

**MTTTS17**

**Dimensionality Reduction and  
Visualization**

**Spring 2020, 5cr  
Jaakko Peltonen**

**Lecture 10: Neighbor Embedding  
part 1**

# Part 1: Neighbor embedding

# Distance Preservation

Most dimensionality reduction methods discussed so far have been based on **preservation of distances**.

$$E_{\text{mMDS}} = \frac{1}{a} \sum_{ij} w_{ij} (d_{\mathcal{X}}(\mathbf{x}^i, \mathbf{x}^j) - d_{\mathcal{E}}(\xi^i, \xi^j))^2$$

Metric MDS stress function

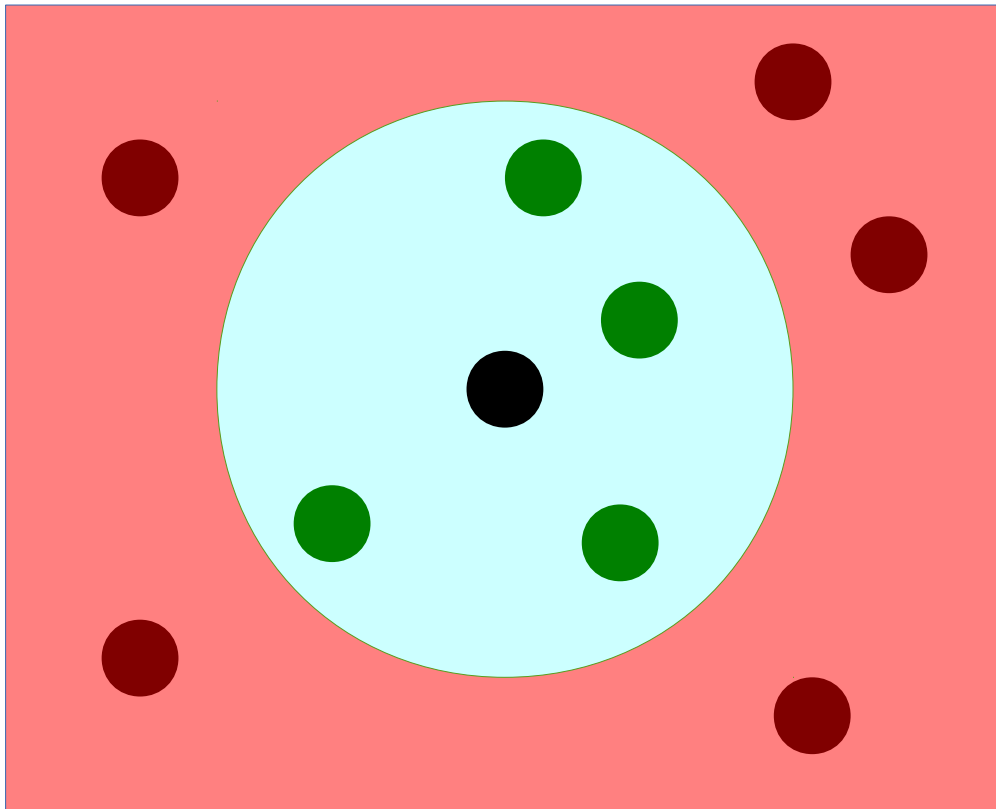
Other variants have modified which distances are most important to preserve: for example Sammon's mapping considers small distances the most important to preserve. But the aim is still to preserve distances.

Are distances the important thing to the analyst, if the aim is information visualization?

# Alternative Idea: Preserve Neighborhoods

**Neighbors** are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.

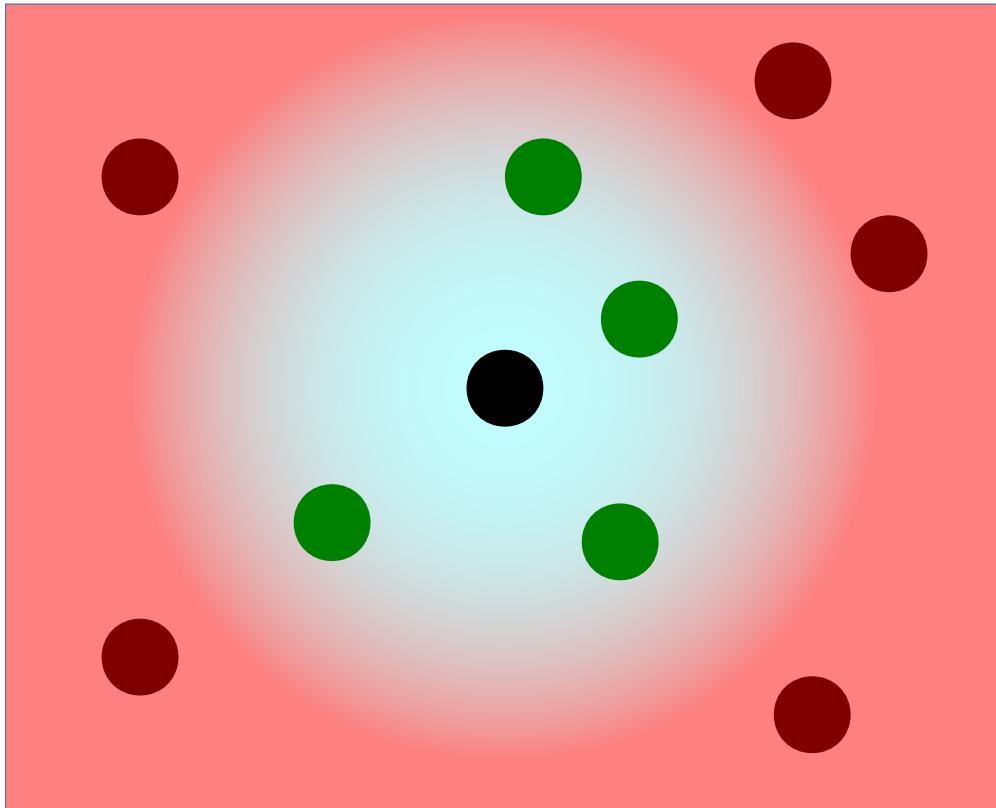


Hard neighborhood -  
each point is a **neighbor**  
or a **non-neighbor**

# Alternative Idea: Preserve Neighborhoods

**Neighbors** are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

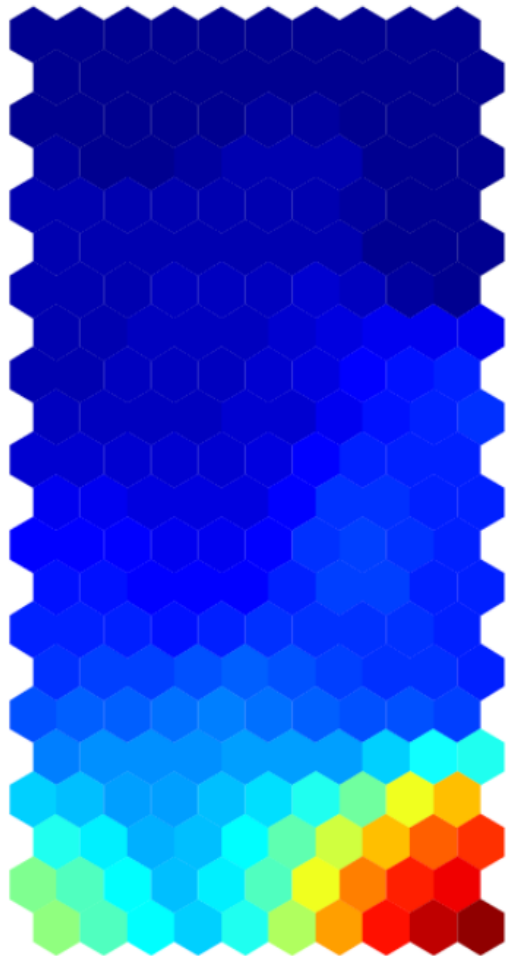
In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.



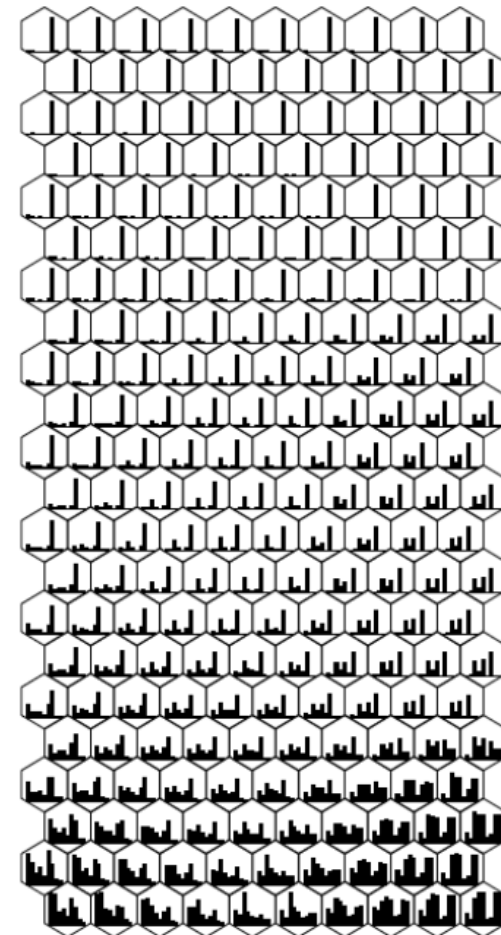
Soft neighborhood -  
each point is a **neighbor with some weight** and a **non-neighbor with some weight**

# Neighborhoods in the Self-Organizing Map

**The Self-Organizing Map** was discussed on an earlier lecture. It adapts a grid of prototype vectors, moving them closer to a set of data points. Soft neighborhoods between prototypes on the grid are used to adapt the grid in an organized fashion.



Values of the prototype vectors  
shown on the SOM grid.  
Left: one of the feature values.  
Right: many of the feature values.



From Juha Vesanto's  
doctoral thesis, 2002

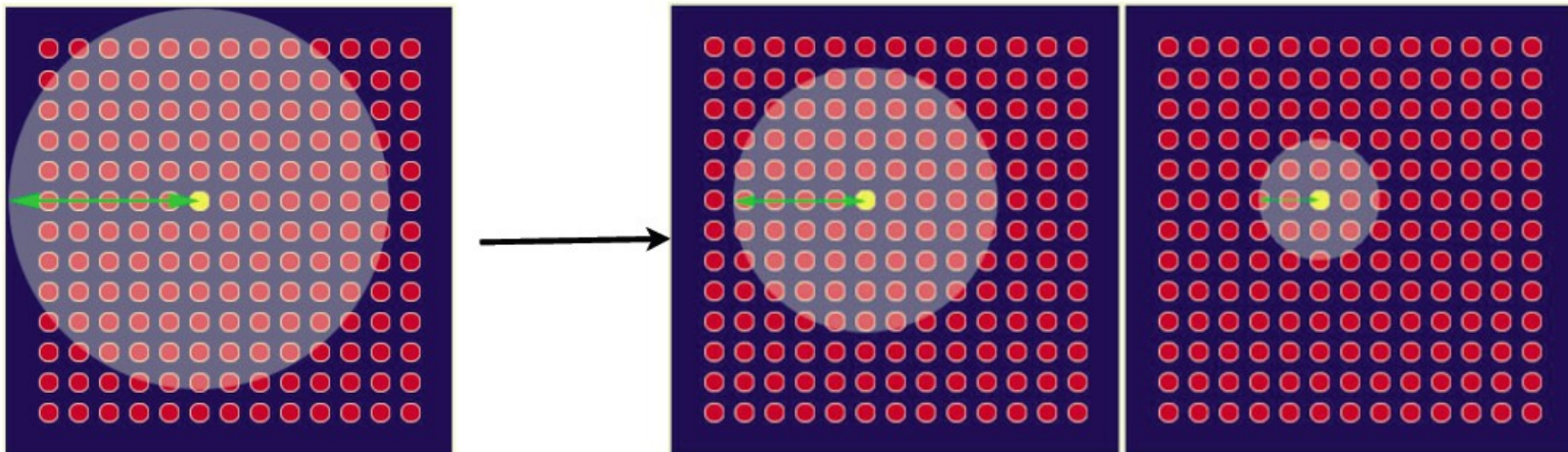
# Neighborhoods in the Self-Organizing Map

After random or PCA initialization of prototypes  $u$ , iterate:

- Pick a data point  $x$  from the training set
- Find the nearest prototype  $v$
- Update all prototypes  $u$  towards  $x$ , based on **neighborhood** of  $v$

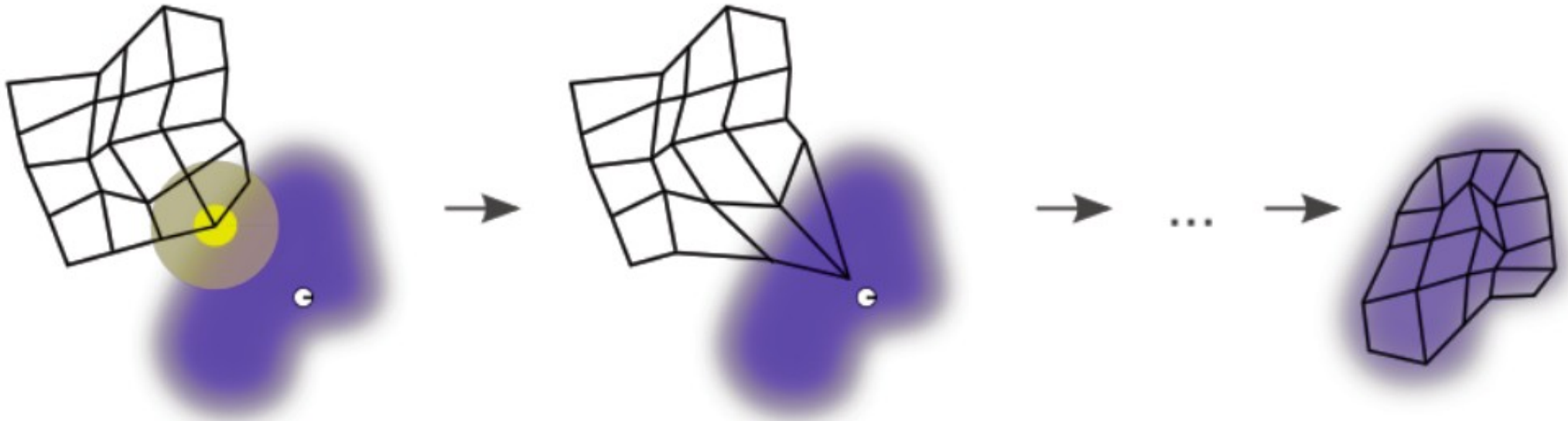
$$u(t+1) = u(t) + h_{u,v}(t)\alpha(t)(x(t) - u(t))$$

$$h_{u,v}(t) = \exp\left(\frac{-d_{grid}(u,v)}{\sigma(t)}\right) \quad \sigma(t) = \sigma_0 \exp\left(\frac{-t}{\lambda}\right)$$



Images  
from  
T-61.5010  
course  
slides

# Neighborhoods in the Self-Organizing Map



The SOM uses neighborhoods on the grid (on the output display), but does not use neighborhoods on the input space.

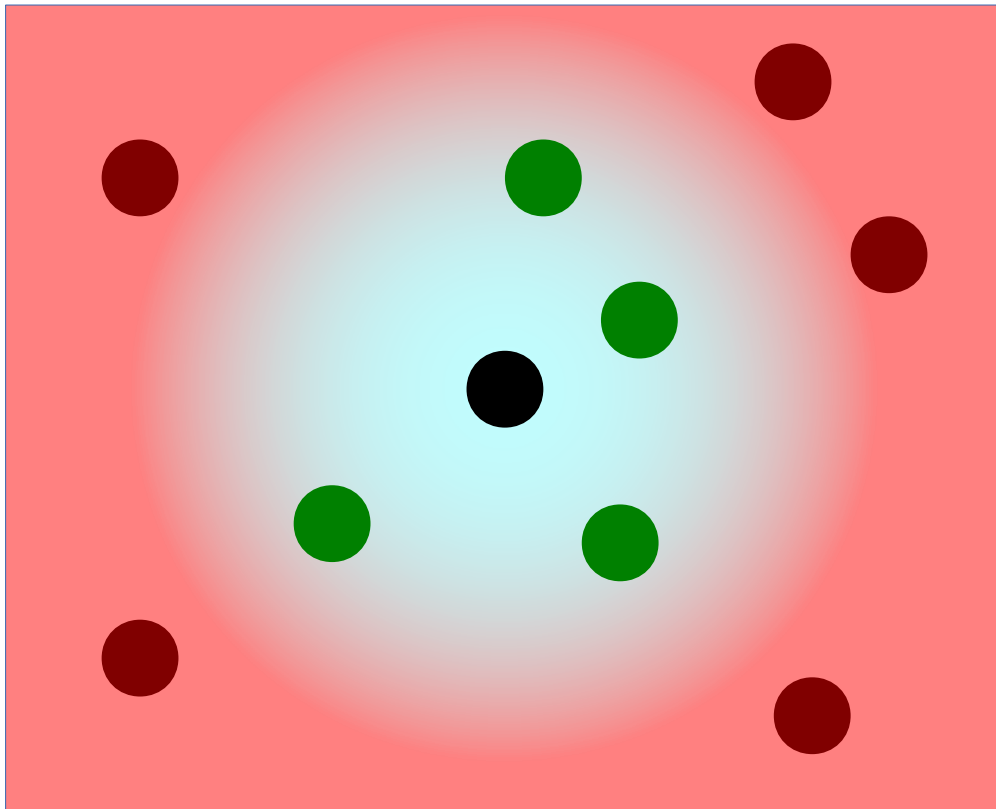
We will next discuss methods that involve both **input** and **output neighborhoods**.



# Alternative Idea: Preserve Neighborhoods

**Neighbors** are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.



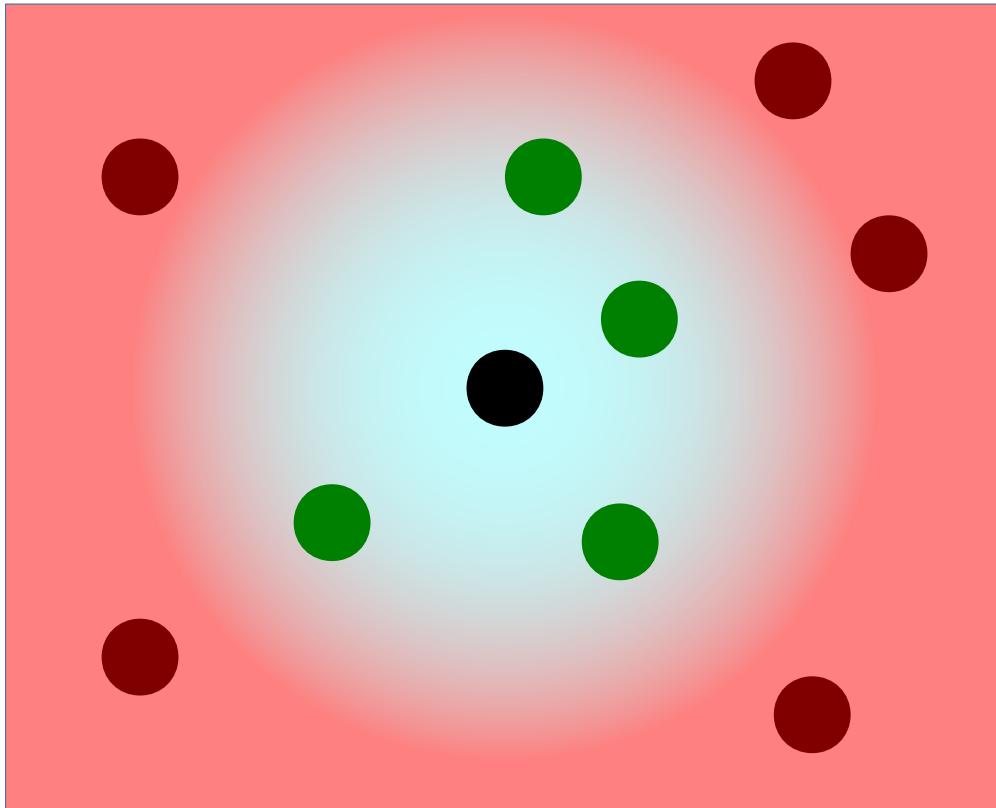
Soft neighborhood -  
each point is a **neighbor with some weight** and a **non-neighbor with some weight**

Some images and equations on the following slides are from the SNE paper (Roweis & Hinton '02).

# Alternative Idea: Preserve Neighborhoods

**Neighbors** are an important concept in many applications: neighboring cities, friends on social networks, followers of blogs, links between webpages.---> **Preserve neighbors** instead of distances?

In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.



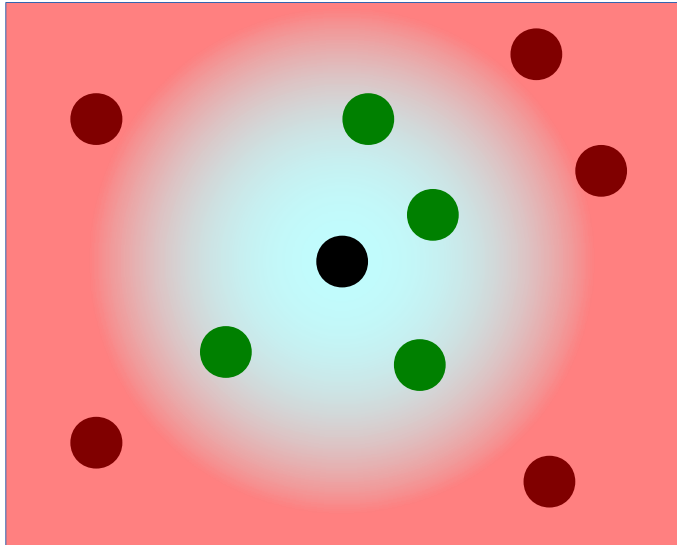
Probabilistic neighborhood

$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}$$

(probability to be picked as a neighbor in input space. Unlike in SOM, sums to 1)

# Alternative Idea: Preserve Neighborhoods

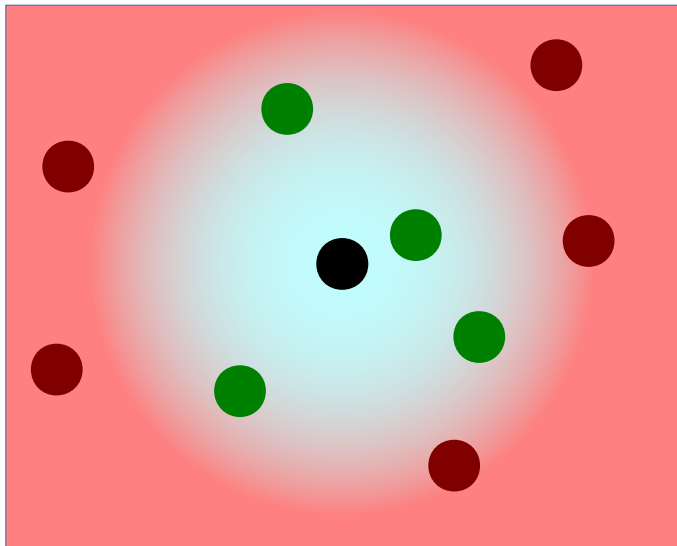
In vectorial data, if nothing else is known, it is reasonable that close-by points in some metric can be considered neighbors.



Probabilistic input neighborhood

$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}$$

(probability to be picked as a neighbor)



Probabilistic output neighborhood

$$q_{ij} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{y}_i - \mathbf{y}_k\|^2)}$$

(probability based on display coords.)

# Stochastic Neighbor Embedding

Two probability distributions over a set of items can be compared by the **Kullback-Leibler (KL) divergence** = relative entropy = amount of surprise when encountering items from the 1<sup>st</sup> distribution when items were expected to come from the 2<sup>nd</sup> .

Use KL divergence to compare neighborhoods between the input and the output!

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} = \sum_i KL(P_i || Q_i)$$

KL divergence is nonnegative, and zero if and only if the distributions are equal. The value of the divergence sum depends on output coordinates, and can be minimized with respect to them. This is **Stochastic Neighbor Embedding**.


# Stochastic Neighbor Embedding, details

How to set the size of the input neighborhood?

- controlled by a scale parameter in the input distance

$$d_{ij}^2 = \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}$$

scale parameter



- a scale parameter is hard to set without knowing a lot about the original data features
- it's beneficial to set the scale parameter differently for each individual point: in a sparsely populated region, neighborhood probability should maybe decrease less rapidly?

Idea: set an “**effective number of neighbors**”. In a uniform distribution over  $k$  neighbors, the entropy is  $\log(k)$ . Find the scale parameter value (by binary search in some interval) so that the entropy of  $p_{ij}$  becomes  $\log(k)$  for a desired value of  $k$ . This value becomes different around each point.

# Stochastic Neighbor Embedding, details

Adjusting the output coordinates is done by gradient descent to minimize the sum of KL divergences. Start from a random initial output configuration, then iteratively take steps along the gradient.

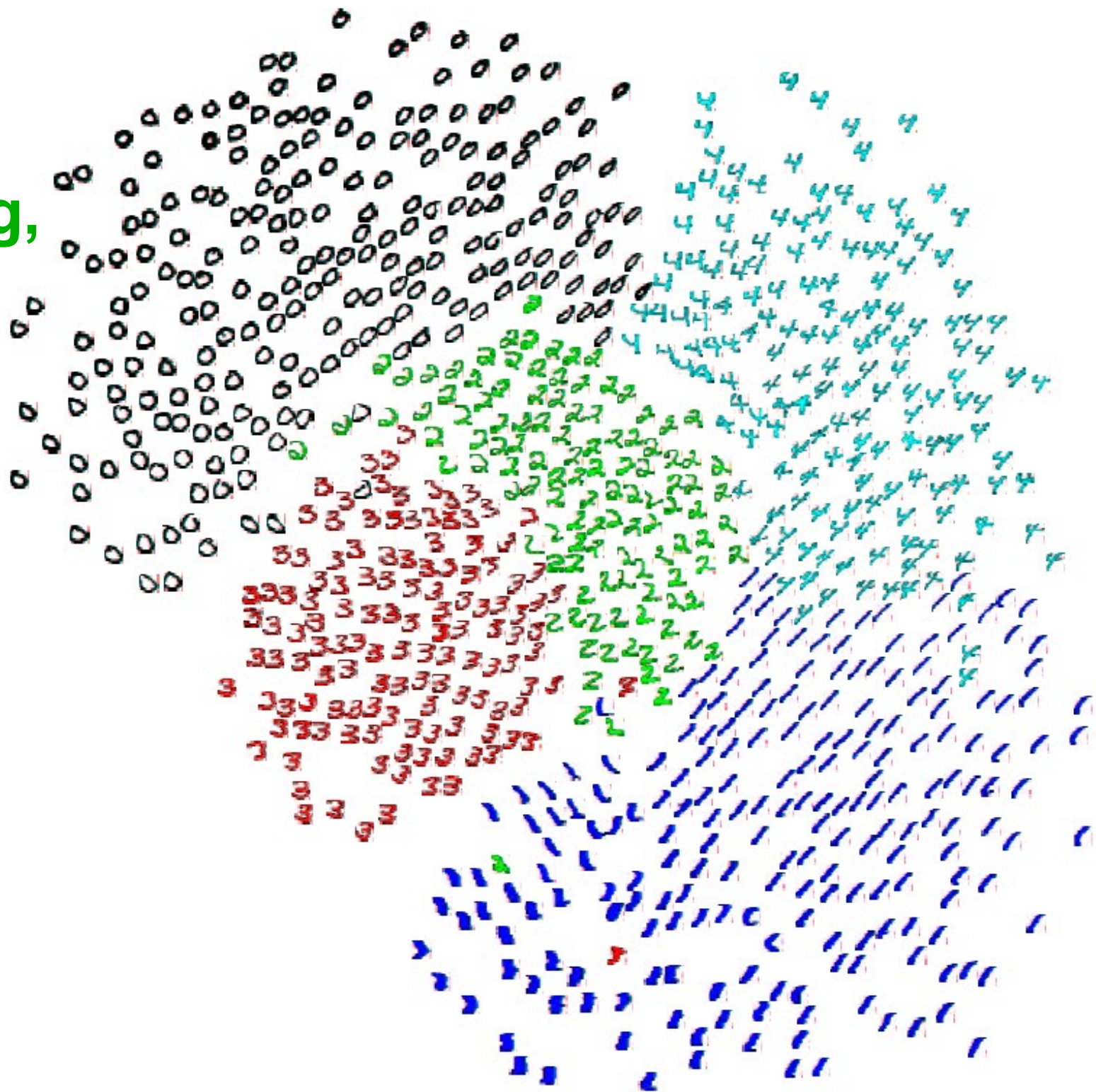
$$\frac{\partial C}{\partial \mathbf{y}_i} = 2 \sum_j (\mathbf{y}_i - \mathbf{y}_j) (p_{ij} - q_{ij} + p_{ji} - q_{ji})$$

Can be seen as “forces” pushing and pulling between pairs of points to make the input and output probabilities more similar.

# Stochastic Neighbor Embedding, example

SNE applied to grayscale bitmap images of handwritten digits.

Features = pixel values.

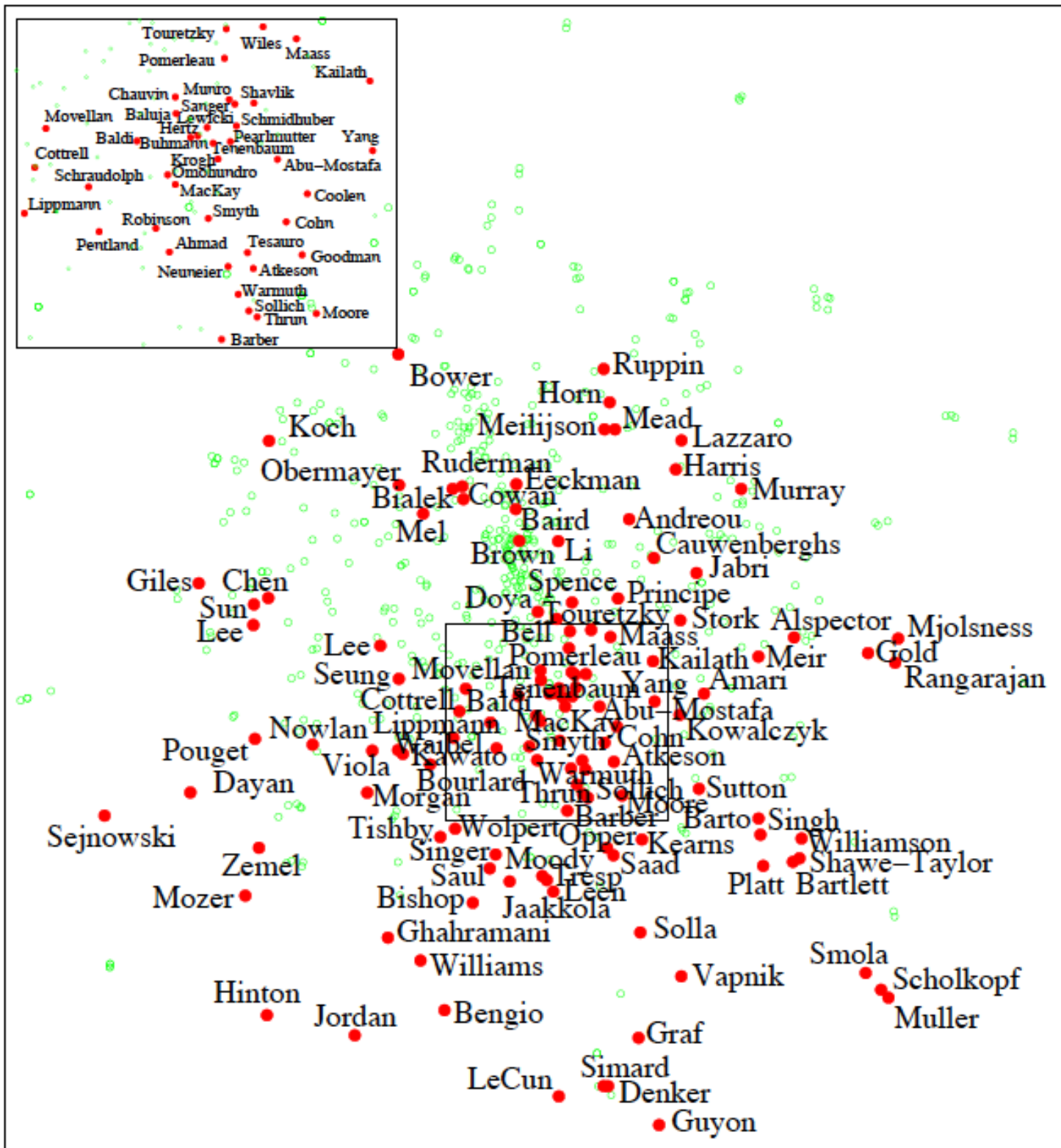


# Stochastic Neighbor Embedding, NIPS authors

Authors of NIPS papers

Features = vectors of word counts

(how many of each word does an author have in his/her NIPS papers)





# Stochastic Neighbor Embedding with multiple output locations for each data item?

An interesting alternative of SNE is possible, where each data item has **multiple locations on the display**.

$$q_{ij} = \sum_b \pi_{i_b} \sum_c \frac{\pi_{j_c} \exp(-\|\mathbf{y}_{i_b} - \mathbf{y}_{j_c}\|^2)}{\sum_k \sum_d \pi_{k_d} \exp(-\|\mathbf{y}_{i_b} - \mathbf{y}_{k_d}\|^2)}$$

Occupancy weight of i at b:th location

Perhaps useful if some relationships are hard to capture on-screen with just one location per item.

# Crowding Problem

When the output dimensionality is smaller than the effective dimensionality of data on the input, the **neighborhoods are mismatched**:

- in a high-dimensional space, points can have many close-by neighbors in different directions. In a 2D space, you essentially have to arrange close-by neighbors in a circle around the central point, which constrains relationships among neighbors.
- in a high-dimensional space you can have many points that are equidistant from one another; in a 2D space at most 3 points are equidistant from each other.
- volume of a sphere scales as  $r^d$  in  $d$  dimensions

Some images and equations on the following slides are from the t-SNE paper (van der Maaten & Hinton '08).

# Crowding Problem

-----> On a 2D display, there is much less area available at radius  $r$  than the corresponding volume in the original space.

-----> Arranging high-dimensional items into a 2D space can easily place items more far off than they should be, because “there is no room left nearby”.

-----> In contrast, some items end up crowded in the center to stay close to all of the far-off points. This is the **crowding problem**.

# t-distributed Stochastic Neighbor Embedding

Idea: avoid crowding phenomenon by using a more **heavy-tailed neighborhood distribution** in the low-dimensional output space than in the input space. Neighborhood probability falls off less rapidly ----> less need to push some points far-off and crowd remaining points close together in the center.

Use student-t distribution with 1 degree of freedom = Cauchy distribution = infinite mixture of Gaussians.

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$

(Note these are joint probabilities, not conditional probabilities like in SNE)

Numerator approaches an inverse-square law---> joint probabilities for far-off points almost invariant to map scale

# t-distributed Stochastic Neighbor Embedding

$$P_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} \quad P_{ij} = \frac{P_{j|i} + P_{i|j}}{2n}$$

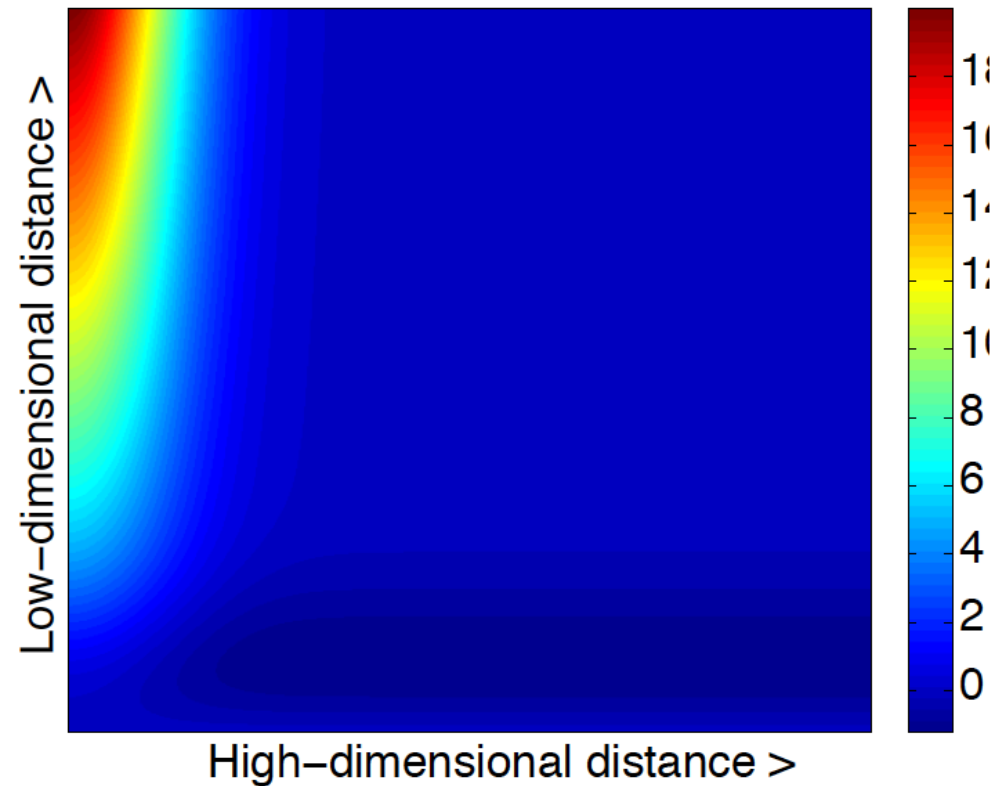
$$Q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}$$

Minimize divergence between symmetric probabilities

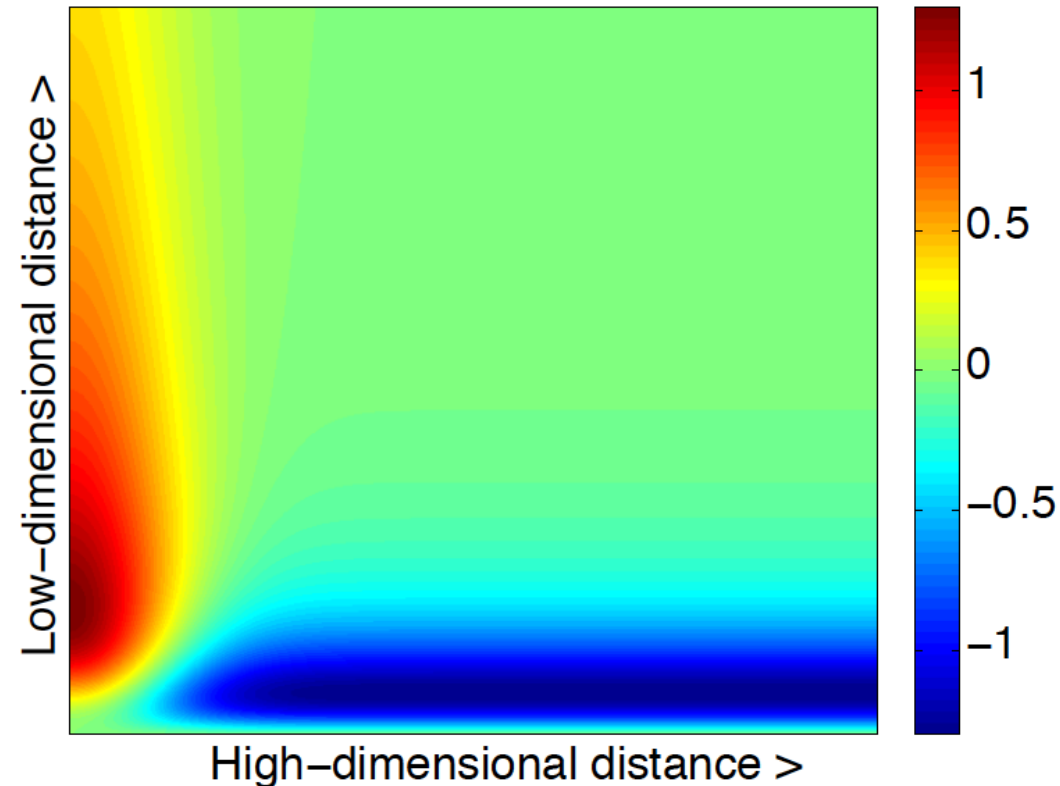
$$C = KL(P||Q) = \sum_i \sum_j P_{ij} \log \frac{P_{ij}}{Q_{ij}}$$

# t-distributed Stochastic Neighbor Embedding

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) (1 + \|y_i - y_j\|^2)^{-1}$$



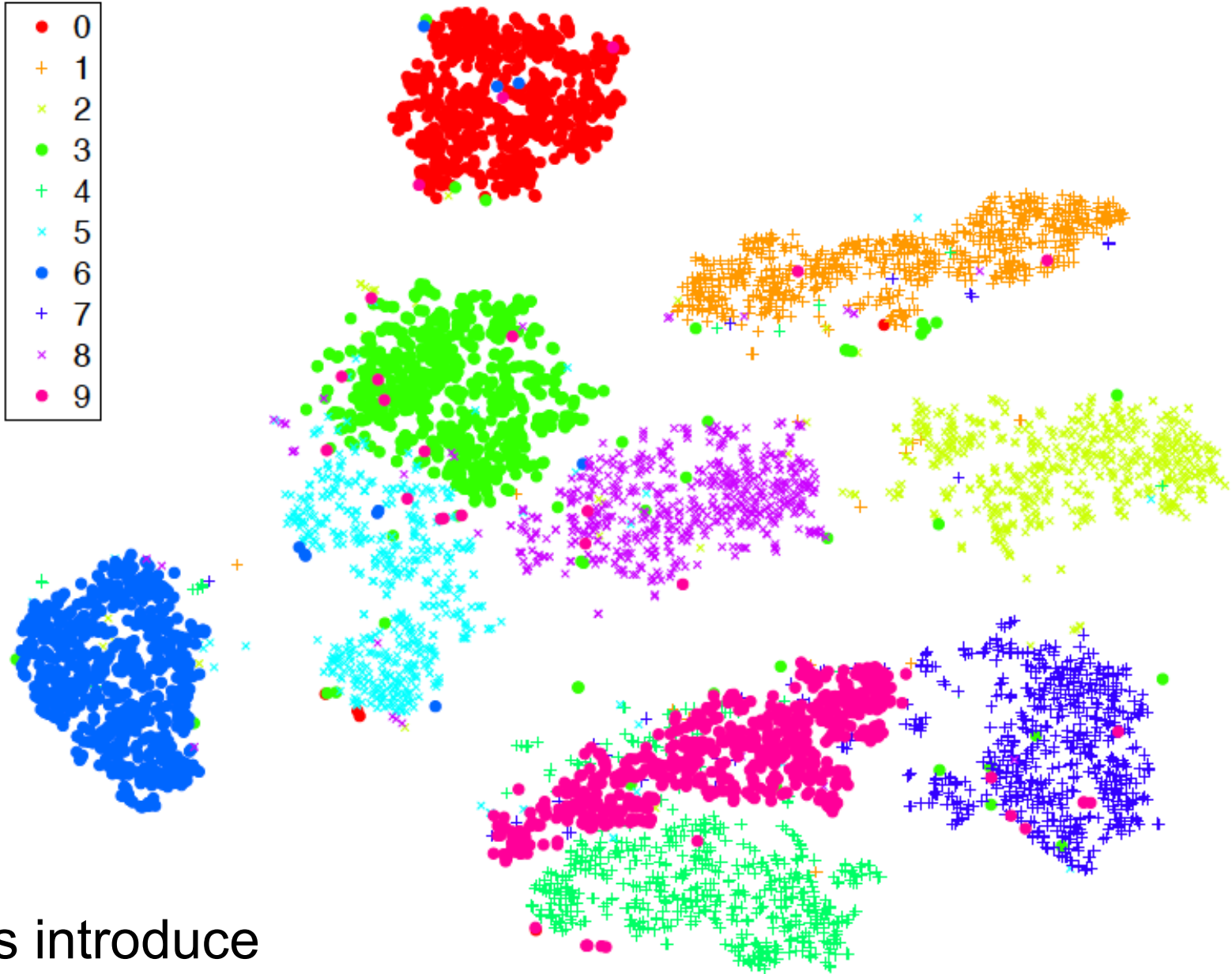
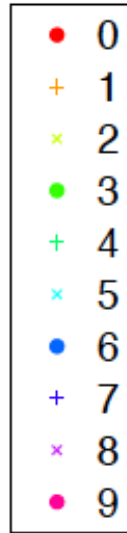
SNE: large gradient when high-dim distance is small and low-dim distance is large



t-SNE: positive gradient when high-dim dist. is small, low-dim is high, negative when other way around

# t-distributed Stochastic Neighbor Embedding

t-SNE on  
digit data



Personal  
observation:  
can sometimes introduce  
“phantom subclusters”  
where no subcluster-division exists

# t-distributed Stochastic Neighbor Embedding with output locations in multiple maps

An interesting alternative of t-SNE is possible, where each data item has is **projected to multiple displays**.

$$q_{ij} = \frac{\sum_m \pi_i^{(m)} \pi_j^{(m)} (1 + \|\mathbf{y}_i^{(m)} - \mathbf{y}_j^{(m)}\|^2)^{-1}}{\sum_k \sum_{l \neq k} \sum_{m'} \pi_k^{(m')} \pi_l^{(m')} (1 + \|\mathbf{y}_k^{(m')} - \mathbf{y}_l^{(m')}\|^2)^{-1}}$$

Occupancy weight of i at k:th display

Claimed useful for 1) visualizing original data similarities that do not satisfy the triangle inequality, 2) visualizing objects with high centrality (they are the closest neighbors of a lot of other points)



Part 2: Can we create “optimal” visualizations in some sense?

# Preservation Optimal for a Task?

What is a “good visualization”, is it the “nicest looking one”?  
Aesthetic considerations etc. are subjective ---> different  
visualization methods best for different analysts?

Algorithmic approaches to preserve various things can be seen  
as guesswork about what will produce the most useful  
visualization for an analyst.

-----> Useful for what?

-----> For doing something (a task)?

Purpose of visualization (one possible definition): to generate  
insights about the data in the mind of the analyst.

----> hard to quantify what works best for this (many kinds of  
possible “insights”, how to reach them depends on the “way of  
thinking” of each analyst)

-----> instead of “finding insight”, is there some simpler task that  
we could make visualizations for?

# Preservation Optimal for a Task?

It is reasonable to assume that in scientific visualization, the analyst wants to achieve insight by analyzing something about the original data, based on what is visible on the display.

-----> That is, analyzing some aspect of the data is a subtask of generating insight.

-----> Then preservation approaches should focus on preserving the thing that the analyst wants to analyze.

Preservation of distances is good if the analyst wants to measure distances between data points. But is that often a task the analyst does?

-----> Maybe in some applications (map projections!)

-----> But in general nonlinear dimensionality reduction, output axes don't have a simple meaning  
--->distances are less informative.

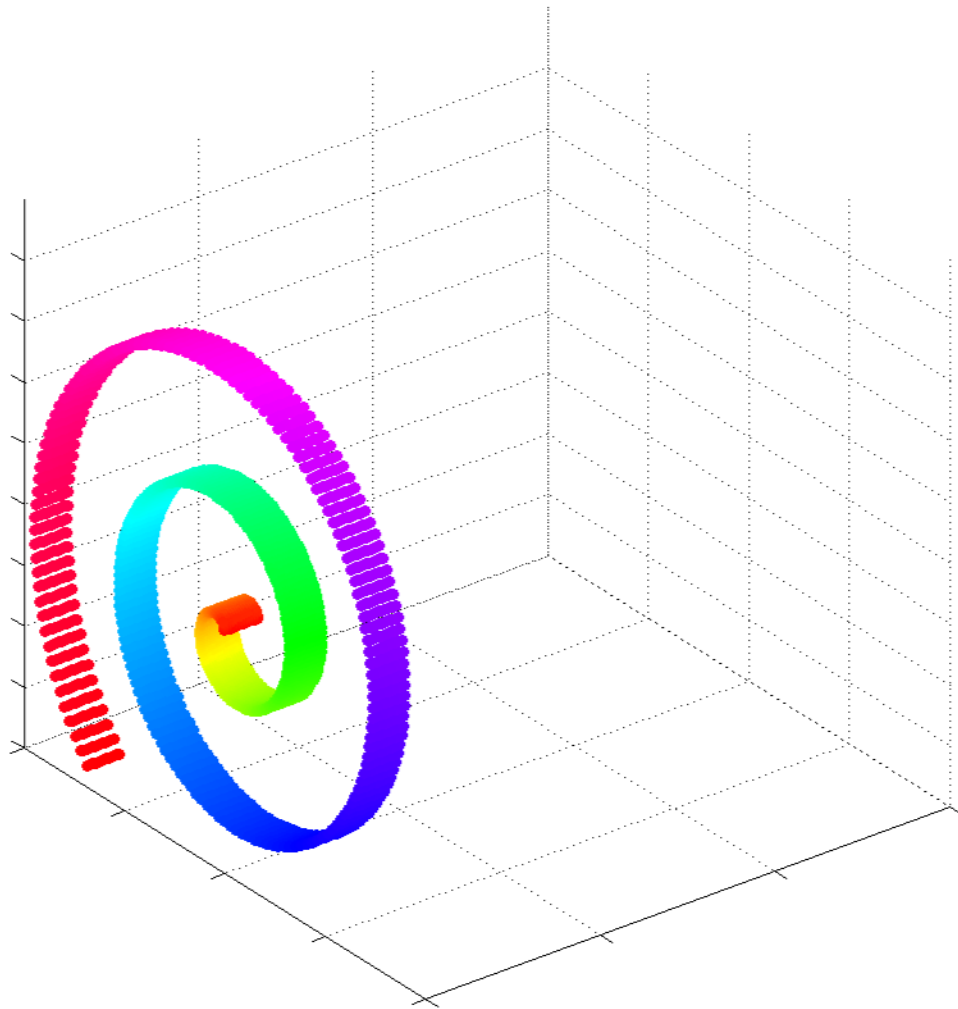
# Manifold Learning vs. Visualization

Many dimensionality reduction approaches using distances are based on the concept of **manifold learning**: the high-dimensional data is assumed to lie on a lower-dimensional “sheet” folded into a complicated shape in the high-dimensional space.

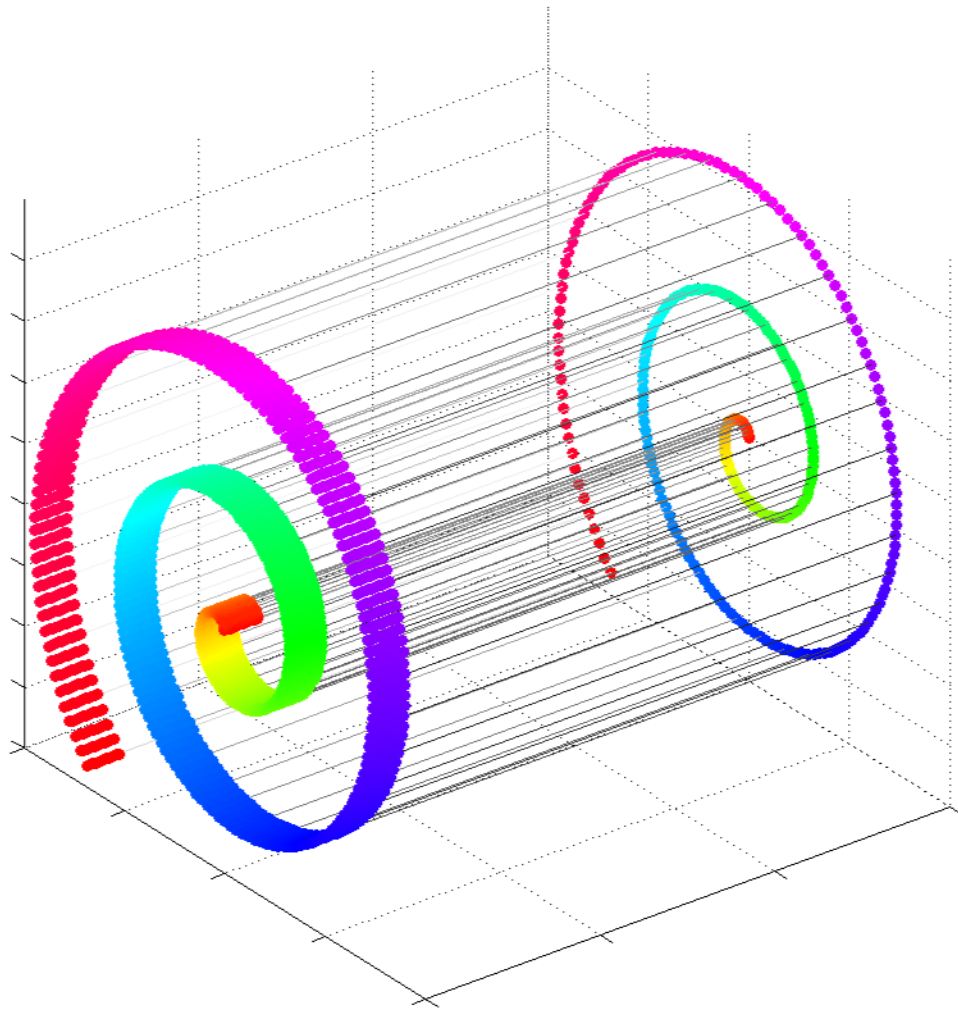
The idea is: if we know the dimensionality of the underlying manifold, then using that as the output dimensionality of nonlinear dimensionality reduction can “recover” the manifold.

-----> Benchmark tests with artificial manifolds  
(swiss roll, loops, s-curve, etc.)

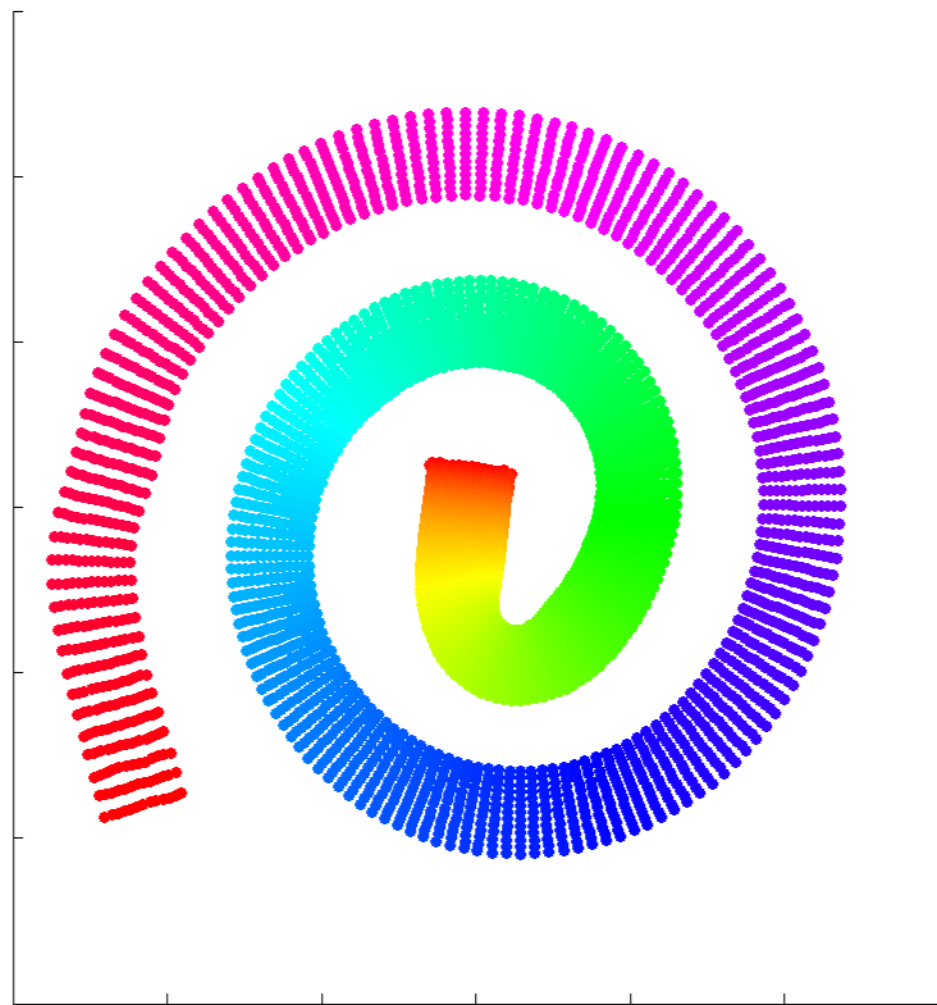
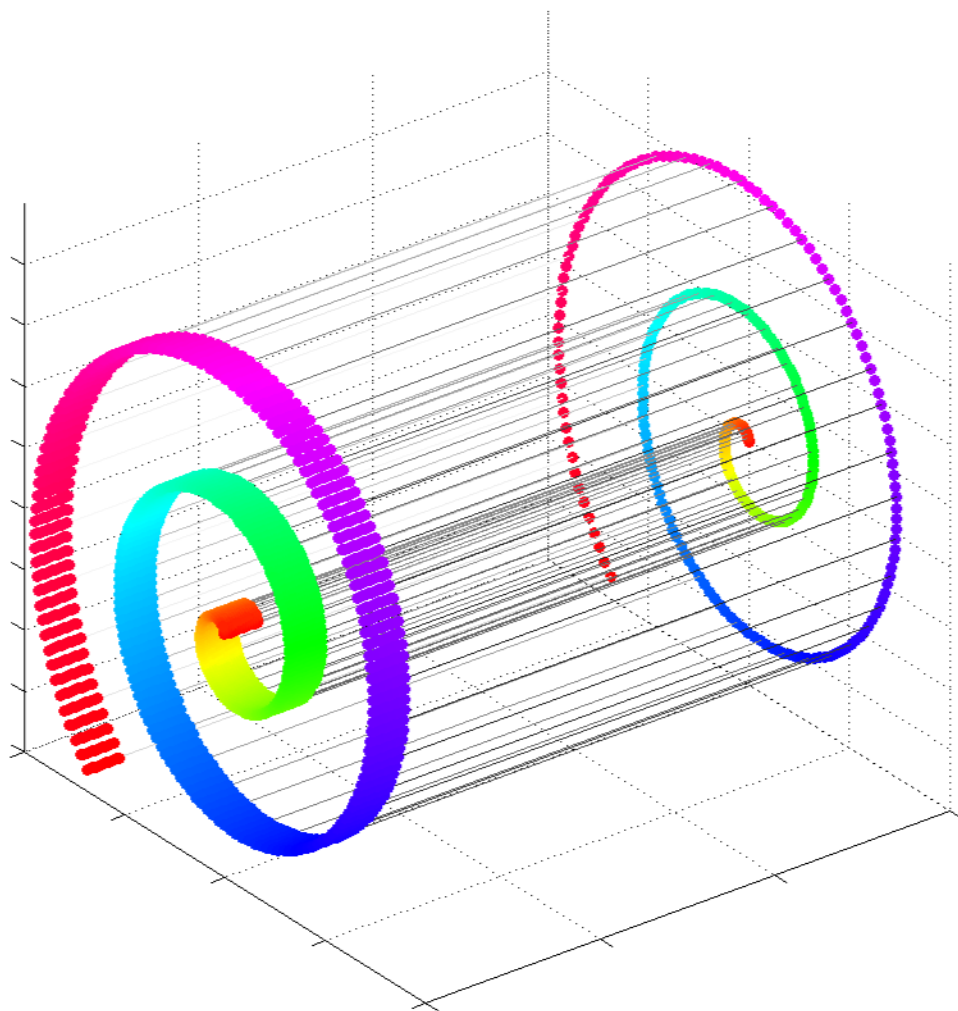
# Manifold learning on the most common artificial data set, “swiss roll”



# Manifold learning on the most common artificial data set, “swiss roll”



# Manifold learning on the most common artificial data set, “swiss roll”



# Manifold Learning vs. Visualization

Many dimensionality reduction approaches using distances are based on the concept of **manifold learning**: the high-dimensional data is assumed to lie on a lower-dimensional “sheet” folded into a complicated shape in the high-dimensional space.

The idea is: if we know the dimensionality of the underlying manifold, then using that as the output dimensionality of nonlinear dimensionality reduction can “recover” the manifold.

-----> Benchmark tests with artificial manifolds  
(swiss roll, loops, s-curve, etc.)

Even if manifolds exist in real-life data sets, their dimensionality may be too high to be visualized. -----> The manifold learning assumption can be ill-suited for visualization.

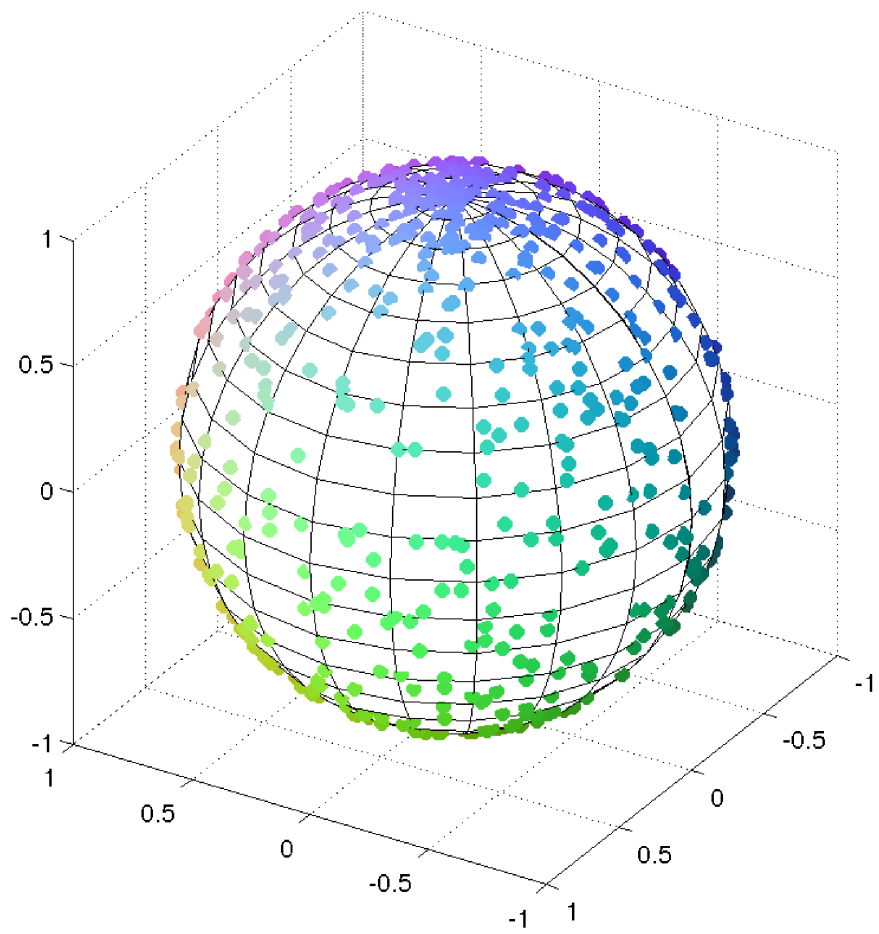
Good methods should prepare for inevitable losses (modeling!).



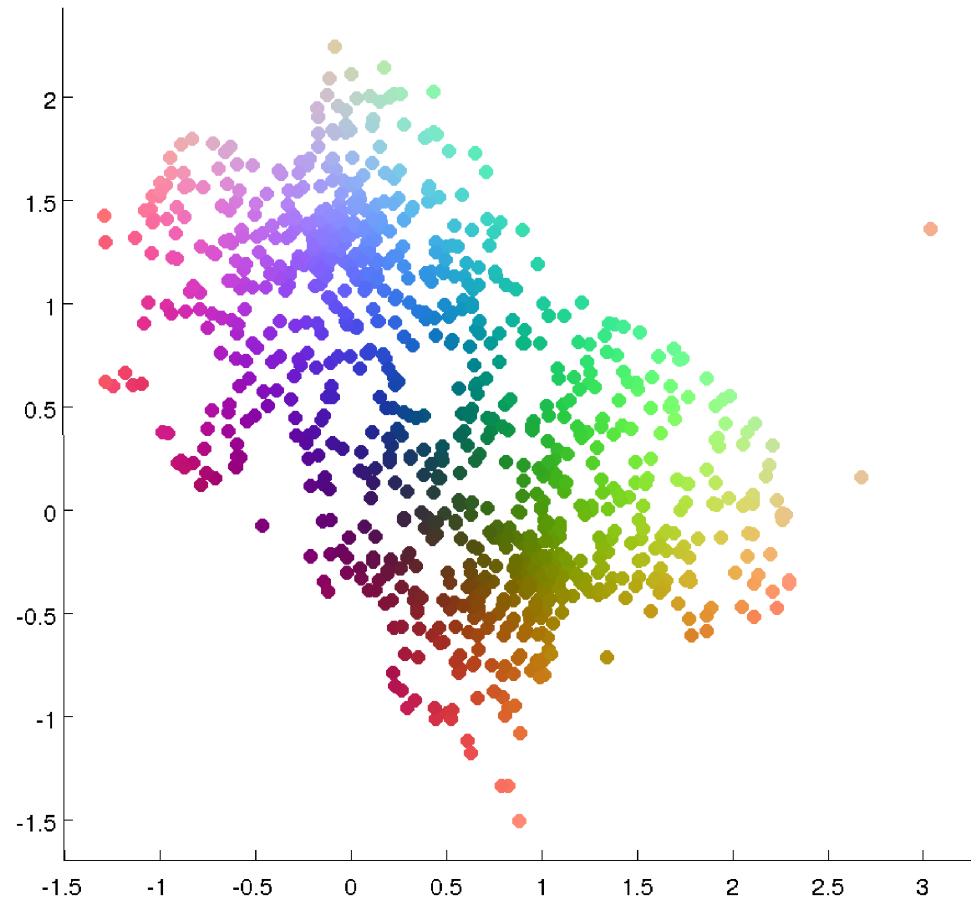
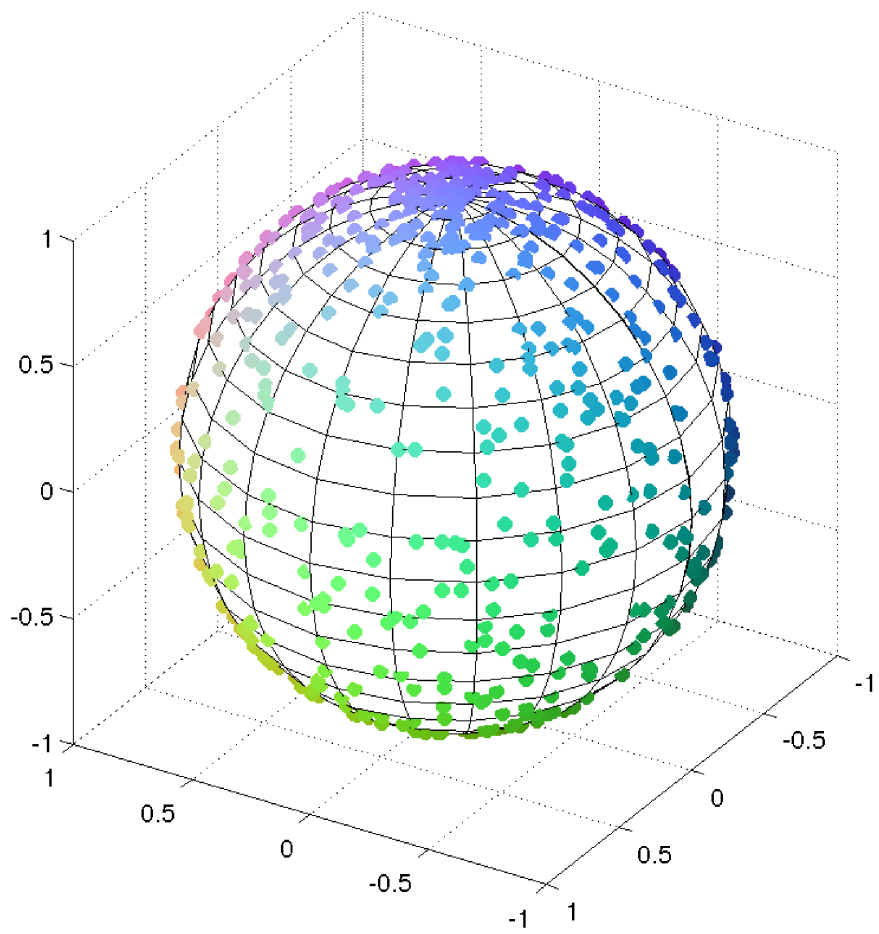
# Preservation of Neighborhoods for a Task?

Analyzing local neighborhoods of data items might be a subtask that some analysts perform to gain higher-level insight: for example, high-level structure of a graph (hubs, outlier areas) is built out of the set of local neighborhoods.

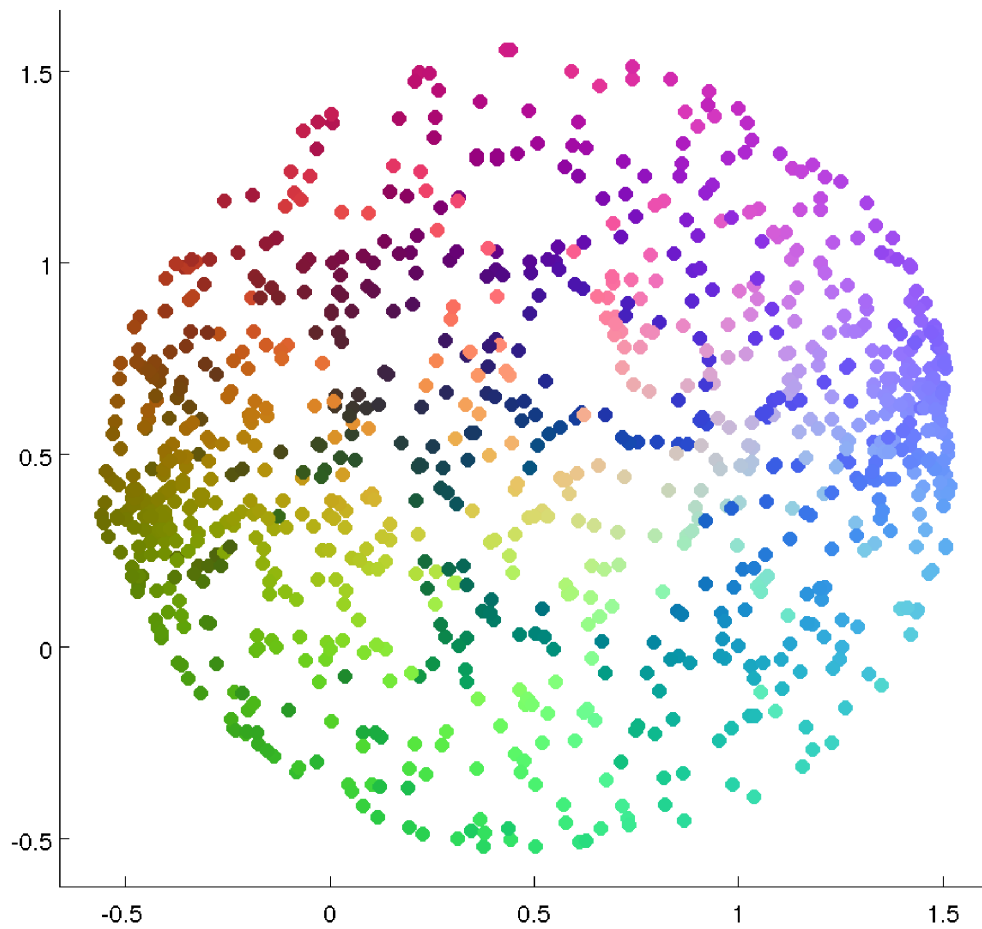
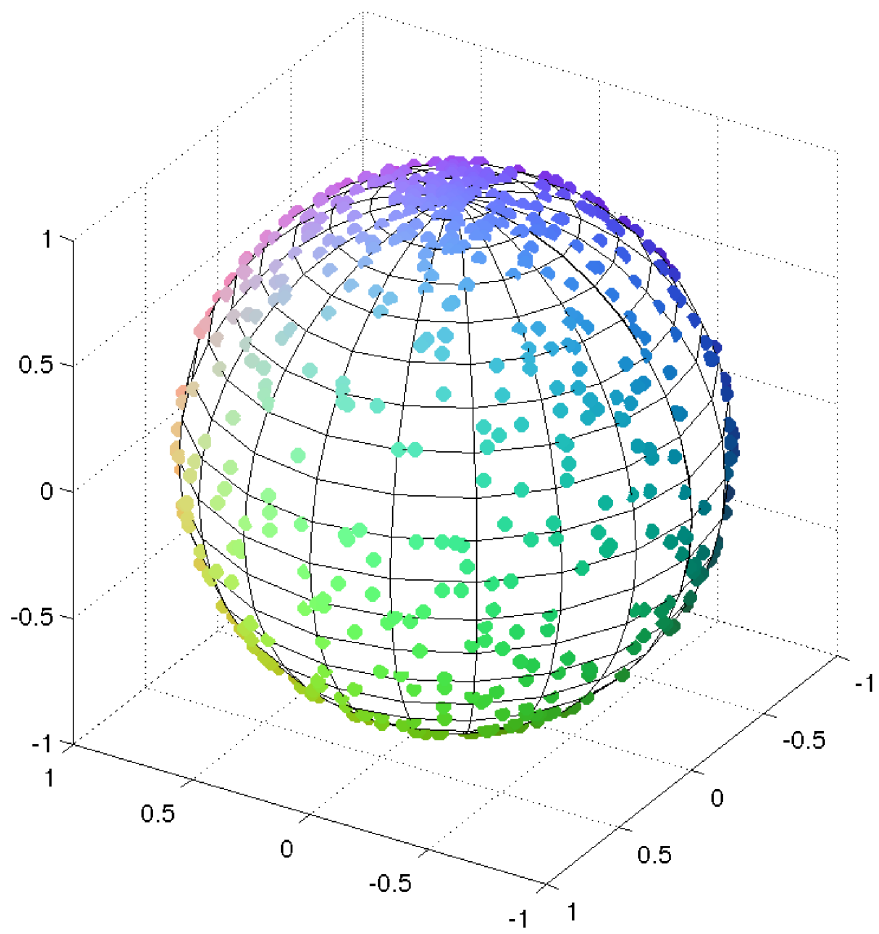
It turns out that preservation of neighborhoods can be formulated as **optimization of an information retrieval task**.



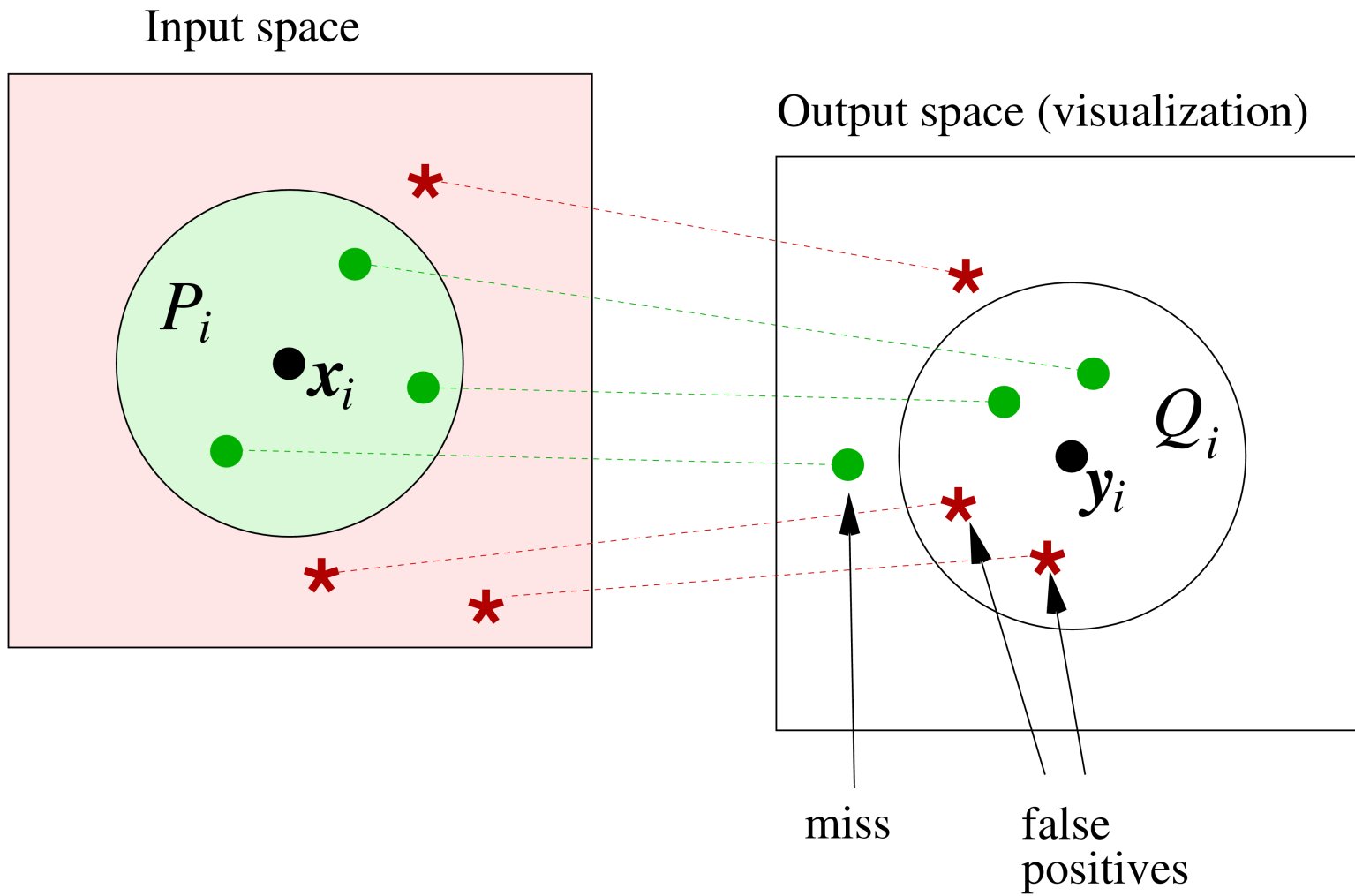
Example data set



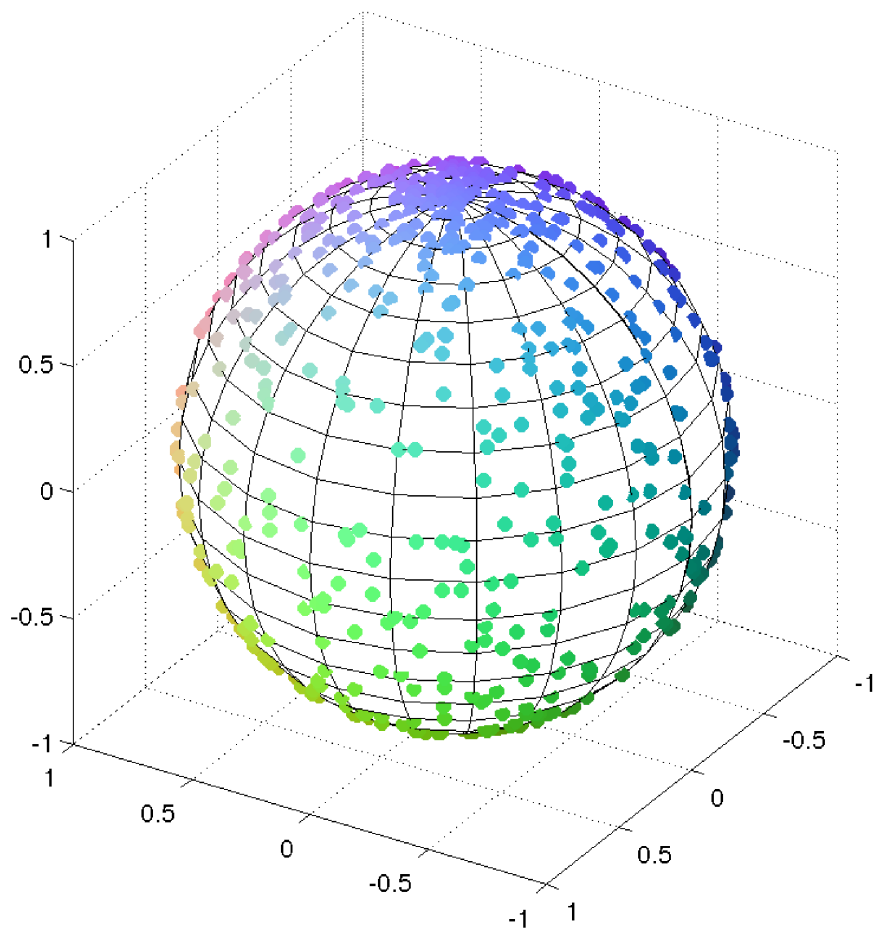
“Orange-peel map”



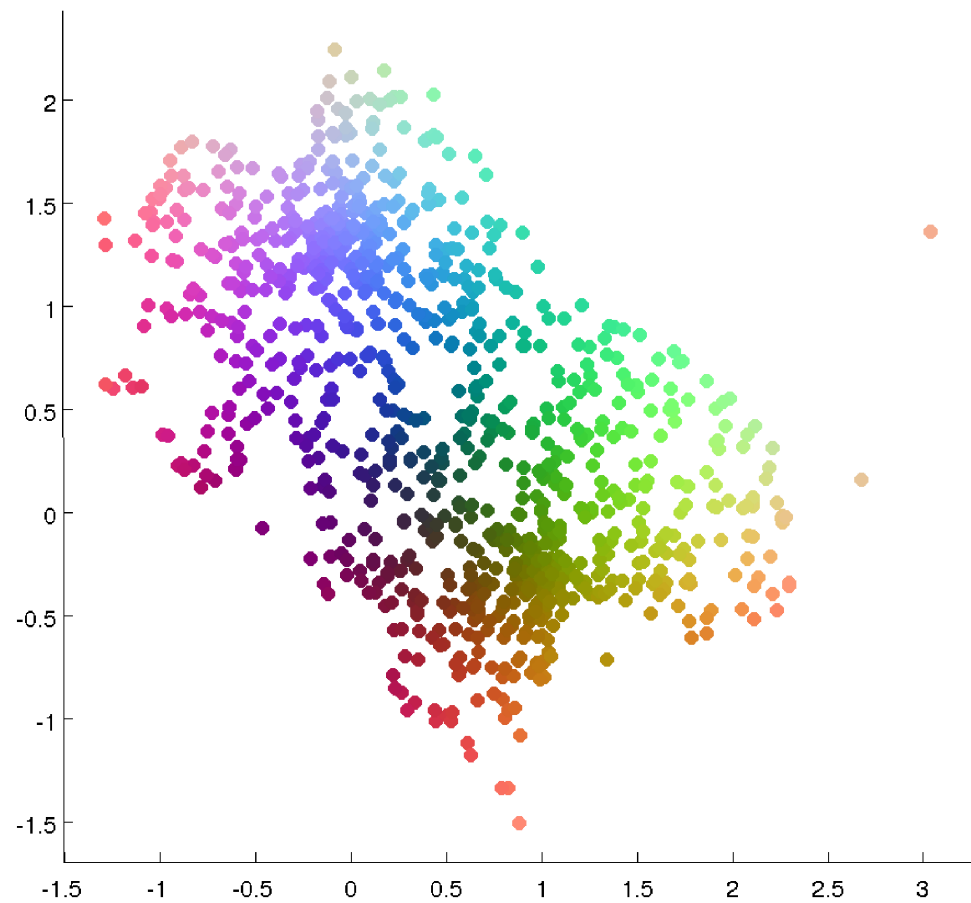
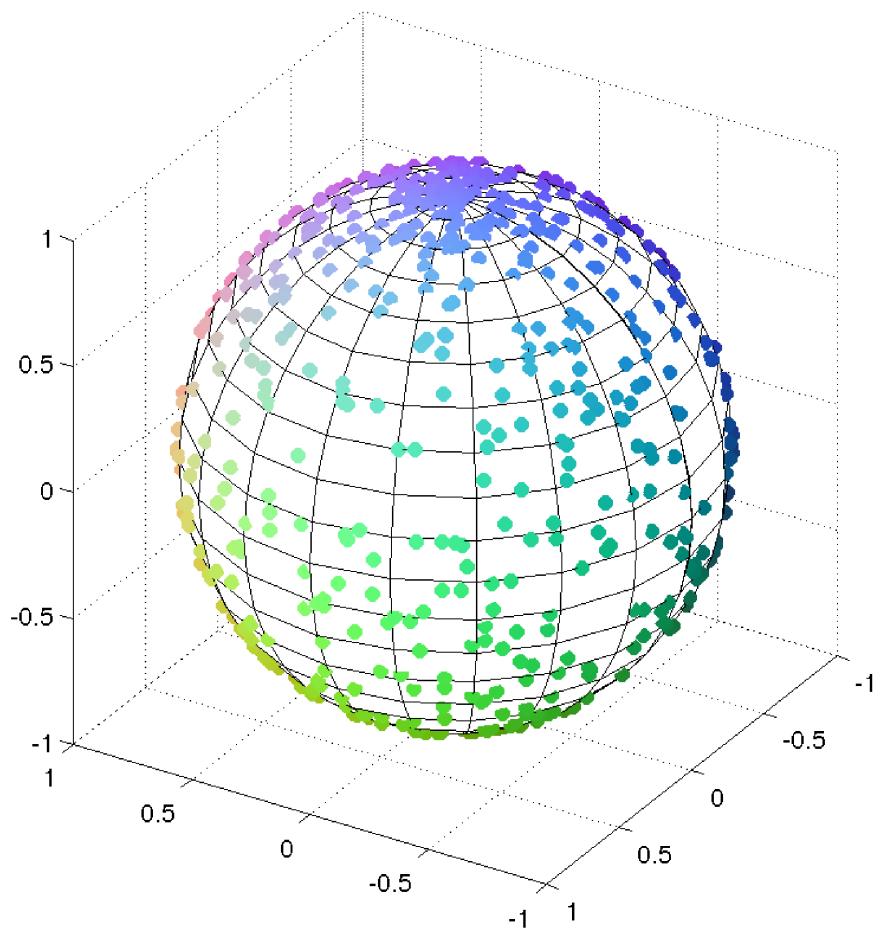
“Squashed-flat sphere”



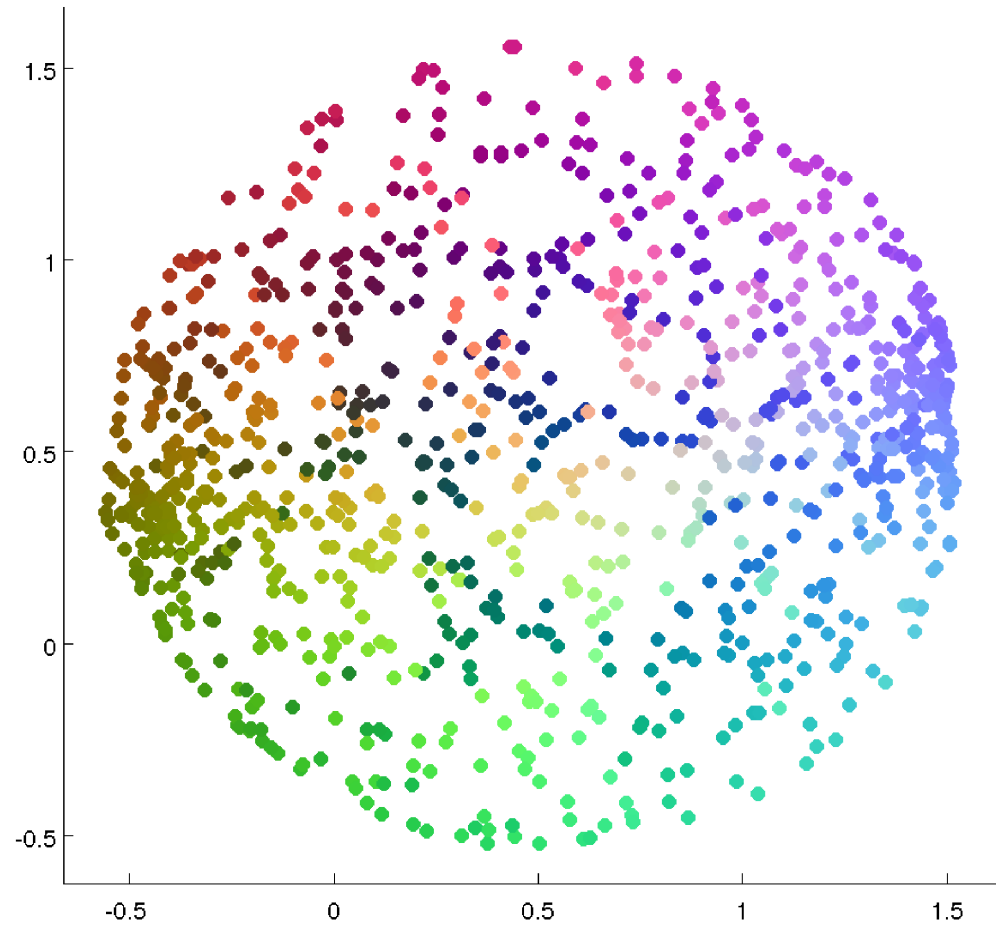
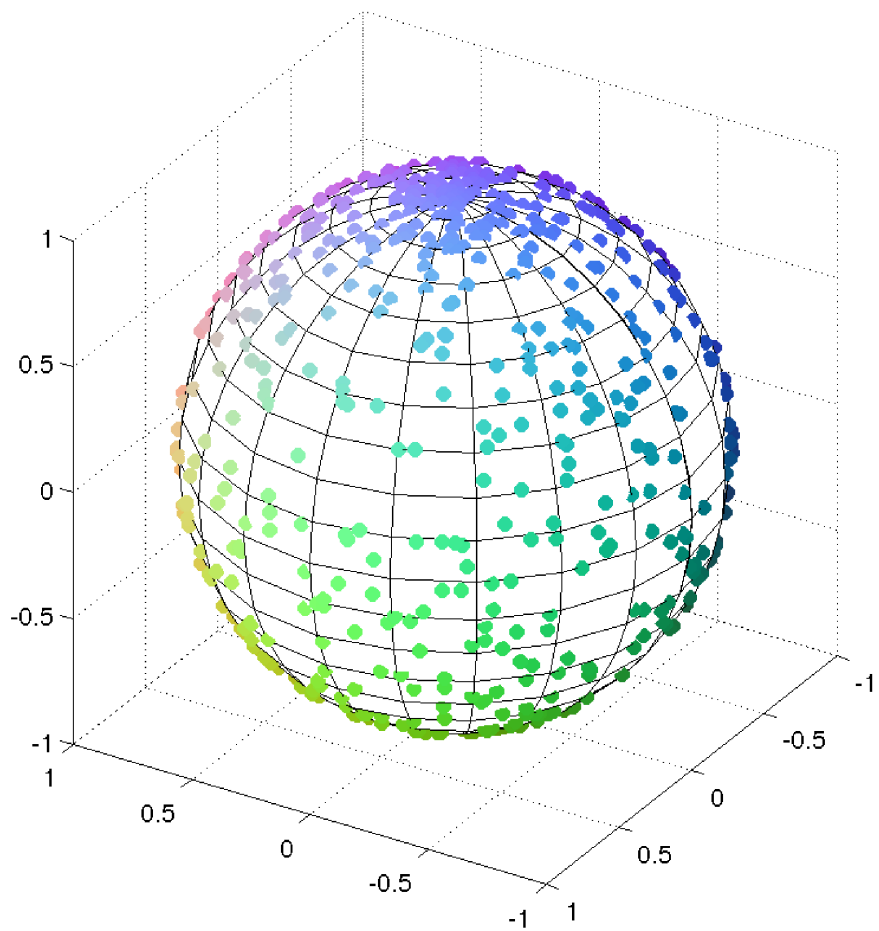
Minimize errors for best information retrieval.



Example data set



Embedding minimizes false positives  
(falsely retrieved neighbors)



Embedding minimizes misses (neighbors that were not retrieved)



$$1 - \textit{precision} = \frac{P_i^c \cap Q_i}{|Q_i|}$$

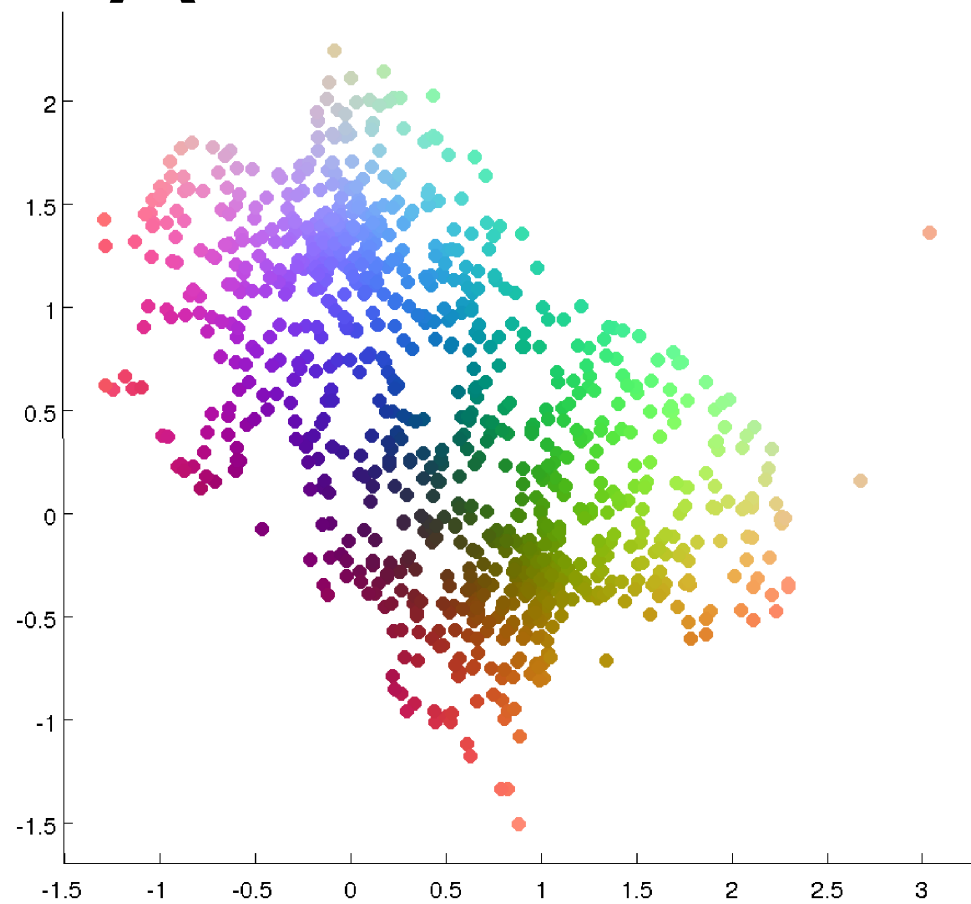
Proportion of  
false positives

$$1 - \textit{recall} = \frac{Q_i^c \cap P_i}{|P_i|}$$

Proportion of  
missed neighbors

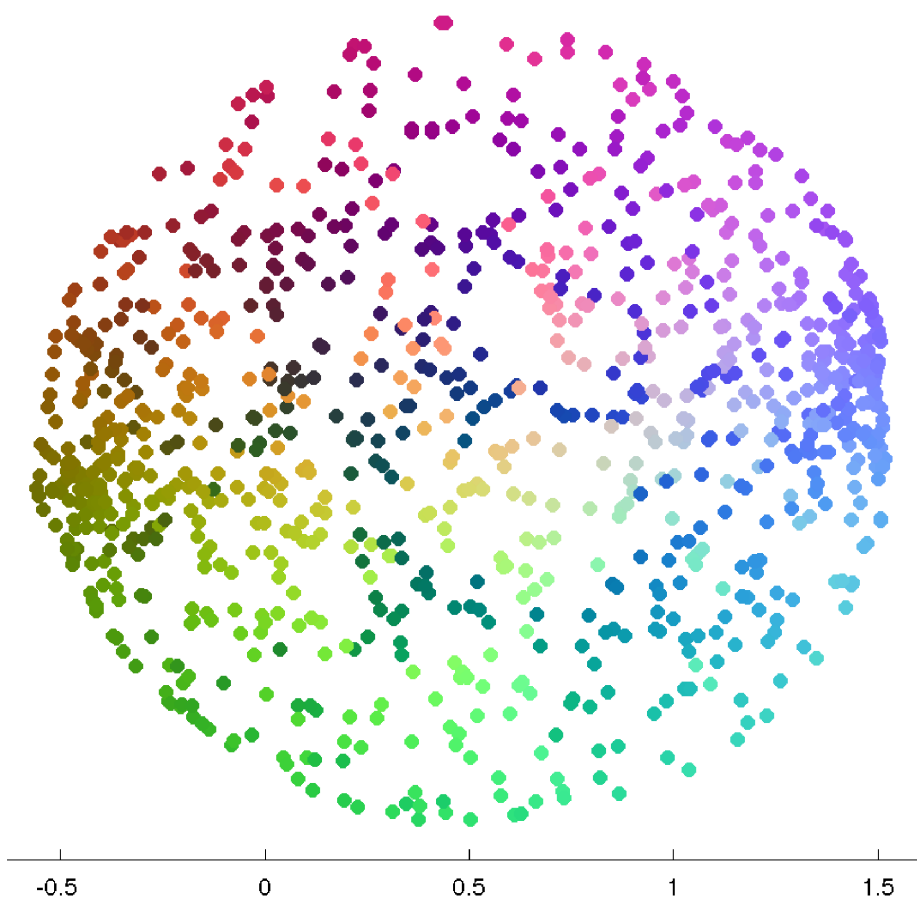
A visualization must make a **tradeoff** between false positives and misses. All methods end up with some tradeoff. A good visualization method should allow the user to specify the desired tradeoff.

# A



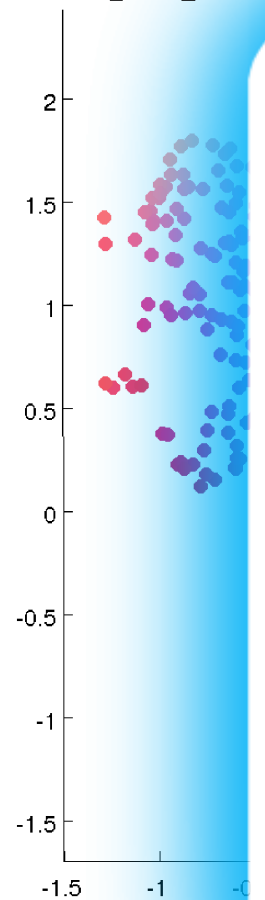
Embedding minimizes false positives  
(falsely retrieved neighbors)

# B



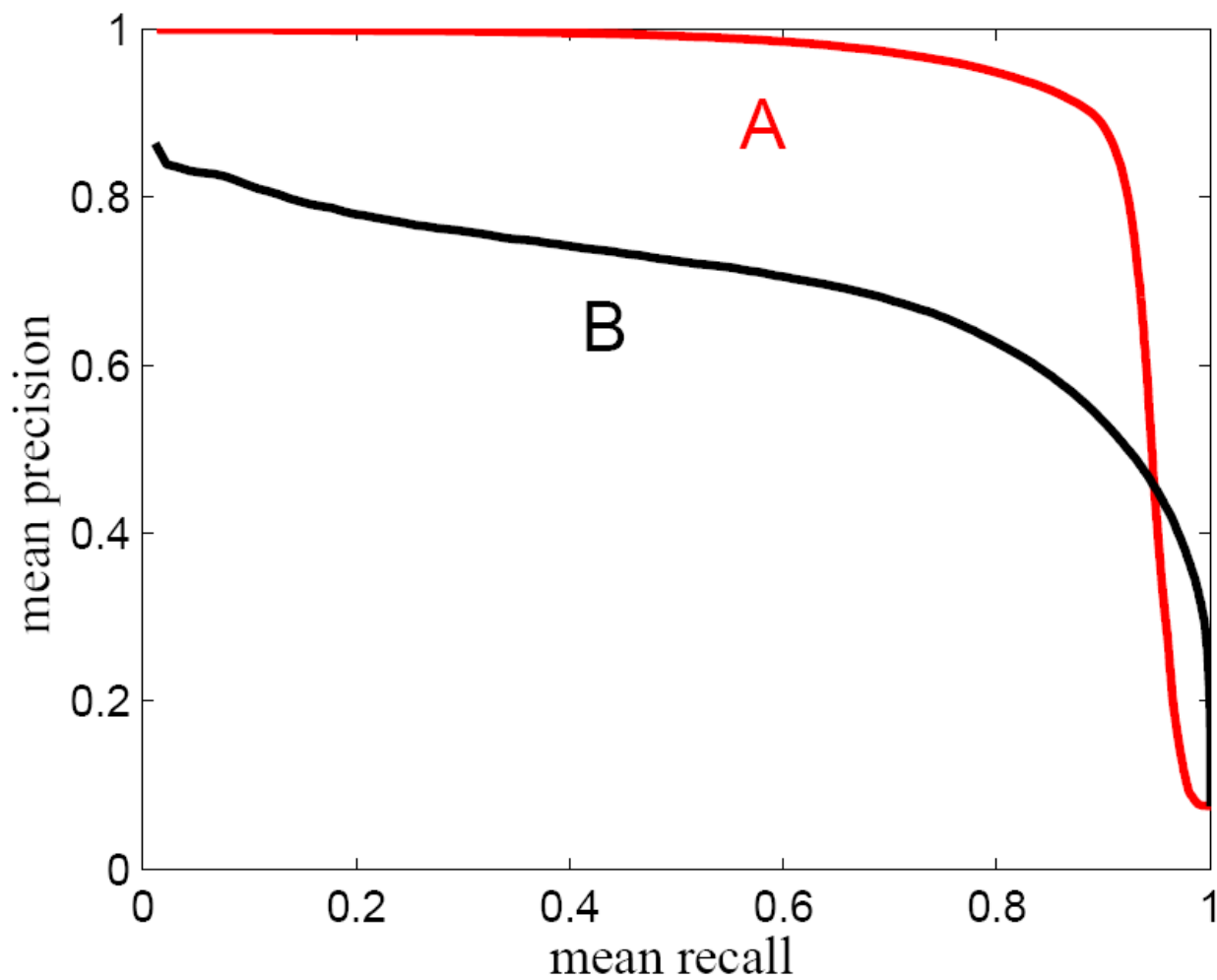
Embedding minimizes misses (neighbors  
that were not retrieved)

# A



Embeddings  
(falsely retrieved neighbors)

# B



Embeddings  
(that were not retrieved)

### Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

### Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Recall:

$$\sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Proof of connection in Venna et al. 2010: assume some probabilities are uniformly-large, some are uniformly-small. Then there are 4 different kinds of terms in the sum. Show that above KL divergence is dominated by a cost that is proportional to a constant times number of misses.

**Input neighborhood**

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

**Output neighborhood**

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

**Precision:**

$$\sum_i \sum_{j \neq i} q_{j|i} \log \frac{q_{j|i}}{p_{j|i}}$$

Proof of connection is similar to recall: assume some probabilities are uniformly-large, some are uniformly-small. Then there are 4 different kinds of terms in the sum. Show that above KL divergence is dominated by a term proportional to a constant times number of false neighbors.

### Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

### Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

### Tradeoff measure

$$E_{\text{NeRV}} = \lambda \mathbb{E}_i[D(p_i, q_i)] + (1 - \lambda) \mathbb{E}_i[D(q_i, p_i)]$$

Minimize with respect to output coordinates  $y_i$

# Neighbor Retrieval Visualizer

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

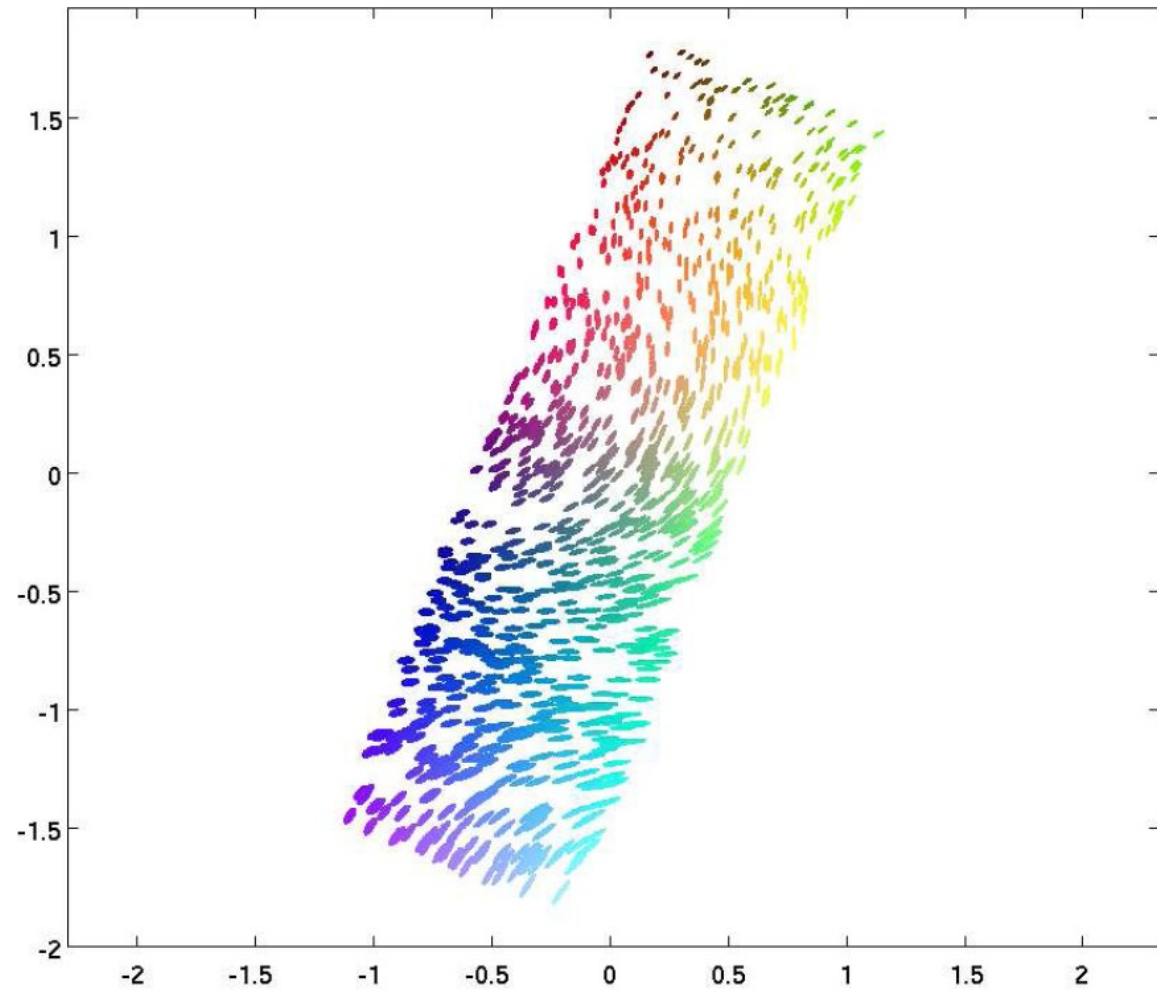
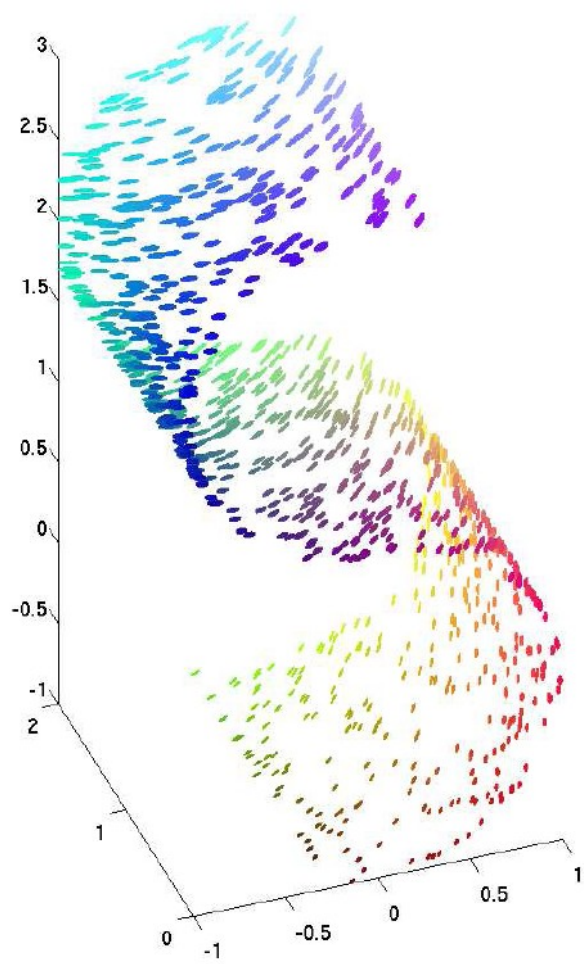
Tradeoff measure

$$E_{\text{NeRV}} = \lambda \mathbb{E}_i [D(p_i, q_i)] + (1 - \lambda) \mathbb{E}_i [D(q_i, p_i)]$$

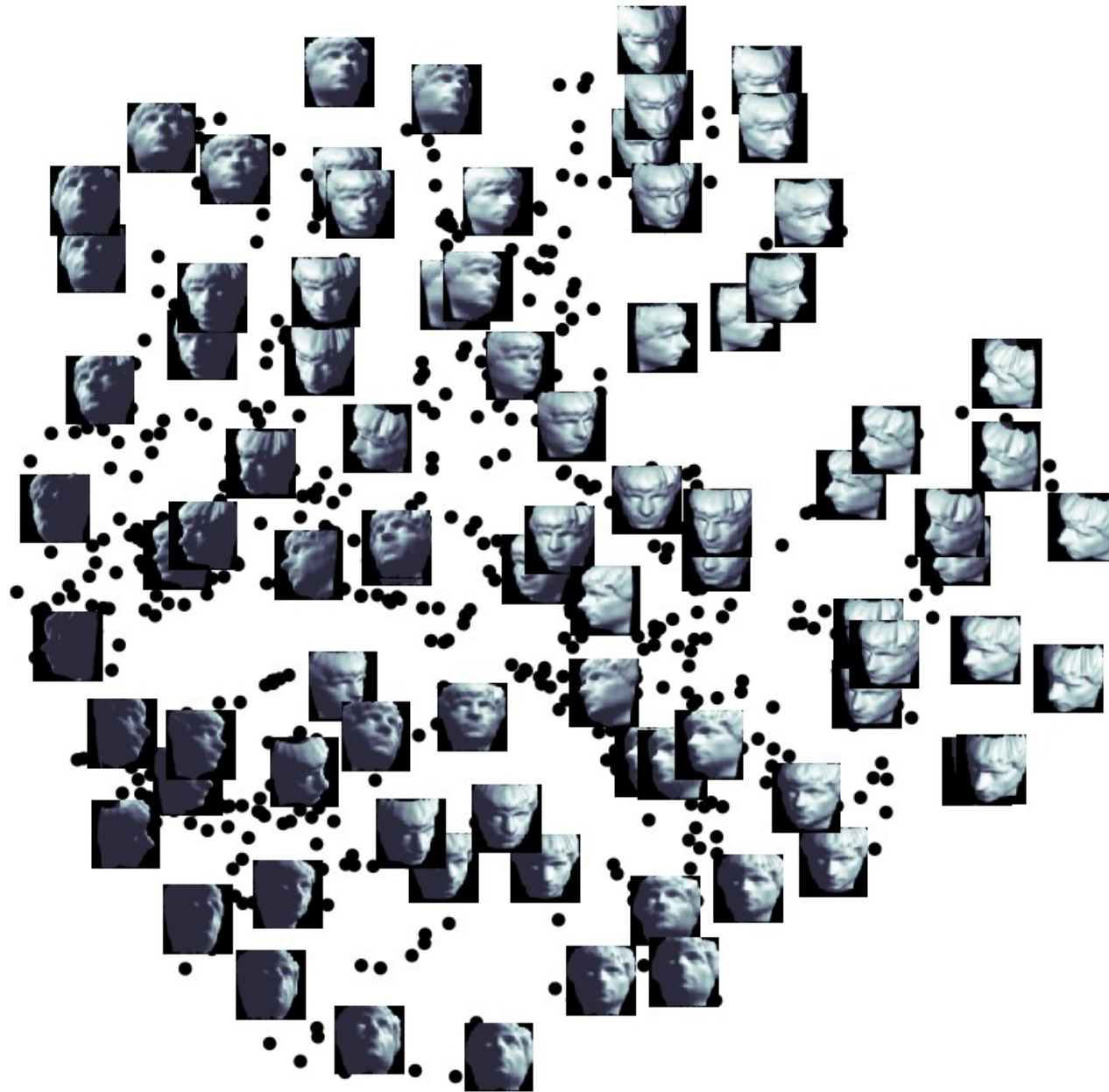
Minimize with respect to output coordinates  $y_i$



# NeRV visualization

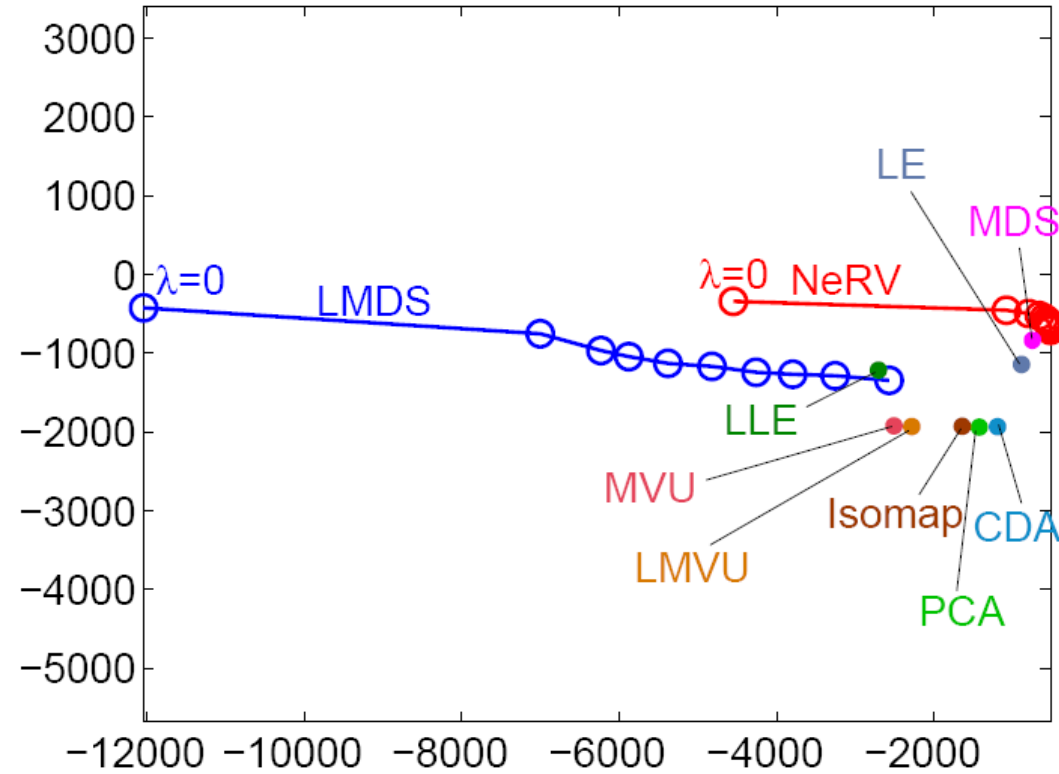


Of course NeRV can unfold the simple cases.

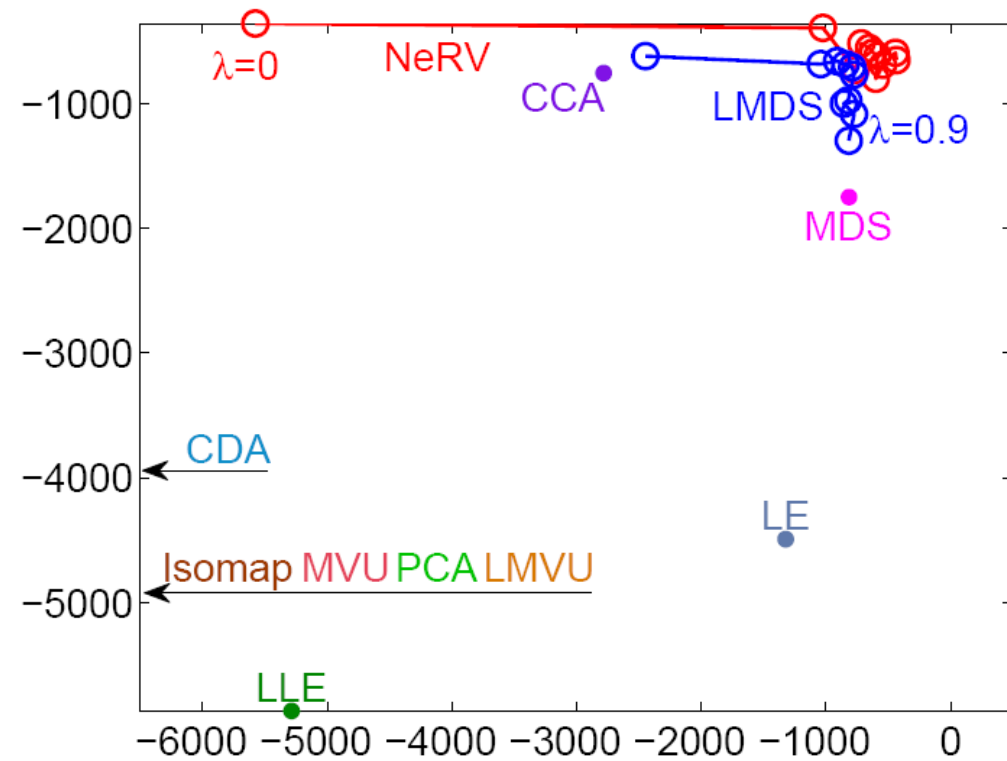


NeRV visualization of a complicated face image data set

### Faces

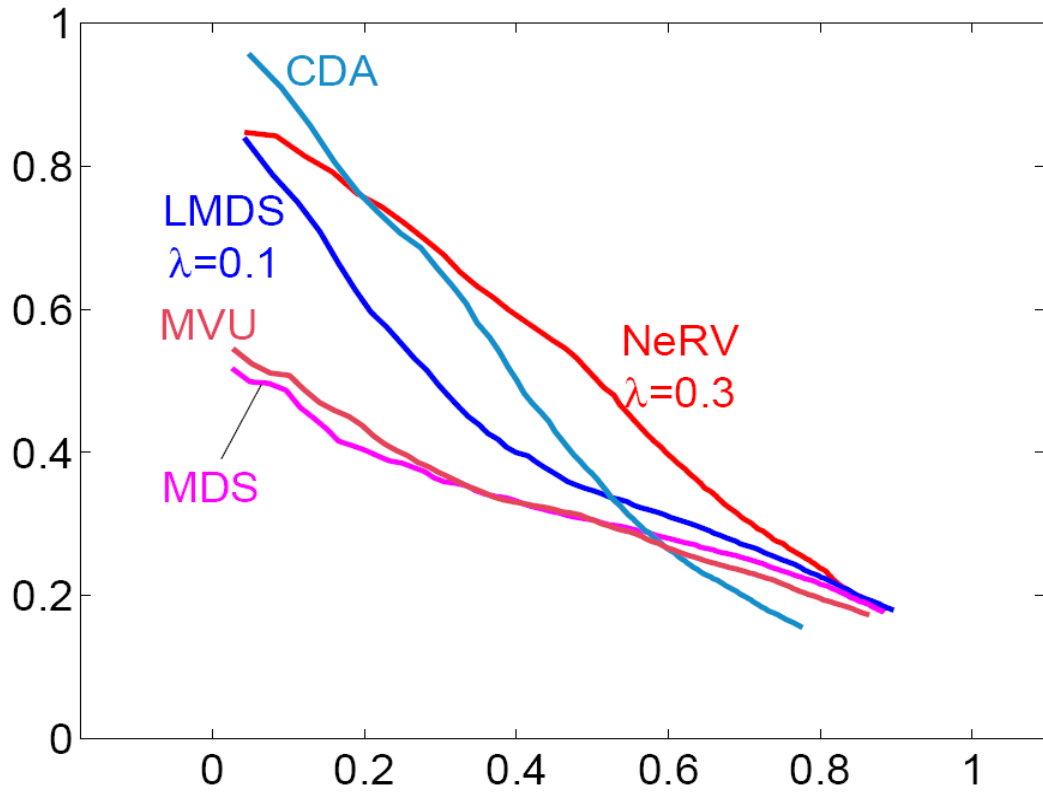


### Seawater temperature time series

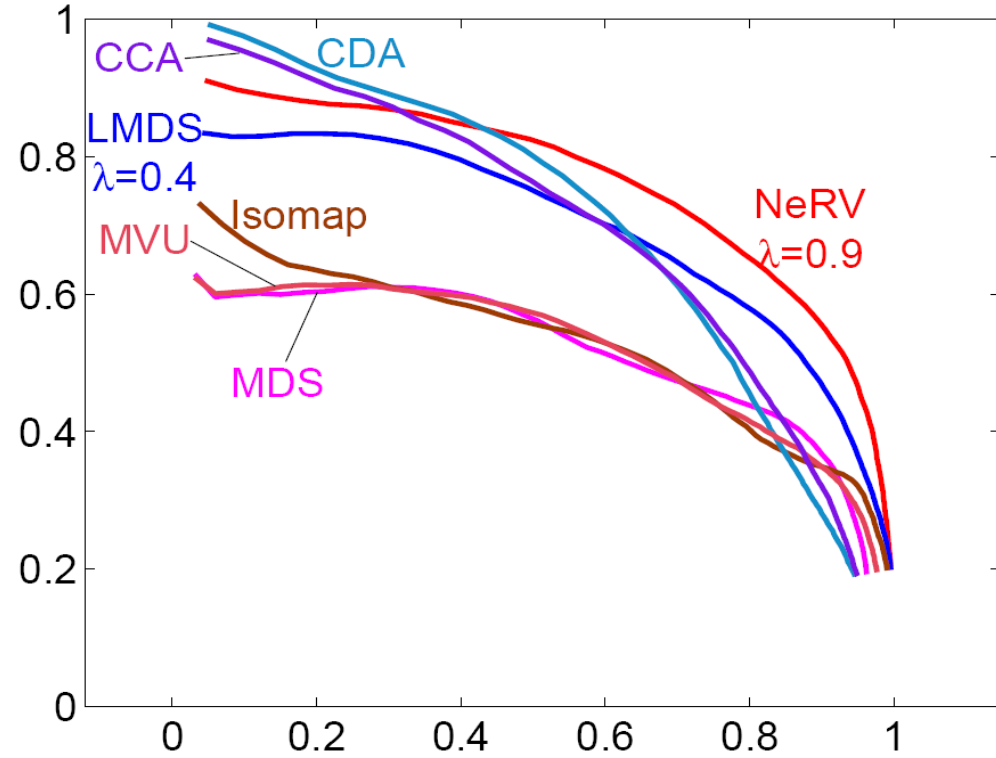


New measures: smoothed precision (vertical axes) / recall (horizontal axes)

### Faces



### Seawater temperature time series



Standard precision / recall curves (novel for visualization!)

# Next Lecture

Some variants of NeRV

Neighbor Embedding by Generative Modeling

Some supervised neighbor embedding approaches

Fast approximations for embedding large data

Quality assessment techniques

If time allows, graph embedding approaches

# References

Geoffrey Hinton and Sam Roweis. **Stochastic neighbor embedding**. In Proceedings of NIPS 2002.

Laurens van der Maaten and Geoffrey Hinton. **Visualizing non-metric similarities in multiple maps**. Machine Learning, 2011.

Laurens van der Maaten and Geoffrey Hinton. **Visualizing data using t-SNE**. Journal of Machine Learning Research, 2008.

Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. **Information retrieval perspective to nonlinear dimensionality reduction for data visualization**. Journal of Machine Learning Research, 2010.

# t-distributed NeRV

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

$$p_{i,j} = \frac{1}{2n} (p_{i|j} + p_{j|i})$$

Output neighborhood

$$q_{i,j} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

Tradeoff measure = t-NeRV cost function

$$E_{\text{t-NeRV}} = \lambda \sum_i \sum_{j \neq i} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} + (1 - \lambda) \sum_i \sum_{j \neq i} q_{i,j} \log \frac{q_{i,j}}{p_{i,j}} = \lambda D(p, q) + (1 - \lambda) D(q, p)$$

# NeRV with a linear projection

Input neighborhood

$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

Tradeoff measure = NeRV cost function

$$E_{\text{NeRV}} = \lambda \mathbb{E}_i [D(p_i, q_i)] + (1 - \lambda) \mathbb{E}_i [D(q_i, p_i)]$$

$$\text{Restrict } \mathbf{y}_i = \mathbf{W}^T \mathbf{x}_i$$

Minimize cost with respect to projection  $\mathbf{W}$



# NeRV with a linear projection

Input neighborhood

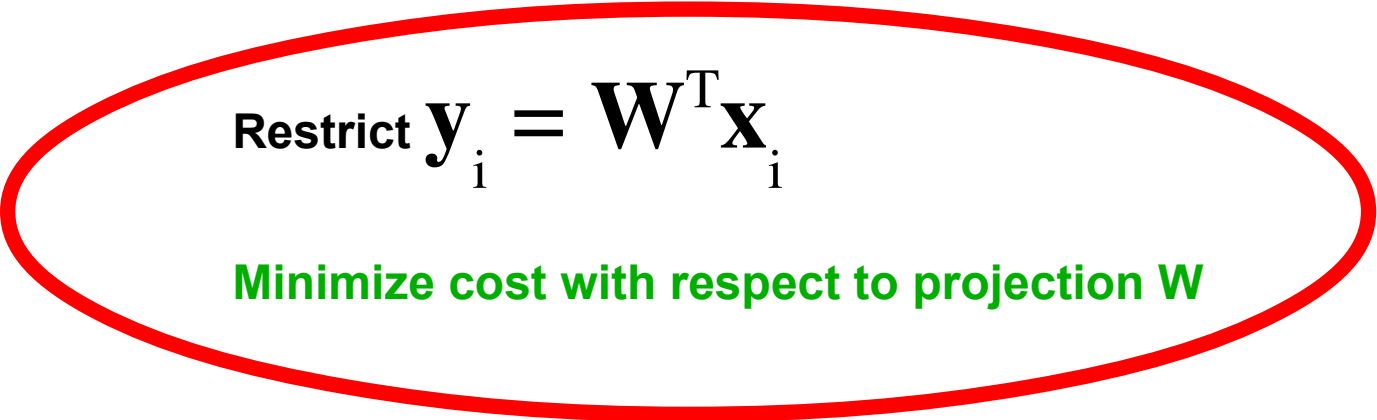
$$p_{j|i} = \frac{\exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_k)^2}{\sigma_i^2}\right)}$$

Output neighborhood

$$q_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_k\|^2}{\sigma_i^2}\right)}$$

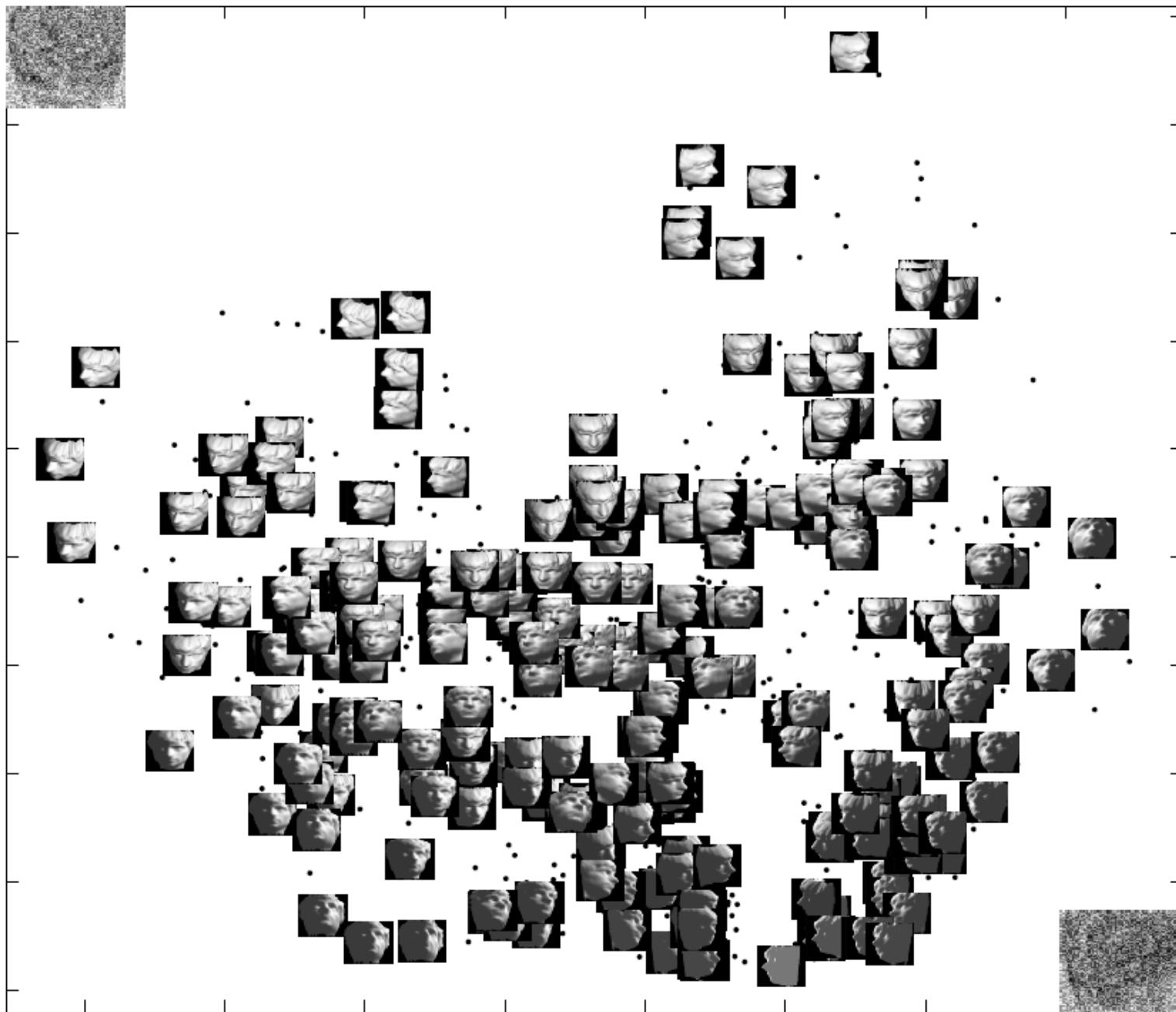
Tradeoff measure = NeRV cost function

$$E_{\text{NeRV}} = \lambda \mathbb{E}_i [D(p_i, q_i)] + (1 - \lambda) \mathbb{E}_i [D(q_i, p_i)]$$


$$\text{Restrict } \mathbf{y}_i = \mathbf{W}^T \mathbf{x}_i$$

Minimize cost with respect to projection  $\mathbf{W}$

**Extension 1:** NeRV with a linear projection.  
Neighborhoods and features can be given separately.



# NeRV by generative modeling

Stochastic Neighbor Embedding can be seen as a generative model, but it only focuses on recall (misses) because its cost function is **dominated by misses**.

Idea: **change the retrieval model so that misses become less dominant, so that the model can also focus on false positives.**

New retrieval distribution: mixture of the **user model** and an **explaining away model**.

$$q_{ij} \propto r_{ij} + \gamma p_{ij}$$

*new retrieval distribution*      *plain user model*      *explaining away model = true neighborhood distribution*

*amount of explaining away*

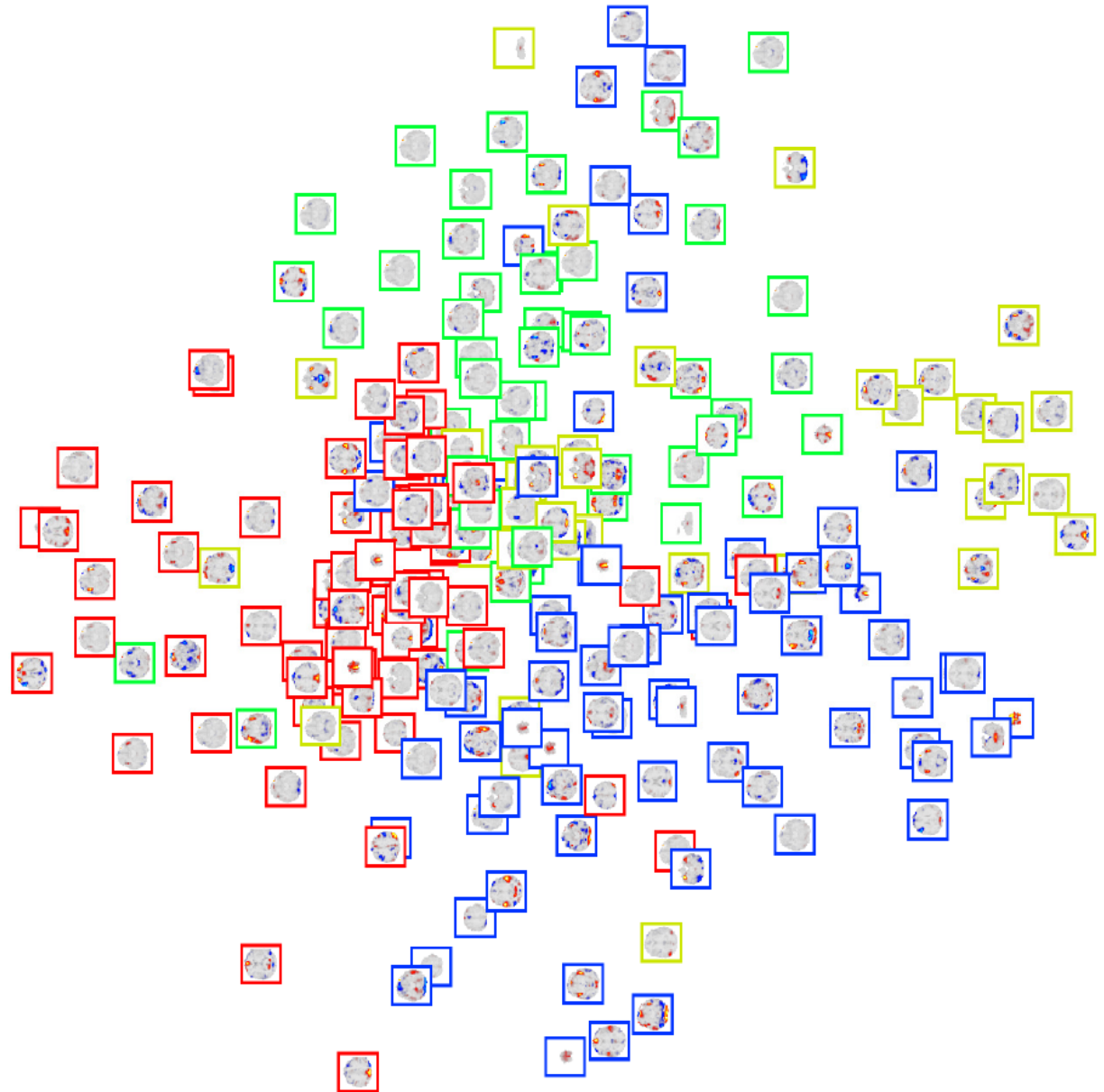
Cost function is log-likelihood (generative modeling):

$$L = \sum_i \sum_{j \neq i} p_{ij} \log q_{ij}$$

fMRI measurements of 6 adults who received four types of stimuli:

- tactile (red)
- auditory tone (yellow)
- auditory voice (green)
- visual (blue).

Visualization by the new method, strong explaining away used during training. Different stimuli types become separated in the (unsupervised) visualization.



# Some References for Next Week

Zhirong Yang, Jaakko Peltonen, and Samuel Kaski. **Scalable Optimization of Neighbor Embedding for Visualization**. In ICML 2013, International Conference on Machine Learning. In Proceedings of ICML 2013, International Conference on Machine Learning.

Jaakko Peltonen and Konstantinos Georgatzis. **Efficient Optimization for Data Visualization as an Information Retrieval Task**. In Proceedings of MLSP 2012.

Jaakko Peltonen, Helena Aidos, and Samuel Kaski. **Supervised Nonlinear Dimensionality Reduction by Neighbor Retrieval**. In Proceedings of ICASSP 2009.

Jaakko Peltonen and Samuel Kaski. **Discriminative Components of Data**. IEEE Transactions on Neural Networks, 2005.