EuroVis 2018
Machine Learning Methods in
Visualization for Big Data 2018
4 June 2018, Brno, Czech Republic

Dimensionality Reduction

Jaakko Peltonen[1,2]

[1]University of Tampere, Faculty of Natural Sciences
[2]Aalto University, Department of Computer Science

Dimensionality reduction

High-dimensional data is hard to explore thoroughly, unfeasible to explore exhaustively

Dimensionality reduction

- Reveals hidden variable dependencies
- Quantifies importance of variables and variable combinations
- Reduces overfitting of learning models
- Reduces storage, memory, or data transmission consumption
- Makes it easier for humans to explore and get an overview of data
- Can build coordinates out of comparisons (distances/neighbors)

Linear methods

Linear methods: represent important variation along latent axes

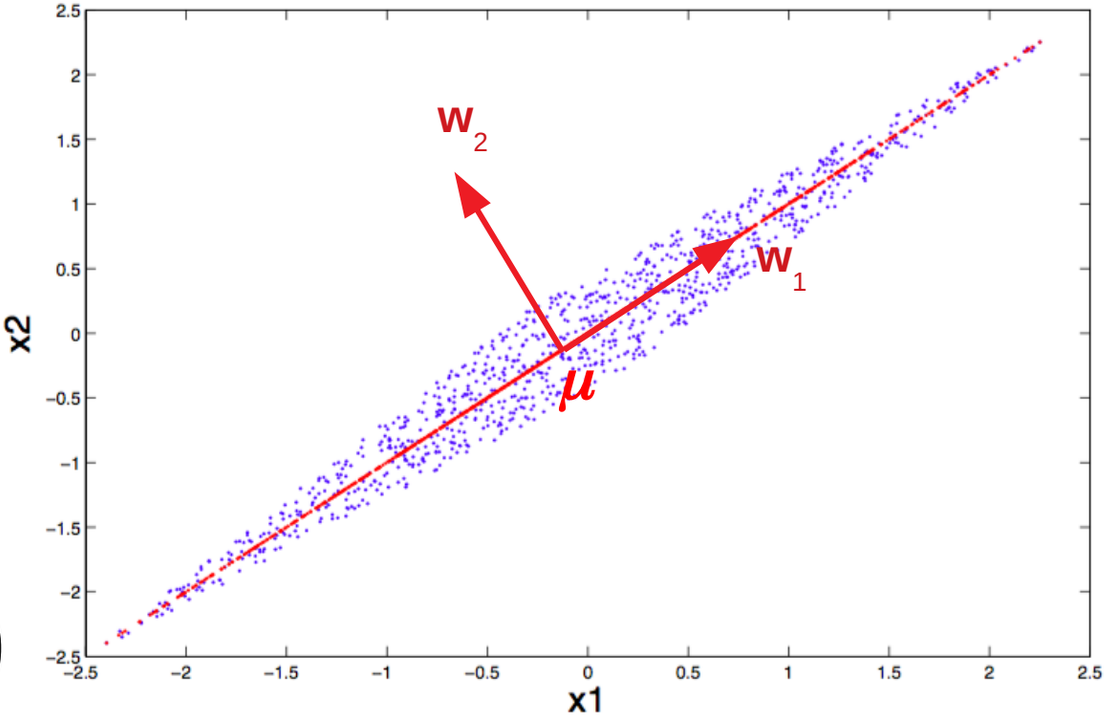
Feature selection: select a subset of features,

- by a measure of their individual goodness: e.g. correlation with a target variable
- or by joint goodness: e.g. performance of a classifier on those features only
- May need a search strategy, e.g. forward selection: add 1 feature per step, choose it to maximize joint goodness

Linear methods

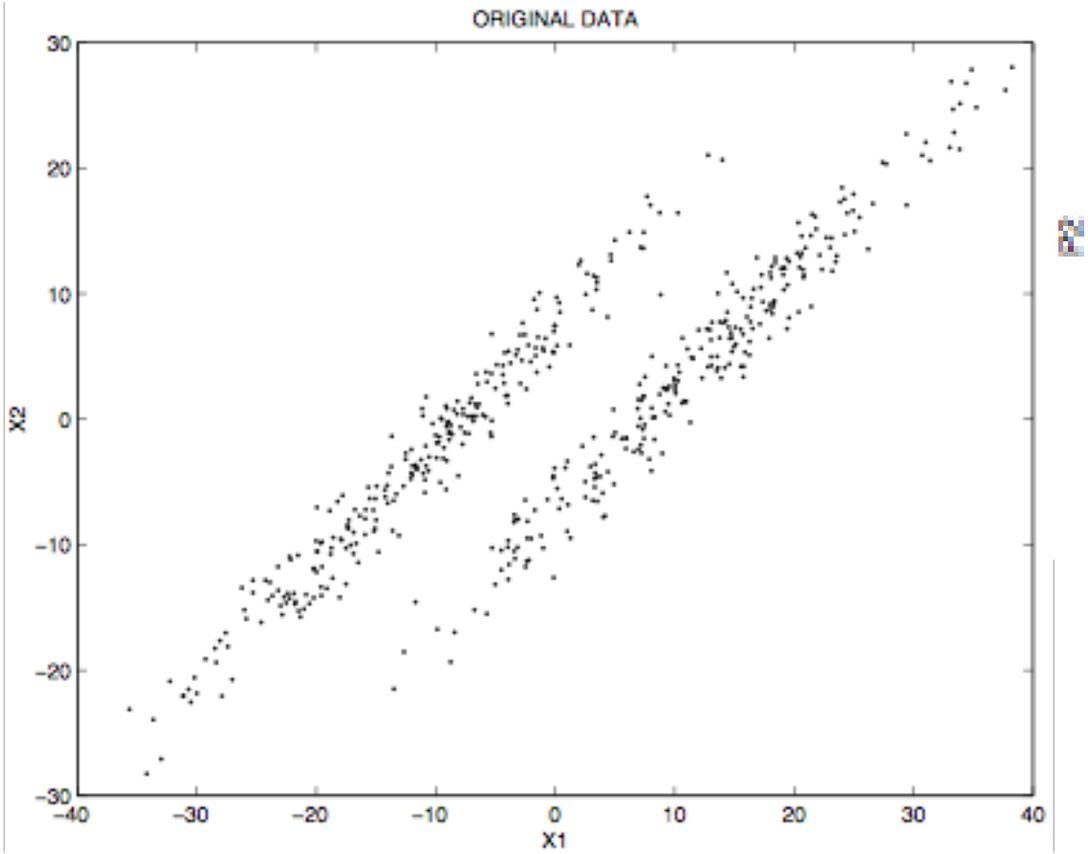
Principal component analysis:

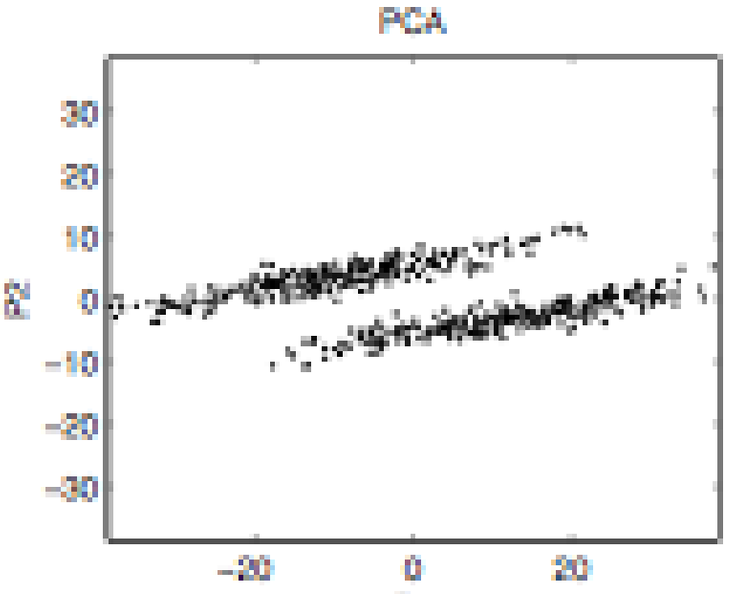
- Finds orthogonal axes containing maximal variance, $\boldsymbol{z} = \boldsymbol{W}^T (\boldsymbol{x} - \boldsymbol{\mu})$
- Minimizes error of linear reconstruction: reconstruct $\boldsymbol{x} \approx \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}$
- Simple: eigenvectors of largest eigenvalues of the covariance matrix
- Solution affected by variable scales, outlier data
- Generative model:
  $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{\mu} + \boldsymbol{e}$
  $\boldsymbol{z} \sim N(0, \boldsymbol{I}_d) \quad \boldsymbol{e} \sim N(0, \sigma^2 \boldsymbol{I}_D)$
- Unsupervised, class information not used

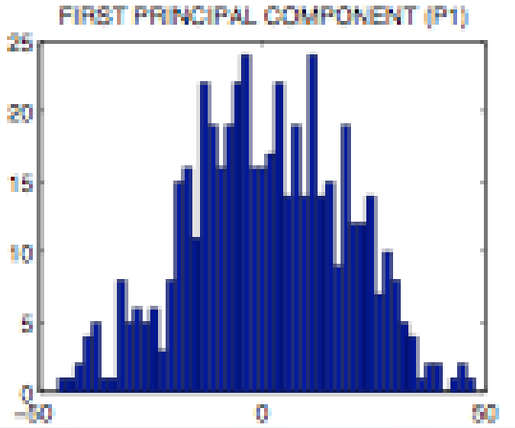


Linear projection may not be enough

PCA fails to separate the clusters (you don't see cluster structure from the 1D visualization)







For this data a different linear projection could be enough.

Linear discriminant analysis 1

- **Linear Discriminant Analysis** (LDA): supervised method for dimensionality reduction in classication problems.
- Goal: find directions that maximize between-class scatter, minimize within-class scatter.
- Solution as an eigenvalue problem
- Yields up to number of classes – 1 components
- Classification in the low-dimensional space can be done e.g. by finding the nearest class centroid of a new point

- Goal is not the same as maximizing class separability: Pairs of classes that are far apart dominate the LDA criterion, and can leave overlap between the remaining classes

Linear discriminant analysis 2

How to compute the LDA solution:

- Within-class scatter matrix $\hat{\Sigma}_w = \sum_{i=1}^{n} \sum_{\mathbf{x} \in c_i} (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)'$

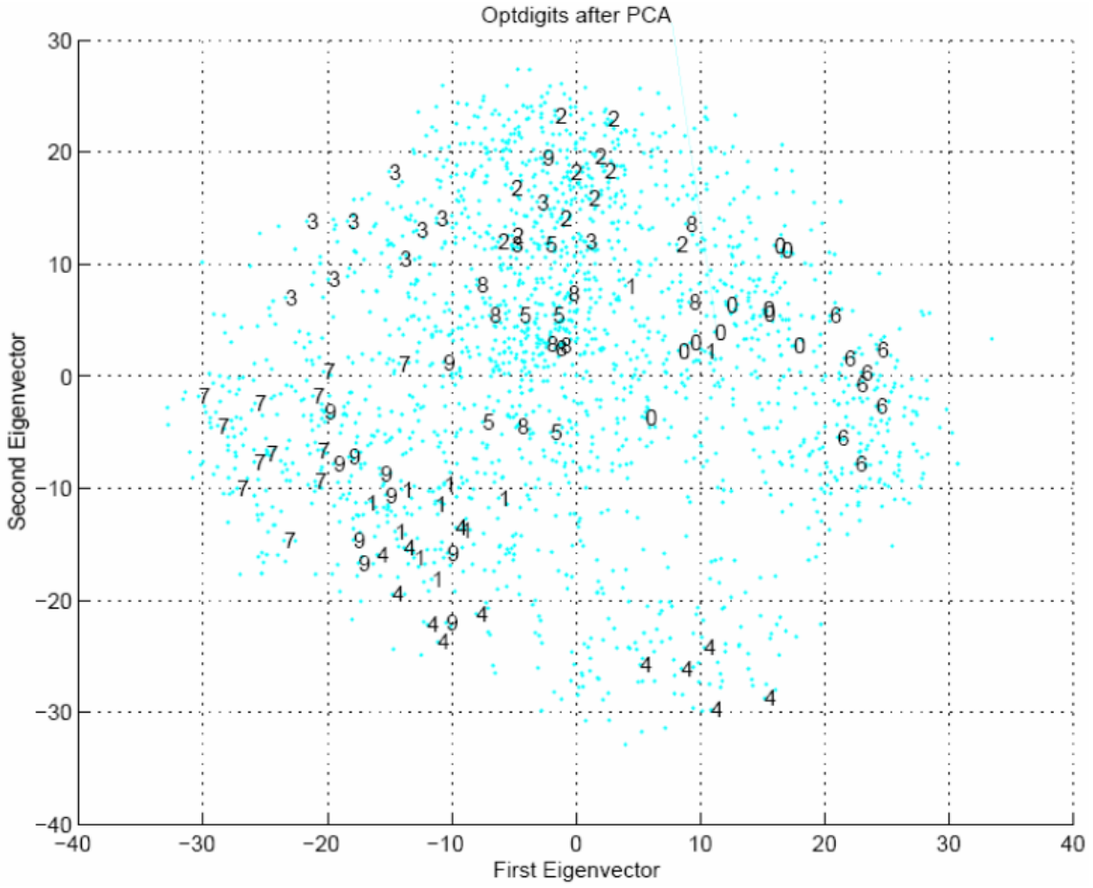
  where $\bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in c_i} \mathbf{x}$ and m_i is the number of samples in c_i
- Between-class scatter matrix

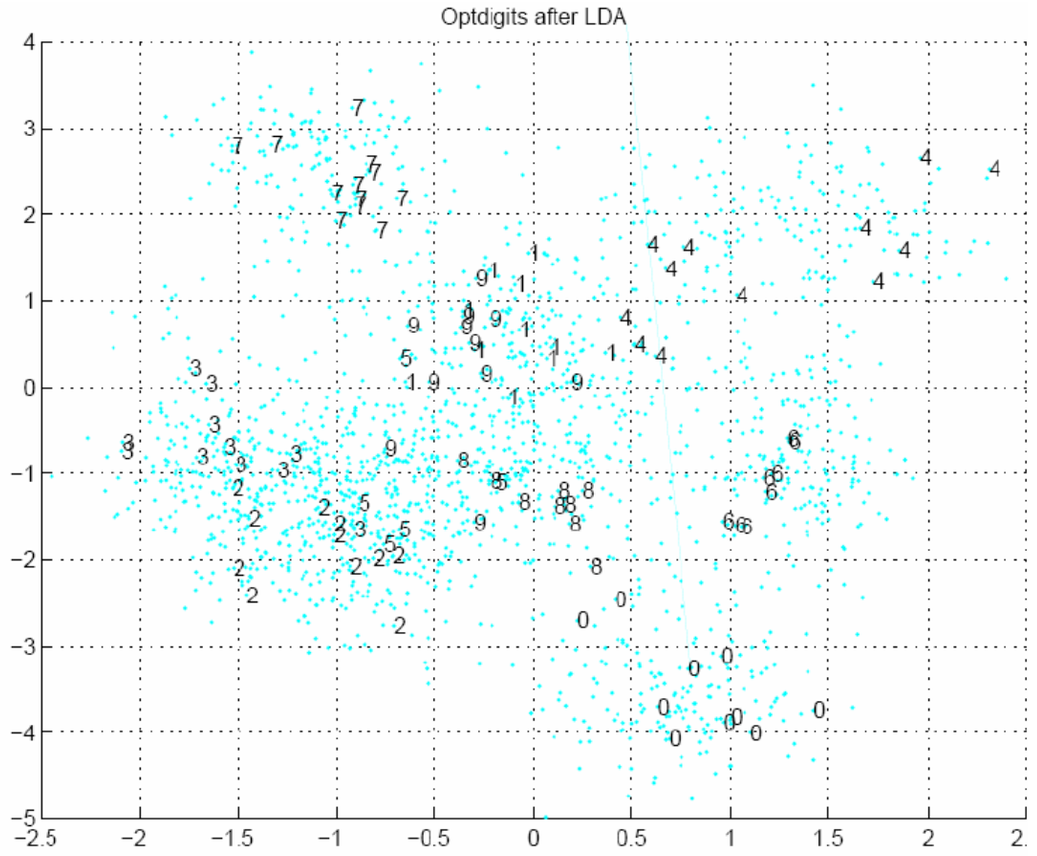
$$\hat{\Sigma}_b = \sum_{i=1}^{n} m_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})'$$

- Optimize projection matrix Φ to maximize ratio of between-class to within-class scatter: $\mathcal{J}(\Phi) = \frac{|\Phi^T \hat{\Sigma}_b \Phi|}{|\Phi^T \hat{\Sigma}_w \Phi|}$
- Optimized matrix Φ given by solving the generalized eigenvalue problem $\hat{\Sigma}_b \Phi = \lambda \hat{\Sigma}_w \Phi$

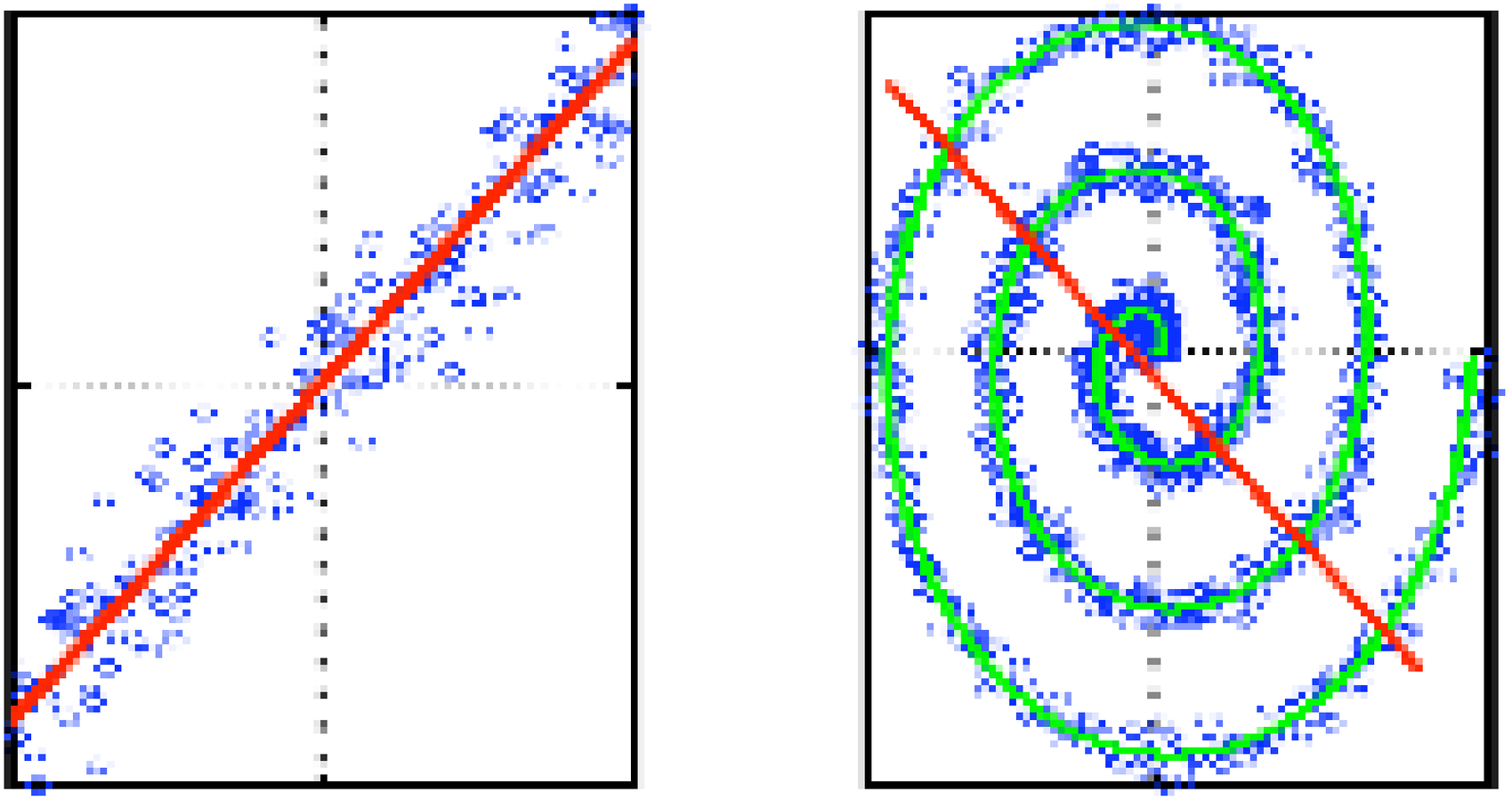
Linear discriminant analysis 3

- OptDigits example:



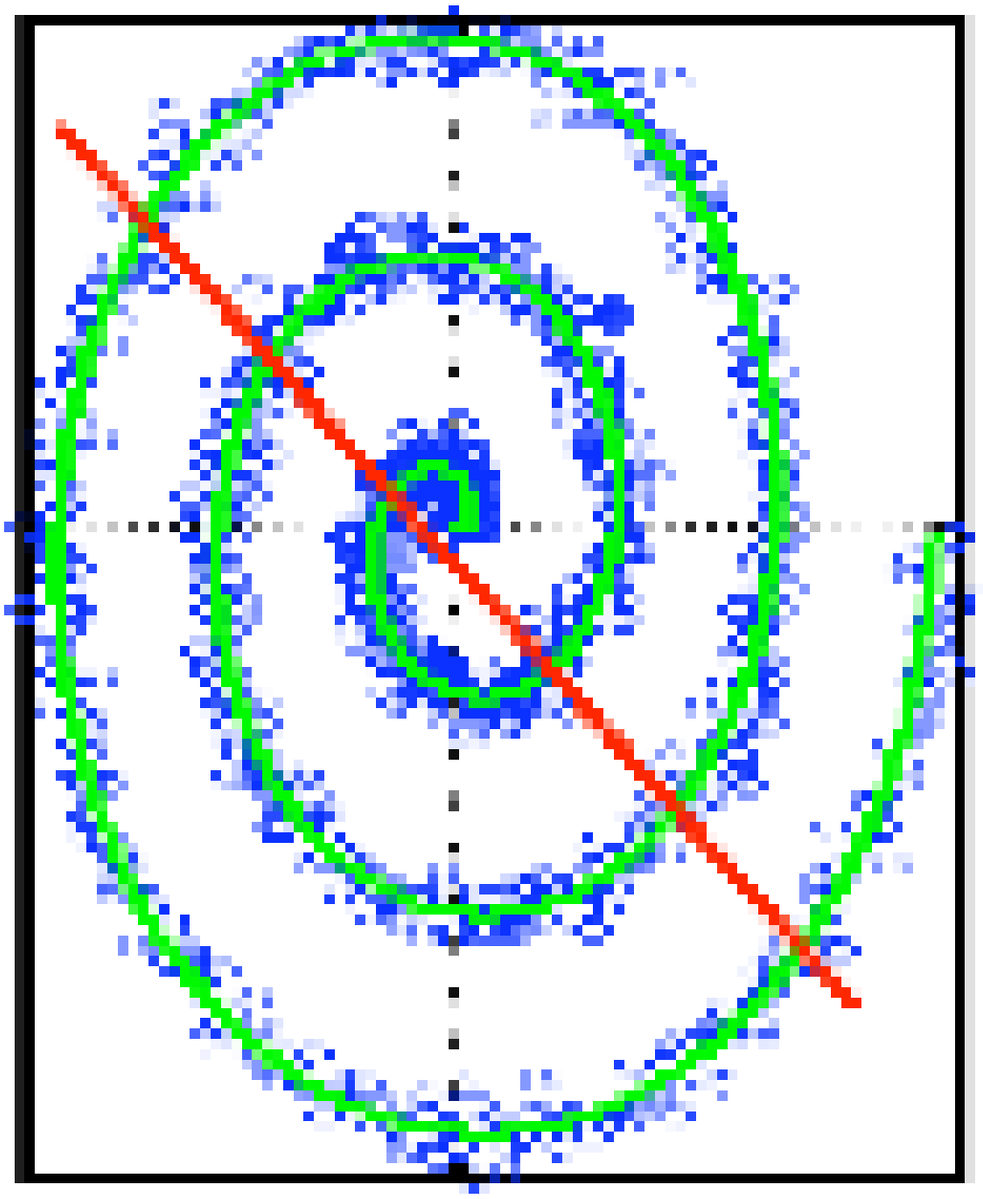


Nonlinear data



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

Manifolds



- PCA would not find the "correct" 1D manifold (green) because a) PCA is constrained to a linear mapping and b) PCA tries to preserve global features.
- Often, preserving local features, like neighborhoods, is more important than global properties.

Manifolds

Many dimensionality reduction approaches are based on the concept of **manifold learning**: the high-dimensional data is assumed to lie on a lower-dimensional "sheet" folded into a complicated shape in the high-dimensional space.

The idea is: if we know the dimensionality of the underlying manifold, then using that as the output dimensionality of nonlinear dimensionality reduction can "recover" the manifold.

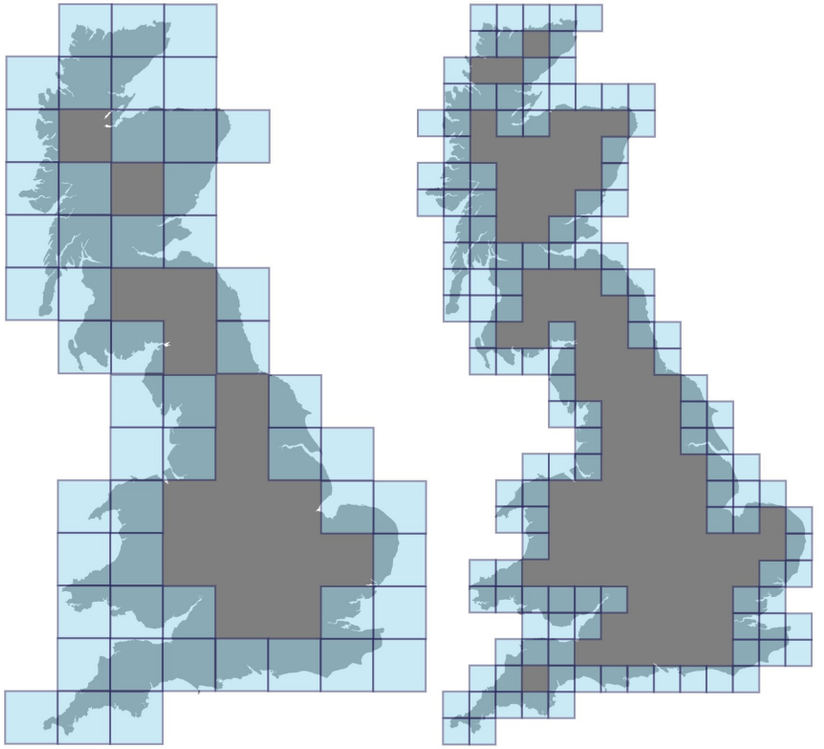
--------> Benchmark tests with artificial manifolds
(swiss roll, loops, s-curve, etc.)

Intrinsic dimensionality

How many dimensions are needed with nonlinear dimensionality reduction?

Box-counting dimension:

- make a hypercube around all the data
- divide it into a grid of smaller hypercubes of length ϵ
- only $N(\epsilon)$ of them contain data
- when ϵ shrinks, estimates $\frac{\log N(\epsilon)}{\log(1/\epsilon)}$ dimensionality

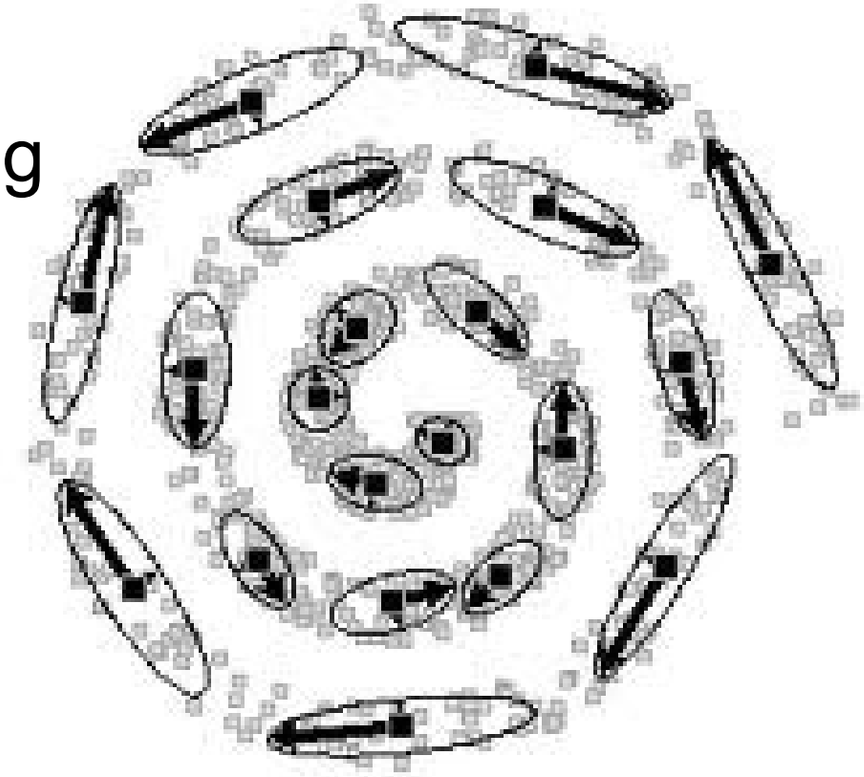


Intrinsic dimensionality

How many dimensions are needed with nonlinear dimensionality reduction?

PCA-based local estimate:

- decompose the space into local patches, e.g. a grid or by clustering
- carry out PCA on each local patch, find out number of dimensions needed to preserve e.g. 90% of variance.
- dimension of the manifold = average over local PCAs (weighted by number of points in each patch)



Nonlinear dimensionality reduction

Aim: Represent high-dimensional data by low-dimensional counterparts preserving as much information as possible

Ill-posed problem: which information in data is relevant to the user? (dependent on the specific data domain and situation at hand)

Huge variety of methods proposed with different properties

Nonlinear methods

- **Multidimensional scaling (MDS)** - preserves global distances
- **Sammon's projection** - variation of MDS, focuses on short distances
- **Isometric mapping of data manifolds (ISOMAP)** - a graph-based method
- **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)
- **Neighbor retrieval visualizer (NeRV)**: family of methods preserving neighborhood relationships as an information retrieval task. Special cases include **stochastic neighbor embedding** and **t-SNE**

Group 1: Spectral methods

Spectral techniques: they 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

Group 2: Nonparametric methods

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 in the training set)

Group 3: Explicit mapping

Explicit mapping functions: some methods explicitly learn a (non-)linear mapping function

linear functions: Principal Component Analysis, Linear Discriminant Analysis

nonlinear functions: autoencoder networks, locally linear coordination, self-organizing map

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 psychology
- MDS can be used as
  - an exploratory visualization technique to find the structure of the data; and
  - a tool to test hypothesis.
- MDS only requires distances, known high-dim. coordinates are not needed

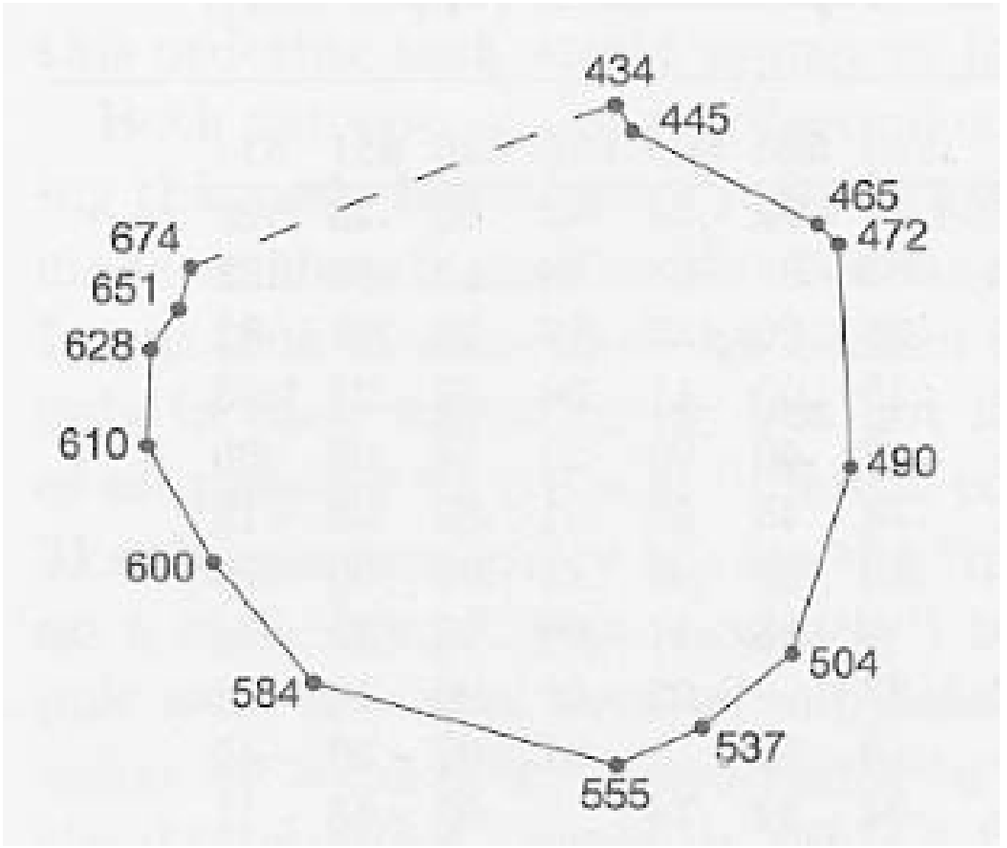
MDS for colors

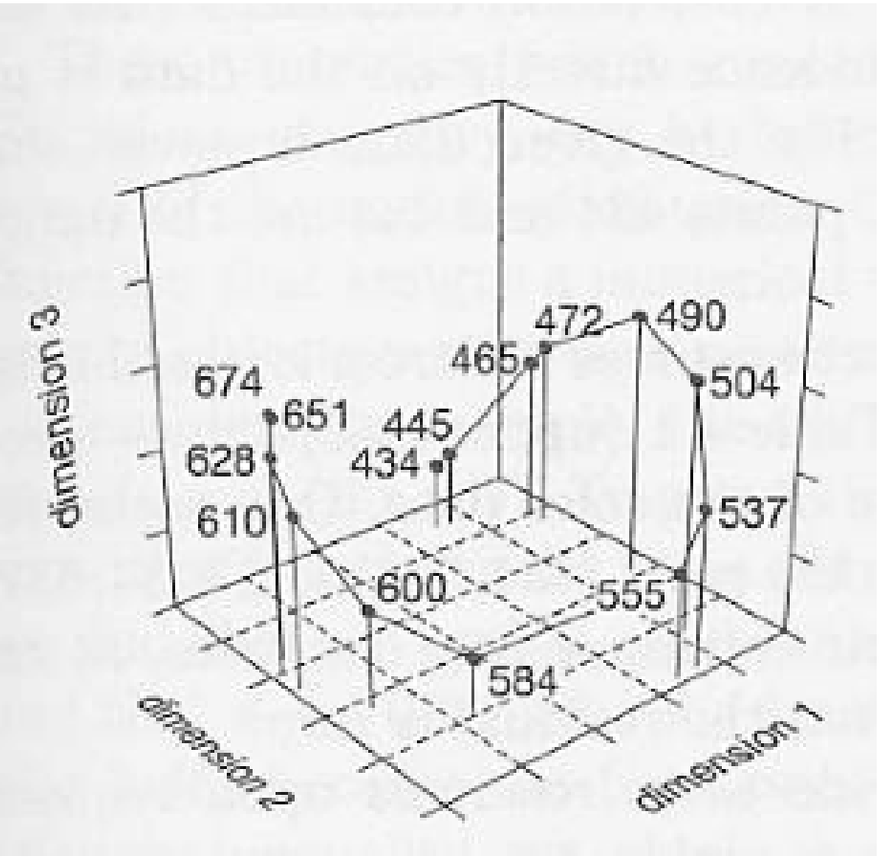
- Psychological test in 1950's: how is the similarity of colors perceived?
- Pairs of 14 colors were rated by 31 people. Ratings were averaged.

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	–

MDS for colors, result

The 14 colors were 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]

MDS definition

- An MDS algorithm is given the original distances p_{ij} (called *proximities)* between data points *i* and *j*
- MDS tries to find a low-dimensional (usually 2-3D) representation for the points with some coordinates *X*
- MDS minimizes the error function (*stress*)

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

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).

MDS variants

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

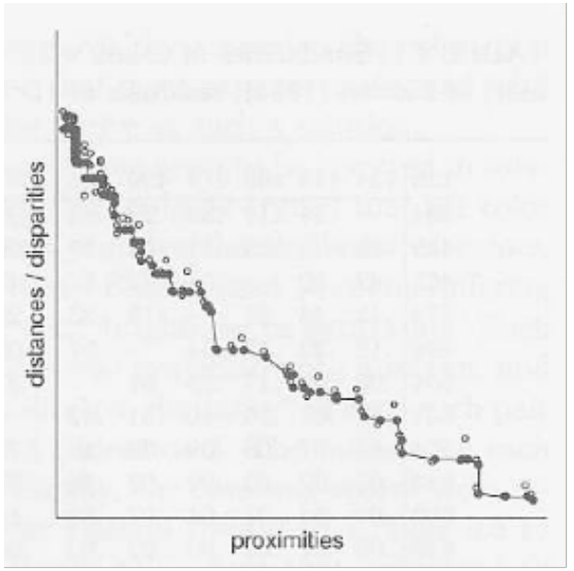
The choice of *f* defines the MDS model. For example:

- $f(p_{ij})=p_{ij}$ - 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*)

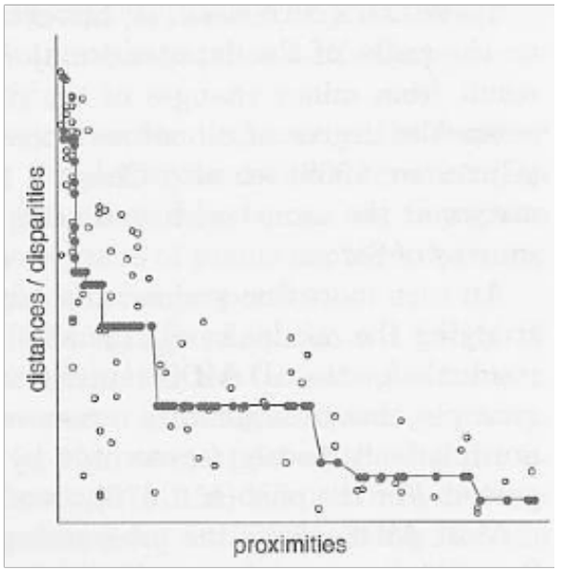
Goodness of MDS

There are two classical visualizations of MDS goodness:

- *Shepard diagram* shows low-dimensional distances d_{ij} (white circles) and target disparities $f(p_{ij})$ (filled circles) as a function of high-dimensional proximities p_{ij}.
- *Scree plot* helps pick an output dimensionality. It shows the MDS cost (stress) as a function of the dimensionality



2D MDS



1D MDS

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

MDS properties

- Often, large distances yield large stress if they are not preserved. In such a situation, MDS 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

- 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)

MDS properties

- 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.

Sammon mapping

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

- The Sammon mapping increases the importance of small distances and decreases the importance of large distances → nonlinear mapping
- 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 PCA.
- The minimization can be performed e.g. 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 PCA components as a starting configuration.

Curvilinear component analysis

- *Curvilinear component analysis* (CCA; Demartines, Hérault, 1997) is like MDS, but only **short distances on the display** are taken into account.
- The cost function is

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

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.)

Isomap

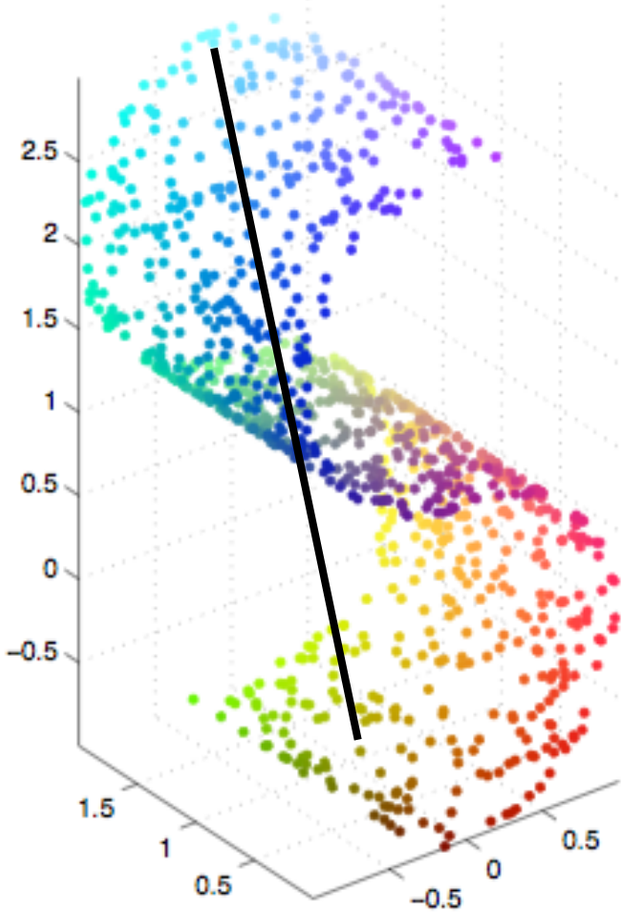
Multidimensional scaling methods tried to preserve all squared distances —> preservation of largest distances had biggest effect on the cost

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.

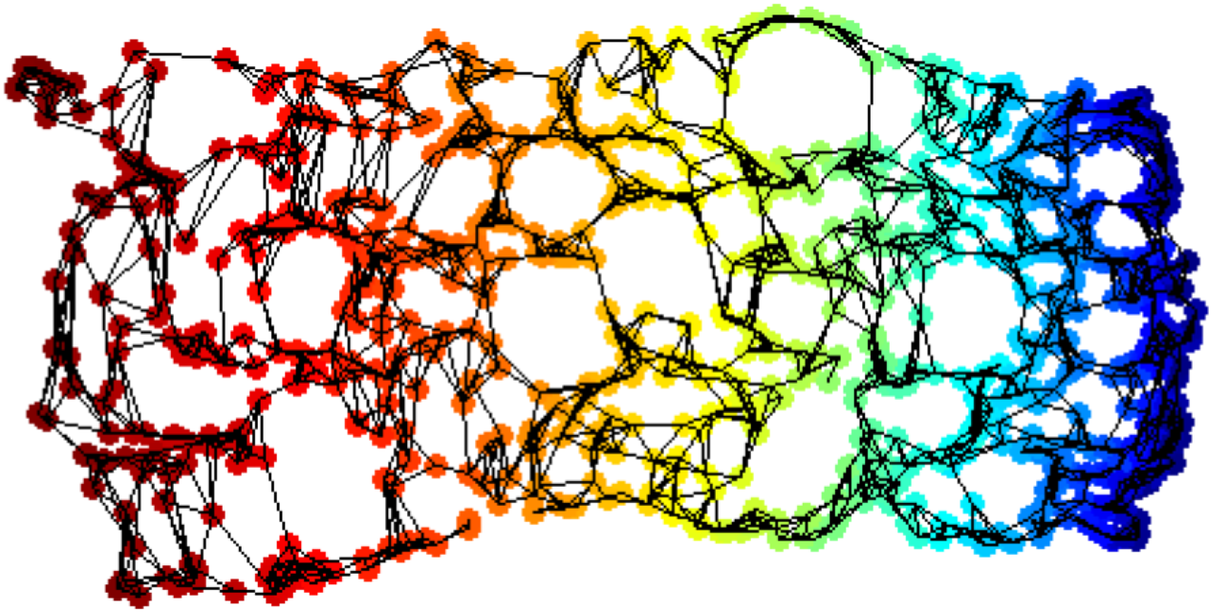


Euclidean distance between two points corresponds to the length of the line connecting the points. The line usually does not follow the manifold of the data.

Isomap

Euclidean distance in the original space might not be appropriate. Isomap:

- 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.

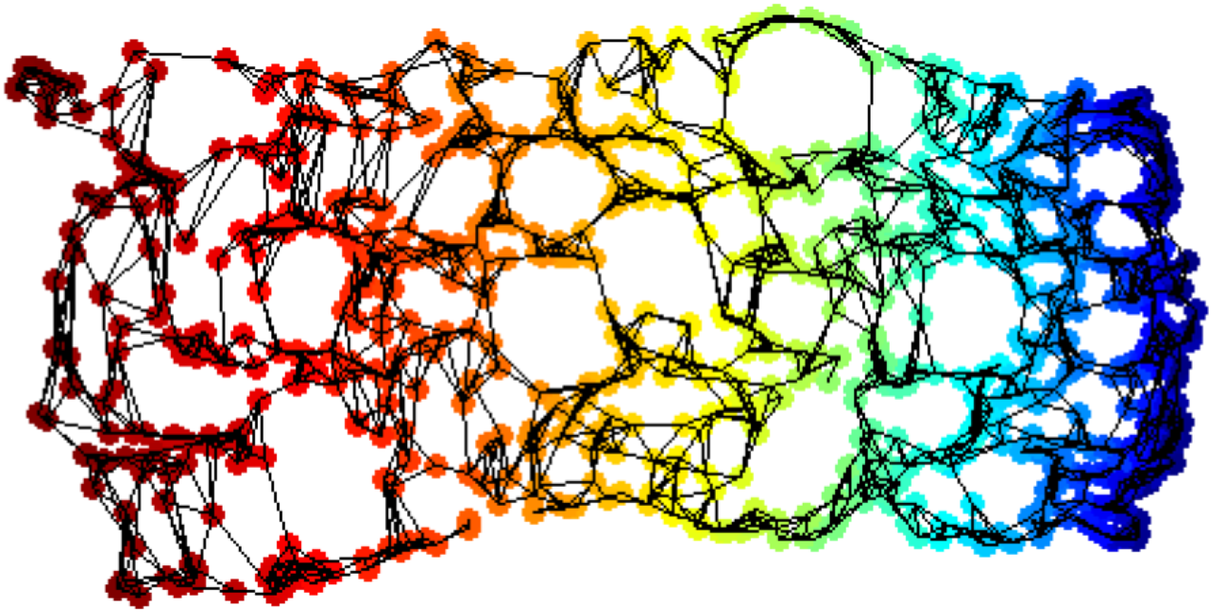


Isomap

Construct neighborhood graph (closest k neighbors of each point, or all points closer than a threshold).

Compute shortest path length along the graph between all pairs of points:

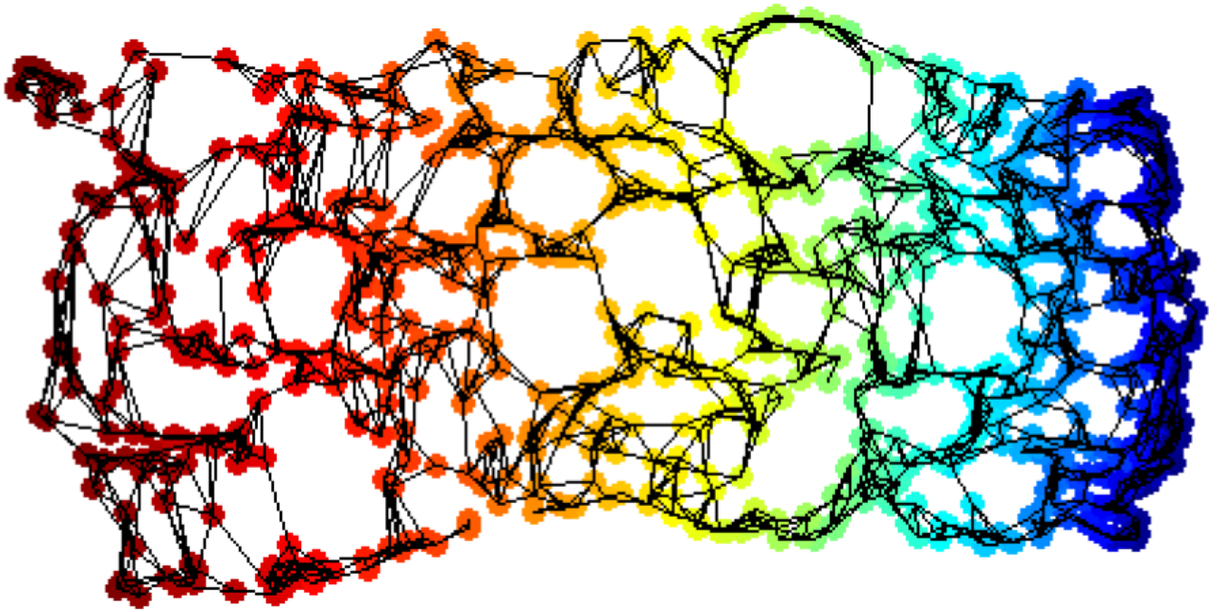
- first compute distance to the neighbors of each point, and write this as a distance matrix (set distance to non-neighbors to infinity).
- Then use Dijkstra's algorithm to find shortest distance along the graph from a point to all other points.



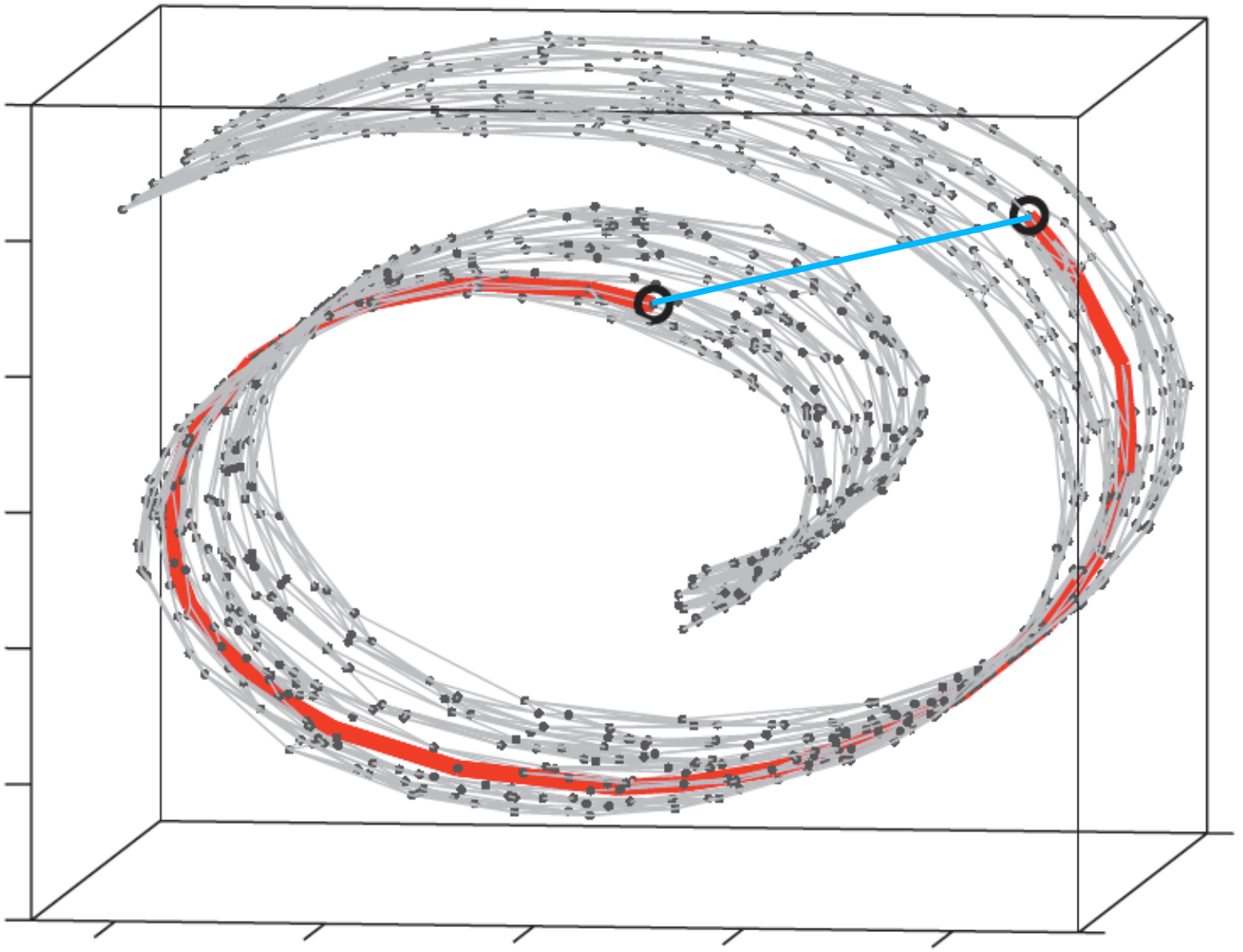
Isomap

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

Yields low-dimensional coordinates whose Euclidean squared distances best approximate the squared shortest-path distances.



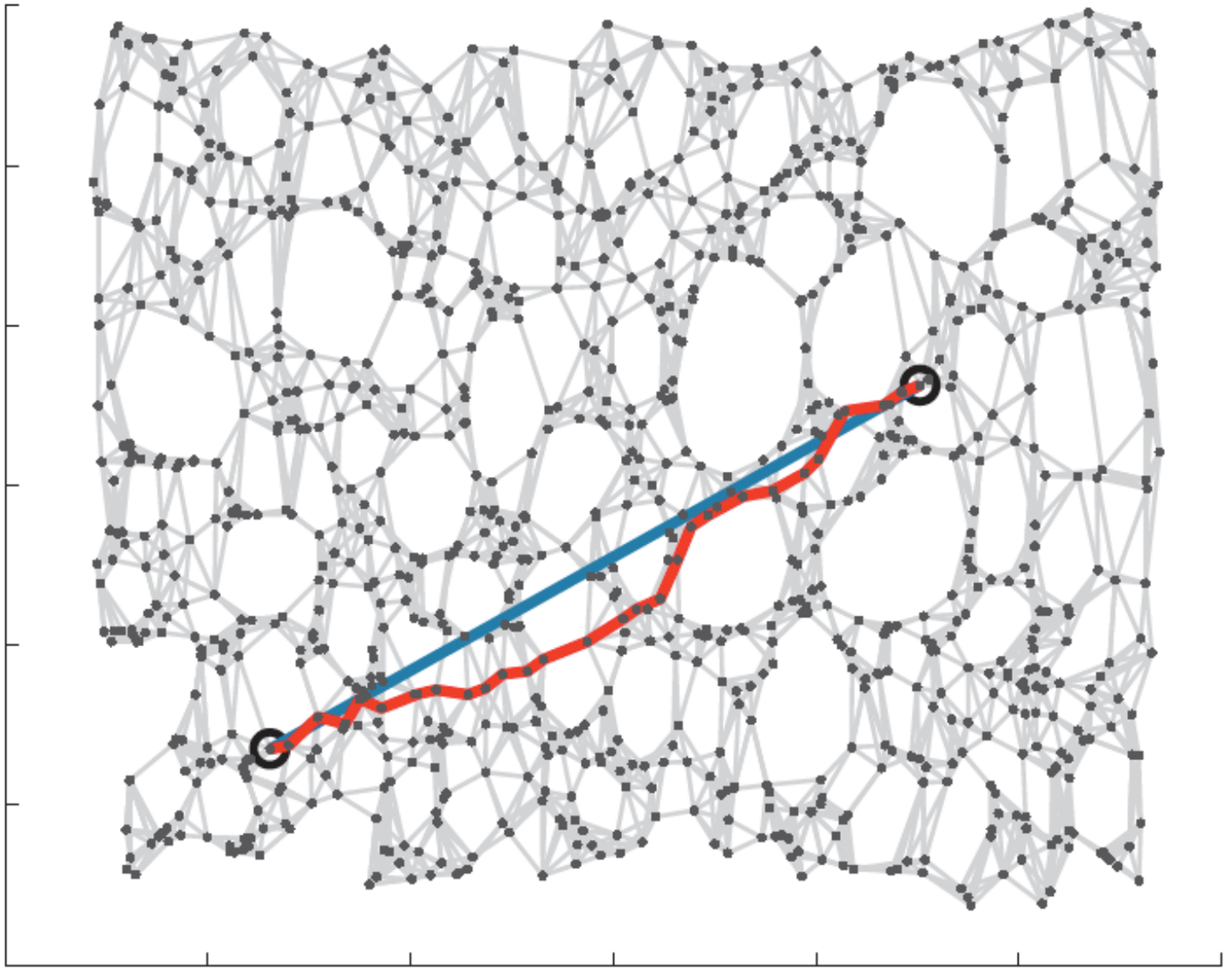
Isomap



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

Neighborhood graph (1000 data points, 7 neighbors connected to each), geodesic approximated by shortest path along the graph.

Isomap



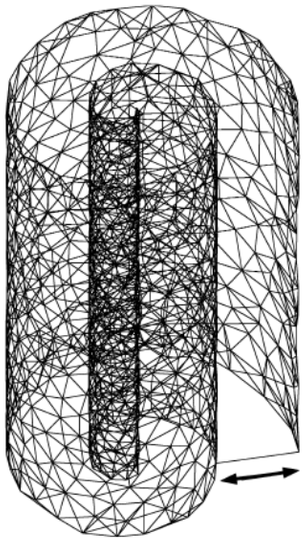
Two-dimensional embedding computed by MDS, to preserve the approximate squared geodesic distances, as squared Euclidean distances on the display.

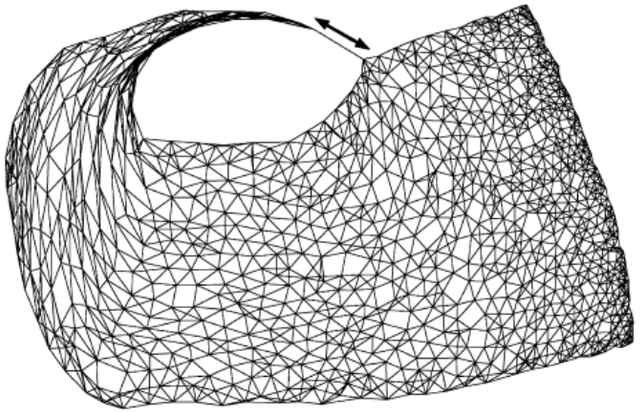
High-dim. geodesics are approximated directly by the low-dim. straight-line distances (not by shortest paths along the graph)

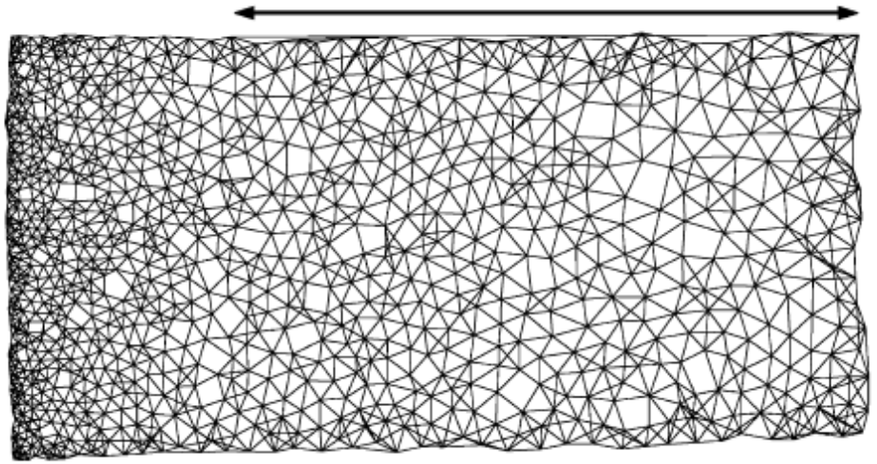
Curvilinear distance analysis

- Recall *Curvilinear component analysis* (CCA) is like MDS, but only short distances on the display are taken into account.
- 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!
- Short distances on the display preserve high-dim. geodesics

Poorly approximated long distances can distort Isomap. CDA distorts less, since it focuses on small distances.





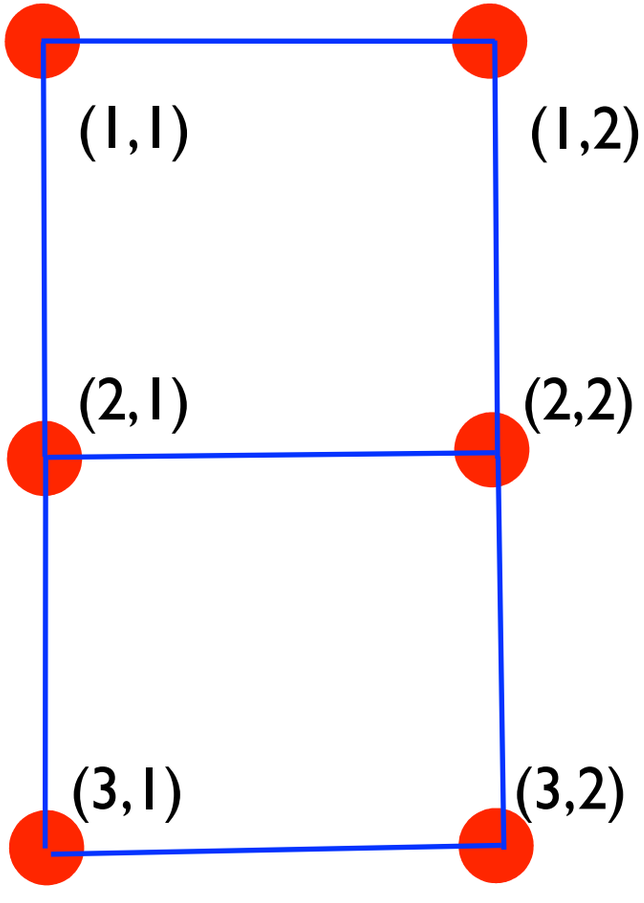


Self-Organizing Maps

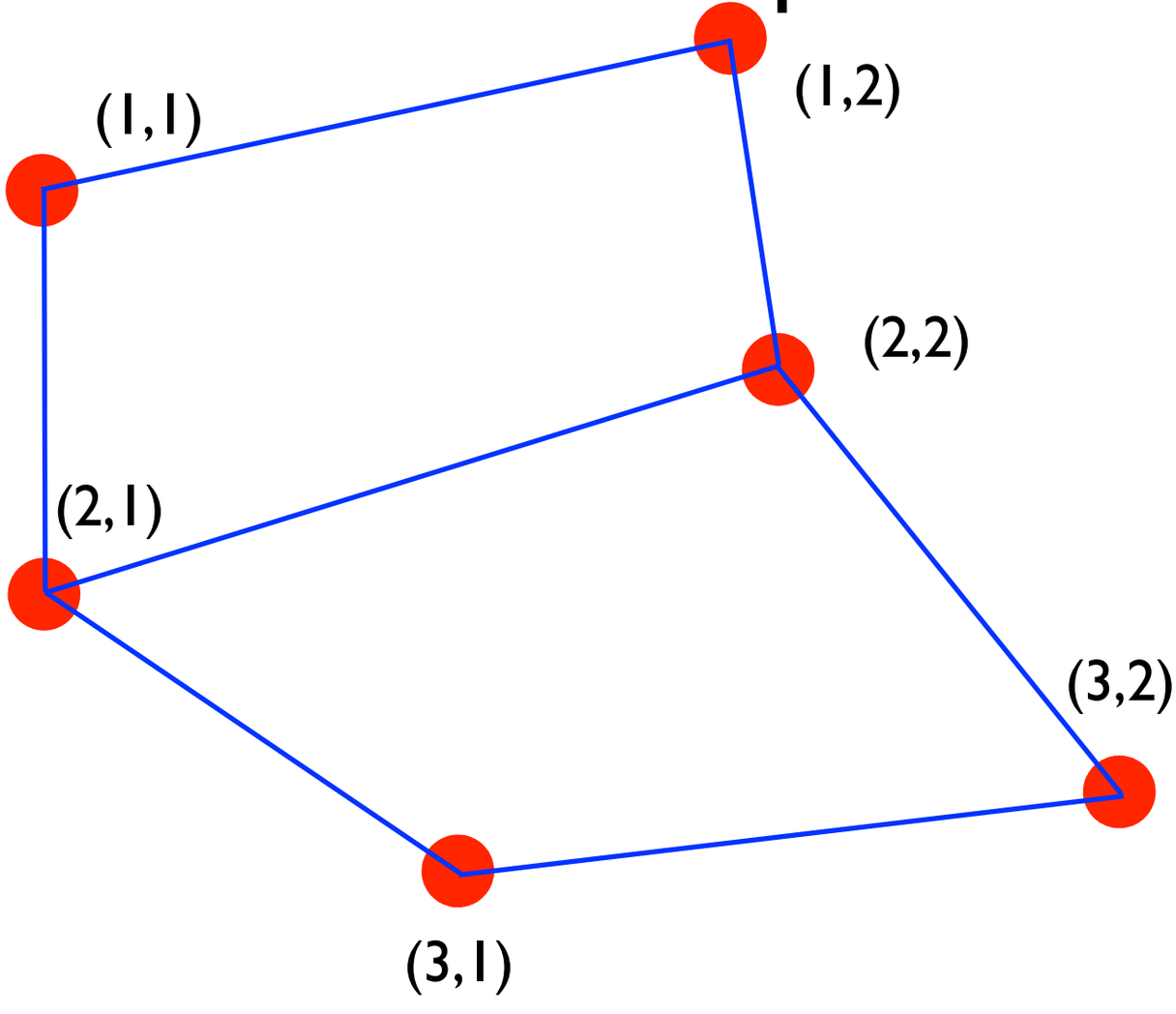
- Continues to be very popular in analytics: 2710 new references in Google Scholar in 2018 so far
- Maps data items to a grid of units, usually a 2D rectangle or hexagonal grid
- Each unit has a prototype in the data space
- Intuition: Like stretching a sheet of plastic by pulling on certain points on it

Self-Organizing Maps

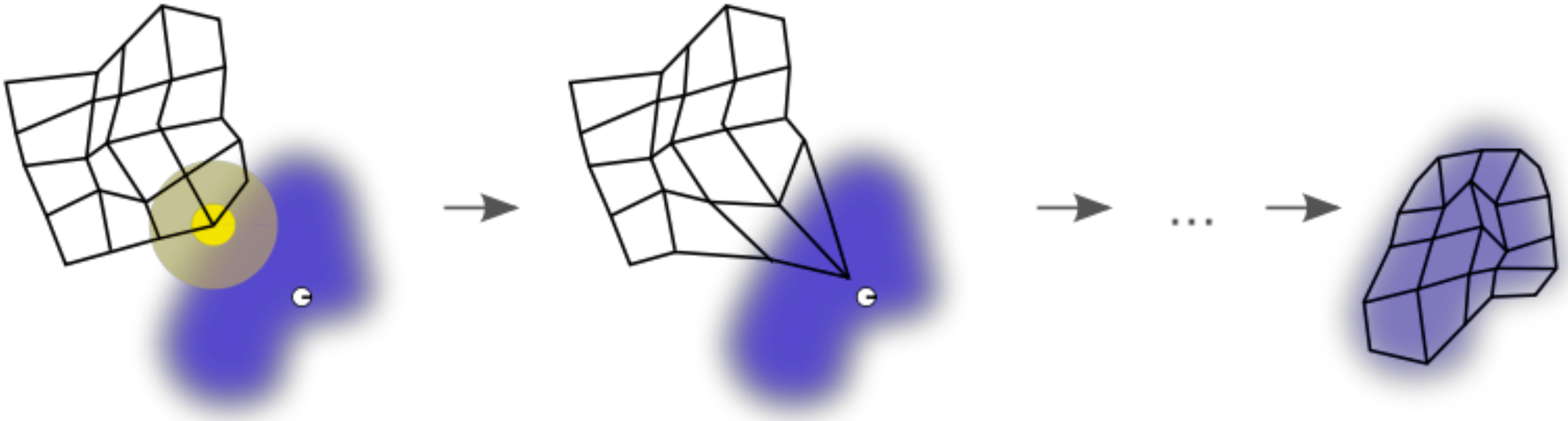
Structure of the map (definition)



Map in the data space



Self-Organizing Maps

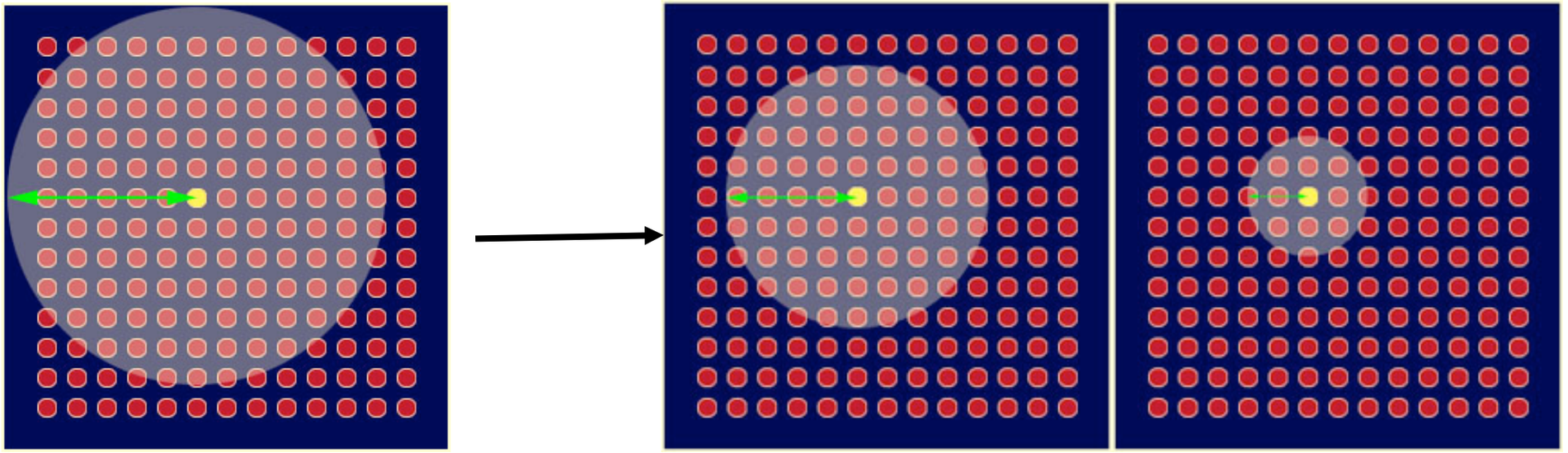


Self-Organizing Maps

- Random initialization or PCA initialization
- Iterations

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

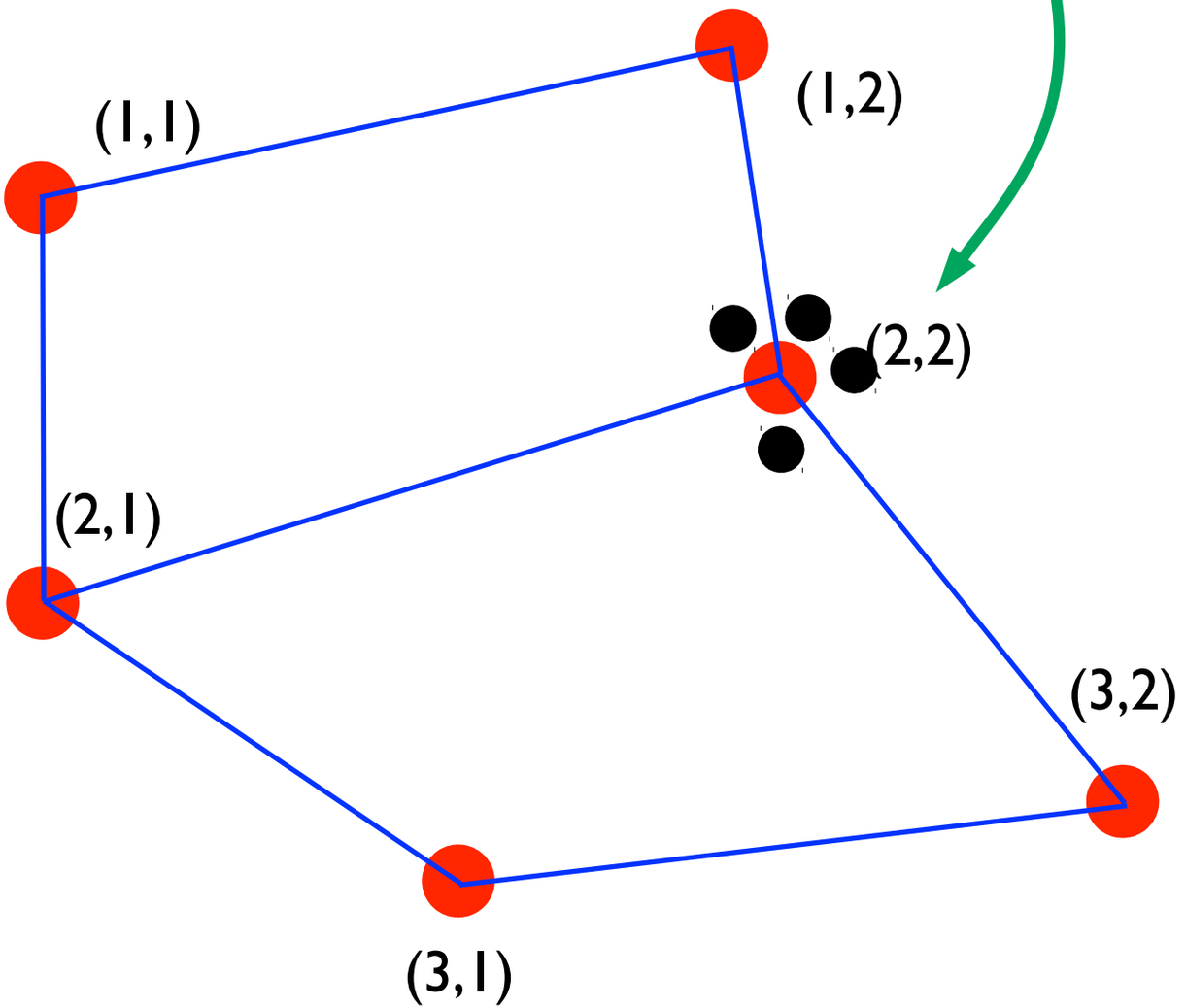
$$h_{u,v}(t)=\exp\left(\frac{-d_{grid}(u,v)}{\sigma(t)}\right) \quad \sigma(t)=\sigma_0\exp\left(\frac{-t}{\lambda}\right)$$

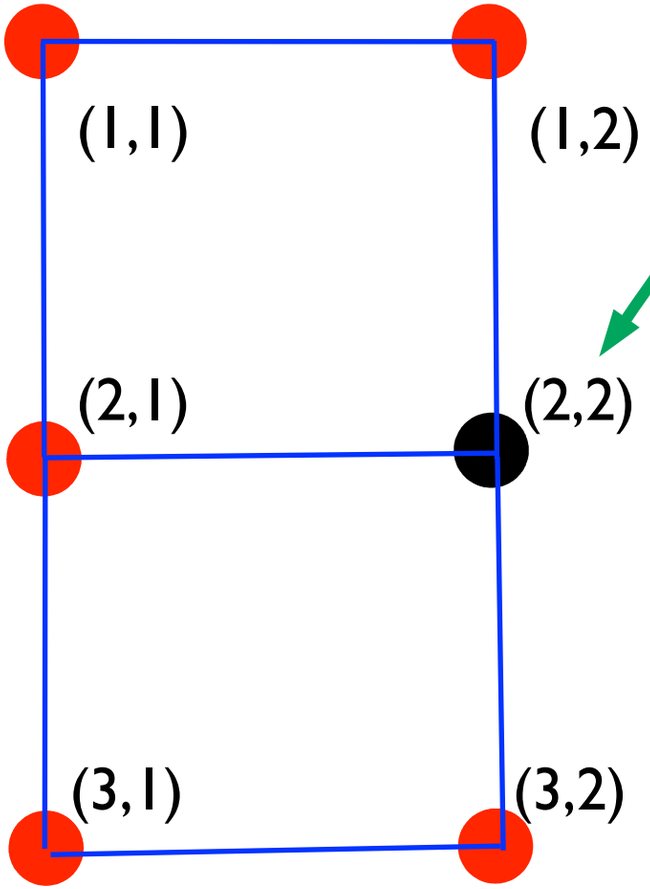


Self-Organizing Maps

Associate data items to their nearest prototype

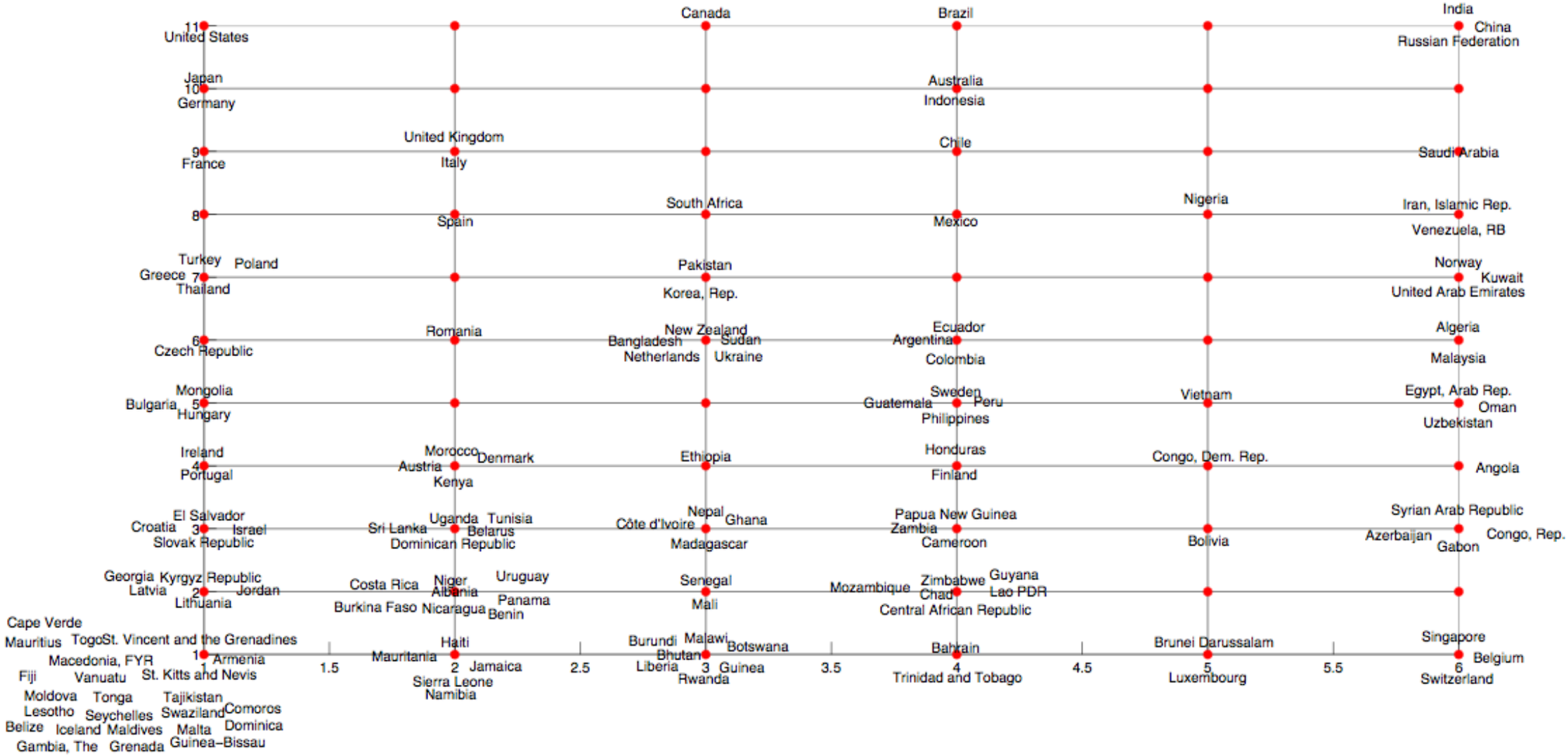
Summarize them at the corresponding grid unit



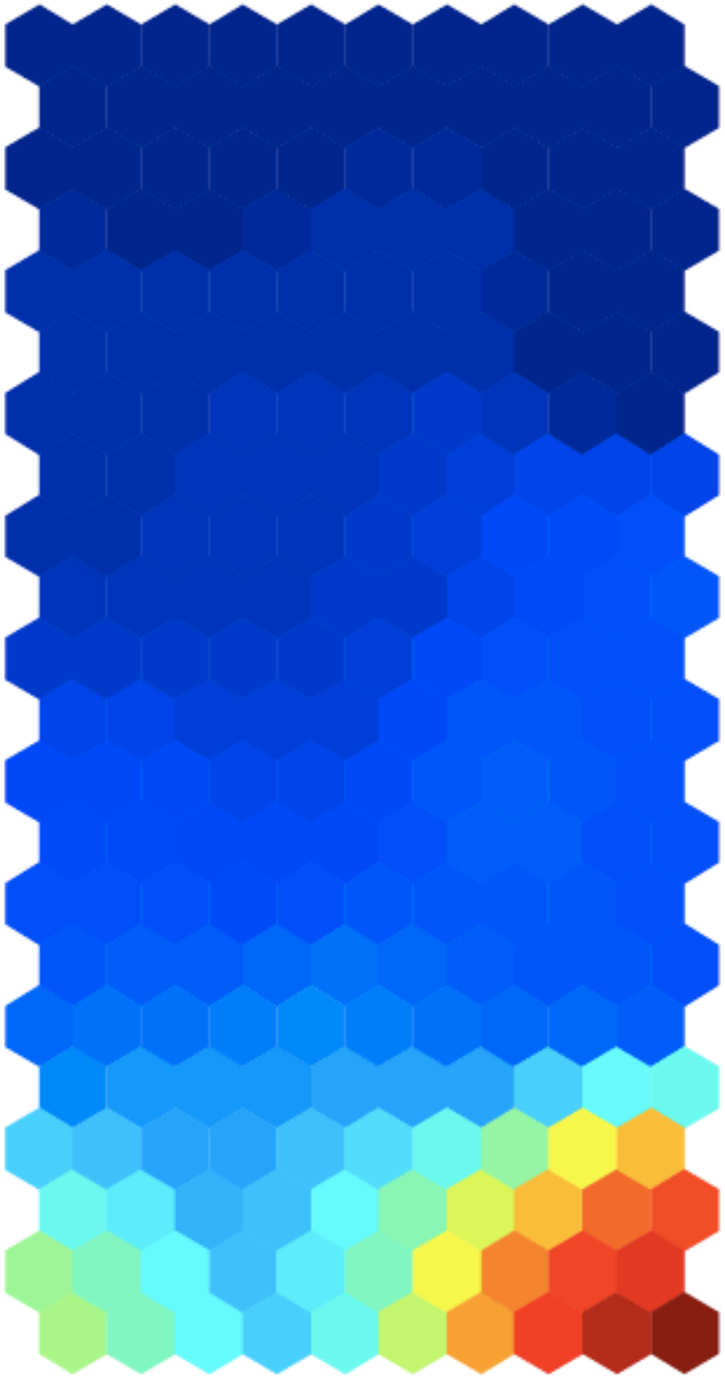


Self-Organizing Maps

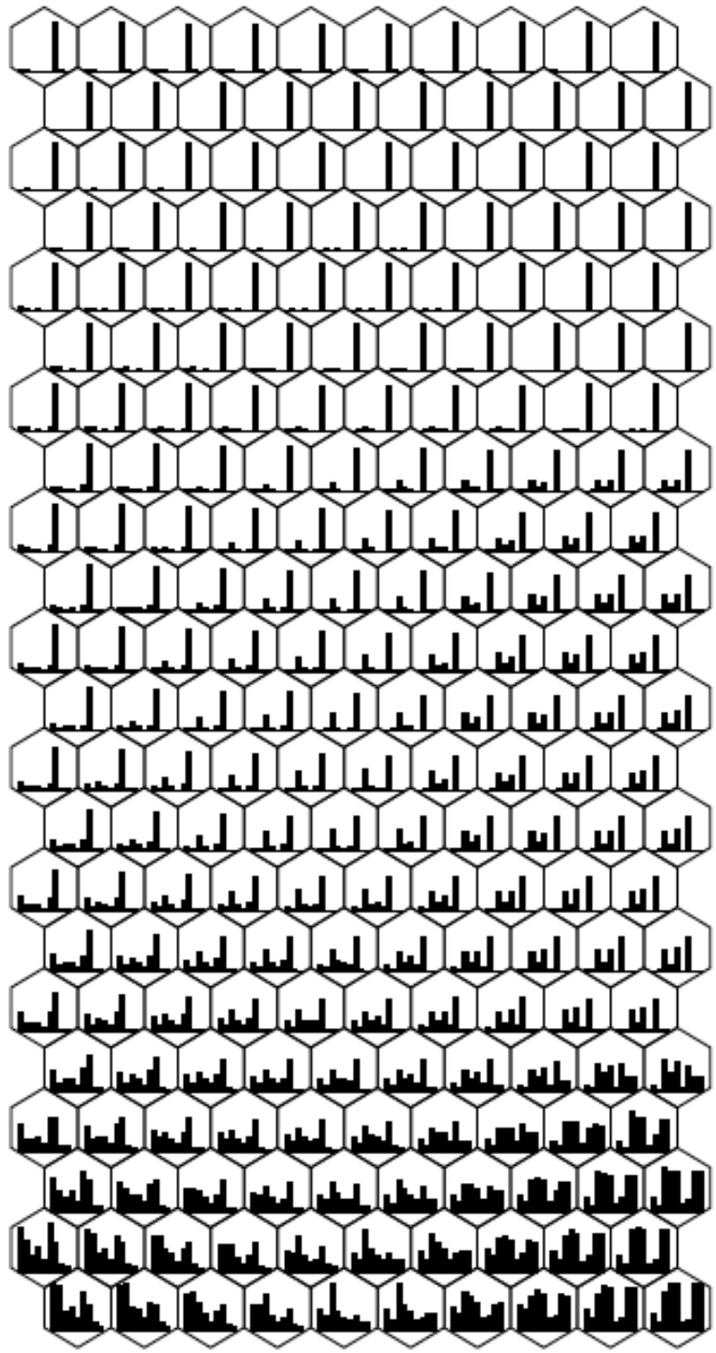
Names of data items shown on the SOM grid.



Self-Organizing Maps



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

Neighborhood preservation

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

- E.g. MDS stress measured distance preservation.

Other variants have modified which distances are most important to preserve:

- Sammon's mapping and CCA consider small distances the most important to preserve.
- Isomap and CDA modified the original distances to be preserved.

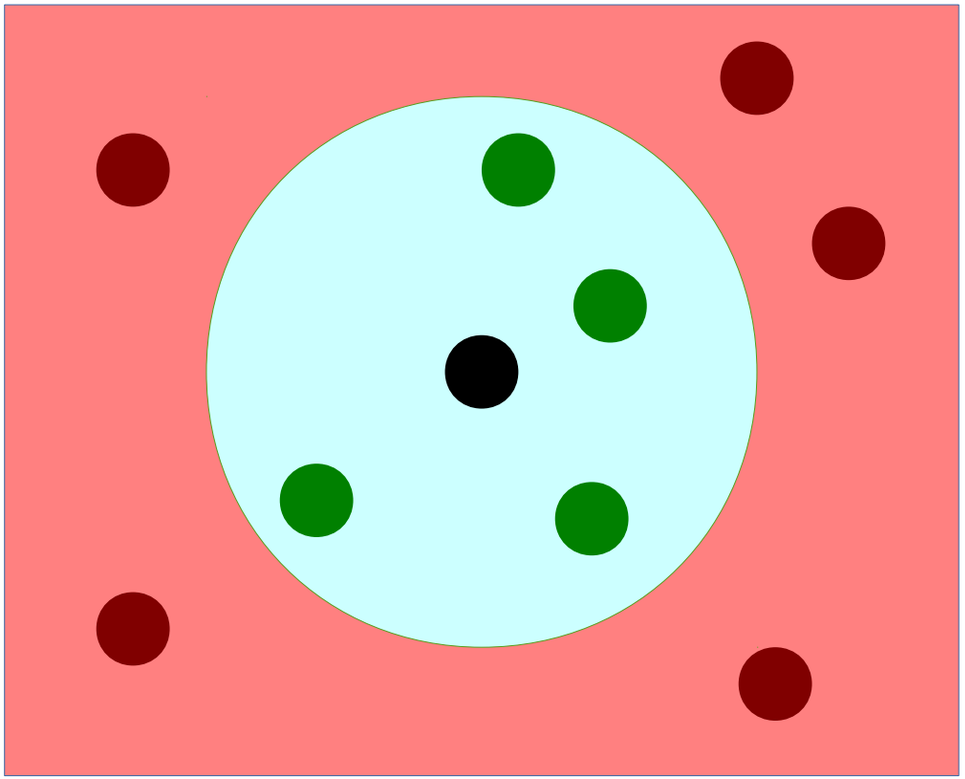
But the aim is still to preserve distances.

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

Neighborhood preservation

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.

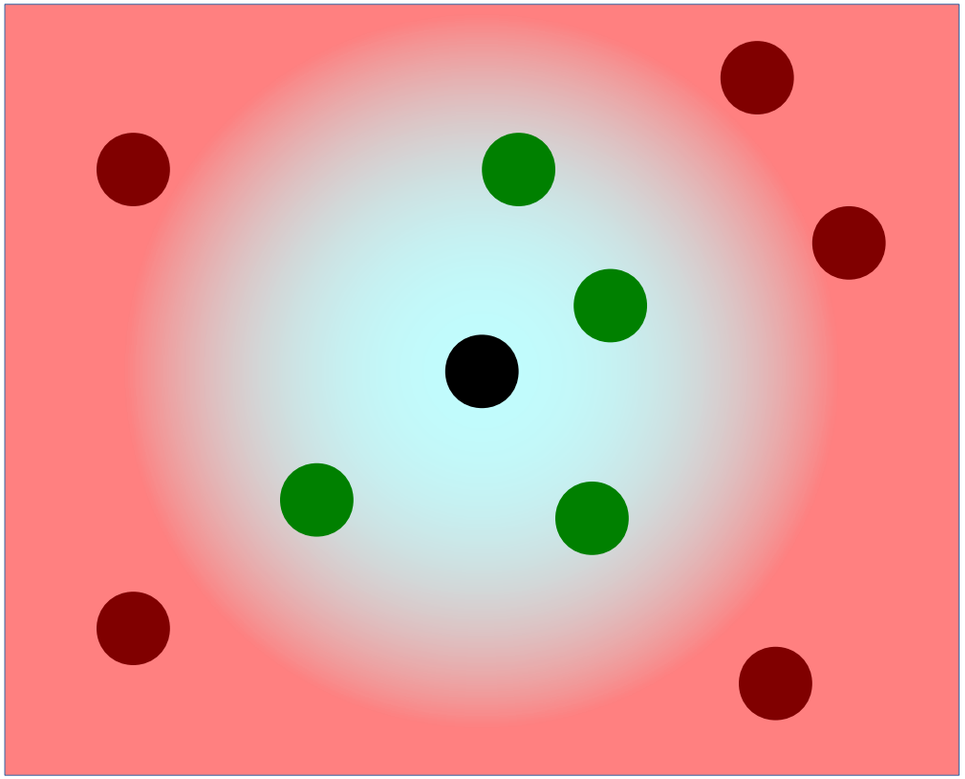


Hard neighborhood - each point is a neighbor or a non-neighbor

Neighborhood preservation

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.

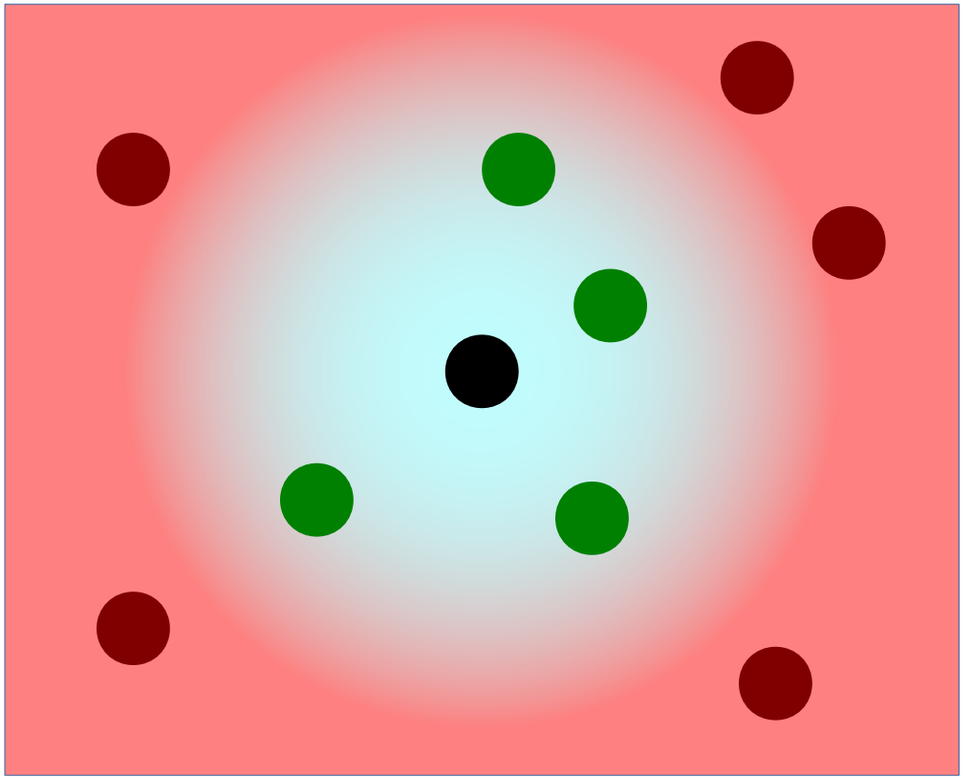


Soft neighborhood - each point is a neighbor with some weight and a non-neighbor with some weight

Neighborhood preservation

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.



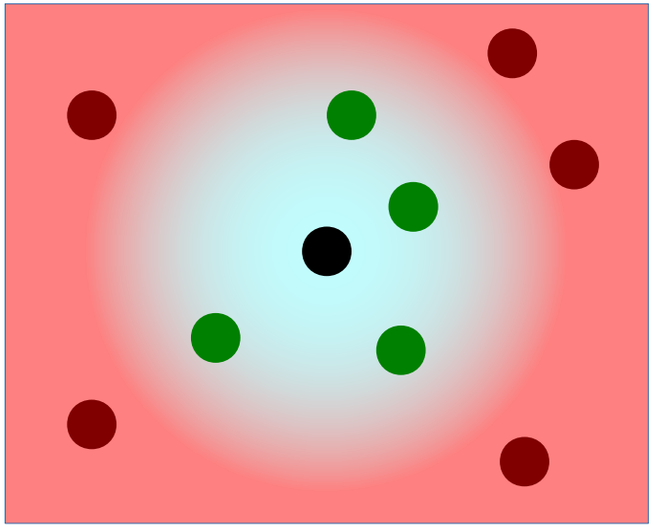
Probabilistic neighborhood

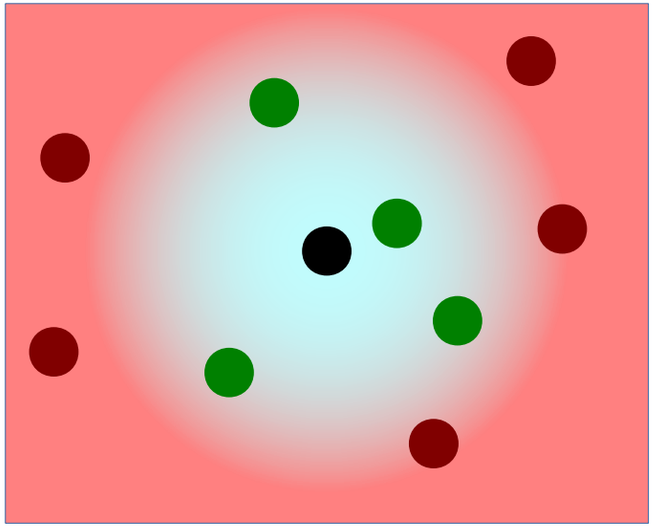
$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}$$

(probability for j to be picked as a neighbor of i **in input space**)

Neighborhood preservation

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!

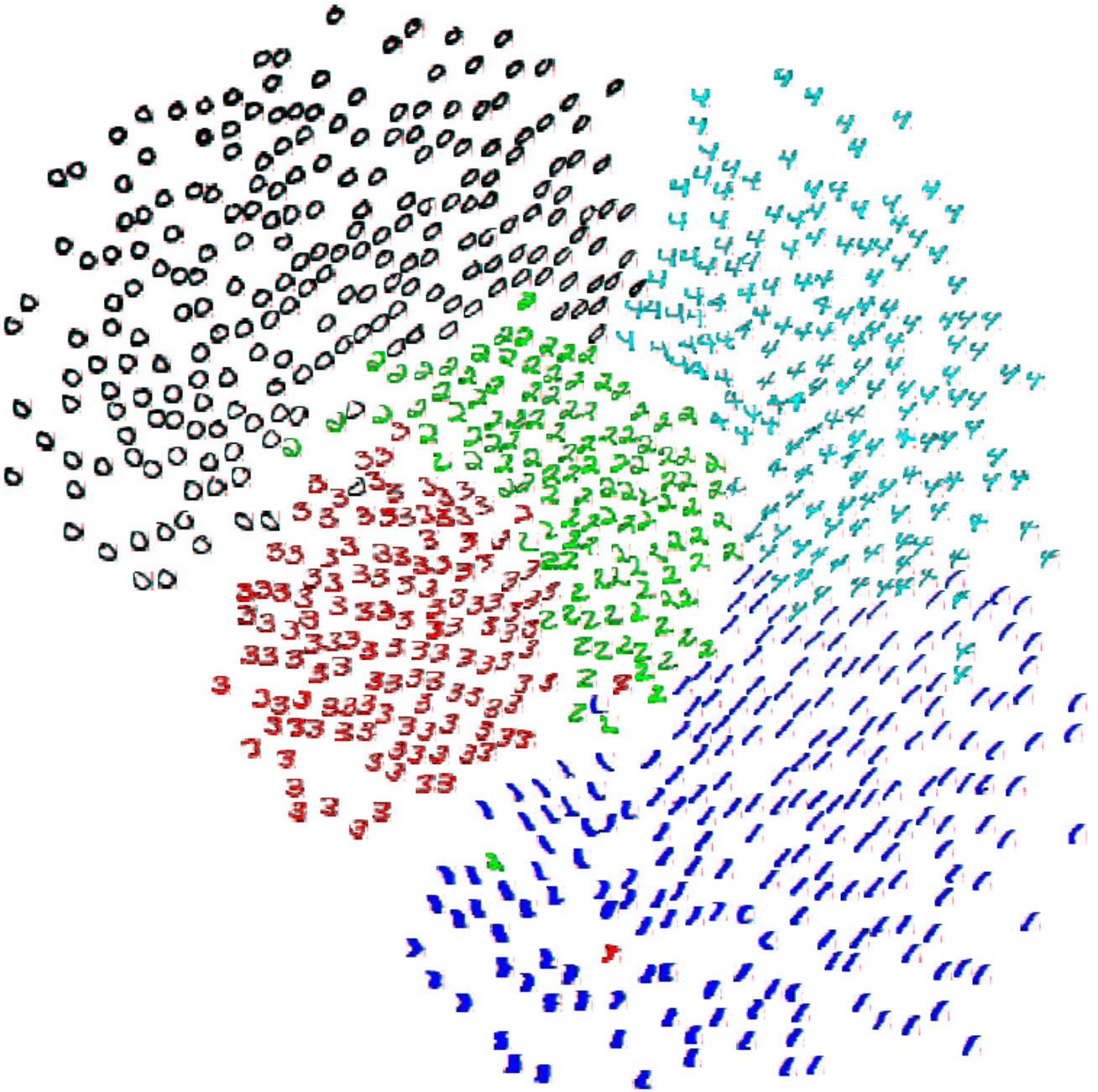
$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} = \sum_i KL(P_i || Q_i)$$

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

SNE applied to grayscale bitmap images of handwritten digits.

Features = pixel values



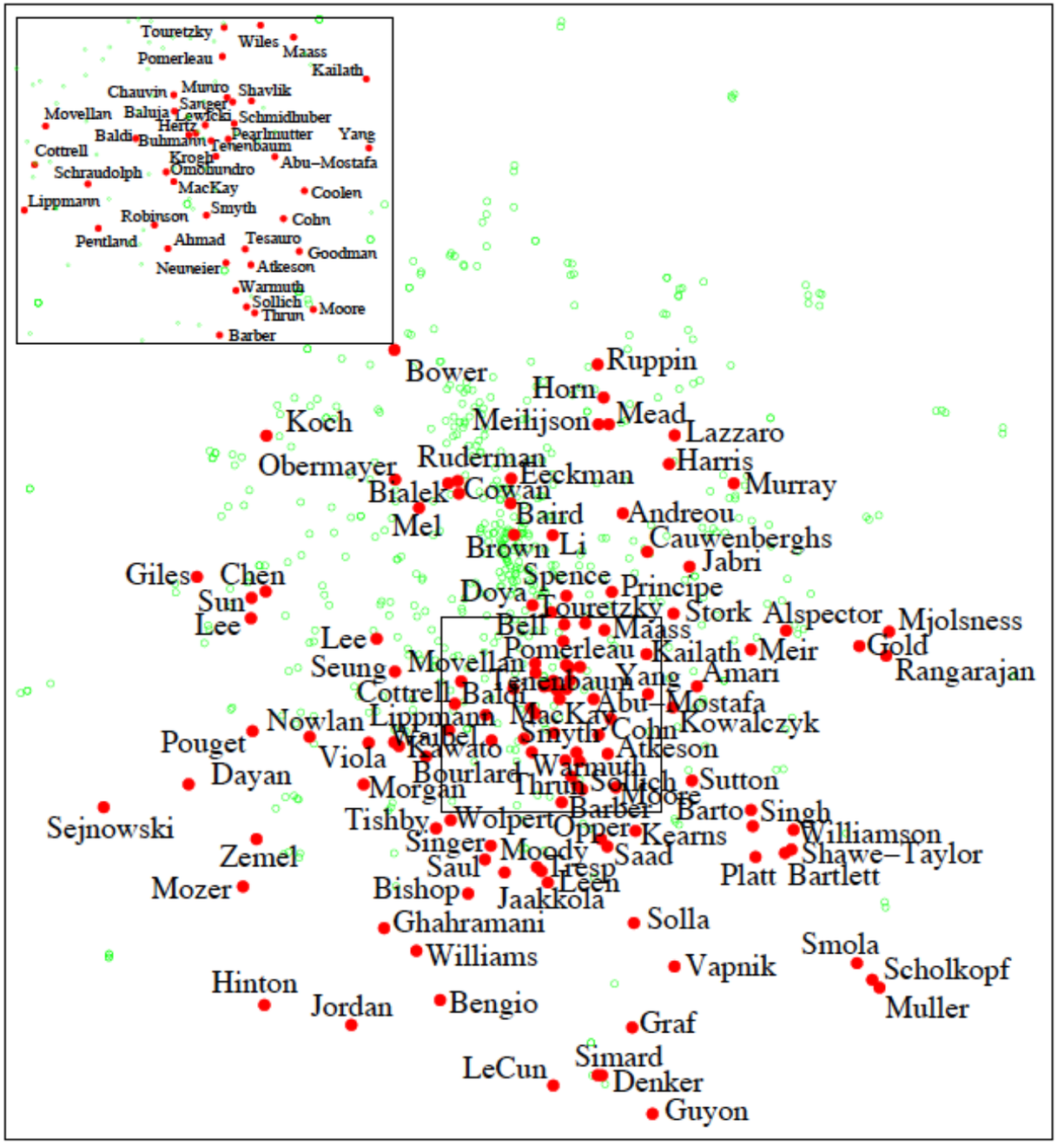
Stochastic neighbor embedding

SNE of NIPS conference authors

Data items = 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

Sometimes if the output space is much lower dimensional than the input space, it can be hard to keep all neighbors close by. This can lead to a **crowding problem** where a lot of data ends up clumped near the middle of a display.

Proposal to avoid crowding: 1. Use a joint distribution over pairs instead of conditional distributions of neighbors, 2. Use a different mathematical form for the output-space distribution.

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

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

Minimizing this objective is called **t-distributed SNE**.

It can give better embeddings. Downside: it can cause distortions because forms of input/output neighborhoods do not match.

Dimensionality reduction for a task

Algorithmic approaches to preserve various things can be seen as guesswork about what will produce the most useful visualization for an analyst.

Manifold learning assumes the manifold can be found and unfolded. Even if manifolds exist in real-life data sets, their dimensionality may be too high to be visualized. -----> The manifold learning assumption can be ill-suited for visualization.

Quality of dimensionality reduction should be quantified for a task of visualization.

Dimensionality reduction for a task

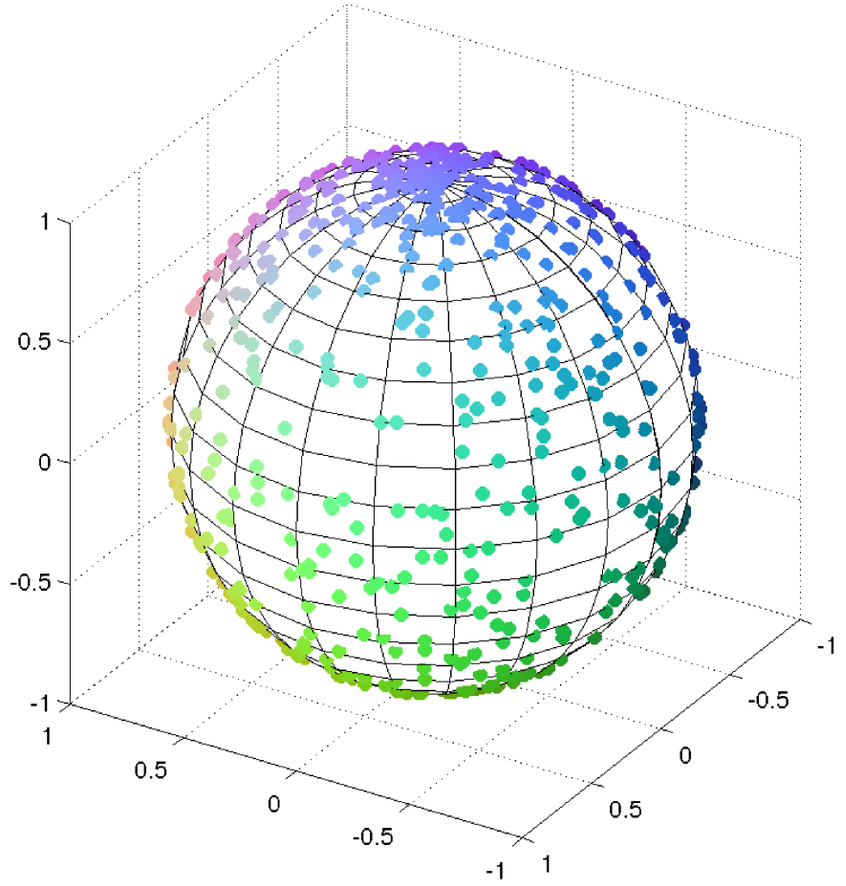
Purpose of visualization (one possible definition): to generate **insights** about the data in the mind of the analyst.
----> hard to quantify what works best for this
-----> instead of “finding insight”, is there some simpler task that we could perform dimensionality reduction for?

Preservation of distances is good if the analyst wants to measure distances between data points. But is that a common task?
------> Maybe in some applications (map projections), not generally

Analyzing neighborhoods can be a subtask in gaining high-insight: e.g. high-level graph structure (hubs, outliers) arises out of local neighborhoods.

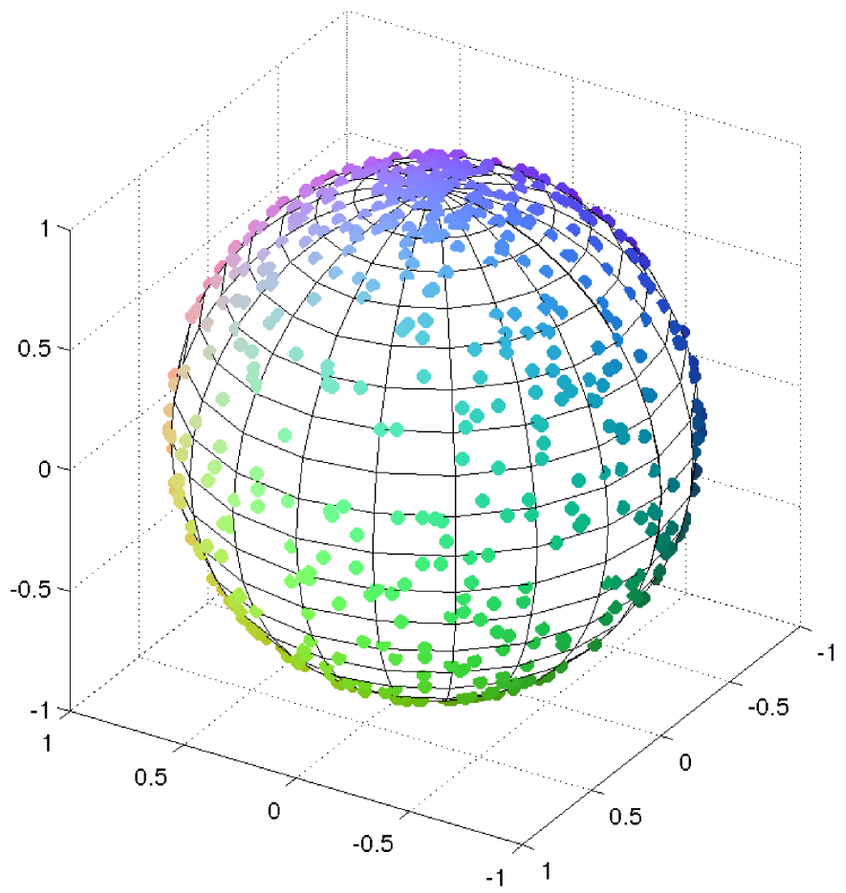
It turns out that preservation of neighborhoods can be formulated as **optimization of an information retrieval task**.

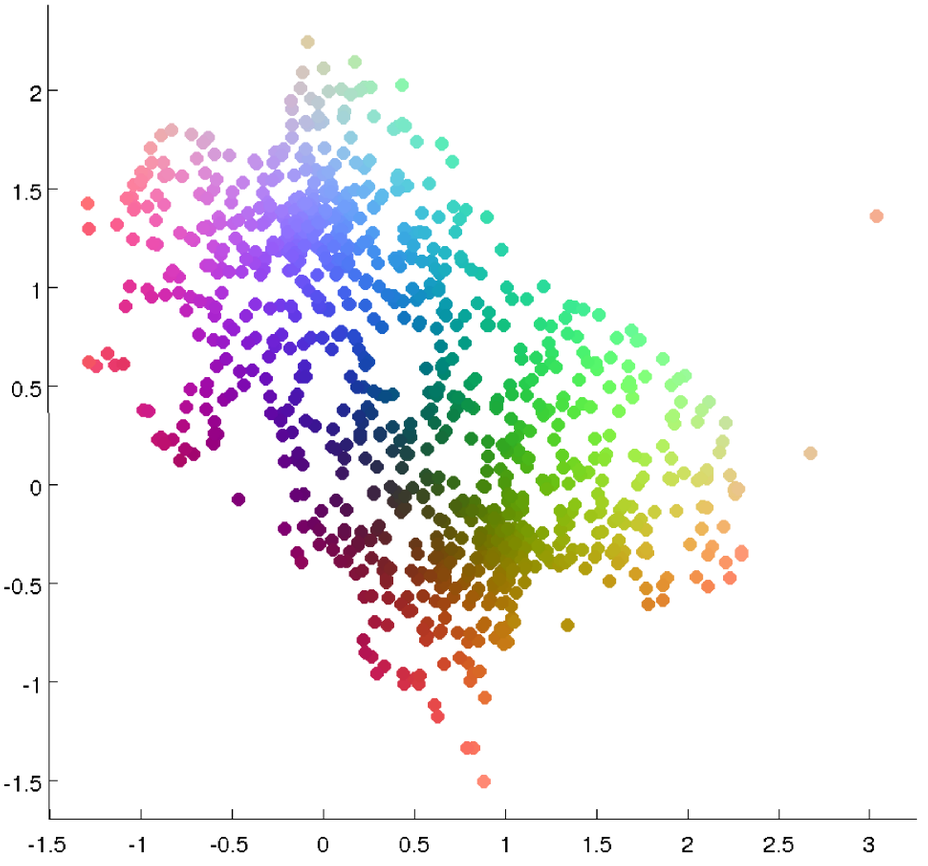
Neighbor retrieval visualizer



Example data set

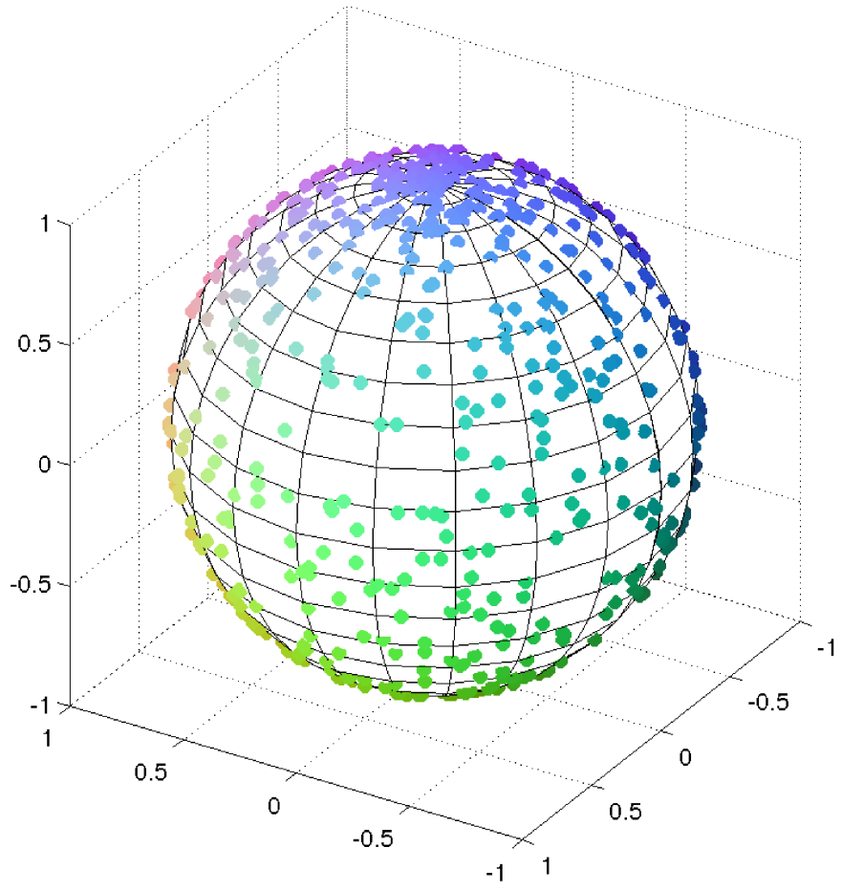
Neighbor retrieval visualizer

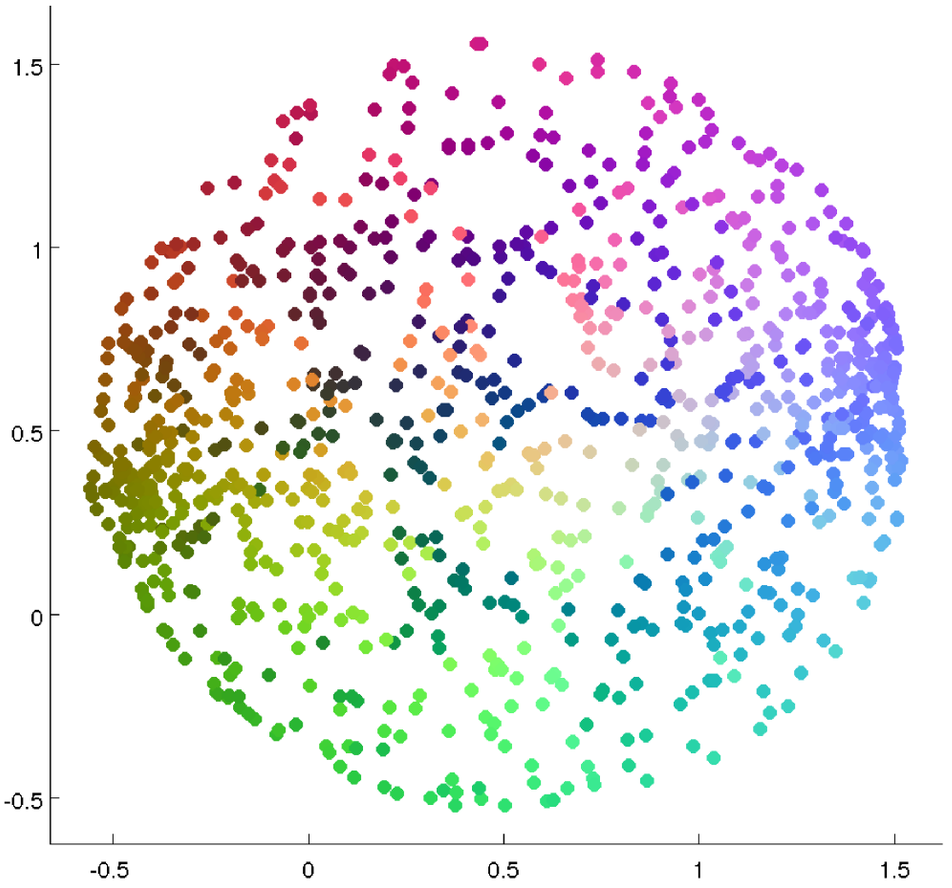




“Orange-peel map”

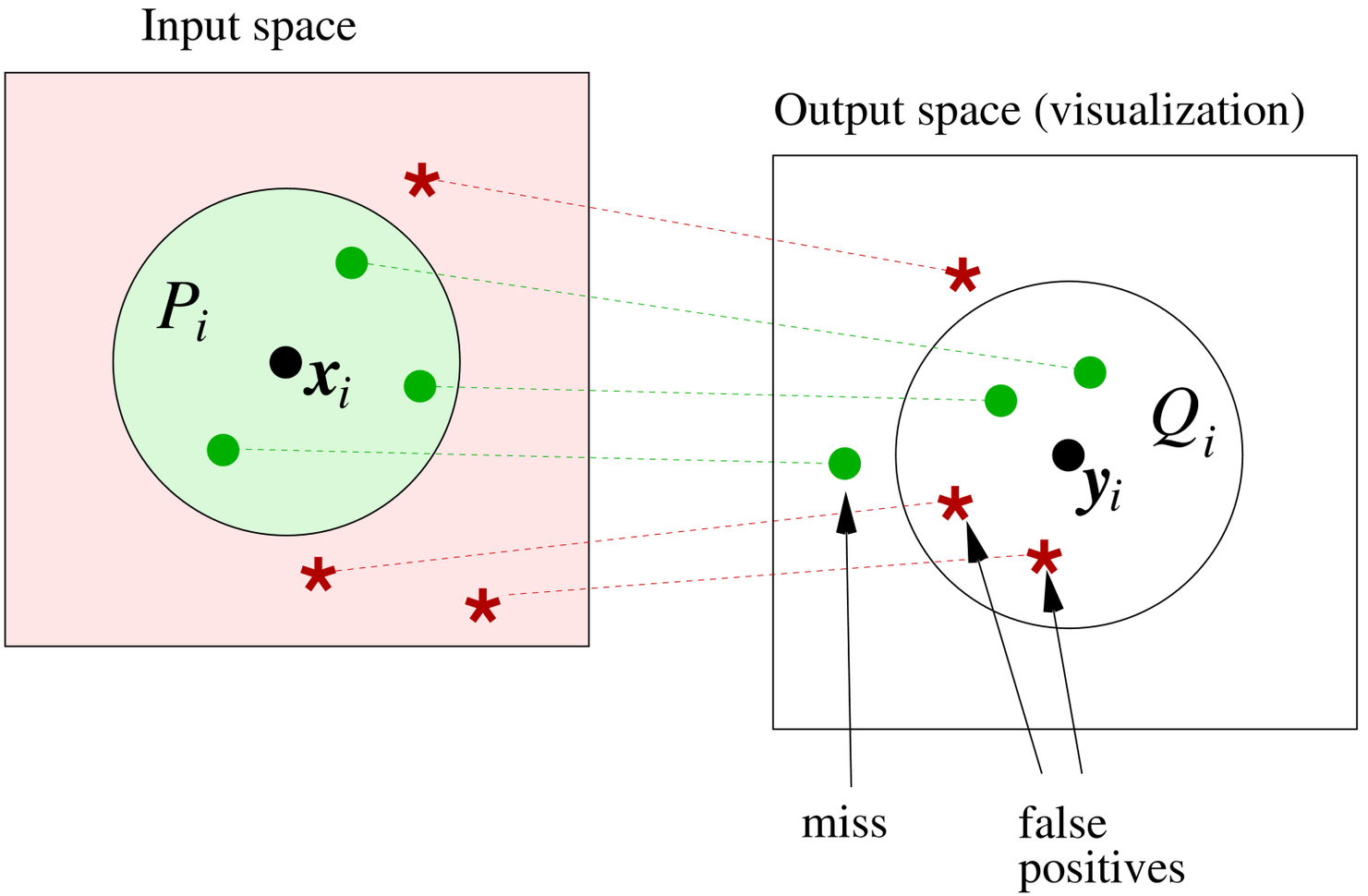
Neighbor retrieval visualizer





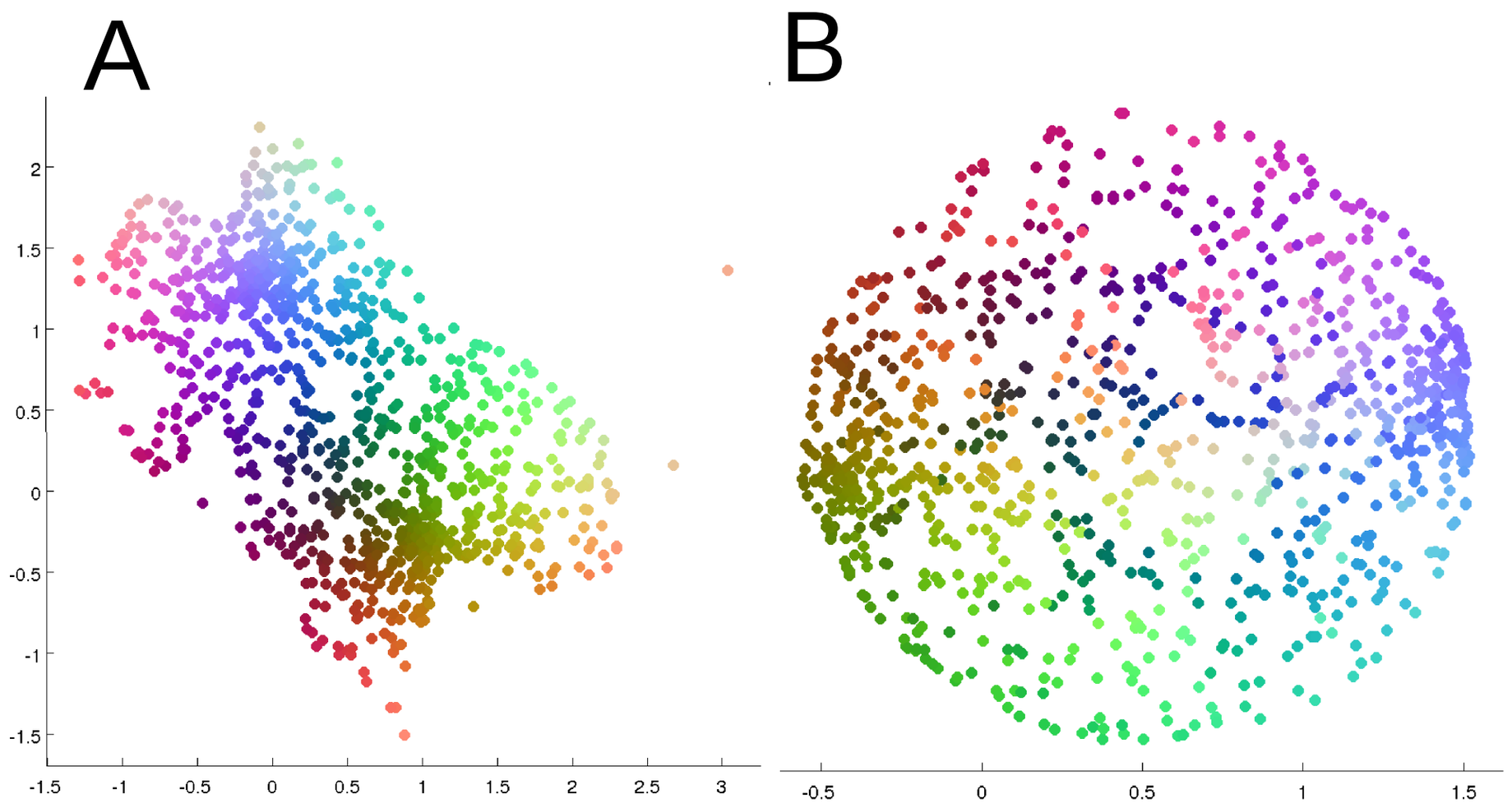
“Squashed-flat sphere”

Neighbor retrieval visualizer



Minimize errors for best information retrieval.

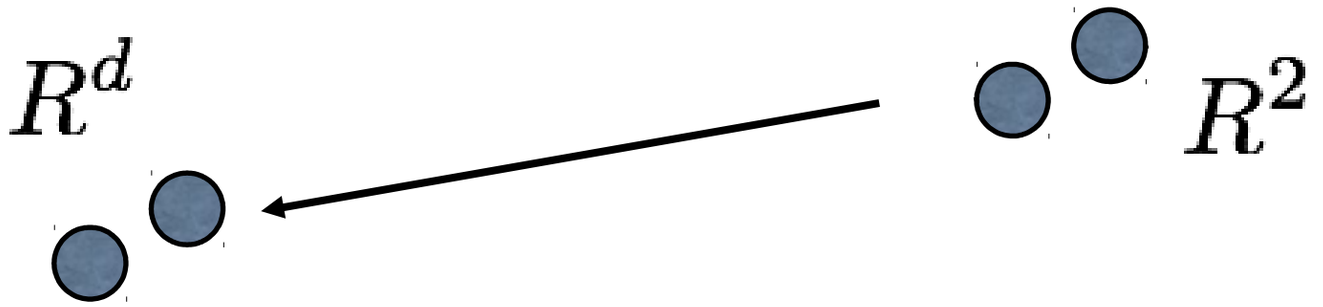
Neighbor retrieval visualizer



Embedding minimizes false positives (falsely retrieved neighbors)

Embedding minimizes misses (neighbors that were not retrieved)

Neighbor retrieval visualizer

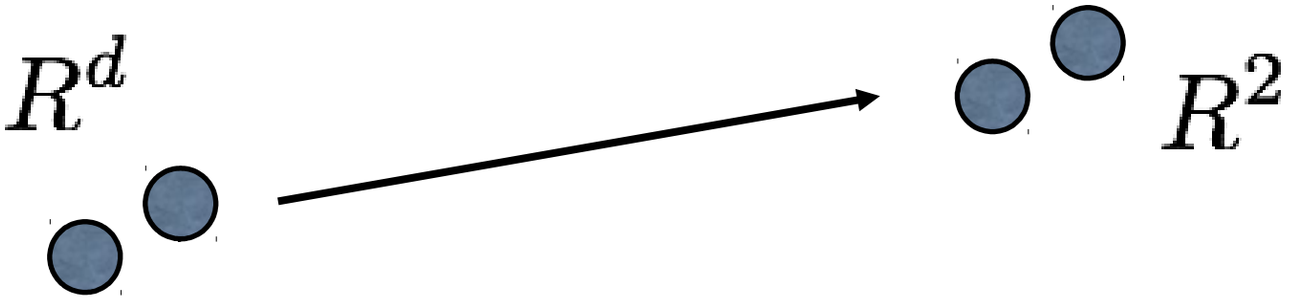


Good Precision: Points that are close in the "reduced" space are close in the original space

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

Proportion of false positives

Neighbor retrieval visualizer



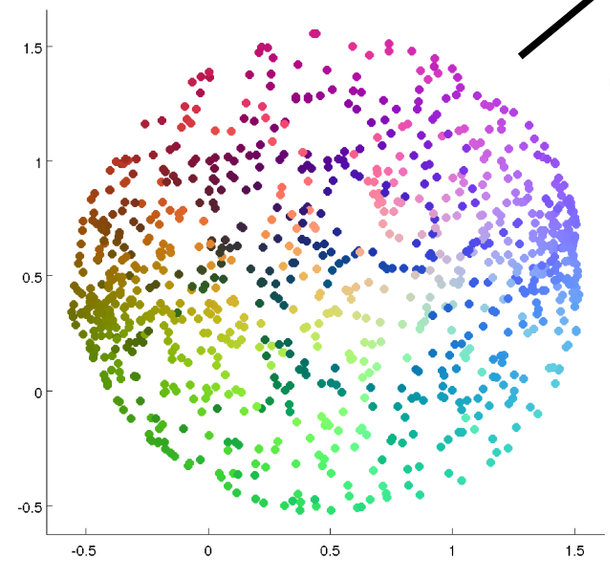
Good Recall: Points that are close in the original space are close in the "reduced" space

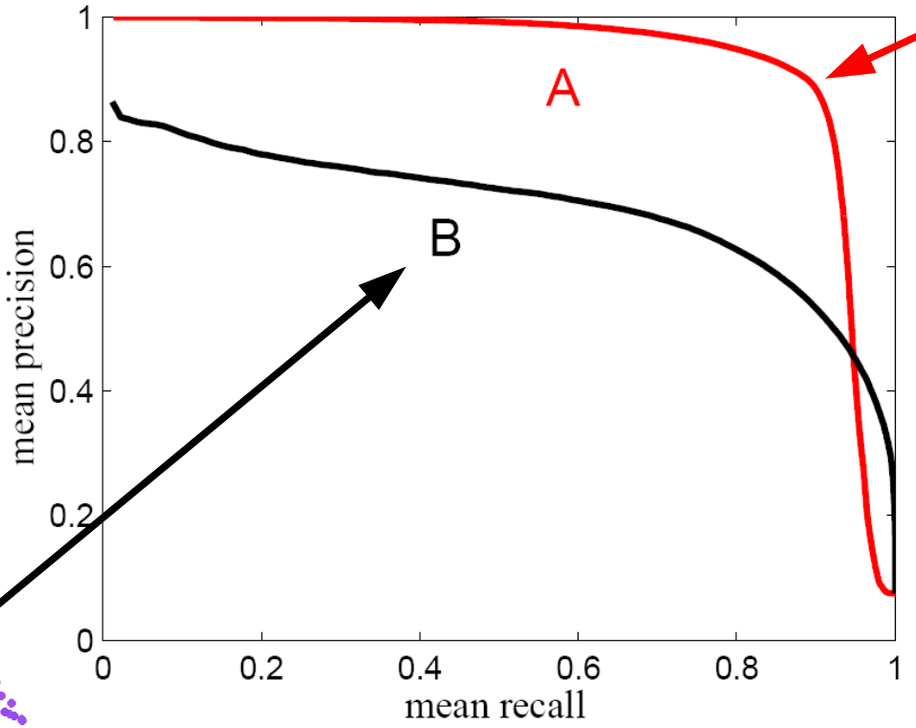
$$1 - recall = \frac{Q_i^C \cap P_i}{|P_i|}$$

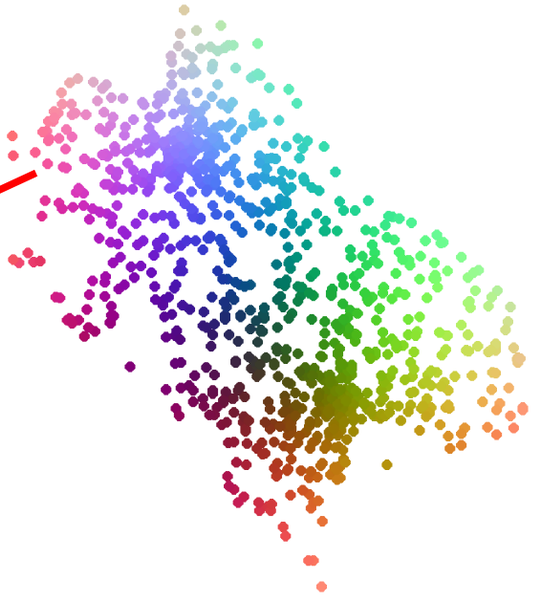
Proportion of missed neighbors

- Sometimes shown as precision-recall curves with respect to size of output neighborhood.
- In general, cannot get both best precision and best recall

Neighbor retrieval visualizer







Neighbor retrieval visualizer

Precision and recall can be extended to probabilistic neighborhoods

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:

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

Neighbor retrieval visualizer

Precision and recall can be extended to probabilistic neighborhoods

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:

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

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 for recall (cost of misses) and precision (cost of false neighbors)

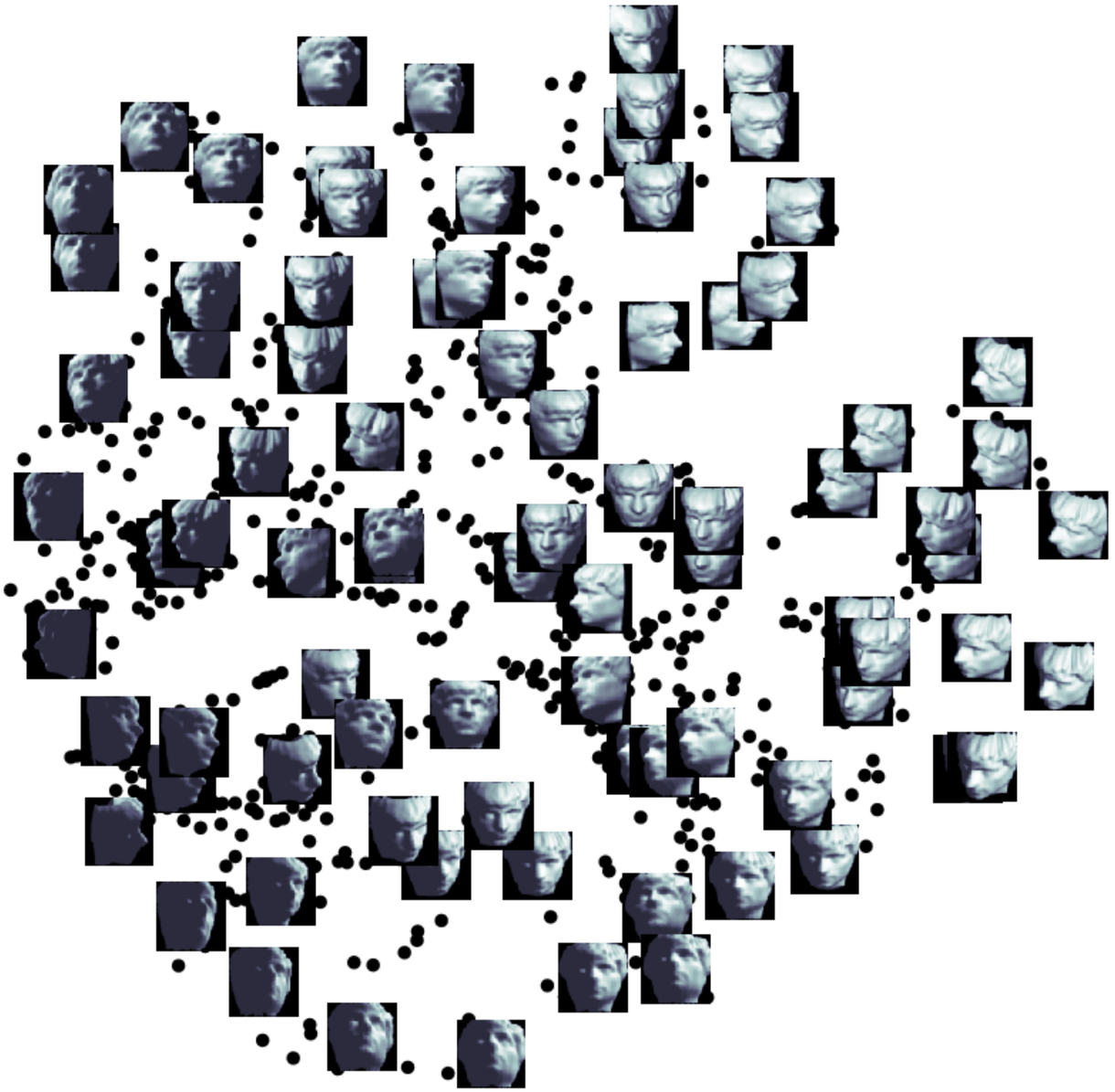
$$E_{\mathrm{NeRV}} = \lambda\mathbb{E}_i[D(p_i,q_i)] + (1-\lambda)\mathbb{E}_i[D(q_i,p_i)]$$

Minimize with respect to output coordinates y_i

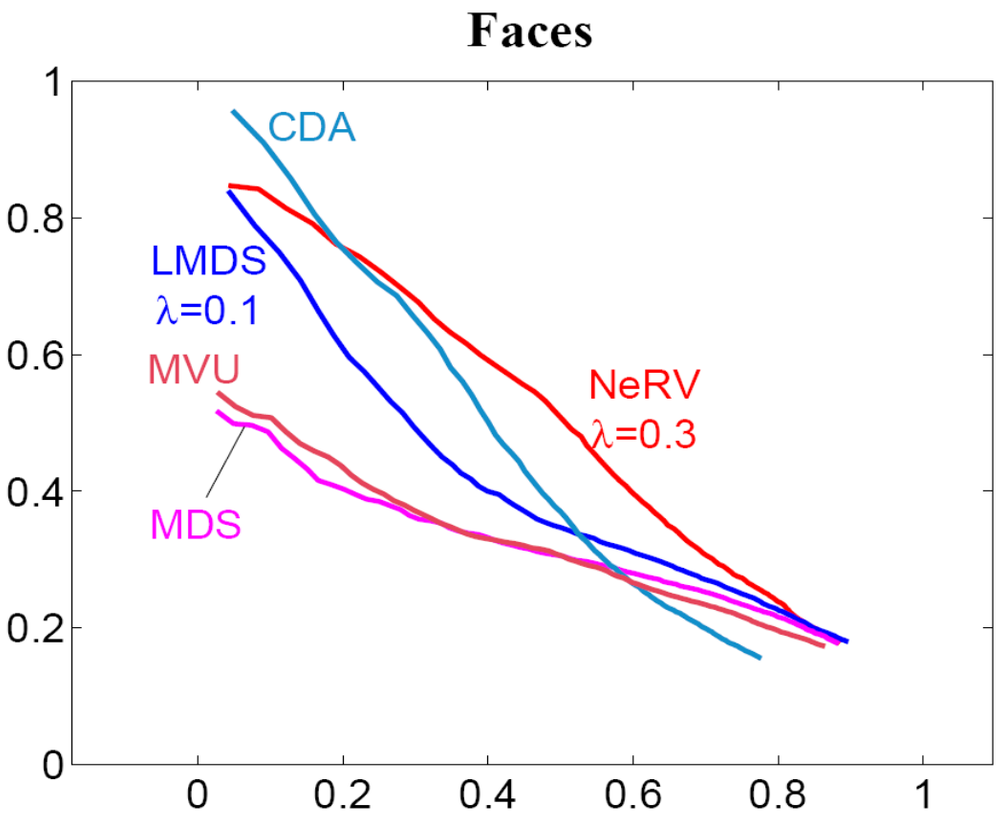
- Cost function directly measures suitability of the visualization for a neigbor retrieval task
- Stochastic neighbor embedding is the special case $\lambda = 0$, only minimizes misses. NeRV minimizes any tradeoff.

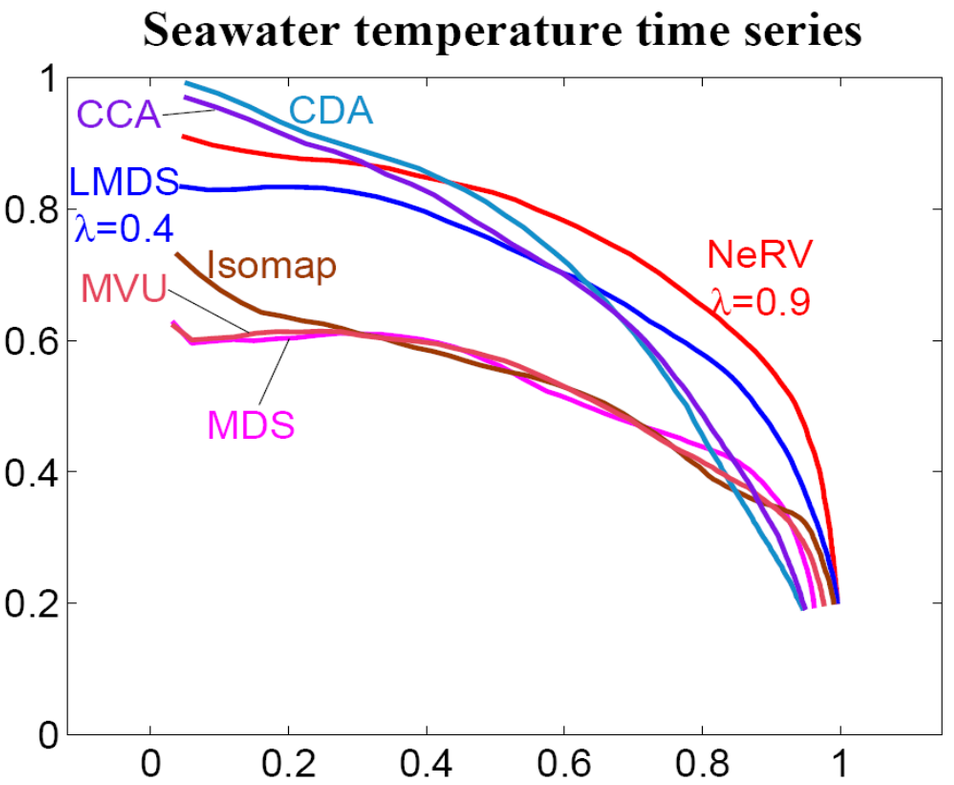
Neighbor retrieval visualizer

- NeRV visualization of face images



Neighbor retrieval visualizer

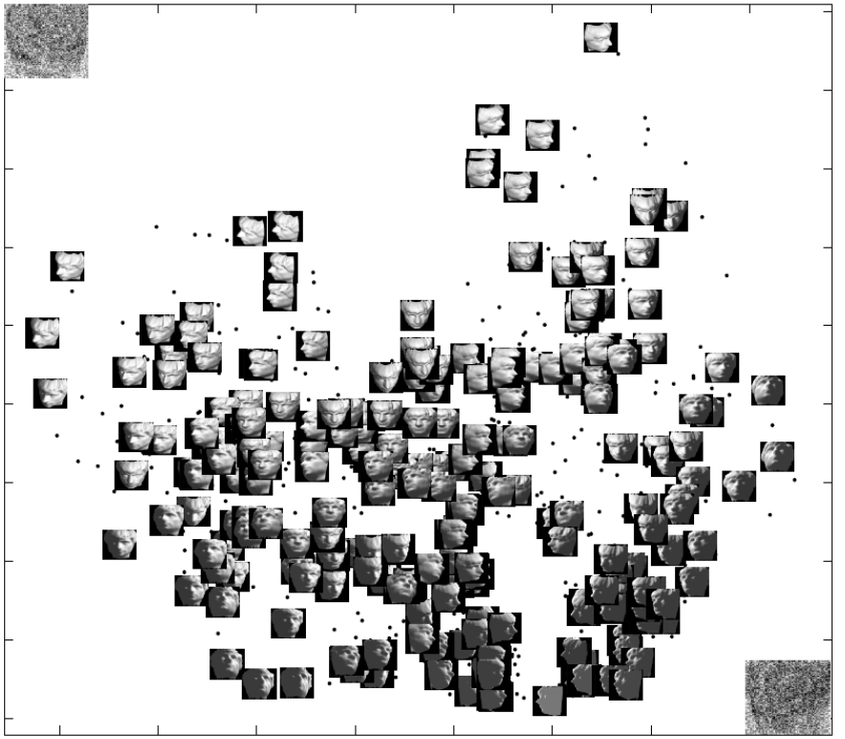




Comparison among several visualization methods in terms of the novel (for visualization) information retrieval measures

Neighbor retrieval visualizer, variants

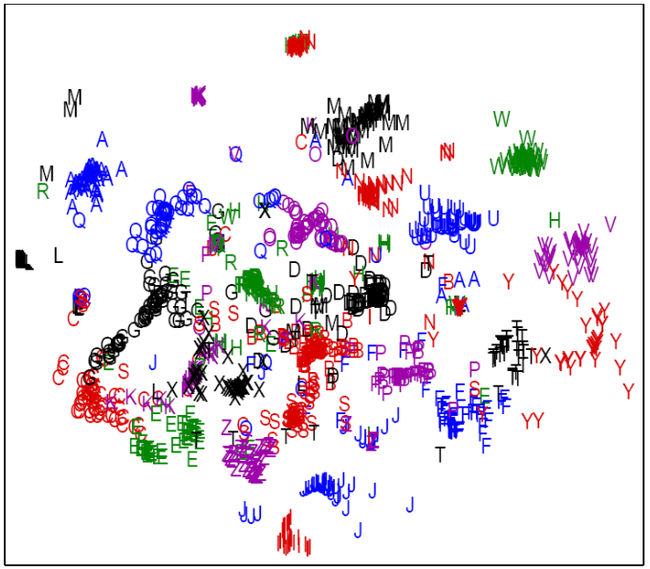
- Linear projection: same cost function, projection restricted to be linear



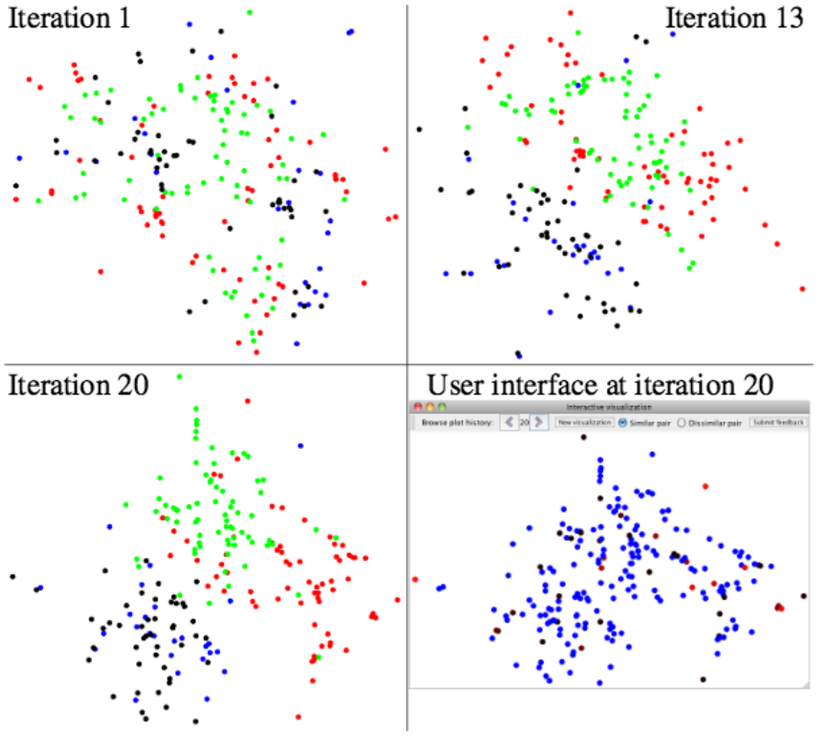
- t-distribution in the output space. t-SNE is a special case minimizing misses only

Neighbor retrieval visualizer, variants

- supervised visualization: define neighbors in a (nonlinear) metric learnt from labels



- interactive visualization: learn a metric from user feedback on which data should be neighbors, visualize iteratively to preserve neighbors in estimated metric

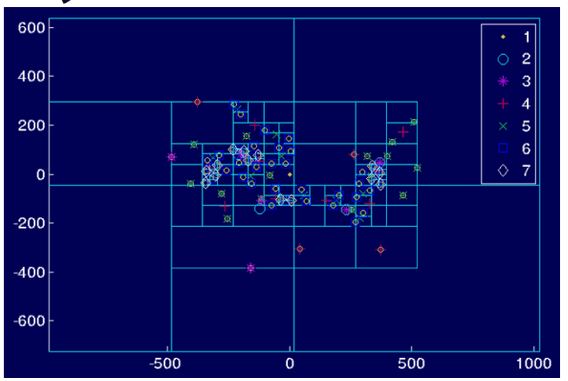


Neighbor retrieval, fast computation

Neighbor embedding is state of the art but takes quadratic time.

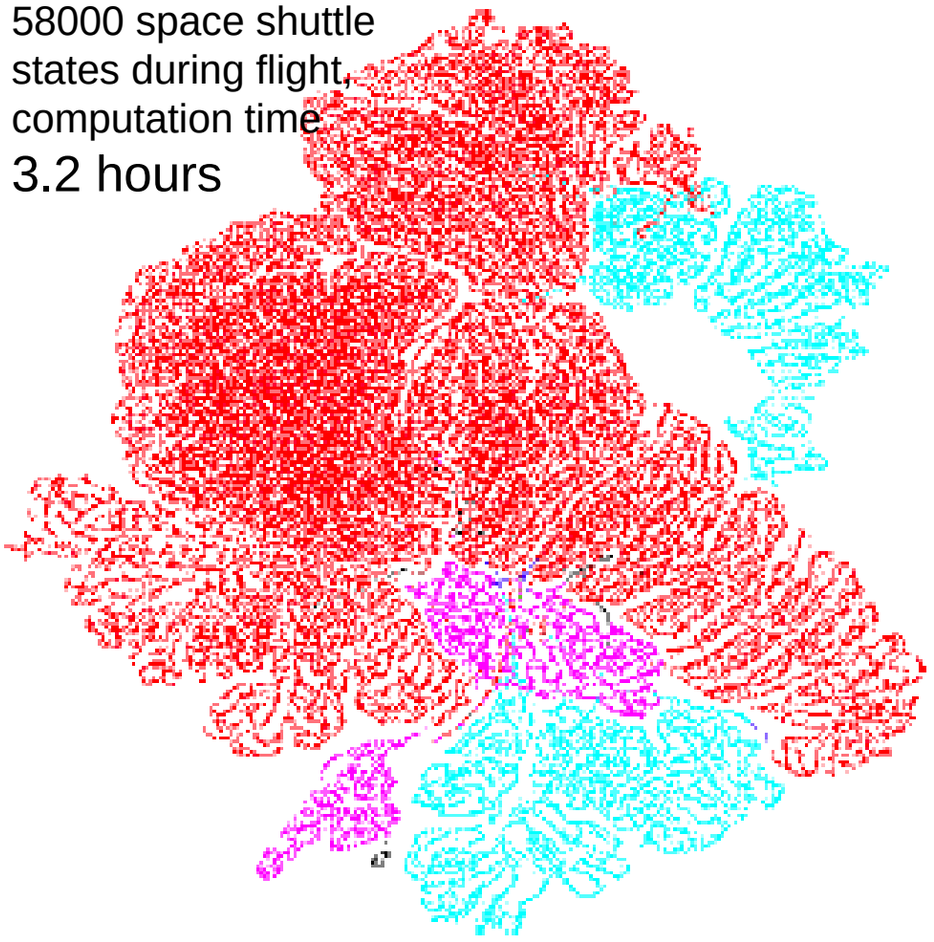
New O(N logN) methods based on Barnes-Hut approximation:

sums over far-away neighbors approximated by cluster-means

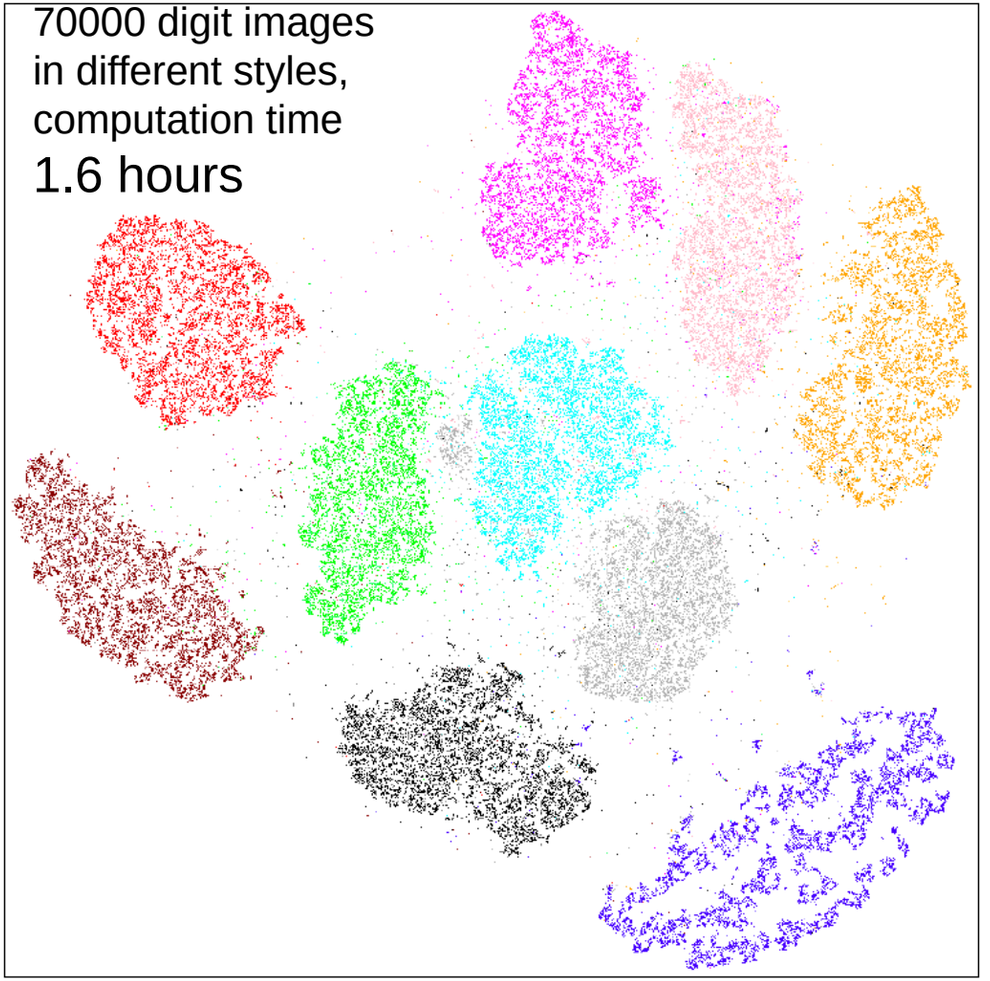


(Yang, Peltonen and Kaski, ICML 2013)

58000 space shuttle states during flight, computation time 3.2 hours



70000 digit images in different styles, computation time 1.6 hours



Summary

- **Principal Component Analysis**: maximizes variance
- **Linear Discriminant Analysis**: maximizes ratio of between-class to within-class scatter
- **Multidimensional Scaling**: preserves distances
- **Sammon mapping**: preserves small original distances
- **Curvilinear Component Analysis**: preserves small output distances
- **Isomap** and **Curvilinear Distance Analysis**: preserves geodesic distances
- **Self-Organizing Map**: maps data to regular grid
- **Neighbor retrieval visualizer**: preserves neighborhoods, optimized for information retrieval. **Stochastic neighbor embedding** and **t-SNE** are special cases minimizing misses only.