# CPSC 340: Machine Learning and Data Mining

## Multi-Dimensional Scaling

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

# Admin

- Assignment 5:
  - Due Friday

- Assignment 6:
  - Remember to request partner

# Latent-Factor Models for Visualization

- 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.
- 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.

# Multi-Dimensional Scaling

- Multi-dimensional scaling (MDS) is a crazy idea:
  - Let's directly optimize 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:

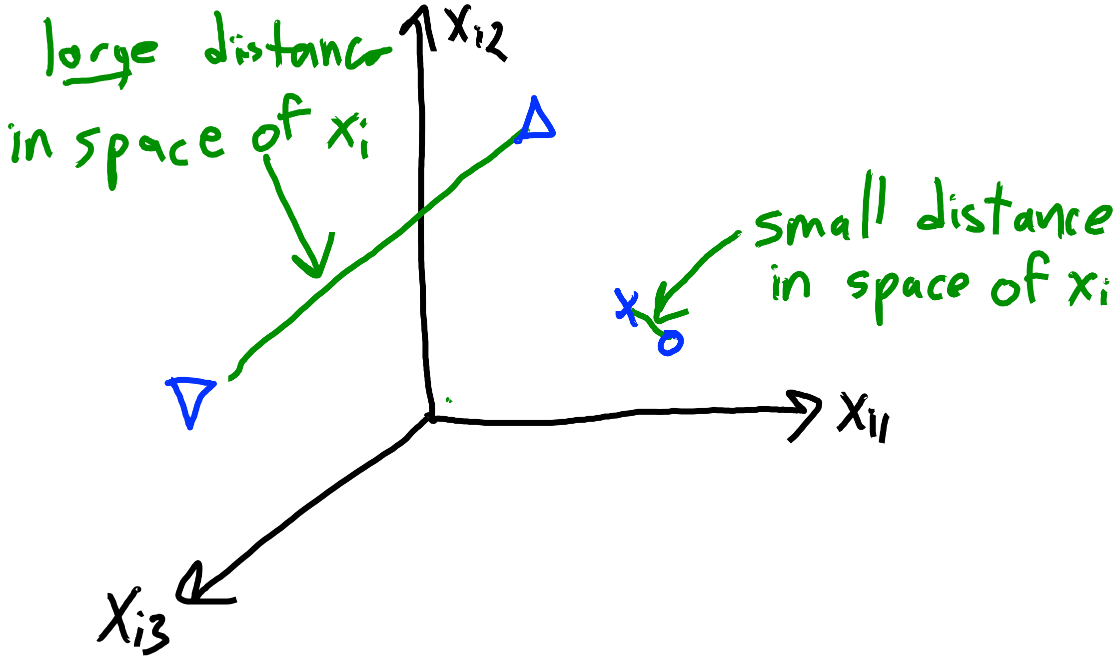
$$f(Z) = \underbrace{\sum_{i=1}^{n}\sum_{j=i+1}^{n}}_{\text{sum over pairs of examples}} \Big(\underbrace{\|z_i - z_j\|}_{\text{distance in scatterplot}} - \underbrace{\|x_i - x_j\|}_{\text{Distance between points in original 'd' dimensions}}\Big)^2$$

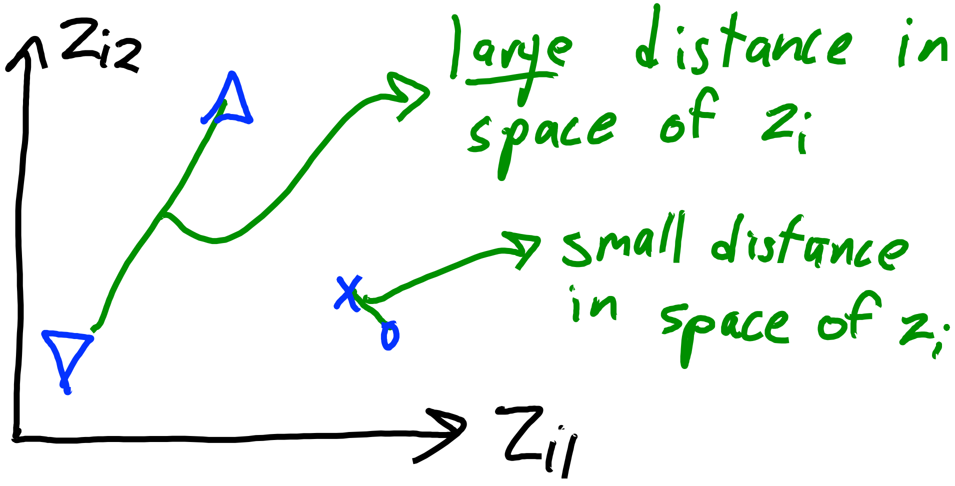
"Try to make scatterplot distances match high-dimensional distance"

# Multi-Dimensional Scaling

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

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



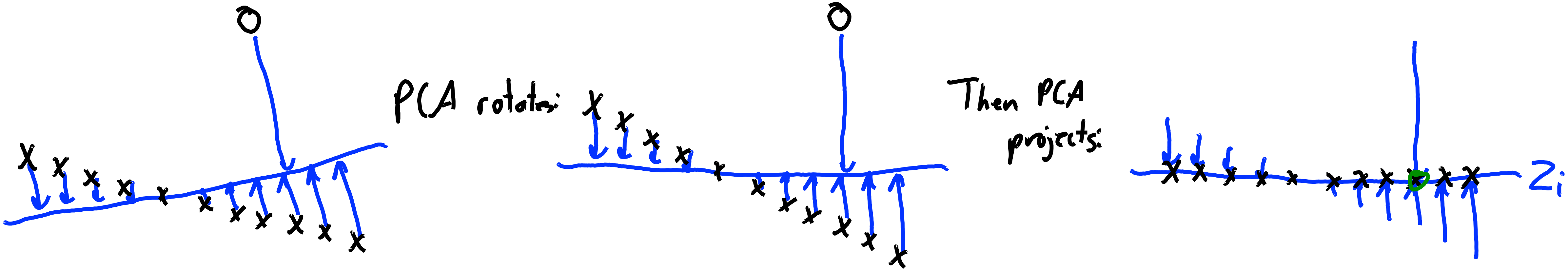



# Multi-Dimensional Scaling

- 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 $x_i$.

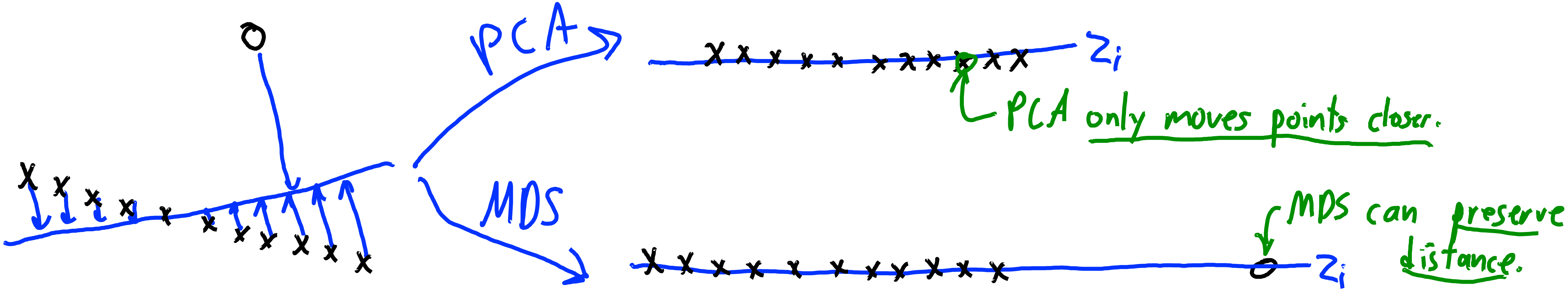


# Multi-Dimensional Scaling

- 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 $x_i$.

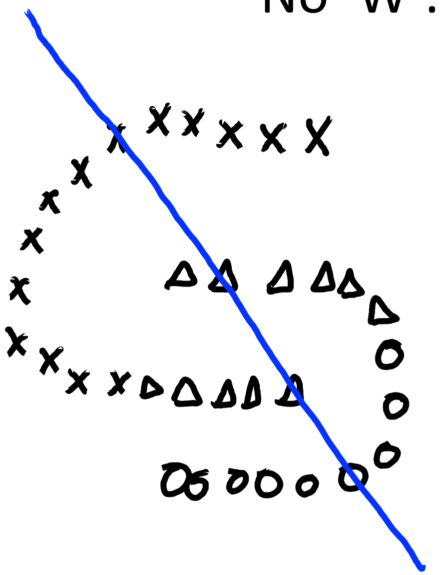


# Multi-Dimensional Scaling

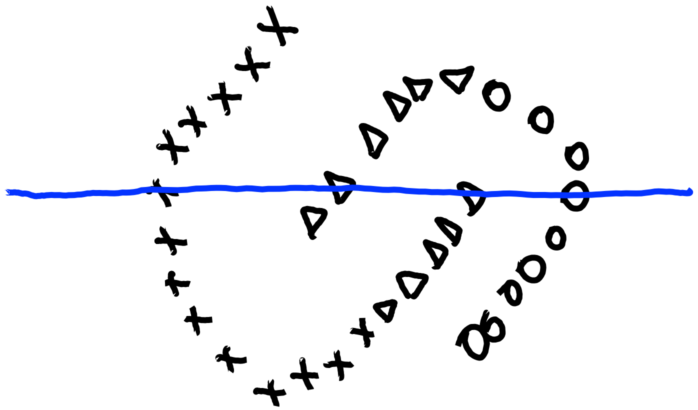
- Multi-dimensional scaling (MDS):
  - Directly optimize the final locations of the $z_i$ values.

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

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



PCA rotation:



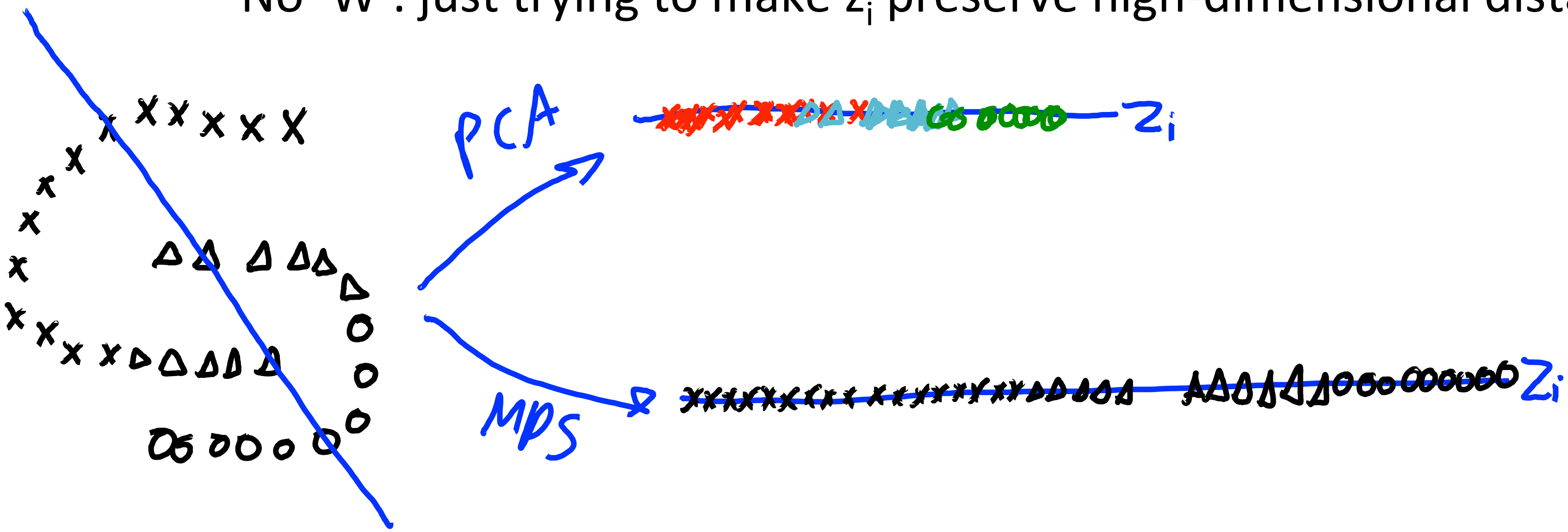
PCA projection:

# 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.

# 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:

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

- PCA is a special case of MDS
  - using $d_1(x_i,x_j) = x_i^T x_j$ and $d_2(z_i,z_j) = z_i^T z_j$ and centered $x_i$).

# 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.

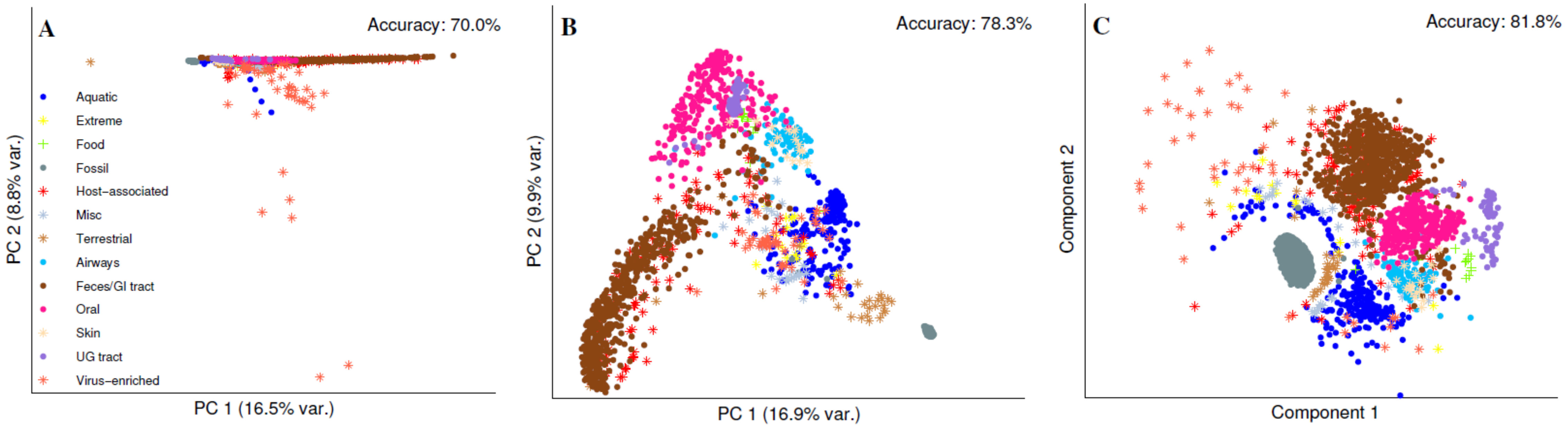
$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left( \frac{d_2(z_i, z_j) - d_1(x_i, x_j)}{d_1(x_i, x_j)} \right)^2$$

  - Denominator reduces focus on large distances.

# Sammon’s Mapping

- Visualizing “metagenomes”

PCA MDS MDS + Sammon



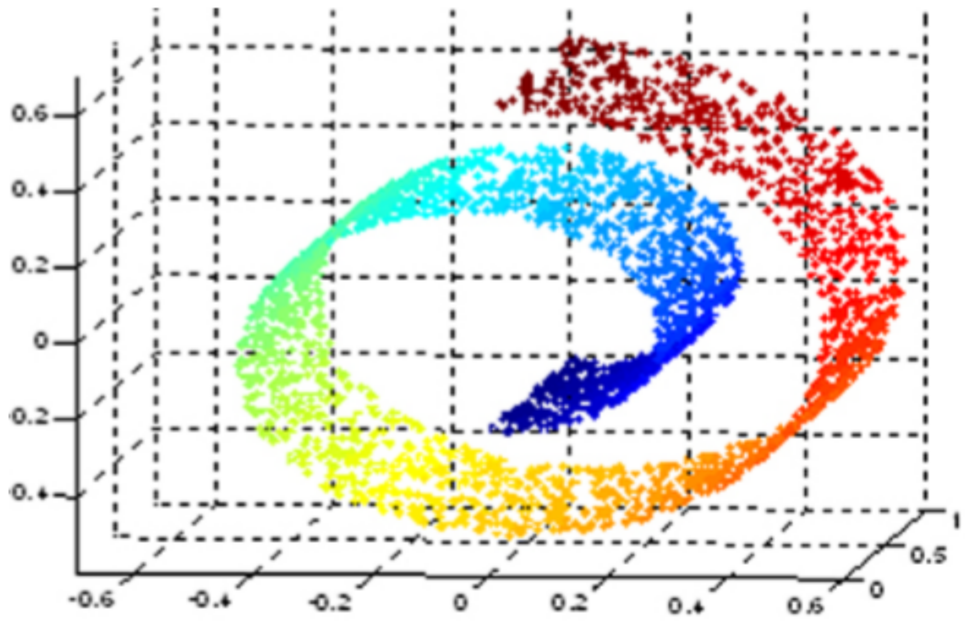

http://www.mdpi.com/1422-0067/15/7/12364/htm

(pause)

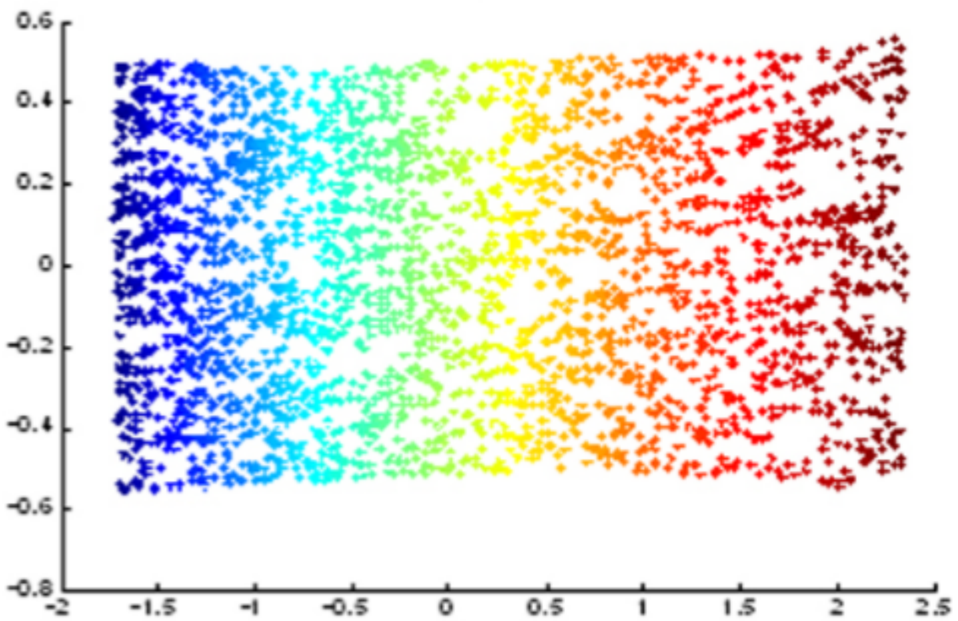
# Learning Manifolds

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

Original data

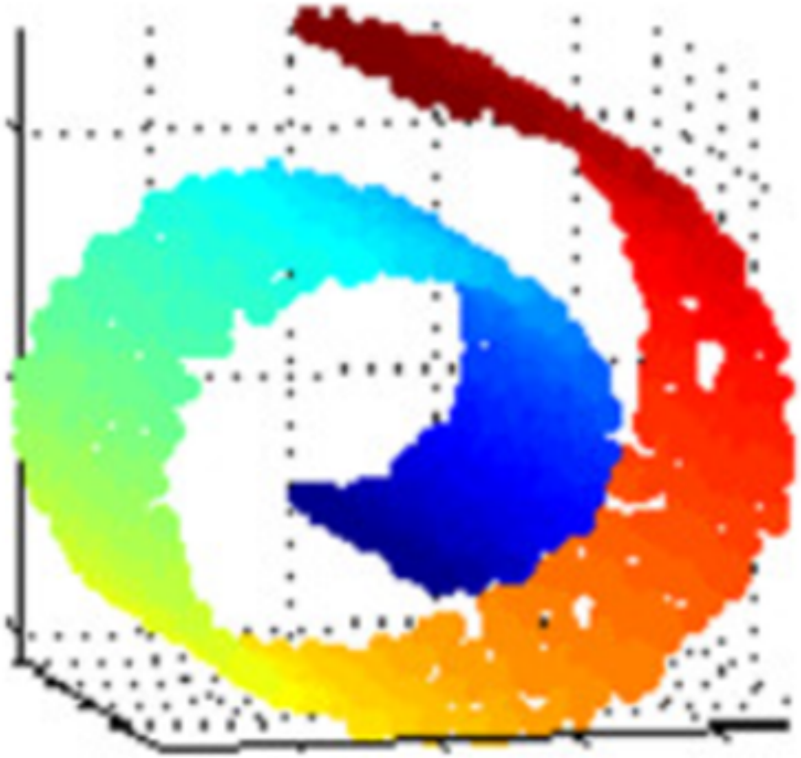



Two-dimensional manifold

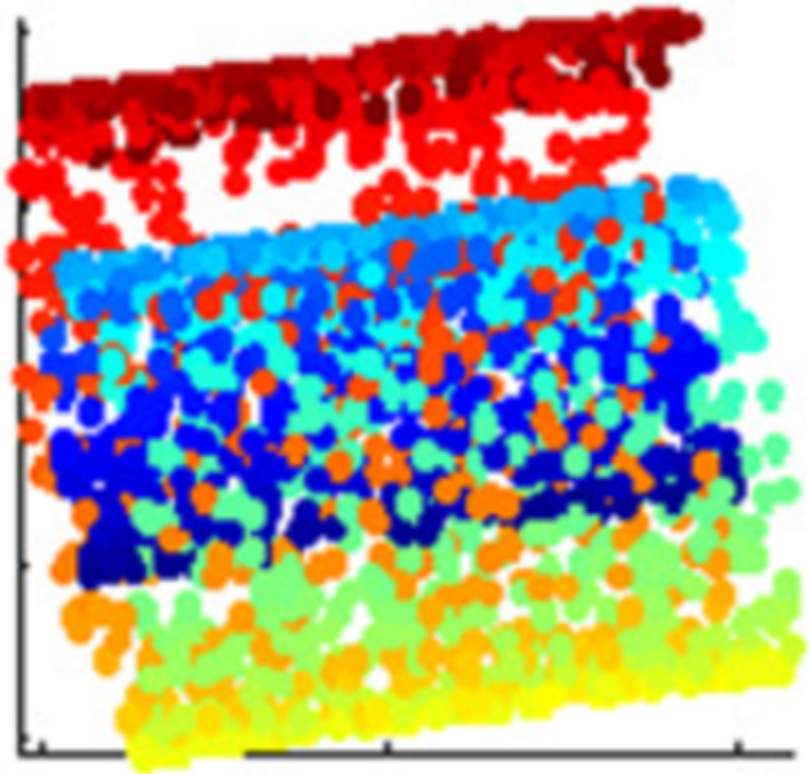


# Learning Manifolds

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



Original data

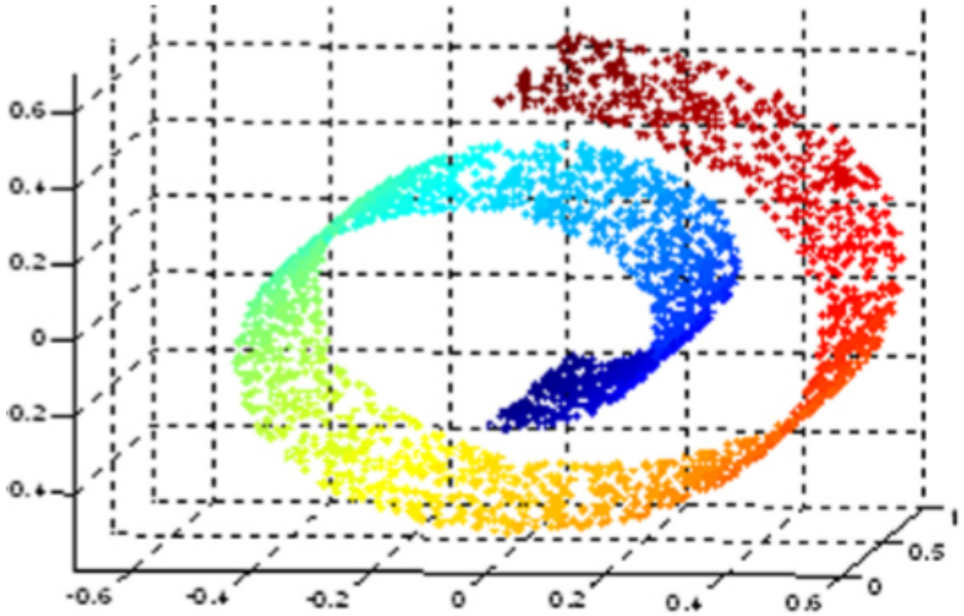


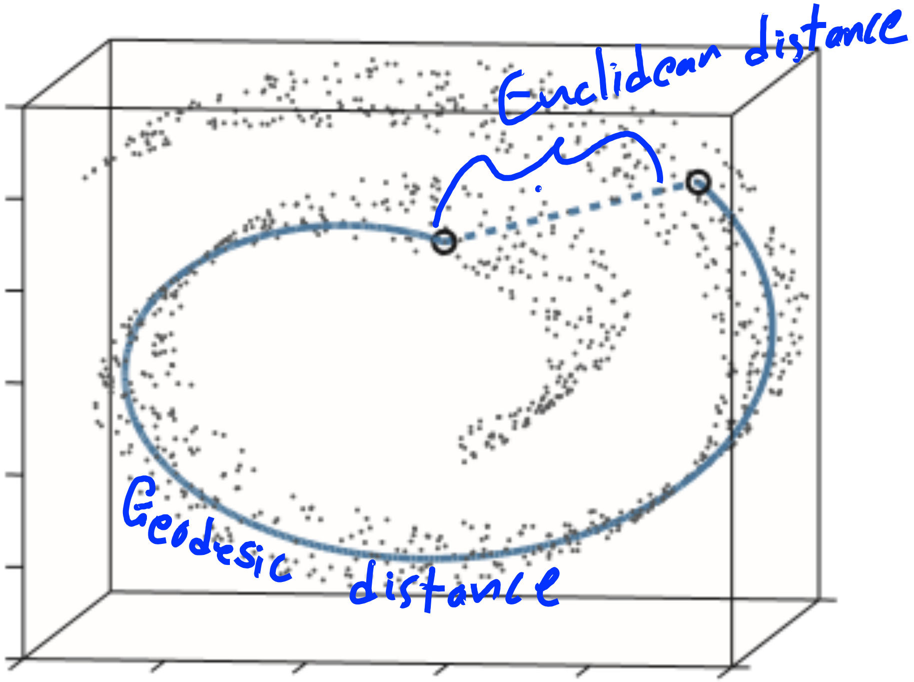
PCA

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.

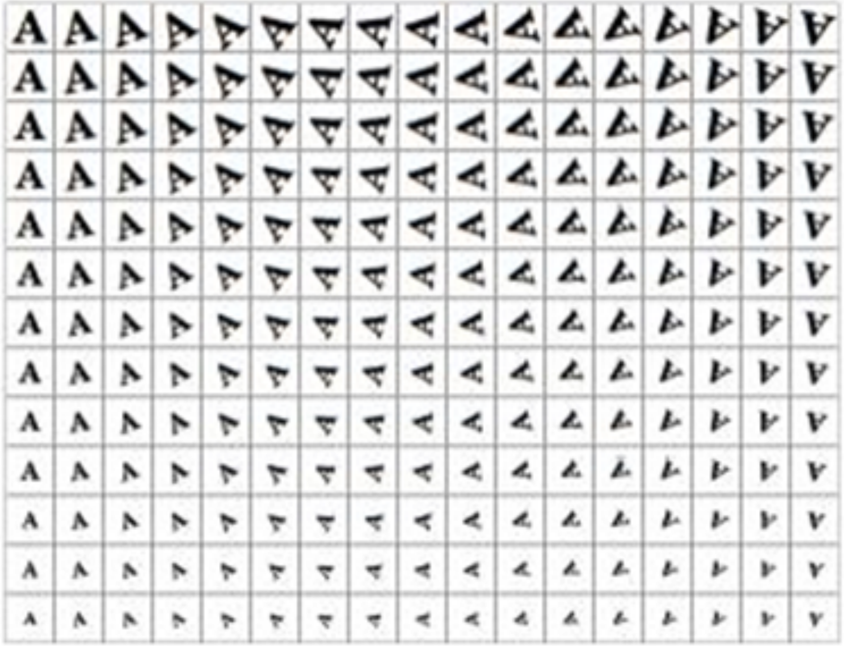





http://www.biomedcentral.com/content/pdf/1471-2105-13-S7-S3.pdf
http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

# Manifolds in Image Space

- Consider slowly-varying transformation of image:



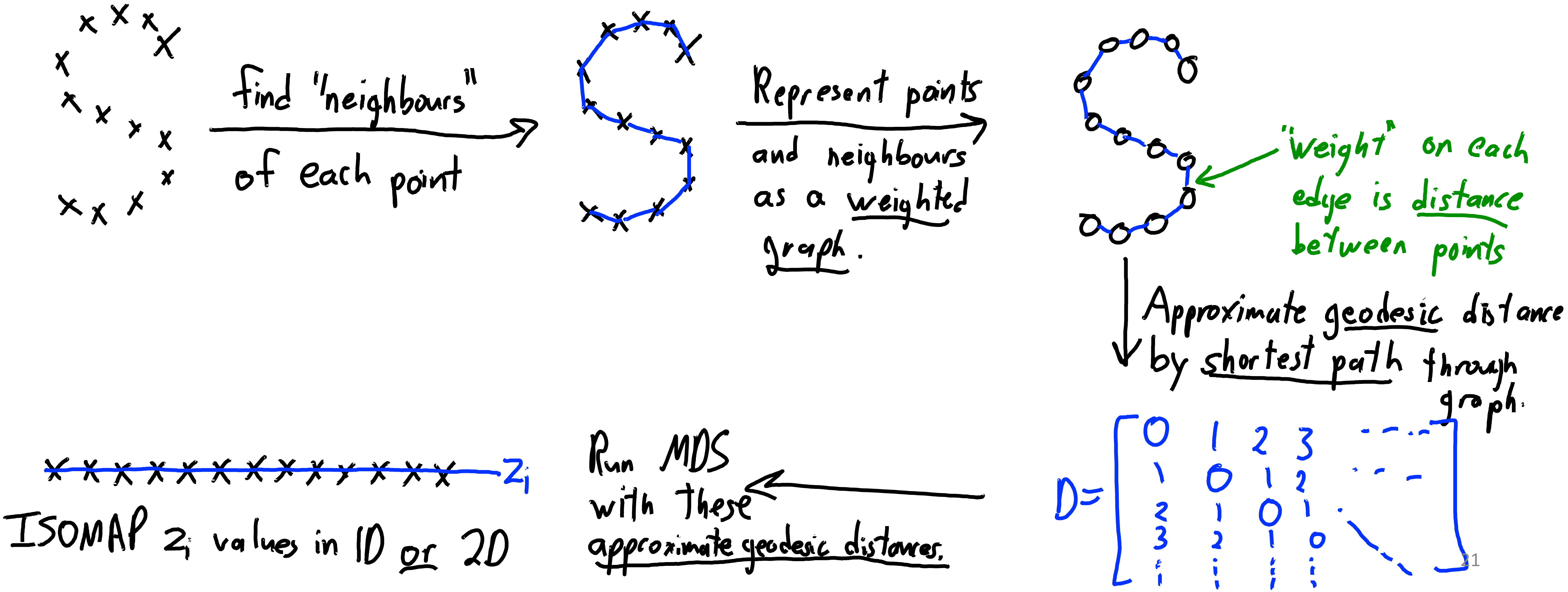
- Images are on a manifold in the high-dimensional space.
  - Euclidean distance doesn't reflect manifold structure.
  - Geodesic distance is distance through space of rotations/resizings.



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

# ISOMAP

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



# 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 $\varepsilon$.
    - 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 $x_j$.

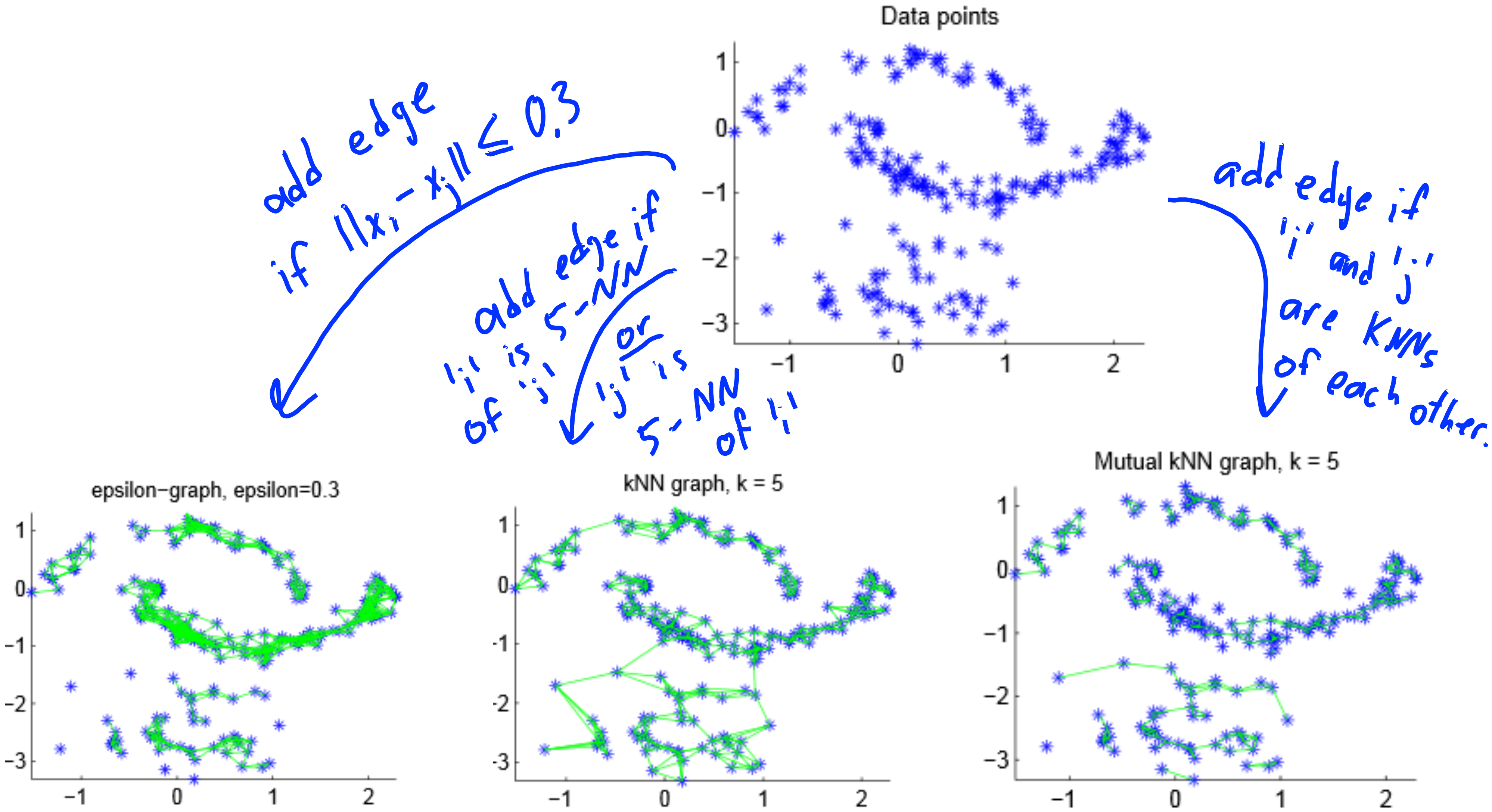
http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf

# Converting from Features to Graph

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

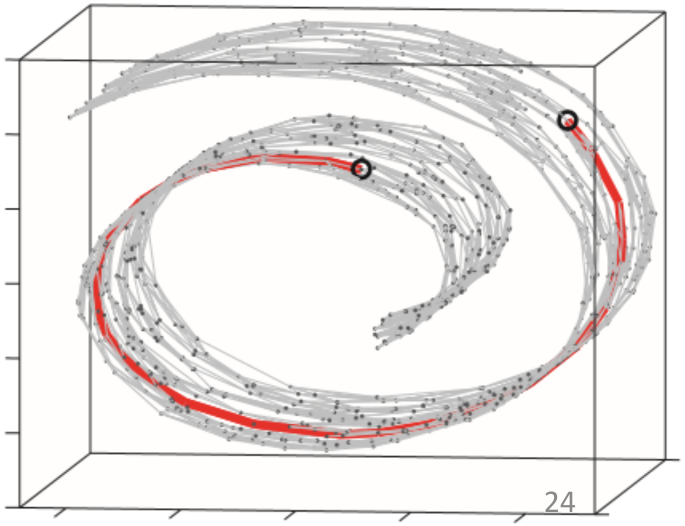
add edge if 'i' is 5-NN of 'j' or 'j' is 5-NN of 'i'

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



# 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.

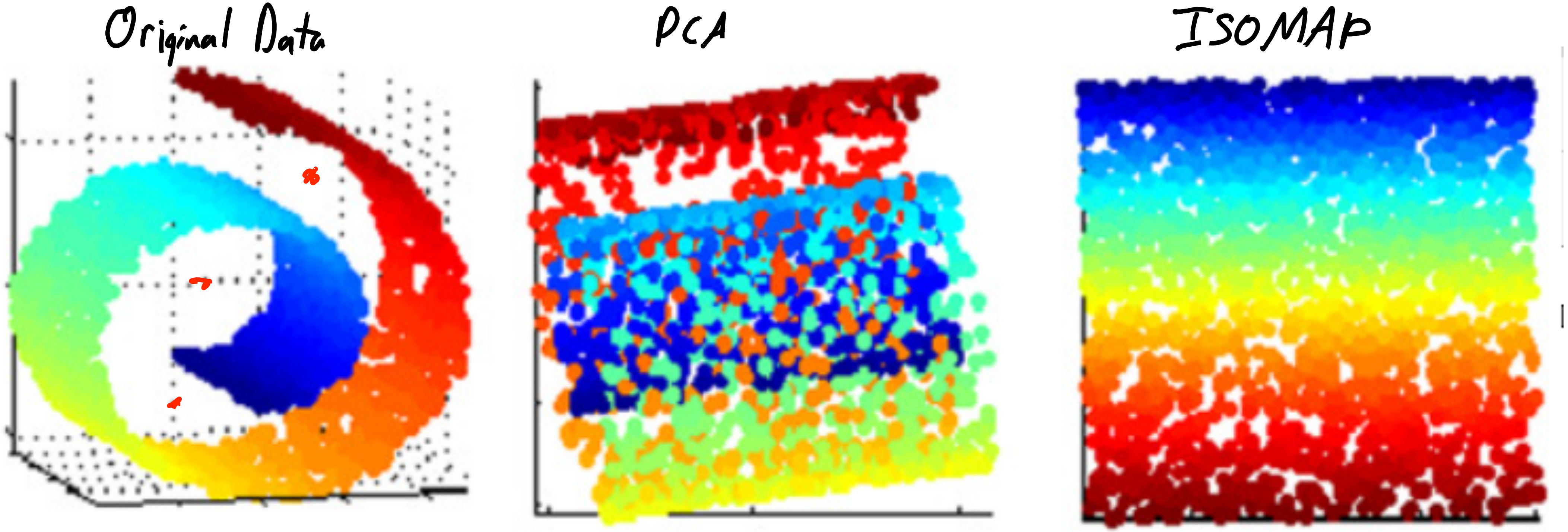




http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

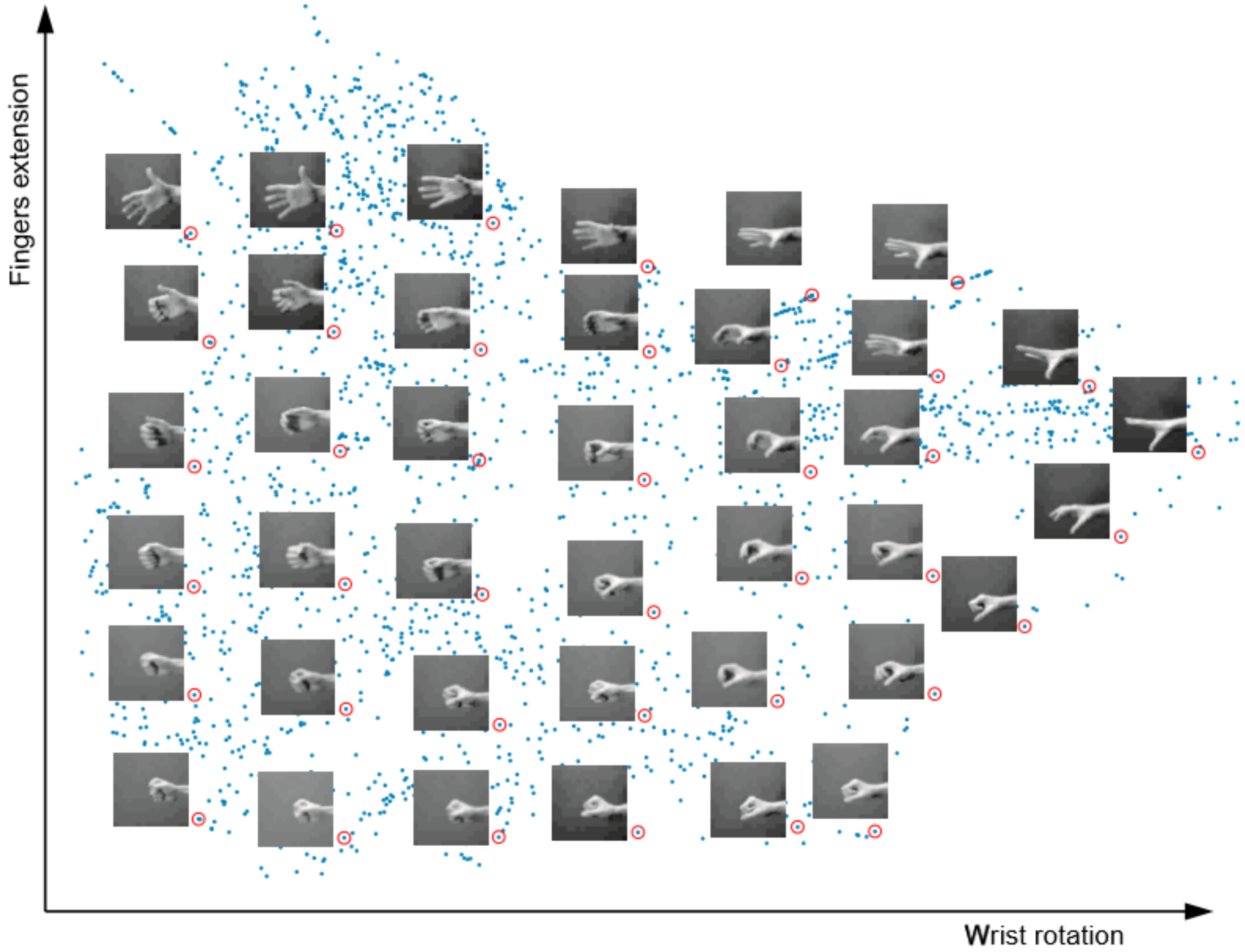
# ISOMAP

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



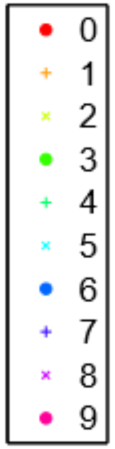

- Sensitive to having the right graph:
  - Points off of manifold and gaps in manifold cause problems.

# ISOMAP on Hand Images



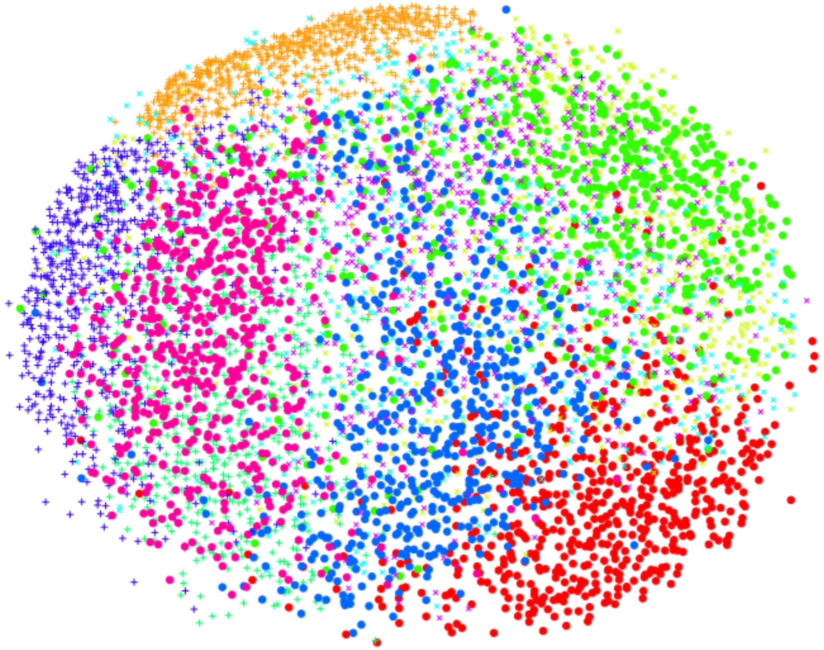



http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

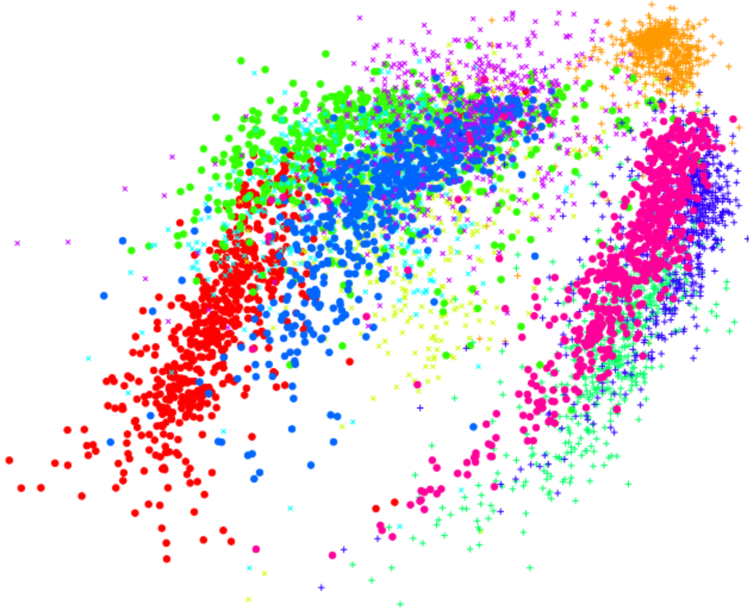



# MNIST digits: Sammon’s Map vs. ISOMAP vs. PCA

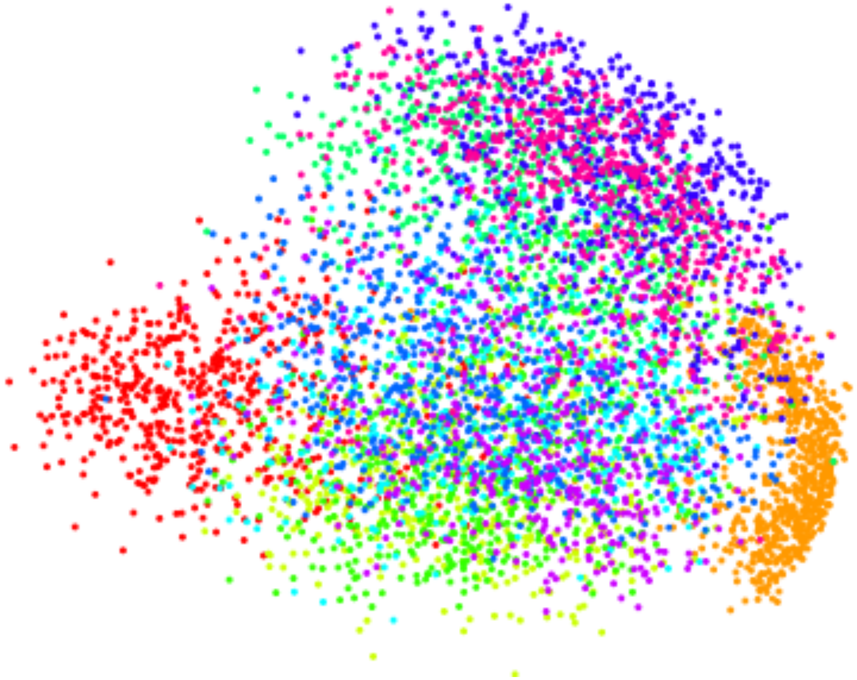
Sammon Map



ISOMAP

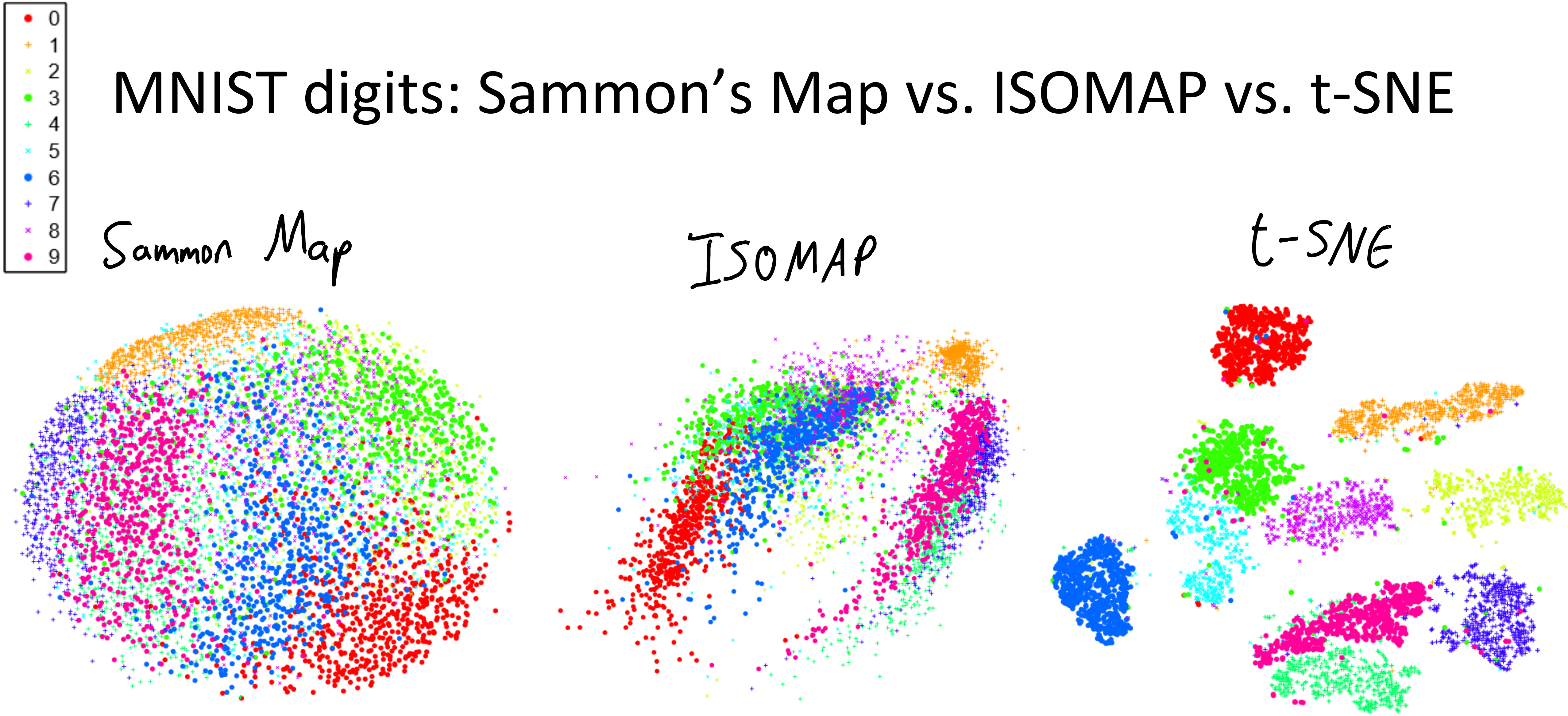


PCA

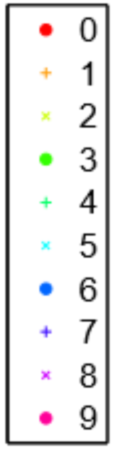


http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

# MNIST digits: Sammon's Map vs. ISOMAP vs. t-SNE




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



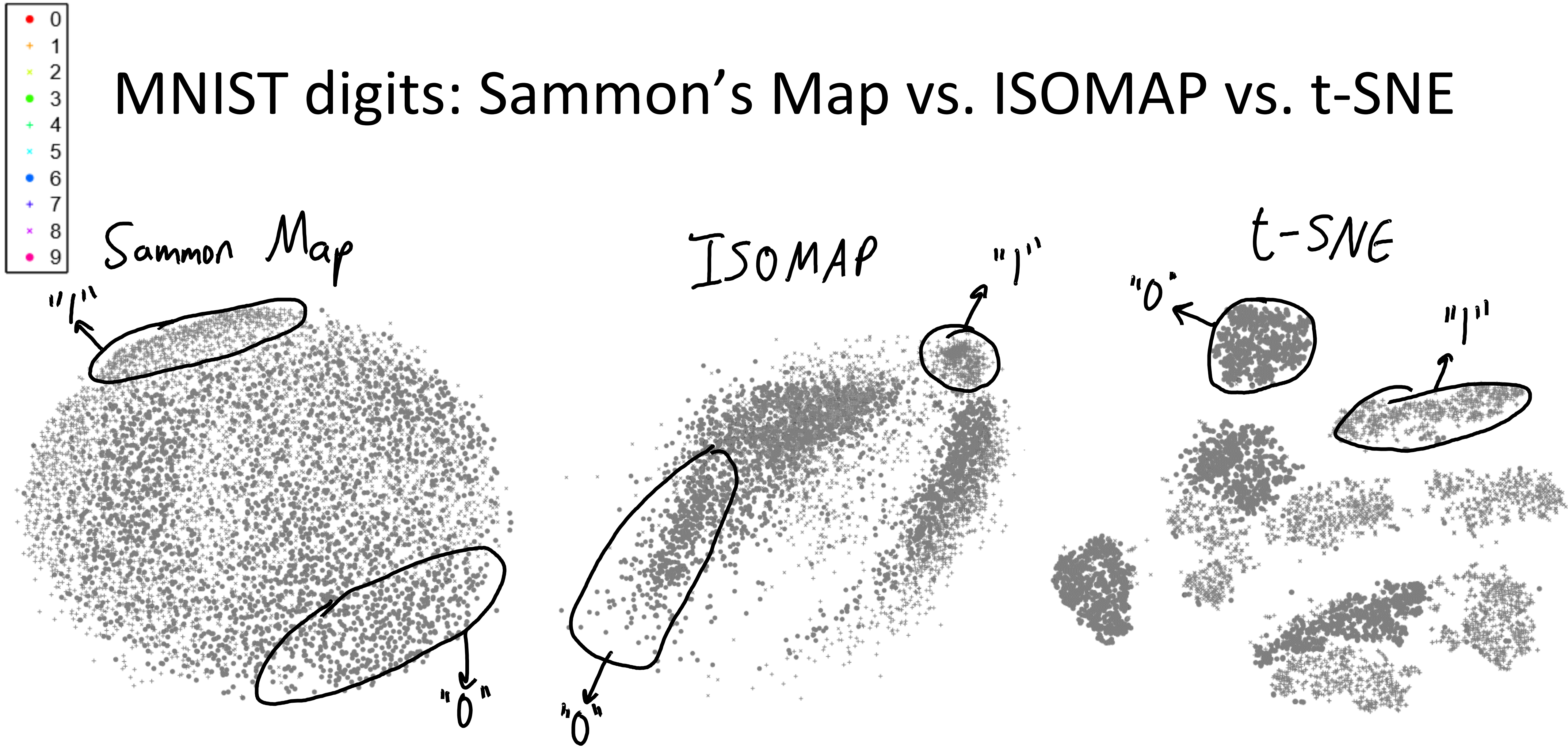

# MNIST digits: Sammon's Map vs. ISOMAP vs. t-SNE

Sammon Map          ISOMAP          t-SNE

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

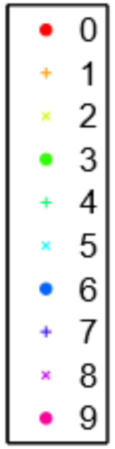
http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

# MNIST digits: Sammon's Map vs. ISOMAP vs. t-SNE

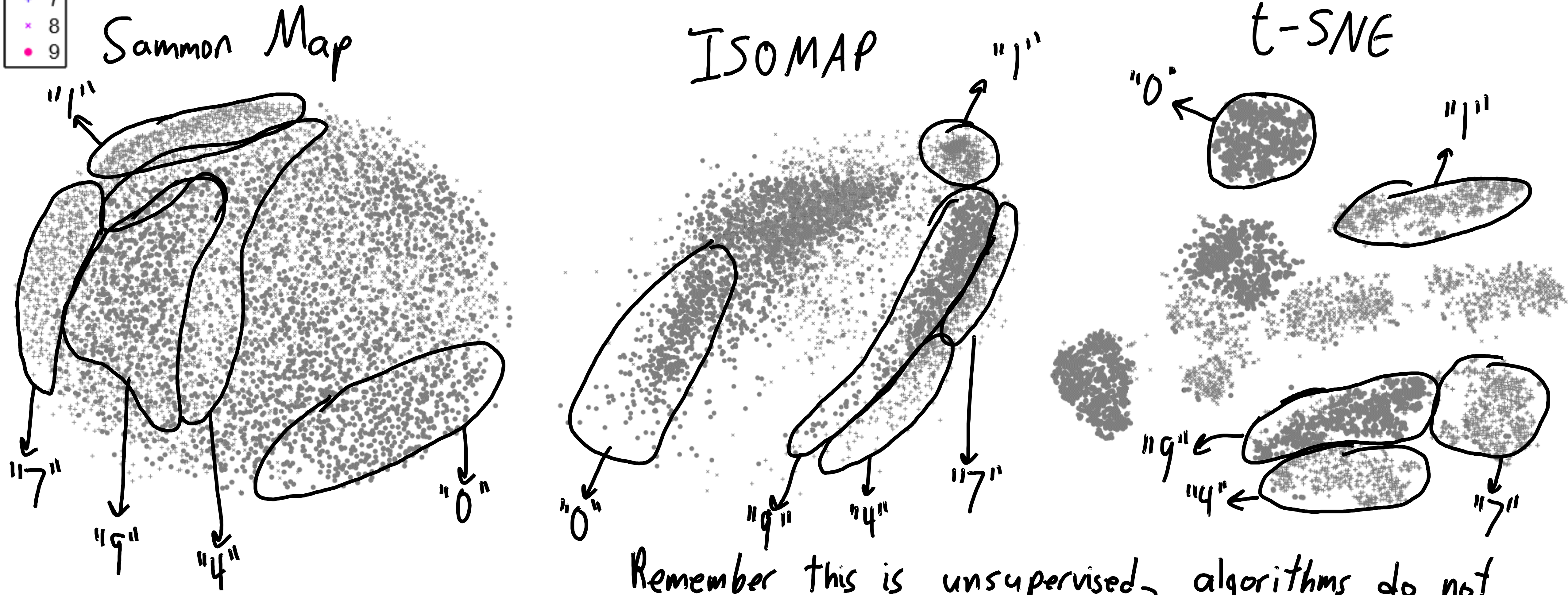



Remember this is unsupervised, algorithms do not know the labels.

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

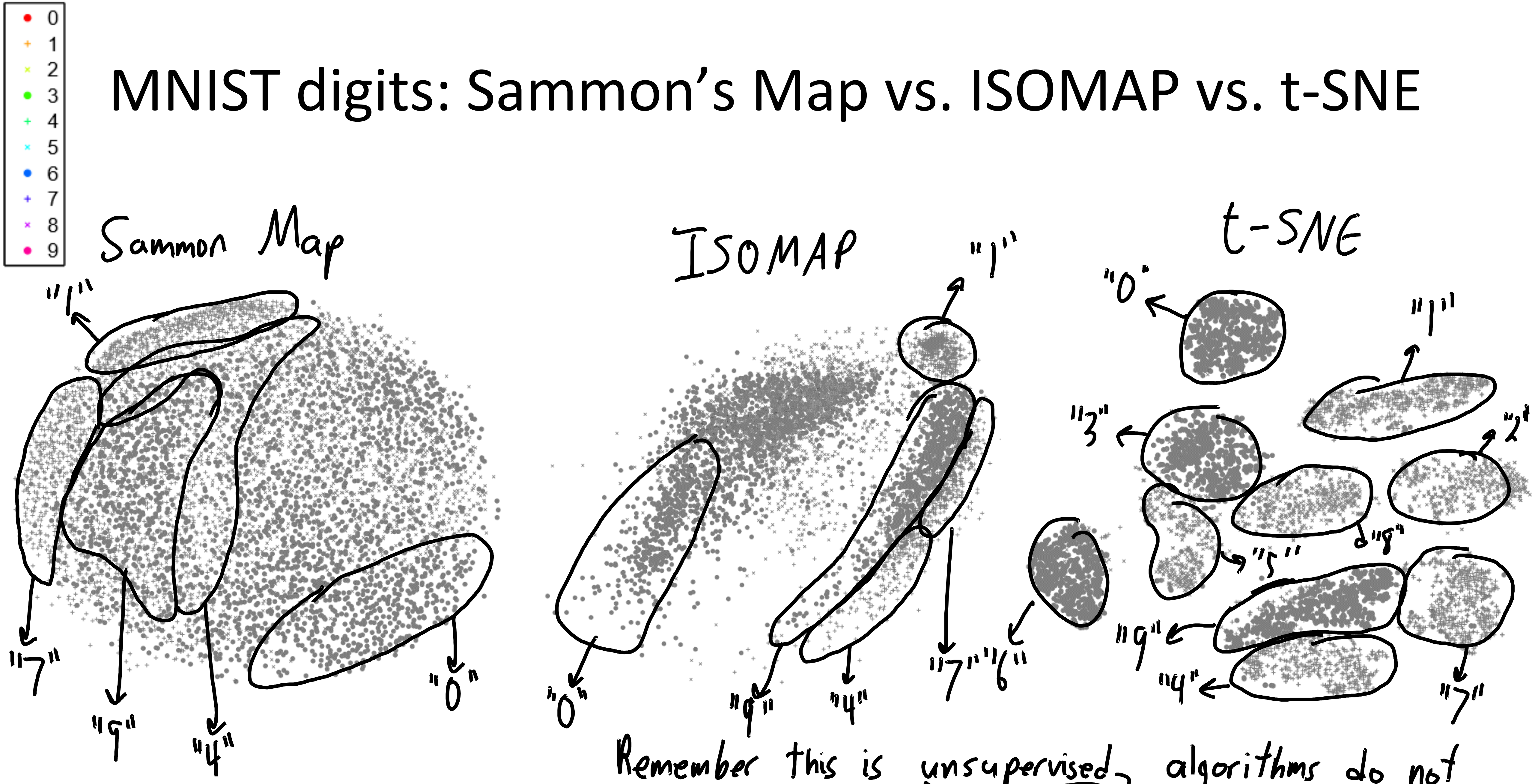



# MNIST digits: Sammon's Map vs. ISOMAP vs. t-SNE




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

# MNIST digits: Sammon's Map vs. ISOMAP vs. t-SNE

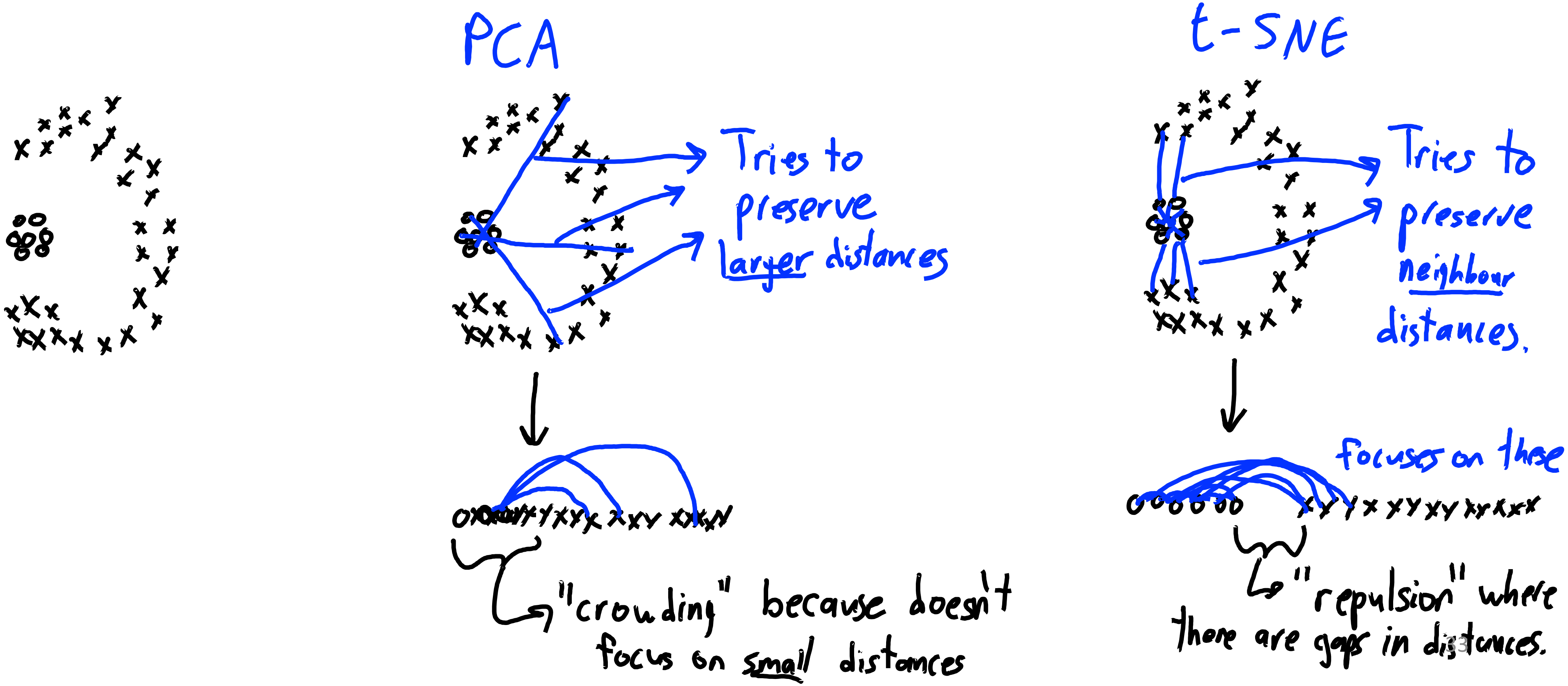



Remember this is unsupervised, algorithms do not know the labels.

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

# t-Distributed Stochastic Neighbour Embedding

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



# End of Part 4: Key Concepts

- We discussed linear latent-factor models:

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

$$= \sum_{i=1}^{n} \|W^T z_i - x_i\|^2$$

$$= \|ZW - X\|_F^2$$

- Represent 'X' as linear combination of latent factors '$w_c$'.
  - Latent features '$z_i$' give a lower-dimensional version of each '$x_i$'.
  - When k=1, finds direction that minimizes squared 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:

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

- Principal component analysis (PCA):
  - Often uses orthogonal factors and fits them sequentially (via SVD).
- 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 neighbour distances.

# Summary

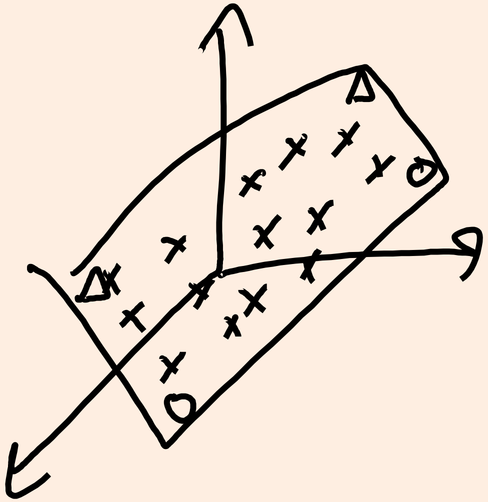
- Multi-dimensional scaling is a non-parametric latent-factor model.
- Different MDS distances/losses/weights usually gives better results.
- Manifold learning focuses on low-dimensional curved structures.
- ISOMAP is most common approach:
  - Approximates geodesic distance by shortest path in weighted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

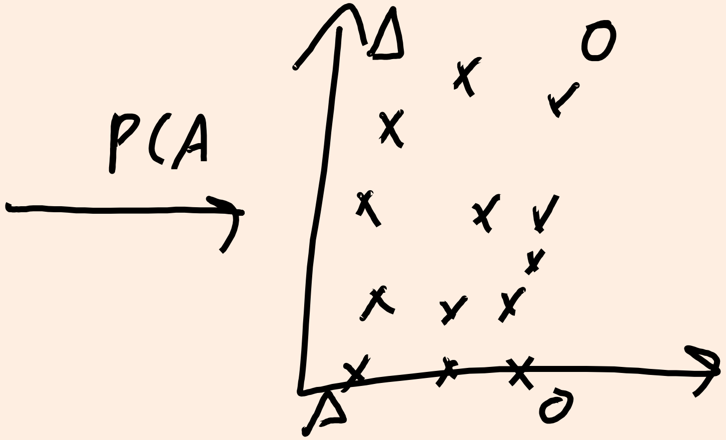
# Related method to ISOMAP

- “local linear embedding”.

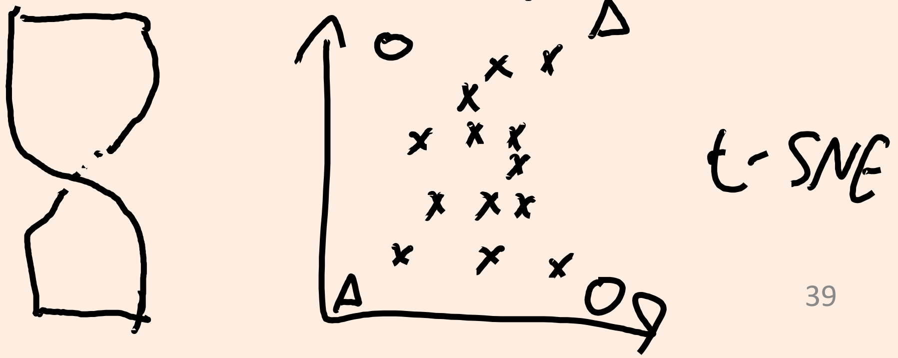
# Does t-SNE always outperform PCA?

- 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.



# Latent-Factor Representation of Words

- For natural language, we often represent words by an index.
  - E.g., “cat” is word 124056.
- 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…

# Word2Vec

- Two variations on objective in word2vec:
  - Try to predict word from surrounding words (continuous bag of words).
  - Try to predict surrounding words from word (skip-gram).

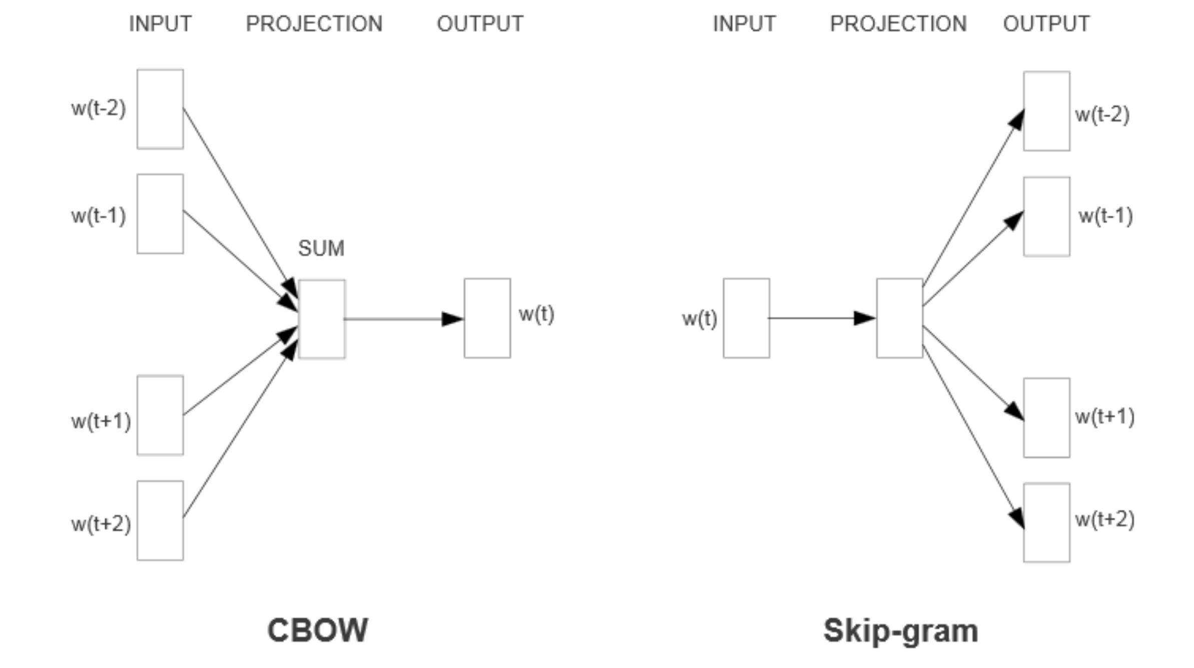



Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

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, we optimize the likelihood:

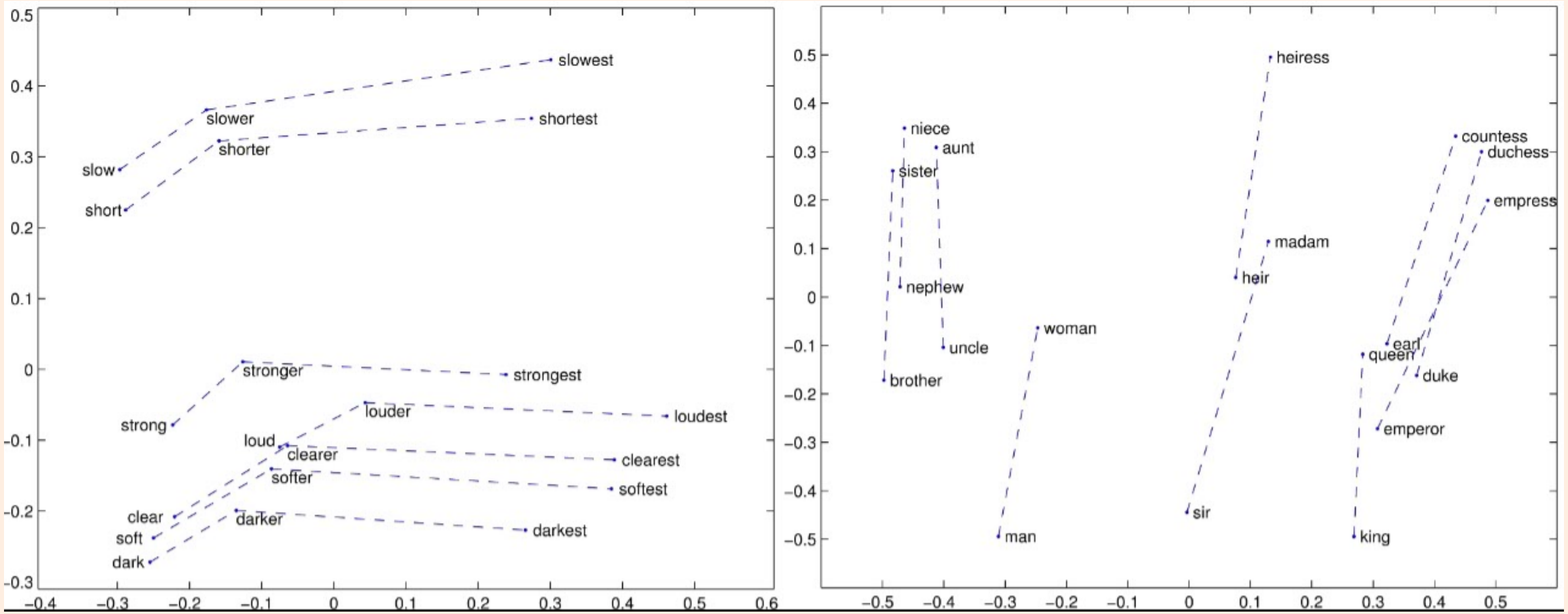
$$p(x_i \mid x_{surround}) = \prod_{j \in surround} p(x_i \mid x_j) \quad \text{(independence assumption)}$$

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

- Denominator sums over all words.
- For skip-gram it will be over all possible surrounding words.
  - Common trick to speed things up: samples terms in denominator.
    - "Negative sampling".

# Word2Vec Example

- MDS visualization of a set of related words:




- Distances between vectors might represent semantics.



http://ruder.io/secret-word2vec/

# Word2Vec

- Subtracting word vectors to find related vectors.

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

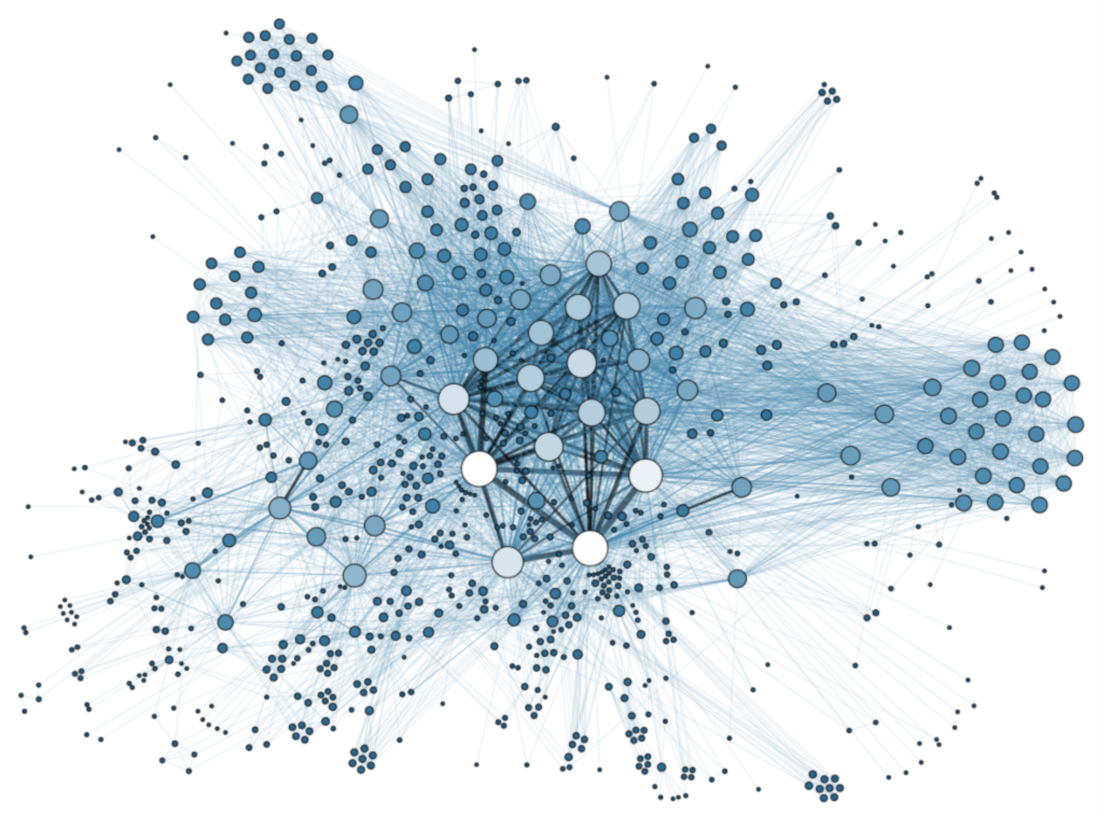
| Relationship | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| France - Paris | Italy: Rome | Japan: Tokyo | Florida: Tallahassee |
| big - bigger | small: larger | cold: colder | quick: quicker |
| Miami - Florida | Baltimore: Maryland | Dallas: Texas | Kona: Hawaii |
| Einstein - scientist | Messi: midfielder | Mozart: violinist | Picasso: painter |
| Sarkozy - France | Berlusconi: Italy | Merkel: Germany | Koizumi: Japan |
| copper - Cu | zinc: Zn | gold: Au | uranium: plutonium |
| Berlusconi - Silvio | Sarkozy: Nicolas | Putin: Medvedev | Obama: Barack |
| Microsoft - Windows | Google: Android | IBM: Linux | Apple: iPhone |
| Microsoft - Ballmer | Google: Yahoo | IBM: McNealy | Apple: Jobs |
| Japan - sushi | Germany: bratwurst | France: tapas | USA: pizza |

Table 8 shows words that follow various relationships. We follow the approach described above: the relationship is defined by subtracting two word vectors, and the result is added to another word. Thus for example, *Paris - France + Italy = Rome*. As it can be seen, accuracy is quite good, although

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

# 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:

$$\nabla[\tfrac{1}{2}(w^Tx_i - y_i)^2]$$

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

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

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

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

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

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

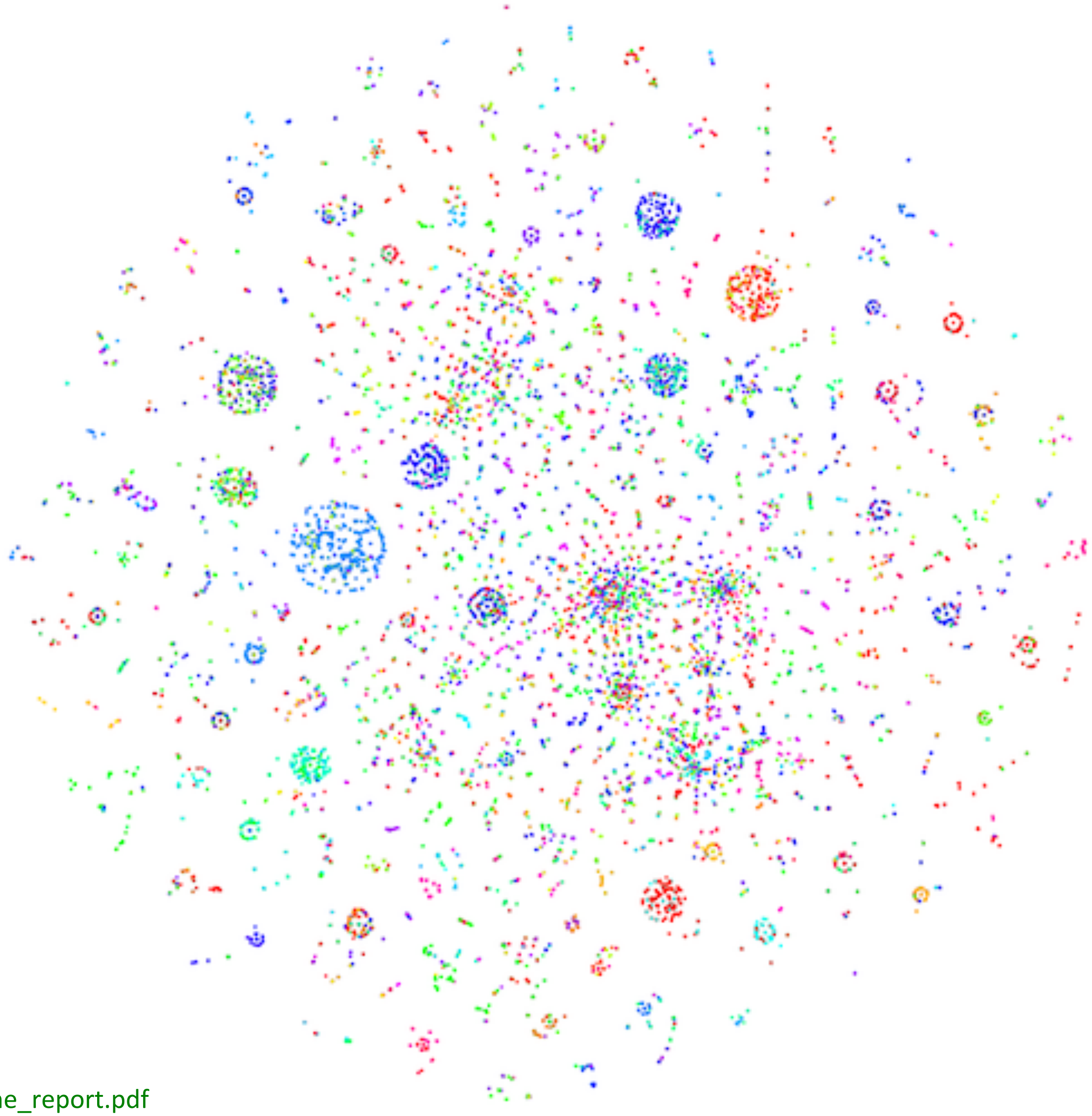
$$\nabla_{z_i} g(d_1(x_i, x_j), d_2(z_i, z_j)) = \underbrace{-(d_1(x_i, x_j) - d_2(z_i, z_j))}_{\substack{g'(d_1, d_2) \\ \text{(move distances closer)}}} \underbrace{\left[ - \frac{(z_i - z_j)}{2\|z_i - z_j\|} \right]}_{\substack{\nabla_{z_i} d_2(z_i, z_j) \\ \text{(how distance changes in } z\text{-space)}}}$$

$\hookrightarrow$ Assuming $z_i \neq z_j$

# t-Distributed Stochastic Neighbour Embedding

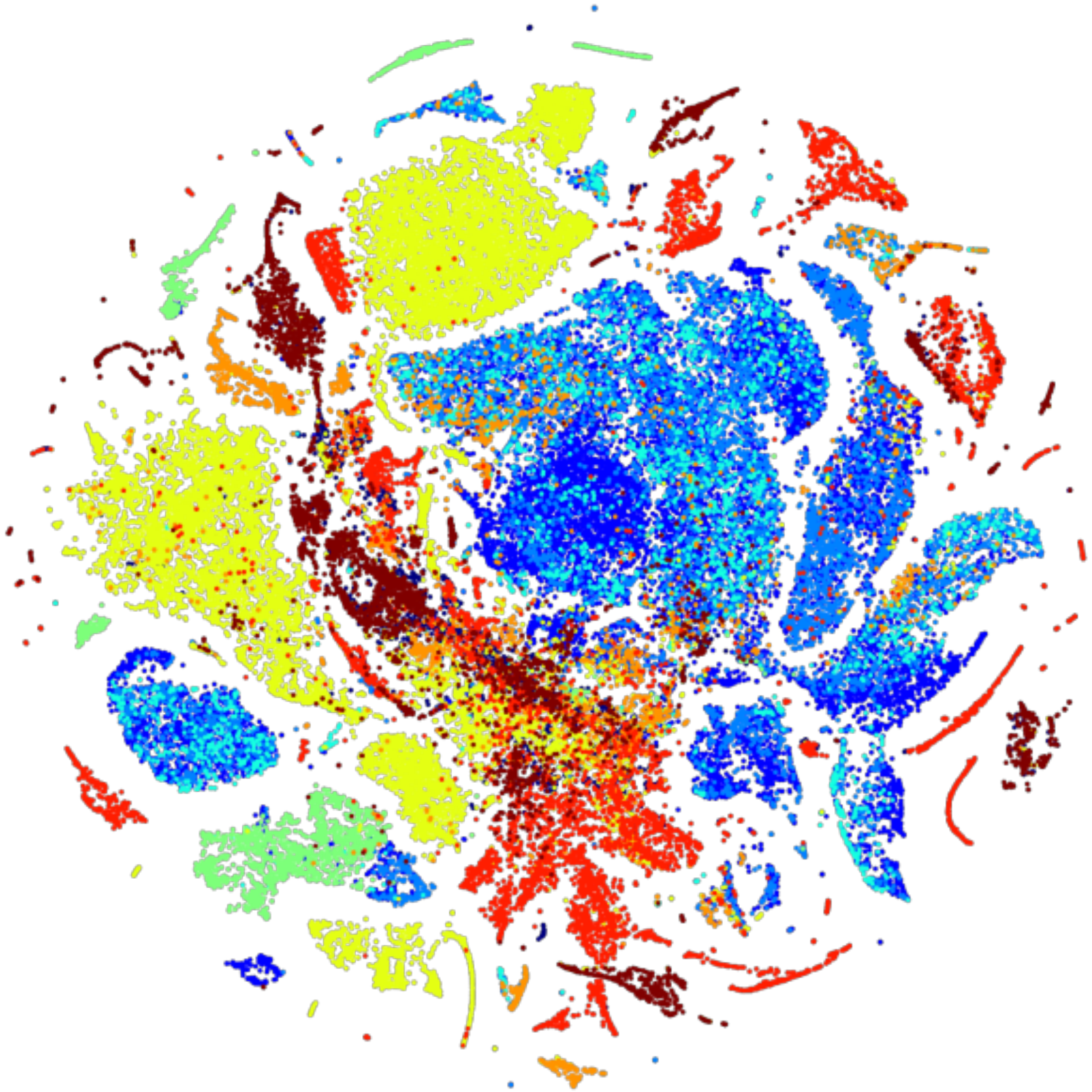
- t-SNE is a special case of MDS (specific $d_1$, $d_2$, and $d_3$ choices):
  - $d_1$: for each $x_i$, compute probability that each $x_j$ is a 'neighbour'.
    - Computation is similar to k-means++, but most weight to close points (Gaussian).
    - Doesn't require explicit graph.
  - $d_2$: for each $z_i$, 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



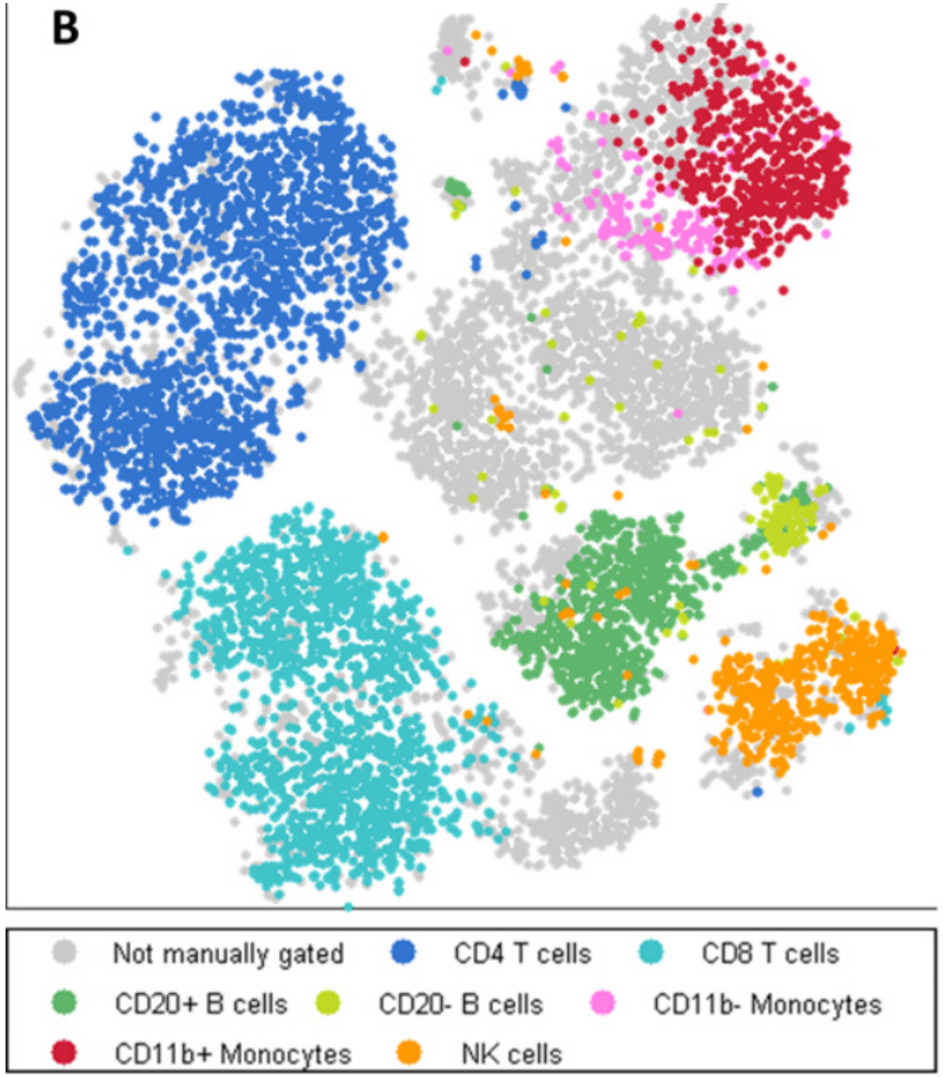
http://jasneetsabharwal.com/assets/files/wiki_tsne_report.pdf

# 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/