

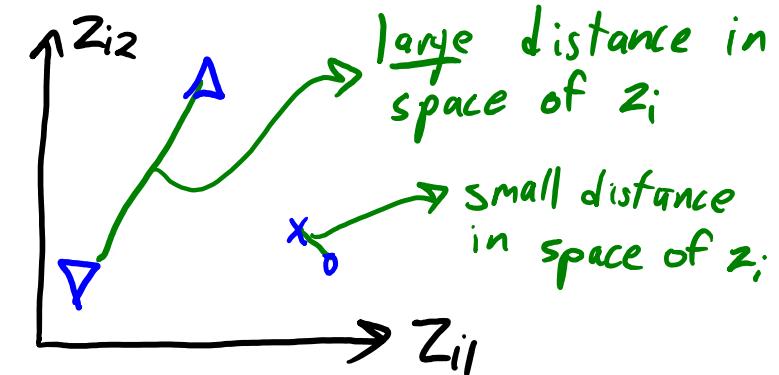
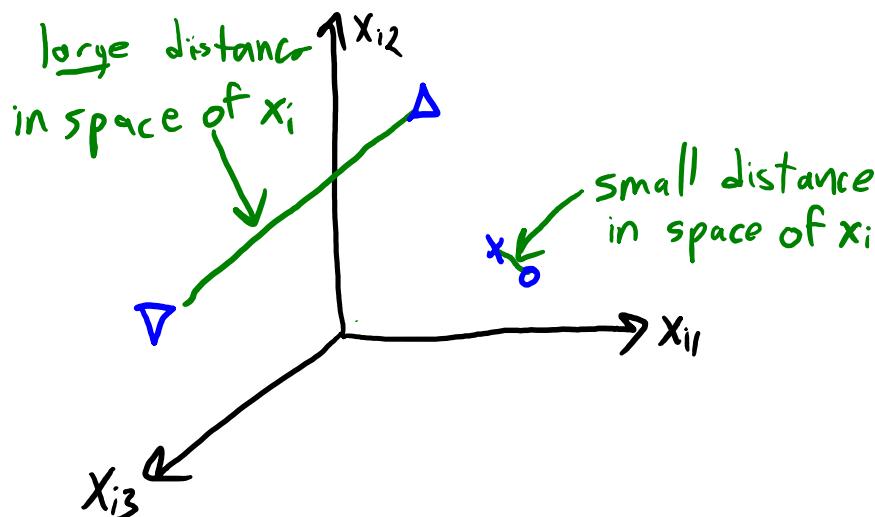
CPSC 340: Machine Learning and Data Mining

Multi-Dimensional Scaling

Last Time: Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS):
 - Non-parametric visualization: directly optimize the z_i locations.

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



- Traditional MDS methods lead to a “crowding” effect.

Different MDS Cost Functions

- MDS default objective: squared difference of Euclidean norm:

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

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

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

- 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.
- the functions d_1, d_2, d_3 are not necessarily the same

Classic Multi-Dimensional Scaling (MDS)

- MDS default objective function with general distances/similarities:

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

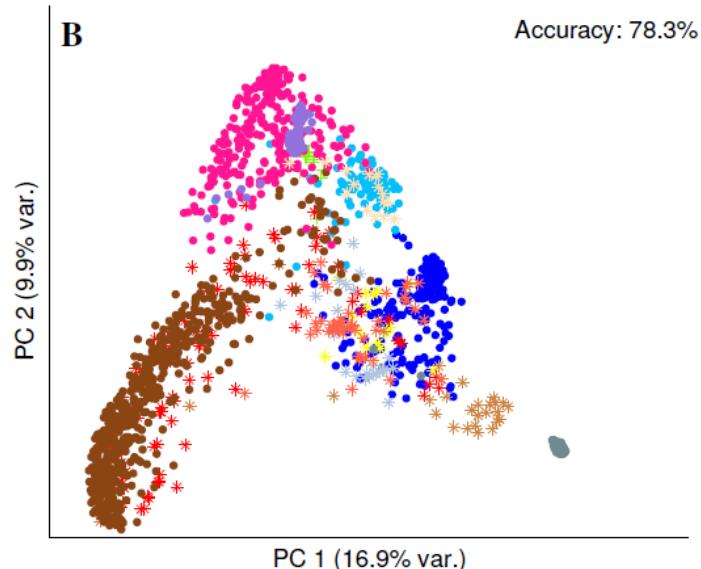
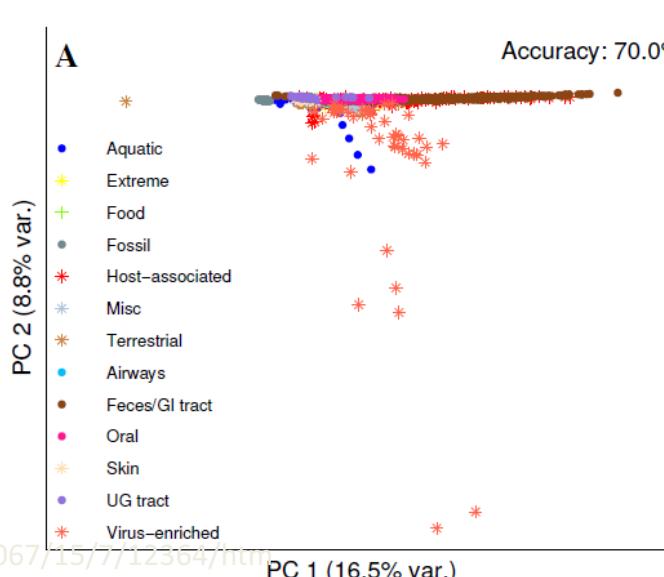
- “Classic” MDS uses $d_1(x_i, x_j) = x_i^T x_j$ and $d_2(z_i, z_j) = z_i^T z_j$.
 - We obtain PCA in this special case (for centered x_i).
 - Not a great choice because it’s a linear model.

Non-Euclidean Multi-Dimensional Scaling (MDS)

- MDS default objective function with general distances/similarities:

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

- 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

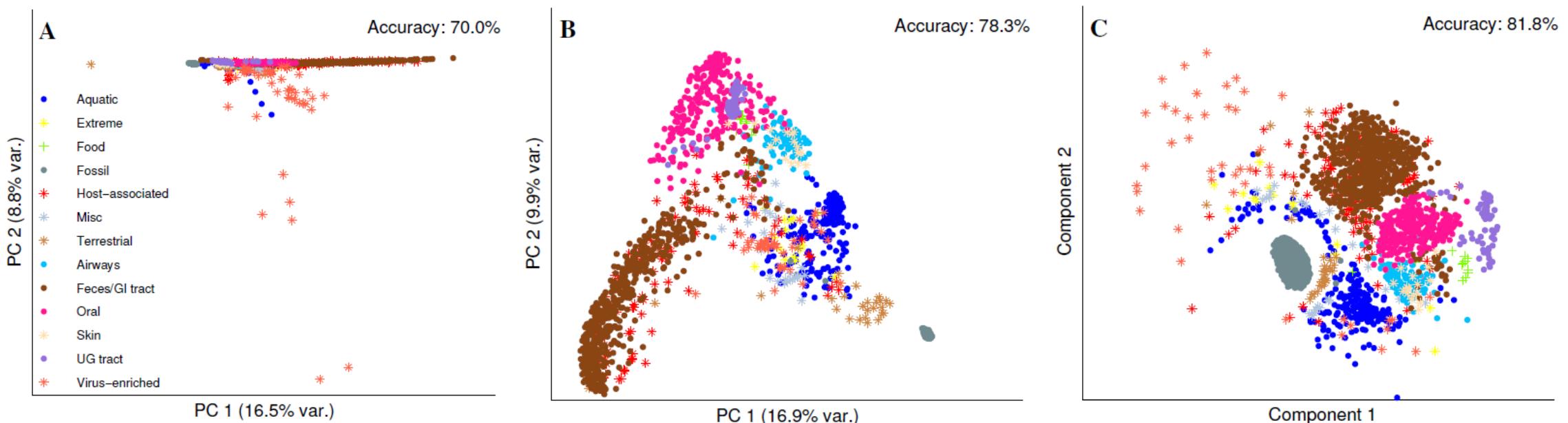
- Challenge for most MDS models: they **focus on large distances**.
 - Leads to “crowding” effect like with PCA.
- Early attempt to address this is **Sammon’s mapping**:
 - **Weighted MDS** so large/small distances more comparable.

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

- Denominator reduces focus on large distances.

Sammon's Mapping

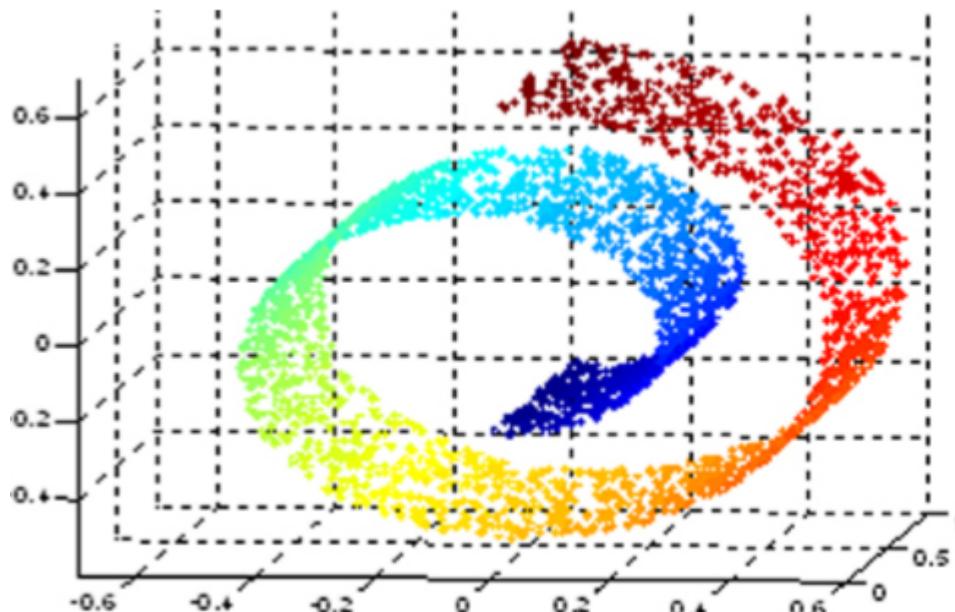
- Challenge for most MDS models: they **focus on large distances**.
 - Leads to “crowding” effect like with PCA.
- Early attempt to address this is **Sammon’s mapping**:
 - Weighted MDS** so large/small distances more comparable.



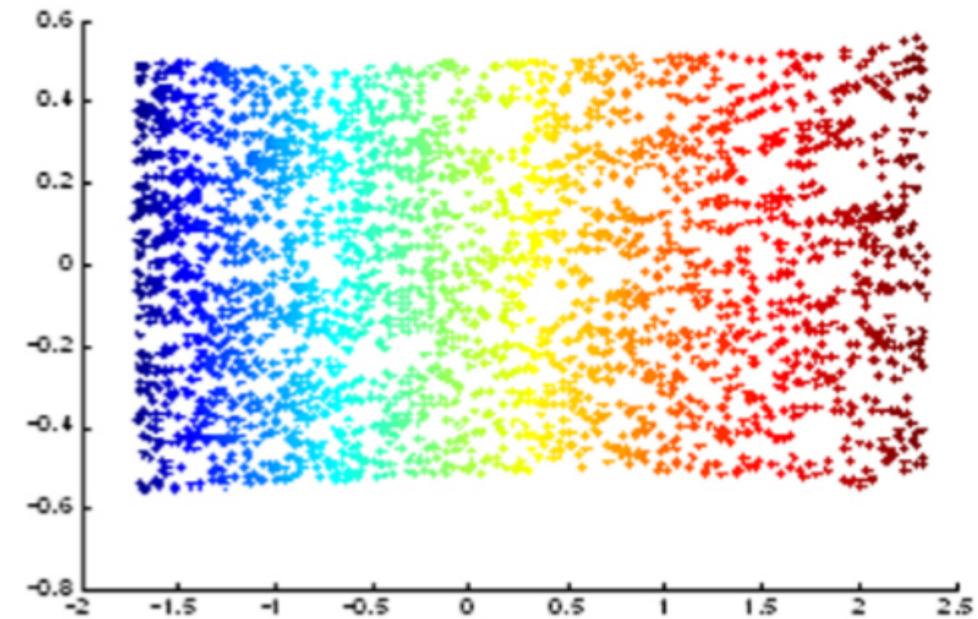
Learning Manifolds

- Consider data that lives on a **low-dimensional “manifold”**.
- Example is the ‘Swiss roll’:

Original data

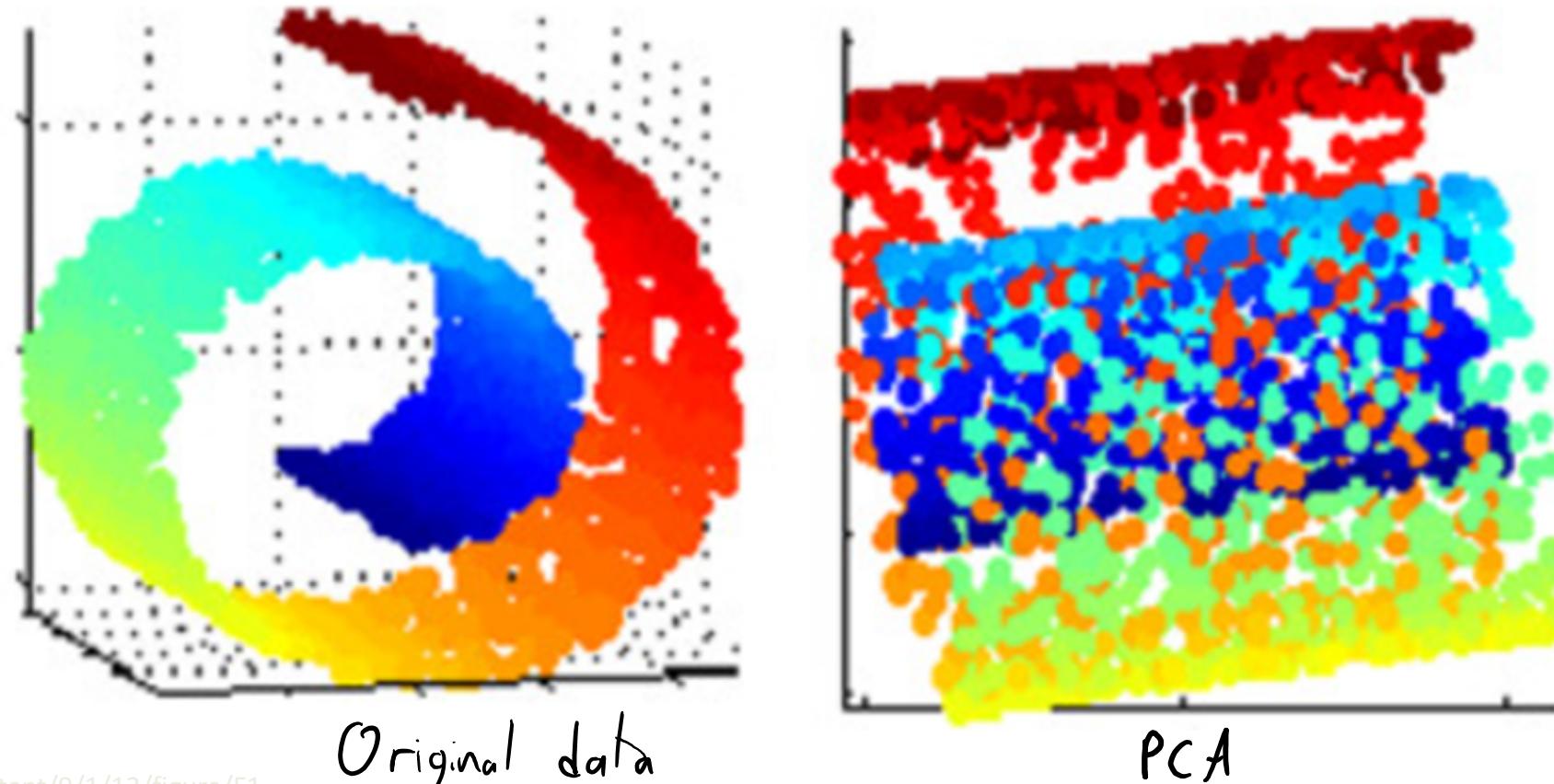


Two-dimensional manifold



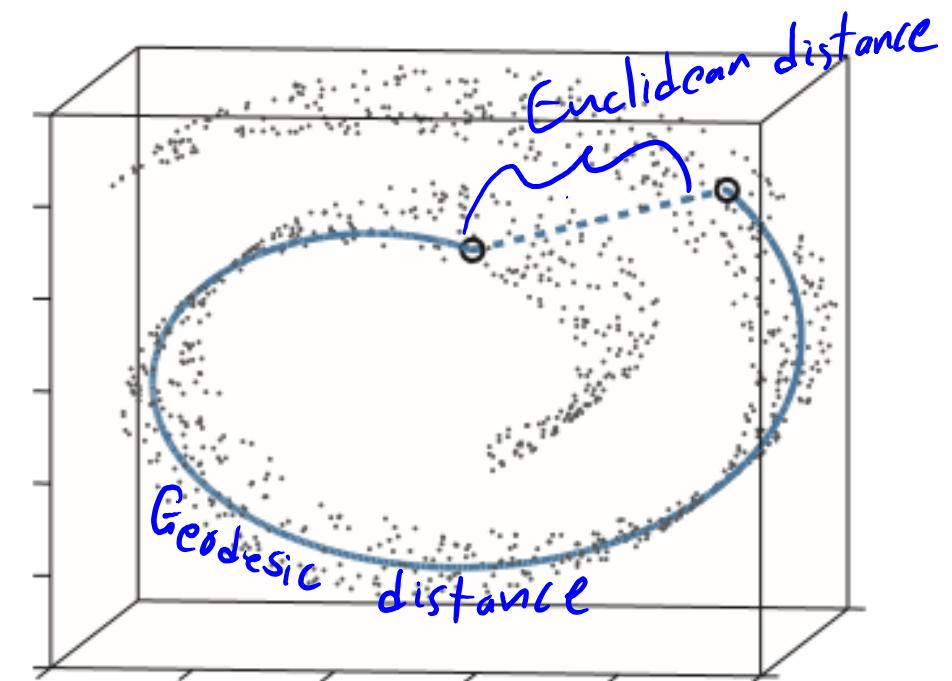
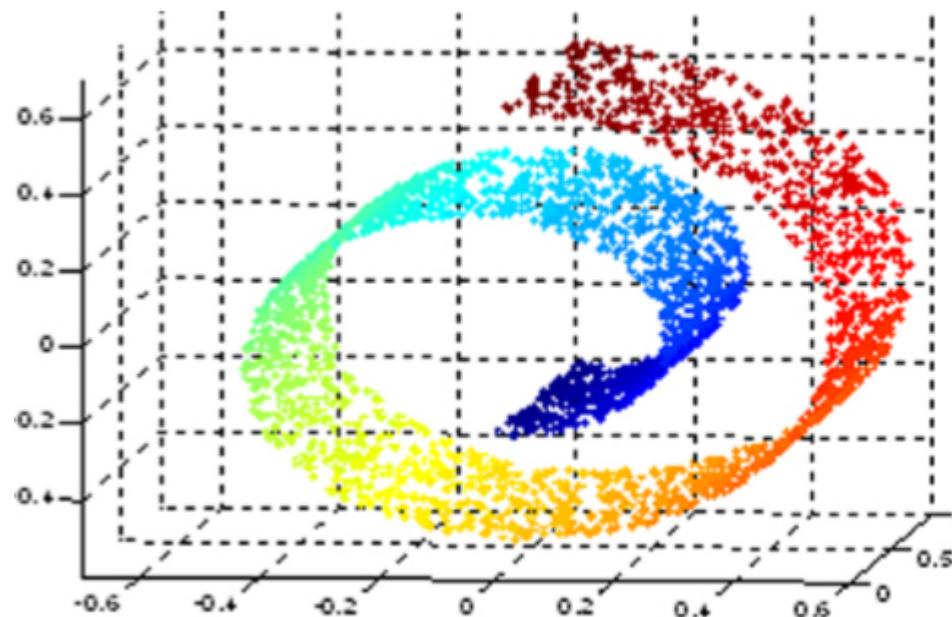
Learning Manifolds

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



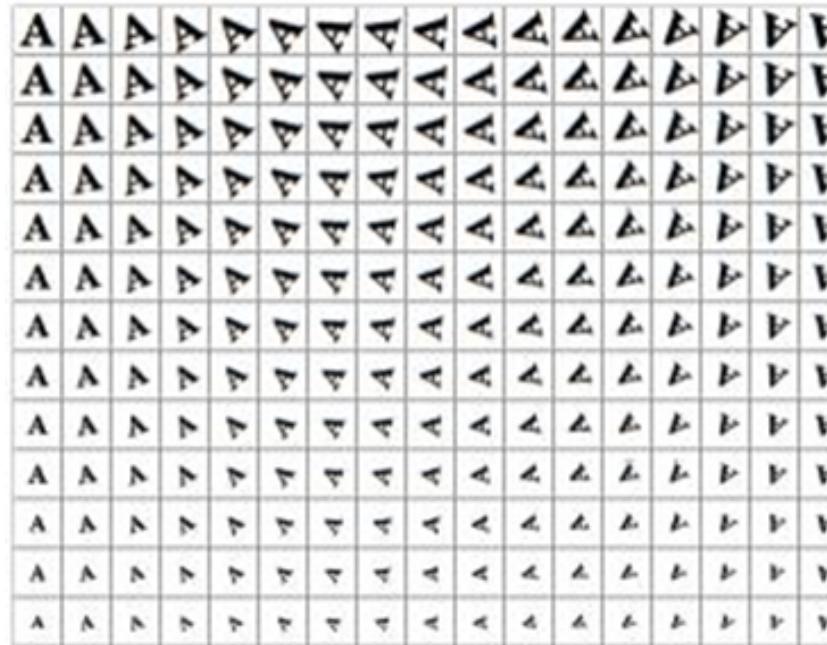
Learning Manifolds

- 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.



Manifolds in Image Space

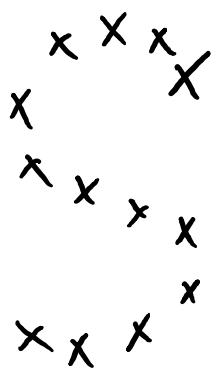
- Consider slowly-varying transformation of image:

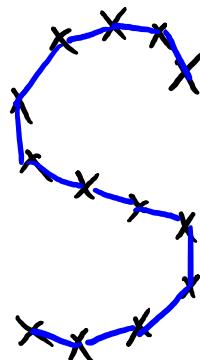


- Images are on a manifold in the high-dimensional space.
 - Euclidean distance doesn't reflect manifold structure.
 - Geodesic distance is distance through space of rotations/resizings.

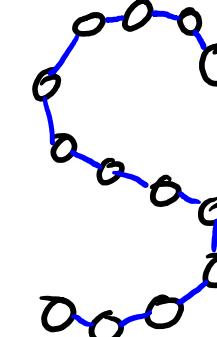
ISOMAP

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

 find "neighbours" of each point



Represent points and neighbours as a weighted graph.

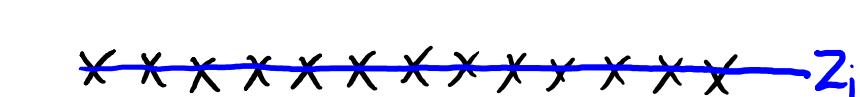


"weight" on each edge is distance between points

↓ Approximate geodesic distance by shortest path through graph.

$$D = \begin{bmatrix} 0 & 2 & 4 & 6 & \cdots \\ 2 & 0 & 2 & 4 & \cdots \\ 4 & 2 & 0 & 2 & \cdots \\ 6 & 4 & 2 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}_{12 \times 12}$$

I've assumed "weights" are all 2.
Run MDS with these approximate geodesic distances.

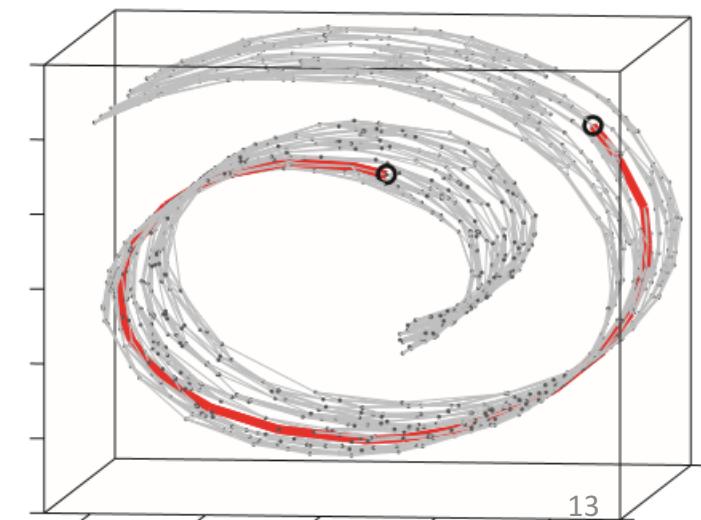
 ISOMAP z_i values in 1D or 2D

ISOMAP

- ISOMAP is latent-factor model for visualizing data on manifolds:
 1. Find the neighbours of each point.
 - Either “k-nearest neighbours graph”, or points within a radius ‘r’.
 2. Compute edge weights:
 - Usually distance between neighbours.
 3. Compute weighted shortest path between all points.
 4. Run MDS using these distances.

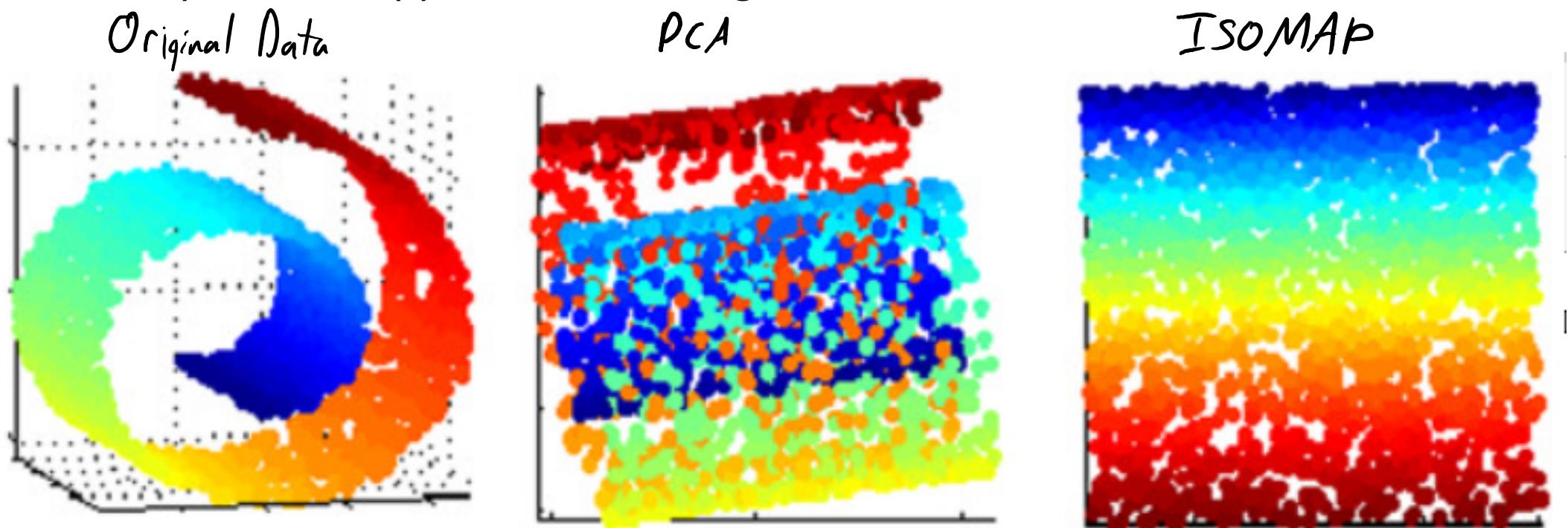
k-nearest neighbour graph:

- Find KNN for every point x_i
- “Or”-version: take edge $i-j$ if x_i is neighbour of x_j or vice versa.
- “And”-version: x_i and x_j must be KNNs of each other.



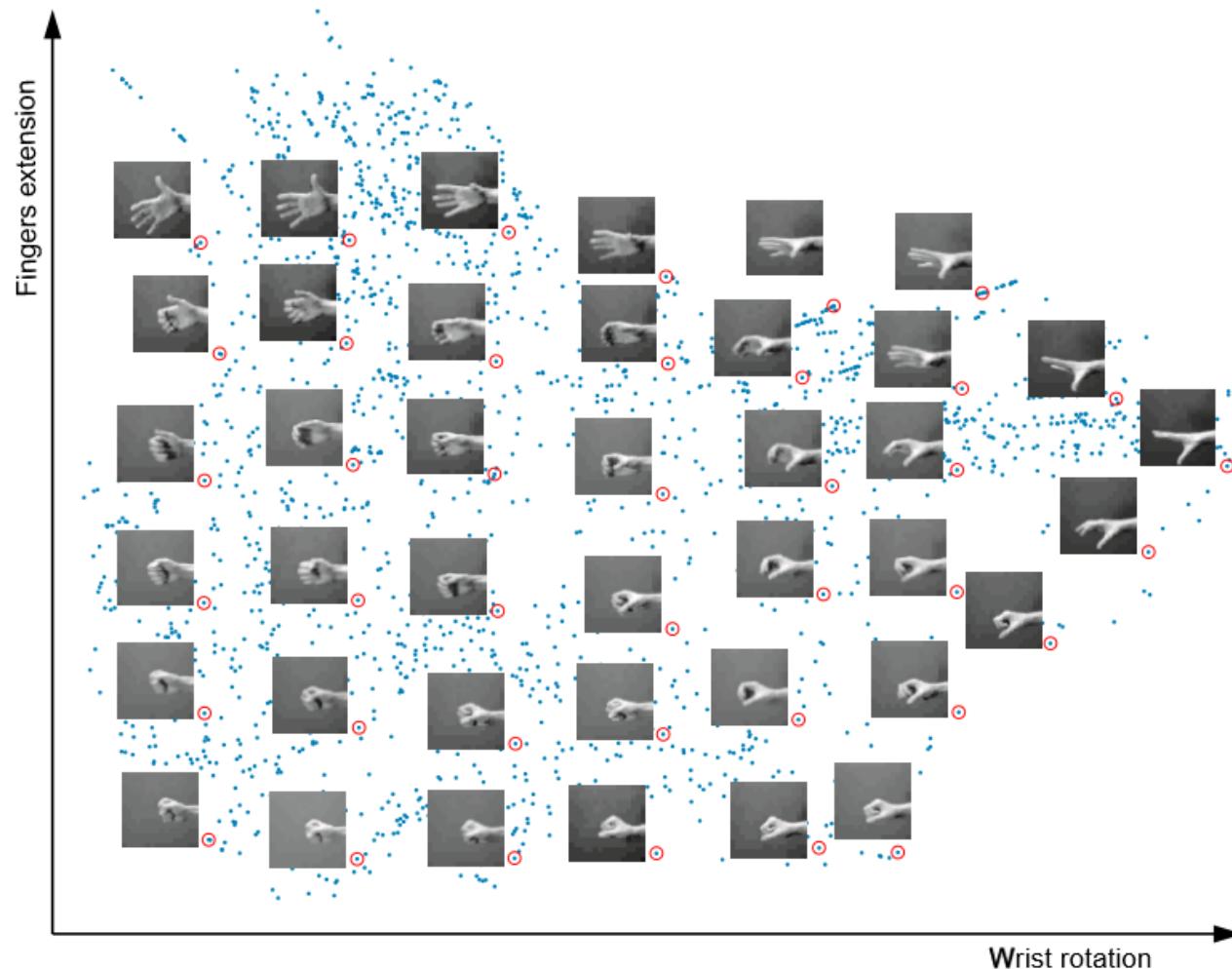
ISOMAP

- 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”.

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

Sammon's Map vs. ISOMAP vs. PCA

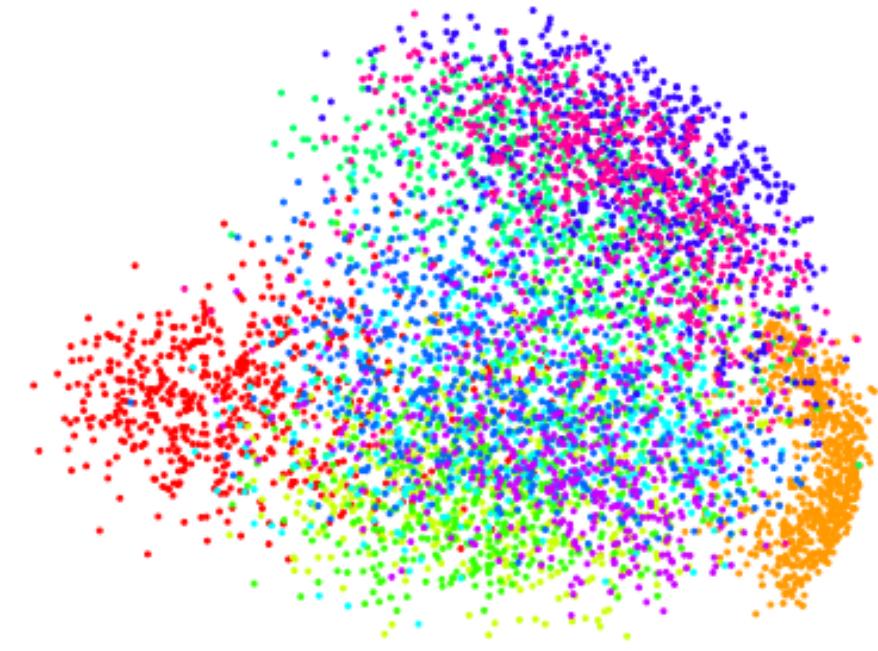
Sammon Map

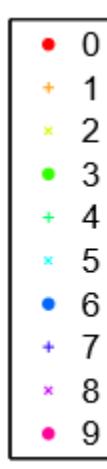


ISOMAP



PCA





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

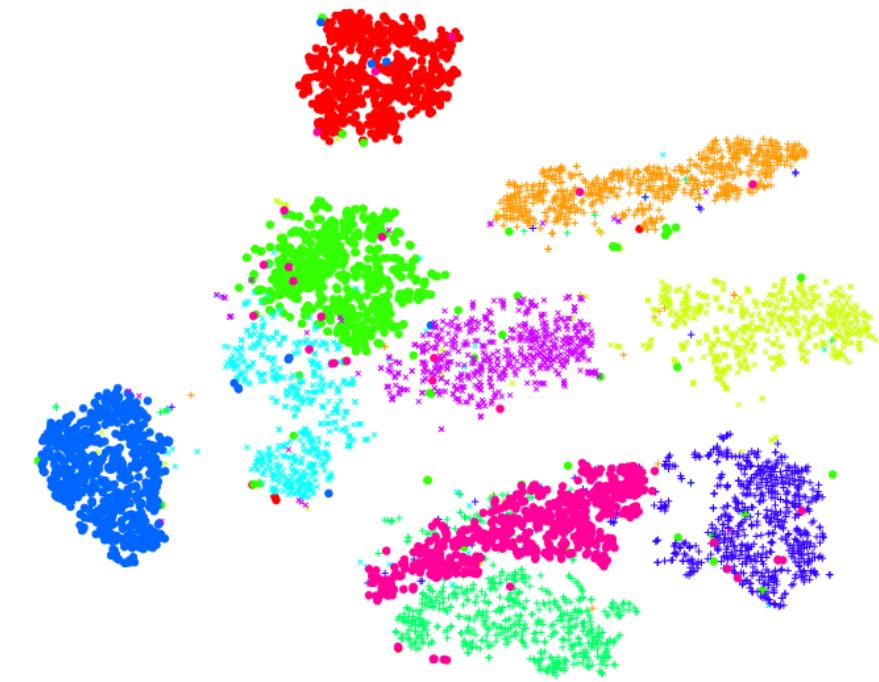
Sammon Map

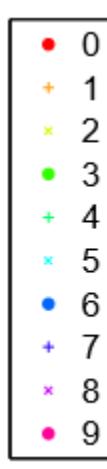


ISOMAP



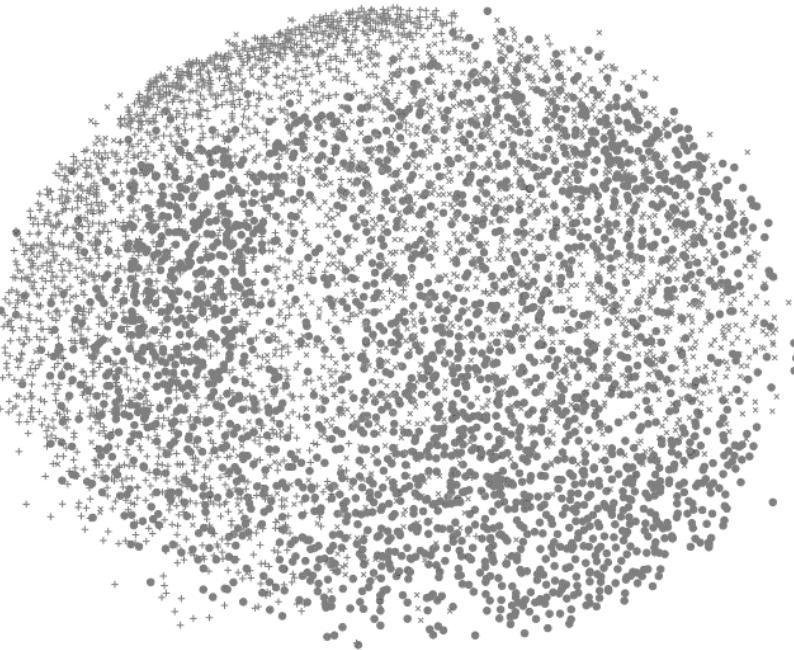
t-SNE



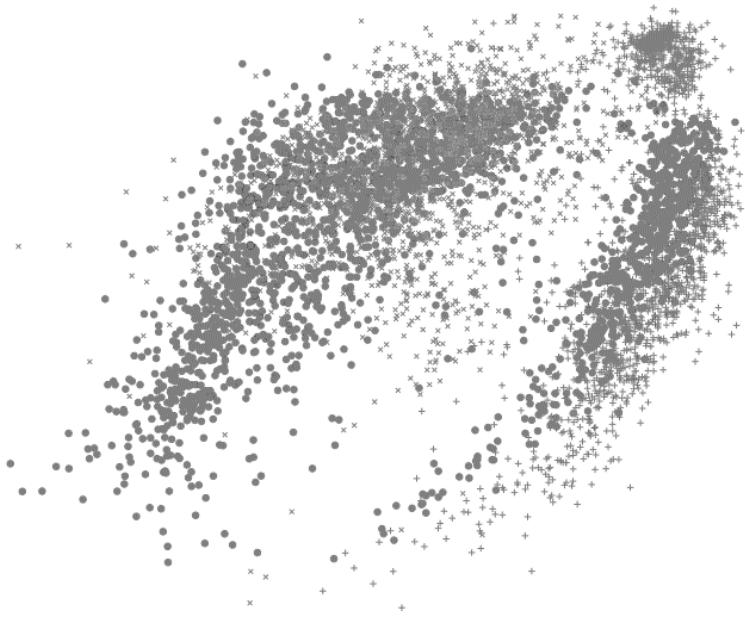


Sammon's Map vs. ISOMAP vs. t-SNE

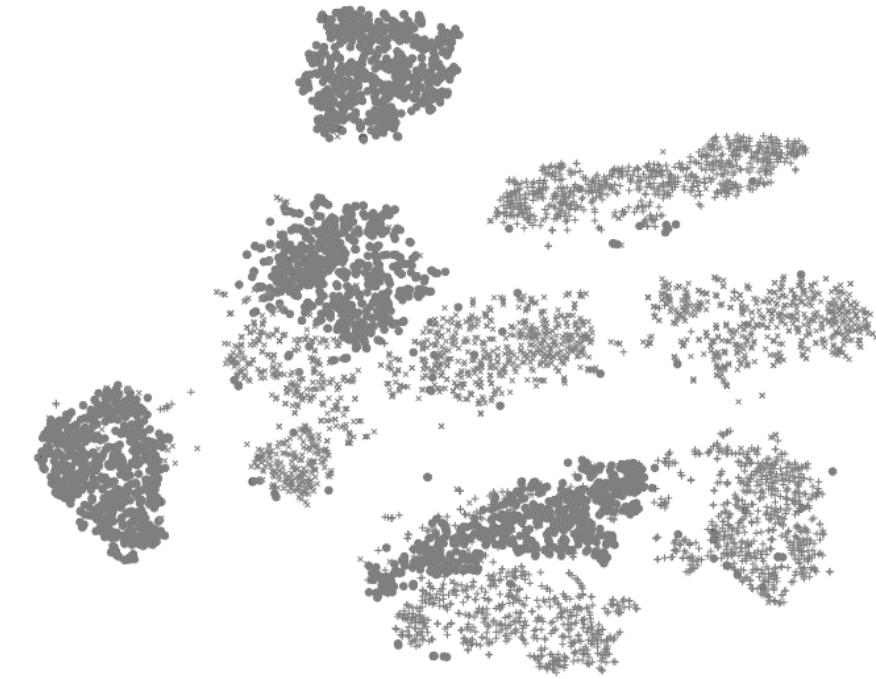
Sammon Map



ISOMAP



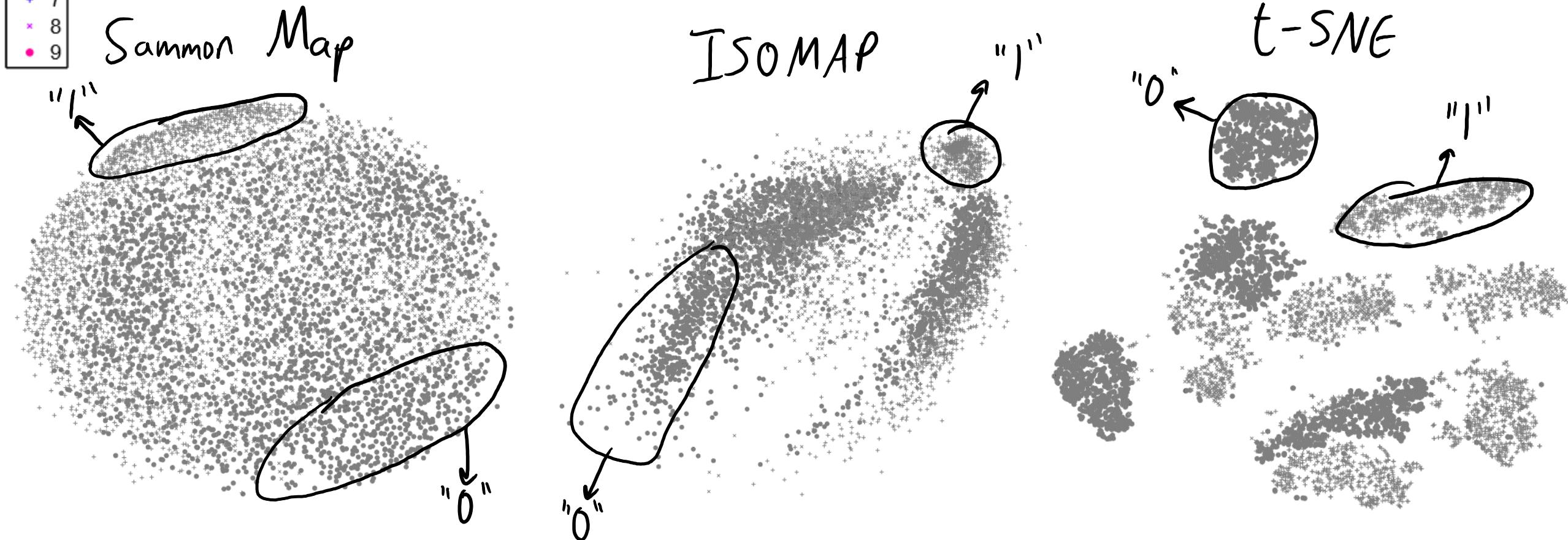
t-SNE



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

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

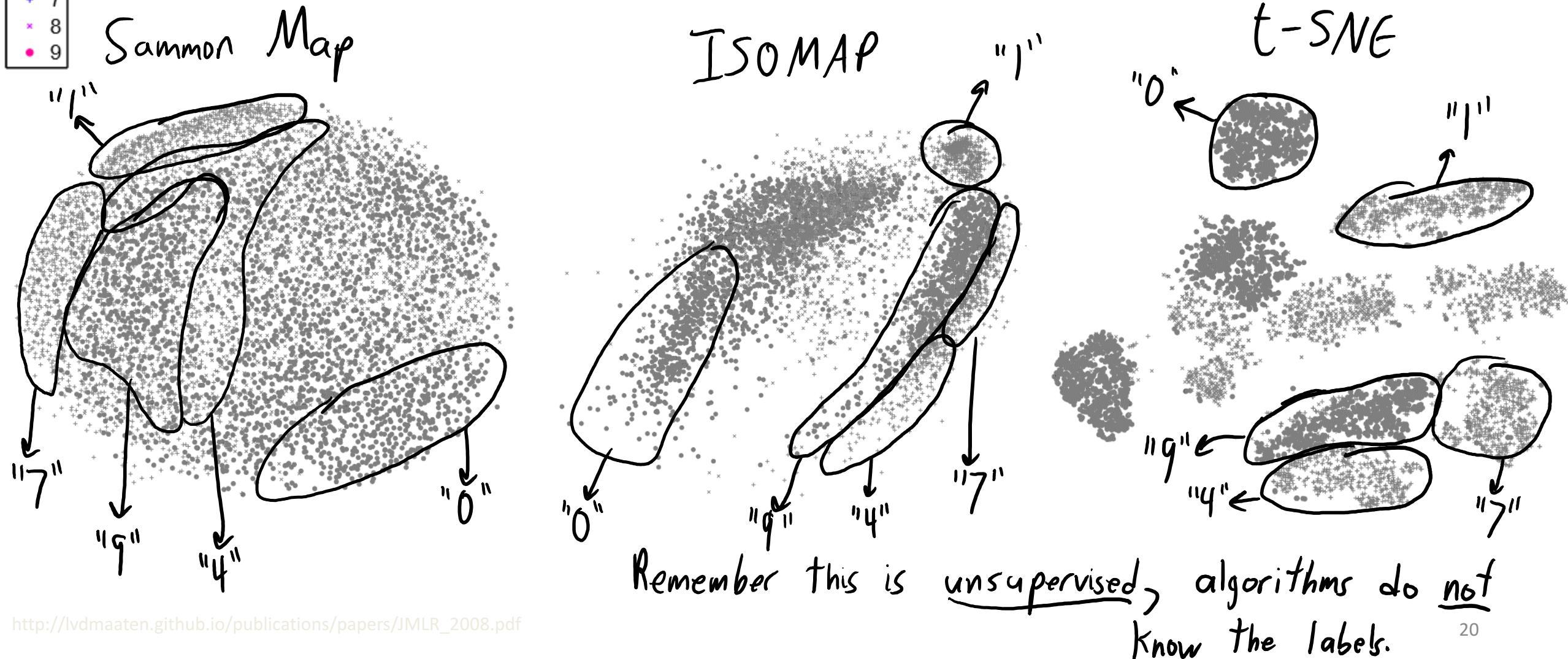
Sammon's Map vs. ISOMAP vs. t-SNE



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

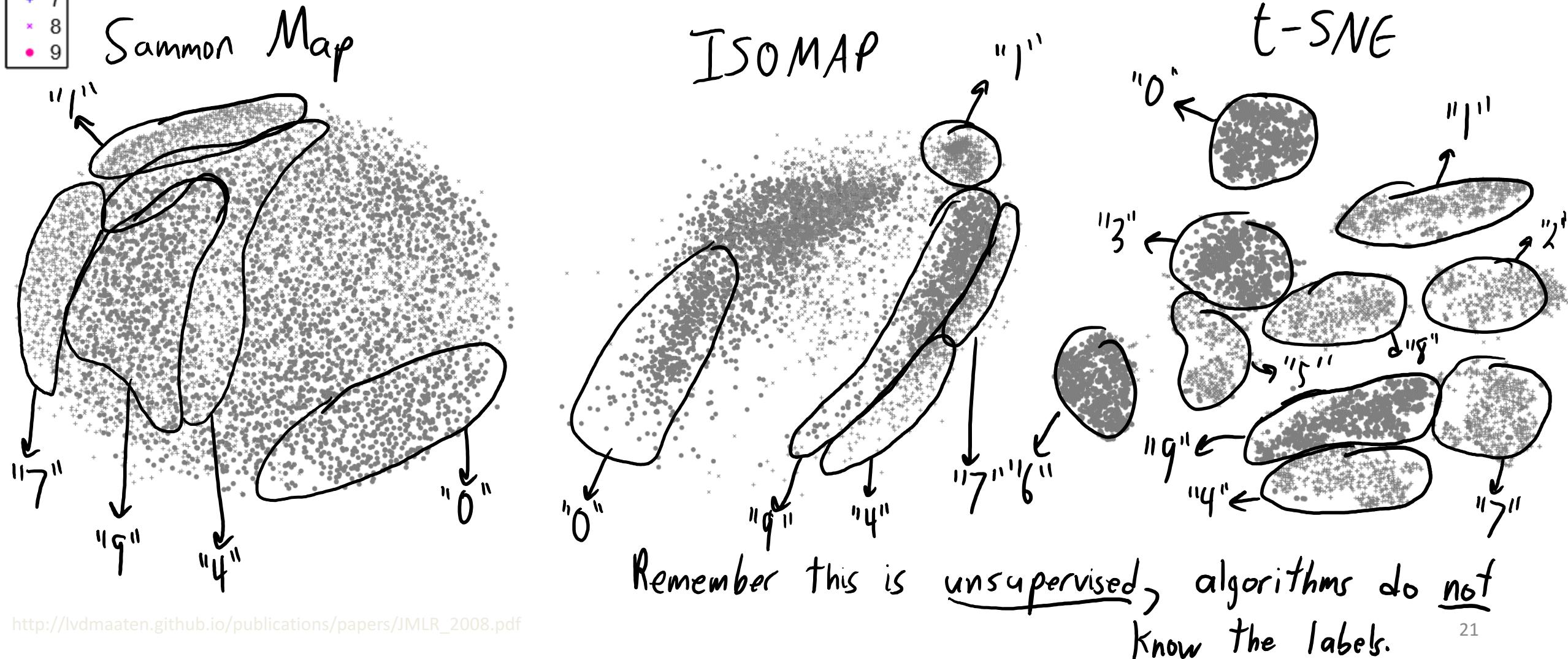
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

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



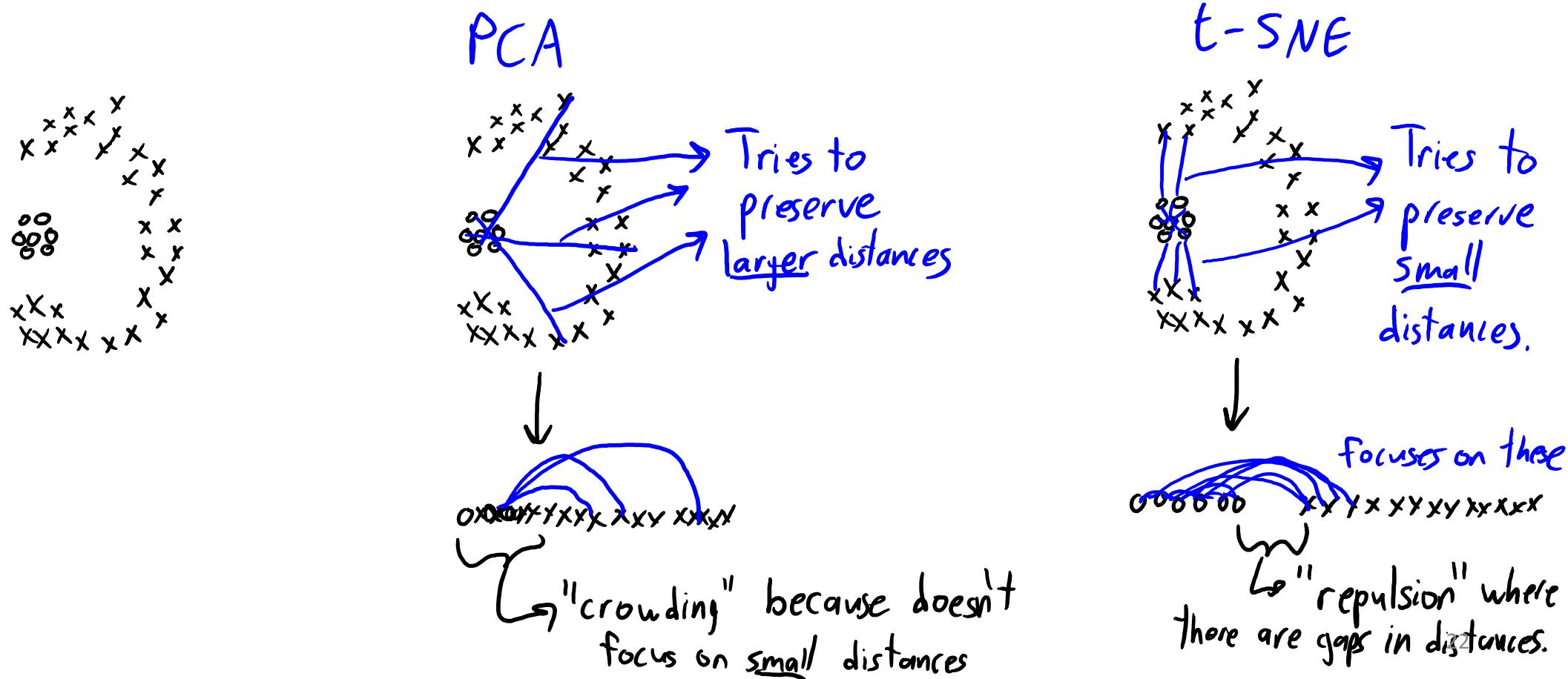
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

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



t-Distributed Stochastic Neighbour Embedding

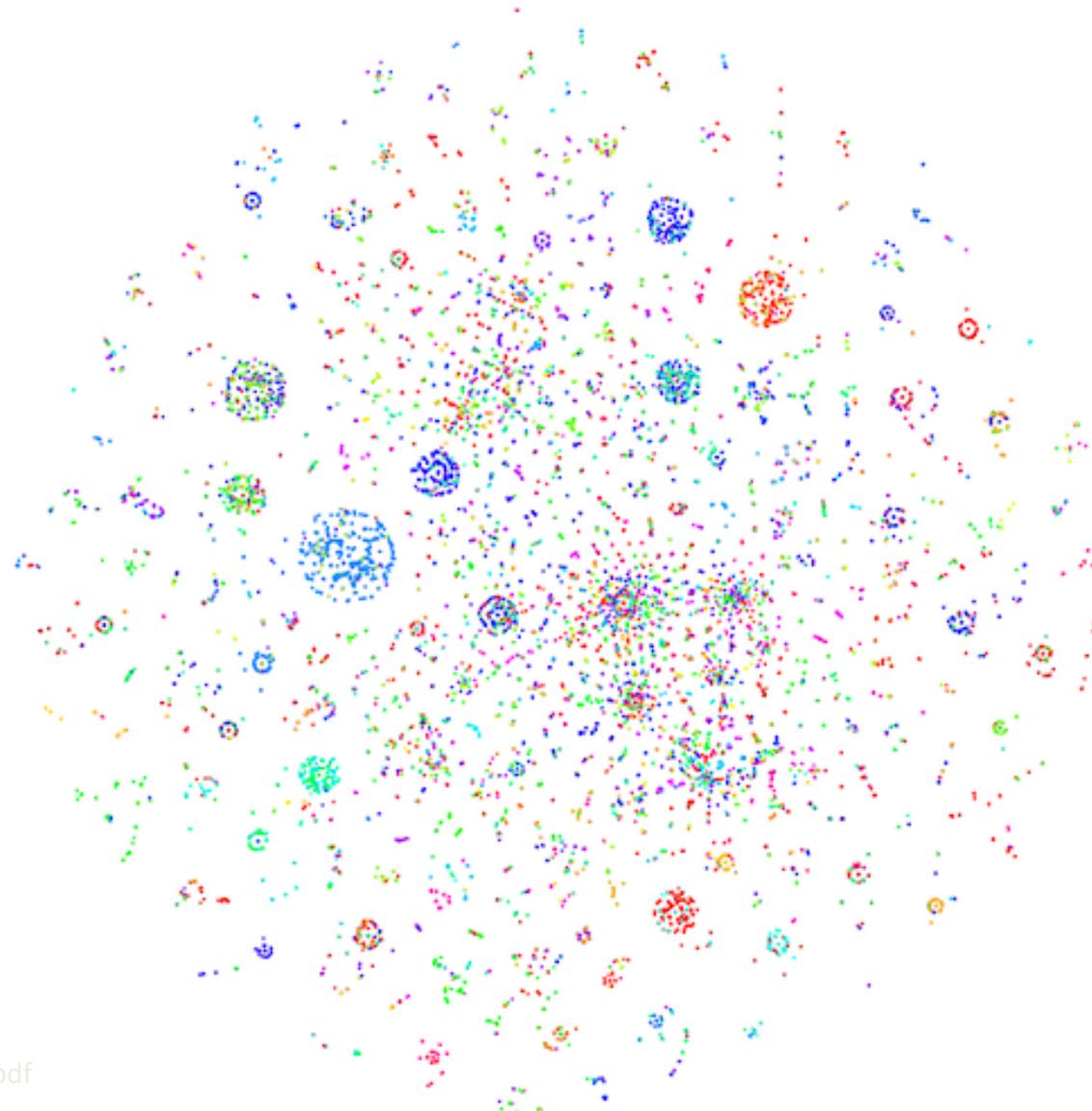
- One key idea in t-SNE:
 - Focus on small distances by allowing large variance in large distances.



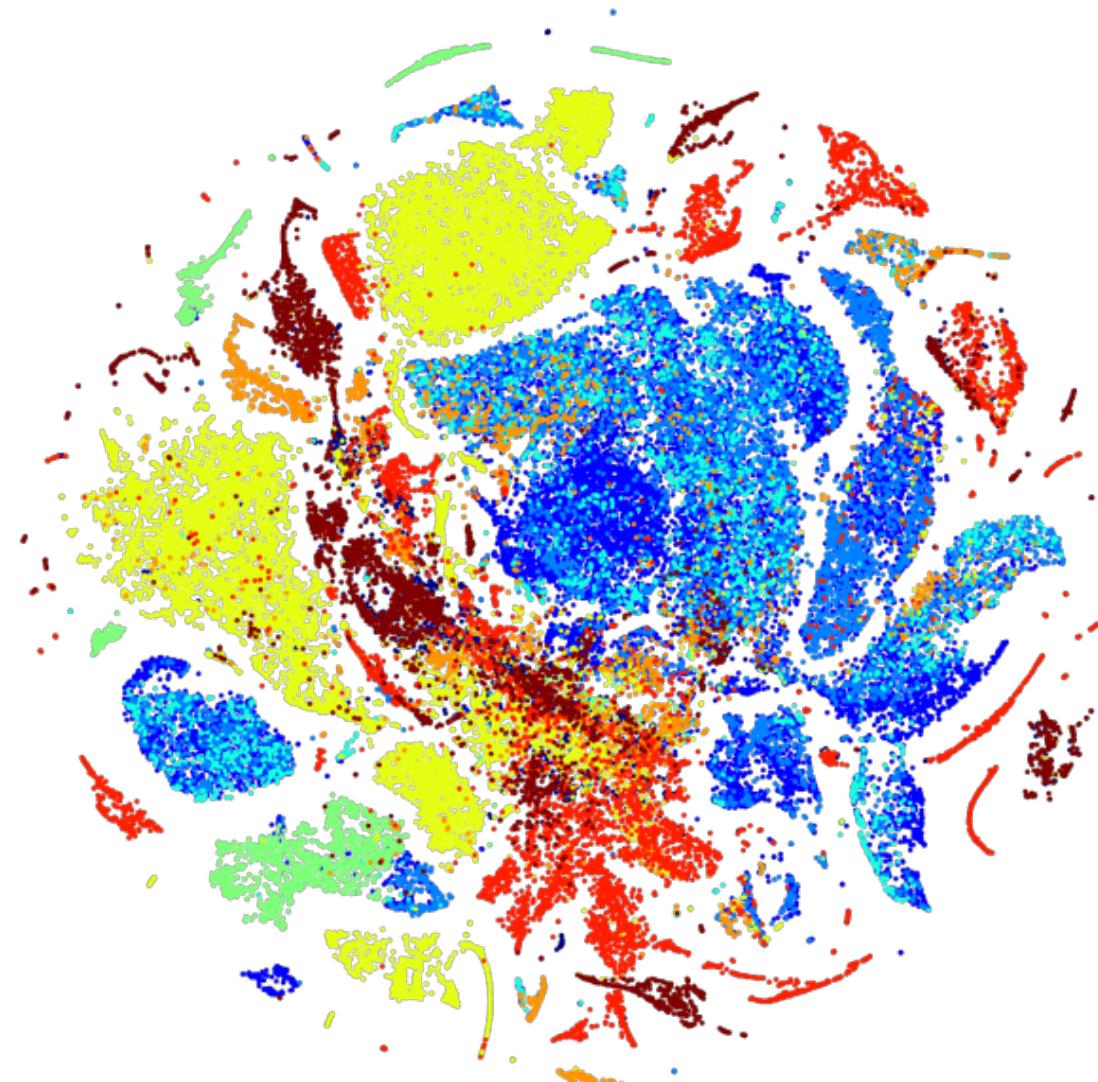
t-Distributed Stochastic Neighbour Embedding

- 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)?

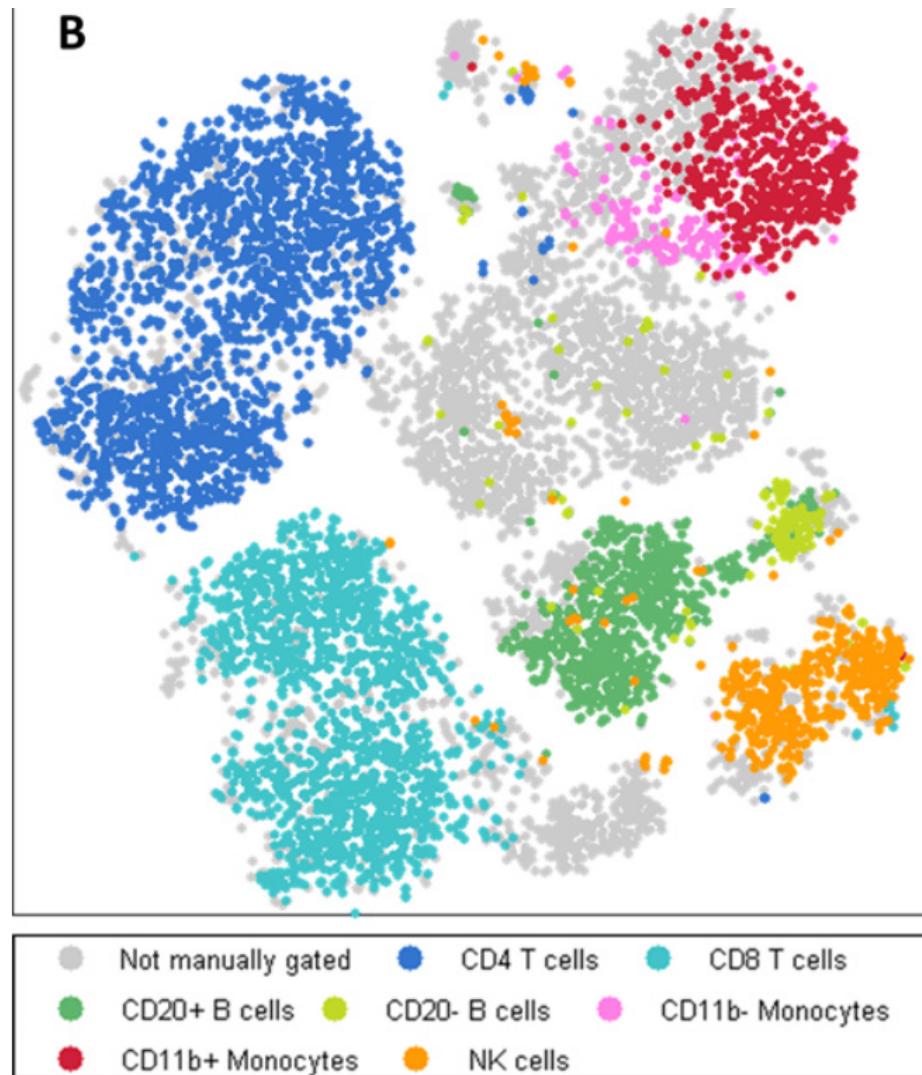
t-SNE on Wikipedia Articles



t-SNE on Product Features



t-SNE on Leukemia Heterogeneity



End of Part 4: Key Concepts

- We discussed **linear latent-factor models**:

$$\begin{aligned} f(W, Z) &= \sum_{i=1}^n \sum_{j=1}^d (w_j^\top z_i - x_{ij})^2 \\ &= \sum_{i=1}^n \|W^\top z_i - x_i\|^2 \\ &= \|Z^\top W - X\|_F^2 \end{aligned}$$

- Represent 'X' as linear combination of **latent factors 'W'**.
 - **Latent features 'Z'** gives a lower-dimensional version of 'X'.
 - When $k=1$, finds **direction that minimizes orthogonal distance**.
- Applications:
 - Outlier detection, dimensionality reduction, data compression, features for linear models, visualization, factor discovery, filling in missing entries.

End of Part 4: Key Concepts

- We discussed linear latent-factor models:

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d (w_j^\top z_i - x_{ij})^2$$

- Principal component analysis (PCA):

- Uses orthogonal factors and fits them sequentially.

- Non-negative matrix factorization:

- Uses non-negative factors giving sparsity.
 - Can be minimized with projected gradient.

- Many variations are possible:

- Different regularizers (sparse coding) or loss functions (robust/binary PCA).
 - Missing values (recommender systems) or change of basis (kernel PCA).

End of Part 4: Key Concepts

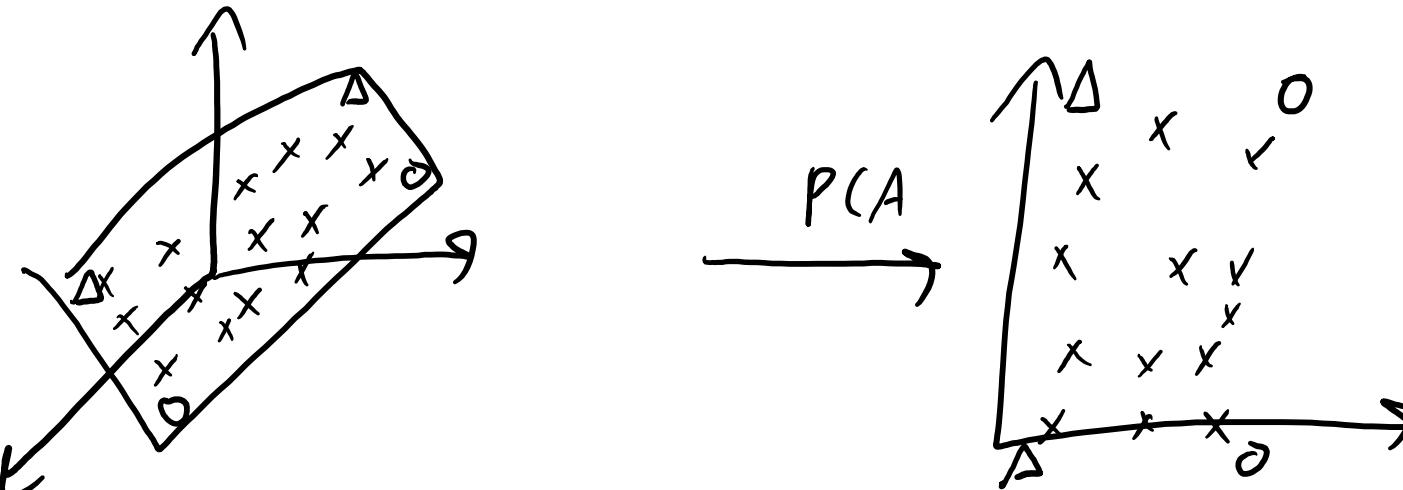
- We discussed **multi-dimensional scaling (MDS)**:
 - Non-parametric method for high-dimensional data visualization.
 - Tries to match distance/similarity in high-/low-dimensions.
 - “Gradient descent on scatterplot points”.
- Main challenge in MDS methods is “crowding” effect:
 - Methods focus on large distances and lose local structure.
- Common solutions:
 - Sammon mapping: use weighted cost function.
 - ISOMAP: approximate geodesic distance using via shortest paths in graph.
 - T-SNE: give up on large distances and focus on small distances.

Summary

- MDS with different 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: why is it called “neural”?

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.

