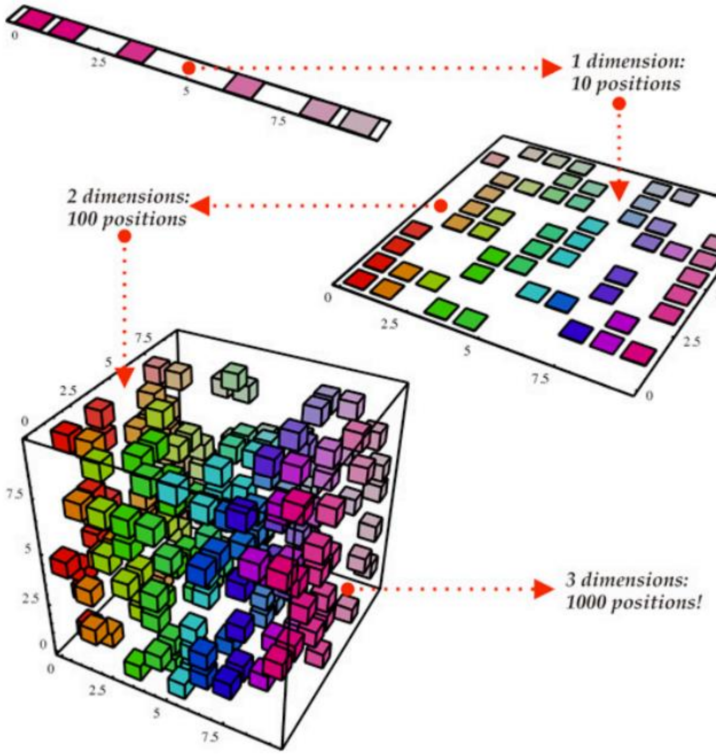
CPSC 340:
Machine Learning and Data Mining

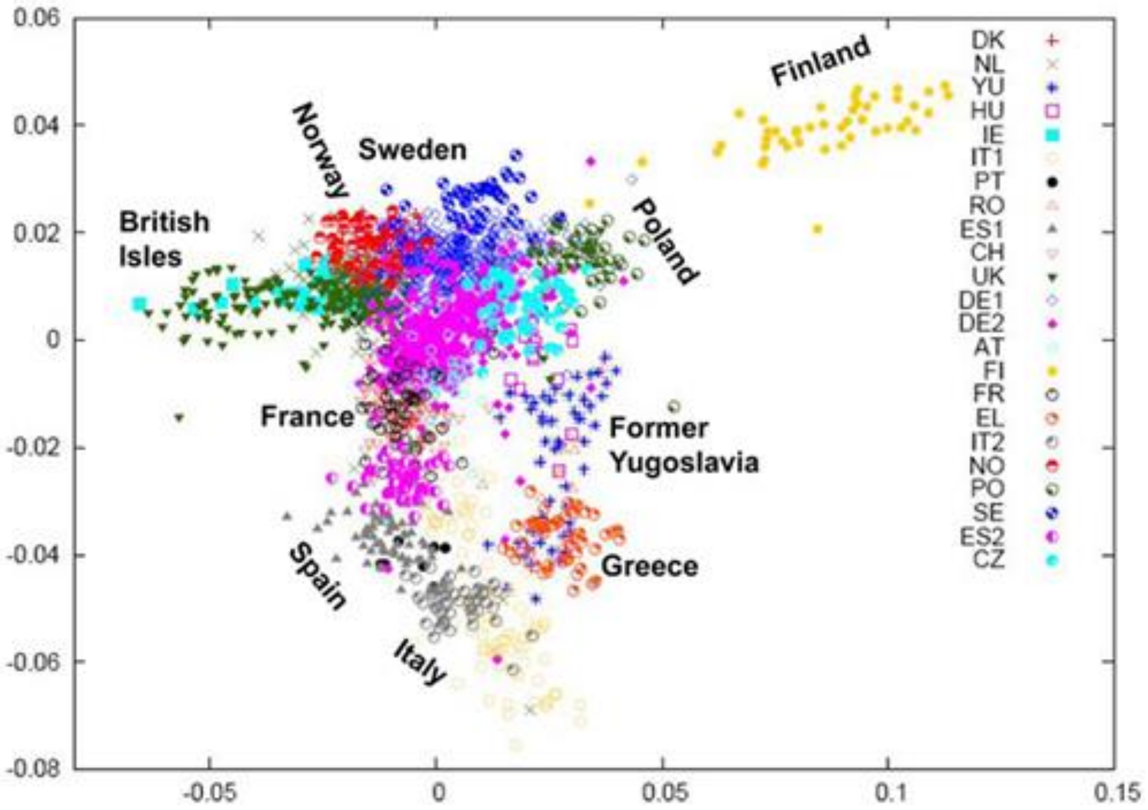
Multi-Dimensional Scaling

Fall 2018

Latent-Factor Models for Visualization

- PCA takes features x_i and gives k-dimensional approximation z_i.
- If k is small, we can use this to visualize high-dimensional data.

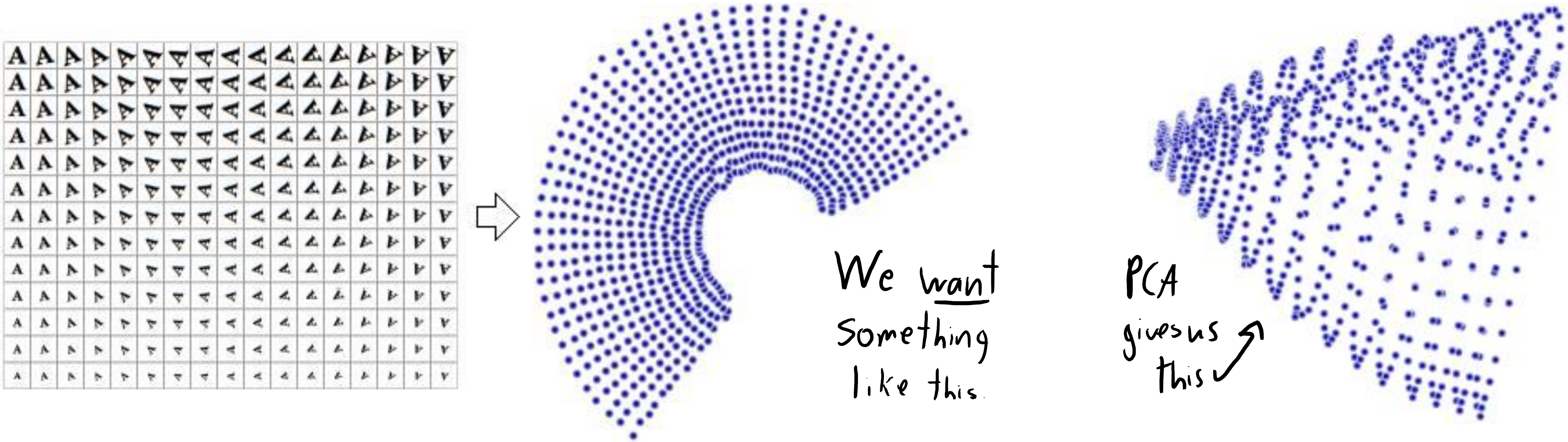




http://www.turingfinance.com/artificial-intelligence-and-statistics-principal-component-analysis-and-self-organizing-maps/
http://scienceblogs.com/gnxp/2008/08/14/the-genetic-map-of-europe/

Motivation for Non-Linear Latent-Factor Models

- But PCA is a parametric linear model
- PCA may not find obvious low-dimensional structure.



- We could use change of basis or kernels: but still need to pick basis.

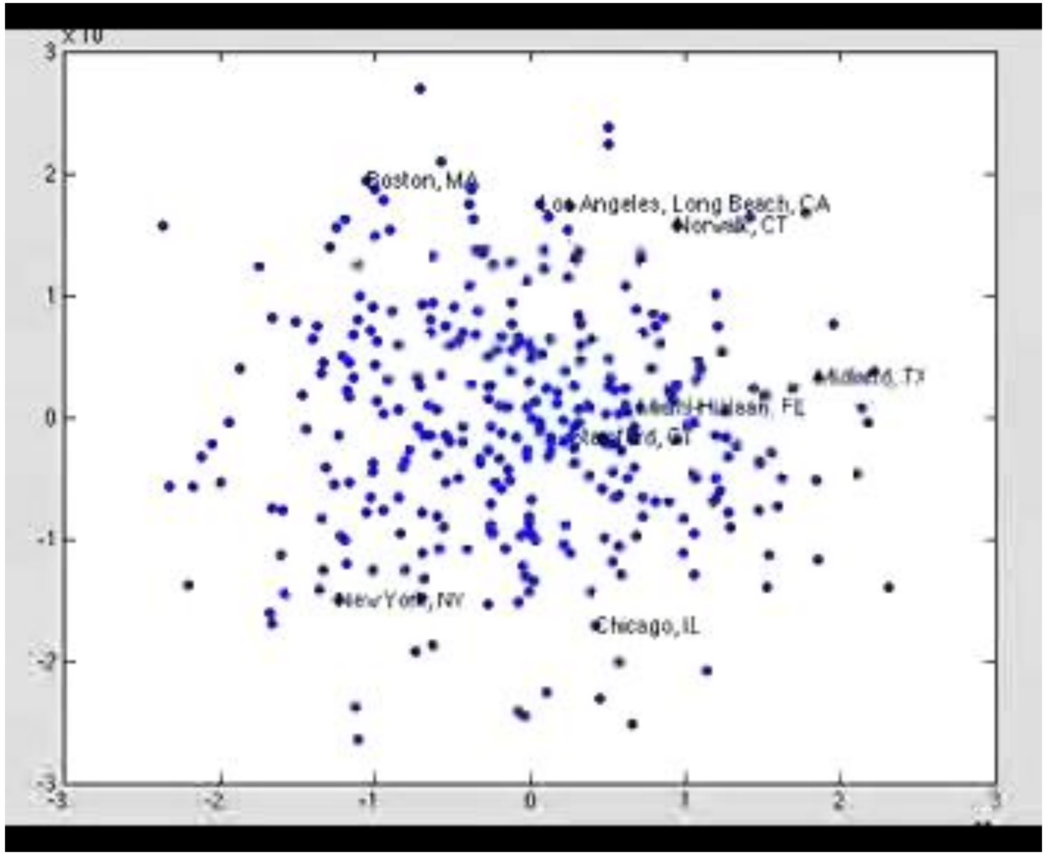
https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

Multi-Dimensional Scaling

- PCA for visualization:
  - We're using PCA to get the location of the z_i values.
  - We then plot the z_i values as locations in a scatterplot.
- Multi-dimensional scaling (MDS) is a crazy idea:
  - Let's directly optimize the pixel locations of the z_i values.
    - "Gradient descent on the points in a scatterplot".
  - Needs a "cost" function saying how "good" the z_i locations are.
    - Traditional MDS cost function:

$$f(Z) = \underbrace{\sum_{i=1}^{n} \sum_{j=i+1}^{n}}_{\text{sum over pairs of examples}} \overbrace{\Big(\underbrace{\|z_i - z_j\|}_{\text{distance in scatterplot}} - \underbrace{\|x_i - x_j\|}_{\text{Distance between points in original 'd' dimensions}} \Big)^2}^{\text{"Try to make scatterplot distances match high-dimensional distance"}}$$

MDS Method ("Sammon Mapping") in Action

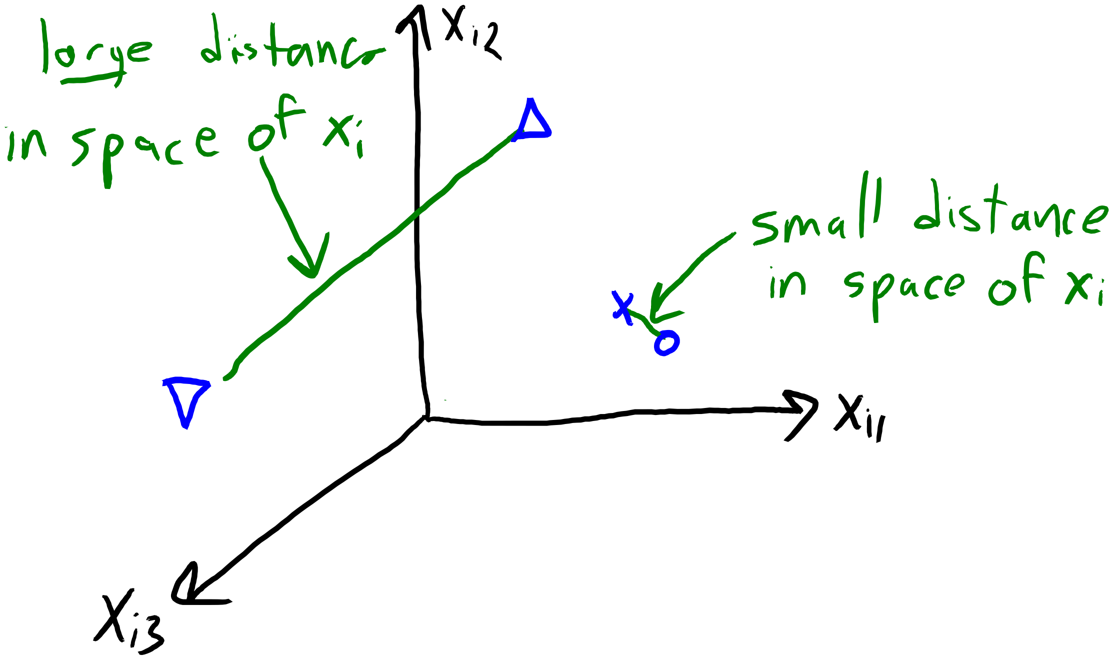


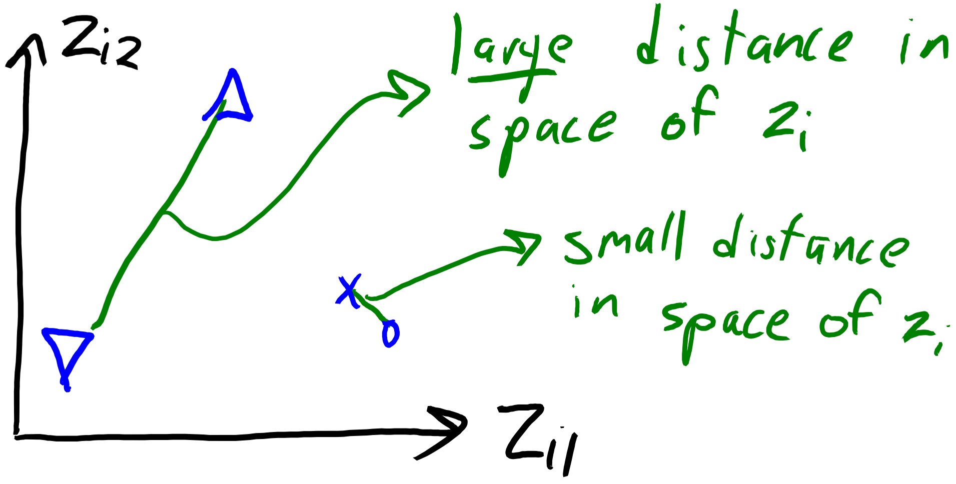
- Unfortunately, MDS often does not work well in practice.

Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS):
  - Directly optimize the final locations of the z_i values.

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\|z_i - z_j\| - \|x_i - x_j\| \right)^2$$



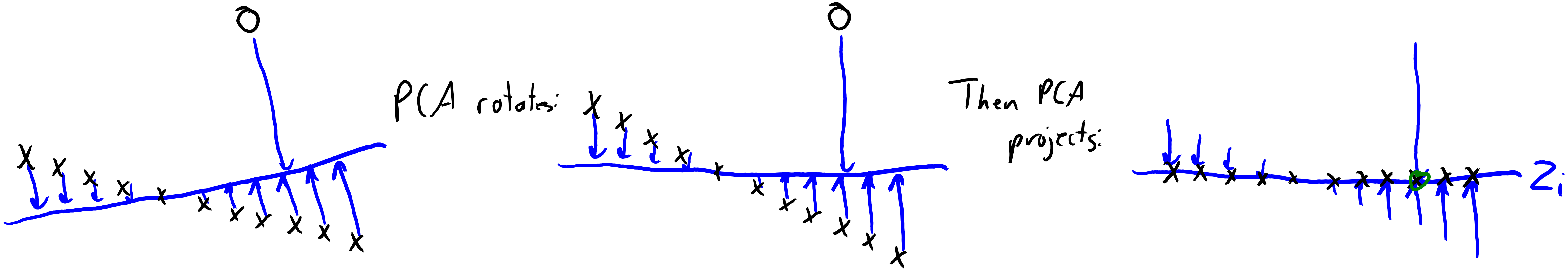


Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS):
  - Directly optimize the final locations of the z_i values.

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\|z_i - z_j\| - \|x_i - x_j\| \right)^2$$

  - Non-parametric dimensionality reduction and visualization:
    - No 'W': just trying to make z_i preserve high-dimensional distances between x_i.

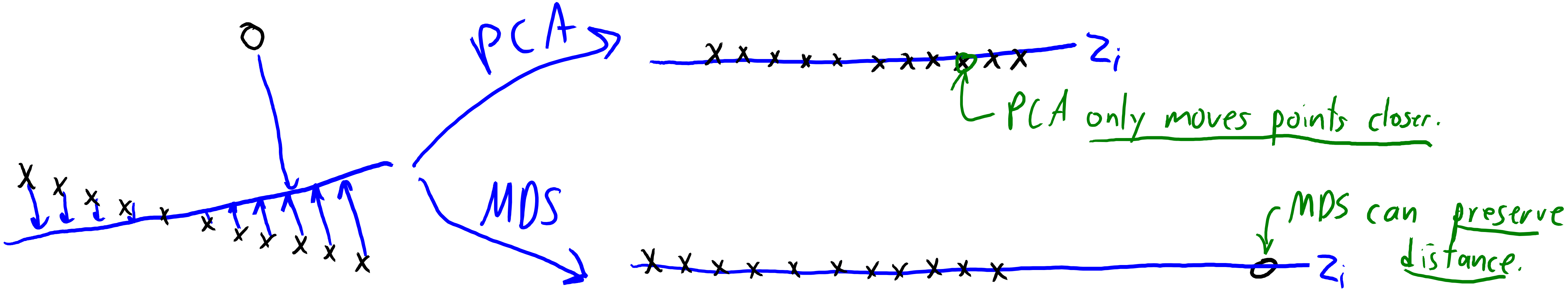


Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS):
  - Directly optimize the final locations of the z_i values.

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\|z_i - z_j\| - \|x_i - x_j\| \right)^2$$

  - Non-parametric dimensionality reduction and visualization:
    - No ‘W’: just trying to make z_i preserve high-dimensional distances between x_i.

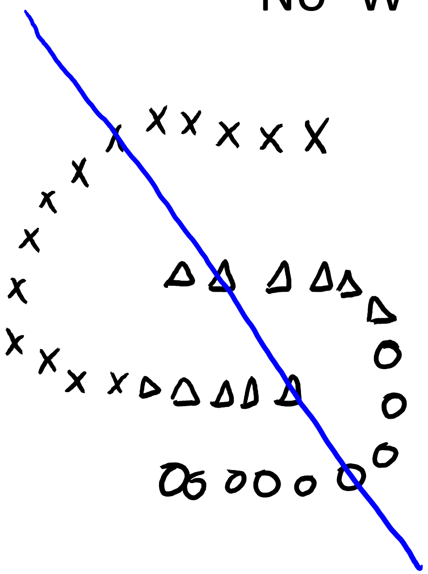


Multi-Dimensional Scaling

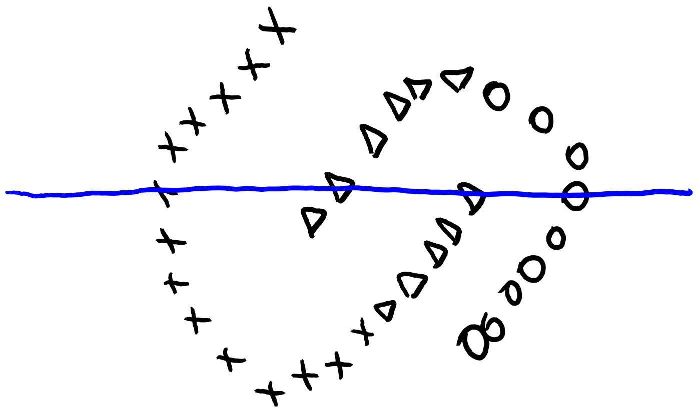
- Multi-dimensional scaling (MDS):
  - Directly optimize the final locations of the z_i values.

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\|z_i - z_j\| - \|x_i - x_j\| \right)^2$$

  - Non-parametric dimensionality reduction and visualization:
    - No ‘W’: just trying to make z_i preserve high-dimensional distances between x_i.



PCA rotation:



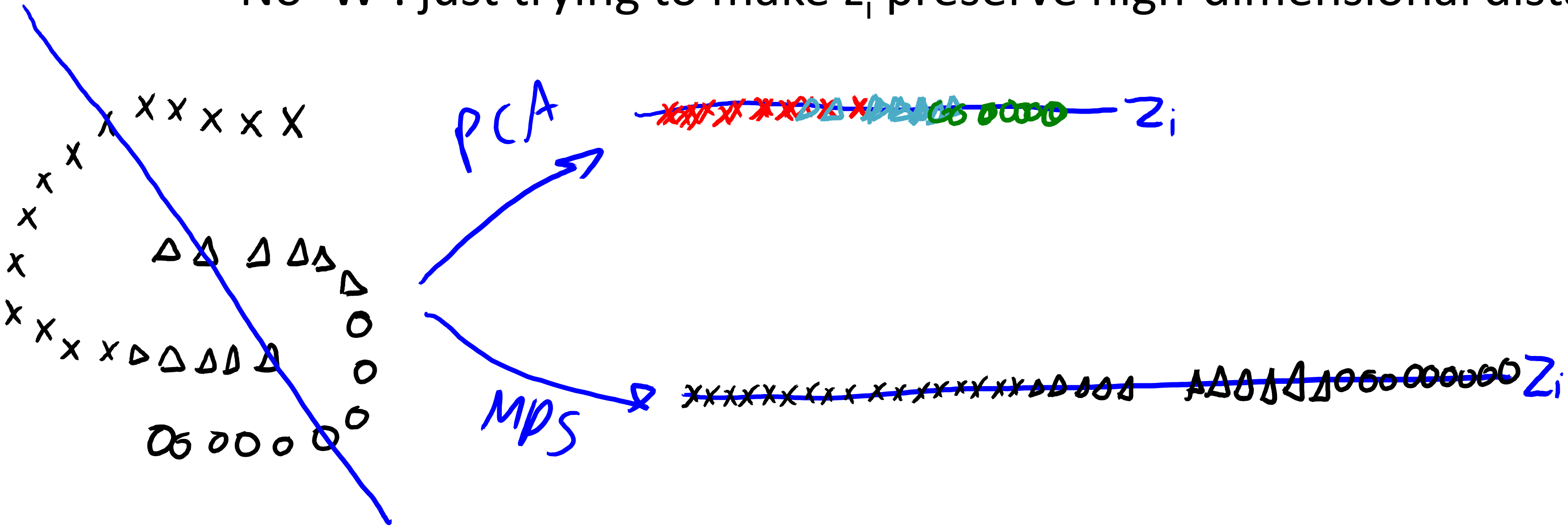
PCA projection:

Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS):
  - Directly optimize the final locations of the z_i values.

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\|z_i - z_j\| - \|x_i - x_j\| \right)^2$$

  - Non-parametric dimensionality reduction and visualization:
    - No 'W': just trying to make z_i preserve high-dimensional distances between x_i.



Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS):
  - Directly optimize the final locations of the z_i values.

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\|z_i - z_j\| - \|x_i - x_j\| \right)^2$$

- Cannot use SVD to compute solution:
  - Instead, do gradient descent on the z_i values.
  - You "learn" a scatterplot that tries to visualize high-dimensional data.
  - Not convex and sensitive to initialization.
    - And solution is not unique due to various factors like translation and rotation.

Different MDS Cost Functions

- MDS default objective: squared difference of Euclidean norms:

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\|z_i - z_j\| - \|x_i - x_j\|)^2$$

- But we can make z_i match different distances/similarities:

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_3(d_2(z_i, z_j) - d_1(x_i, x_j))$$

  - Where the functions are not necessarily the same:
    - d_1 is the high-dimensional distance we want to match.
    - d_2 is the low-dimensional distance we can control.
    - d_3 controls how we compare high-/low-dimensional distances.

Different MDS Cost Functions

- MDS default objective function with general distances/similarities:

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_3(d_2(z_i, z_j) - d_1(x_i, x_j))$$

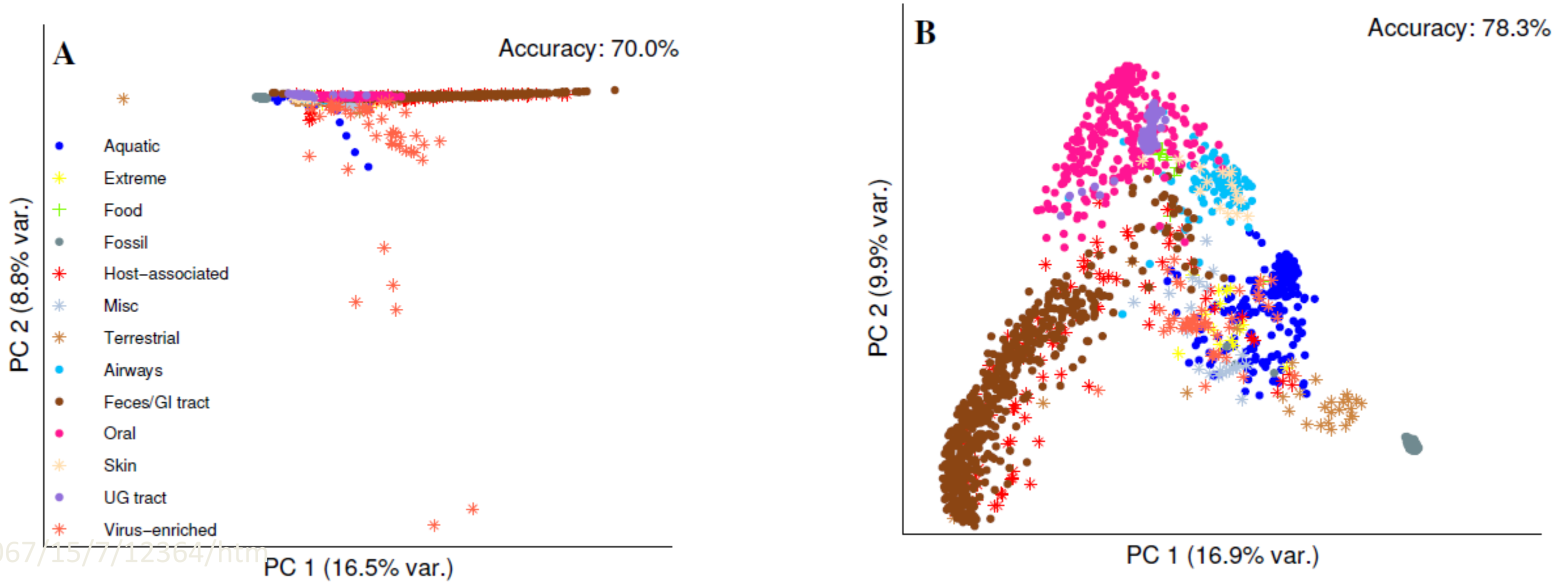
- "Classic" MDS uses $d_1(x_i,x_j) = x_i^T x_j$ and $d_2(z_i,z_j) = z_i^T z_j$.
  - We obtain PCA in this special case (centered x_i, d_3 as the squared L2-norm).
  - Not a great choice because it's a linear model.

Different MDS Cost Functions

- MDS default objective function with general distances/similarities:

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_3(d_2(z_i, z_j) - d_1(x_i, x_j))$$

- Another possibility: $d_1(x_i,x_j) = ||x_i - x_j||_1$ and $d_2(z_i,z_j) = ||z_i - z_j||$.
  - The z_i approximate the high-dimensional L_1-norm distances.



Sammon's Mapping

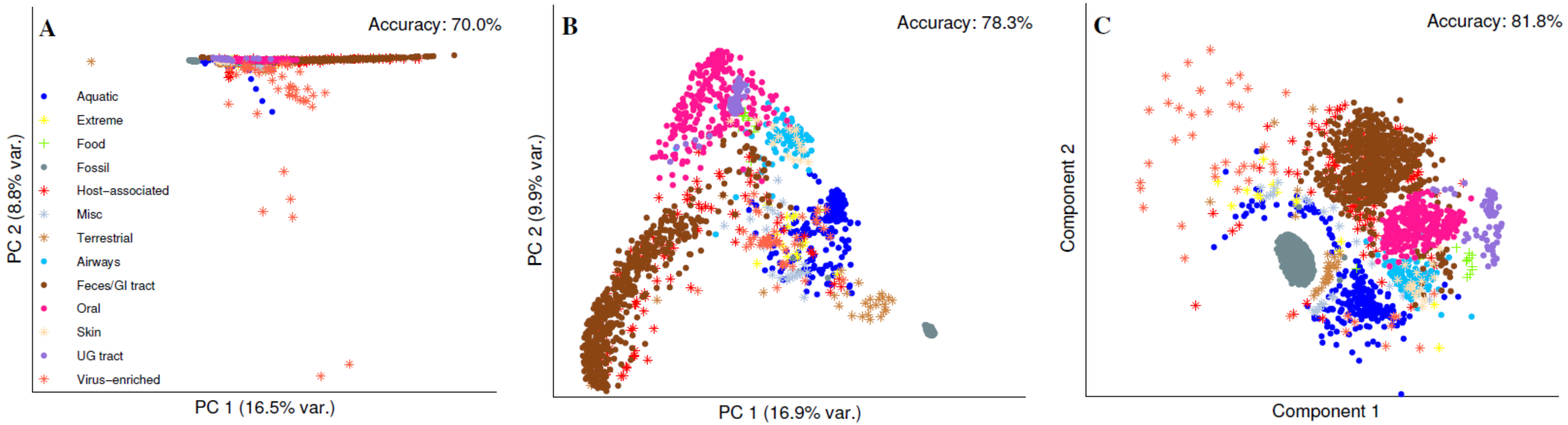
- Challenge for most MDS models: they focus on large distances.
  - Leads to "crowding" effect like with PCA.
- Early attempt to address this is Sammon's mapping:
  - Weighted MDS so large/small distances are more comparable.

$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\frac{d_2(z_i, z_j) - d_1(x_i, x_j)}{d_1(x_i, x_j)} \right)^2$$

  - Denominator reduces focus on large distances.

Sammon's Mapping

- Challenge for most MDS models: they focus on large distances.
  – Leads to "crowding" effect like with PCA.
- Early attempt to address this is Sammon's mapping:
  – Weighted MDS so large/small distances are more comparable.

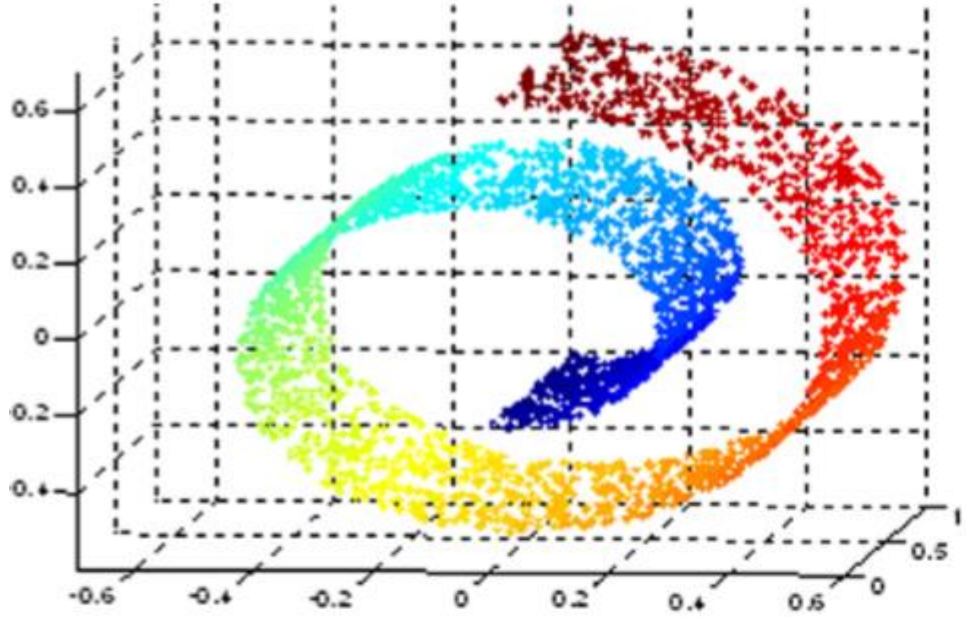


(pause)

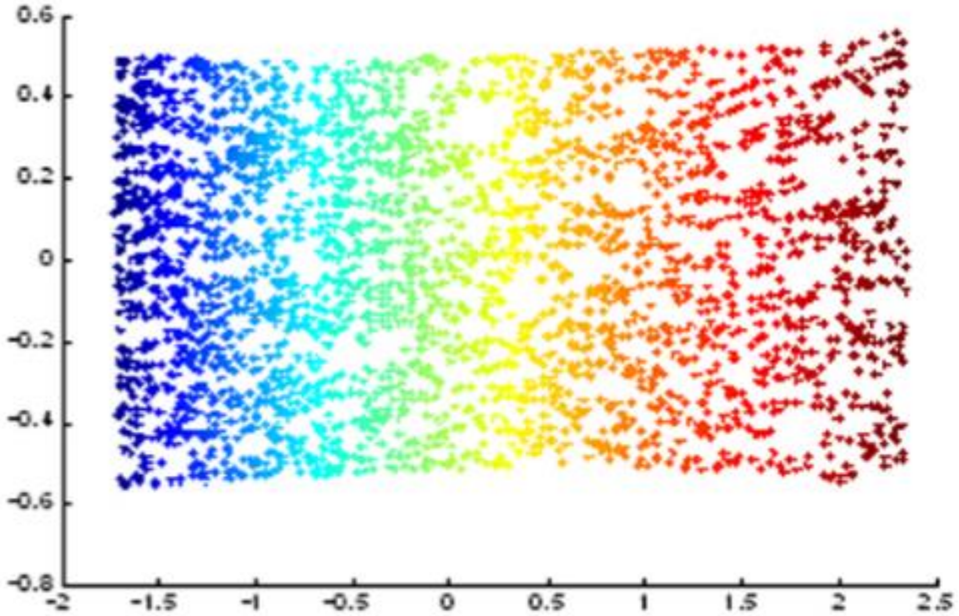
Learning Manifolds

- Consider data that lives on a low-dimensional "manifold".
- Example is the 'Swiss roll':

Original data



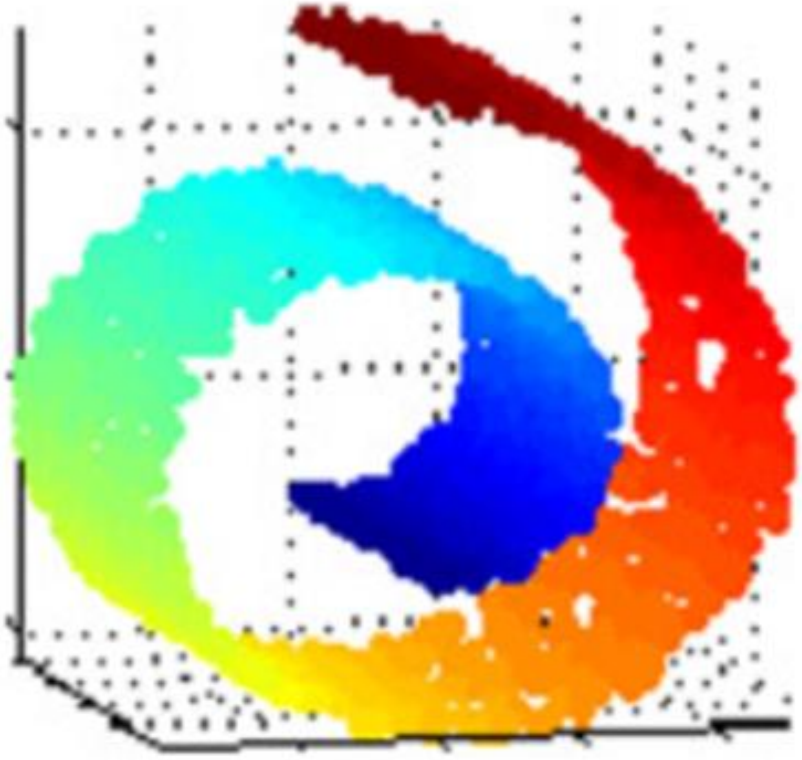
Two-dimensional manifold



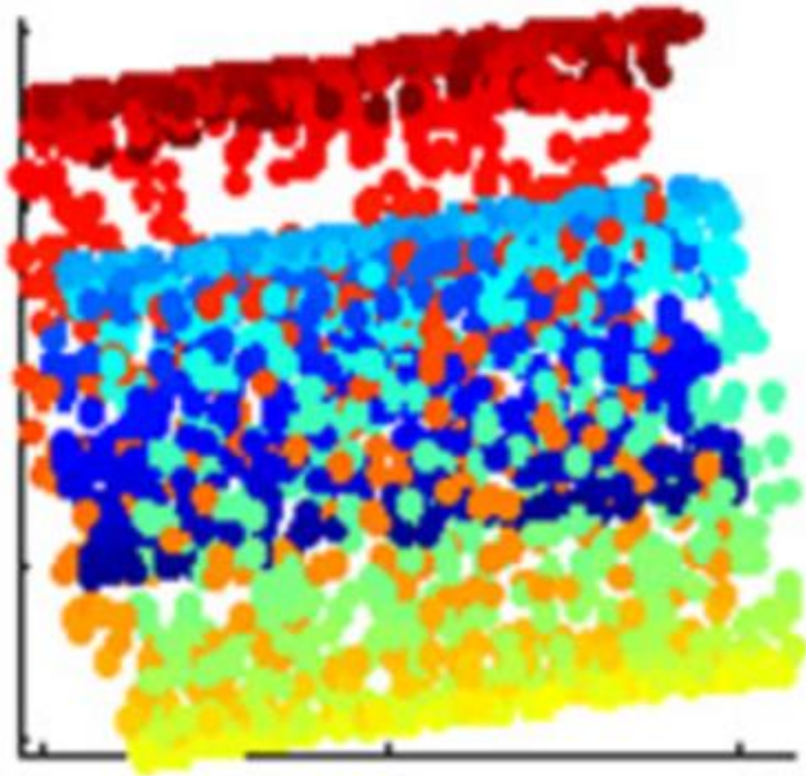
http://www.biomedcentral.com/content/pdf/1471-2105-13-S7-S3.pdf

Learning Manifolds

- Consider data that lives on a low-dimensional "manifold".
  - With usual distances, PCA/MDS will not discover non-linear manifolds.



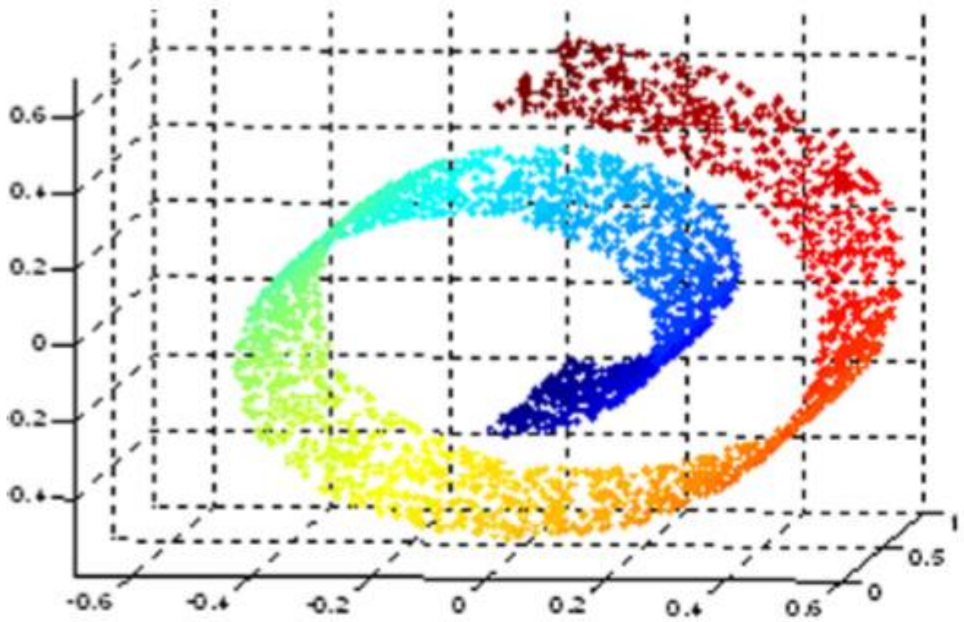
Original data

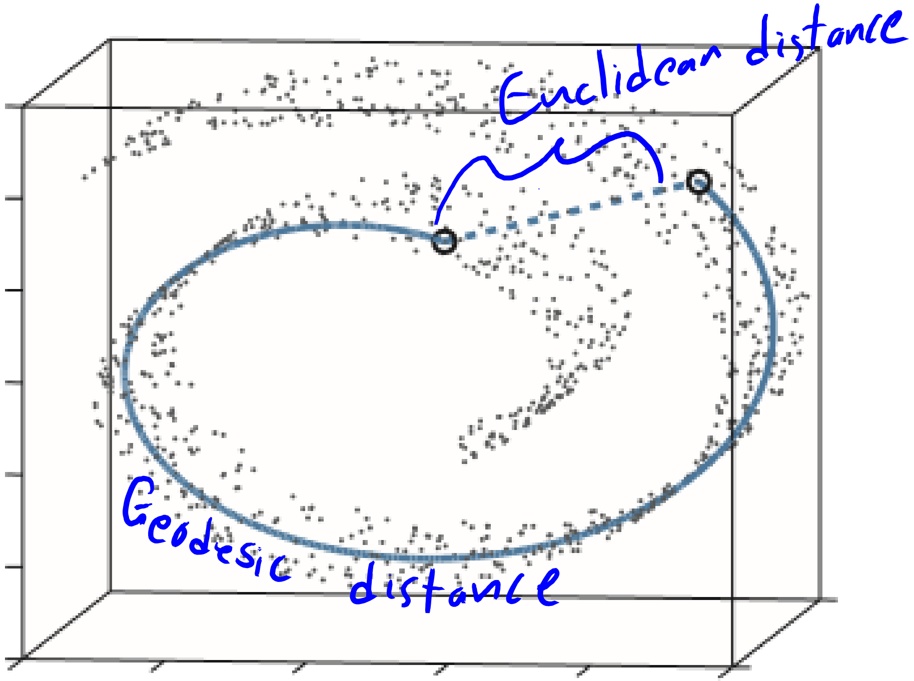


PCA

Learning Manifolds

- Consider data that lives on a low-dimensional "manifold".
  - With usual distances, PCA/MDS will not discover non-linear manifolds.
- We need geodesic distance: the distance *through* the manifold.

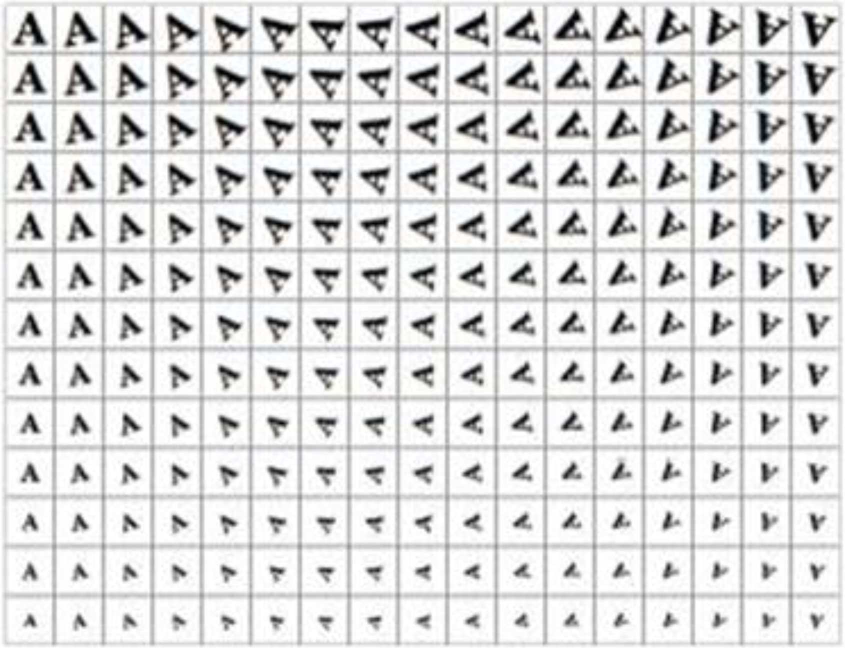




http://www.biomedcentral.com/content/pdf/1471-2105-13-S7-S3.pdf
http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

Manifolds in Image Space

- Consider slowly-varying transformation of image:

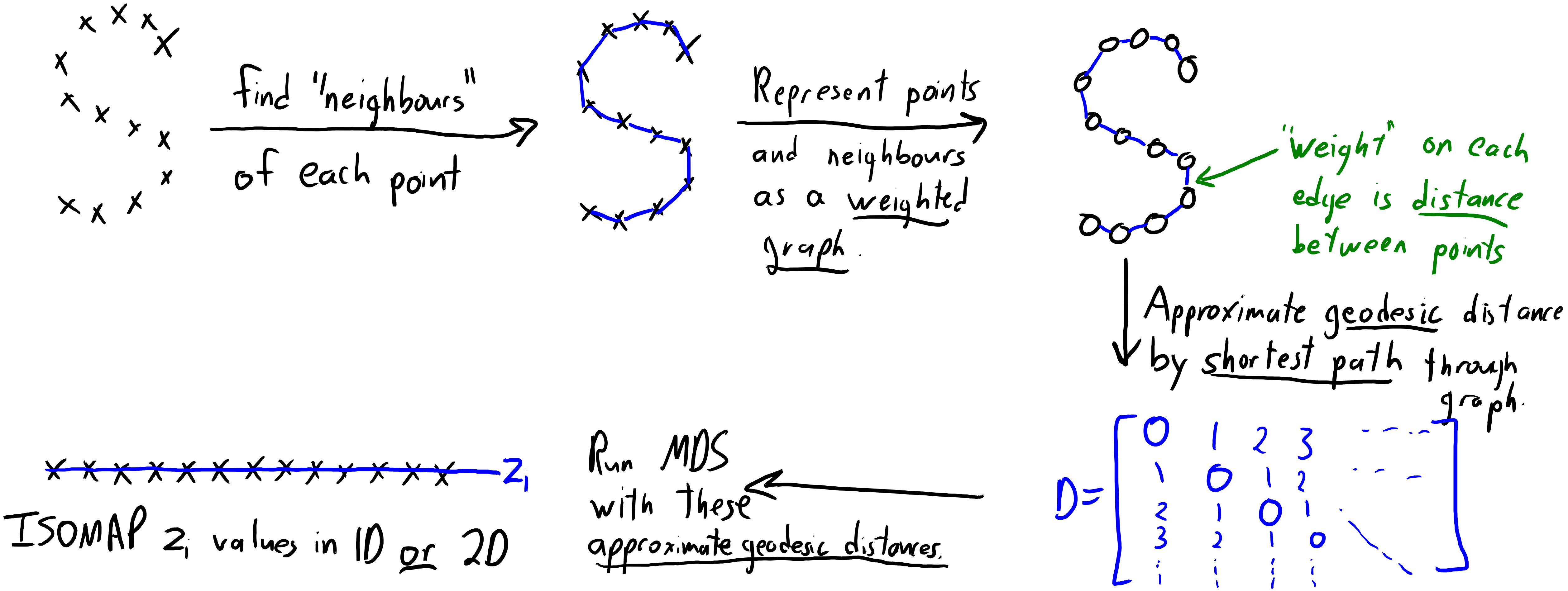


- Images are on a manifold in the high-dimensional space.
  - Euclidean distance doesn't reflect manifold structure.
  - Geodesic distance is distance through space of rotations/resizings.

https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

ISOMAP

- ISOMAP is latent-factor model for visualizing data on manifolds:



Digression: Constructing Neighbour Graphs

- Sometimes you can define the graph/distance without features:
  - Facebook friend graph.
  - Connect YouTube videos if one video tends to follow another.

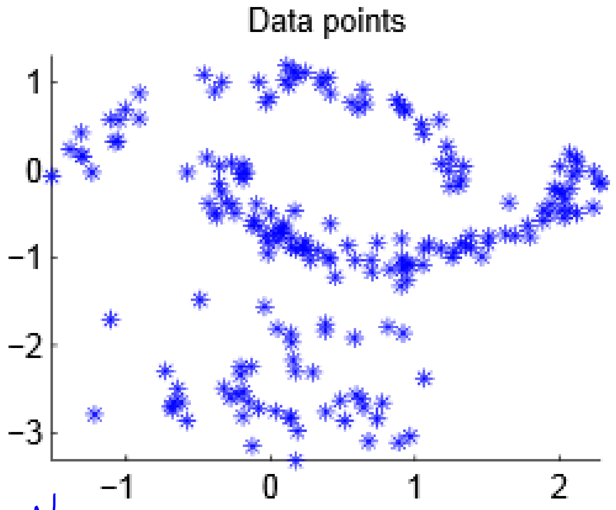
- But we can also convert from features x_i to a "neighbour" graph:
  - Approach 1 ("epsilon graph"): connect x_i to all x_j within some threshold ε.
    - Like we did with density-based clustering.

  - Approach 2 ("KNN graph"): connect x_i to x_j if:
    - x_j is a KNN of x_i **OR** x_i is a KNN of x_j.

  - Approach 2 ("mutual KNN graph"): connect x_i to x_j if:
    - x_j is a KNN of x_i **AND** x_i is a KNN of x_j.

http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf

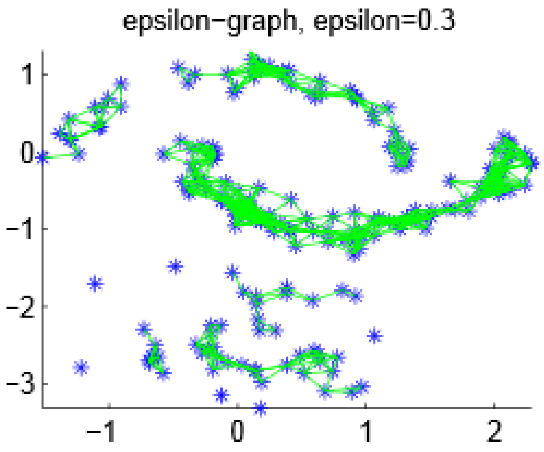
Converting from Features to Graph

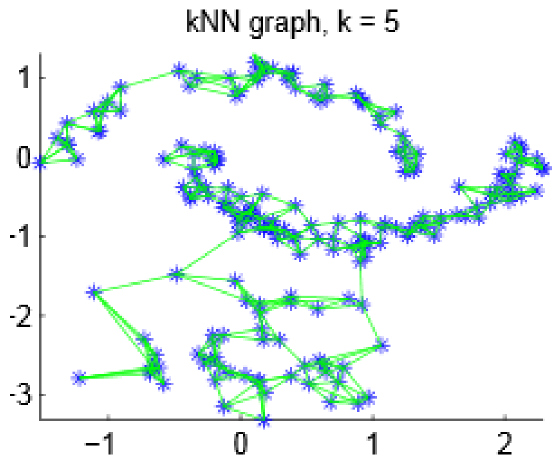


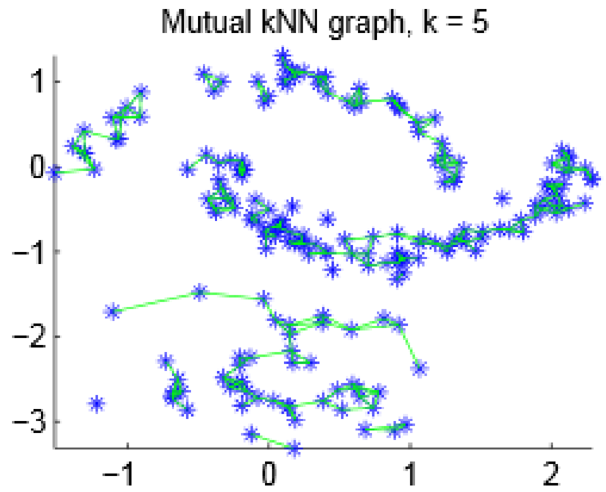
add edge if $\|x_i - x_j\| \le 0.3$

add edge if 'i' is 5-NN of 'j' or 'j' is 5-NN of 'i'

add edge if 'i' and 'j' are KNNs of each other.



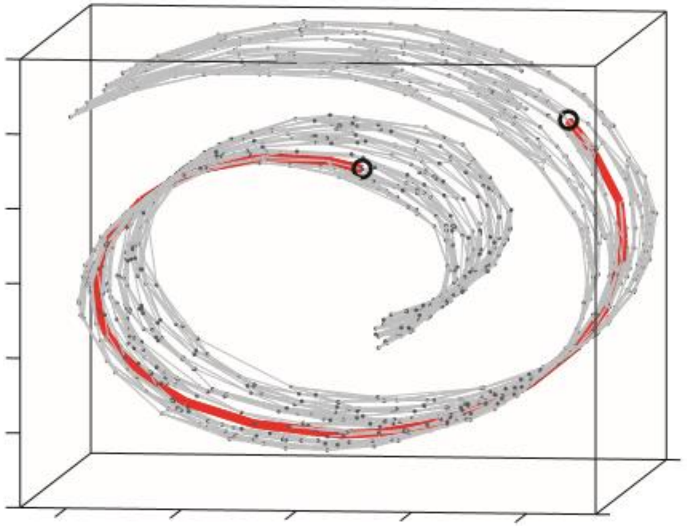




http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/Luxburg07_tutorial_4488%5B0%5D.pdf

ISOMAP

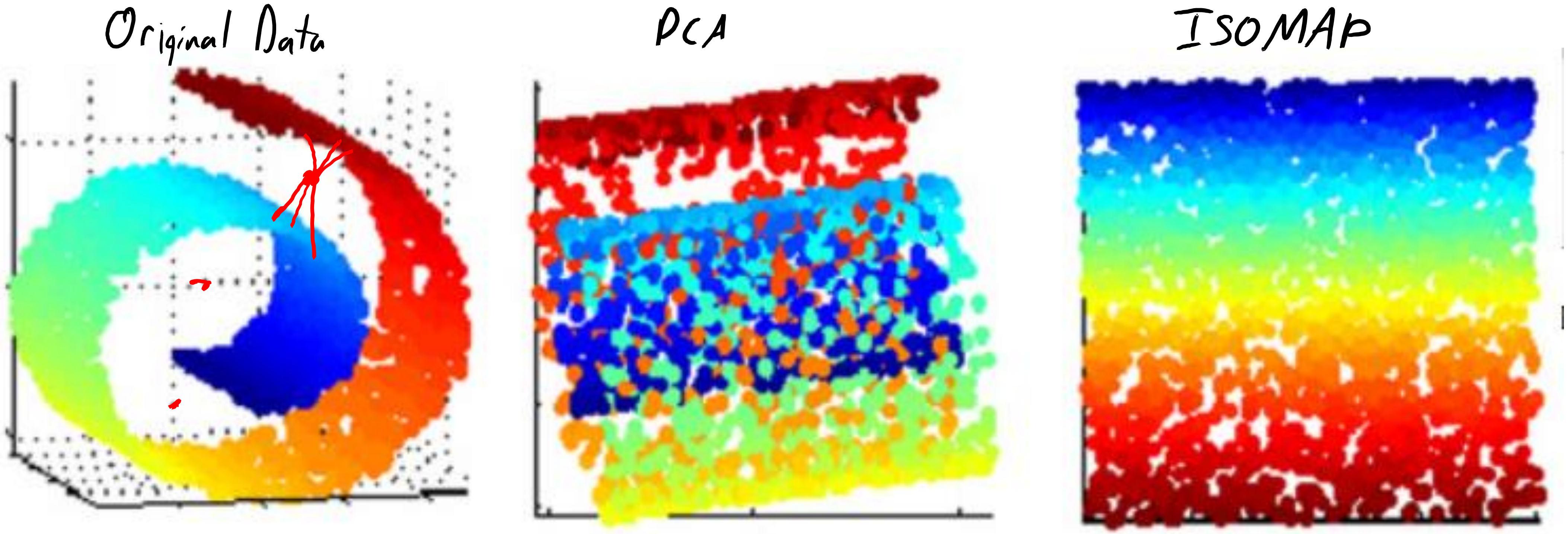
- ISOMAP is latent-factor model for visualizing data on manifolds:
  1. Find the neighbours of each point.
     - Usually "k-nearest neighbours graph", or "epsilon graph".
  2. Compute edge weights:
     - Usually distance between neighbours.
  3. Compute weighted shortest path between all points.
     - Dijkstra or other shortest path algorithm.
  4. Run MDS using these distances.



http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

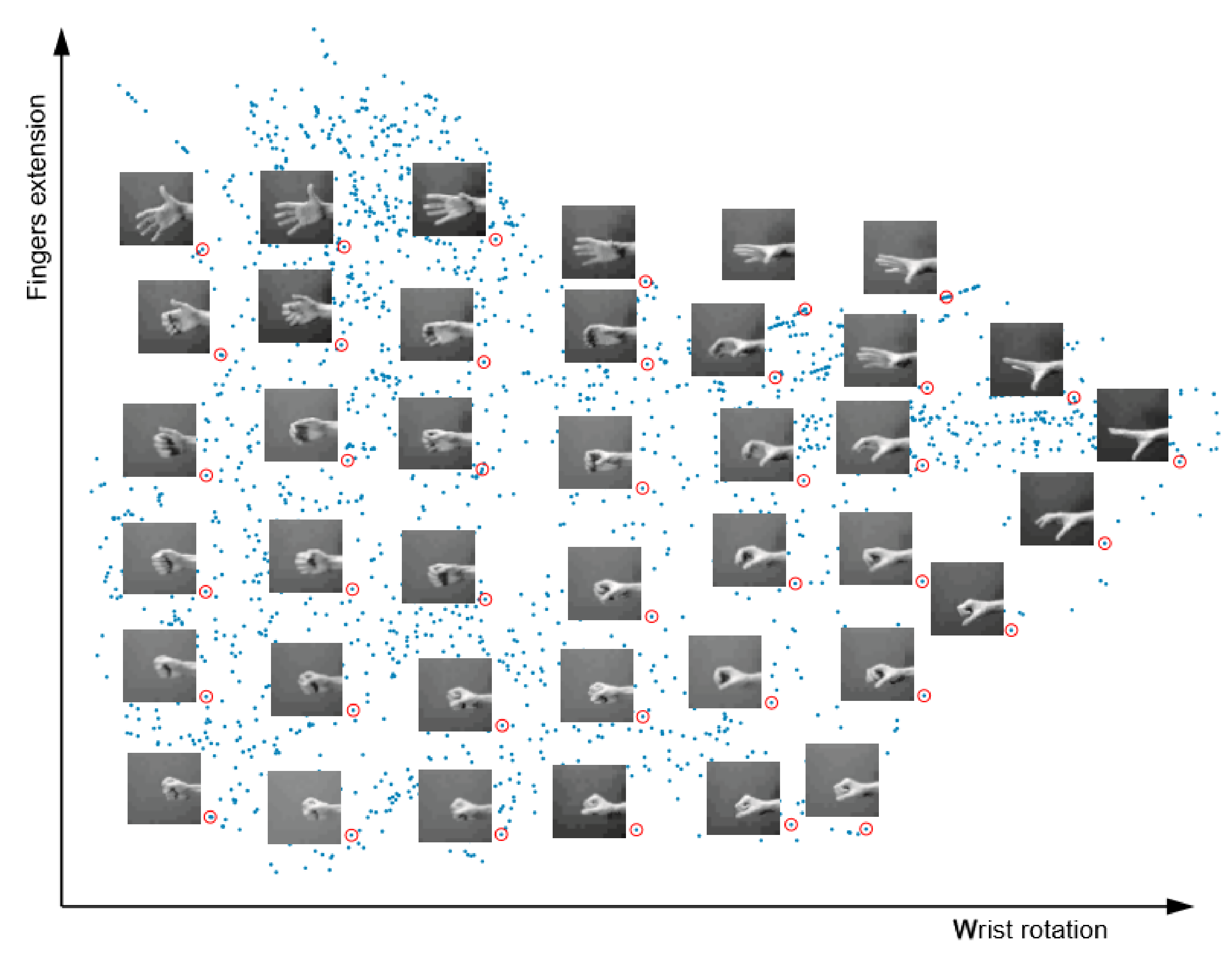
ISOMAP

- ISOMAP can "unwrap" the roll:
  - Shortest paths are approximations to geodesic distances.



- Sensitive to having the right graph:
  - Points off of manifold and gaps in manifold cause problems.

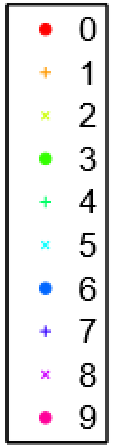
ISOMAP on Hand Images

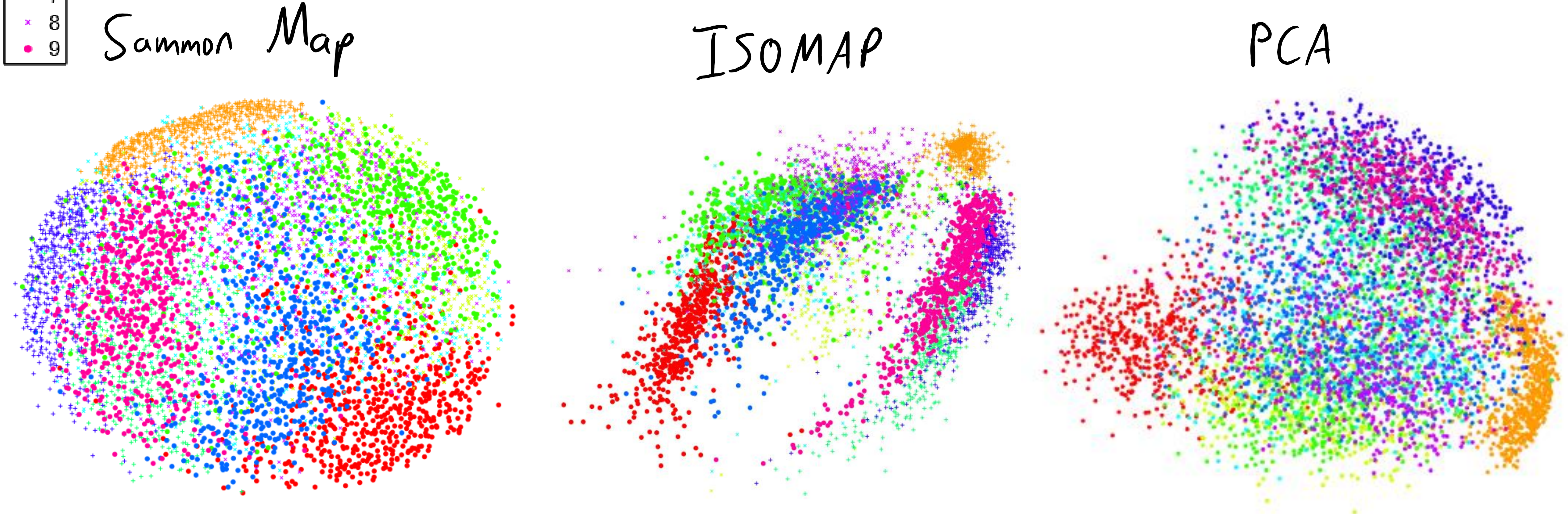


- Related method is “local linear embedding”.

http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

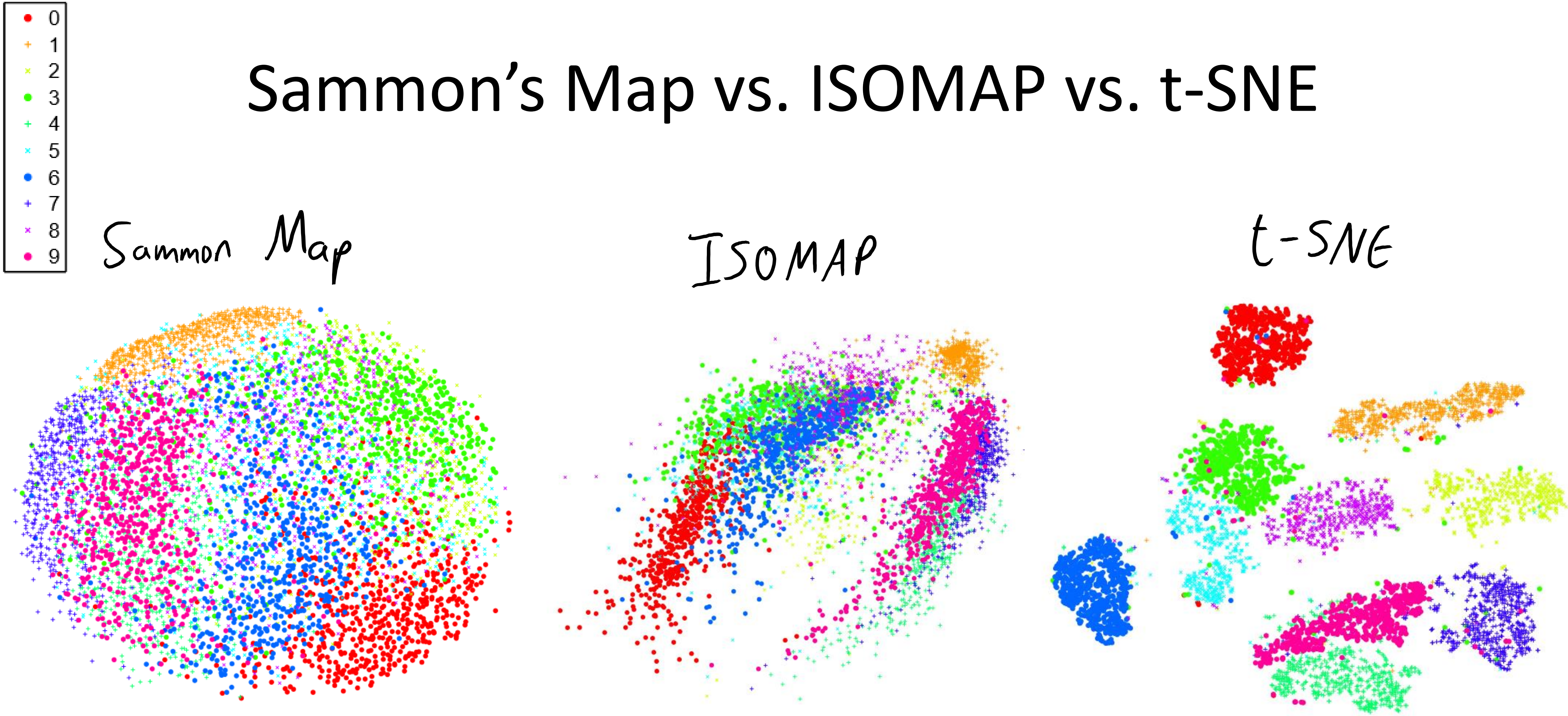
Sammon's Map vs. ISOMAP vs. PCA



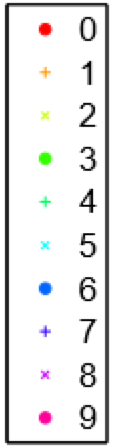


http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

Sammon's Map vs. ISOMAP vs. t-SNE

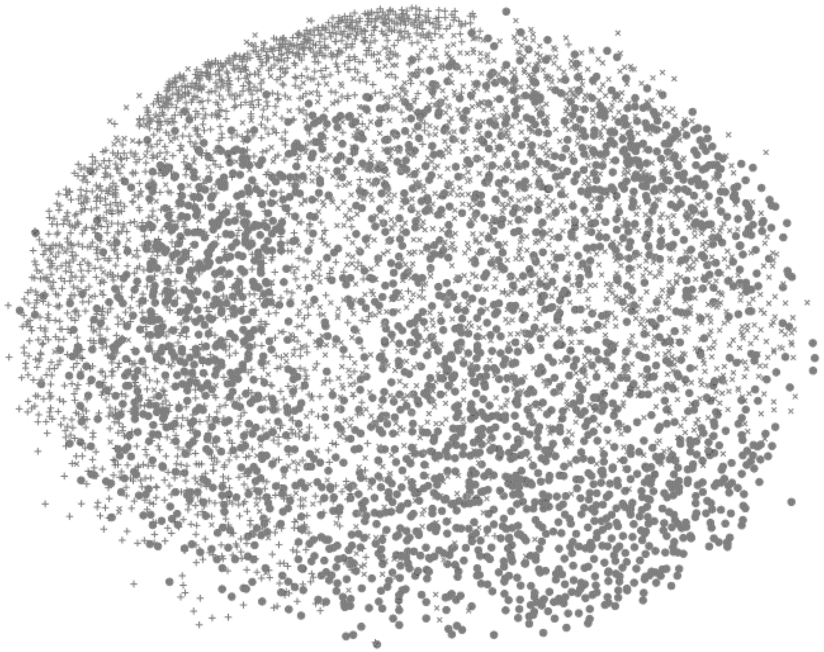


http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf



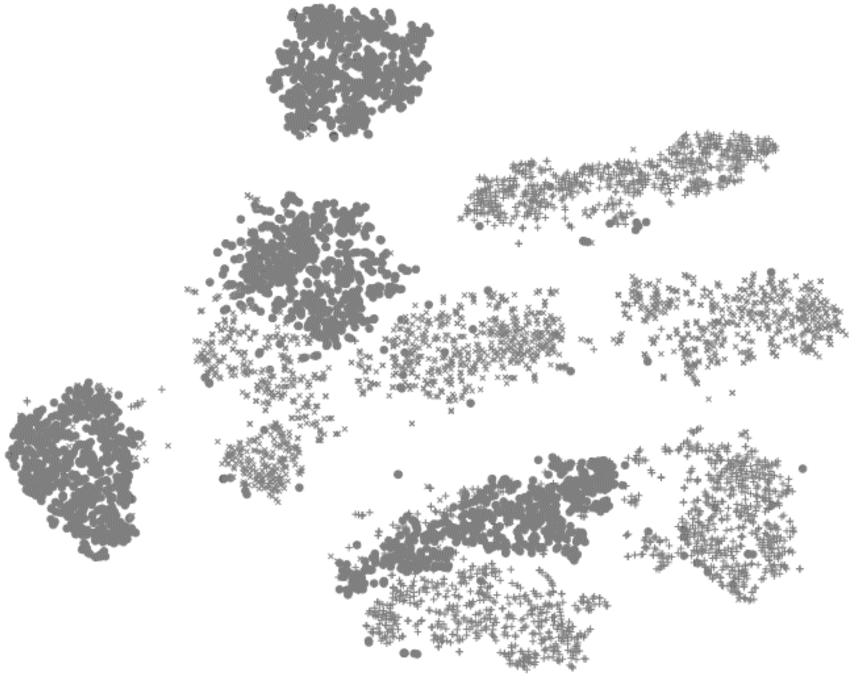
Sammon's Map vs. ISOMAP vs. t-SNE

Sammon Map



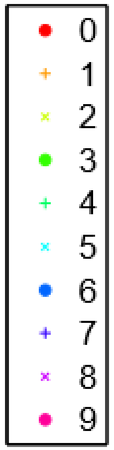
ISOMAP

t-SNE

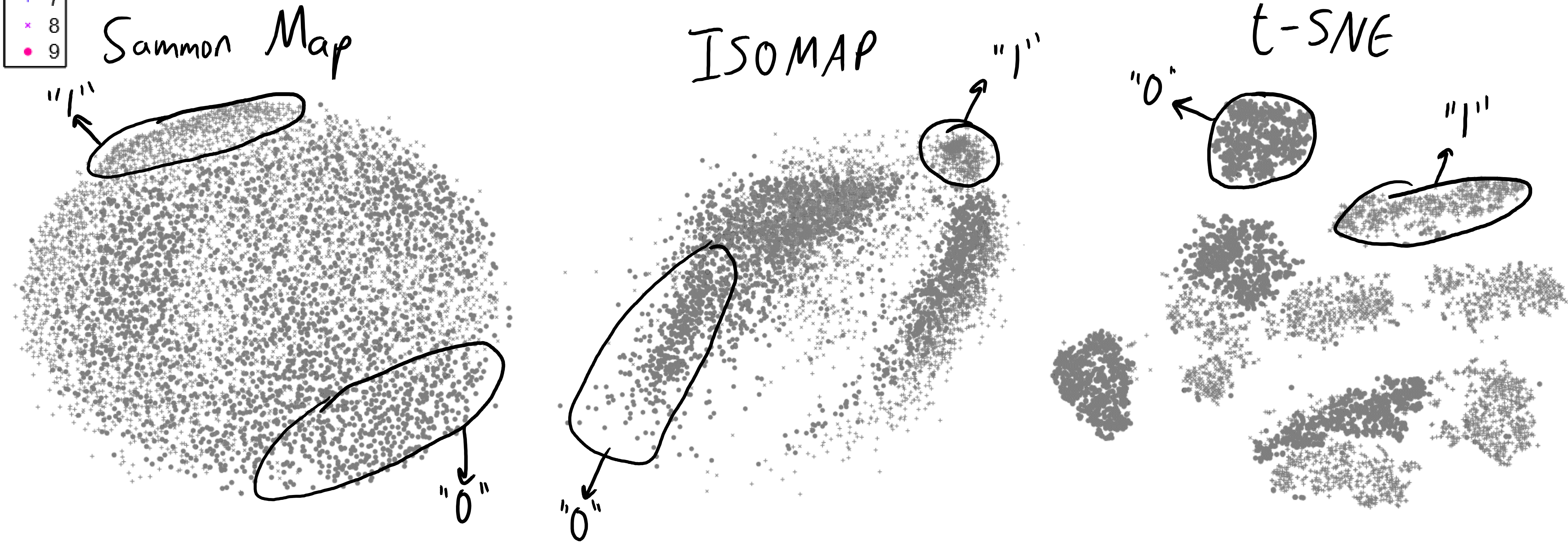


Remember this is unsupervised, algorithms do not know the labels.

http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

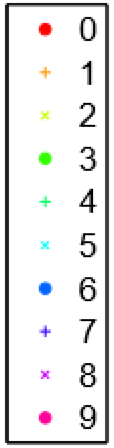


Sammon's Map vs. ISOMAP vs. t-SNE

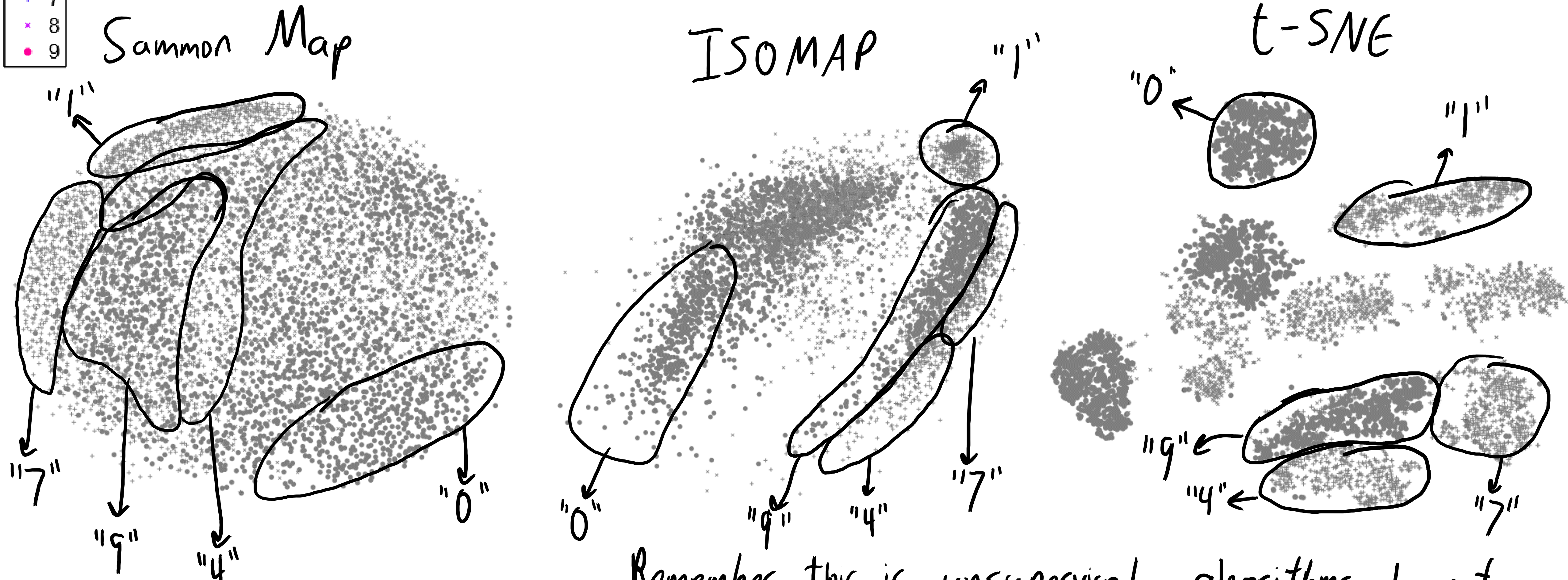


Remember this is <u>unsupervised</u>, algorithms do <u>not</u> know the labels.

http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

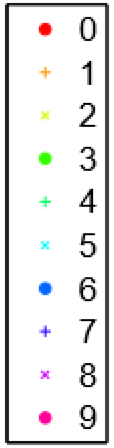


Sammon's Map vs. ISOMAP vs. t-SNE

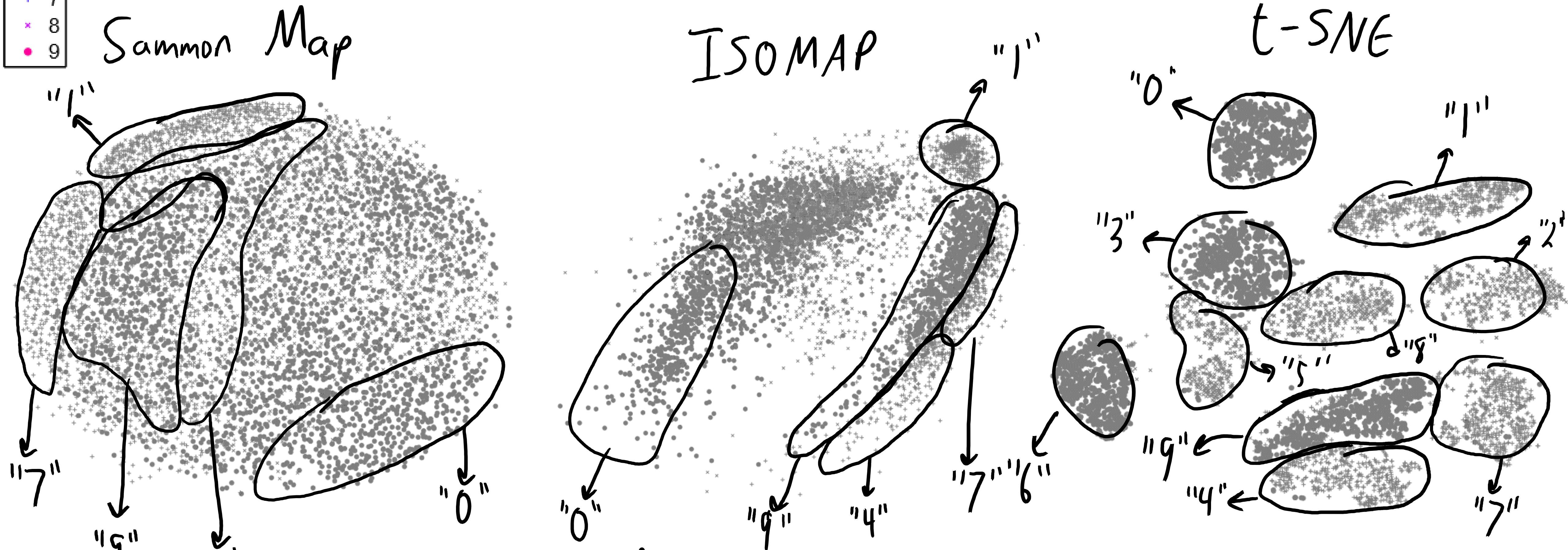


Remember this is <u>unsupervised</u>, algorithms do <u>not</u> know the labels.

http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf



Sammon's Map vs. ISOMAP vs. t-SNE



Remember this is unsupervised, algorithms do not know the labels.

http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

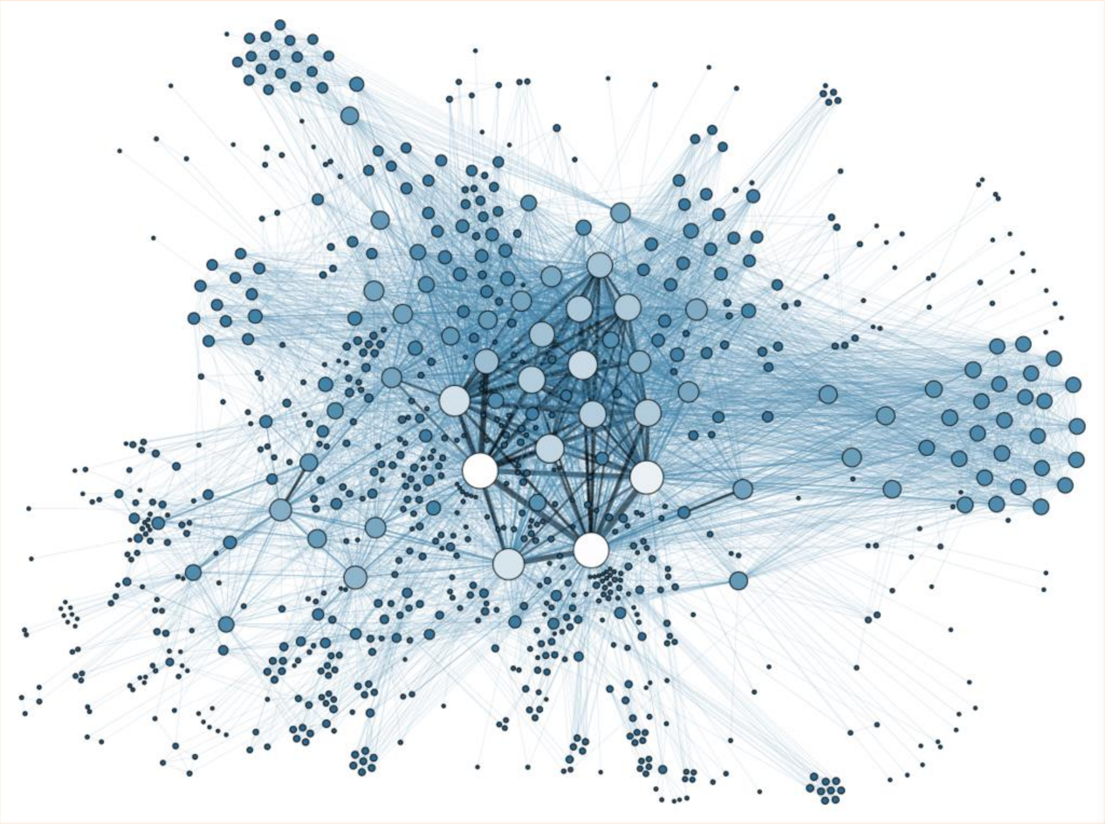
Summary

- Multi-dimensional scaling is a non-parametric latent-factor model.
- Different MDS distances/losses/weights usually gives better results.
- Manifold learning focuses on low-dimensional curved structures.
- ISOMAP is most common approach:
  - Approximates geodesic distance by shortest path in weighted graph.
- t-SNE is promising new data MDS method.

- Next time: deep learning.

Graph Drawing

- A closely-related topic to MDS is graph drawing:
  - Given a graph, how should we display it?
  - Lots of interesting methods: https://en.wikipedia.org/wiki/Graph_drawing



Bonus Slide: Multivariate Chain Rule

- Recall the univariate chain rule:

$$\frac{d}{dw}\left[f(g(w))\right] = f'(g(w))\,g'(w)$$

- The multivariate chain rule:

$$\underbrace{\nabla\left[f(g(w))\right]}_{d \times 1} = \underbrace{f'(g(w))}_{1 \times 1}\,\underbrace{\nabla g(w)}_{d \times 1}$$

- Example:

$$\nabla\left[\tfrac{1}{2}(w^T x_i - y_i)^2\right]$$

$$= \nabla\left[f(g(w))\right]$$

with $g(w) = w^T x_i - y_i$ $\longrightarrow$ $\nabla g(w) = x_i$

and $f(r_i) = \frac{1}{2} r_i^2$ $\longrightarrow$ $f'(r_i) = r_i$

$$\nabla\left[f(g(w))\right] = r_i \, x_i$$

$$= (w^T x_i - y_i)x_i$$

Bonus Slide: Multivariate Chain Rule for MDS

- General MDS formulation:

$$\underset{Z \in \mathbb{R}^{n \times k}}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} g(d_1(x_i, x_j), d_2(z_i, z_j))$$

- Using multivariate chain rule we have:

$$\nabla_{z_i} g(d_1(x_i, x_j), d_2(z_i, z_j)) = g'(d_1(x_i, x_j), d_2(z_i, z_j)) \nabla_{z_i} d_2(z_i, z_j)$$

- Example: If $d_1(x_i, x_j) = \|x_i - x_j\|$ and $d_2(z_i, z_j) = \|z_i - z_j\|$ and $g(d_1, d_2) = \frac{1}{2}(d_1 - d_2)^2$

$$\nabla_{z_i} g(d_1(x_i, x_j), d_2(z_i, z_j)) = \underbrace{-(d_1(x_i, x_j) - d_2(z_i, z_j))}_{\substack{g'(d_1, d_2) \\ \text{(move distances closer)}}} \underbrace{\left[- \frac{(z_i - z_j)}{2\|z_i - z_j\|} \right]}_{\substack{\nabla_{z_i} d_2(z_i, z_j) \\ \text{(how distance changes in } z\text{-space)}}}$$

$\hookrightarrow$ Assuming $z_i \neq z_j$