CPSC 340:
Machine Learning and Data Mining

Multi-Dimensional Scaling

Fall 2019

Last Time: 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:

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

"Try to make scatterplot distances match high-dimensional distance"

sum over pairs of examples

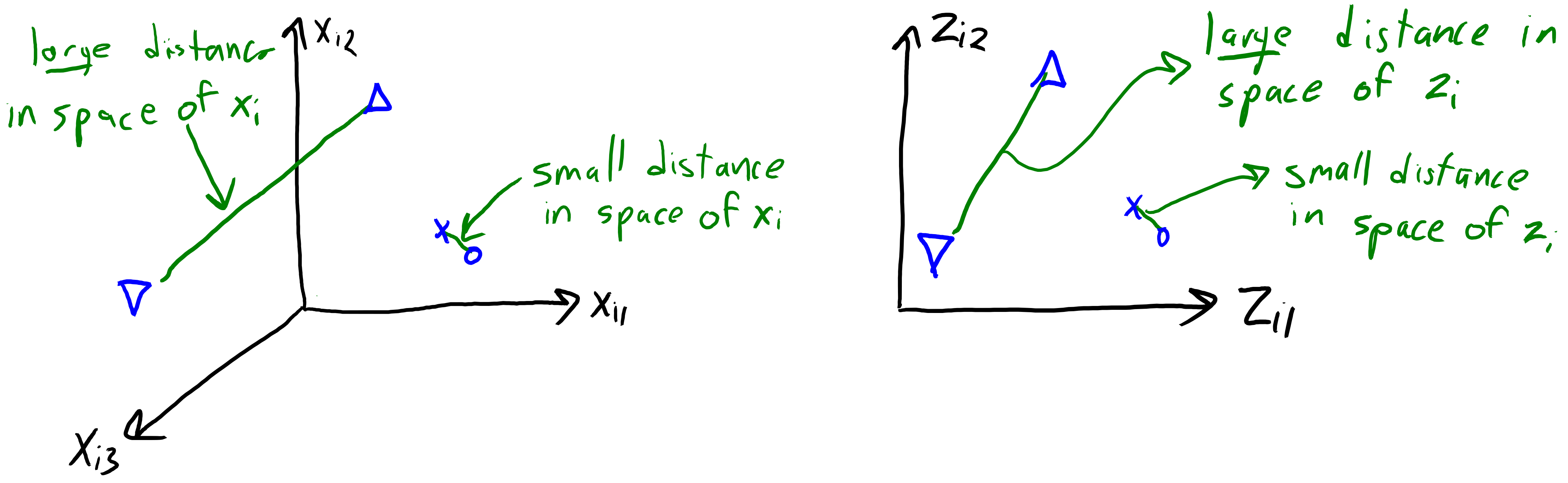
distance in scatterplot

Distance between points in original 'd' dimensions

Multi-Dimensional Scaling

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

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

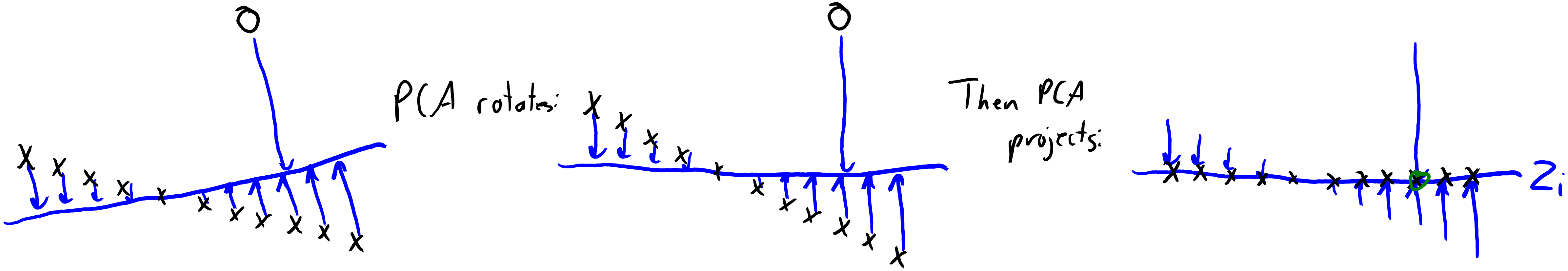


Multi-Dimensional Scaling

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

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

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

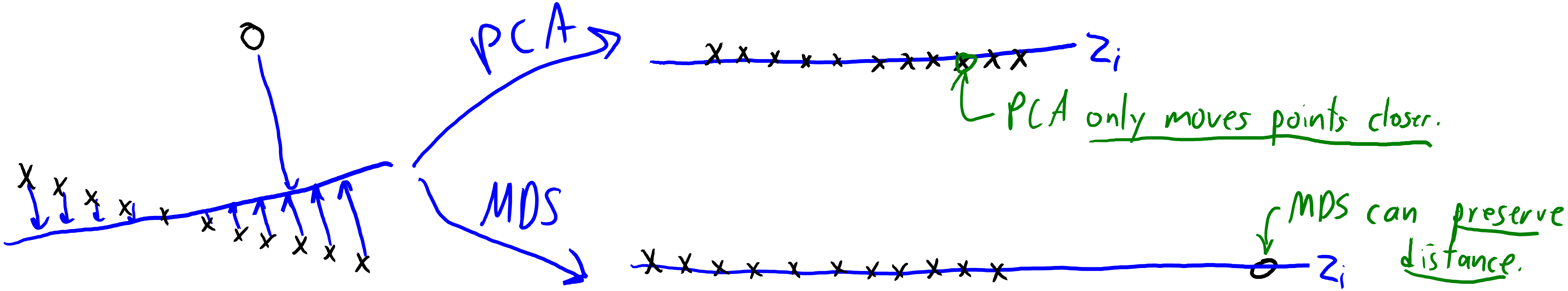


Multi-Dimensional Scaling

- 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.

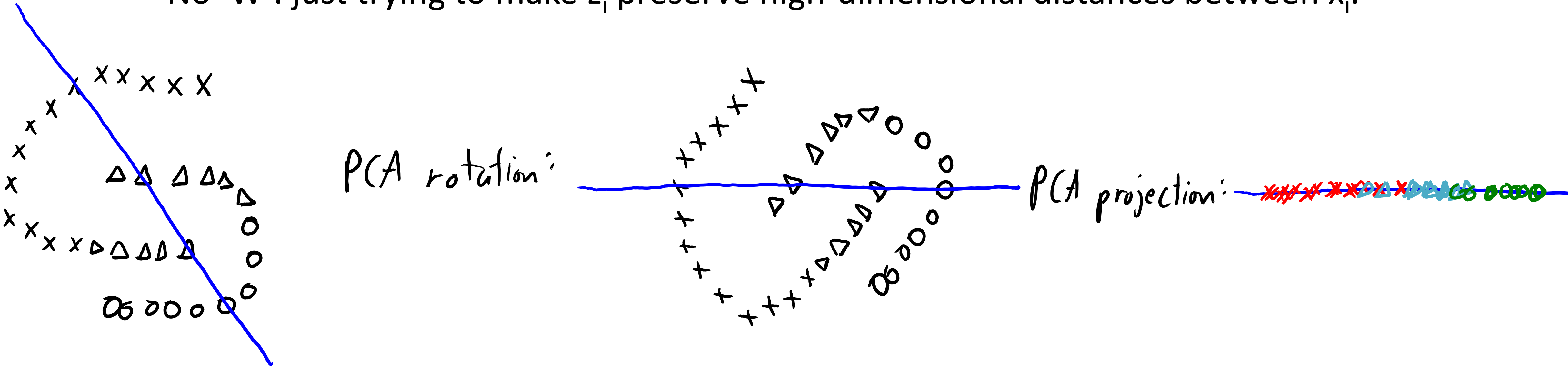


Multi-Dimensional Scaling

- 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.

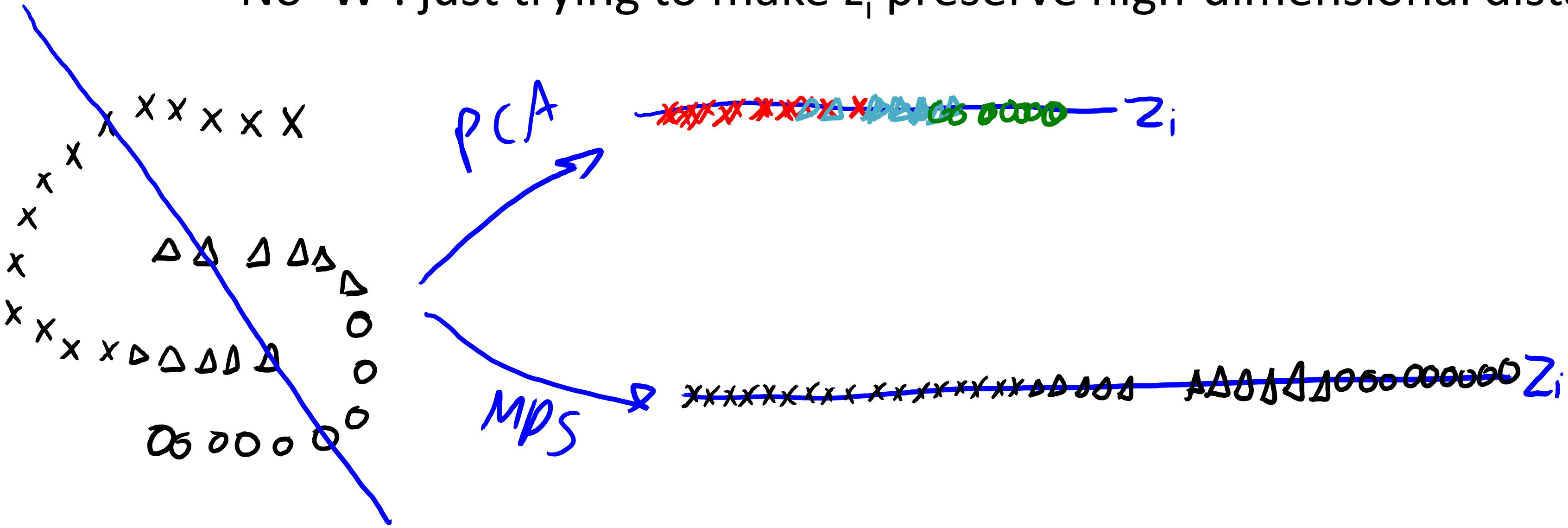


Multi-Dimensional Scaling

- 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:

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

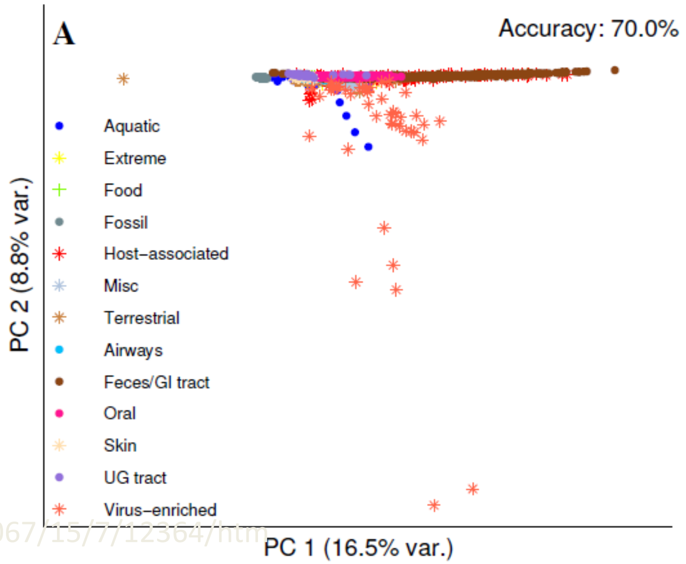
- A possibility is "classic" MDS with $d_1(x_i,x_j) = x_i^T x_j$ and $d_2(z_i,z_j) = z_i^T z_j$.
  - We obtain PCA in this special case (centered x_i, d_3 as the squared L2-norm).
  - Not a great choice because it's a linear model.

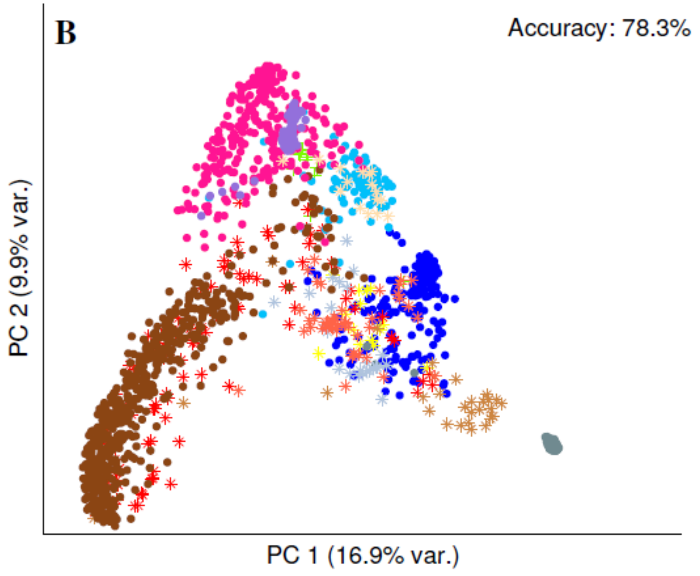
Different MDS Cost Functions

- 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

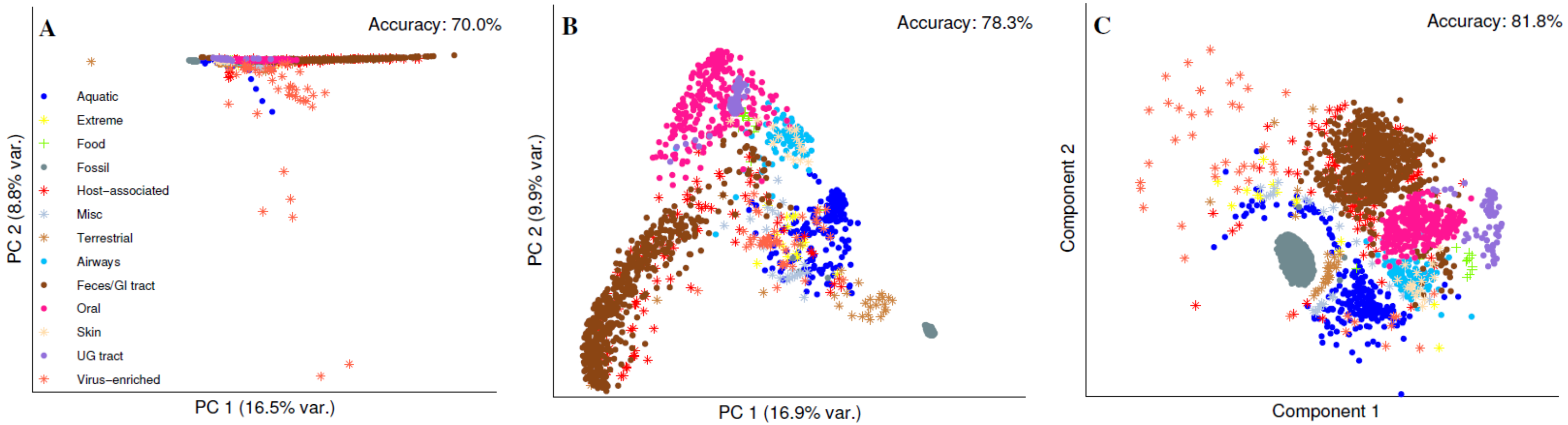
- Challenge for most MDS models: they focus on large distances.
  - Leads to "crowding" effect like with PCA.
- Early attempt to address this is Sammon's mapping:
  - Weighted MDS so large/small distances are more comparable.

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

  - 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.



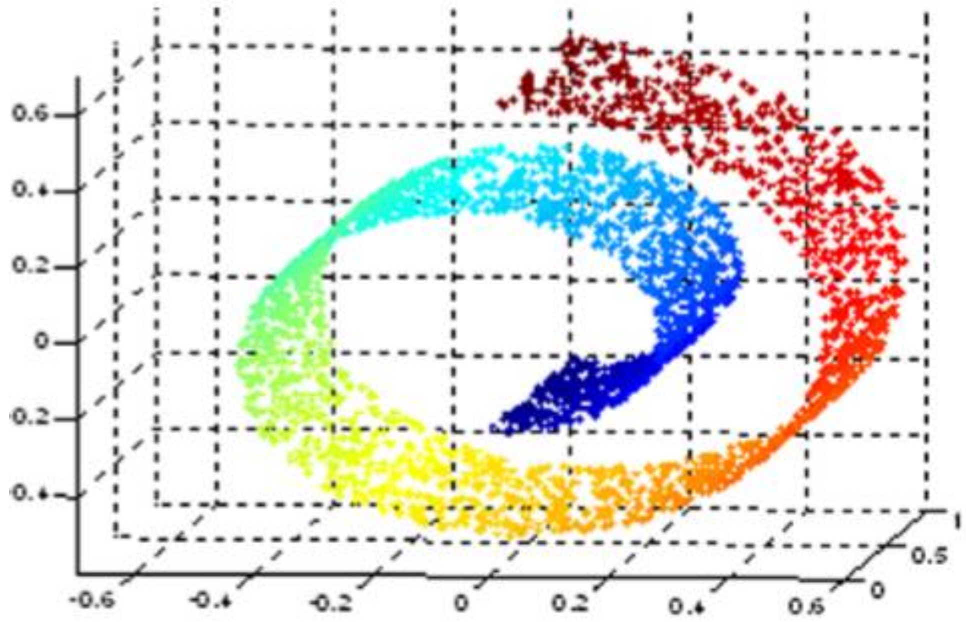
http://www.mdpi.com/1422-0067/15/7/12364/htm

(pause)

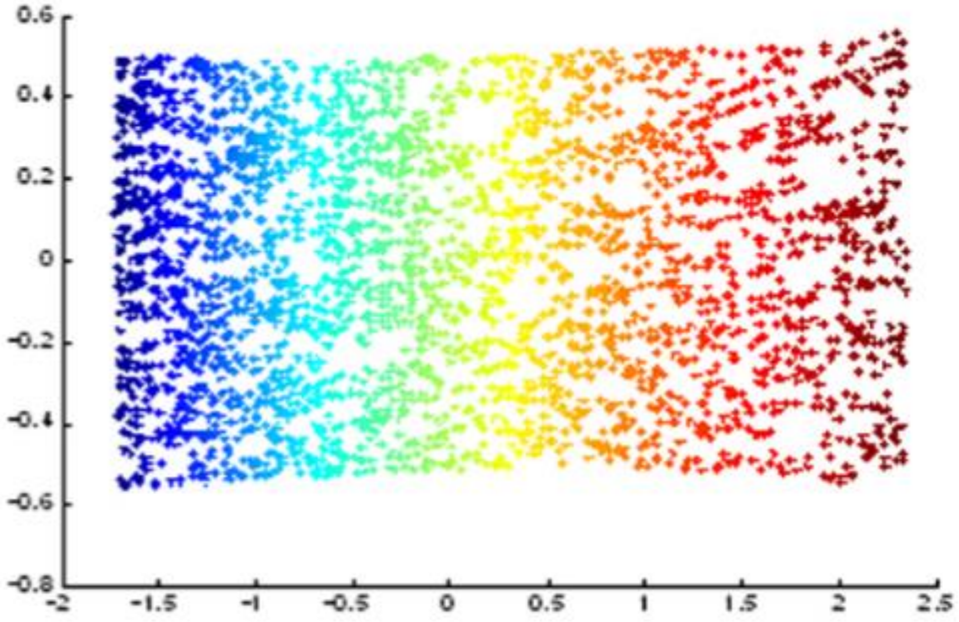
Learning Manifolds

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

Original data



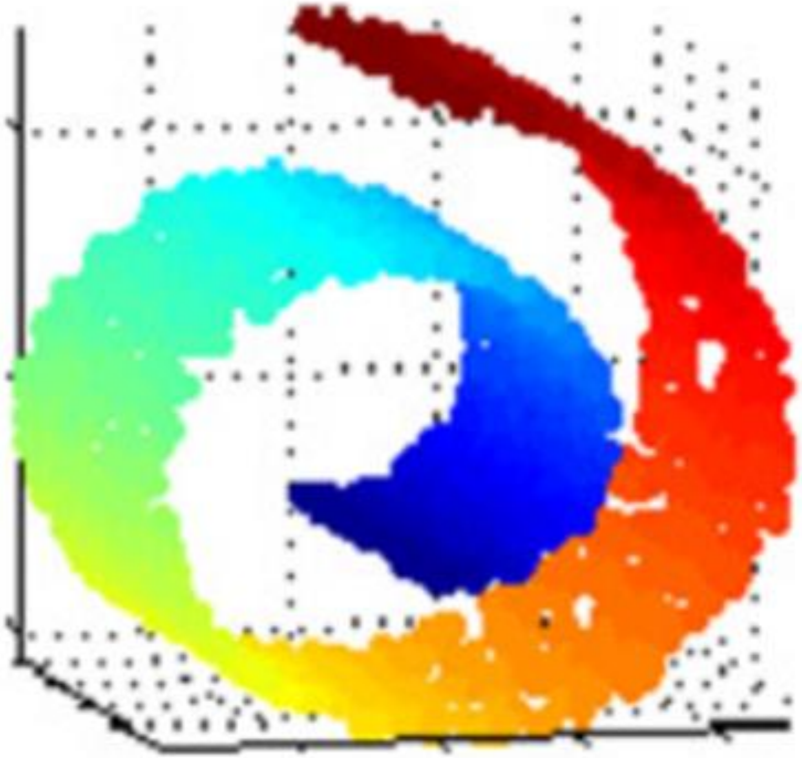
Two-dimensional manifold



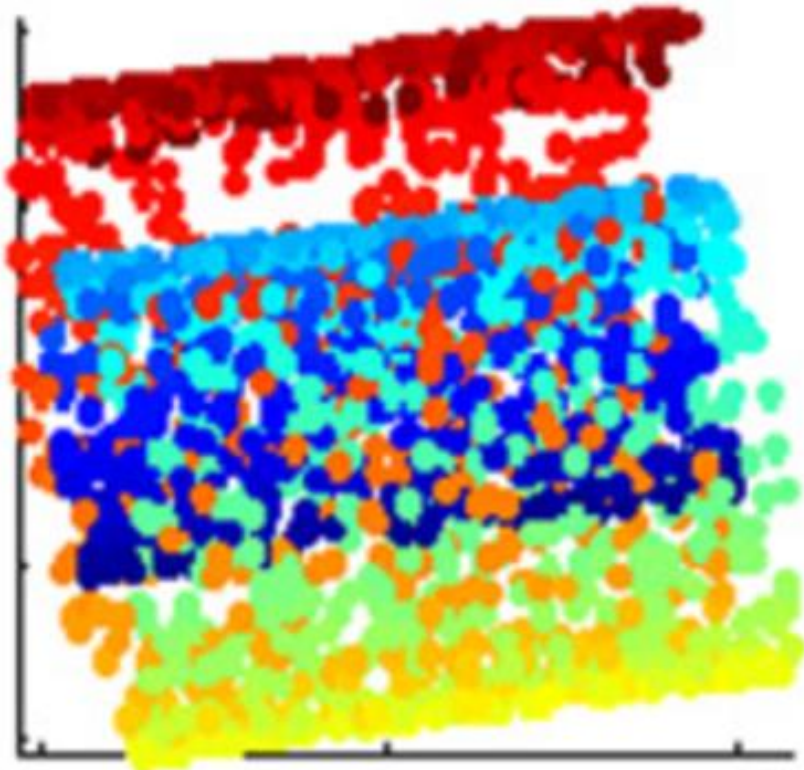
http://www.biomedcentral.com/content/pdf/1471-2105-13-S7-S3.pdf

Learning Manifolds

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



Original data

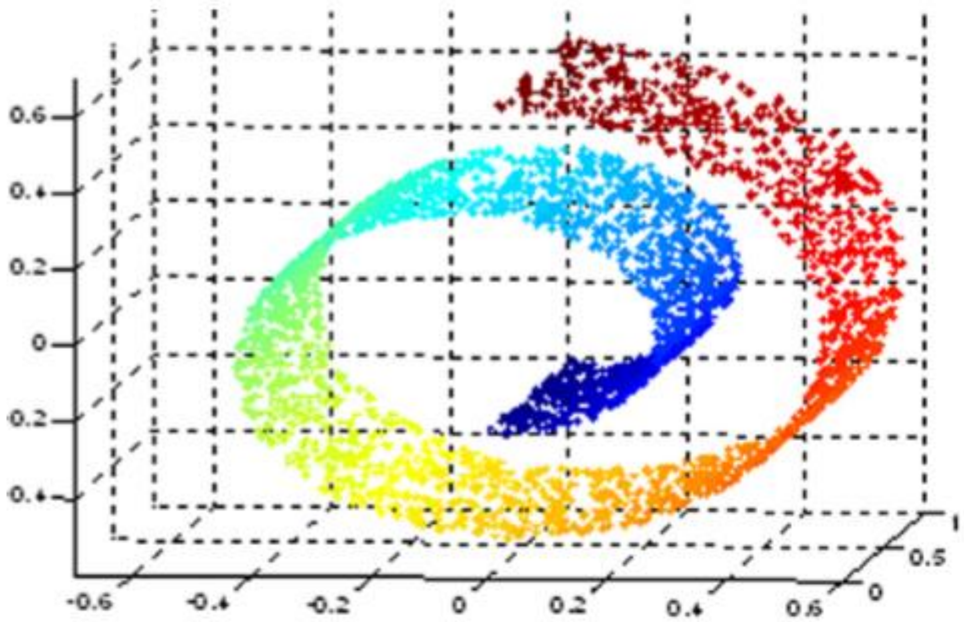


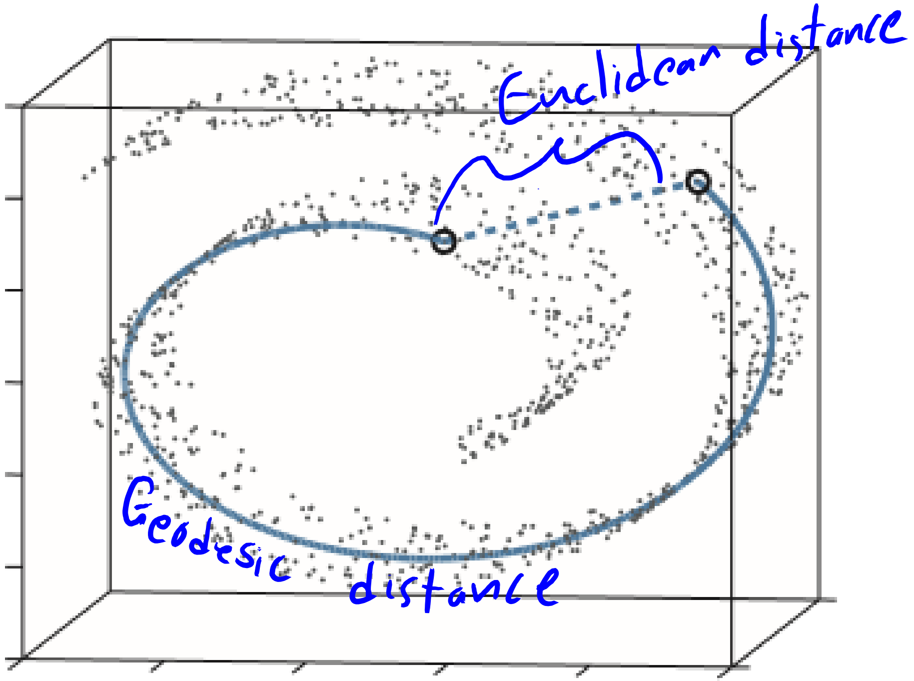
PCA

http://www.peh-med.com/content/9/1/12/figure/F1

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.

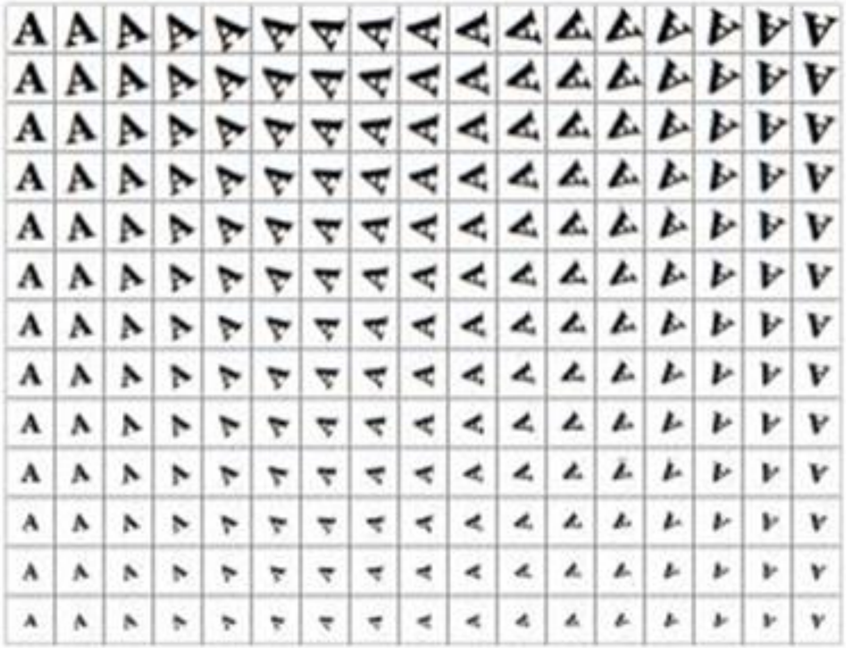




http://www.biomedcentral.com/content/pdf/1471-2105-13-S7-S3.pdf
http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

Manifolds in Image Space

- Consider slowly-varying transformation of image:

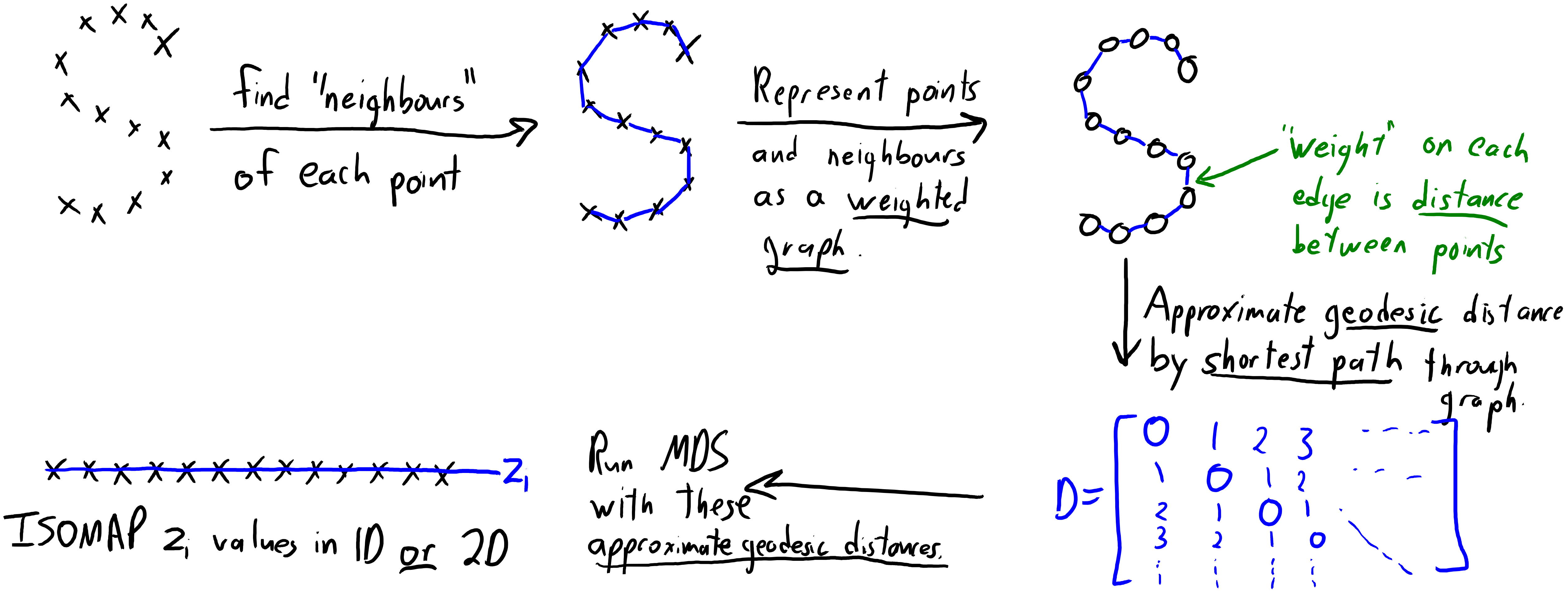


- Images are on a manifold in the high-dimensional space.
  - Euclidean distance doesn't reflect manifold structure.
  - Geodesic distance is distance through space of rotations/resizings.

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

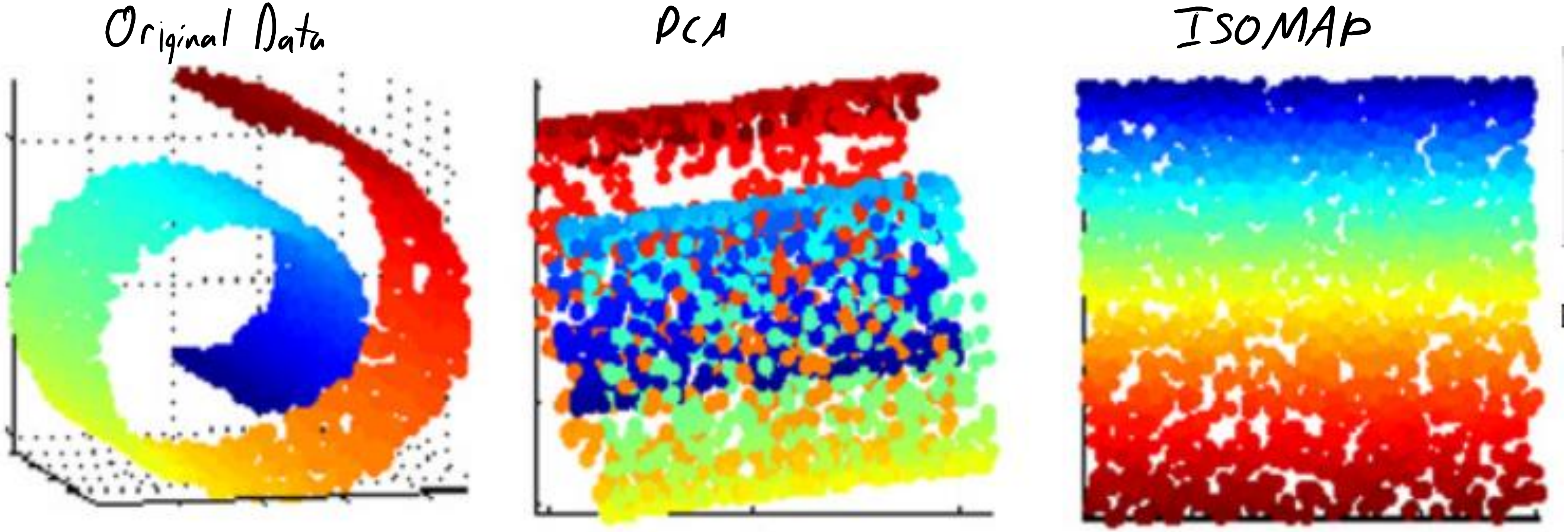
ISOMAP

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



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.

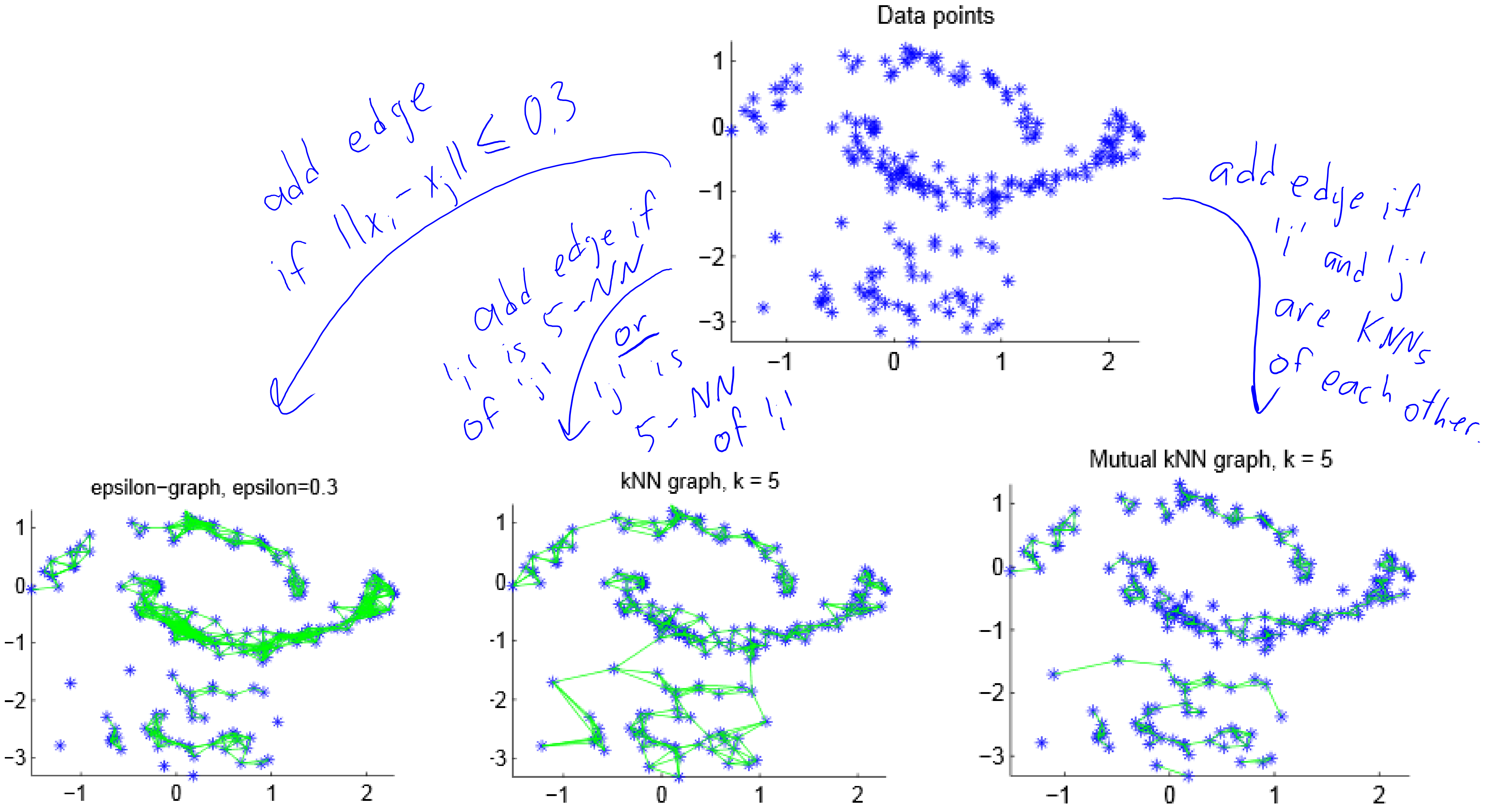
Constructing Neighbour Graphs

- 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

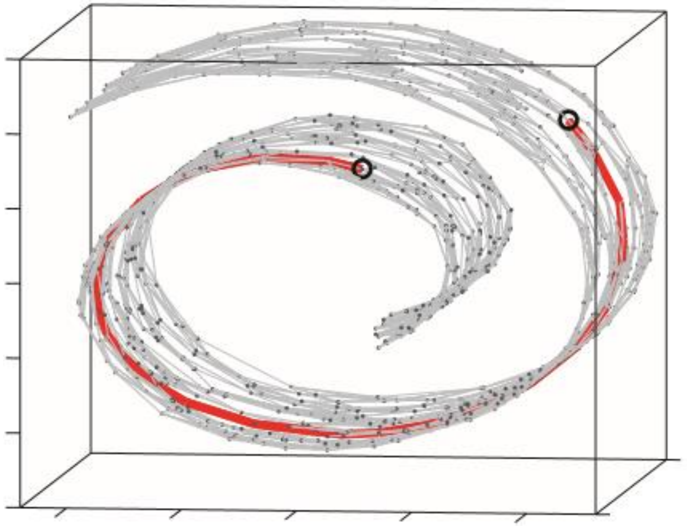
Converting from Features to Graph



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

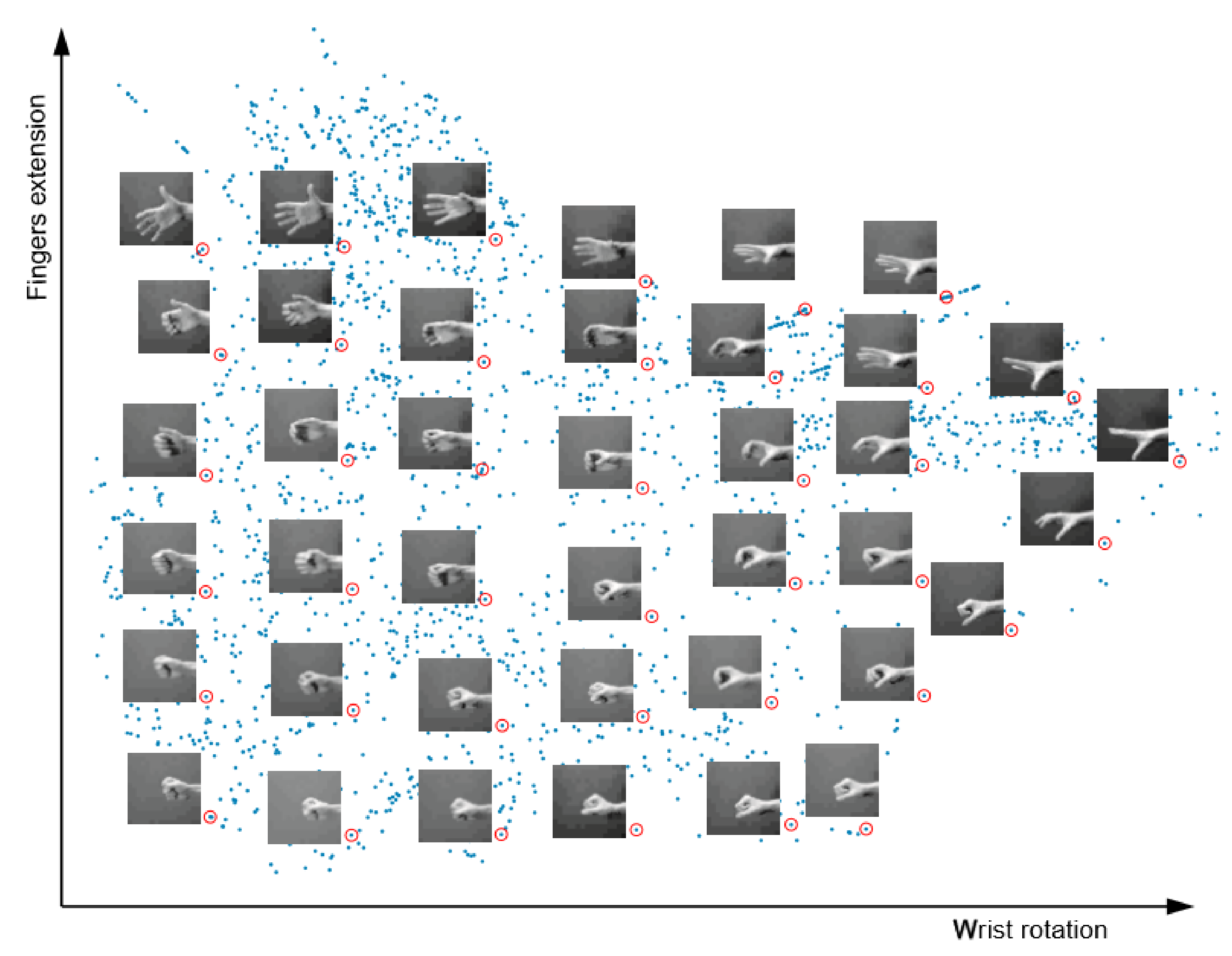
ISOMAP

- 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

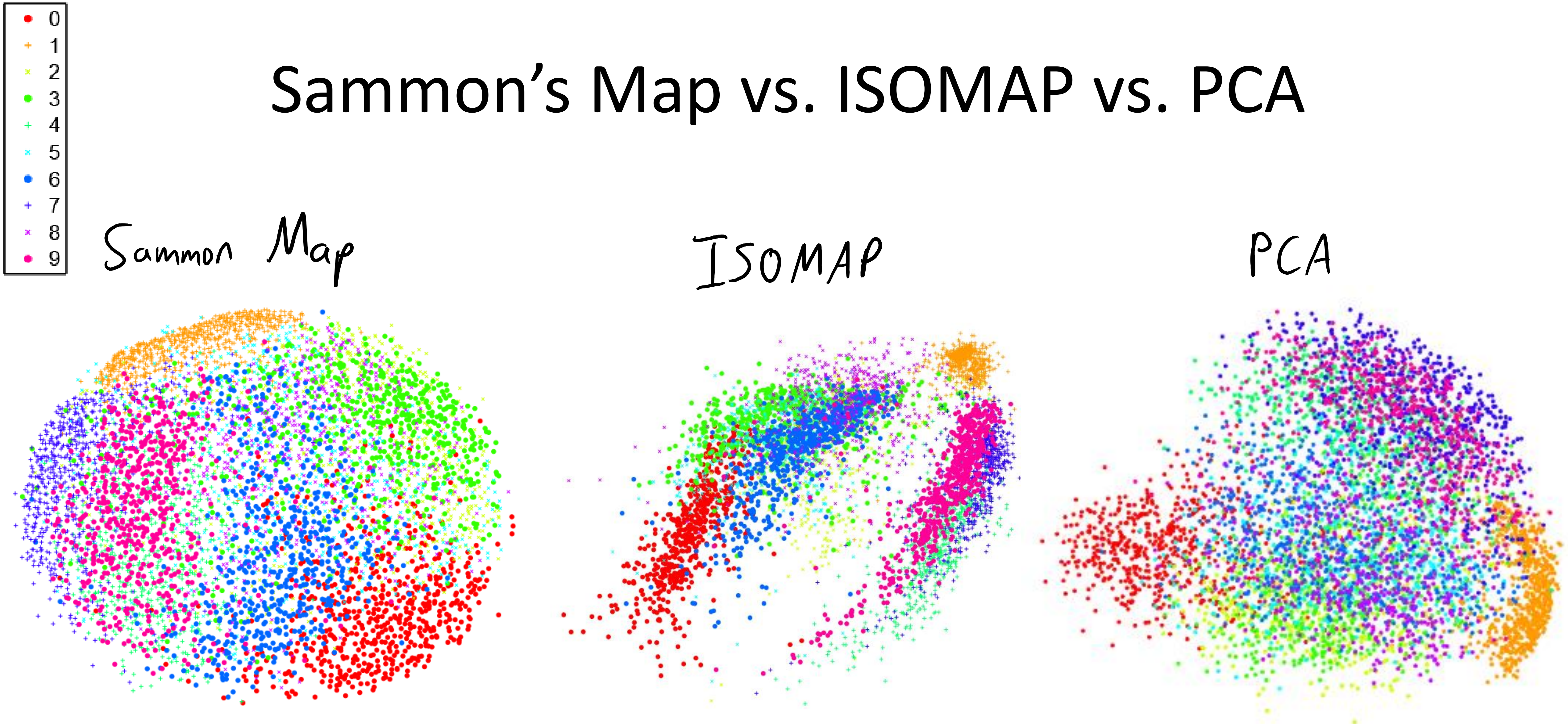
ISOMAP on Hand Images



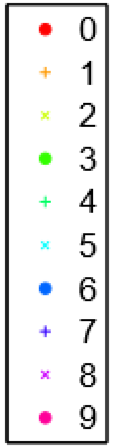
- Related method is “local linear embedding”.

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

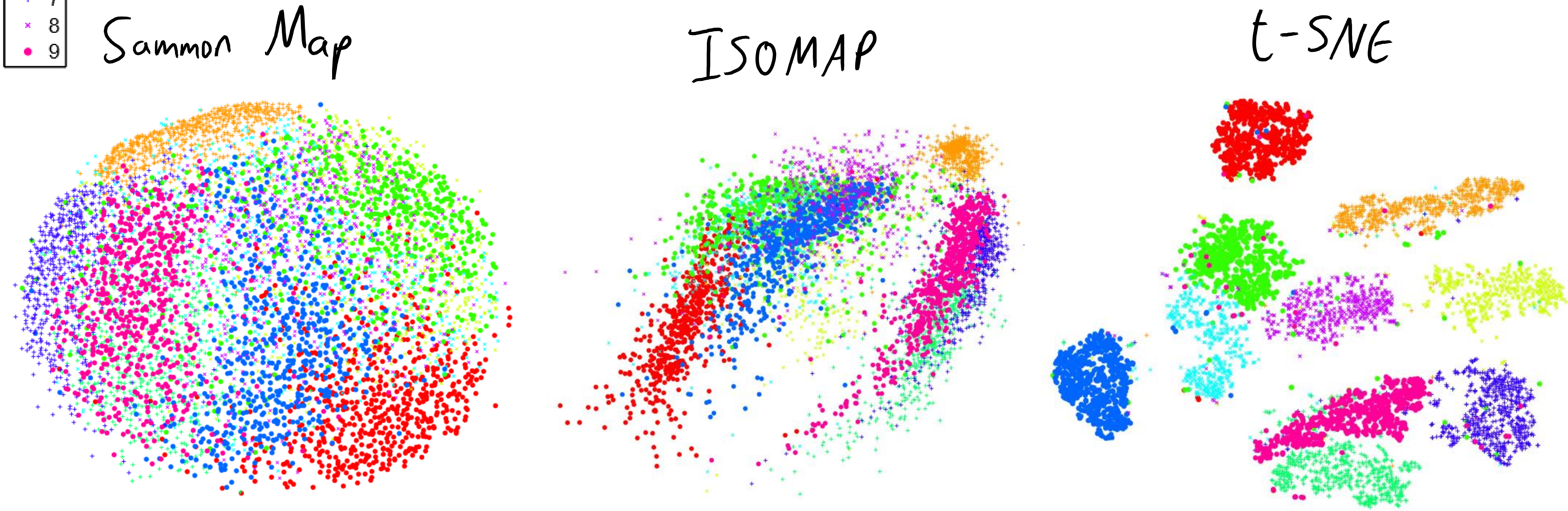
Sammon's Map vs. ISOMAP vs. PCA



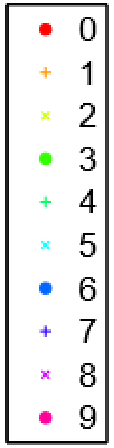
http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf



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

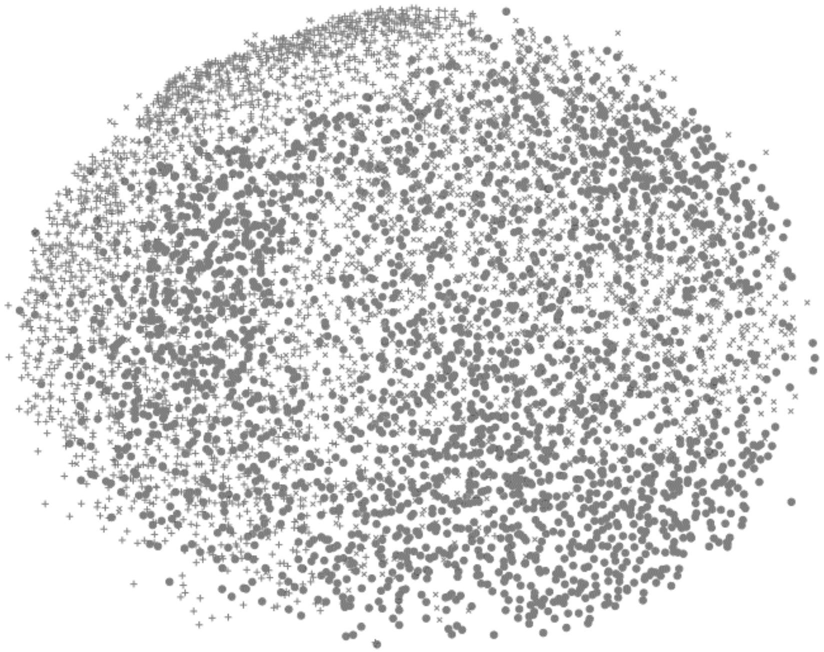


http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf



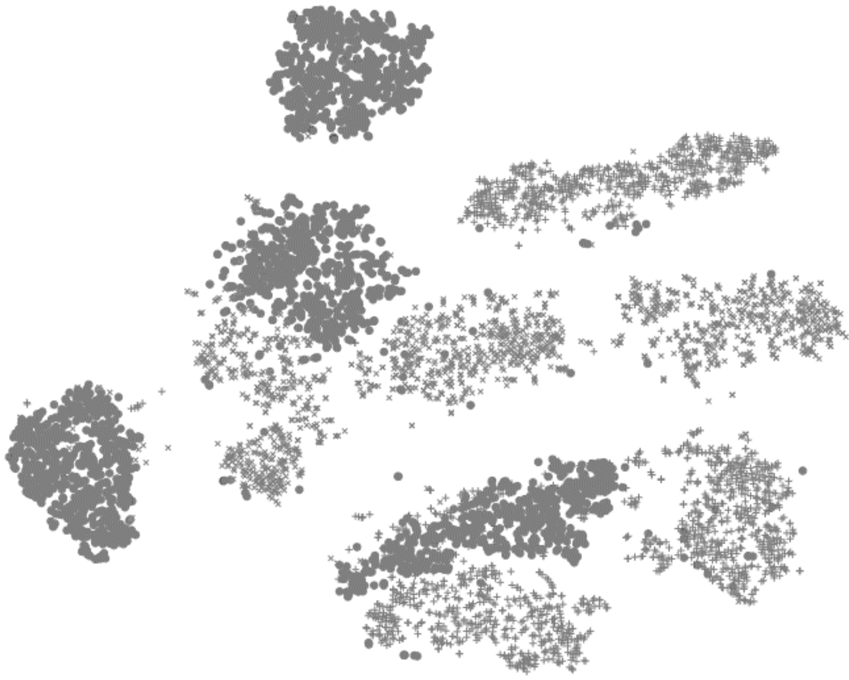
Sammon's Map vs. ISOMAP vs. t-SNE

Sammon Map



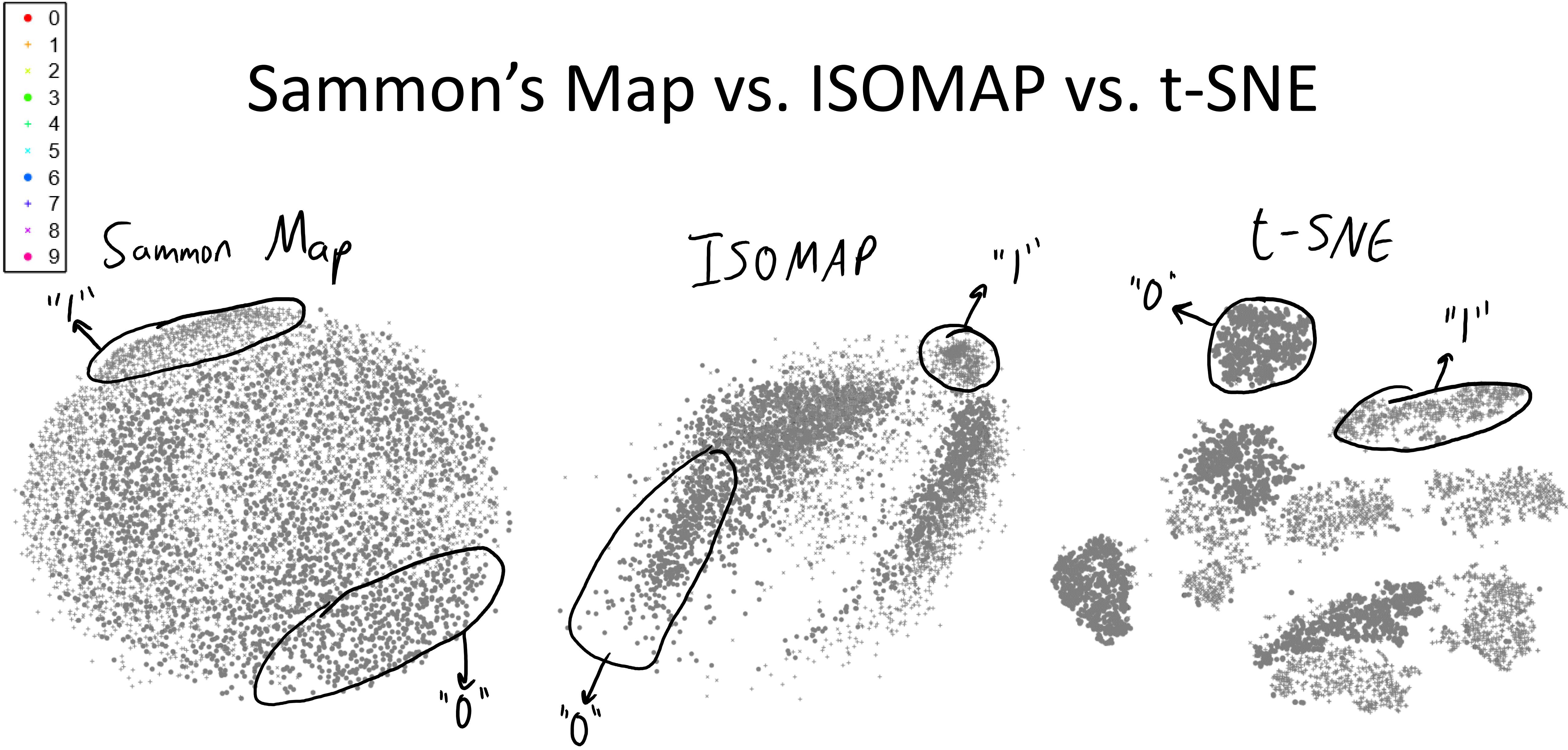
ISOMAP

t-SNE



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

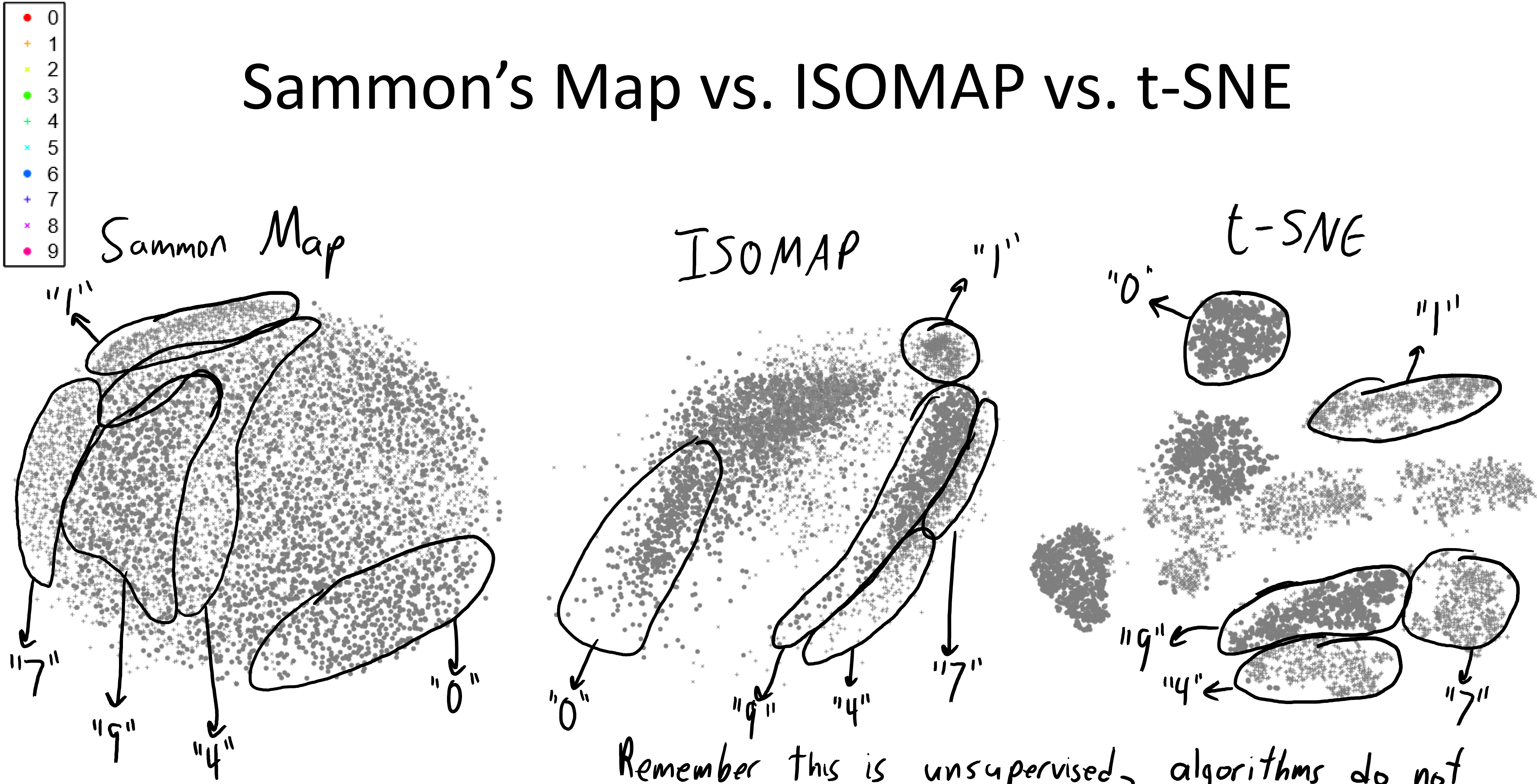
Sammon's Map vs. ISOMAP vs. t-SNE



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

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

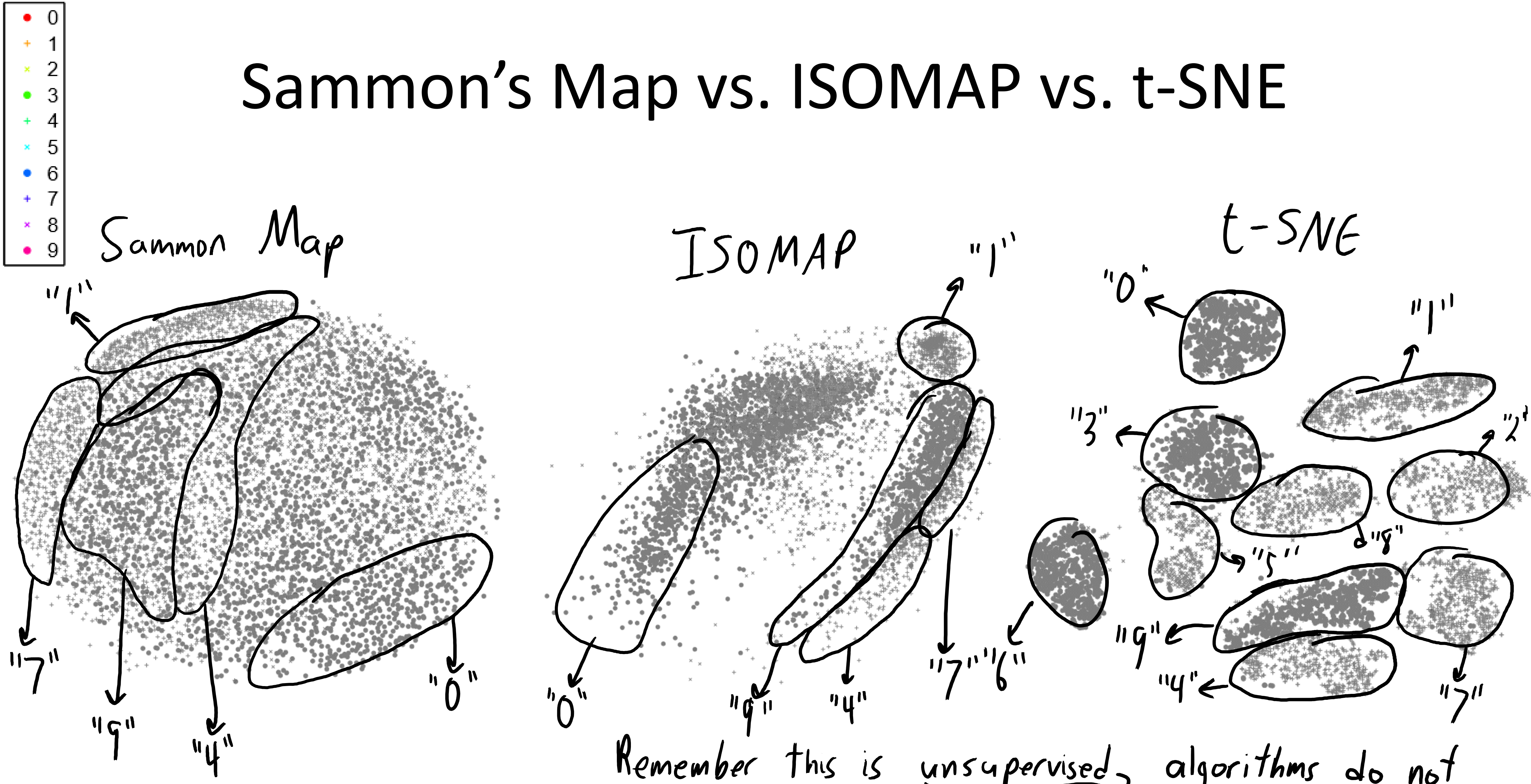
Sammon's Map vs. ISOMAP vs. t-SNE



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

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

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

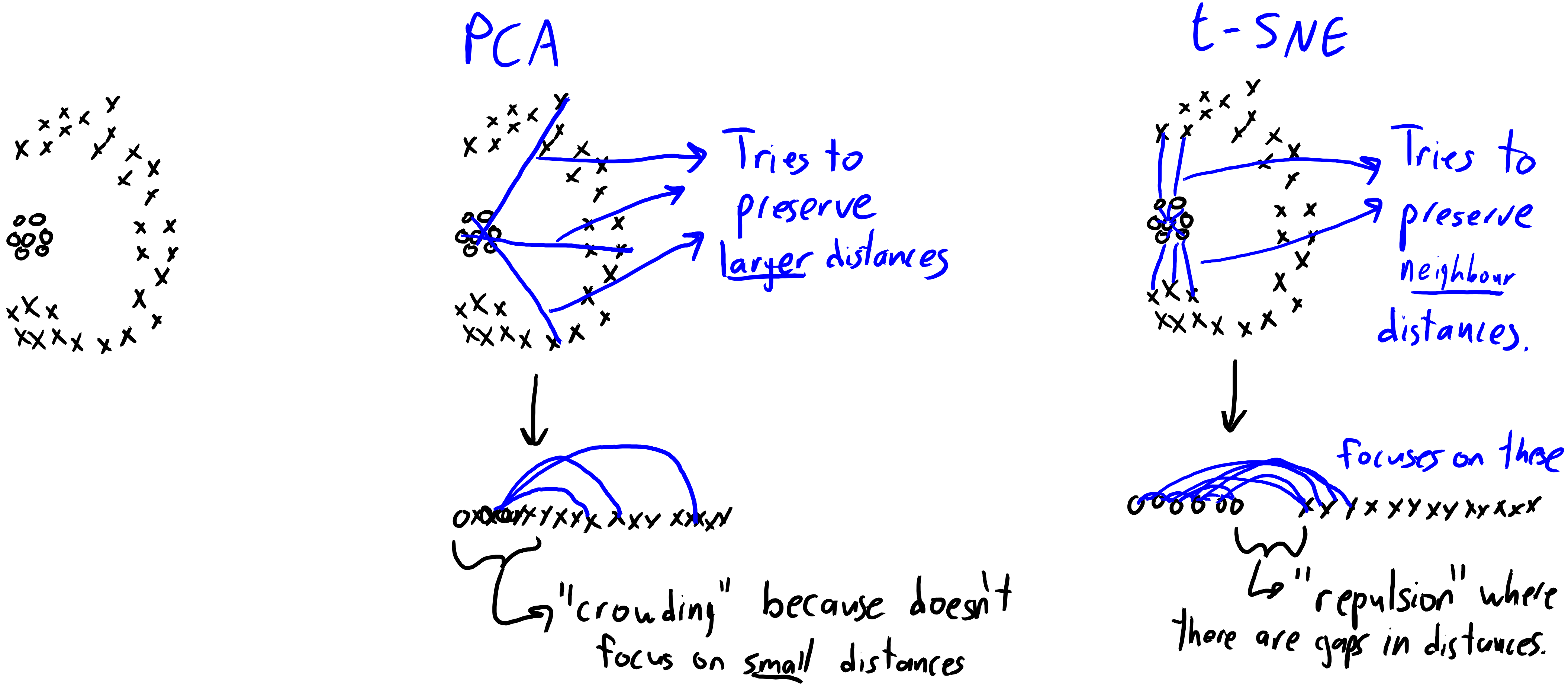


Remember this is unsupervised, algorithms do not know the labels.

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

t-Distributed Stochastic Neighbour Embedding

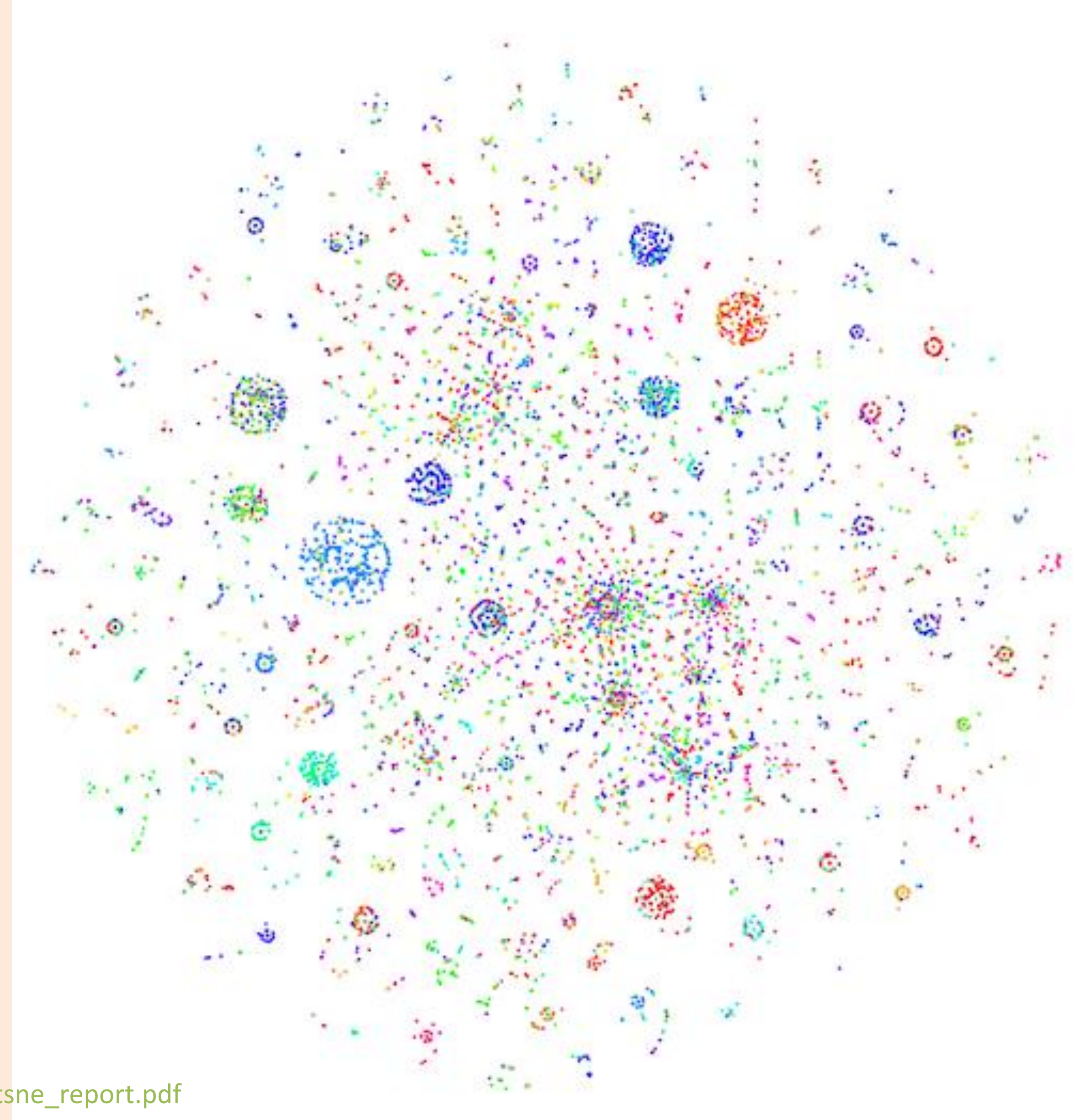
- One key idea in t-SNE:
  - Focus on distance to “neighbours”(allow large variance in other distances)



t-Distributed Stochastic Neighbour Embedding

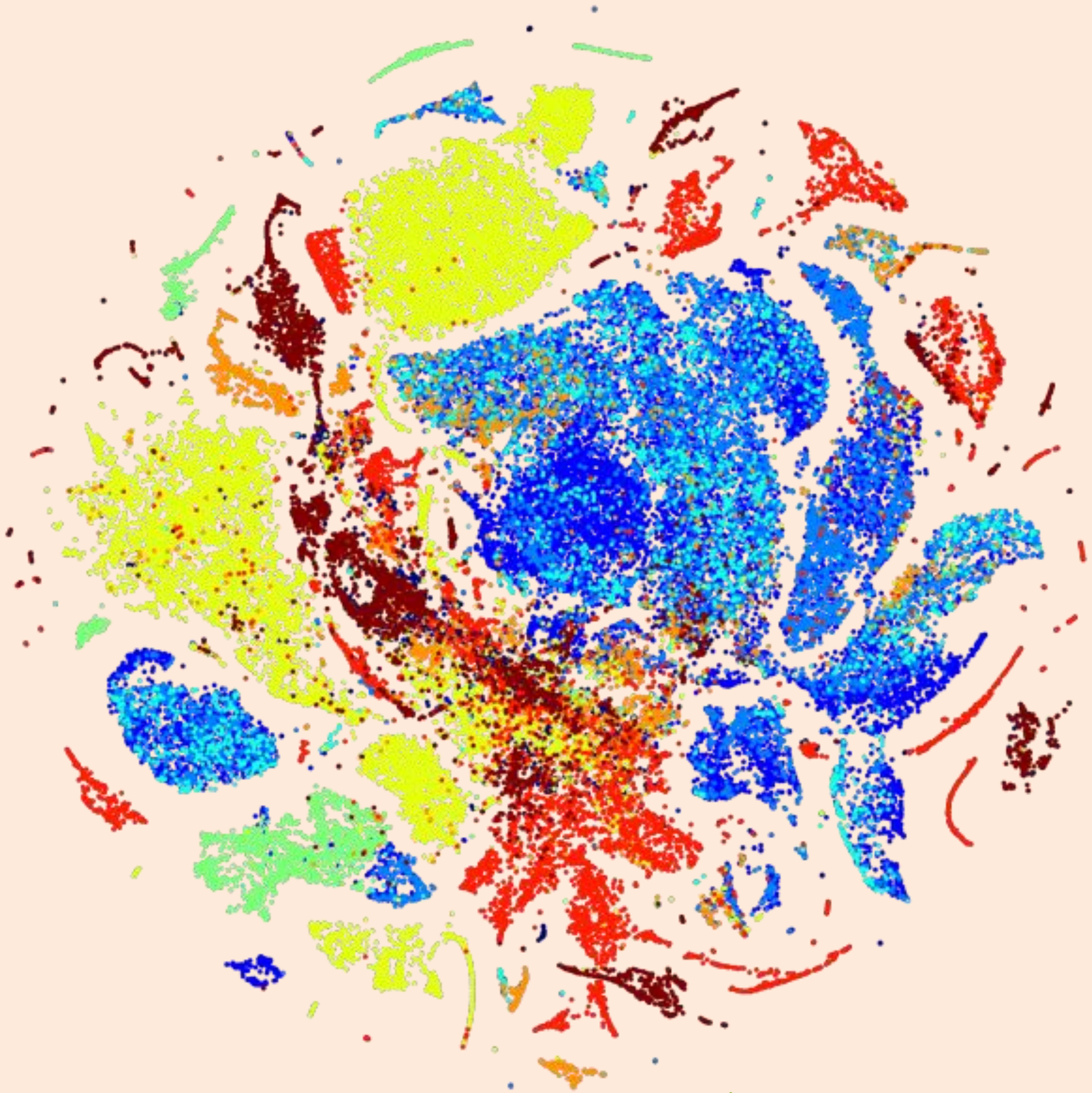
- t-SNE is a special case of MDS (specific d_1, d_2, and d_3 choices):
  - d_1: for each x_i, compute probability that each x_j is a 'neighbour'.
    - Computation is similar to k-means++, but most weight to close points (Gaussian).
    - Doesn't require explicit graph.
  - d_2: for each z_i, compute probability that each z_j is a 'neighbour'.
    - Similar to above, but uses student's t (grows really slowly with distance).
    - Avoids 'crowding', because you have a huge range that large distances can fill.
  - d_3: Compare x_i and z_i using an entropy-like measure:
    - How much 'randomness' is in probabilities of x_i if you know the z_i (and vice versa)?
- Interactive demo: https://distill.pub/2016/misread-tsne

t-SNE on Wikipedia Articles



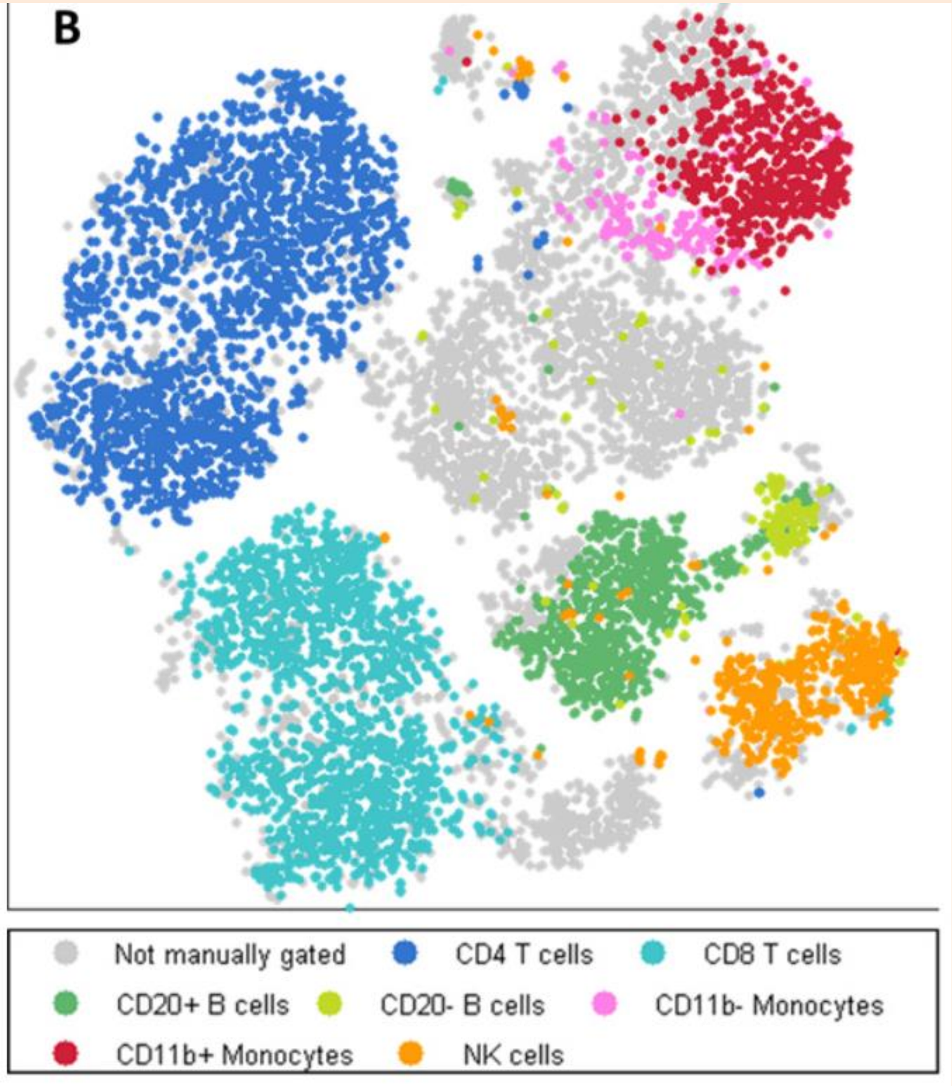
http://jasneetsabharwal.com/assets/files/wiki_tsne_report.pdf

t-SNE on Product Features



http://blog.kaggle.com/2015/06/09/otto-product-classification-winners-interview-2nd-place-alexander-guschin/

t-SNE on Leukemia Heterogeneity



http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4076922/

(pause)

Latent-Factor Representation of Words

- For natural language, we often represent words by an index.
  - E.g., "cat" is word 124056 among a "bag of words".
- But this may be inefficient:
  - Should "cat" and "kitten" share parameters in some way?
- We want a latent-factor representation of individual words:
  - Closeness in latent space should indicate similarity.
  - Distances could represent meaning?
- Recent alternative to PCA/NMF is word2vec…

Using Context

- Consider these phrases:
  - “the cat purred”
  - “the kitten purred”

  - “black cat ran”
  - “black kitten ran”

- Words that occur in the same context likely have similar meanings.

- Word2vec uses this insight to design an MDS distance function.

Word2Vec

- Two common word2vec approaches:
  1. Try to predict word from surrounding words (continuous bag of words).
  2. Try to predict surrounding words from word (skip-gram).

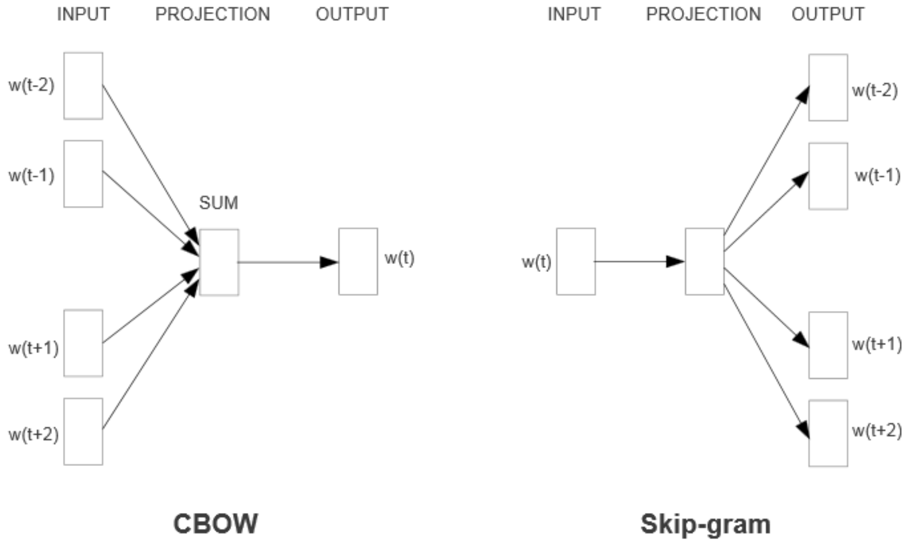


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

- Train latent-factors to solve one of these supervised learning tasks.

https://arxiv.org/pdf/1301.3781.pdf

Word2Vec

- In both cases, each word 'i' is represented by a vector z_i.
- In continuous bag of words (CBOW), we optimize the following likelihood:

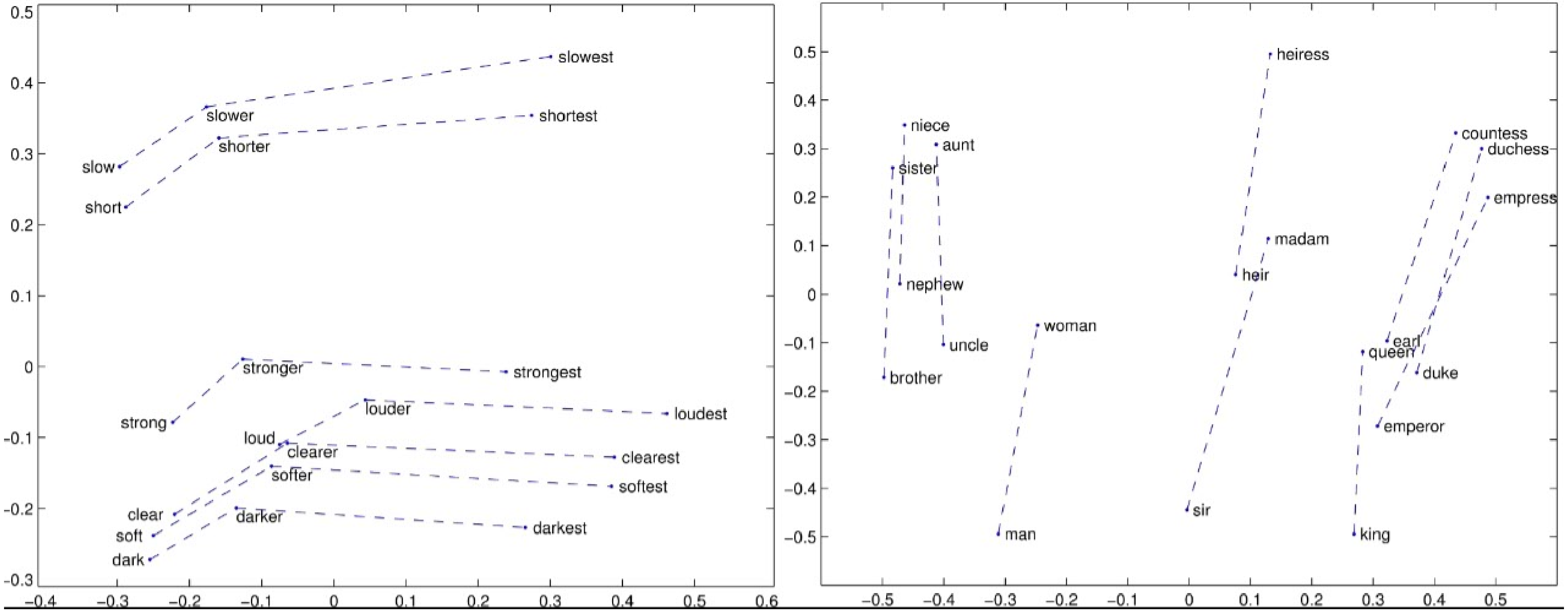
$$p(x_i \mid x_{surround}) = \prod_{j \in surround} p(x_i \mid x_j) \quad \text{(independence assumption)}$$

$$= \prod_{j \in surround} \frac{\exp(z_i^T z_j)}{\sum_{c=1}^{k} \exp(z_c^T z_j)} \quad \text{(softmax over all words)}$$

- Apply gradient descent to logarithm:
  - Encourages $z_i^T z_j$ to be big for words in same context (making z_i close to z_j).
  - Encourages $z_i^T z_j$ to be small for words not appearing in same context (makes z_i and z_j far).
- For CBOW, denominator sums over all words.
- For skip-gram it will be over all possible surrounding words.
  - Common trick to speed things up: sample terms in denominator ("negative sampling").

Word2Vec Example

- MDS visualization of a set of related words:



- Distances between vectors might represent semantics.

http://sebastianruder.com/secret-word2vec

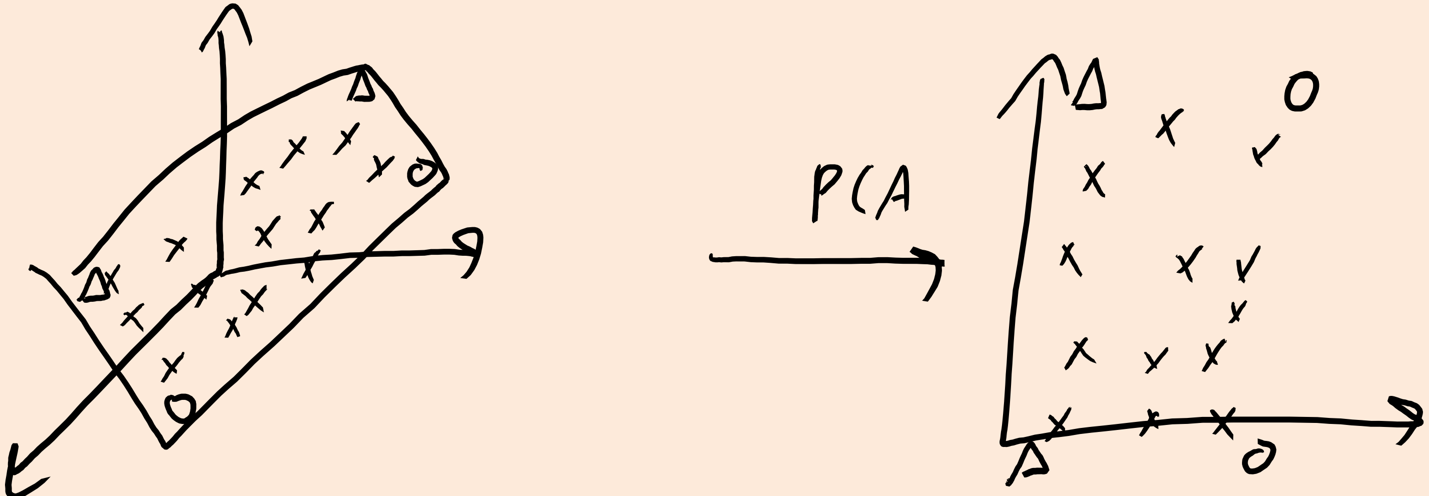
Summary

- 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.
- Word2vec:
  - Latent-factor (continuous) representation of words.
  - Based on predicting word from its context (or context from word).

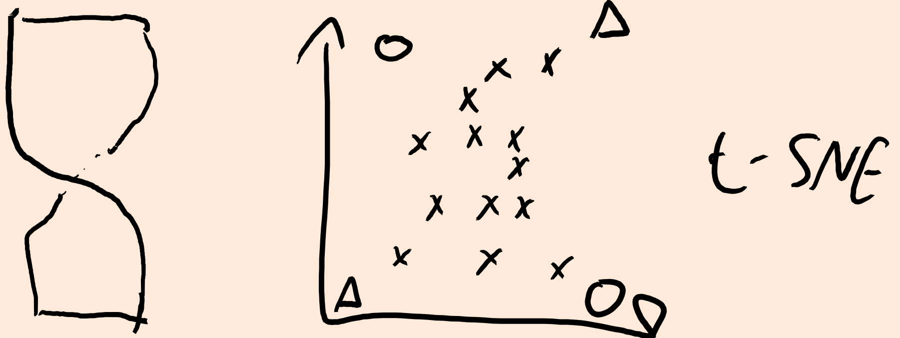
- Next time: deep learning.

Does t-SNE always outperform PCA?

- Consider 3D data living on a 2D hyper-plane:

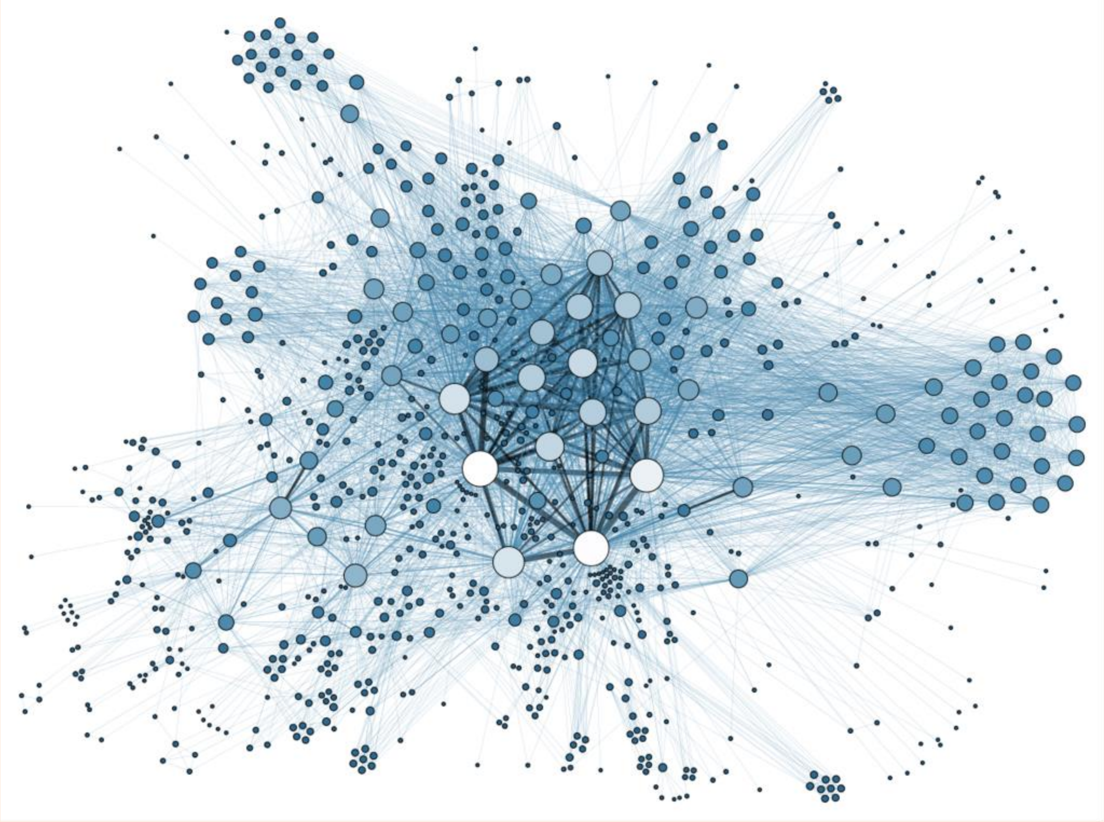


- PCA can perfectly capture the low-dimensional structure.
- T-SNE can capture the local structure, but can “twist” the plane.
  - It doesn’t try to get long distances correct.



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: $\frac{d}{dw}[f(g(w))] = f'(g(w))\,g'(w)$
- The multivariate chain rule: $\underbrace{\nabla[f(g(w))]}_{d \times 1} = \underbrace{f'(g(w))}_{1 \times 1}\underbrace{\nabla g(w)}_{d \times 1}$
- Example:

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

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

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

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

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

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

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 $g(d_1, d_2) = \frac{1}{2}(d_1 - d_2)^2$

$$\nabla_{z_i} g(d_1(x_i, x_j), d_2(z_i, z_j)) = \underbrace{-(d_1(x_i, x_j) - d_2(z_i, z_j))}_{g'(d_1, d_2)} \underbrace{\left[- \frac{(z_i - z_j)}{2\|z_i - z_j\|} \right]}_{\nabla_{z_i} d_2(z_i, z_j)}$$

↳ Assuming $z_i \neq z_j$

$g'(d_1, d_2)$ (move distances closer)

$\nabla_{z_i} d_2(z_i, z_j)$ (how distance changes in z-space)