CPSC 340: Machine Learning and Data Mining

Multi-Dimensional Scaling

Fall 2022

Last Time: Collaborative Filtering with Latent Factors

- We discussed recommender systems using collaborative filtering:
  - Methods that only looks at ratings, not features of movies/users.

$$Y = \begin{bmatrix} ? & 4 & 3 & 2 & 3 & 3 \\ 2 & 1 & ? & 5 & ? & 5 \\ ? & 1 & ? & 5 & 5 & 5 \\ 2 & 3 & 3 & ? & ? & ? \end{bmatrix}$$

- We discussed collaborative filtering with matrix factorization:

$$Y \approx ZW \qquad y_{ij} \approx \langle w^j, z_i \rangle$$

  - Fit to minimize regularized squared error on available ratings (with biases).
    - The learned w^j and z_i can be used to predict unknown y_{ij} values.
  - Can be viewed as “PCA on the available entries”.

Beyond Accuracy in Recommender Systems

- Winning system of Netflix Challenge was never adopted.
- Other issues important in recommender systems:
  - Diversity: how different are the recommendations?
    - If you like ‘Battle of Five Armies Extended Edition’, recommend Battle of Five Armies?
    - Even if you really really like Star Wars, you might want non-Star-Wars suggestions.
  - Persistence: how long should recommendations last?
    - If you keep not clicking on ‘Hunger Games’, should it remain a recommendation?
  - Trust: tell user *why* you made a recommendation.
    - Quora gives explanations for recommendations.
  - Social recommendation: what did your friends watch?
  - Freshness: people tend to get more excited about *new/surprising* things.
    - Collaborative filtering does not predict well for new users/movies.
      - New movies don’t yet have ratings, and new users haven’t rated anything.

Content-Based vs. Collaborative Filtering

- Consider content-based filtering, our usual supervised learning (Part 3):

$$\hat{y}_{ij} = w^T x_{ij}$$

  - Here x_{ij} is a fixed vector of features for the movie/user.
    - Usual supervised learning setup: ‘y’ would contain all the y_{ij}, X would have x_{ij} as rows.
  - Can predict on new users/movies, but can’t learn about each user/movie.
    - If two users have the same features, then they get the exact same recommendations.
- Our latent-factor approach to collaborative filtering (Part 4):

$$\hat{y}_{ij} = \langle \underbrace{w^j}_{\text{"hidden" features of movie}}, \underbrace{z_i}_{\text{"hidden" features of user}} \rangle$$

  - Learns vector of features z_i for each user ‘i’.
  - But can’t predict on new users (with no ratings).

Hybrid Content/Collaborative: SVDfeature

- SVDfeature combines content-based/collaborative filtering:

$$\hat{y}_{ij} = \underbrace{w^T x_{ij}}_{\text{Linear model based on user/movie features } x_{ij}.} + \langle \underbrace{w^j, z_i}_{\text{Latent features } z_i \text{ for user 'i' and latent features } w_j \text{ for movie 'j'}} \rangle$$

- Learns weights ‘w’ on fixed features x_{ij}.
  - Allows predictions for generic users/movies (including new ones).
- And learns movie-specific weights w^j on learned user-specific features z_i.
  - Allows more-accurate predictions for users/movies with lots of data.
- Typically you also have a global bias β, user-specific bias β_i, and movie-specific β_j.
  - And train with SGD (see bonus slides).
- Won “KDD Cup” competition in 2011 and 2012.

Social Regularization

- Many recommenders are now connected to social networks.
  - "Login using your Facebook account".
- Often, people like similar movies to their friends.
- Recent recommender systems use social regularization.
  - Add a "regularizer" encouraging friends' weights to be similar:

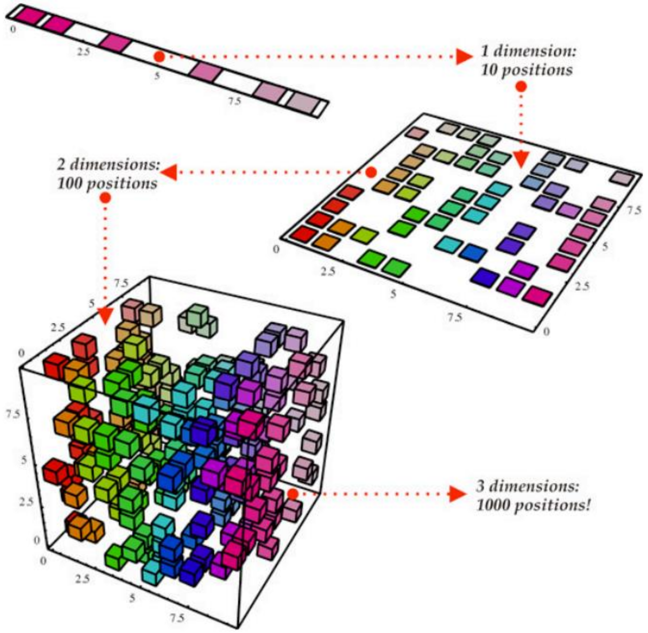
$$\frac{\lambda}{2} \sum_{(i,j) \in \text{"friends"}} \|z_i - z_j\|^2$$

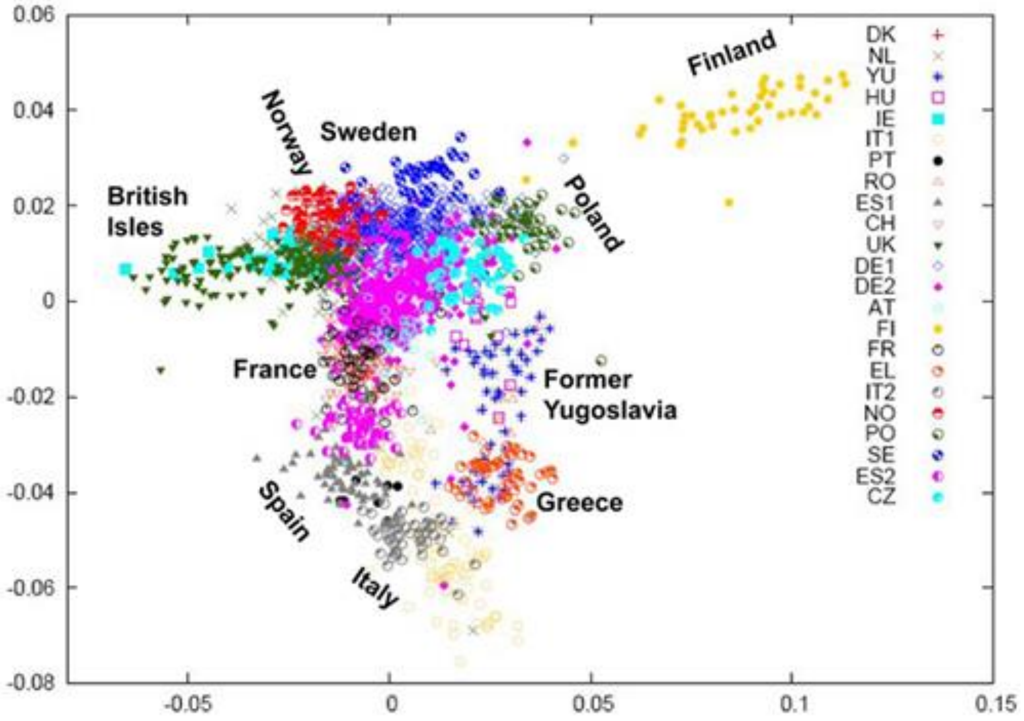
  - If we get a new user, recommendations are based on friend's preferences.

Next Topic: Multi-Dimensional Scaling

Visualization High-Dimensional Data

- PCA for visualizing high-dimensional data:
  - Use PCA ‘W’ matrix to linearly transform data to get the z_i values.
  - And then we plot the z_i values as locations in a scatterplot.





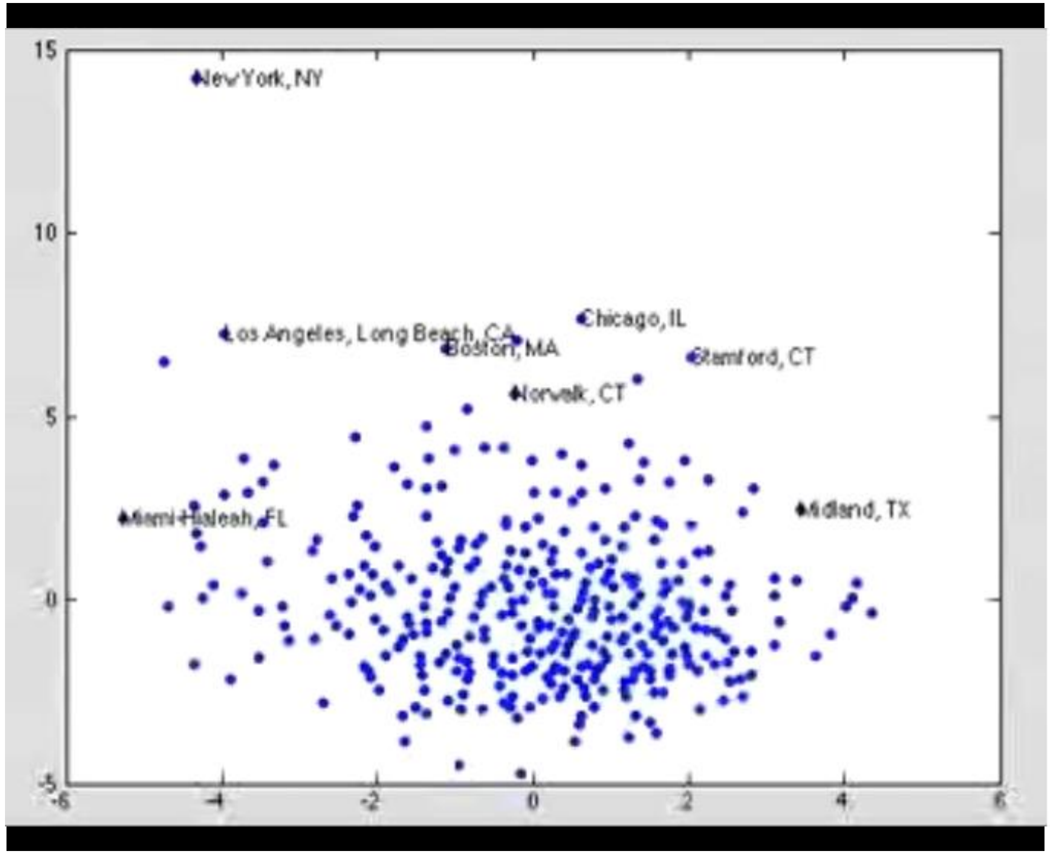
http://www.turingfinance.com/artificial-intelligence-and-statistics-principal-component-analysis-and-self-organizing-maps/
http://scienceblogs.com/gnxp/2008/08/14/the-genetic-map-of-europe/

Visualization High-Dimensional Data

- PCA for visualizing high-dimensional data:
  - Use PCA ‘W’ matrix to linearly transform data to get the z_i values.
  - And then we plot the z_i values as locations in a scatterplot.
- An common alternative is multi-dimensional scaling (MDS):
  - 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:

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

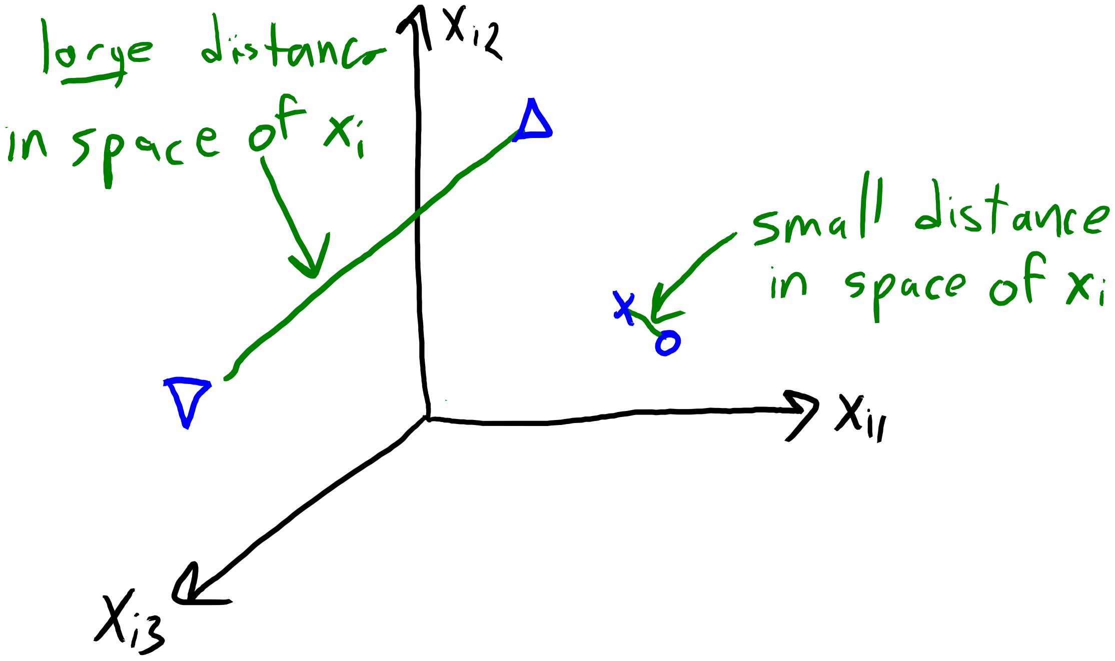
MDS Method (“Sammon Mapping”) Video

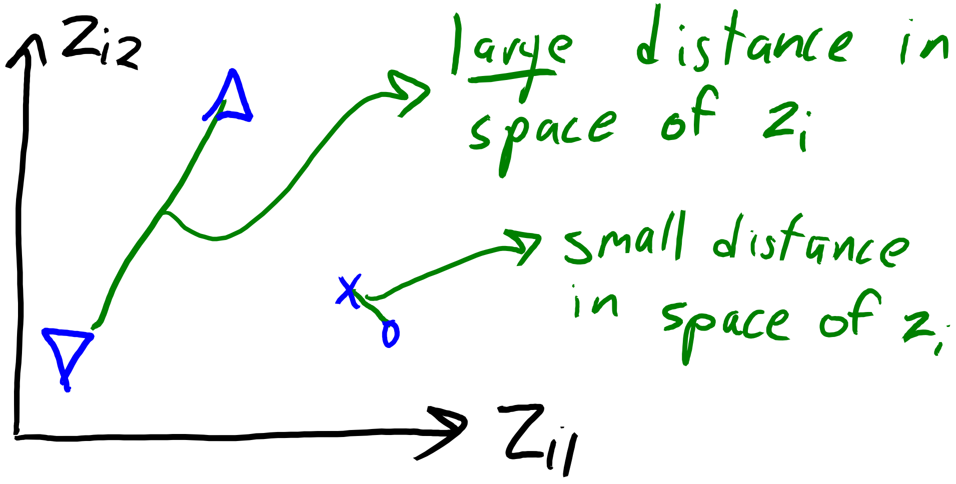


Multi-Dimensional Scaling

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

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


large distance in space of x_i
x_{i2}
small distance in space of x_i
x_{i1}
x_{i3}

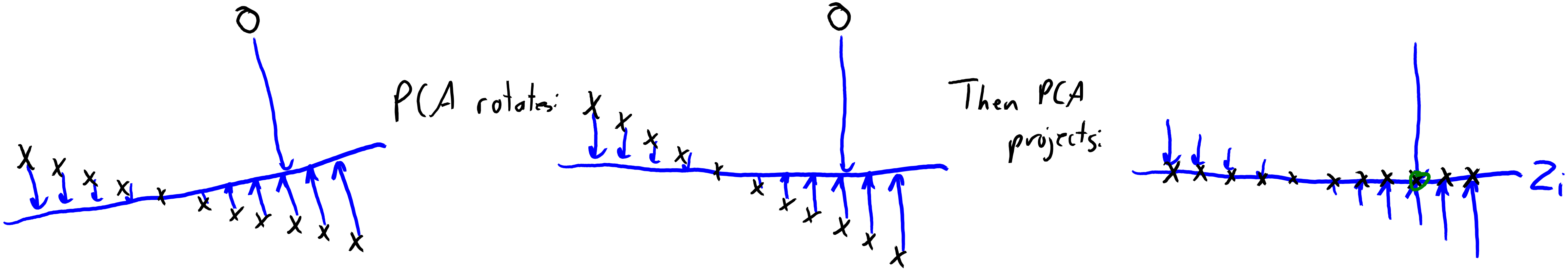

z_{i2}
large distance in space of z_i
small distance in space of z_i
z_{i1}

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.

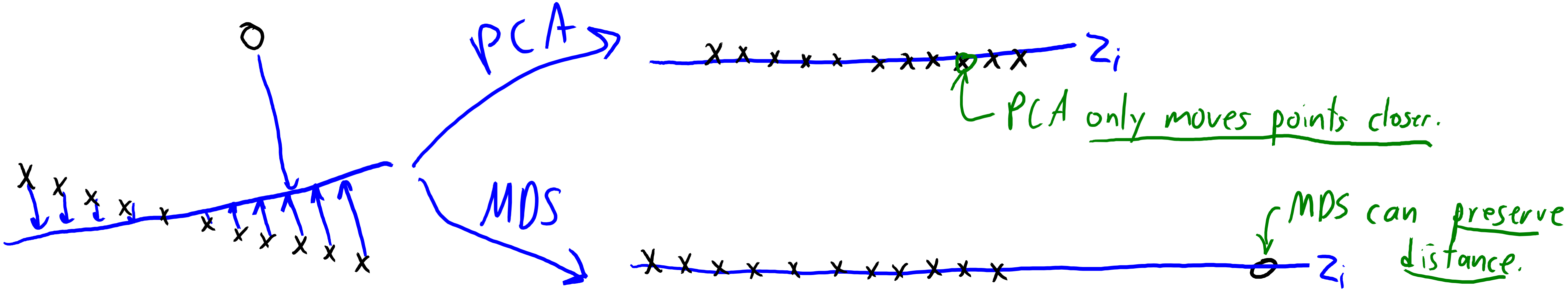


Multi-Dimensional Scaling

- 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.

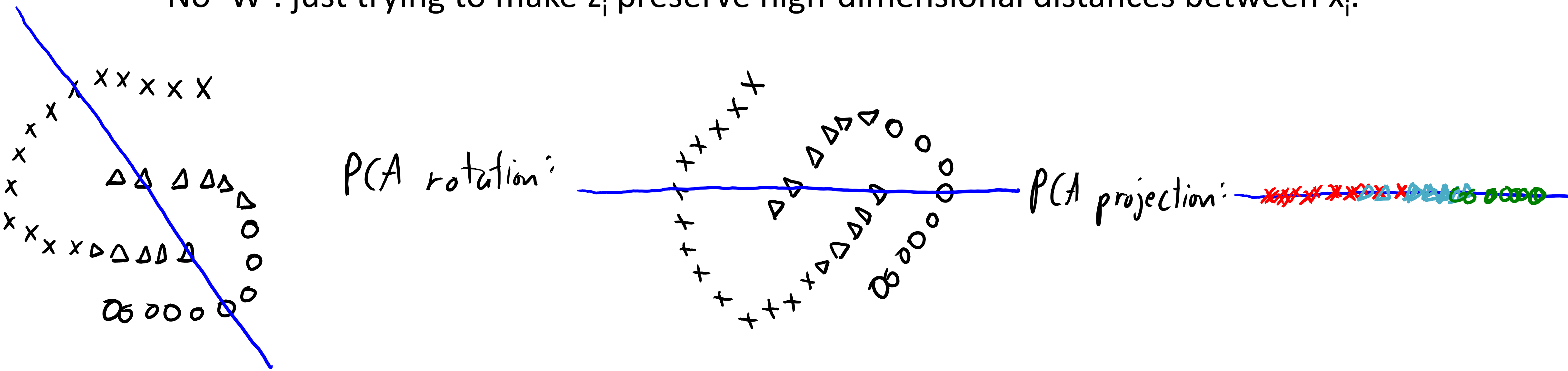


Multi-Dimensional Scaling

- 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.

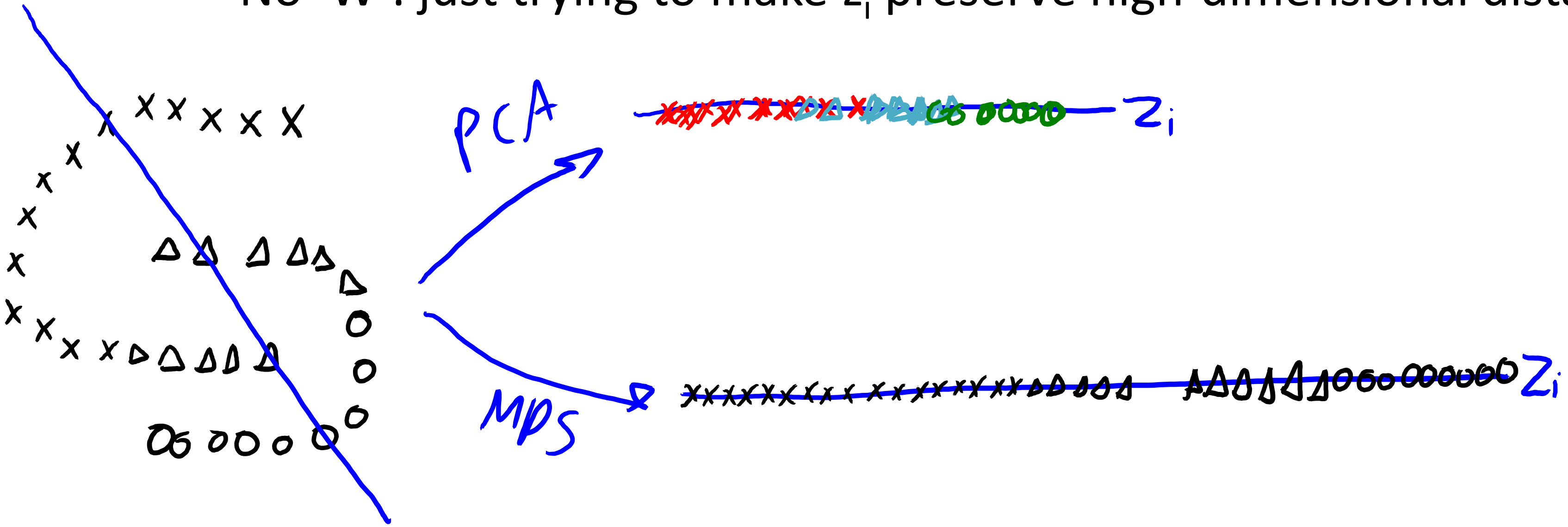


Multi-Dimensional Scaling

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

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

  - 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.

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

- 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

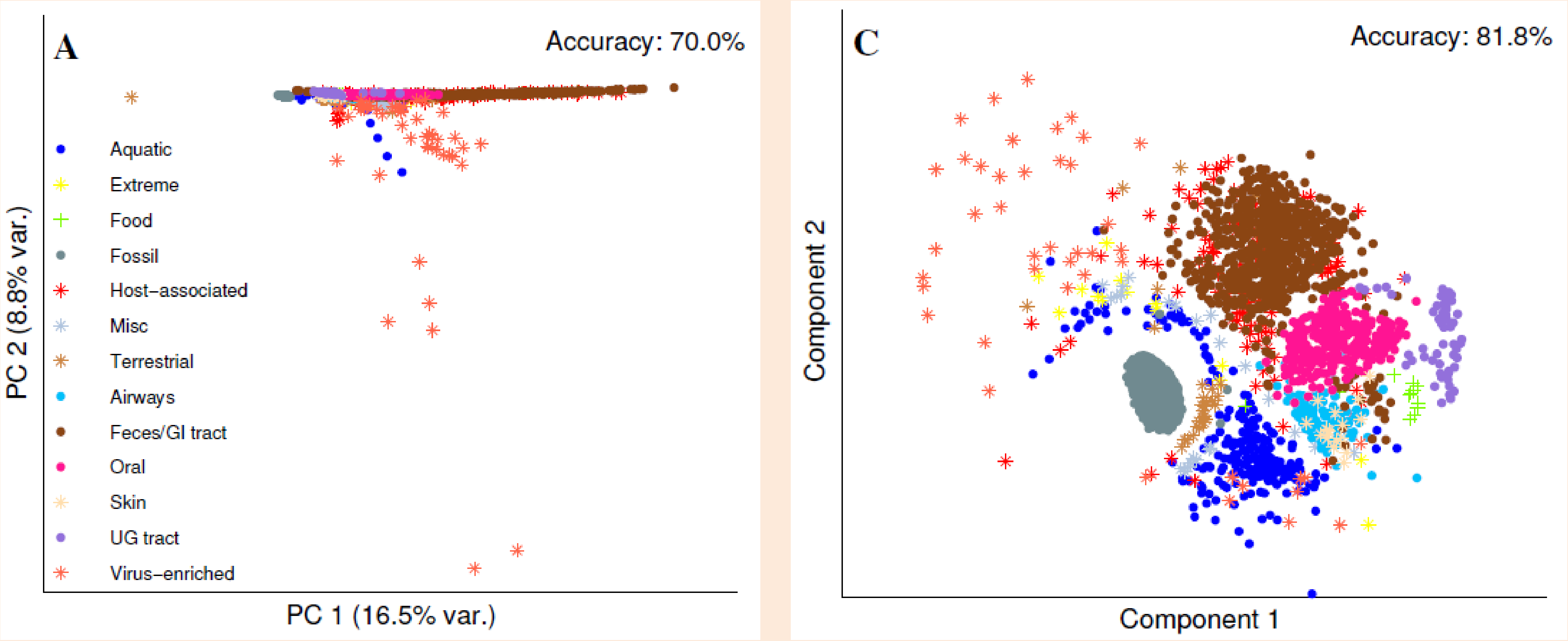
- Unfortunately, MDS often does not work well in practice.
- Problem with traditional MDS methods: focus on large distances.
  - MDS tends to "crowd/squash" all the data points together like PCA.
- But we could consider different distances/similarities:

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

  - 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.
- Early example was Sammon's Mapping (details in bonus).
  - We next discuss t-SNE, a more recent method that tends to work better.

MDS with Squared Distances vs. Sammon’s Map

- MDS based on Eucliean distances (left) vs. Sammon’s Map (right):

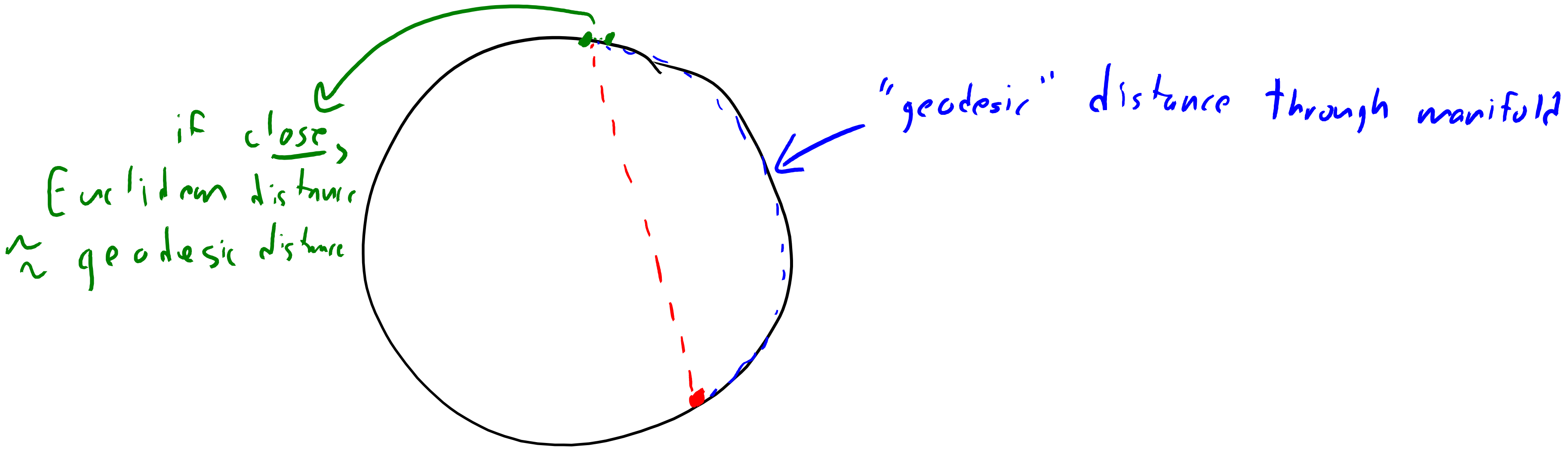


http://www.mdpi.com/1422-0067/15/7/12364/htm

Next Topic: t-SNE

Data on Manifolds

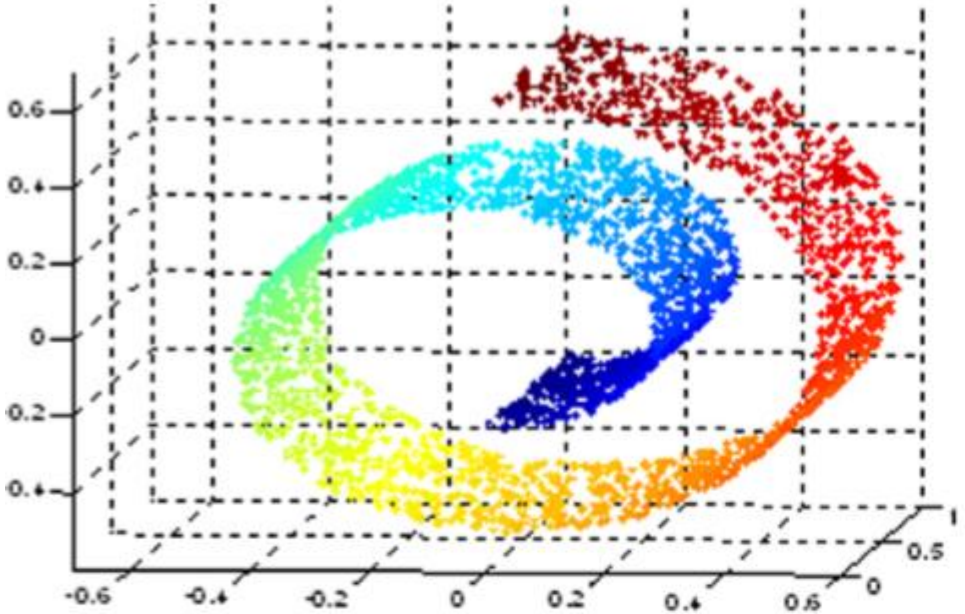
- Consider data that lives on a low-dimensional “manifold”.
  - Where Euclidean distances make sense “locally”.
    - But Euclidean distances may not make sense “globally”.
  - Wikipedia example: Surface of the Earth is “locally” flat.
    - Euclidean distance accurately measures distance “along the surface” locally.
    - For far points Euclidean distance is a poor measure of distance “along the surface”.



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

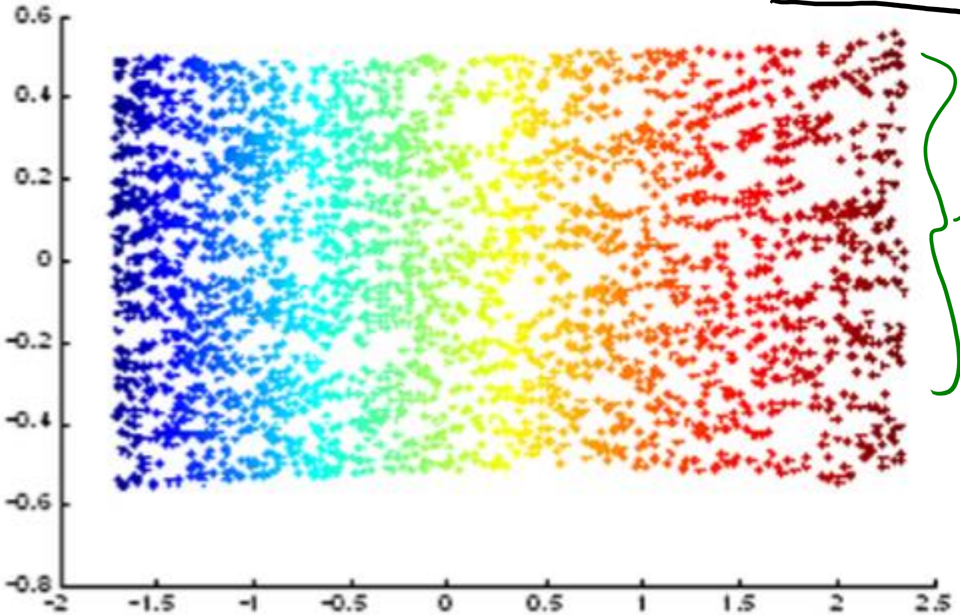
Data on Manifolds

- Consider data that lives on a low-dimensional "manifold".
  - Where Euclidean distances make sense "locally".
    - But Euclidean distances may not make sense "globally".
- Example is the 'Swiss roll':



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

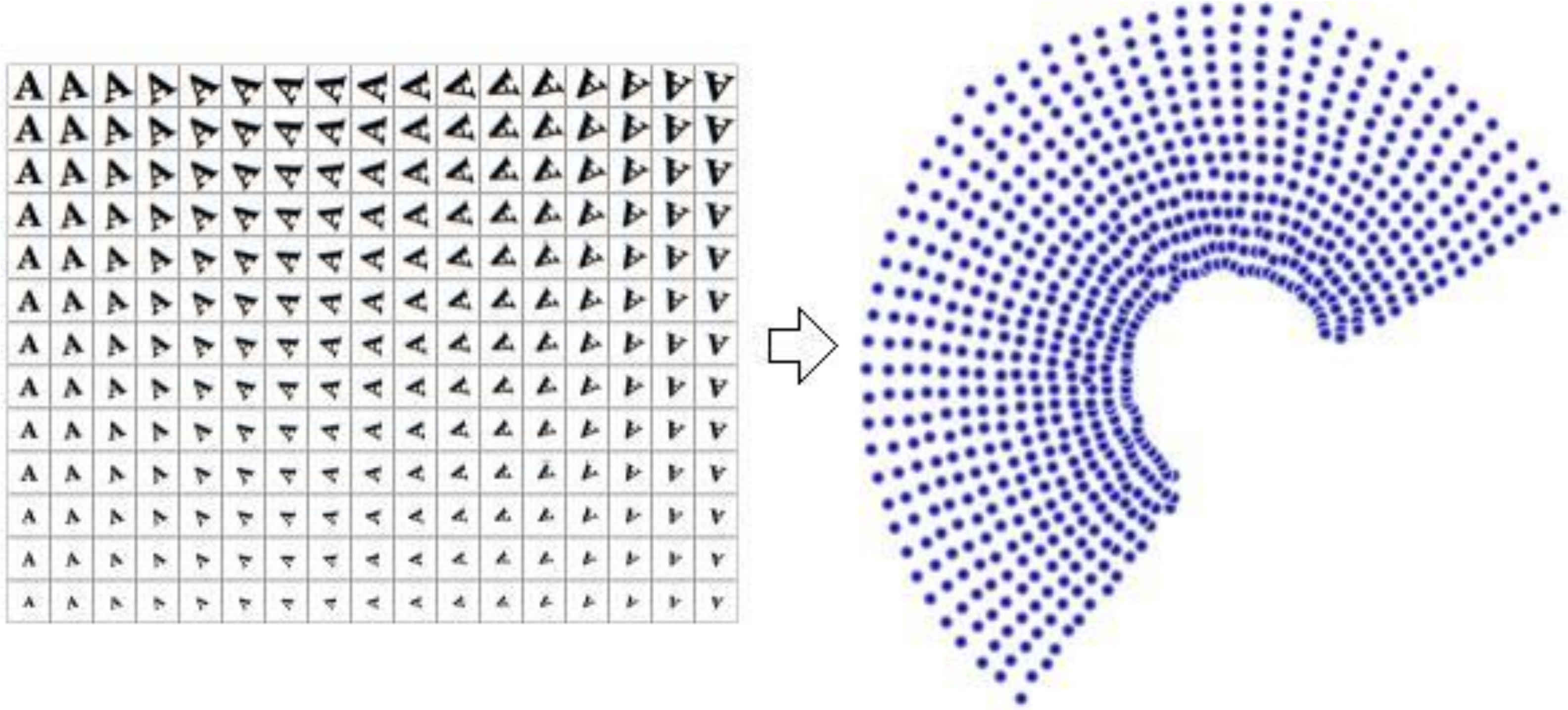
We want an MDS method that visualizes manifolds

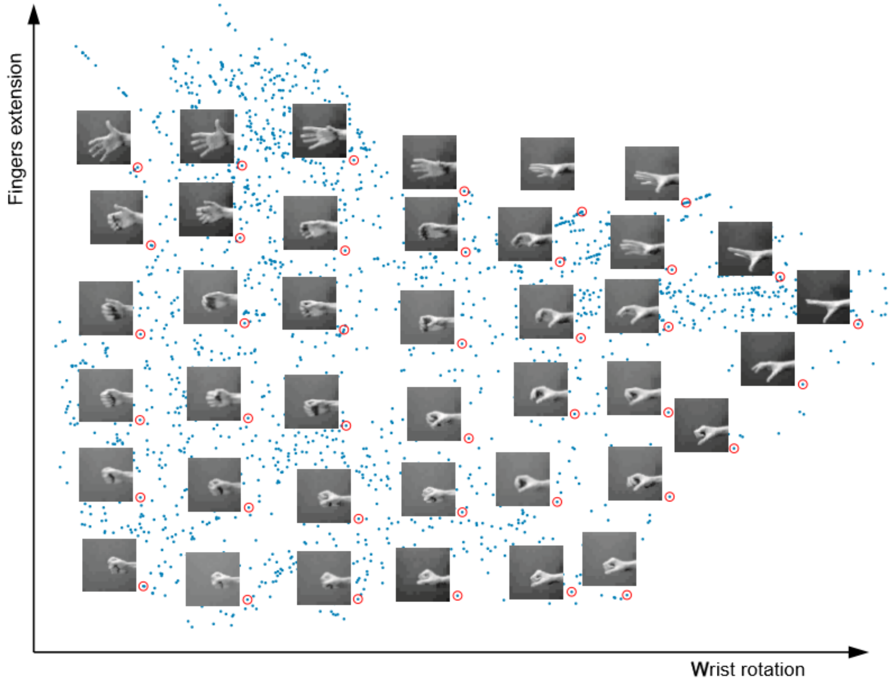


This visualization "unrolls" the Swiss roll.

Example: Manifolds in Image Space

- Slowly-varying image transformations exist on a manifold:



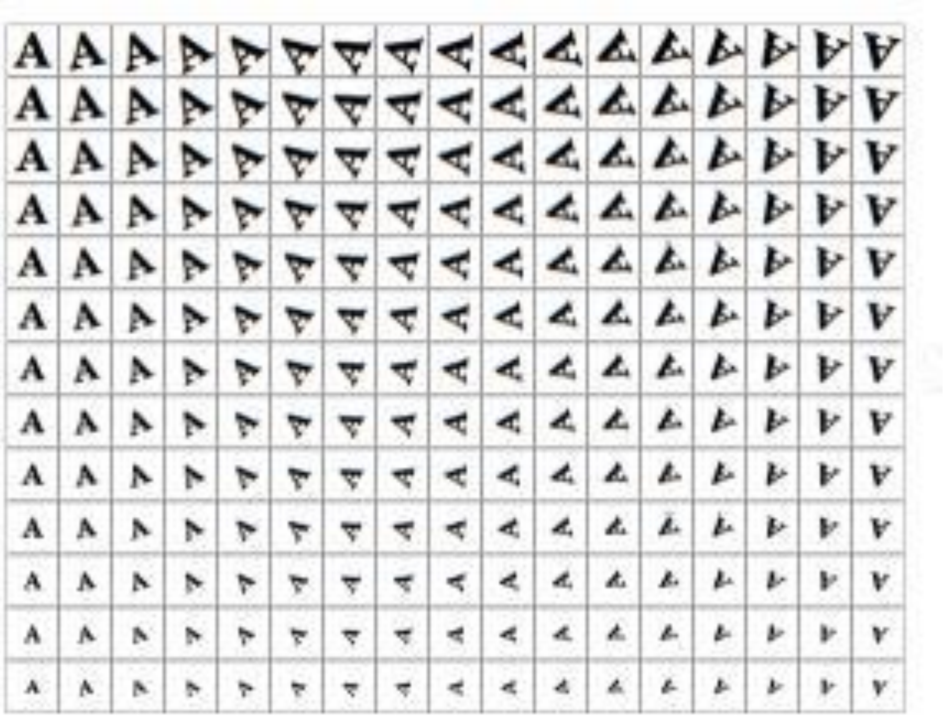


- “Neighbouring” images are close in Euclidean distance.
  – But distances between very-different images are not reliable.

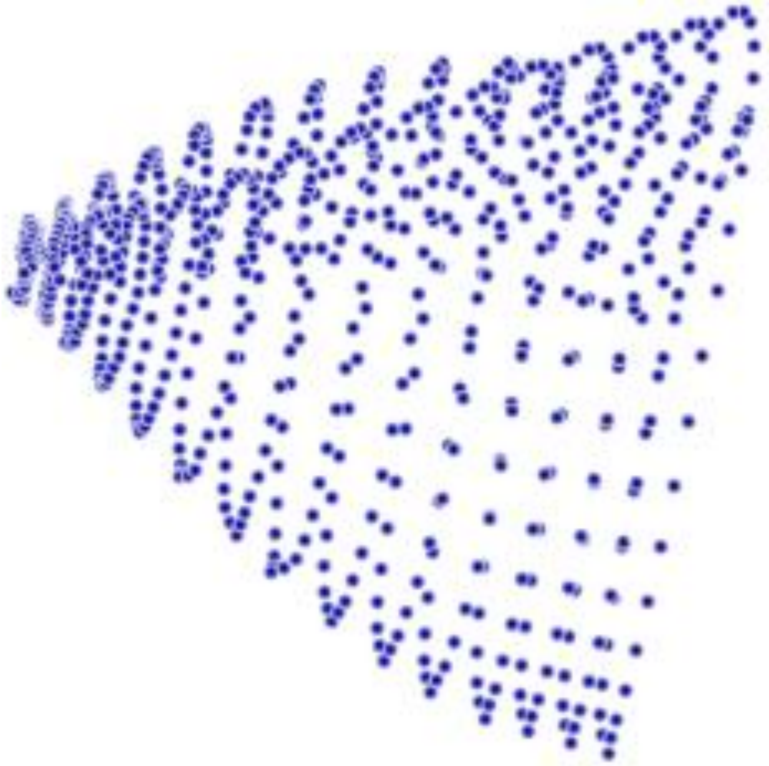
https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction
http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

Learning Manifolds

- With usual distances, PCA/MDS do not discover non-linear manifolds.



Original data

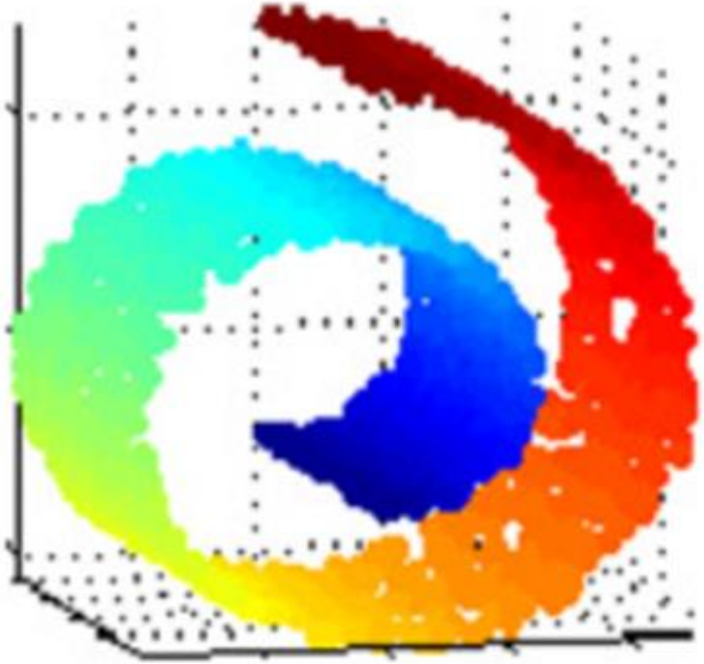


PCA

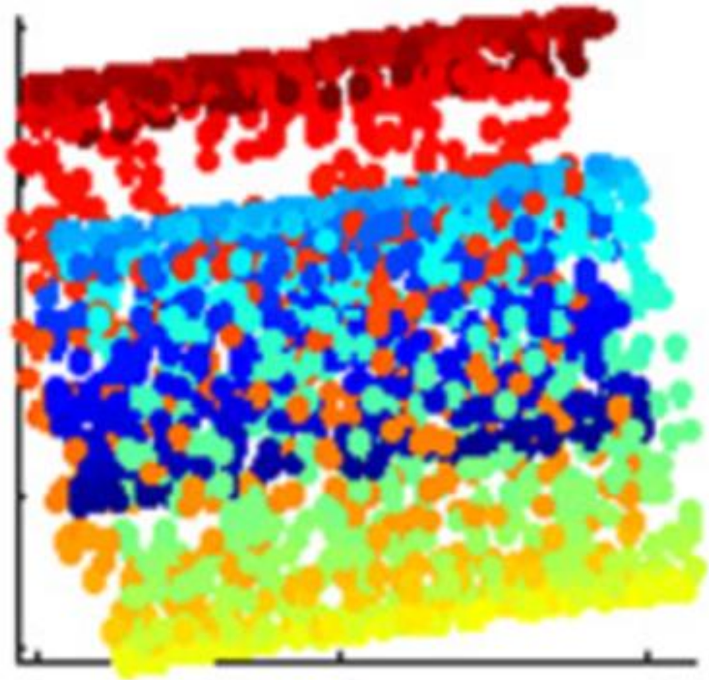
http://www.peh-med.com/content/9/1/12/figure/F1

Learning Manifolds

- With usual distances, PCA/MDS do not discover non-linear manifolds.



Original data

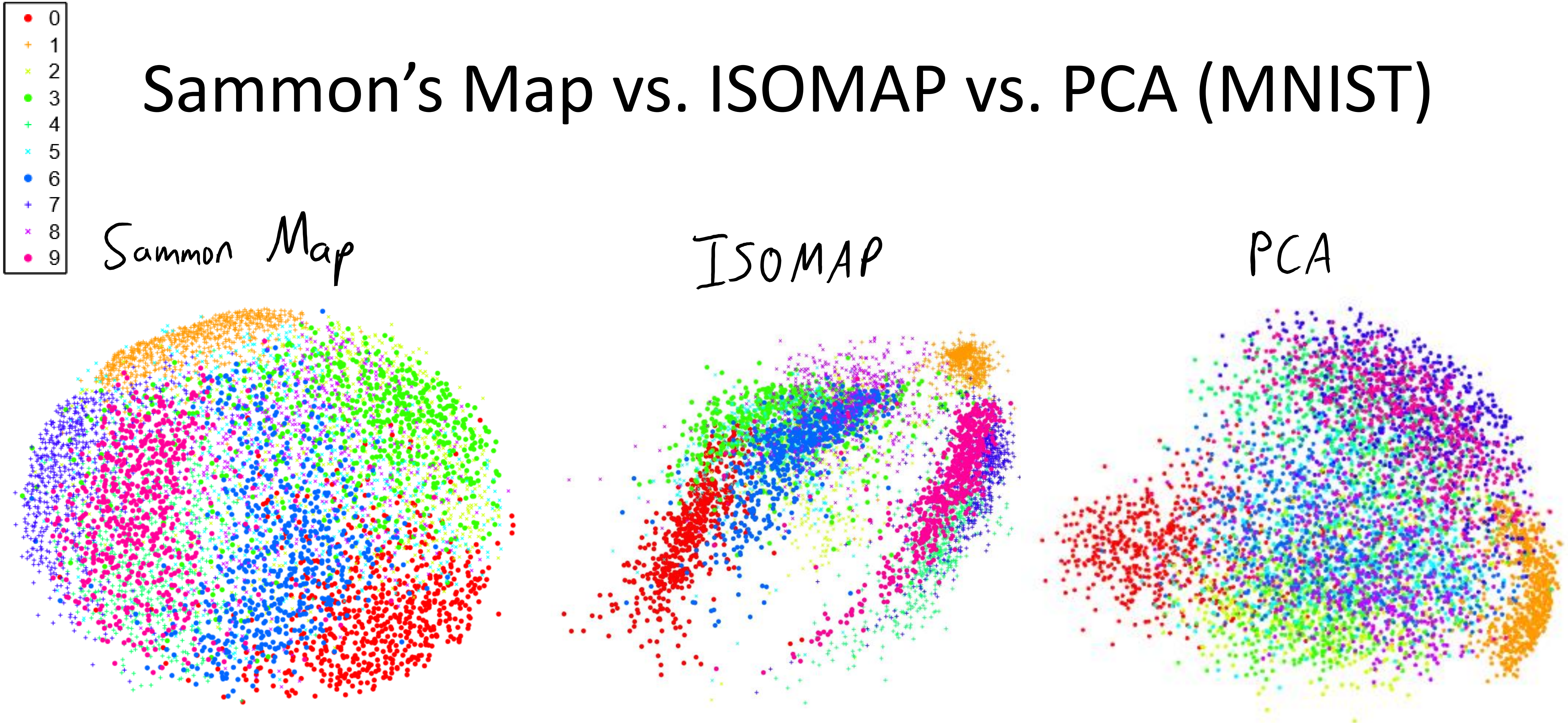


PCA

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

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

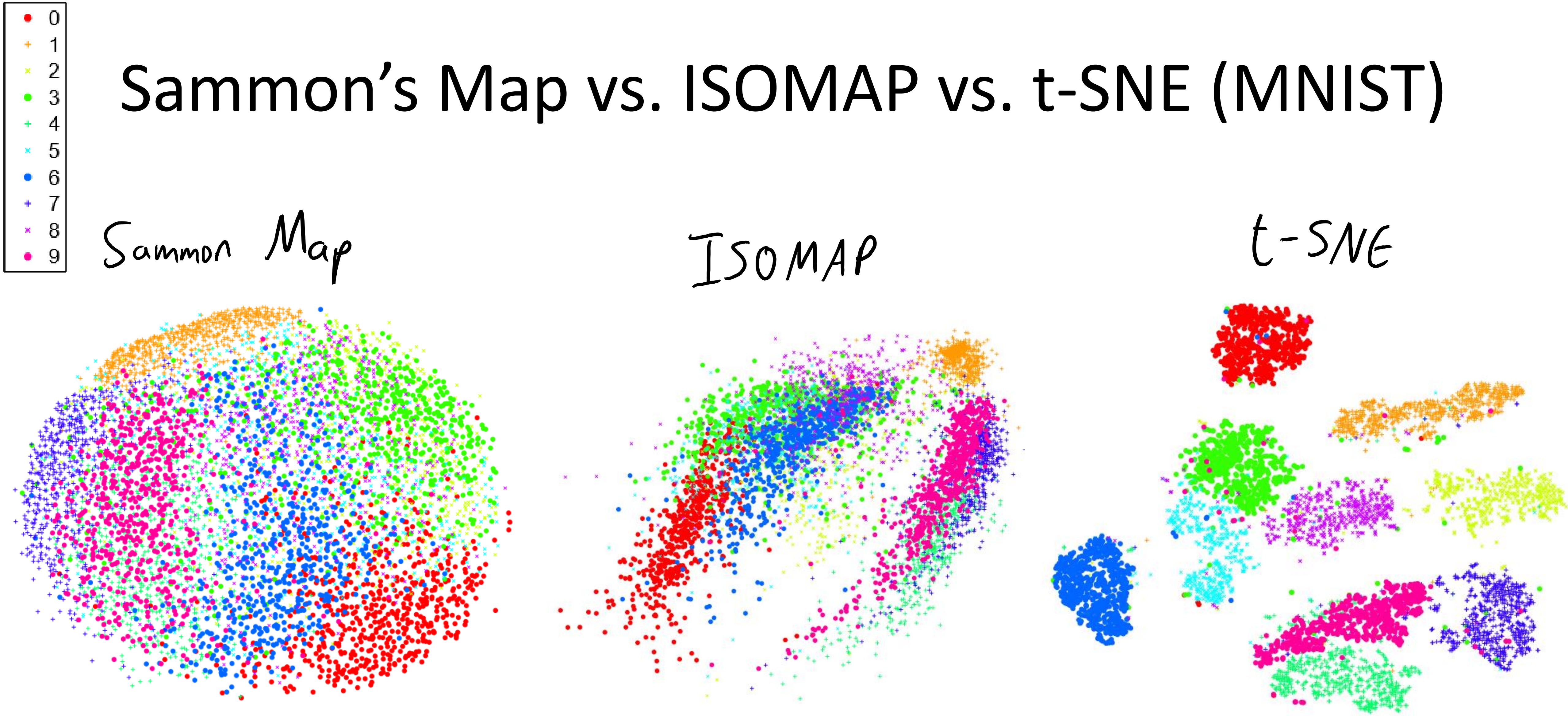
Sammon's Map vs. ISOMAP vs. PCA (MNIST)



- A classic way to visualize manifolds is ISOMAP.
  - Uses approximation of geodesic distance within MDS (see bonus slides).

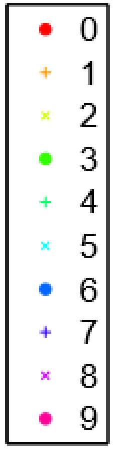
http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

Sammon’s Map vs. ISOMAP vs. t-SNE (MNIST)



- A modern way to visualize manifolds and clusters is t-SNE.

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

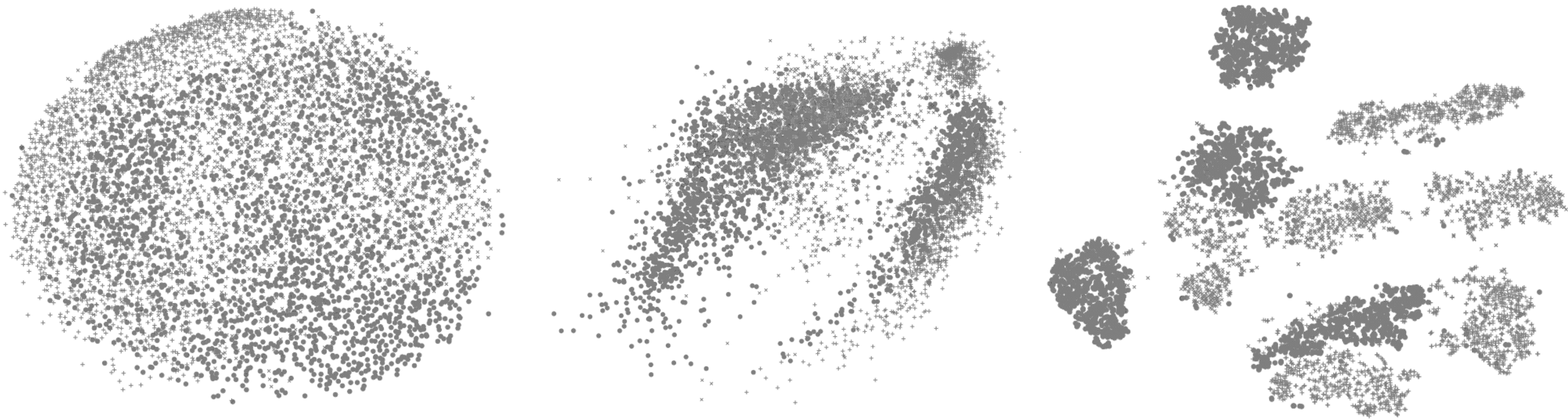


Sammon's Map vs. ISOMAP vs. t-SNE (MNIST)

Sammon Map

ISOMAP

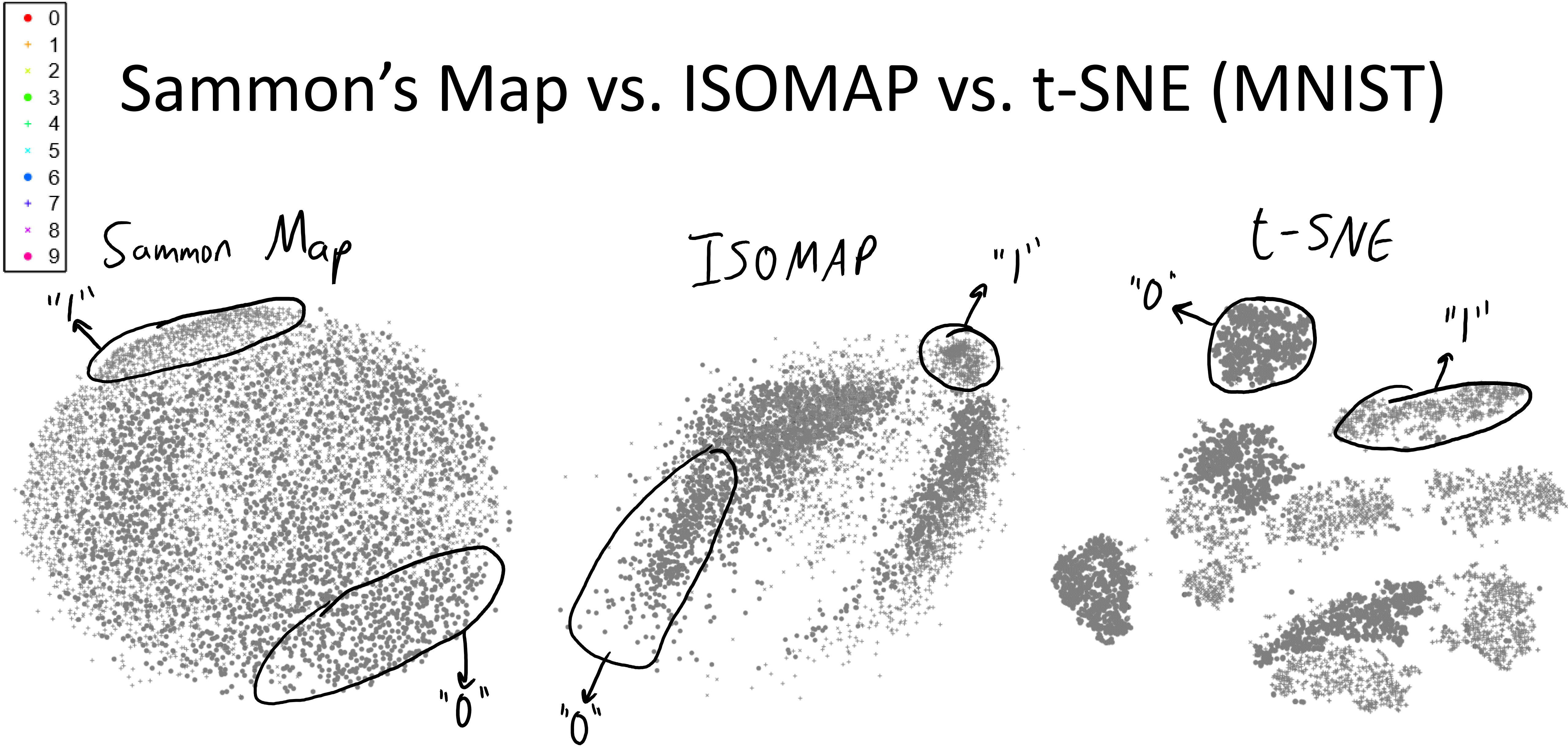
t-SNE



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

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

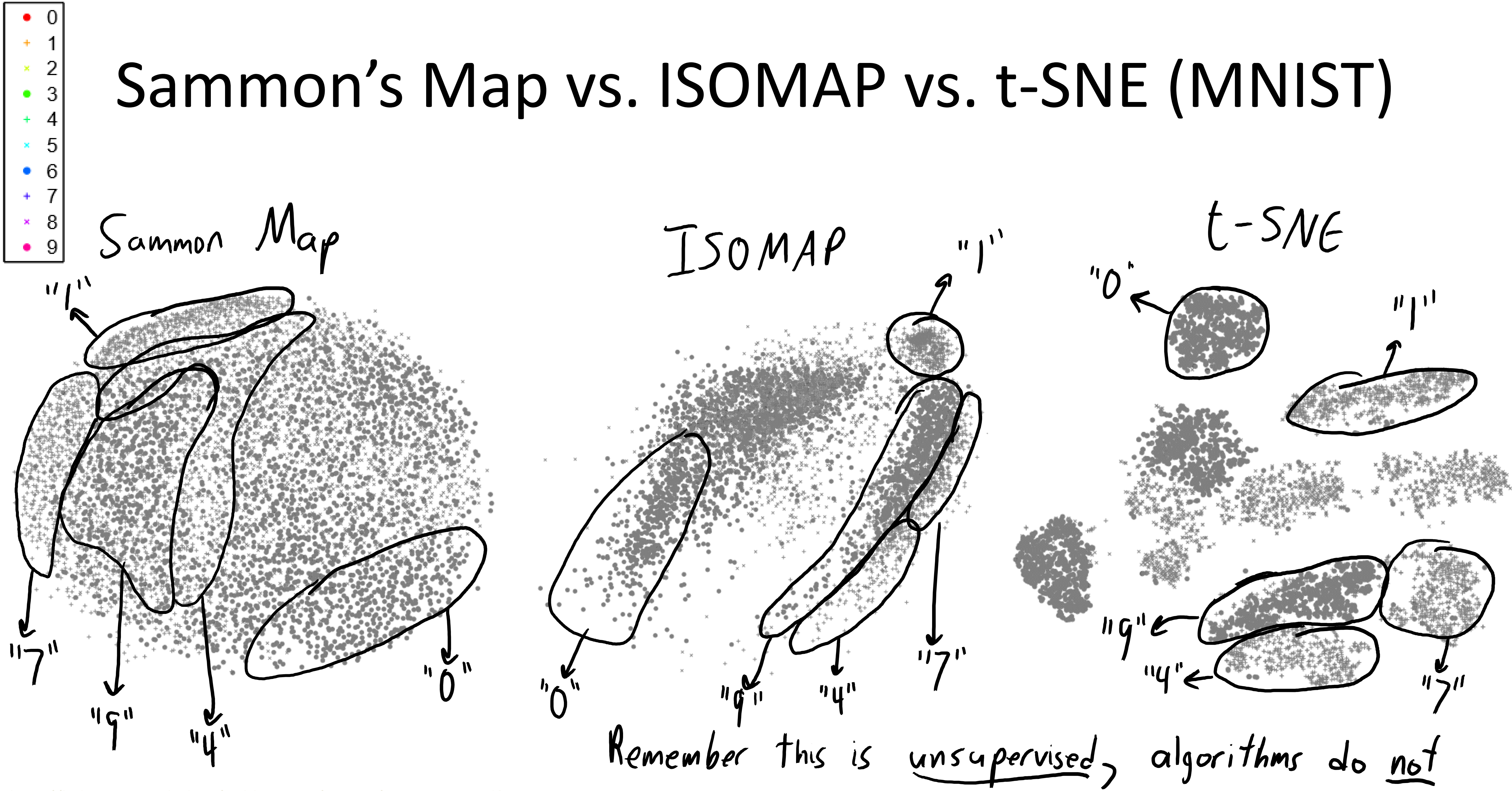
Sammon's Map vs. ISOMAP vs. t-SNE (MNIST)



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

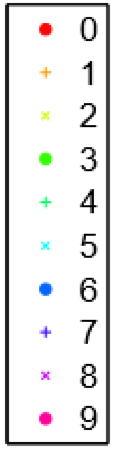
http://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

Sammon's Map vs. ISOMAP vs. t-SNE (MNIST)

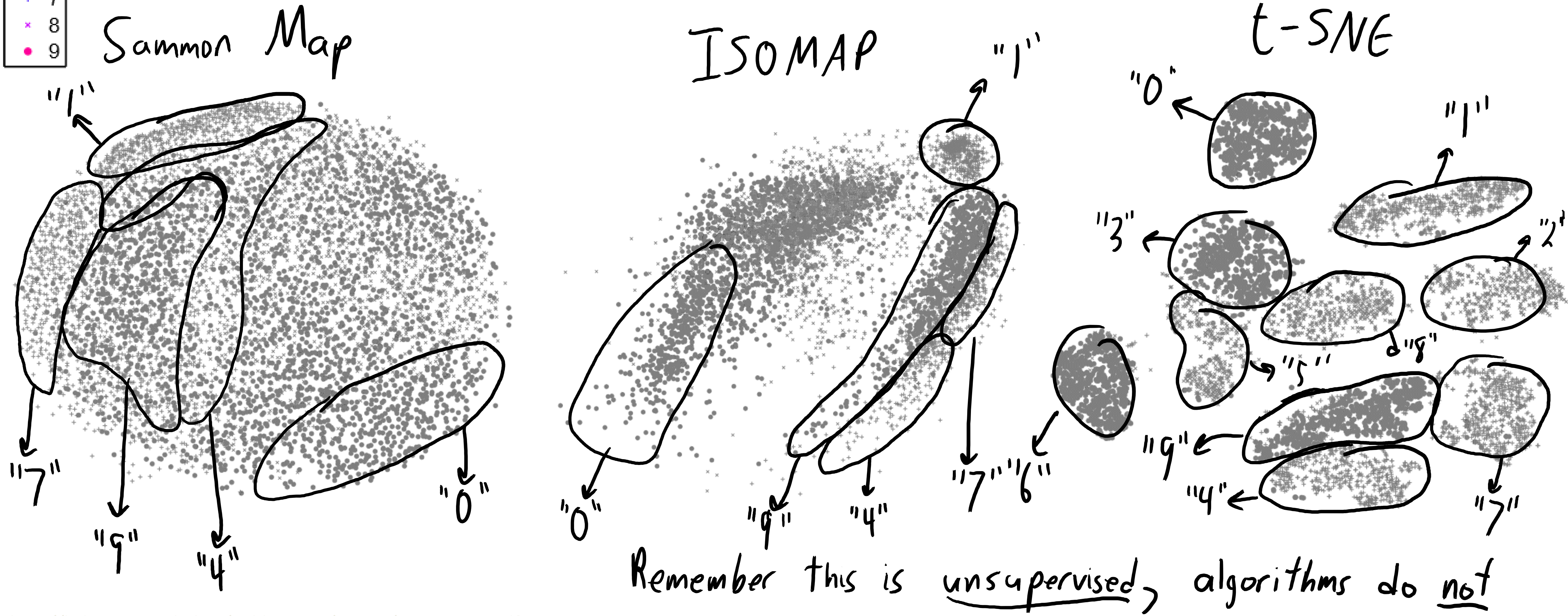


Remember this is unsupervised, algorithms do not know the labels.

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



Sammon’s Map vs. ISOMAP vs. t-SNE (MNIST)

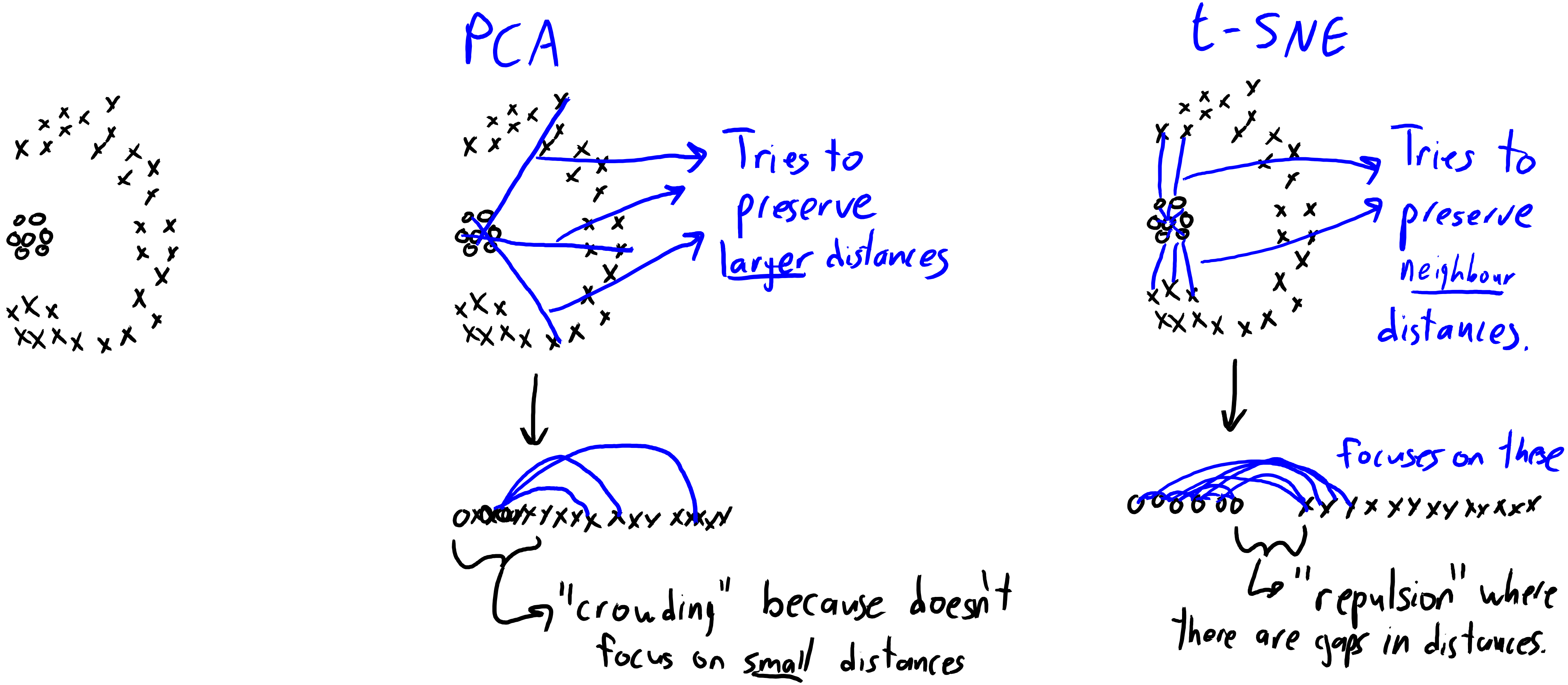


Remember this is unsupervised, algorithms do not know the labels.

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

t-Distributed Stochastic Neighbour Embedding

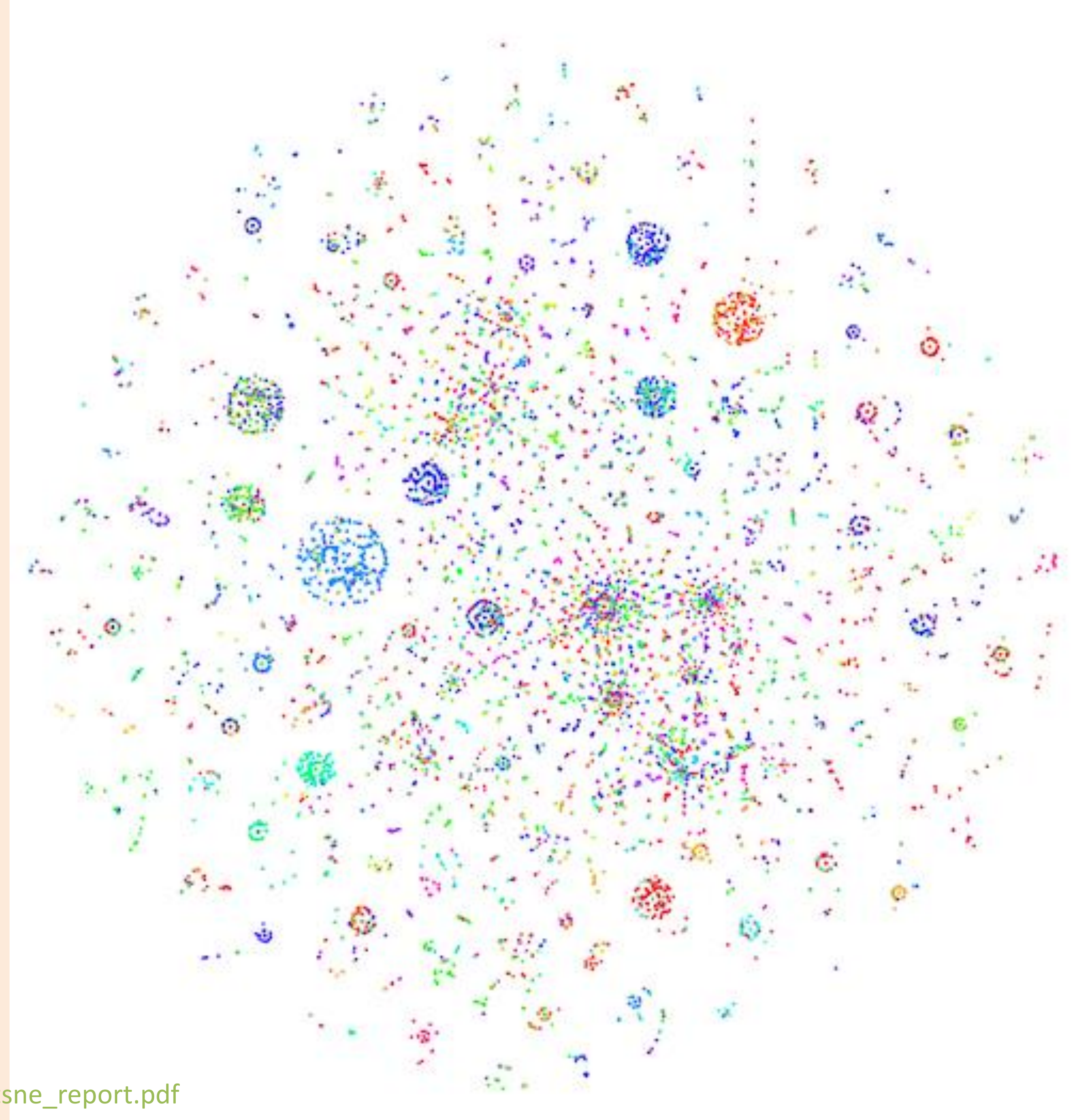
- One key idea in t-SNE:
  - Focus on distance to “neighbours” (allow large variance in other distances)


PCA
t-SNE
Tries to preserve larger distances
"crowding" because doesn't focus on small distances
Tries to preserve neighbour distances.
Focuses on these
"repulsion" where there are gaps in distances.

t-Distributed Stochastic Neighbour Embedding

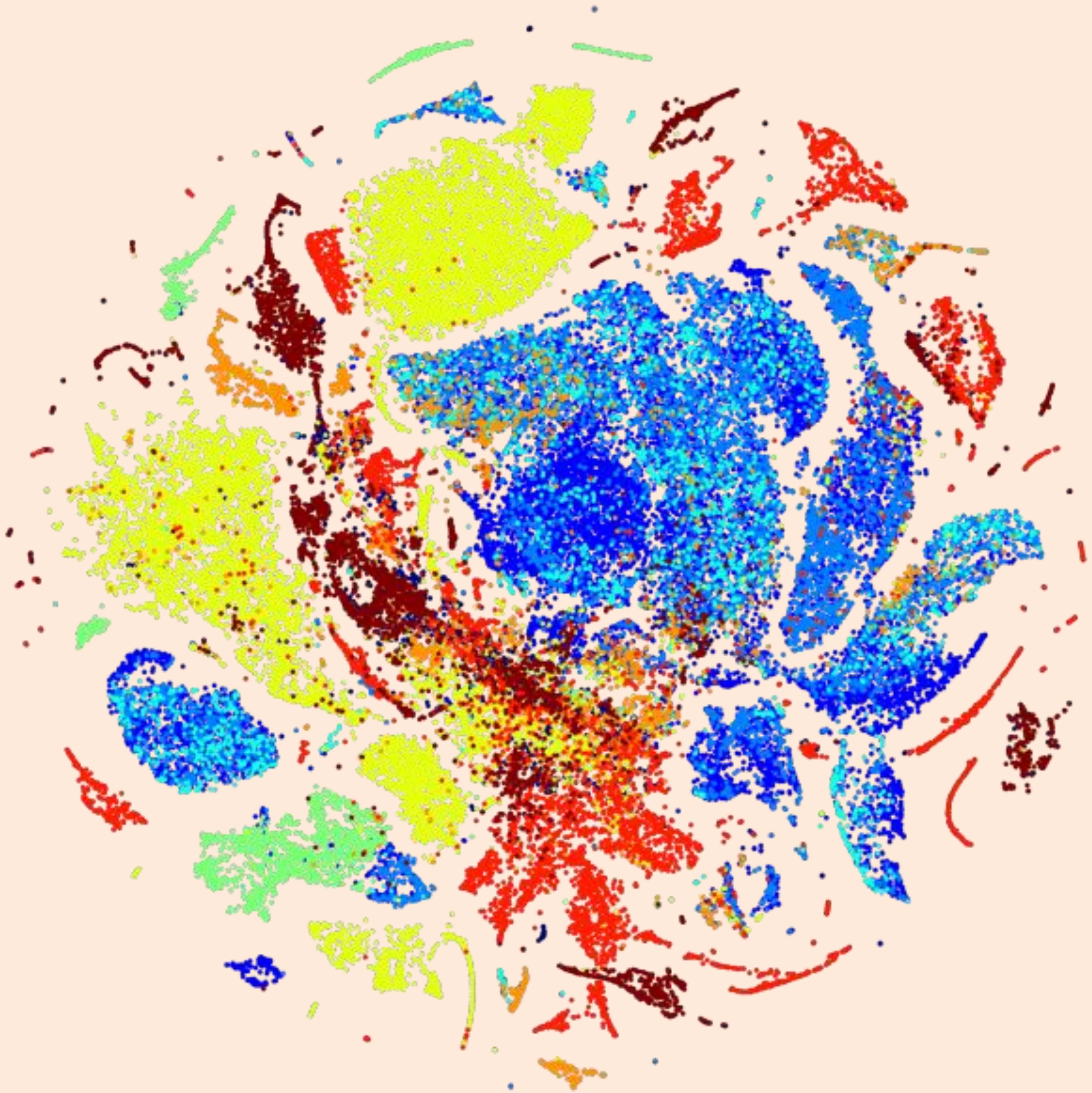
- t-SNE is a special case of MDS (specific d_1, d_2, and d_3 choices):
  - d_1: for each x_i, compute probability that each x_j is a 'neighbour'.
    - Computation is similar to k-means++, but most weight to close points (Gaussian).
    - Does not require explicit geodesic distance approximation.
  -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



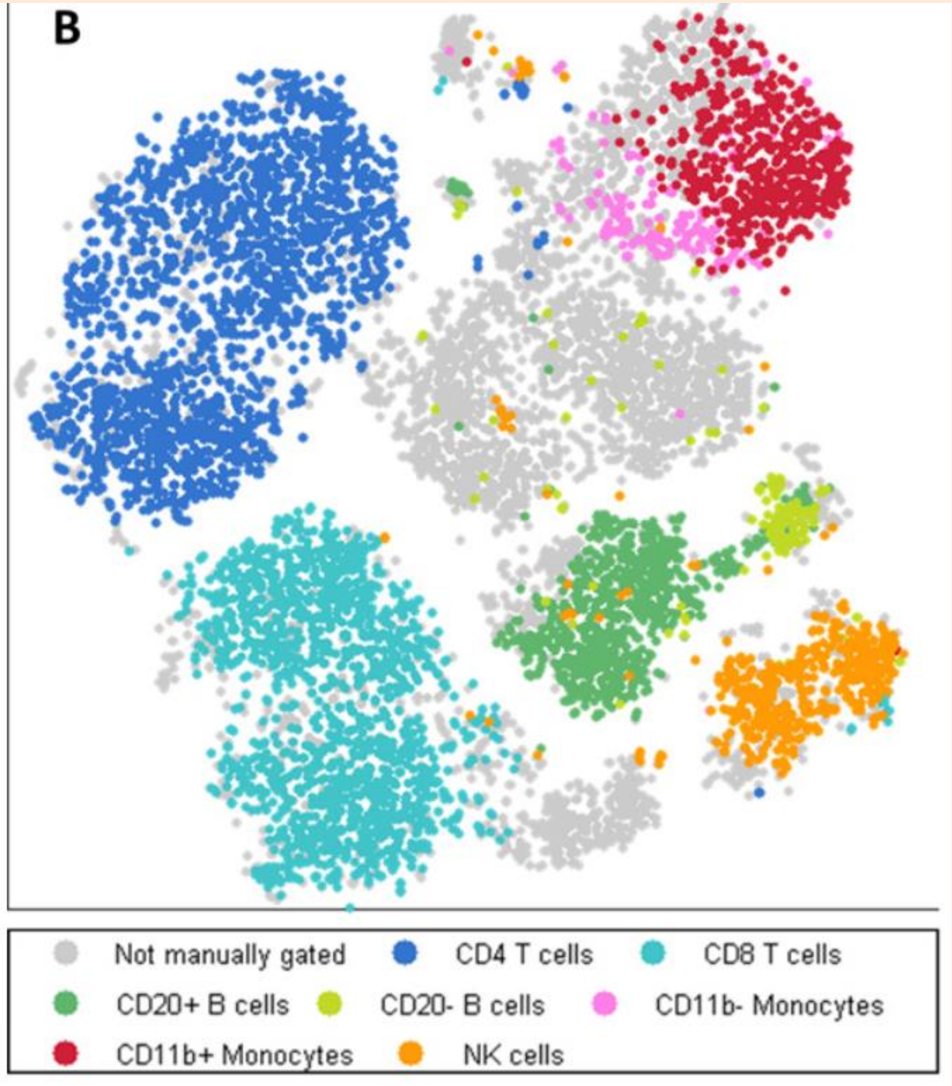
http://jasneetsabharwal.com/assets/files/wiki_tsne_report.pdf

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/

Next Topic: Word2Vec

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” features be related is 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 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 (Continuous Bag of Words)

- A common word2vec approaches (called continuous bag of words):
  - Each word ‘i’ is represented by a vector of real numbers z_i.
  - Training data: sentence fragments with “hidden” middle word:
    - “We introduce basic ~~principles~~ and techniques in”
    - “the fields of ~~data~~ mining and machine”
    - “tools behind the ~~emerging~~ field of data”
    - “techniques are now ~~running~~ behind the scenes”
    - “discover patterns and ~~make~~ predictions in various”
    - “the core data ~~mining~~ and machine learning”
    - “with motivating applications ~~from~~ a variety of”
  - Train so that z_i of “hidden” words is are similar to z_i of surrounding words.

Word2Vec (Continuous Bag of Words)

- Continuous bag of words model probability of middle word ‘i’ as:

$$\prod_{j \in \text{surrounding words}} \frac{\exp(z_i^T z_j)}{\sum_{c=1}^{\#\text{words}} \exp(z_c^T z_j)}$$

- We use gradient descent on negative logarithm of these probabilities:
  - Makes $z_i^T z_j$ big for words appearing in same context (making z_i close to z_j).
  - Makes $z_i^T z_j$ small for words not appearing together (makes z_i and z_j far).

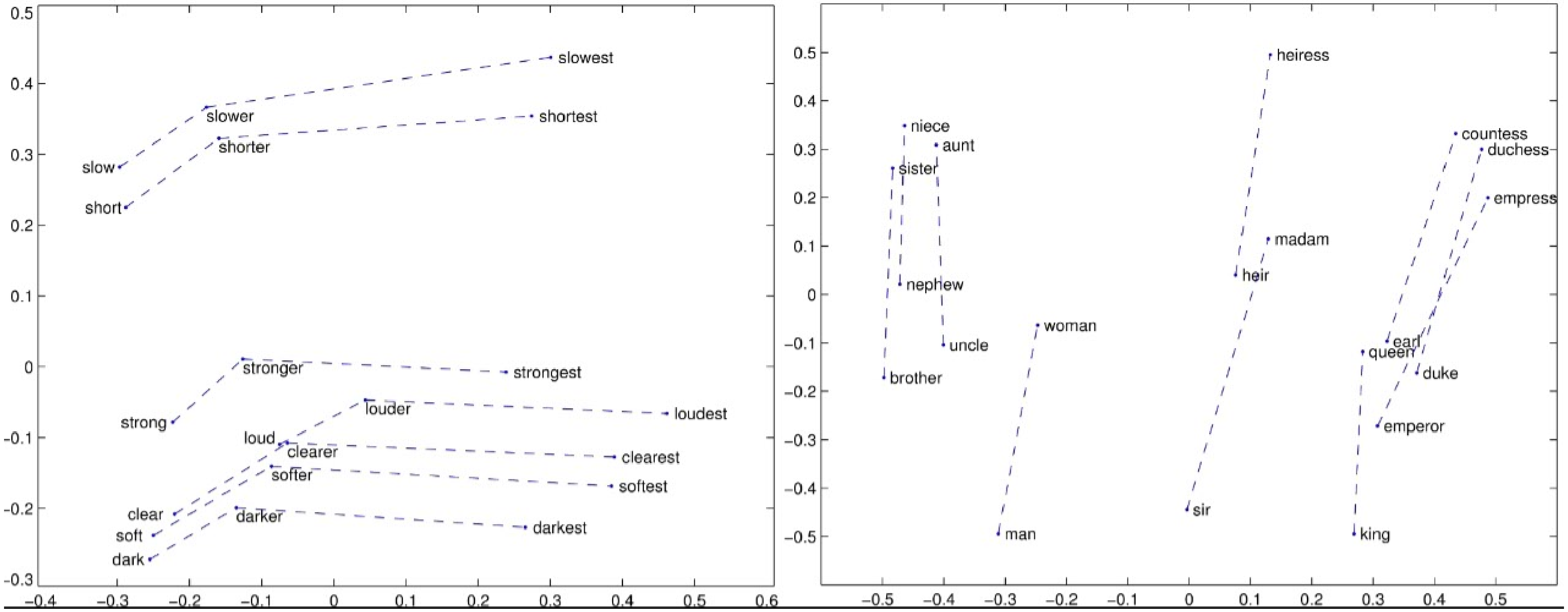
- Once trained, you use these z_i as features for language tasks.
  - Tends to work much better than bag of words.
  - Allows you to get useful features of words from unlabeled text data.

Word2Vec (Skip-Gram)

- A common word2vec approaches (skip gram):
  - Each word ‘i’ is represented by a vector of real numbers z_i.
  - Training data: sentence fragments with “hidden” surrounding word:
    - “~~We introduce basic~~ principles ~~and techniques in~~”
    - “~~the fields of~~ data ~~mining and machine~~”
    - “~~tools behind the~~ emerging ~~field of data~~”
    - “~~techniques are now~~ running ~~behind the scenes~~”
    - “~~discover patterns and~~ make ~~predictions in various~~”
    - “~~the core data~~ mining ~~and machine learning~~”
    - “~~with motivating applications~~ from ~~a variety of~~”
  - Train so that z_i of “hidden” words is are similar to z_i of surrounding words.
    - Uses same probability as continuous bag of words.
      - But denominator sums over all possible surrounding words (often just sample terms for speed).

Word2Vec Example

- MDS visualization of a set of related words:



- Distances between vectors might represent semantics.

http://sebastianruder.com/secret-word2vec

Word2Vec

- Subtracting word vectors to find related vectors.

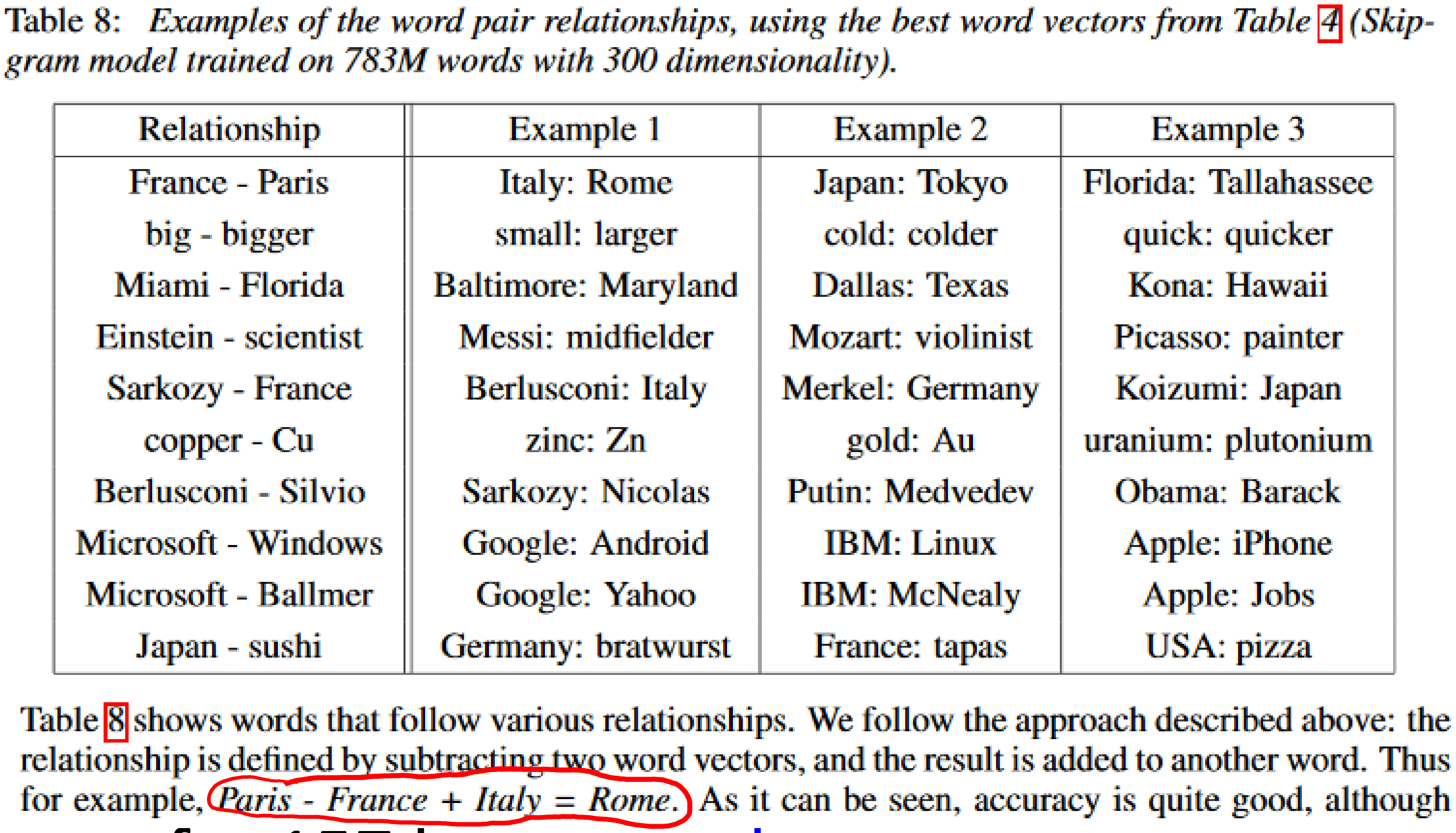


Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

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

- Word vectors for 157 languages here.

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

Summary

- Multi-dimensional scaling is a non-parametric latent-factor model.
- Different MDS distances/losses/weights usually gives better results.
- Manifold: space where local Euclidean distance is accurate.
  - Structured data like images often form manifolds in space.
- t-SNE is an MDS method focusing on matching small distances.
- Word2vec:
  - Latent-factor (continuous) representation of words.
  - Based on predicting word from its context (or context from word).

- Next time: deep learning.

Stochastic Gradient for SVDfeature

- Common approach to fitting SVDfeature is stochastic gradient.
- Previously you saw stochastic gradient for supervised learning:
  - Choose a random example 'i'
  - Update parameters 'w' using gradient of example 'i'
- Stochastic gradient for SVDfeature (formulas as bonus):
  - Choose a random user 'i' and a random product 'j'
  - Update β, β_i, β_j, w, z_i, and w^j based on their gradient for <u>this</u> user-product.

Updated <u>every</u> time

SVDfeature with SGD: the gory details

Objective: $\frac{1}{2}\sum_{(i,j)\in R}(\underbrace{\hat{y}_{ij} - y_{ij}}_{r_{ij}})^2$ with $\hat{y}_{ij} = \beta + \beta_i + \beta_j + w^T x_{ij} + (w^j)^T z_i$

Update based on random (i,j):

$$\underbrace{\begin{aligned} \beta &= \beta - \alpha r_{ij} \\ \beta_i &= \beta_i - \alpha r_{ij} \\ \beta_j &= \beta_j - \alpha r_{ij} \end{aligned}}_{\text{Updates are the } \underline{\text{same}}\text{,}}$$

but 'β' is always update while β_i and β_j are only updated for the specific user + product

$$w = w - \alpha r_{ij} x_{ij} \leftarrow \text{Updated } \underline{\text{every}} \text{ time.}$$

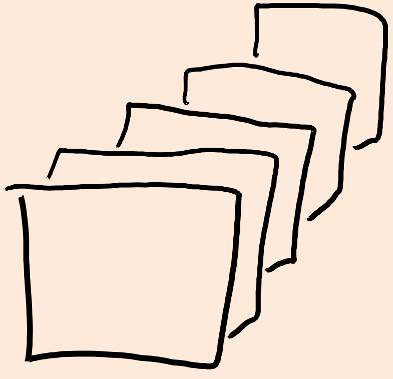
$$\left.\begin{aligned} z_i &= z_i - \alpha r_{ij} w^j \\ w^j &= w^j - \alpha r_{ij} z_i \end{aligned}\right\} \rightarrow \text{Updated for } \underline{\text{Specific}} \text{ user and product.}$$

(Adding regularization adds an extra term)

Tensor Factorization

- Tensors are higher-order generalizations of matrices:

Scalar $\alpha = \underset{1\times 1}{[\;]}$ Vector $a = [\;\;]_{d\times 1}$ Matrix $A = \underset{d \times d}{\left[\quad\quad\quad\right]}$ Tensor $A =$



$d \times d \times d$

- Generalization of matrix factorization is tensor factorization:

$$y_{ijm} \approx \sum_{c=1}^{k} w_{jc} z_{ic} v_{mc}$$

- Useful if there are other relevant variables:
    - Instead of ratings based on {user,movie}, ratings based {user,movie,group}.
    - Useful if you have groups of users, or if ratings change over time.

Field-Aware Matrix Factorization

- Field-aware factorization machines (FFMs):
  - Matrix factorization with multiple z_i or w_c for each example or part.
  - You choose which z_i or w_c to use based on the value of feature.

- Example from "click through rate" prediction:
  - E.g., predict whether "male" clicks on "nike" advertising on "espn" page.
  - A previous matrix factorization method for the 3 factors used:

$$w_{espn} w_{nike} + w_{espn} w_{male} + w_{nike} w_{male}$$

  - FFMs could use:

$$w_{espn}^{A} w_{nike}^{P} + w_{espn}^{G} w_{male}^{P} + w_{nike}^{G} w_{male}^{A}$$

    - wespnA is the factor we use when multiplying by a an advertiser's latent factor.
    - wespnG is the factor we use when multiplying by a group's latent factor.

- This approach has won some Kaggle competitions (link), and has shown to work well in production systems too (link).

Warm-Starting

- We’ve used data {X,y} to fit a model.
- We now have new training data and want to fit new and old data.

- Do we need to re-fit from scratch?

- This is the warm starting problem.
  - It’s easier to warm start some models than others.

Easy Case: K-Nearest Neighbours and Counting

- K-nearest neighbours:
  - KNN just stores the training data, so just store the new data.

- Counting-based models:
  - Models that base predictions on frequencies of events.
  - E.g., naïve Bayes.
  - Just update the counts:

$$p(\text{"vicodin"} \mid \text{"spam"}) = \frac{\text{count of } \{\text{vicodin}, \text{spam}\} \text{ in } \underline{\text{new and old}} \text{ data}}{\text{count of "spam" in } \underline{\text{new and old}} \text{ data}}$$

  - Decision trees with fixed rules: just update counts at the leaves.

Medium Case: L2-Regularized Least Squares

- L2-regularized least squares is obtained from linear algebra:

$$w = (X^TX + \lambda I)^{-1}(X^Ty)$$

  - Cost is O($nd^2 + d^3$) for ‘n’ training examples and ‘d’ features.
- Given one new point, we need to compute:
  - X^Ty with one row added, which costs O(d).
  - Old X^TX plus $x_ix_i^T$, which costs O(d^2).
  - Solution of linear system, which costs O(d^3).
  - So cost of adding ‘t’ new data point is O(td^3).

- With “matrix factorization updates”, can reduce this to O(td^2).
  - Cheaper than computing from scratch, particularly for large d.

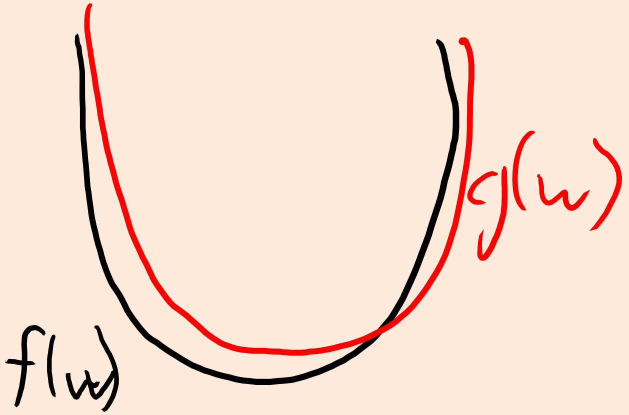
Medium Case: Logistic Regression

- We fit logistic regression by gradient descent on a convex function.

- With new data, convex function f(w) changes to new function g(w).

$$f(w) = \sum_{i=1}^{n} f_i(w) \qquad g(w) = \sum_{i=1}^{n+1} f_i(w)$$

- If we don't have much more data, 'f' and 'g' will be "close".
  - Start gradient descent on 'g' with minimizer of 'f'.
  - You can show that it requires fewer iterations.



Hard Cases: Non-Convex/Greedy Models

- For decision trees:
  - “Warm start”: continue splitting nodes that haven’t already been split.
  - “Cold start”: re-fit everything.

- Unlike previous cases, this won’t in general give same result as re-fitting:
  - New data points might lead to different splits higher up in the tree.

- Intermediate: usually do warm start but occasionally do a cold start.

- Similar heuristics/conclusions for other non-convex/greedy models:
  - K-means clustering.
  - Matrix factorization (though you can continue PCA algorithms).

Different MDS Cost Functions

- MDS default objective function with general distances/similarities:

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

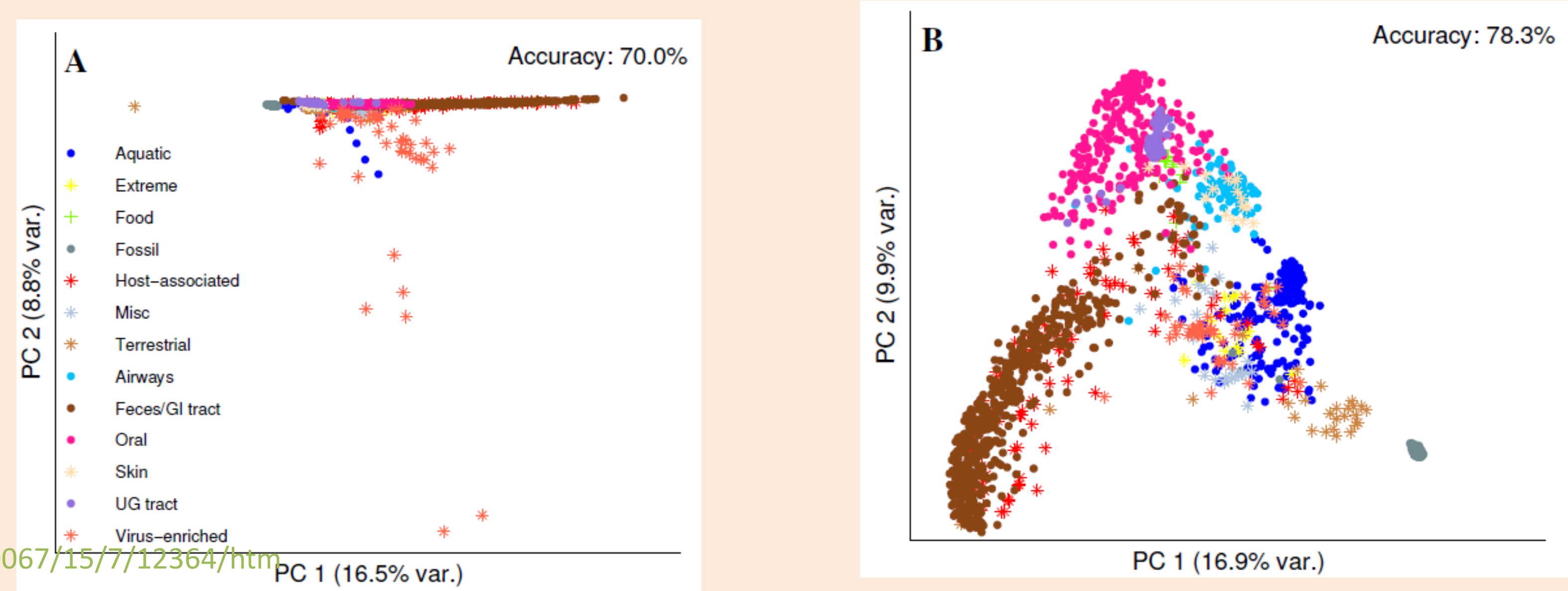
- A possibility is "classic" MDS with $d_1(x_i,x_j) = x_i^T x_j$ and $d_2(z_i,z_j) = z_i^T z_j$.
  - We obtain PCA in this special case (centered x_i, d_3 as the squared L2-norm).
  - Not a great choice because it's a linear model.

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.



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

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.

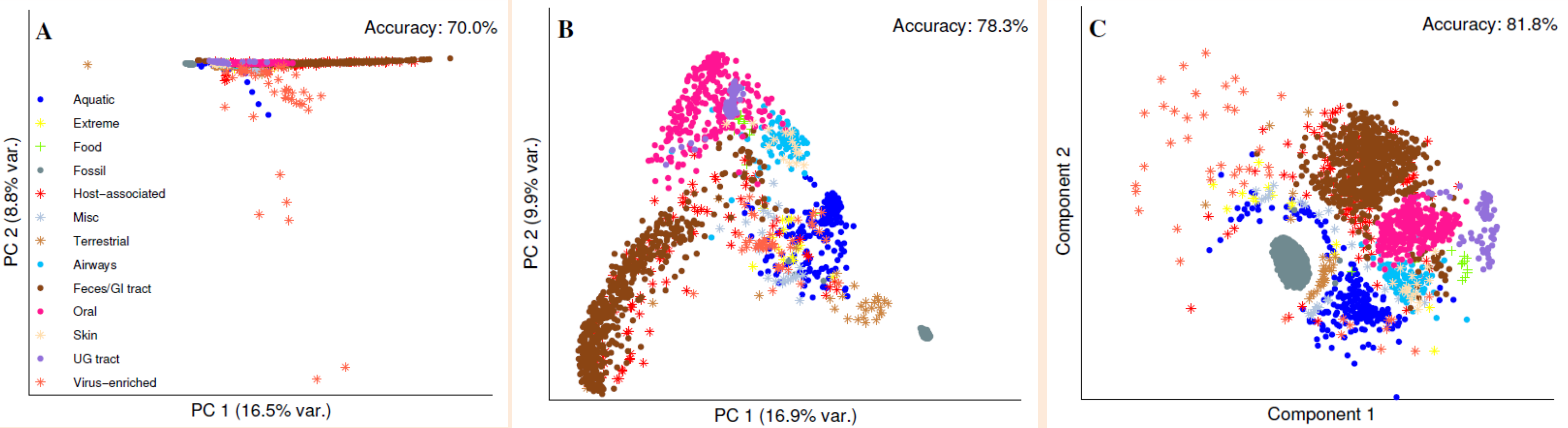
$$f(Z) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left(\frac{d_2(z_i, z_j) - d_1(x_i, x_j)}{d_1(x_i, x_j)} \right)^2$$

  - Denominator reduces focus on large distances.

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

Sammon’s Mapping

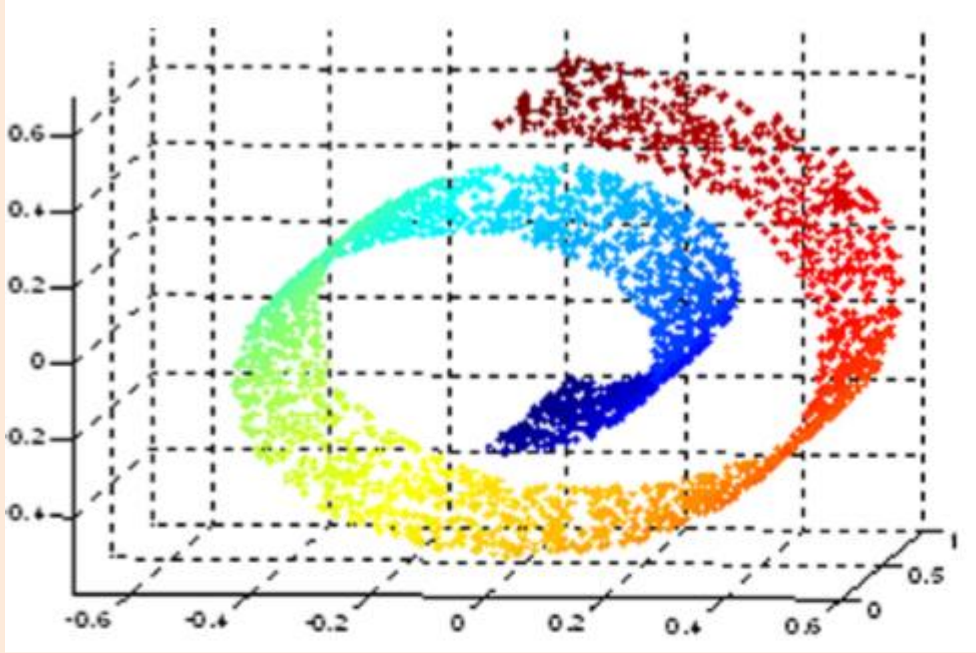
- Challenge for most MDS models: they focus on large distances.
  - Leads to “crowding” effect like with PCA.
- Early attempt to address this is Sammon’s mapping:
  - Weighted MDS so large/small distances are more comparable.

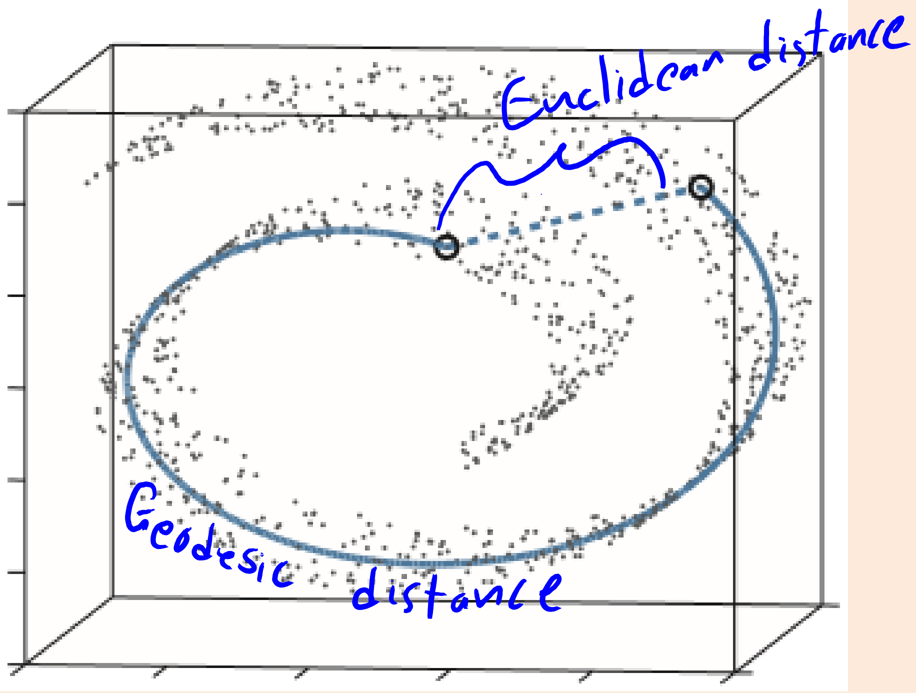


http://www.mdpi.com/1422-0067/15/7/12364/htm

Geodesic Distance on 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.

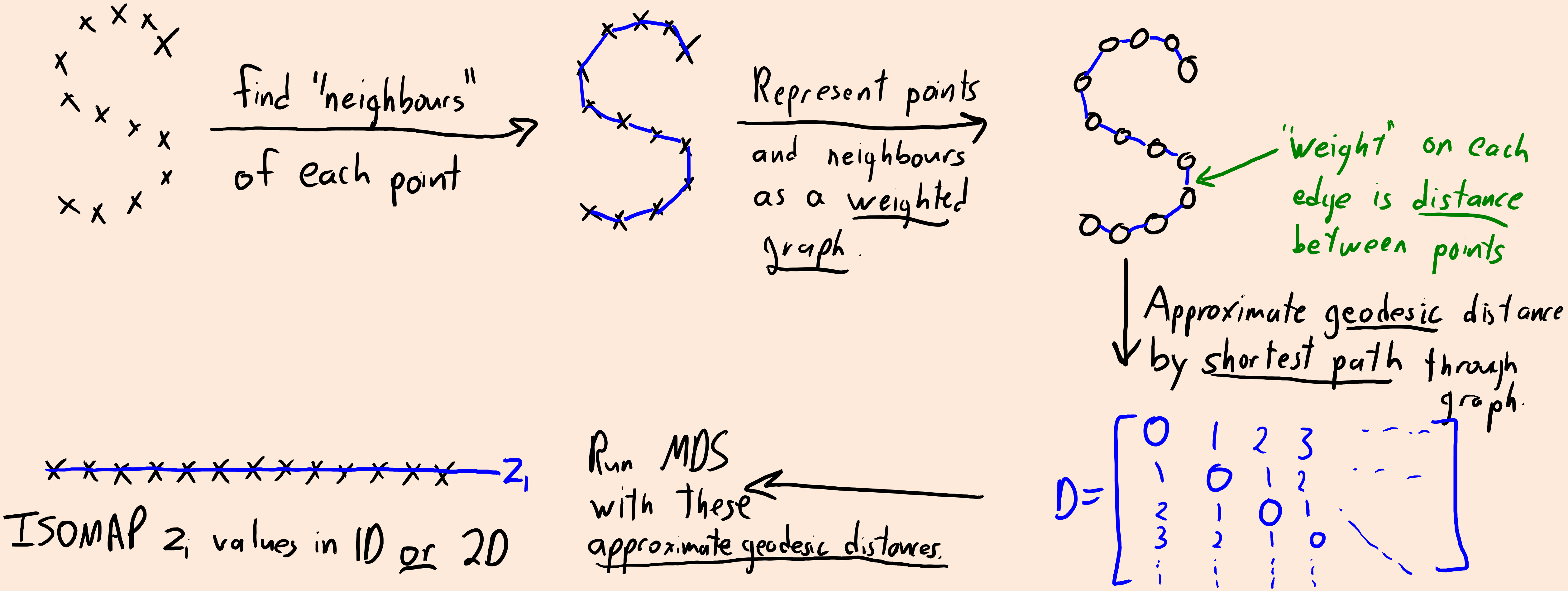




http://www.biomedcentral.com/content/pdf/1471-2105-13-S7-S3.pdf
http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

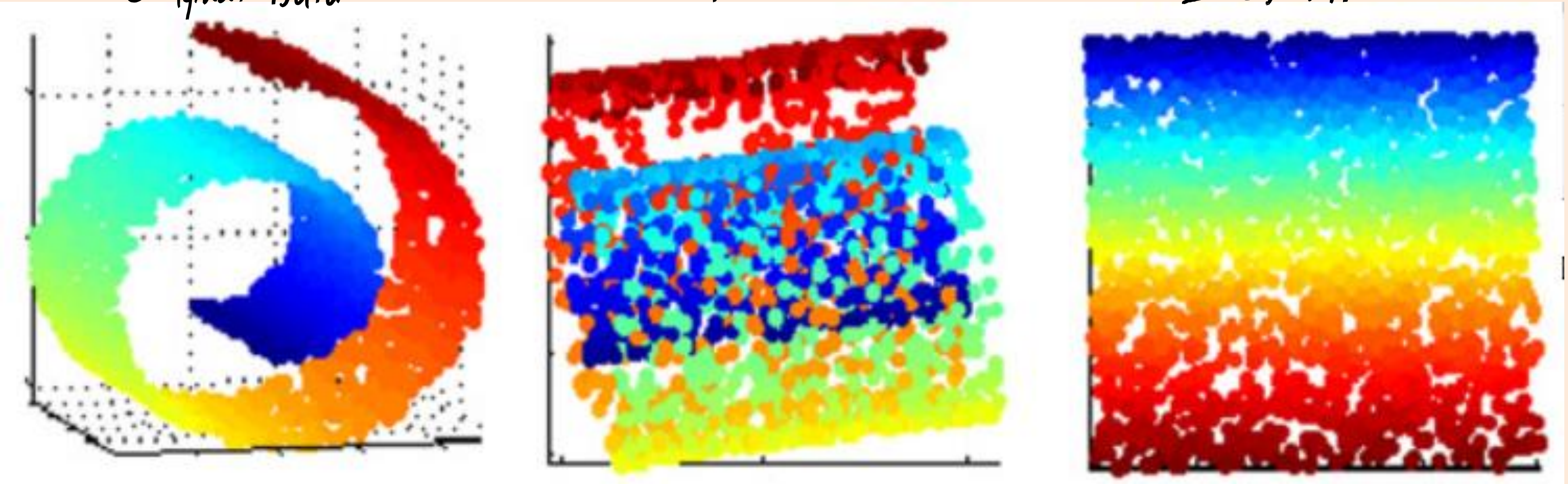
ISOMAP

- 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=
0 1 2 3 ...
1 0 1 2 ...
2 1 0 1
3 2 1 0
⋮ ⋮ ⋮ ⋮
Run MDS with these approximate geodesic distances.
z_i
ISOMAP z_i values in 1D or 2D

ISOMAP

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



Original Data

PCA

ISOMAP

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

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

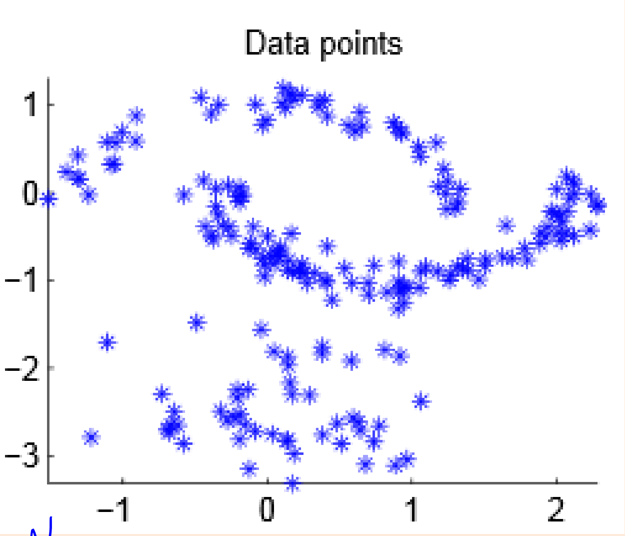
Constructing Neighbour Graphs

- 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

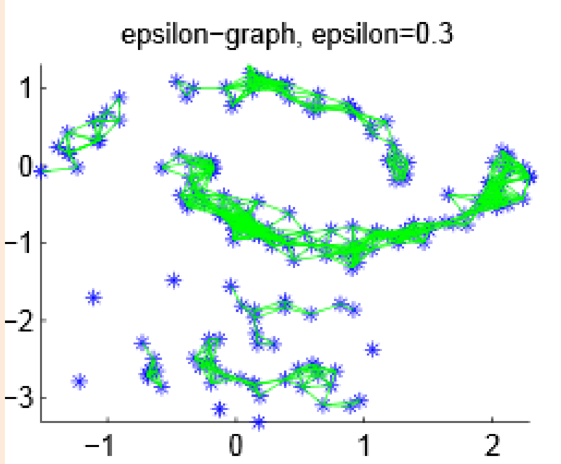
Converting from Features to Graph

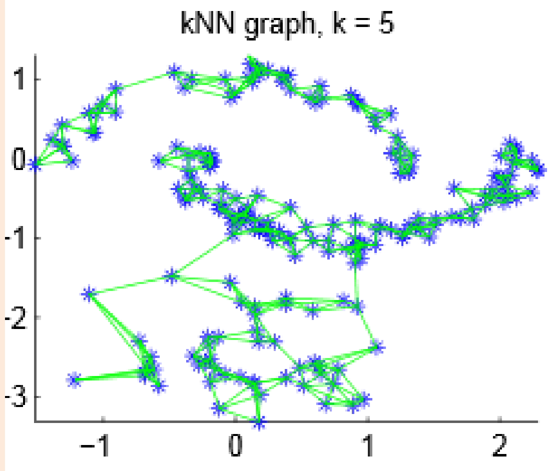


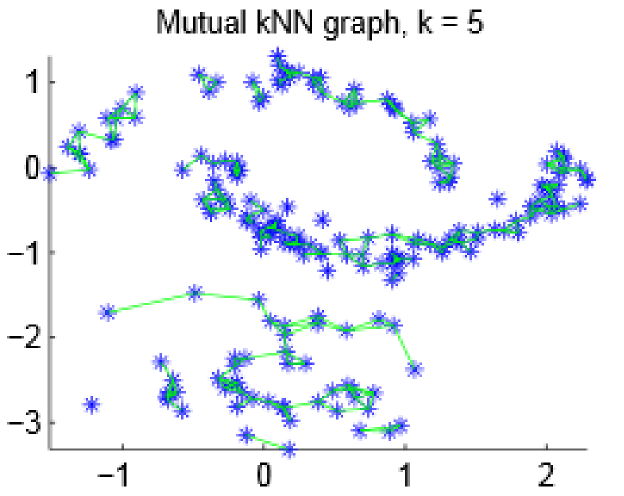
add edge if $\|x_i - x_j\| \le 0.3$

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.



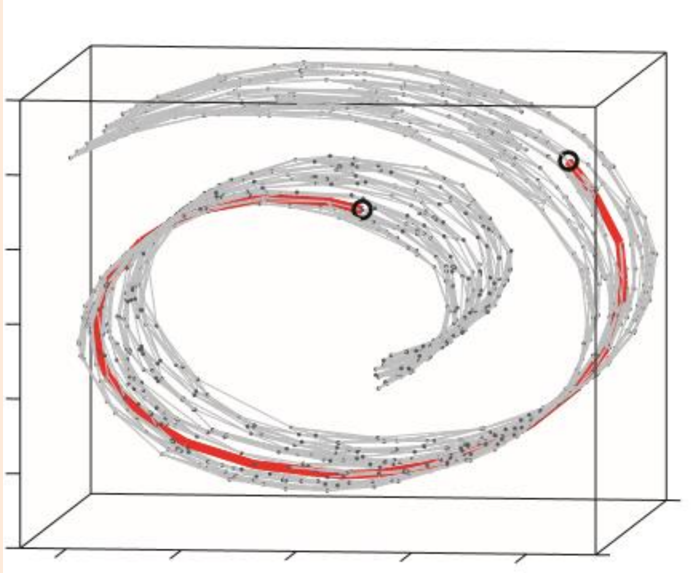




http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/Luxburg07_tutorial_4488%5B0%5D.pdf

ISOMAP

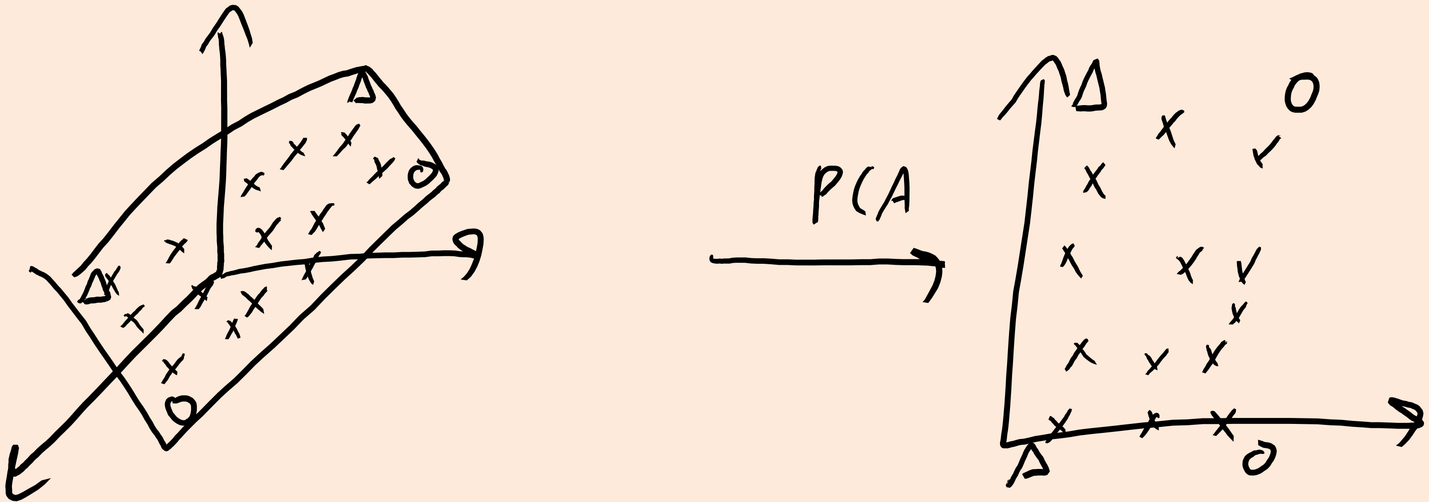
- ISOMAP is latent-factor model for visualizing data on manifolds:
  1. Find the neighbours of each point.
     - Usually "k-nearest neighbours graph", or "epsilon graph".
  2. Compute edge weights:
     - Usually distance between neighbours.
  3. Compute weighted shortest path between all points.
 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



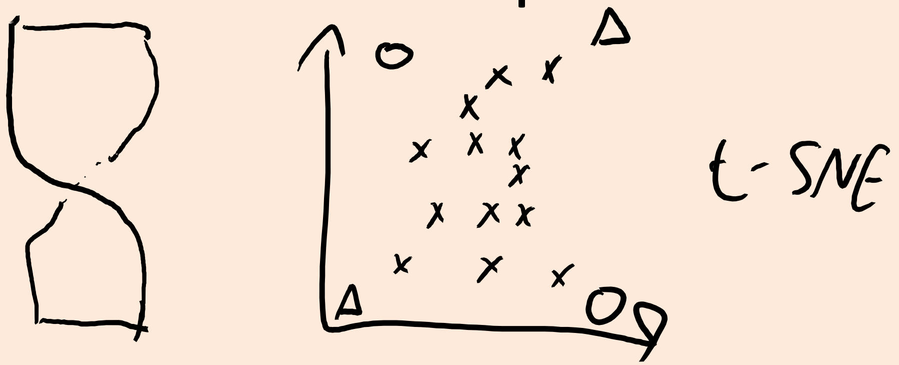
http://wearables.cc.gatech.edu/paper_of_week/isomap.pdf

Does t-SNE always outperform PCA?

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

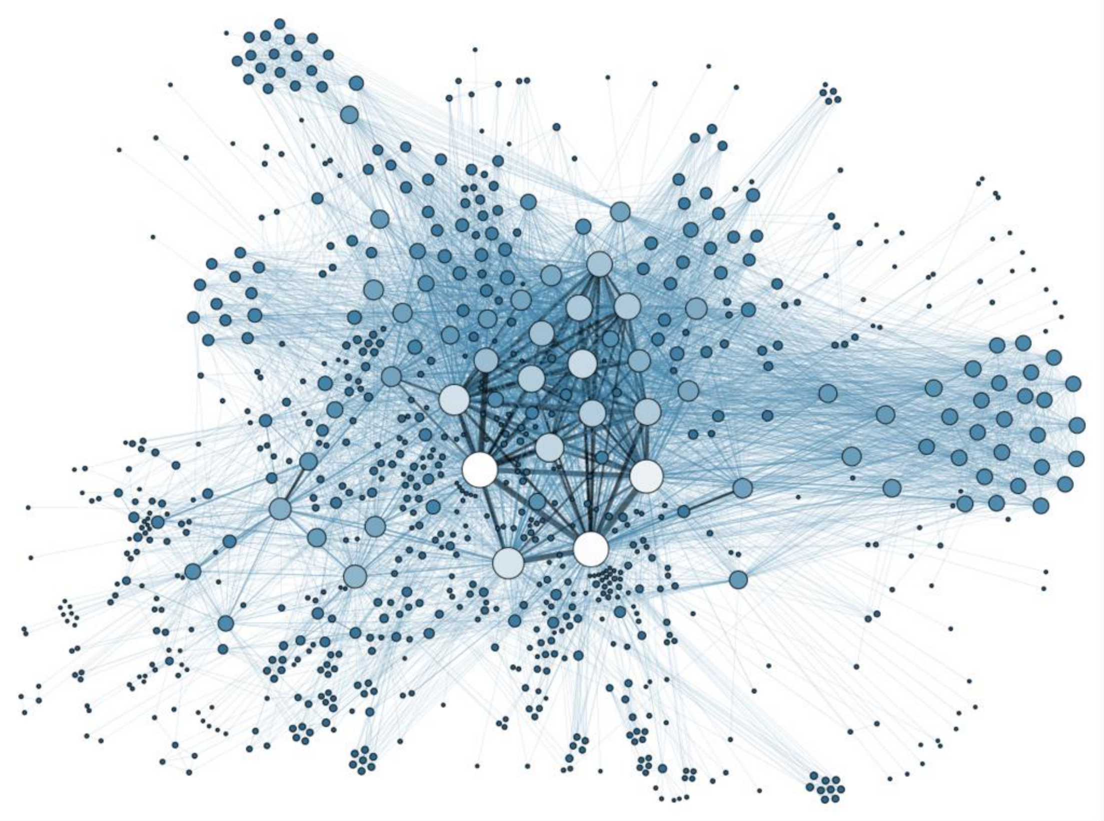


- PCA can perfectly capture the low-dimensional structure.
- T-SNE can capture the local structure, but can “twist” the plane.
  - It doesn’t try to get long distances correct.



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:

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

- 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:

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

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

$$\begin{tikzcd}
\text{with } g(w) = w^Tx_i - y_i \arrow[r] & \nabla g(w) = x_i \arrow[dr] & \\
 & & \nabla\left[f(g(w))\right] = r_i\;x_i \\
\text{and } f(r_i) = \tfrac{1}{2}r_i^2 \arrow[r] & f'(r_i) = r_i \arrow[ur] &
\end{tikzcd}$$

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

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

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

$\hookrightarrow$ Assuming $z_i \neq z_j$

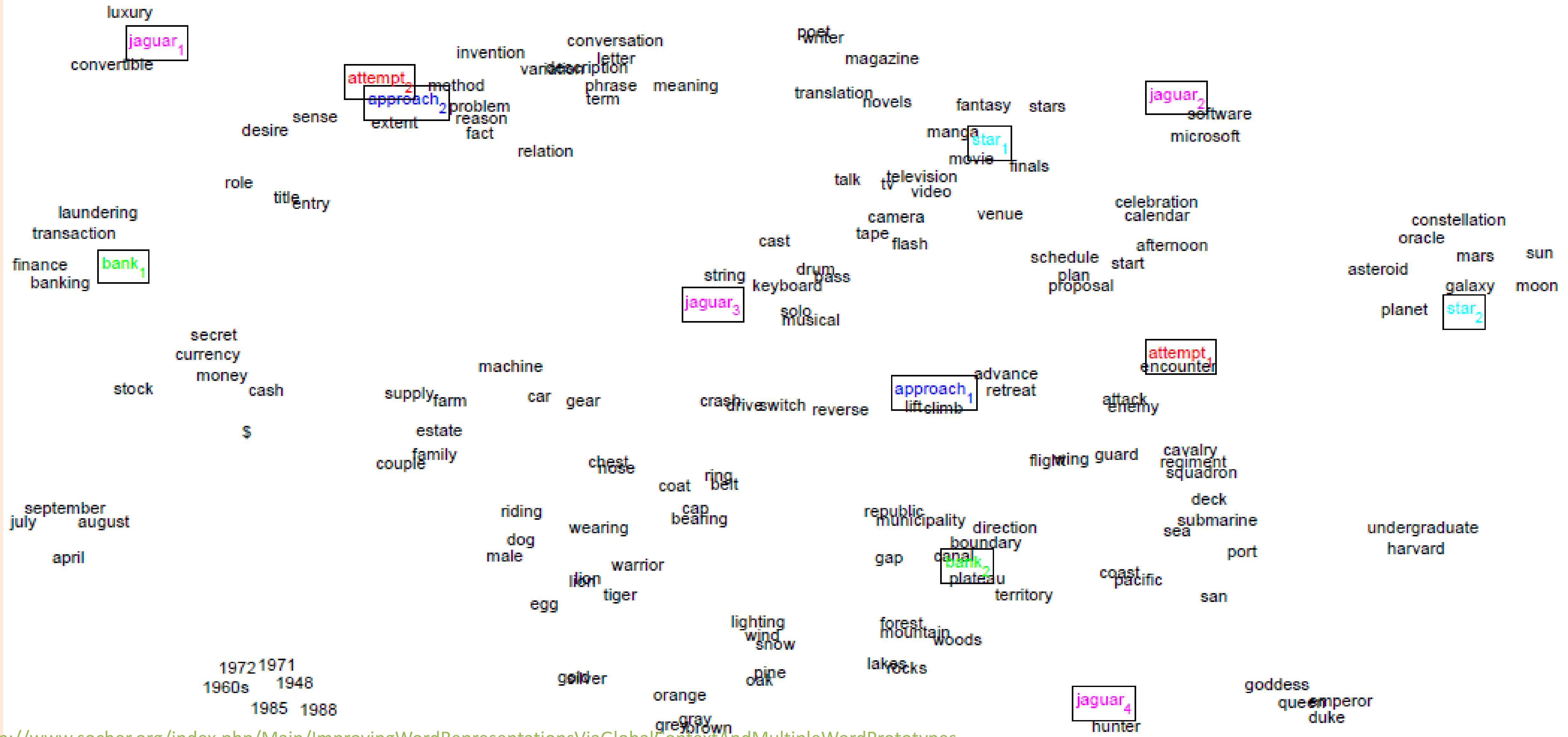
Multiple Word Prototypes

- What about homonyms and polysemy?
  - The word vectors would need to account for all meanings.

- More recent approaches:
  - Try to cluster the different contexts where words appear.
  - Use different vectors for different contexts.

$$X_{jaguar} = \left[\begin{array}{c} \cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots \\ \end{array}\right] \begin{array}{l} z_{j1} \\ z_{j2} \\ z_{j3} \end{array}$$

Multiple Word Prototypes



http://www.socher.org/index.php/Main/ImprovingWordRepresentationsViaGlobalContextAndMultipleWordPrototypes