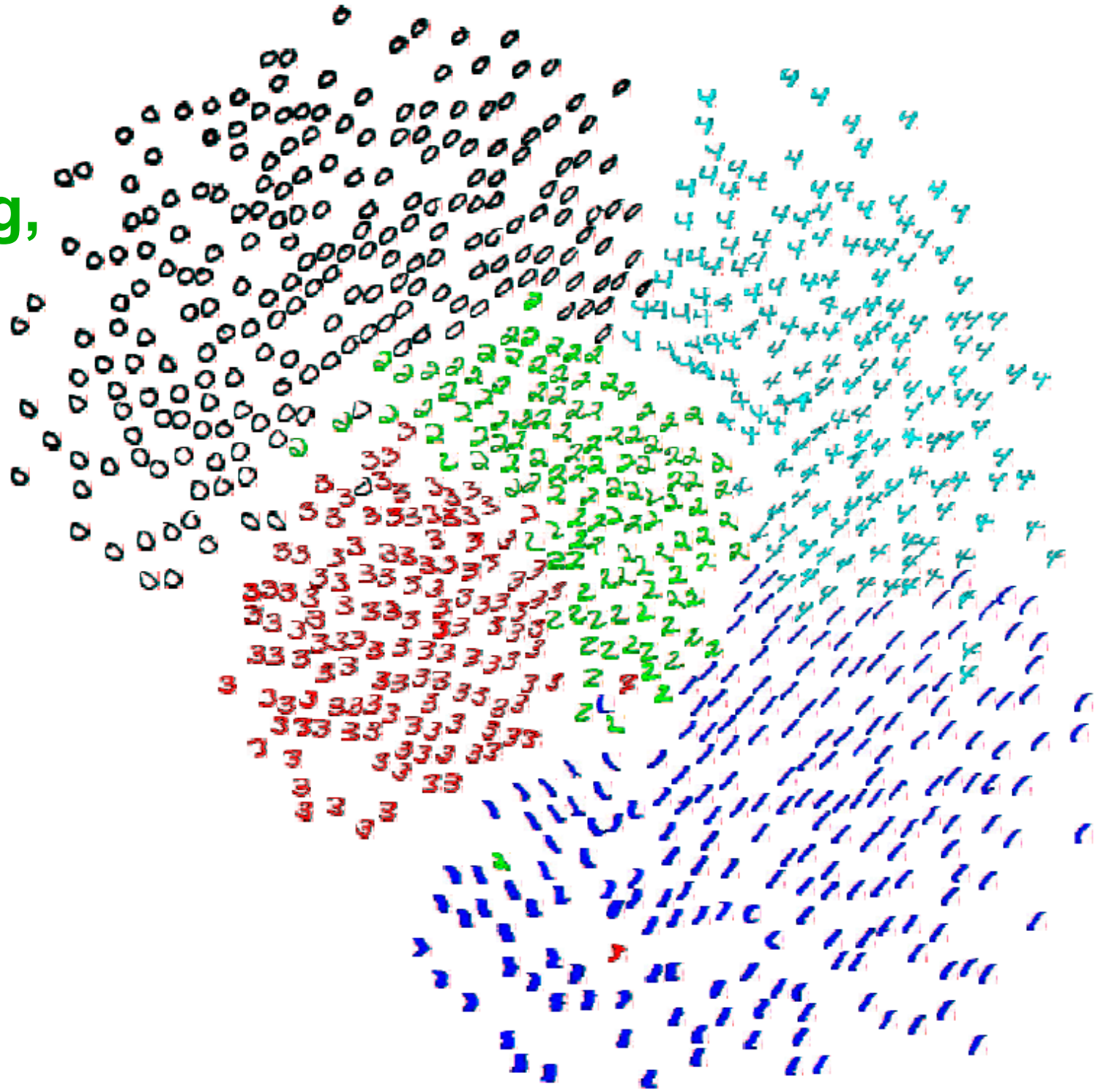
$$\frac{\partial C}{\partial \mathbf{y}_i} = 2 \sum_j (\mathbf{y}_i - \mathbf{y}_j)(p_{ij} - q_{ij} + p_{ji} - q_{ji})$$

Can be seen as “forces” pushing and pulling between pairs of points to make the input and output probabilities more similar.

Stochastic Neighbor Embedding, example

SNE applied to grayscale bitmap images of handwritten digits.

Features = pixel values.

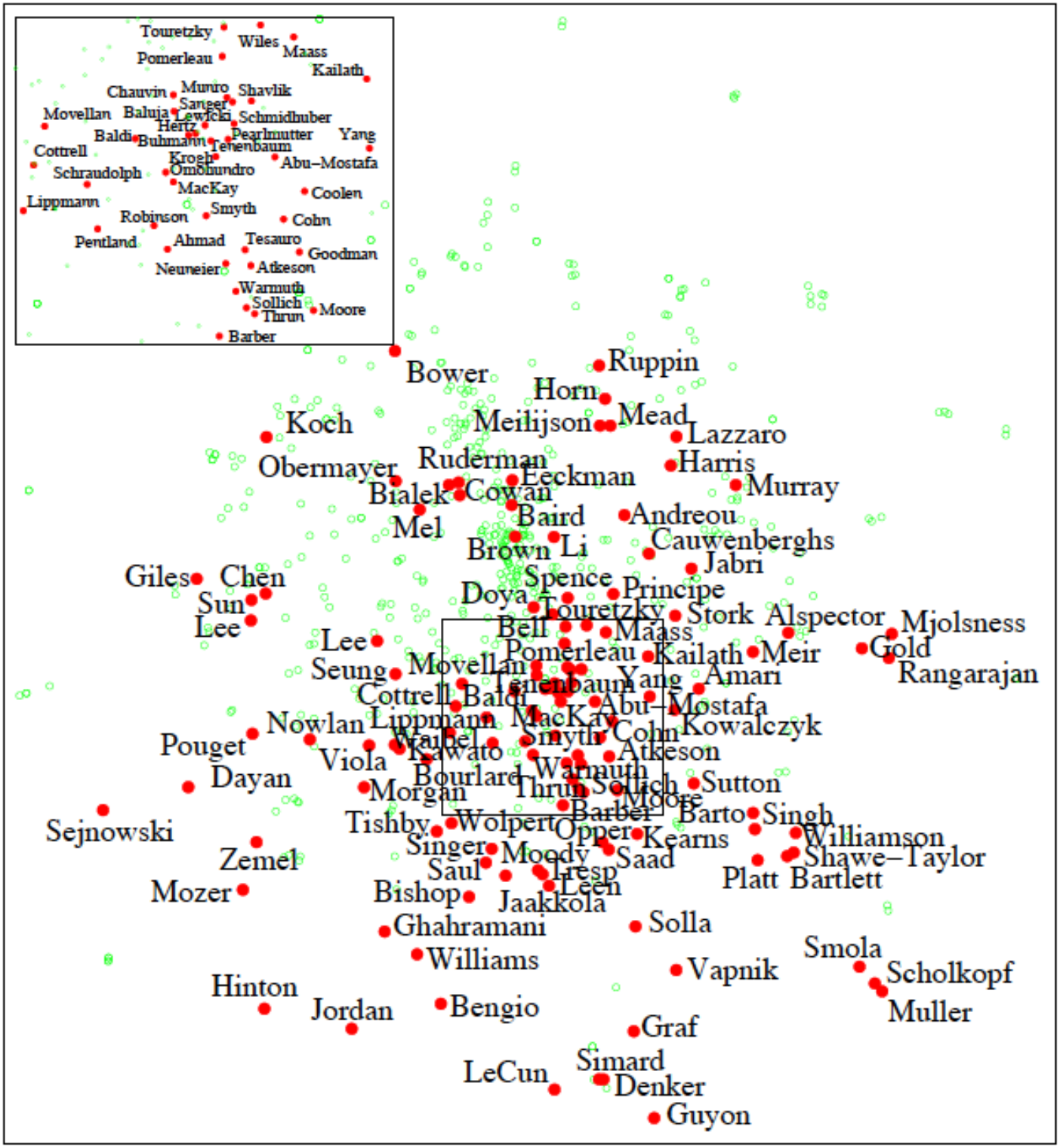


Stochastic Neighbor Embedding, NIPS authors

Authors of NIPS papers

Features= vectors of word counts

(how many of each word does an author have in his/her NIPS papers)



Stochastic Neighbor Embedding with multiple output locations for each data item?

An interesting alternative of SNE is possible, where each data item has **multiple locations on the display.**

$$q_{ij} = \sum_b \pi_{i_b} \sum_c \frac{\pi_{j_c} \exp(-||\mathbf{y}_{i_b} - \mathbf{y}_{j_c}||^2)}{\sum_k \sum_d \pi_{k_d} \exp(-||\mathbf{y}_{i_b} - \mathbf{y}_{k_d}||^2)}$$

Occupancy weight of i at b:th location

Perhaps useful if some relationships are hard to capture on-screen with just one location per item.

Crowding Problem

When the output dimensionality is smaller than the effective dimensionality of data on the input, the **neighborhoods are mismatched**:

- in a high-dimensional space, points can have many close-by neighbors in different directions. In a 2D space, you essentially have to arrange close-by neighbors in a circle around the central point, which constrains relationships among neighbors.
- in a high-dimensional space you can have many points that are equidistant from one another; in a 2D space at most 3 points are equidistant from each other.
- volume of a sphere scales as r^d in d dimensions

Some images and equations on the following slides are from the t-SNE paper (van der Maaten & Hinton '08).

Crowding Problem

-----> On a 2D display, there is much less area available at radius r than the corresponding volume in the original space.
-----> Arranging high-dimensional items into a 2D space can easily place items more far off than they should be, because “there is no room left nearby”.
------> In contrast, some items end up crowded in the center to stay close to all of the far-off points. This is the **crowding problem.**

t-distributed Stochastic Neighbor Embedding

Idea: avoid crowding phenomenon by using a more **heavy-tailed neighborhood distribution** in the low-dimensional output space than in the input space. Neighborhood probability falls off less rapidly ----> less need to push some points far-off and crowd remaining points close together in the center.

Use student-t distribution with 1 degree of freedom = Cauchy distribution = infinite mixture of Gaussians.

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

(Note these are joint probabilities, not conditional probabilities like in SNE)
Numerator approaches an inverse-square law--->joint probabilities for far-off points almost invariant to map scale

t-distributed Stochastic Neighbor Embedding

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)} \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

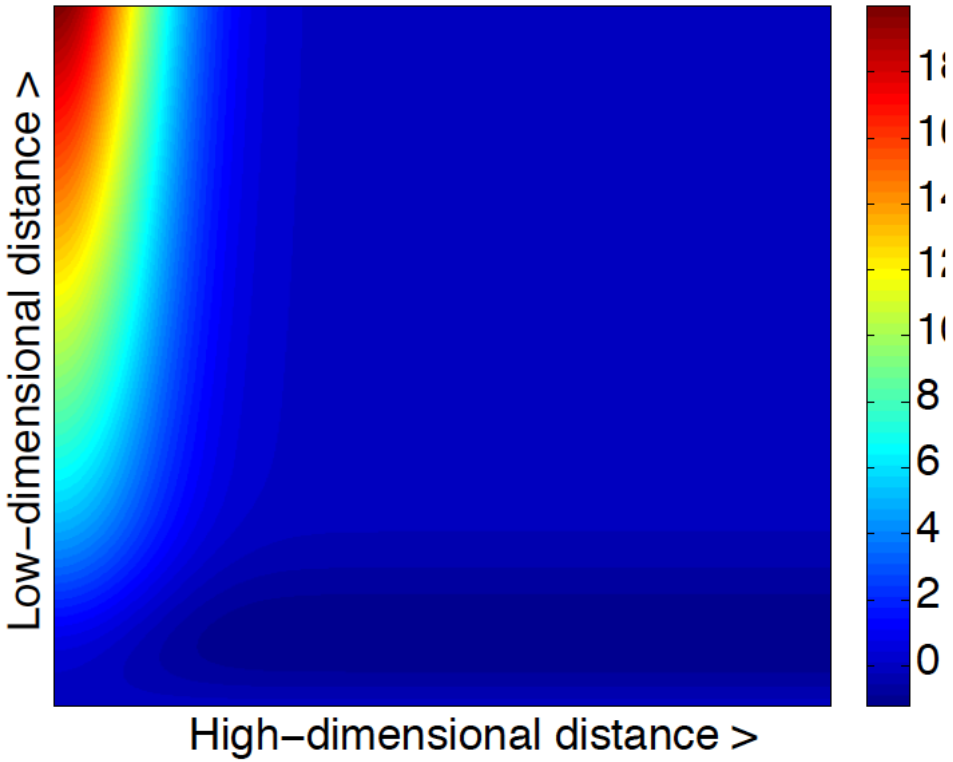
$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

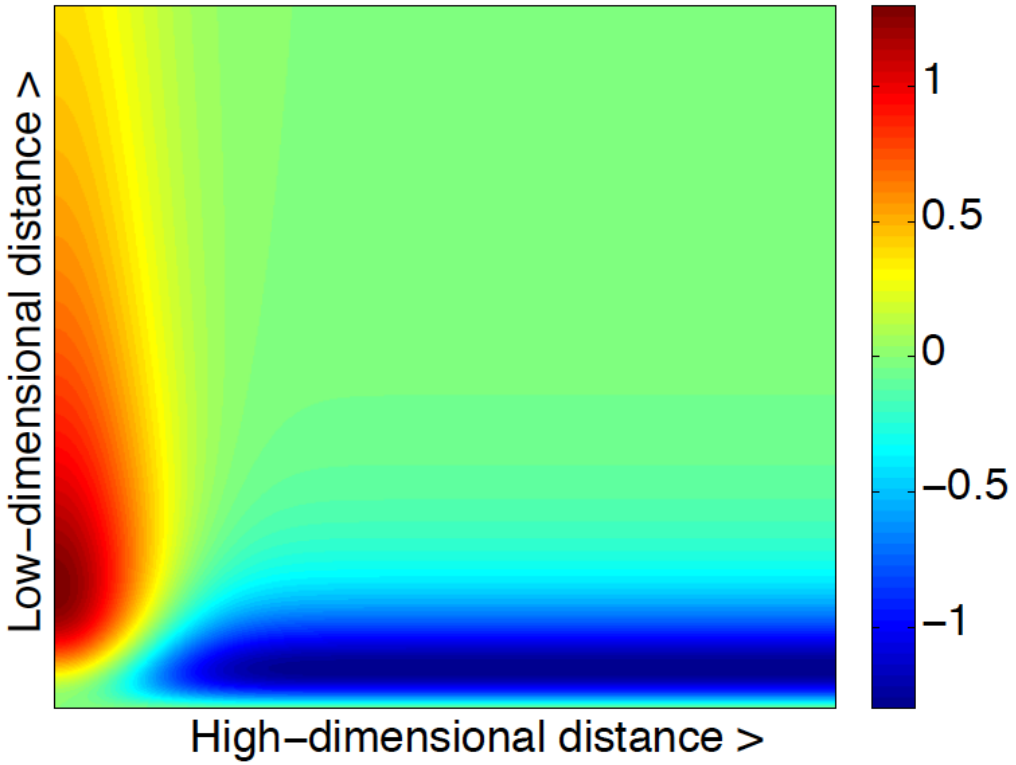
Minimize divergence between symmetric probabilities

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-distributed Stochastic Neighbor Embedding

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \left(1 + \|y_i - y_j\|^2\right)^{-1}$$



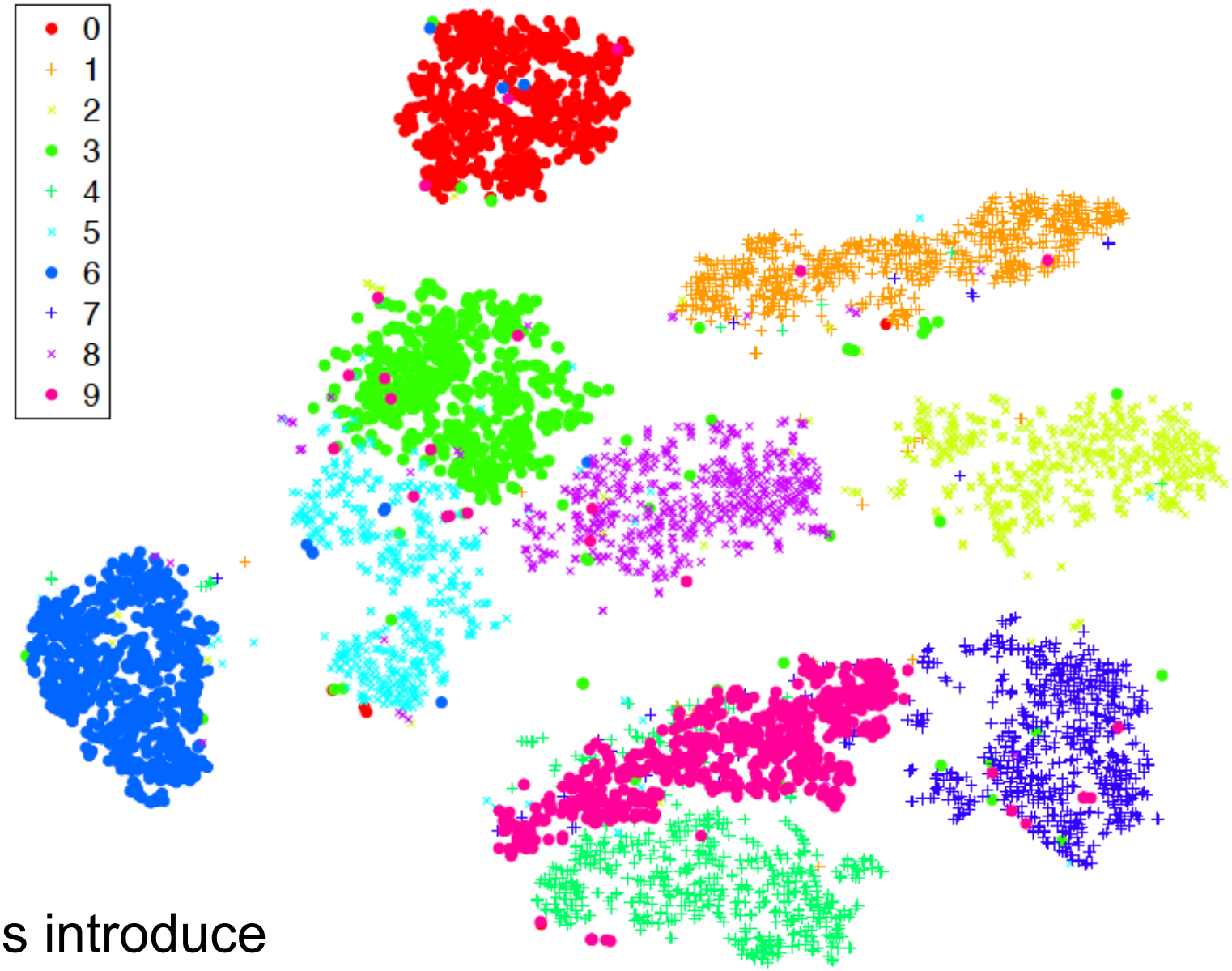


SNE: large gradient when high-dim distance is small and low-dim distance is large

t-SNE: positive gradient when high-dim dist. is small, low-dim is high, negative when other way around

t-distributed Stochastic Neighbor Embedding

t-SNE on digit data



Personal observation:
can sometimes introduce
“phantom subclusters”
where no subcluster-division exists

t-distributed Stochastic Neighbor Embedding with output locations in multiple maps

An interesting alternative of t-SNE is possible, where each data item has is **projected to multiple displays.**

$$q_{ij} = \frac{\sum_m \pi_i^{(m)} \pi_j^{(m)} (1 + \|\mathbf{y}_i^{(m)} - \mathbf{y}_j^{(m)}\|^2)^{-1}}{\sum_k \sum_{l \neq k} \sum_{m'} \pi_k^{(m')} \pi_l^{(m')} (1 + \|\mathbf{y}_k^{(m')} - \mathbf{y}_l^{(m')}\|^2)^{-1}}$$

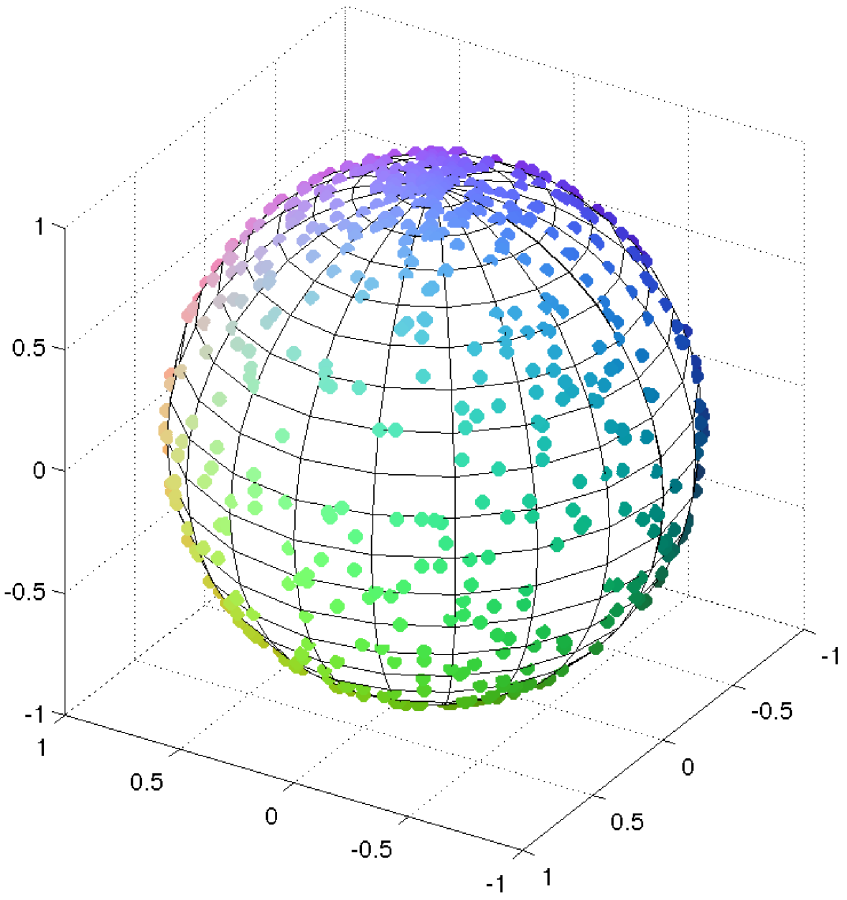
Occupancy weight of i at k:th display

Claimed useful for 1) visualizing original data similarities that do not satisfy the triangle inequality, 2) visualizing objects with high centrality (they are the closest neighbors of a lot of other points)

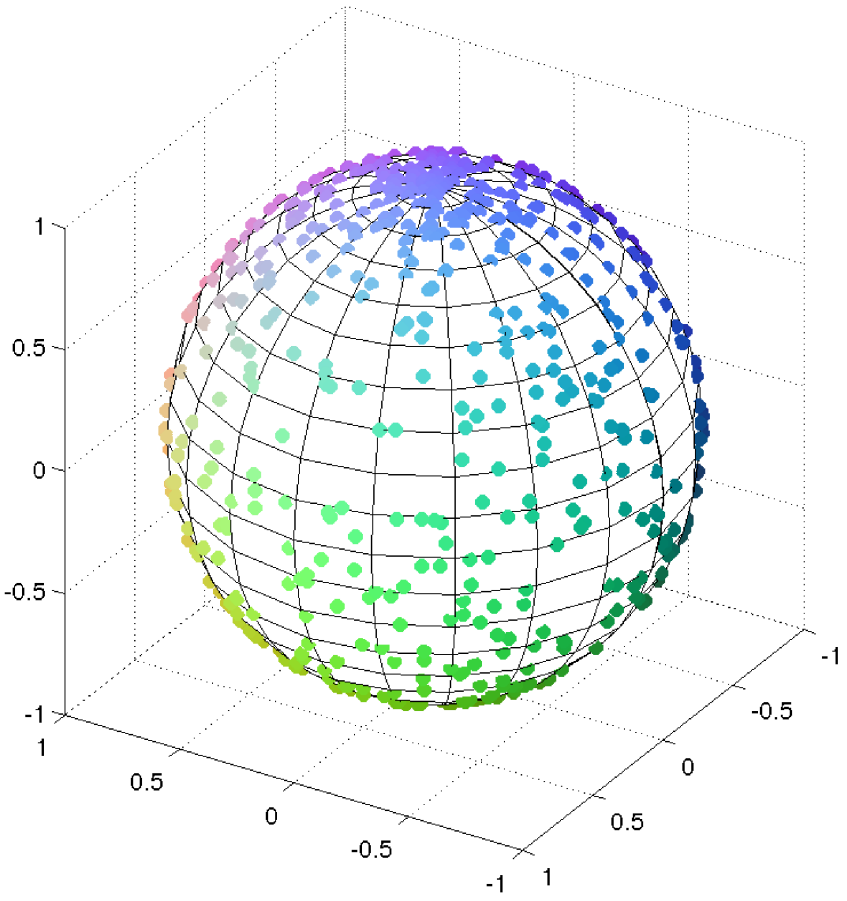
Preservation of Neighborhoods for a Task?

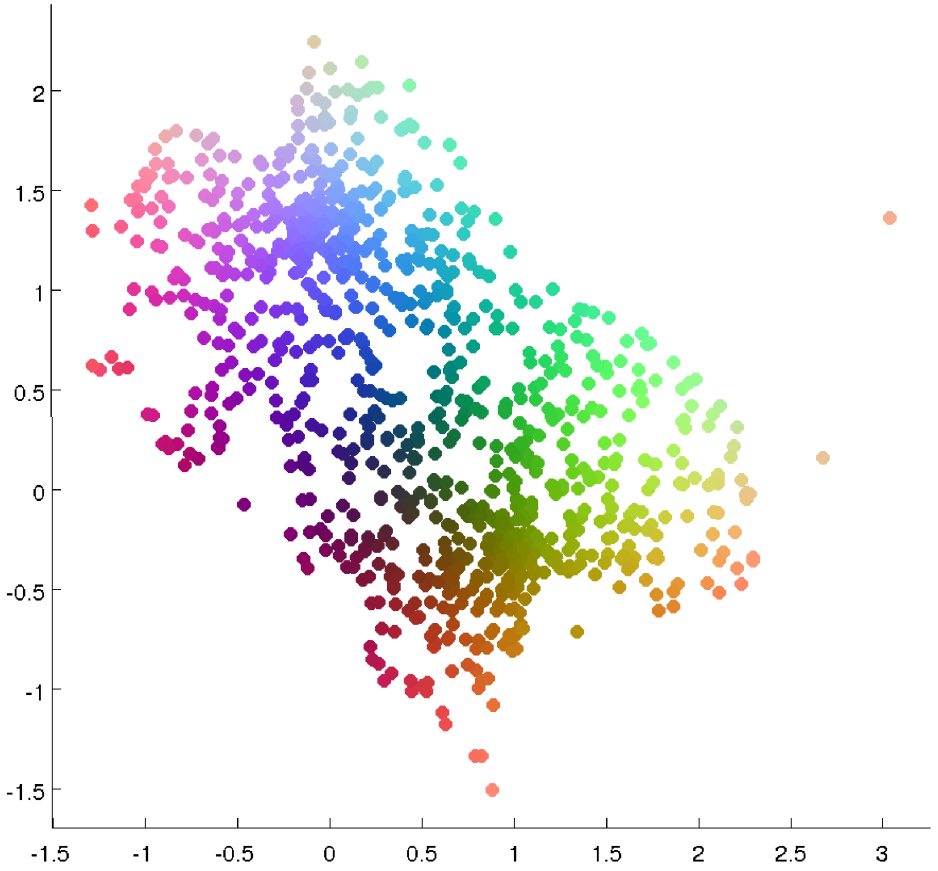
Analyzing local neighborhoods of data items might be a subtask that some analysts perform to gain higher-level insight: for example, high-level structure of a graph (hubs, outlier areas) is built out of the set of local neighborhoods.

It turns out that preservation of neighborhoods can be formulated as **optimization of an information retrieval task**.

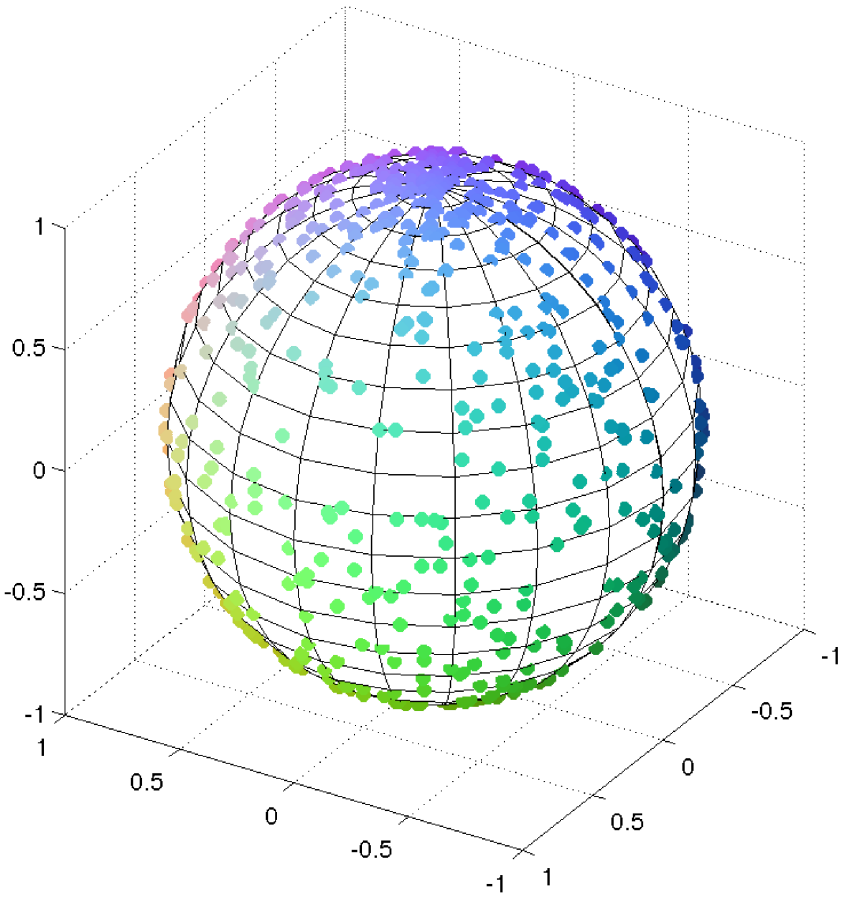


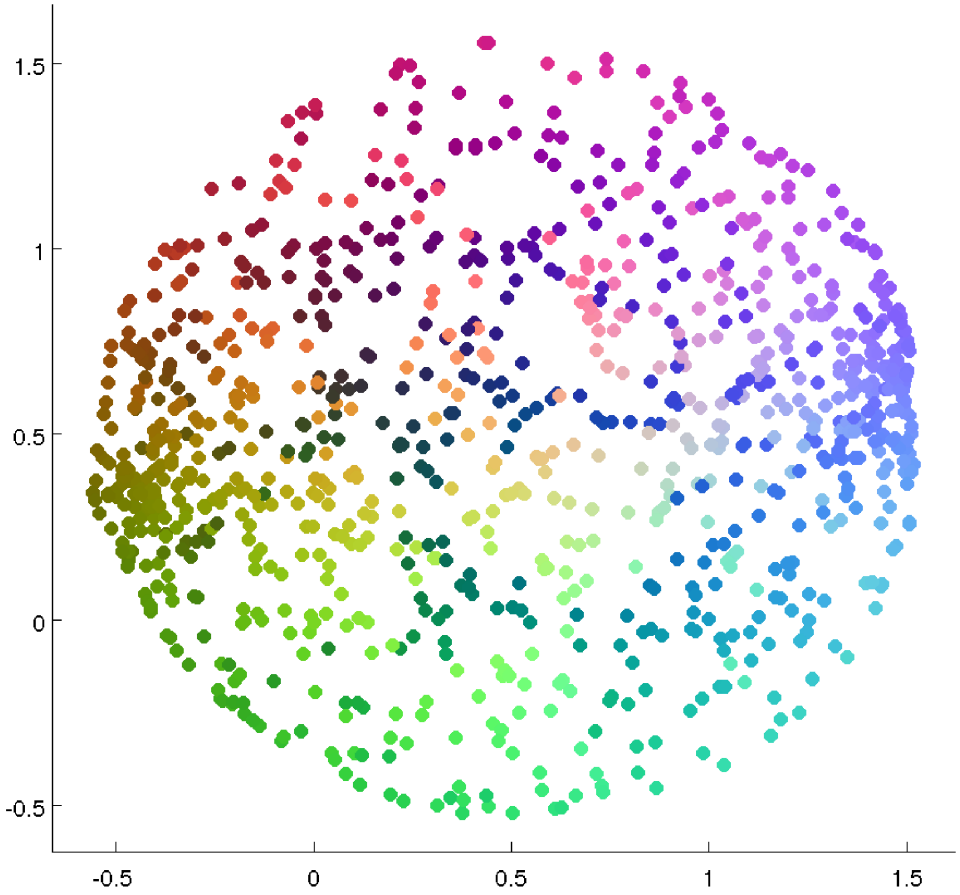
Example data set



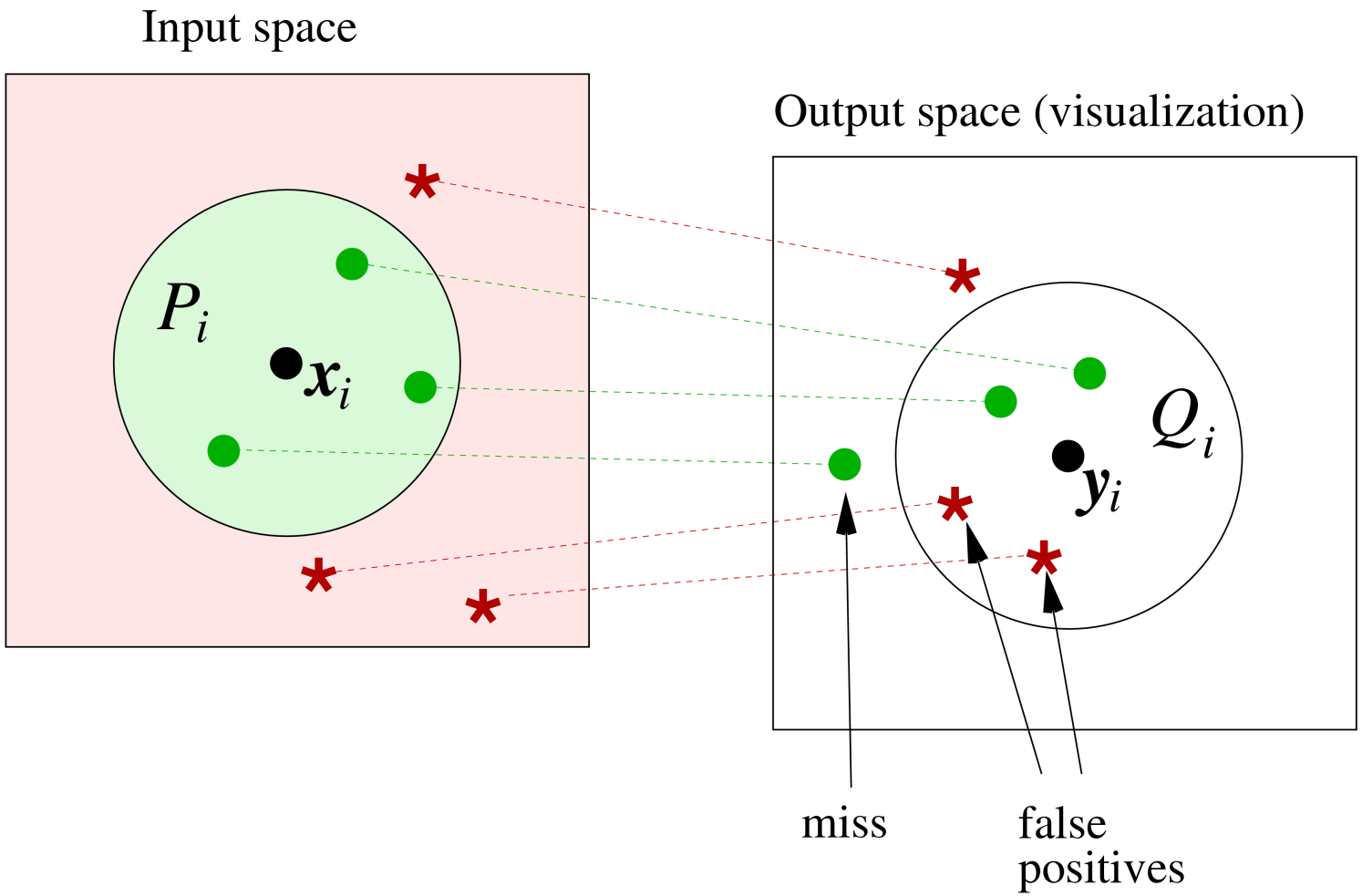


“Orange-peel map”

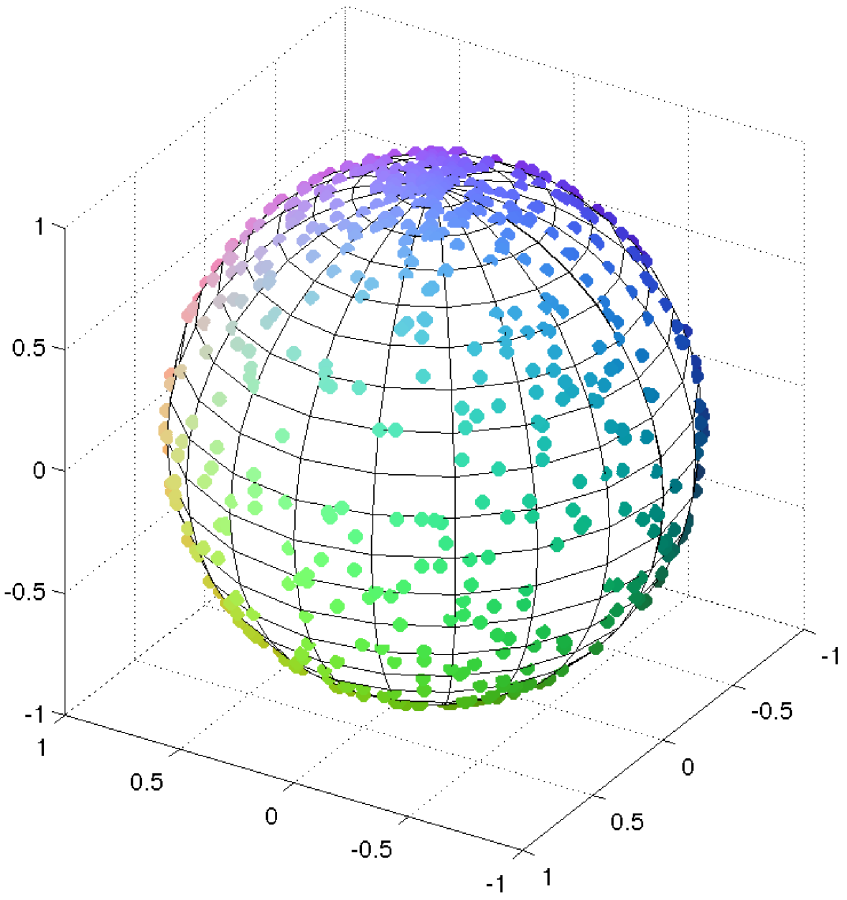




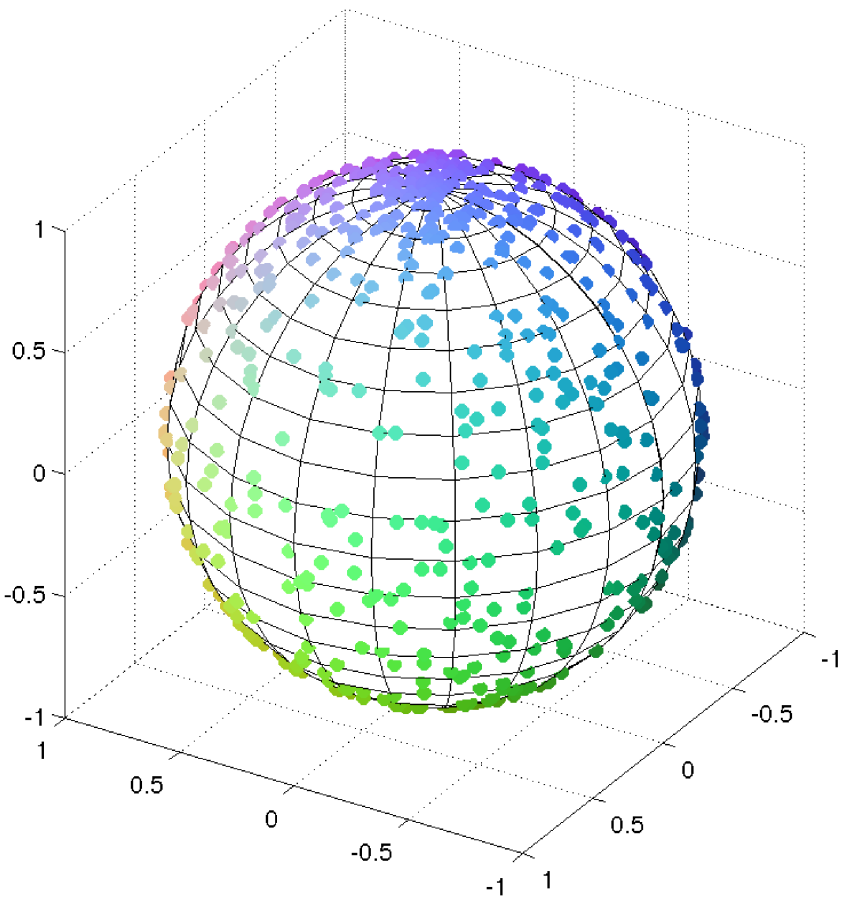
“Squashed-flat sphere”

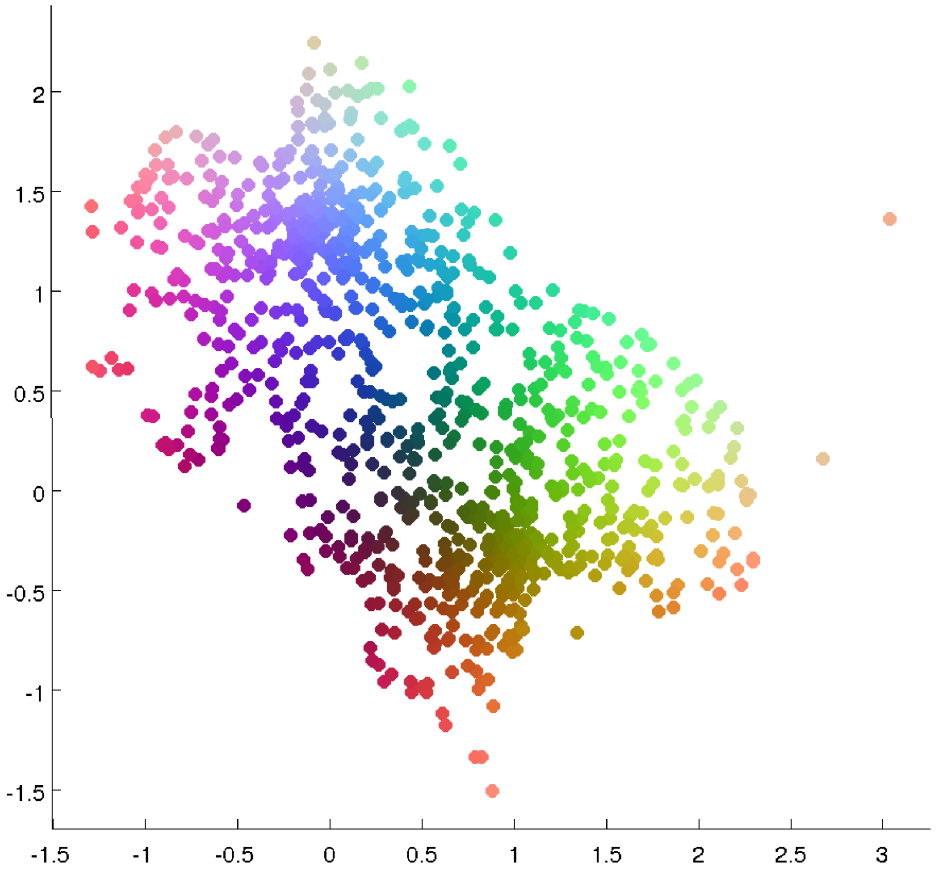


Minimize errors for best information retrieval.

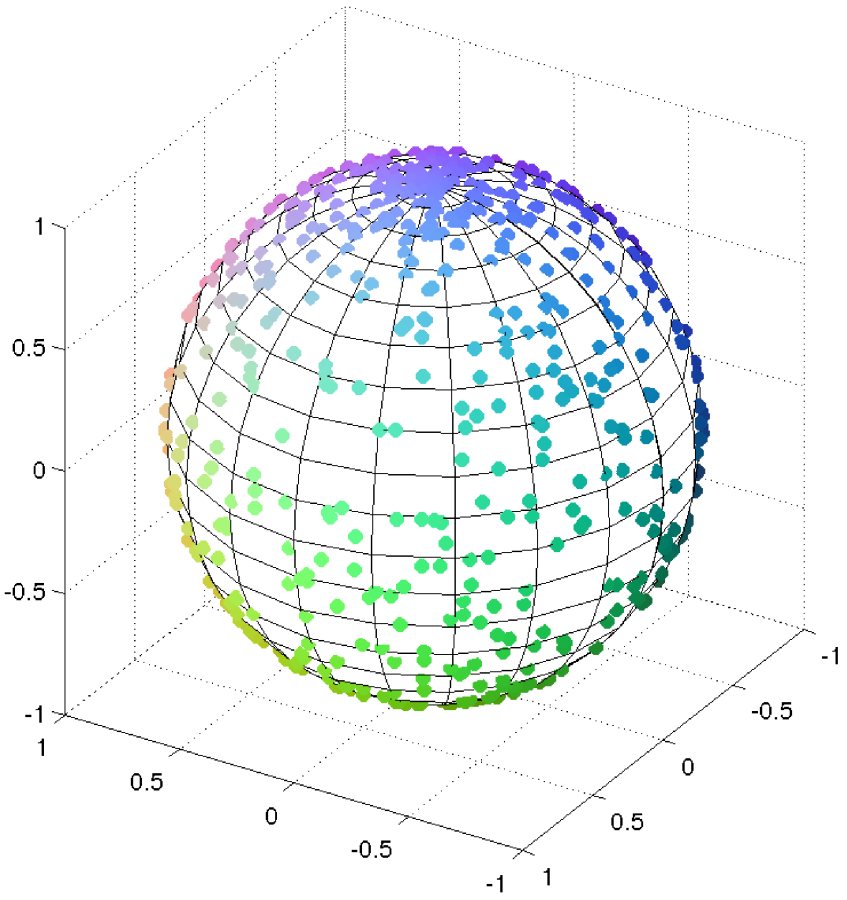


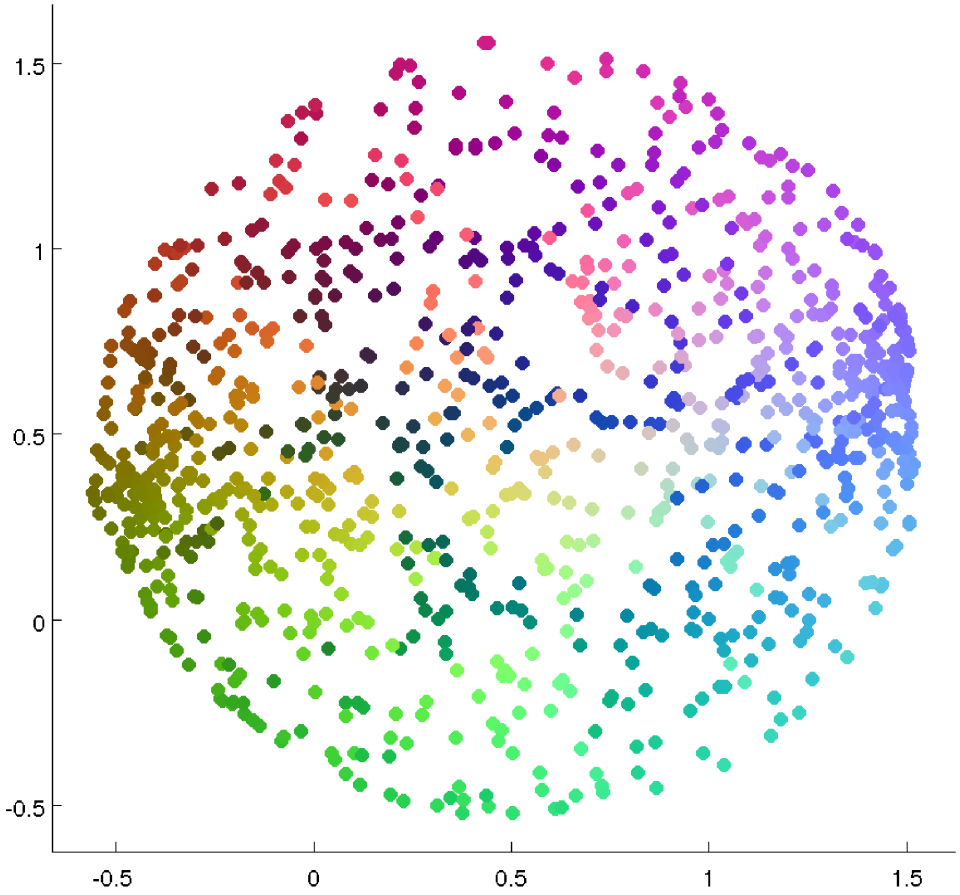
Example data set





Embedding mininizes false positives
(falsely retrieved neighbors)





Embedding mininizes misses (neighbors that were not retrieved)

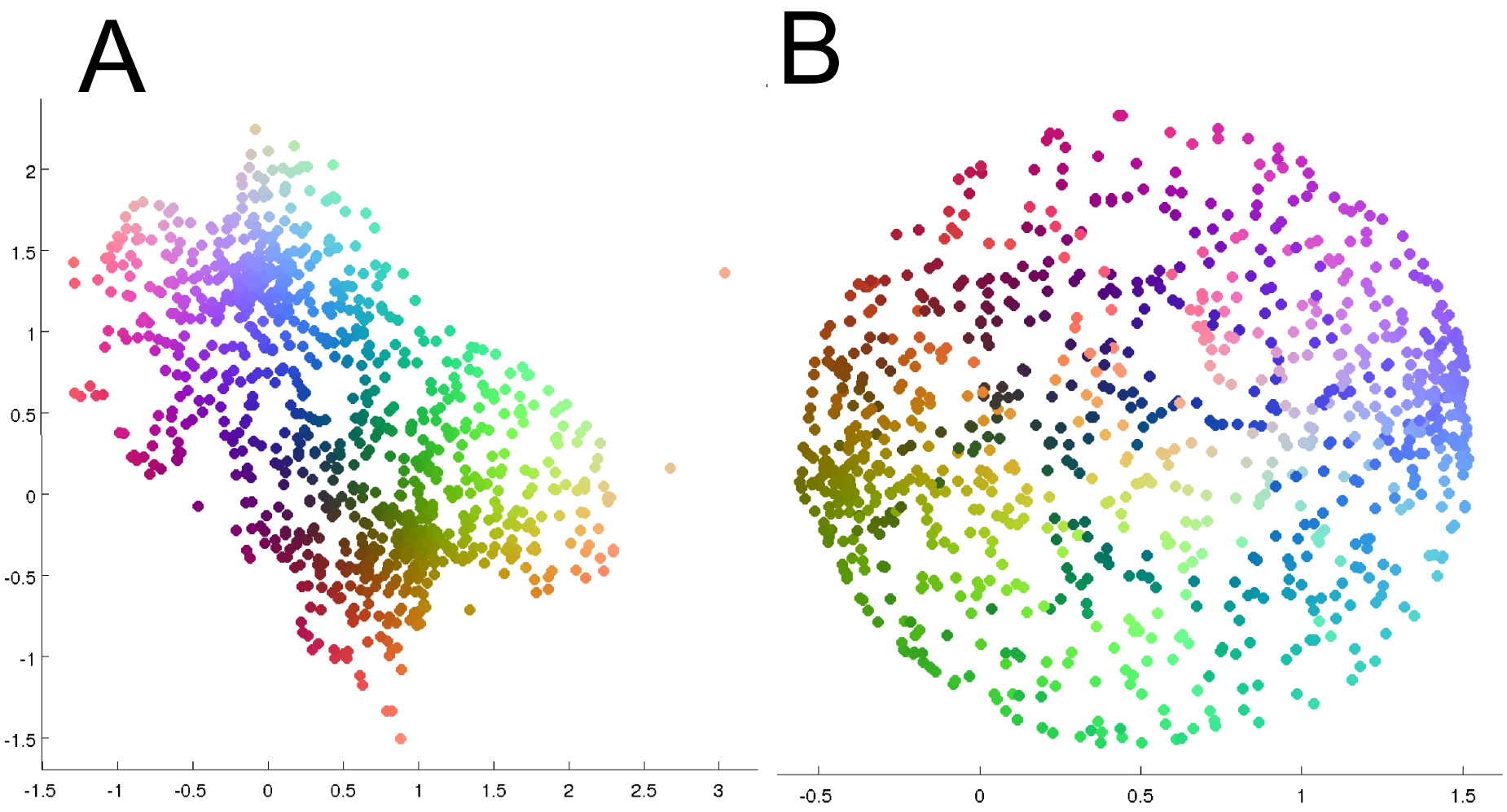
$$1 - precision = \frac{P_i^C \cap Q_i}{|Q_i|}$$

Proportion of false positives

$$1 - recall = \frac{Q_i^C \cap P_i}{|P_i|}$$

Proportion of missed neighbors

A visualization must make a **tradeoff** between false positives and misses. All methods end up with some tradeoff. A good visualization method should allow the user to specify the desired tradeoff.

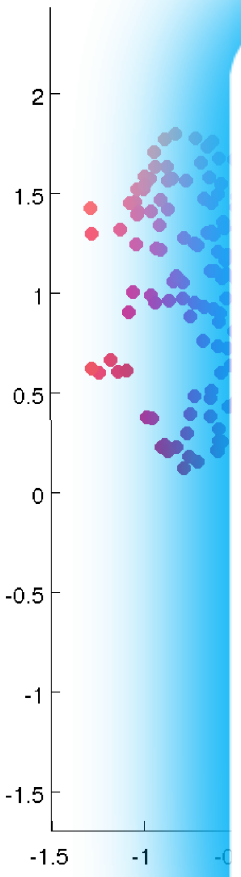


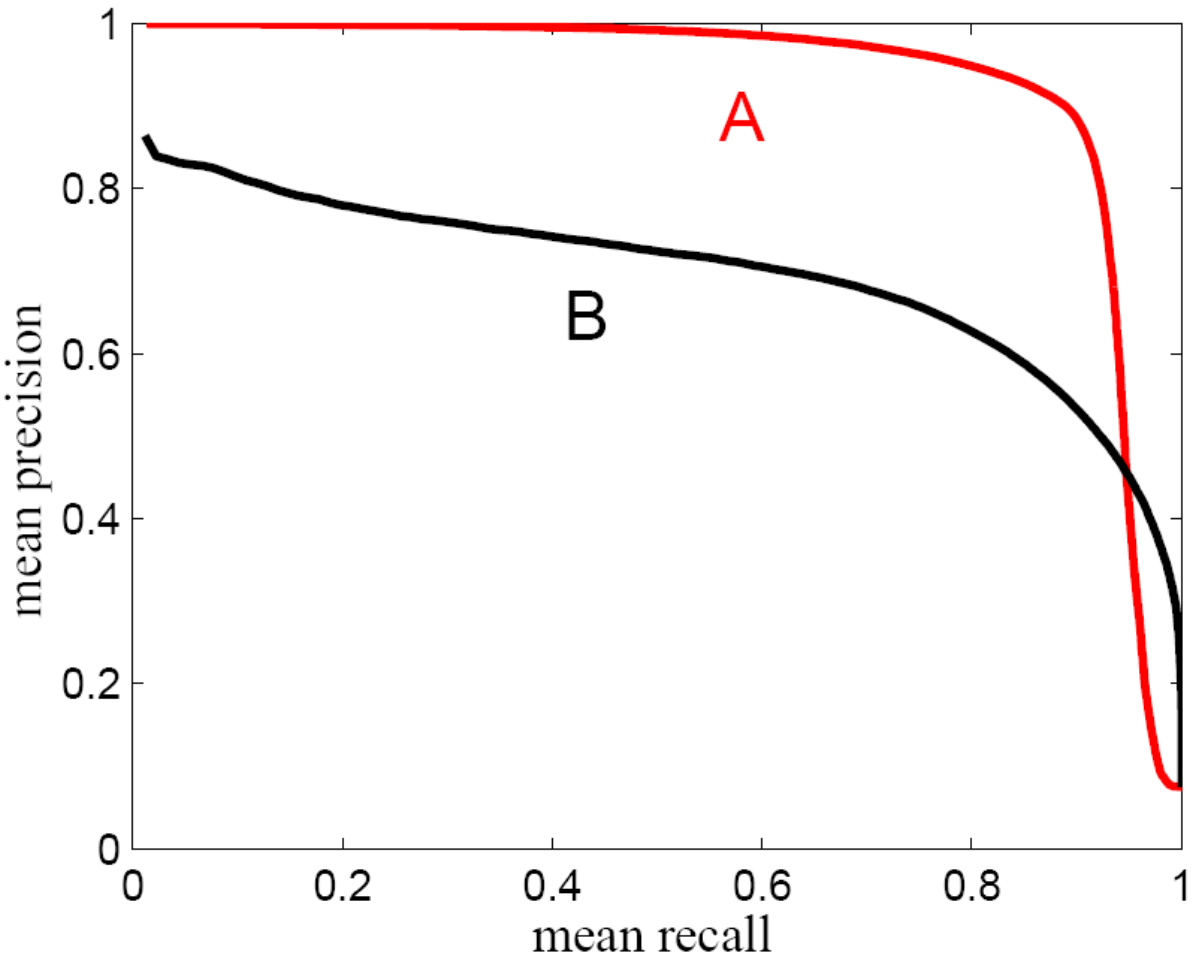
Embedding minimizes false positives (falsely retrieved neighbors)

Embedding minimizes misses (neighbors that were not retrieved)

A

B





Embe

(falsely retrieved neighbors)

that were not retrieved)

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Recall:

$$\sum_i \sum_{j\neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Proof of connection in Venna et al. 2010: assume some probabilities are uniformly-large, some are uniformly-small. Then there are 4 different kinds of terms in the sum. Show that above KL divergence is dominated by a cost that is proportional to a constant times number of misses.

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Precision:

$$\sum_i \sum_{j\neq i} q_{j|i} \log \frac{q_{j|i}}{p_{j|i}}$$

Proof of connection is similar to recall: assume some probabilities are uniformly-large, some are uniformly-small. Then there are 4 different kinds of terms in the sum. Show that above KL divergence is dominated by a term proportional to a constant times number of false neighbors.

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Tradeoff measure

$$E_{\mathrm{NeRV}} = \lambda\mathbb{E}_i[D(p_i,q_i)] + (1-\lambda)\mathbb{E}_i[D(q_i,p_i)]$$

Minimize with respect to output coordinates y_i

Neighbor Retrieval Visualizer

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

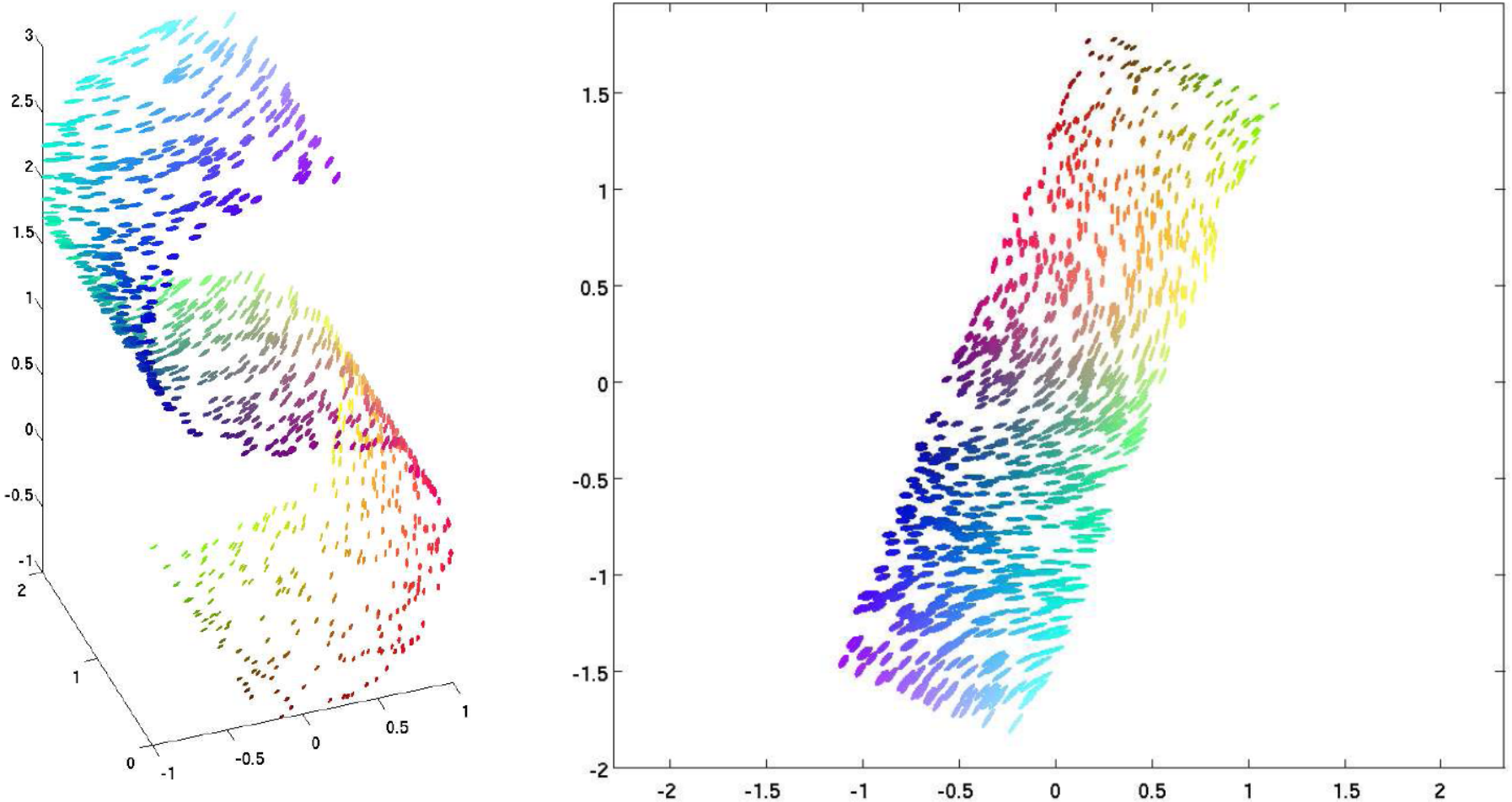
$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{y}_i-\mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Tradeoff measure

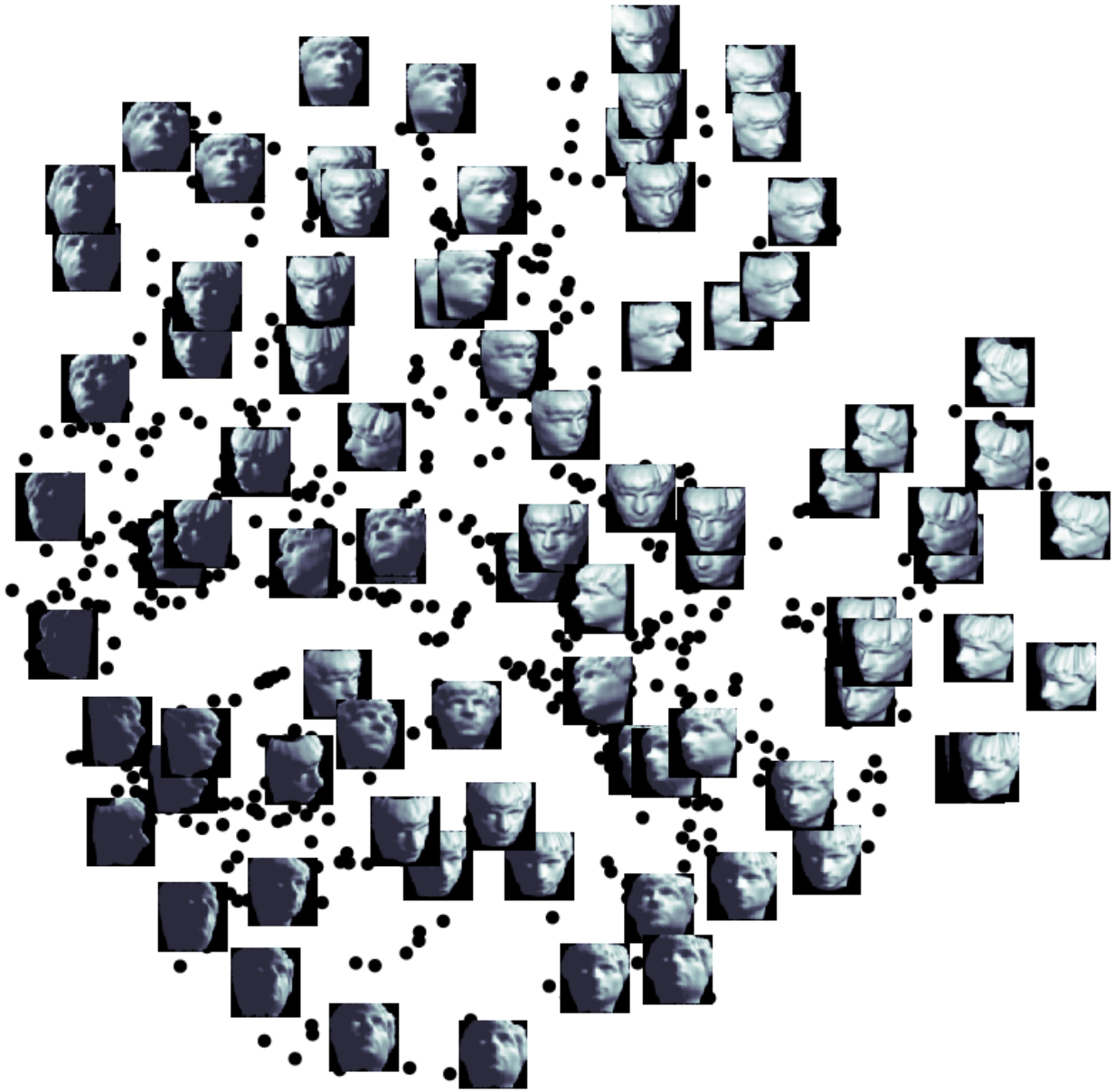
$$E_{\mathrm{NeRV}} = \lambda\mathbb{E}_i[D(p_i,q_i)] + (1-\lambda)\mathbb{E}_i[D(q_i,p_i)]$$

Minimize with respect to output coordinates y_i

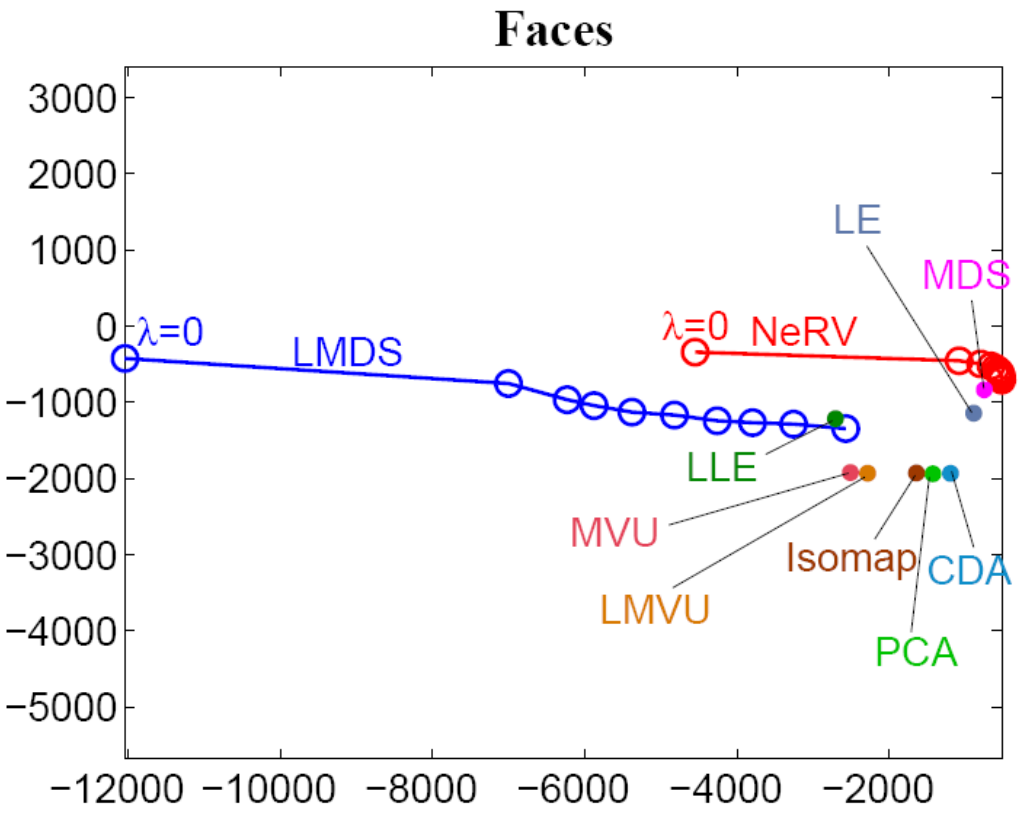
NeRV visualization

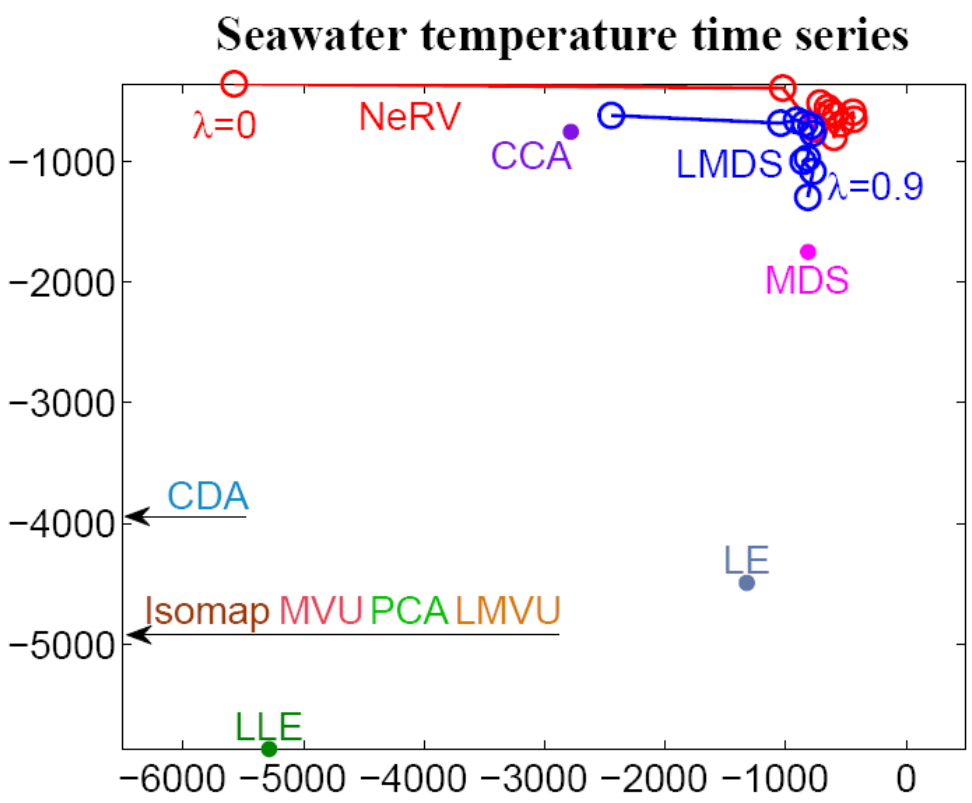


Of course NeRV can unfold the simple cases.

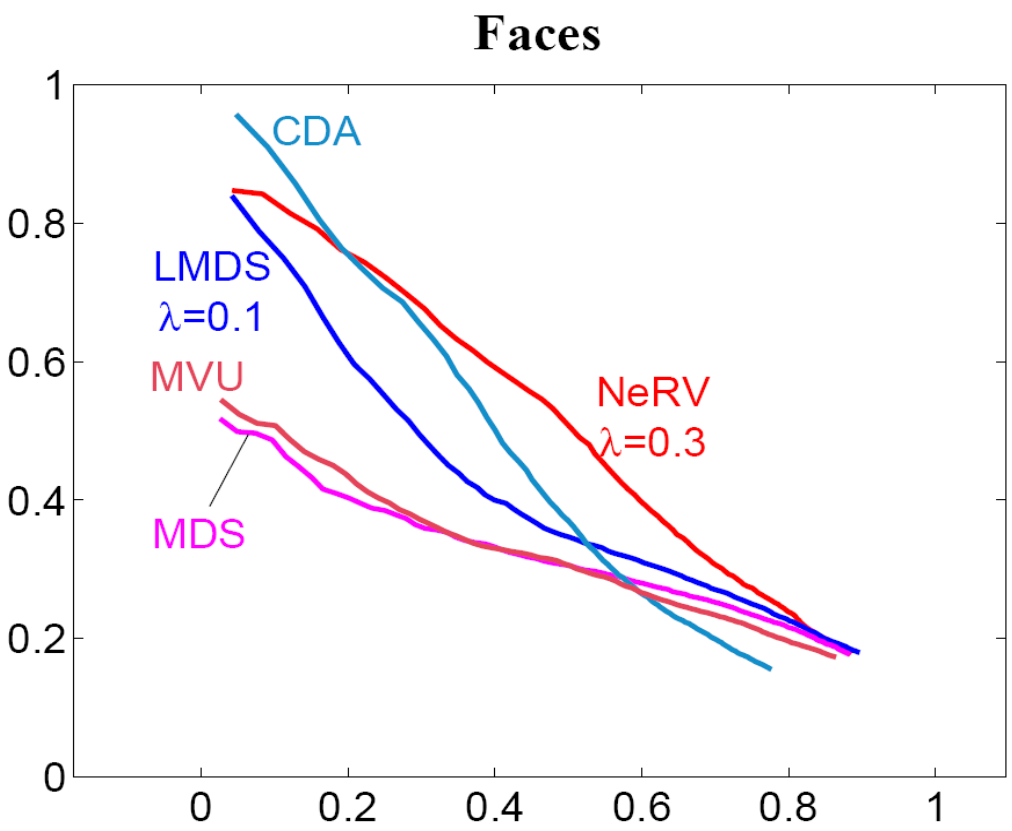


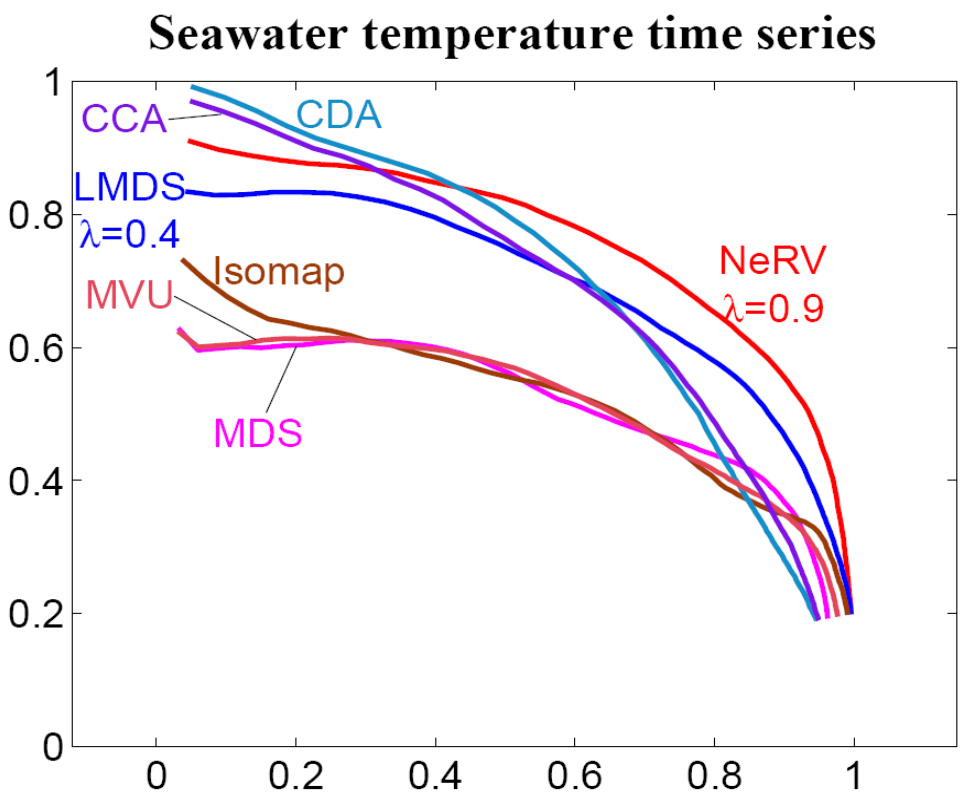
NeRV visualization of a complicated face image data set





New measures: smoothed precision (vertical axes) / recall (horizontal axes)





Standard precision / recall curves (novel for visualization!)

t-distributed NeRV

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{d(\mathbf{x}_i,\mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

$$p_{i,j} = \frac{1}{2n}\left(p_{i|j} + p_{j|i}\right)$$

Output neighborhood

$$q_{i,j} = \frac{\left(1+||\mathbf{y}_i - \mathbf{y}_j||^2\right)^{-1}}{\sum_{k\neq l}\left(1+||\mathbf{y}_k - \mathbf{y}_l||^2\right)^{-1}}$$

Tradeoff measure = t-NeRV cost function

$$E_{\text{t-NeRV}} = \lambda\sum_i\sum_{j\neq i} p_{i,j}\log\frac{p_{i,j}}{q_{i,j}} + (1-\lambda)\sum_i\sum_{j\neq i} q_{i,j}\log\frac{q_{i,j}}{p_{i,j}} = \lambda D(p,q) + (1-\lambda)D(q,p)$$