Unsupervised Learning: t-SNE and UMAP

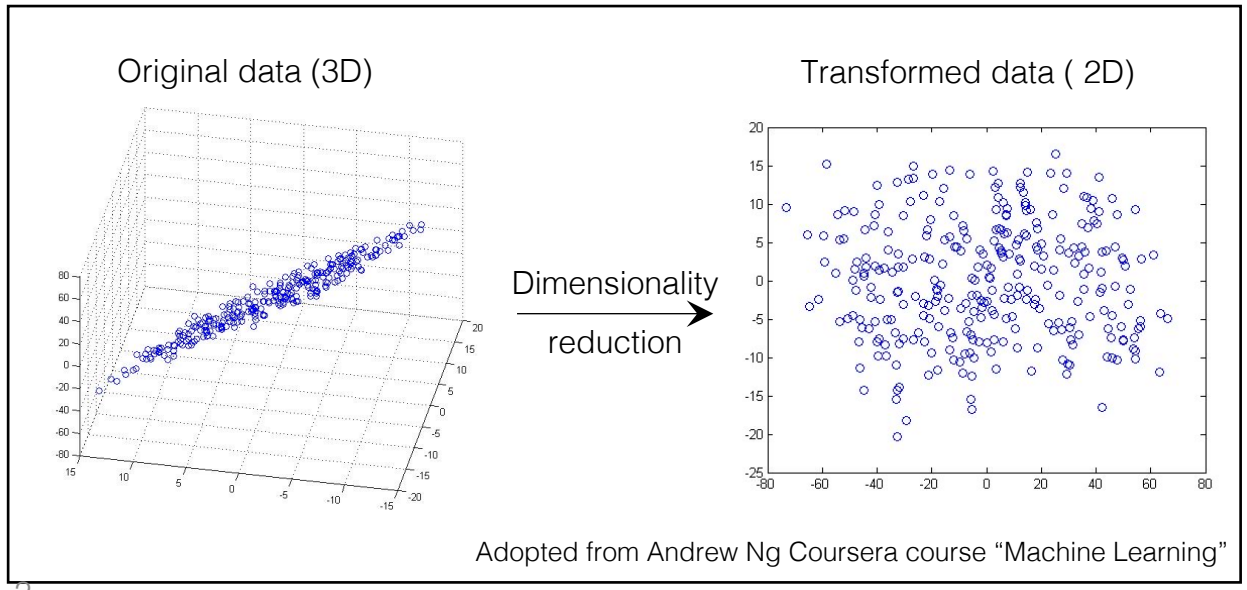
Nezar Abdennur

with slides from Zhipeng Weng and Shaimae Elhajjajy

Review: dimensionality reduction

Response or Output (Y)	Available	Not available	Partially available
	Supervised	**Unsupervised**	**Semi-supervised**
Discrete / Categorical	Classification	Clustering	**Unsupervised techniques** to learn structure of data, **supervised techniques** to make best guess predictions
Continuous	Regression	Dimensionality reduction	
	Find function f: X → Y	Infer the underlying structure of X	

Adapted from An Introduction to Statistical Learning



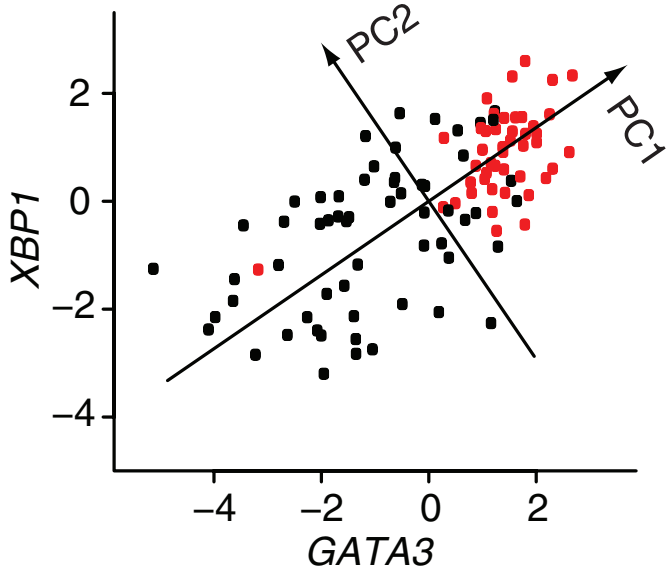
Adopted from Andrew Ng Coursera course "Machine Learning"

Review: principal component analysis (PCA)

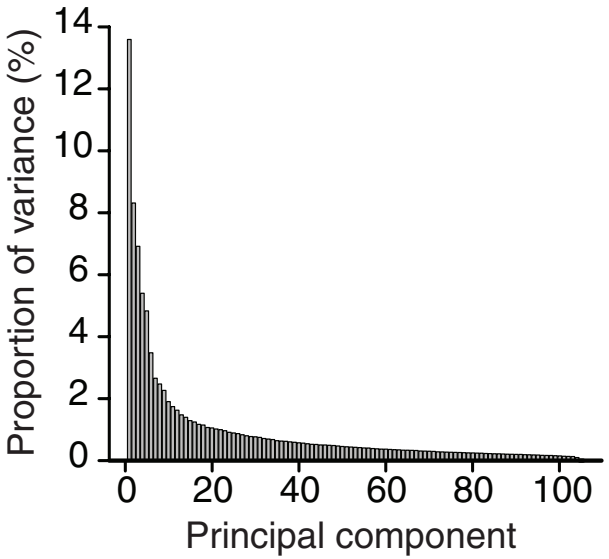
- PCA uses a small number of representative variables (the largest PCs) to collectively explain most of the variance in the original data.

- PCA is a linear approach that preserves global structure.
  - i.e., it makes sure data points that are far apart in high-dimensional space are also far apart in low-dimensional space.
  - PCA preserves large pairwise distances.

Identifying principal components (PCs) that capture variation in the data



The proportion of variance explained by each PC

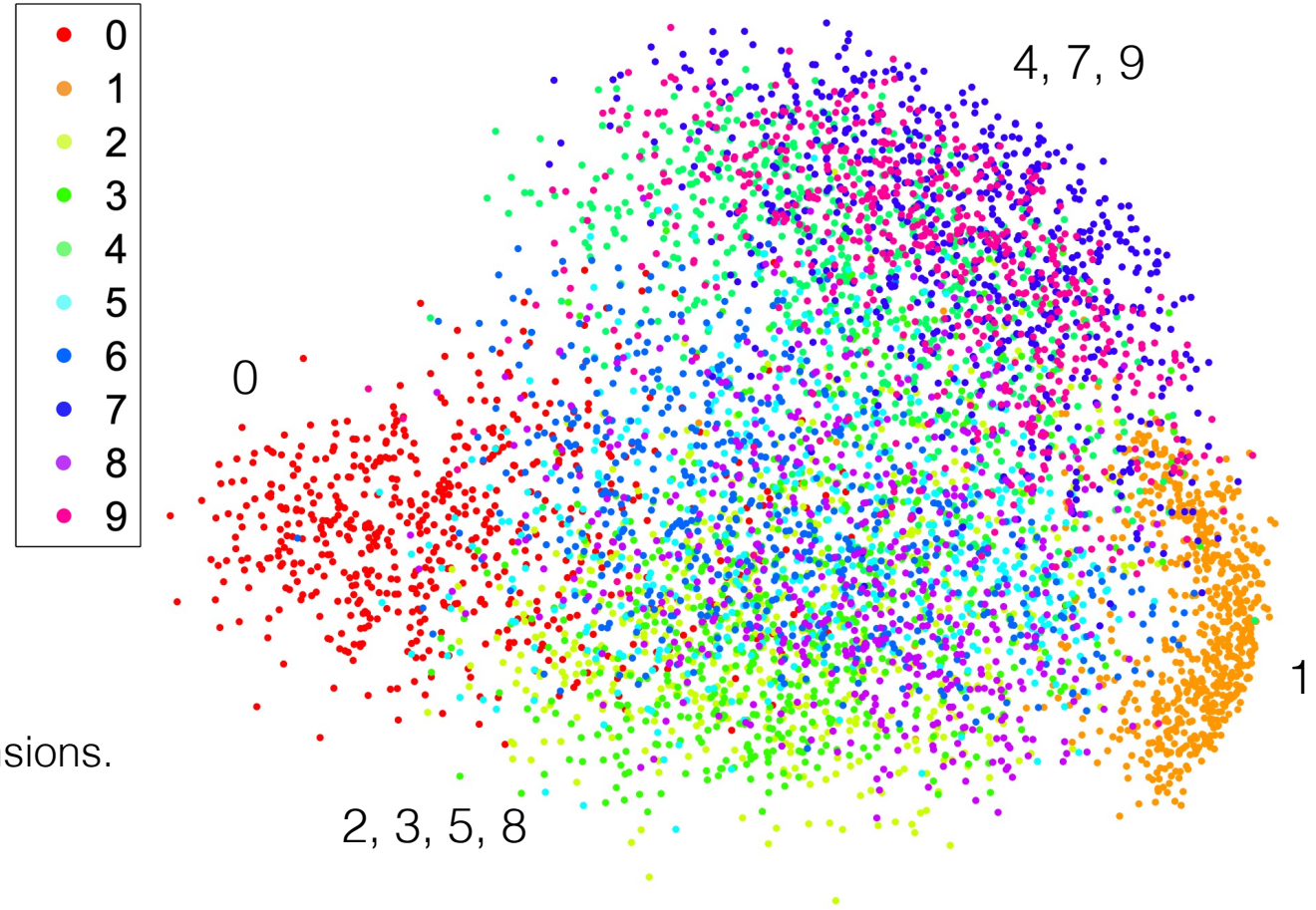


Ringnér, *Nat Biotechnol*, 2008
van der Maaten, *Google Tech Talk*, 2013

Linear dimensionality reduction methods are limited

MNIST dataset

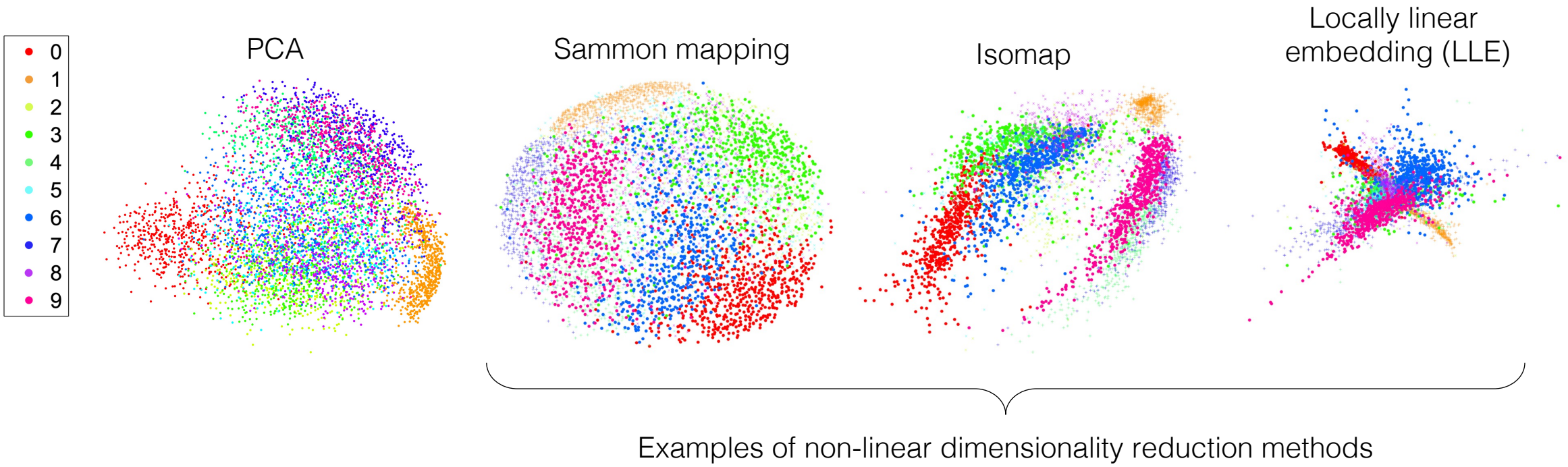
PCA, applied to MNIST



Each image is 28x28 pixels, hence 784 dimensions.

LeCun et al., *Proc. of the IEEE*, 1998
van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
van der Maaten, *AI-STATS*, 2009

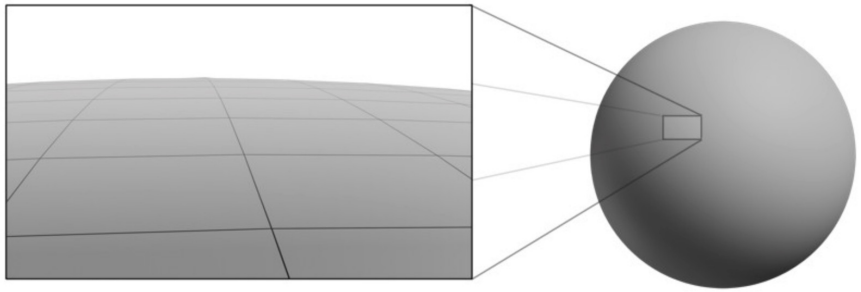
Dimensionality reduction methods applied to MNIST



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008

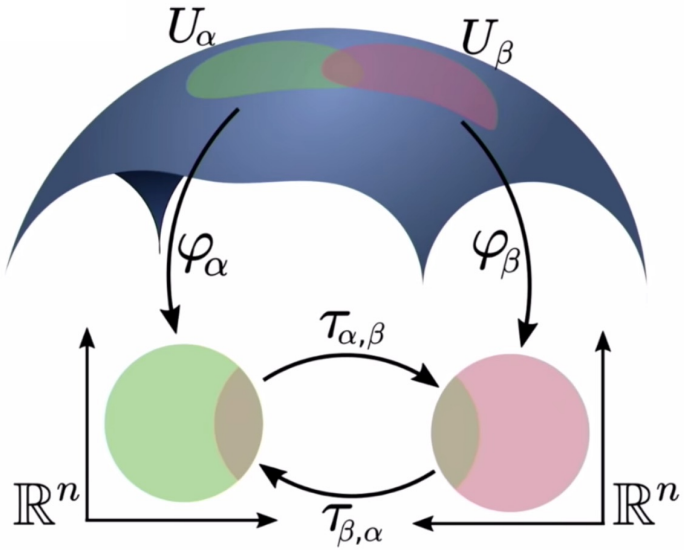
Non-linear dimensionality reduction methods are needed for manifolds

- A manifold captures the internal structure of the data. It is a space that is locally flat (i.e., locally resembles the Euclidean space $\mathbb{R}^n$).



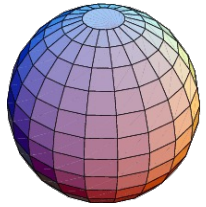
https://sinestesia.co/blog/tutorials/non-manifold-meshes-and-how-to-fix-them/

- Every point on the manifold has a neighborhood that maps one-to-one to a region in flat space ($\mathbb{R}^n$).

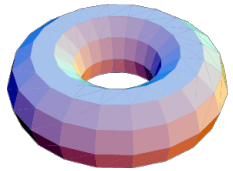


Examples of manifolds:

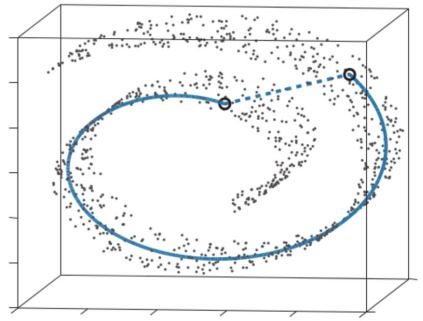
Sphere



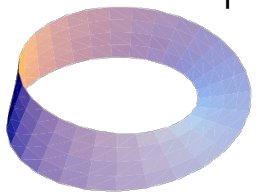
Torus



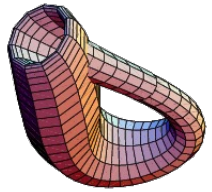
Swiss Roll



Mobius strip



Klein bottle



McInnes, *SciPy*, 2018
Manifold example images from Wolfrom MathWorld

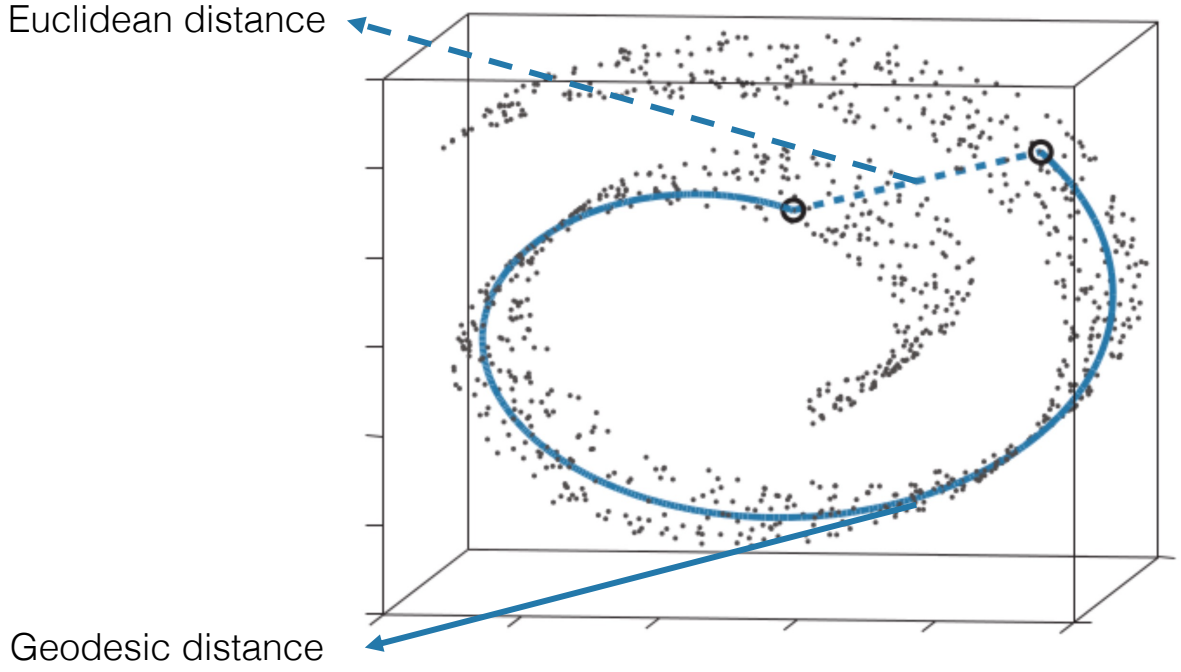
Non-linear dimensionality reduction methods are needed for manifolds

- In $\mathbb{R}^n$, the distance between two points is the length of a line segment that connects these two points. This is measured by the Euclidean distance.

- In a curved manifold, the distance between two points is the length of the shortest curve that connects the two points. This is measured by the geodesic distance.

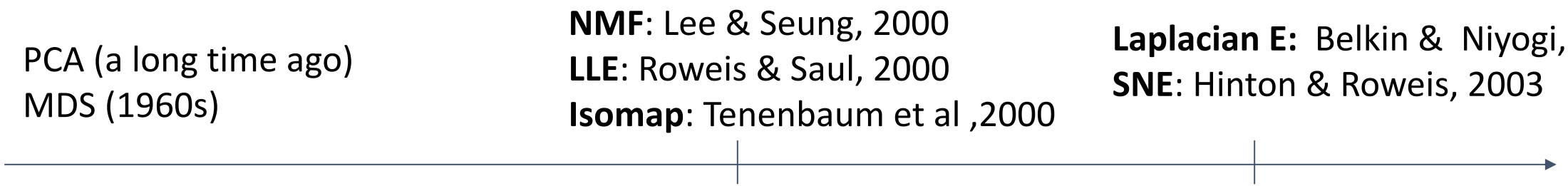
- For manifolds such as the Swiss Roll, non-linear dimensionality reduction methods are necessary to accurately capture the distance, or similarity, between data points.

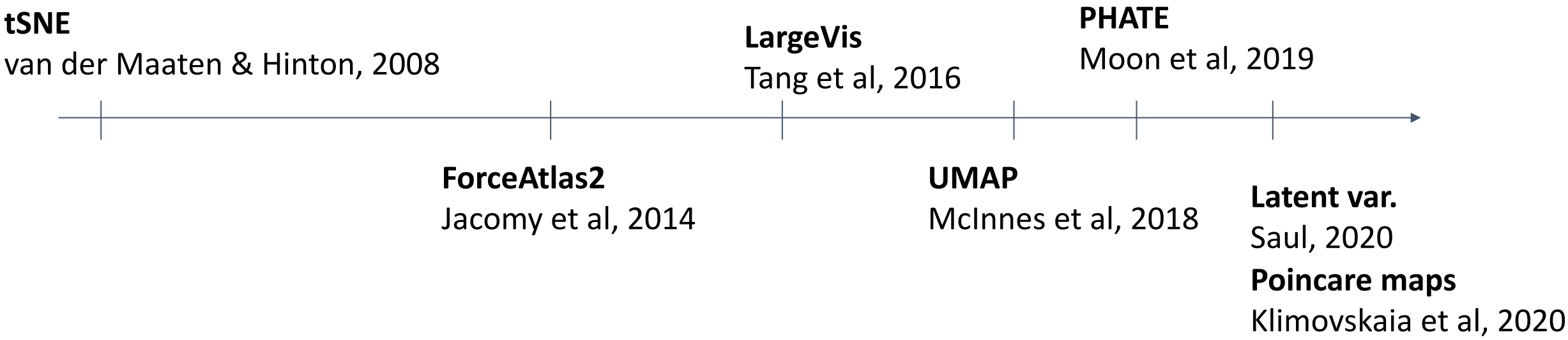
The Swiss Roll



Tenenbaum et al., *Science*, 2000
van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
van der Maaten, *Google Tech Talk*, 2013

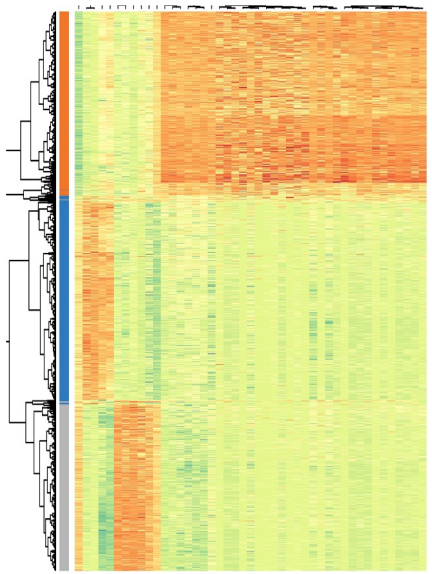
Manifold Learning





PCA

Features (high-dim: p)



n points

(n, p)

mean-center

X

X^TX
feature covariance
(p, p)

svd(X)

XX^T
data covariance
(n, n)

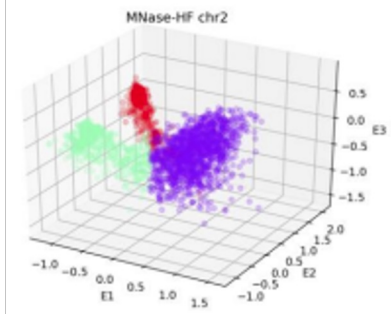
data similarities

$eig(X^TX)$

$eig(XX^T)$

Projections (low-dim: q)

(n, q)



Laplacian Eigenmaps

(sklearn.manifold.SpectralEmbedding)

LETTER Communicated by Joshua B. Tenenbaum

Laplacian Eigenmaps for Dimensionality Reduction and Data Representation

Mikhail Belkin
misha@math.uchicago.edu
Department of Mathematics, University of Chicago, Chicago, IL 60637, U.S.A.

Partha Niyogi
niyogi@cs.uchicago.edu
Department of Computer Science and Statistics, University of Chicago, Chicago, IL 60637 U.S.A.

One of the central problems in machine learning and pattern recognition is to develop appropriate representations for complex data. We consider the problem of constructing a representation for data lying on a low-dimensional manifold embedded in a high-dimensional space. Drawing on the correspondence between the graph Laplacian, the Laplace Beltrami operator on the manifold, and the connections to the heat equation, we propose a geometrically motivated algorithm for representing the high-dimensional data. The algorithm provides a computationally efficient approach to nonlinear dimensionality reduction that has locality-preserving properties and a natural connection to clustering. Some potential applications and illustrative examples are discussed.

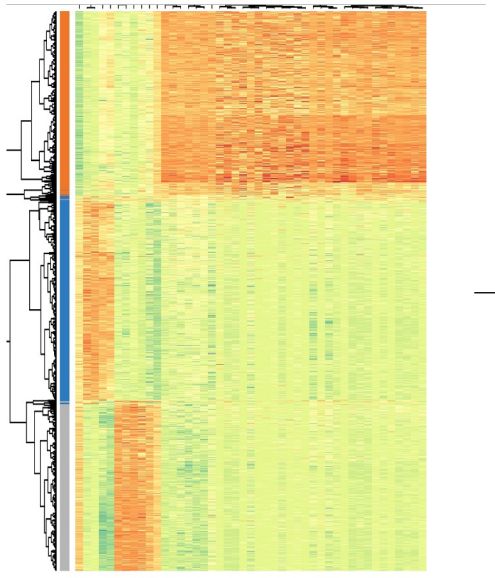
Apply a “kernel” to convert **global distances** to <u>local</u> similarities

Kernels estimate
geodesic proximity

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

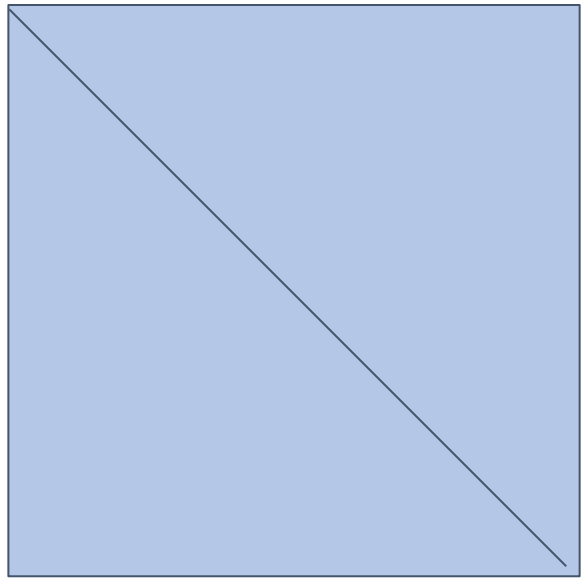
Features (high-dim: p)

n points



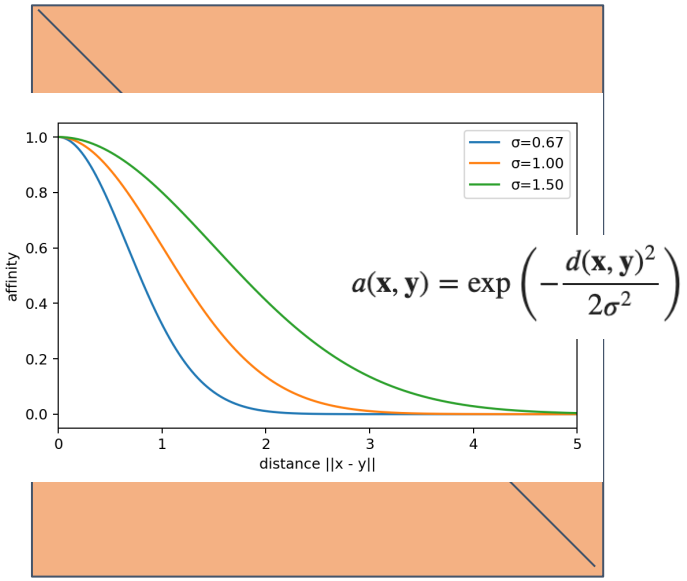
(n, p)

distance
metric



distances D
(n, n)

similarity
kernel



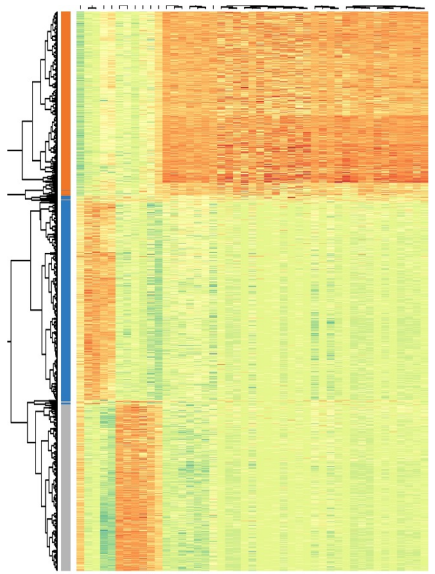
“affinities” A
(n, n)

Laplacian Eigenmaps

~“PCA on geodesic similarity”

Features (high-dim: p)

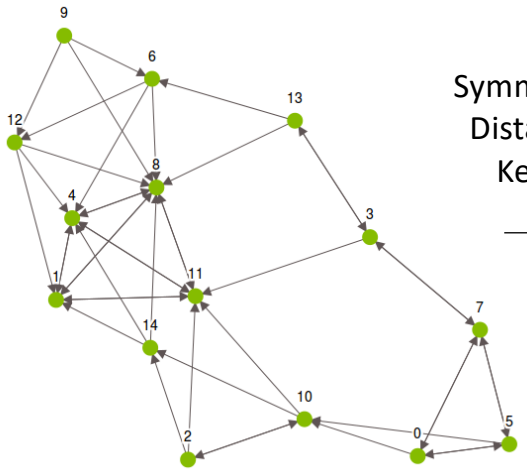
n points



(n, p)

kNN

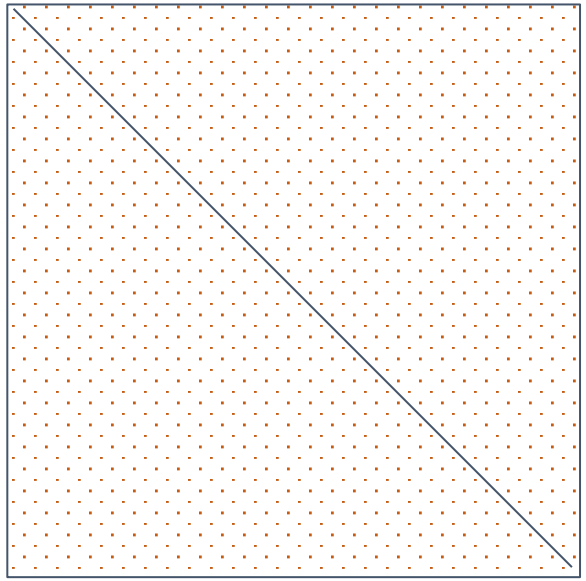
Nearest-neighbor graph



(n, n)
Sparse,
directed graph
(asymmetric)

Symmetrize,
Distances,
Kernel

Local affinities



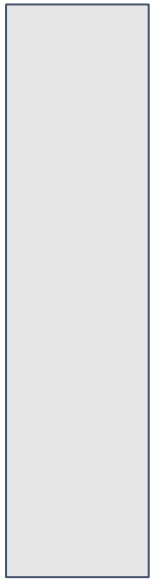
(n, n)
Sparse,
symmetric

$$L = D - A \qquad L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

Laplacian

Eig

Manifold Projection
(low-dim: q)



(n, q)

Laplacian Eigenmaps

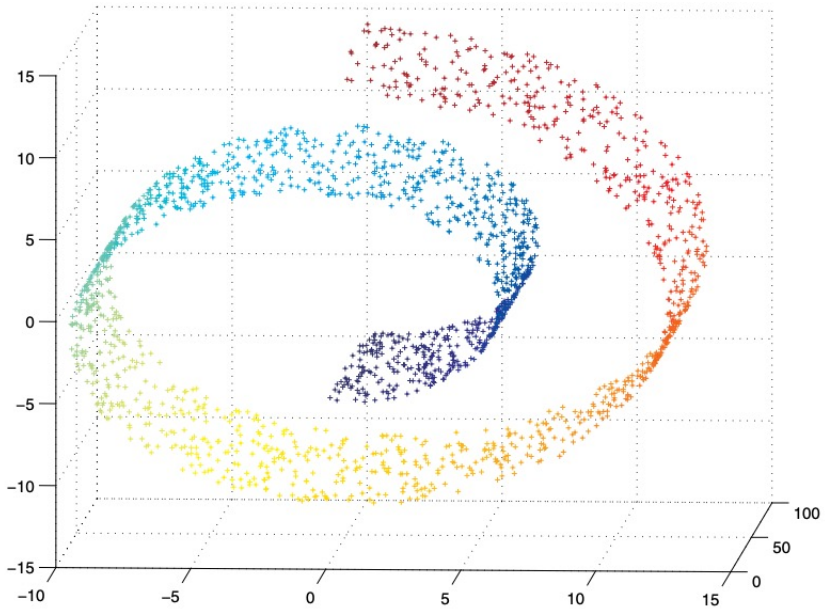


Figure 1: 2000 Random data points on the swiss roll.

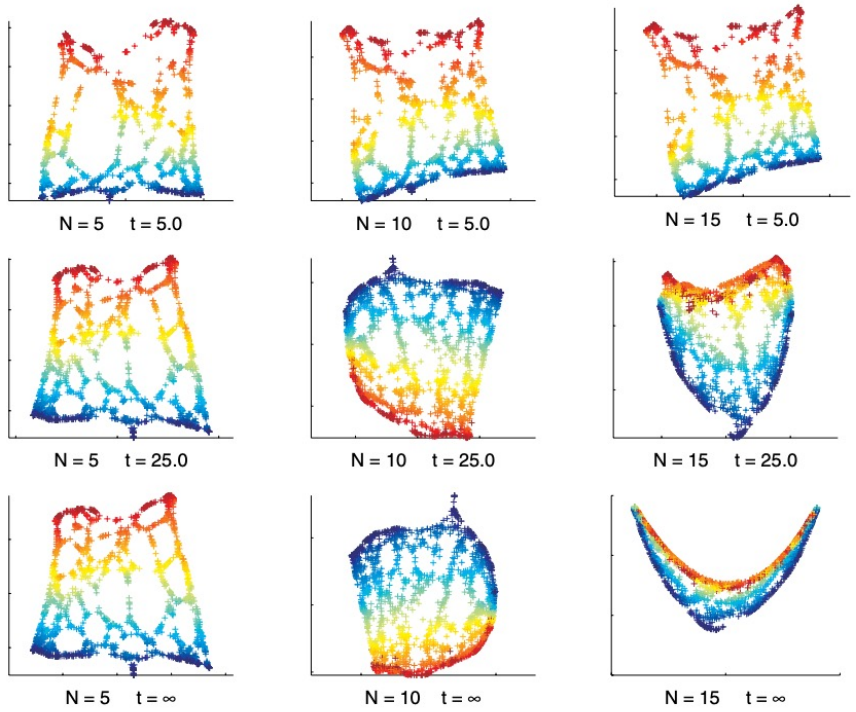


Figure 2: Two-dimensional representations of the swiss roll data, for different values of the number of nearest neighbors N and the heat kernel parameter t. $t = \infty$ corresponds to the discrete weights.

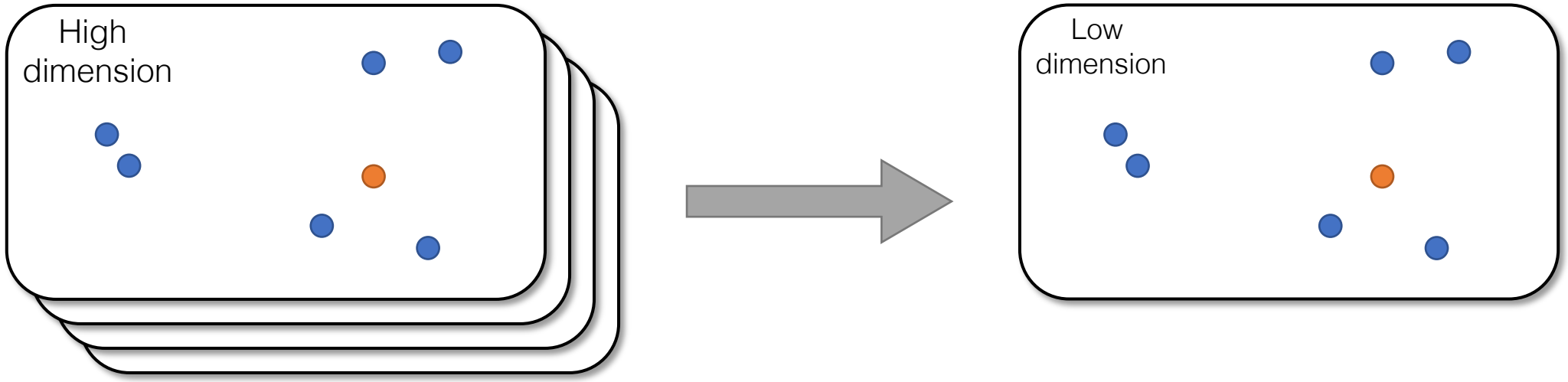
Belkin & Niyogi, 2003

Neighbor-based embedding: a framework

Goals of manifold learning (embedding)

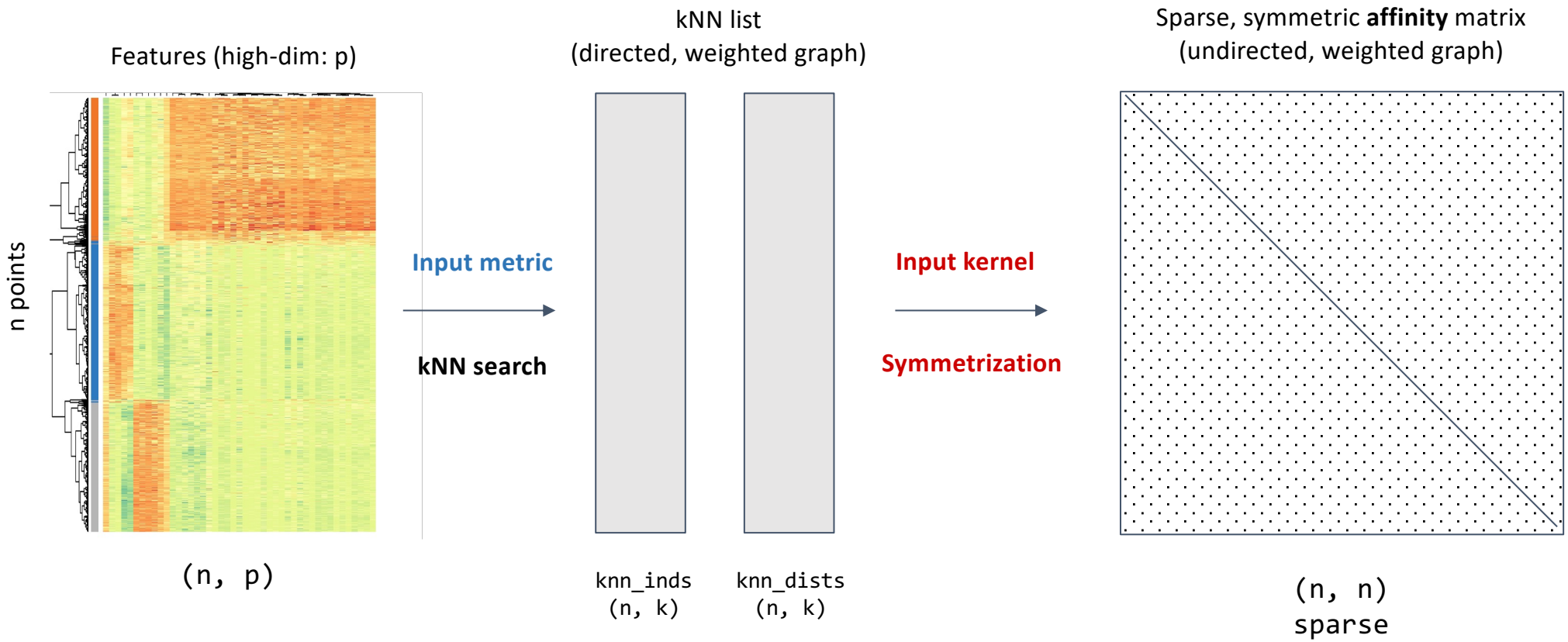
- Given a high-dimensional dataset, we want to construct a low-dimensional map that preserves the structure of the data as much as possible.

- t-SNE and UMAP are non-linear approaches that preserves local structure: i.e., make sure data points that are close together in high-dimensional space are also close together in low-dimensional space.

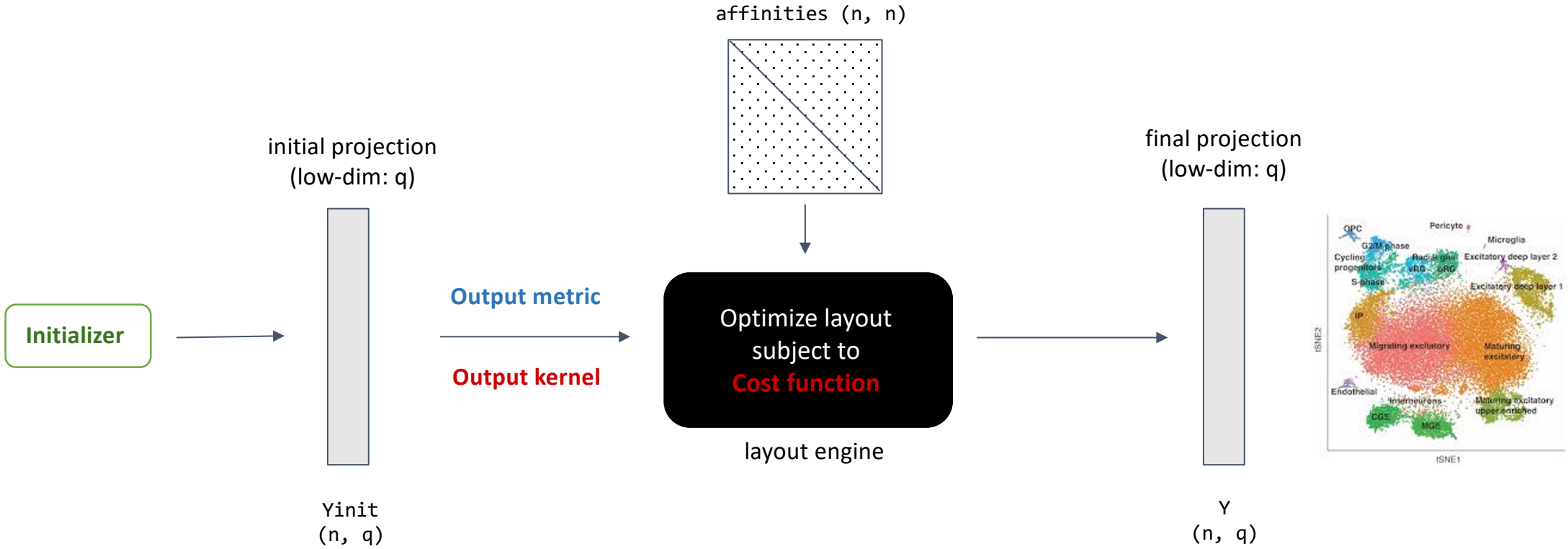


van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

Step 1: Calculate affinities



Step 2: Initialize and optimize the low-dim layout

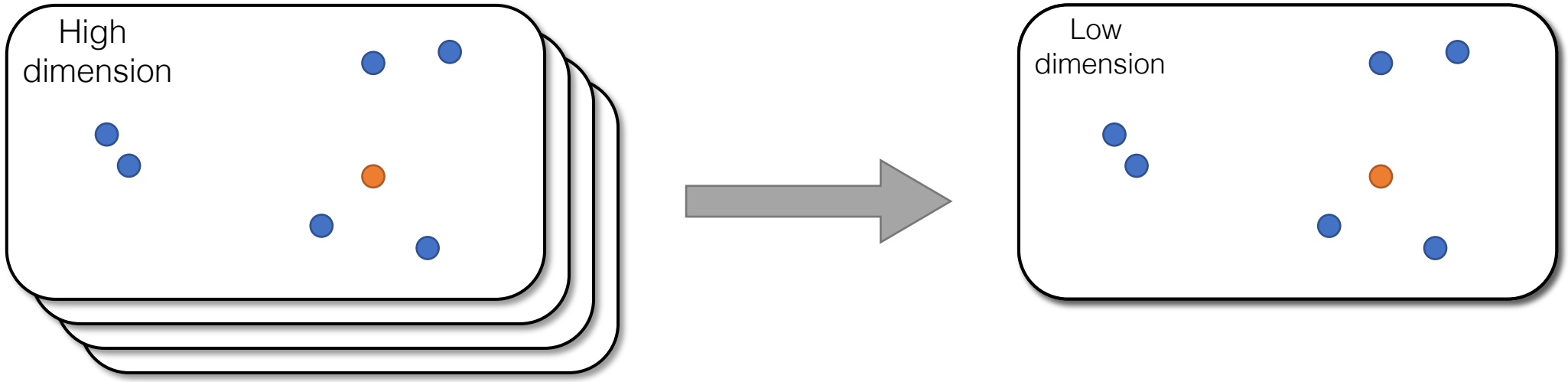


The cost function quantifies the overall difference between pairwise affinities in the **input** (high-dim) space and affinities in the **output** (low-dim) space.

Part 1: t-Distributed Stochastic Neighbor Embedding (t-SNE)

Overview of Stochastic Neighbor Embedding (SNE)

- The general steps of SNE are as follows:
  - We first compute the similarities between data points in the original high-dimensional dataset.
  - We next randomly map the data to a low dimension (2D or 3D) and compute the similarities between data points in the low dimension.
  - We then (iteratively) adjust data points in the low dimension to *minimize* the difference between similarities in the *high* dimension and similarities in the *low* dimension.



Adapted from van der Maaten, *Google Tech Talk*, 2013

Stochastic neighbor embedding (SNE)

1. Compute the similarities between data points in the original high-dimensional dataset.
   - Center a Gaussian distribution on a point x_i.
   - Use conditional probabilities under the Gaussian distribution to represent similarities between data point x_i and **nearest neighbors** x_j.
   - Repeat this computation for pairwise sets of points in the high dimension.

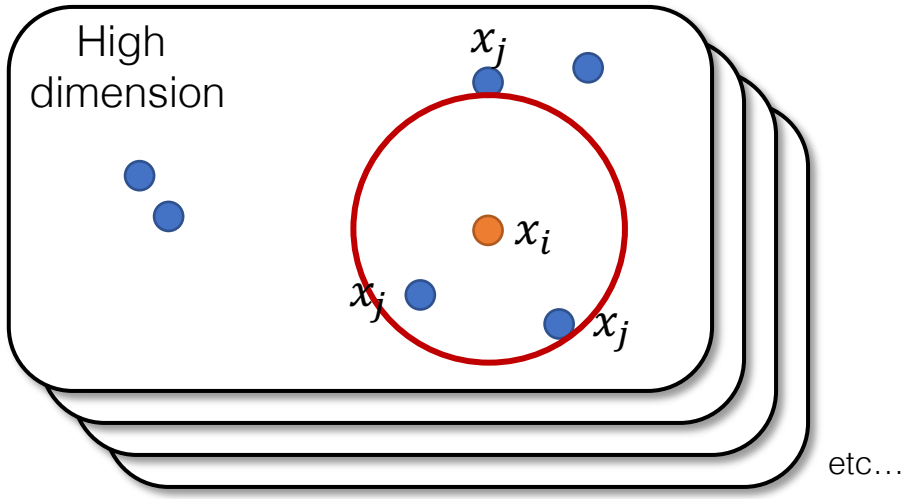
Similarities between data points in **high** dimension (SNE):

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

* $p_{j|i}$ *is large if points are close to each other*

* $p_{j|i}$ *is very small if points are far apart*

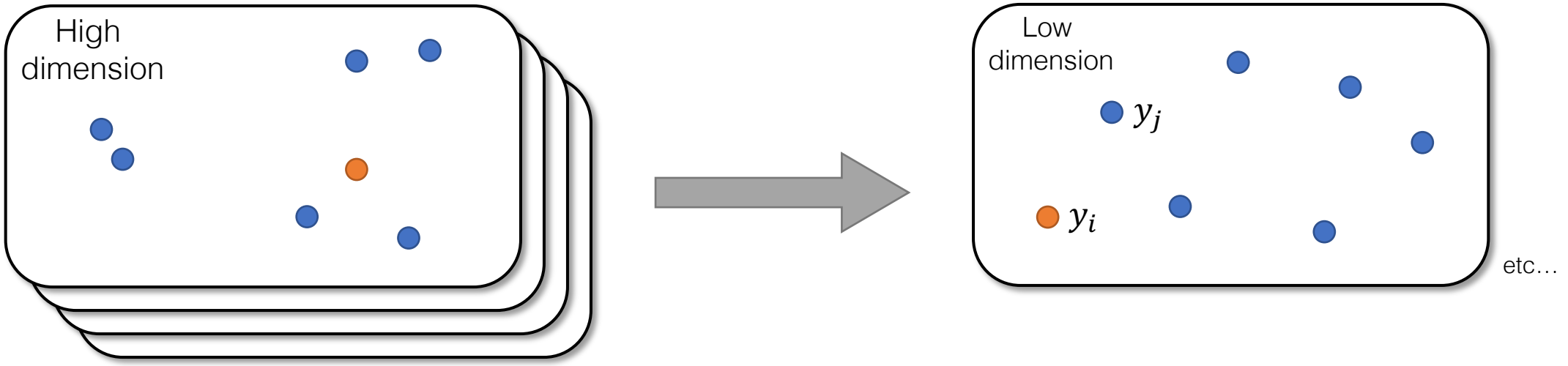
Kernel width σ_i varies by point *i*



Hinton & Roweis, *Advances in Neural Information Processing Systems*, 2002
Adapted from van der Maaten, *Google Tech Talk*, 2013

Stochastic neighbor embedding (SNE)

2. Randomly map data to a low dimension (2D or 3D)



Hinton & Roweis, *Advances in Neural Information Processing Systems*, 2002
Adapted from van der Maaten, *Google Tech Talk*, 2013

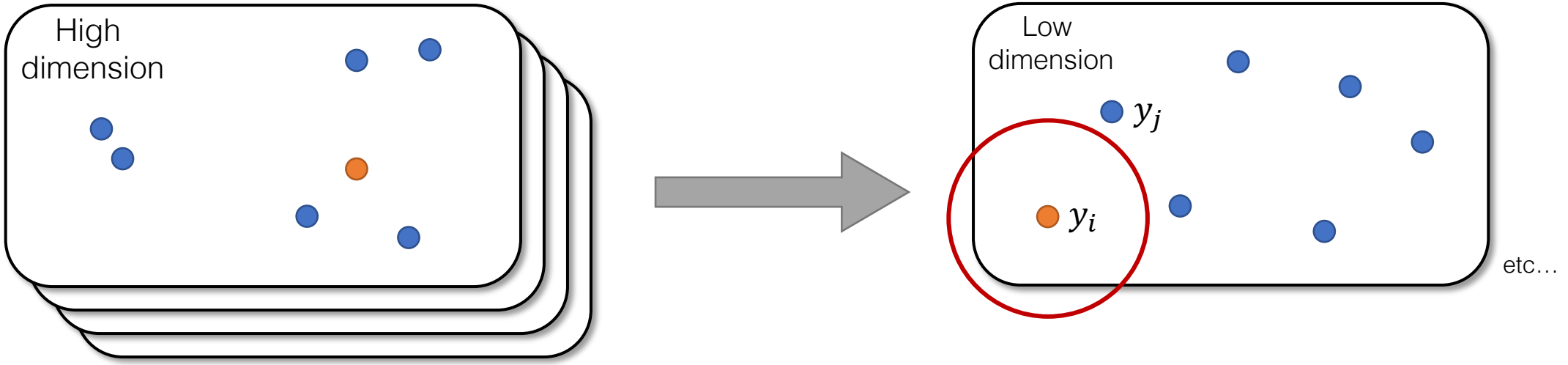
Stochastic neighbor embedding (SNE)

3. Compute the similarities between data points in the low dimension.

Similarities between data points in **low** dimension (SNE):

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}$$

In the low-dim space, we must consider the influence of all other points in the new layout! We can't limit ourselves to those from the high-dim neighbor graph.



Hinton & Roweis, *Advances in Neural Information Processing Systems*, 2002
Adapted from van der Maaten, *Google Tech Talk*, 2013

Stochastic neighbor embedding (SNE)

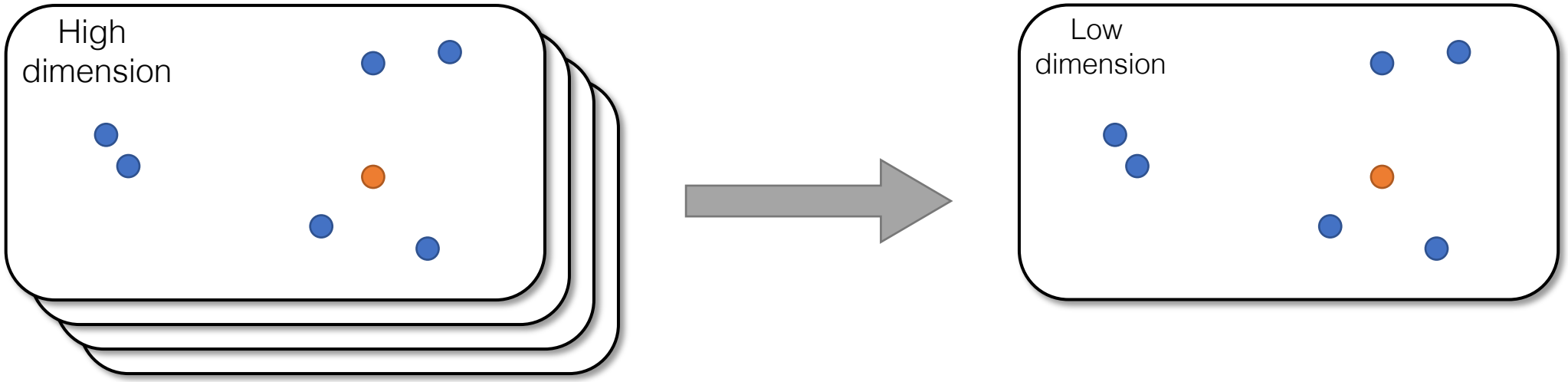
- Minimize the difference between similarities in the high dimension and similarities in the low dimension.
  - Use the sum of Kullback-Leibler divergences as a cost function.
  - Perform gradient descent to minimize the sum of Kullback-Leibler divergences.

Cost function (SNE):

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

Gradient (SNE):

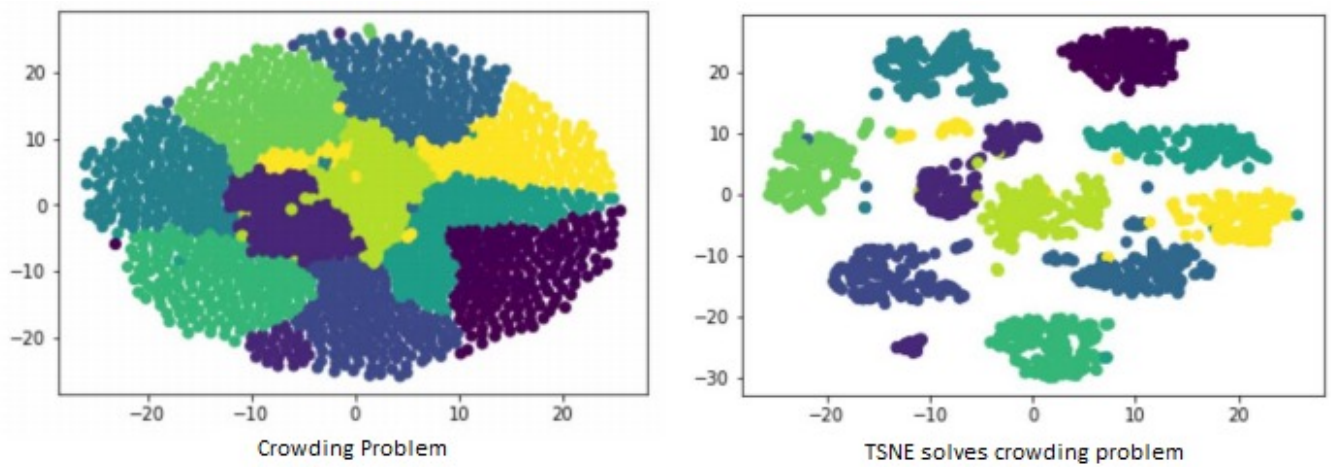
$$\frac{\delta C}{\delta y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$



Hinton & Roweis, *Advances in Neural Information Processing Systems*, 2002
Adapted from van der Maaten, *Google Tech Talk*, 2013

Limitations of SNE

- Similarities based on conditional probabilities are not symmetric (e.g., $p_{j|i} \neq p_{i|j}$), making it difficult to optimize.
- There is a "crowding problem," wherein clusters tend to bunch up near the origin



Solution: **t**-SNE

https://medium.com/rapids-ai/tsne-with-gpus-hours-to-seconds-9d9c17c941db

t-SNE: symmetry of conditional probabilities

Similarities between data points in **high** dimension (t-SNE):

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

This results in a slightly different cost function, which is now simpler to minimize (simpler gradient).

Cost function (t-SNE):

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

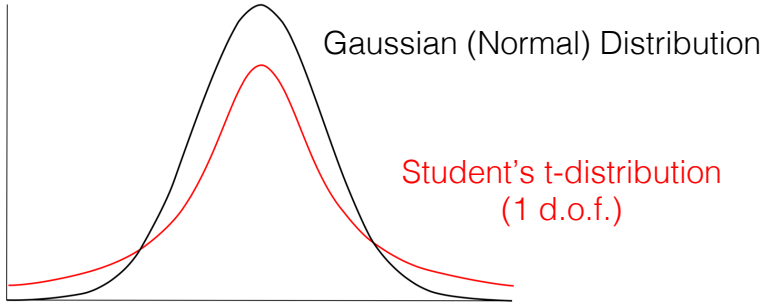
van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

t-SNE: Student's t kernel in the output map

- In the low-dimensional map, replace the Gaussian distribution with a Student's t-distribution (with 1 degree of freedom).
- The t-distribution is shorter at the center and **heavy-tailed**.
- Now, points that are dissimilar in high-dimensional space can be modeled by larger distances in the low-dimensional map.

Similarities between data points in **low** dimension (t-SNE):

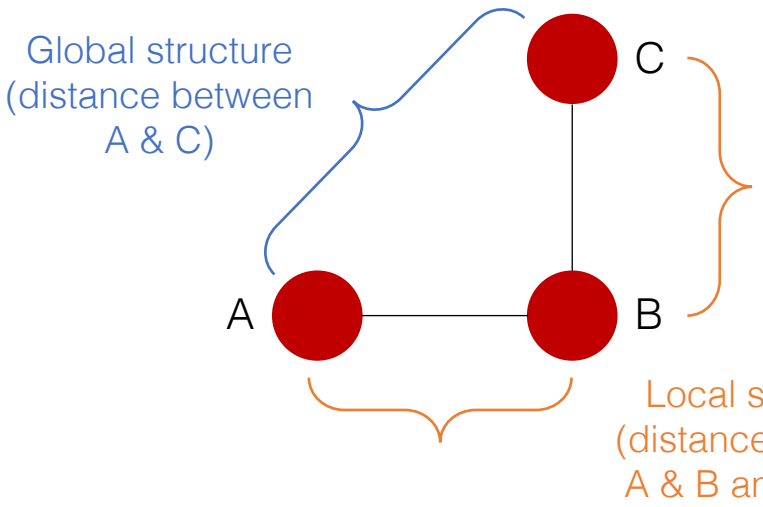
$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k,l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}$$



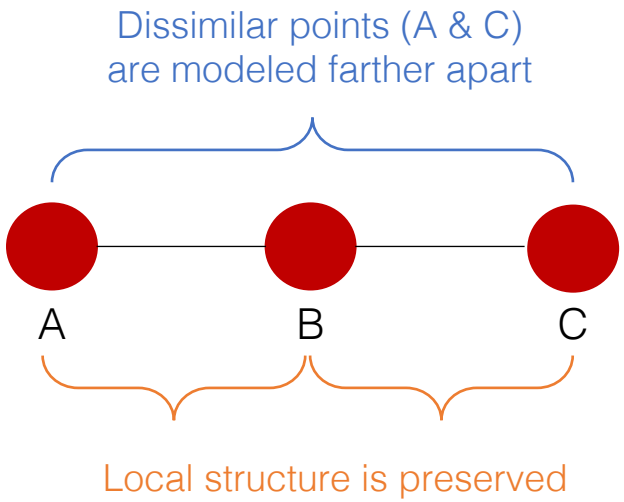
van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013
Adapted from Starmer, *StatQuest*, 2017

Dissimilar points are modeled as farther away

High dimension
(2D)



Low dimension
(1D)



This is enabled by using the t-distribution, thus overcoming the crowding problem and preserving local structure of the data.

Adapted from van der Maaten, *Google Tech Talk*, 2013

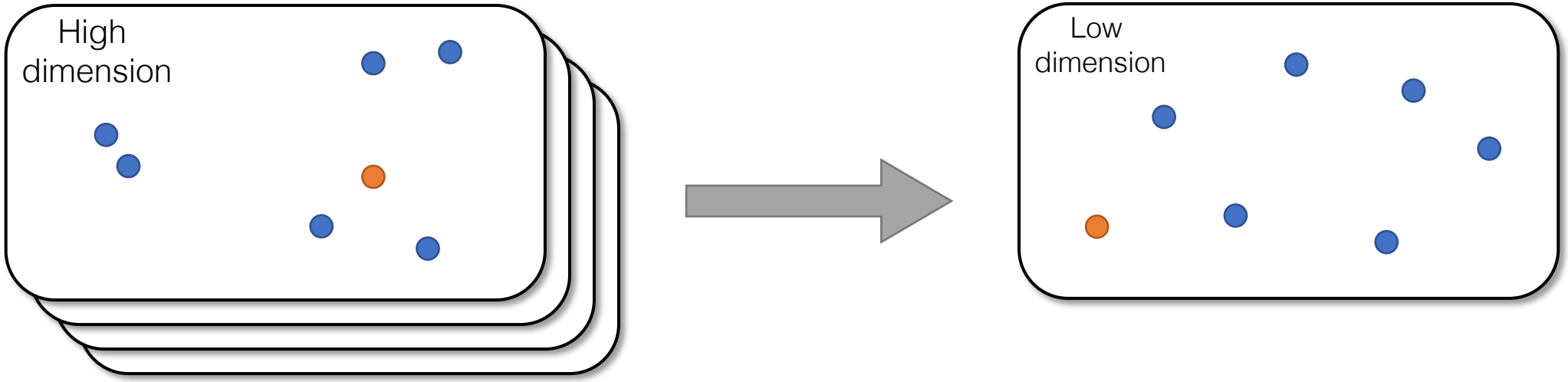
The Kullback-Leibler divergence helps to preserve local structure

- There is a *high* penalty if a *large* p_{ij} is modeled by a *small* q_{ij}.
  - i.e., if two points that are close together in high-dimensional space are modeled as far apart in the low-dimensional map.
- However, there is a *low* penalty if a *small* p_{ij} is modeled by a *large* q_{ij}.

Cost function (t-SNE):

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

(Remember: large probabilities denote that two points are close together.)



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

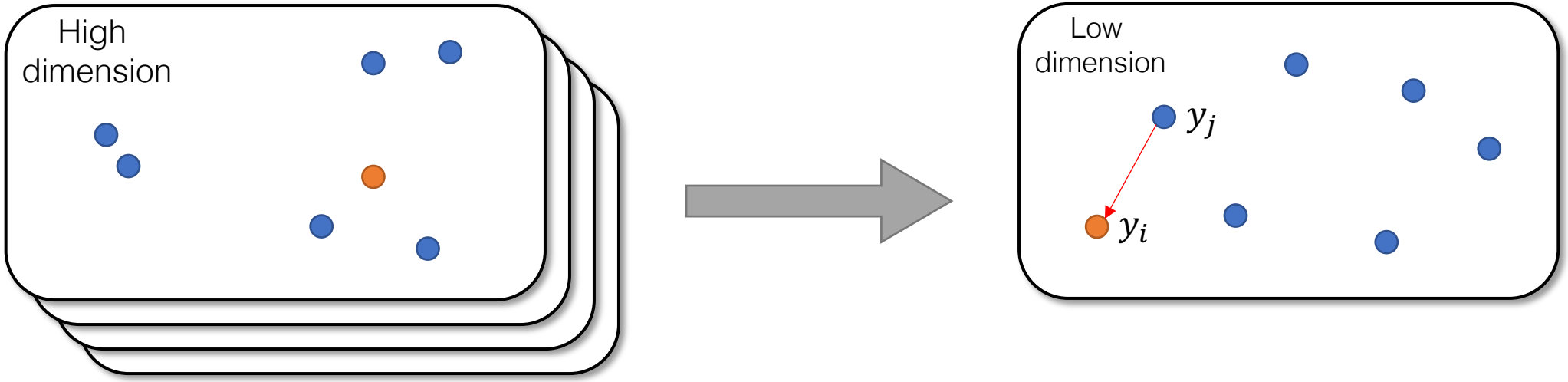
Gradient descent is used to minimize the cost function

- The term $(y_i - y_j)$ represents a spring between a pair of points.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

spring



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

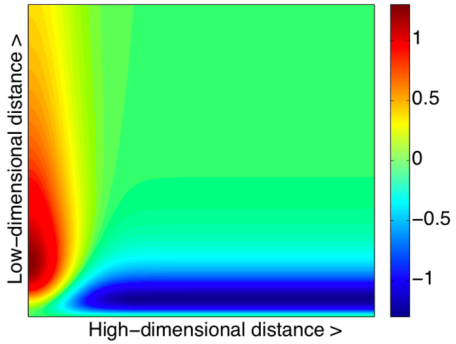
Gradient descent is used to minimize the cost function

- The term $(y_i - y_j)$ represents a spring between a pair of points.
- The term $(p_{ij} - q_{ij})\left(1 + \|y_i - y_j\|^2\right)^{-1}$ represents the exertion or compression of the spring between the pair of points.

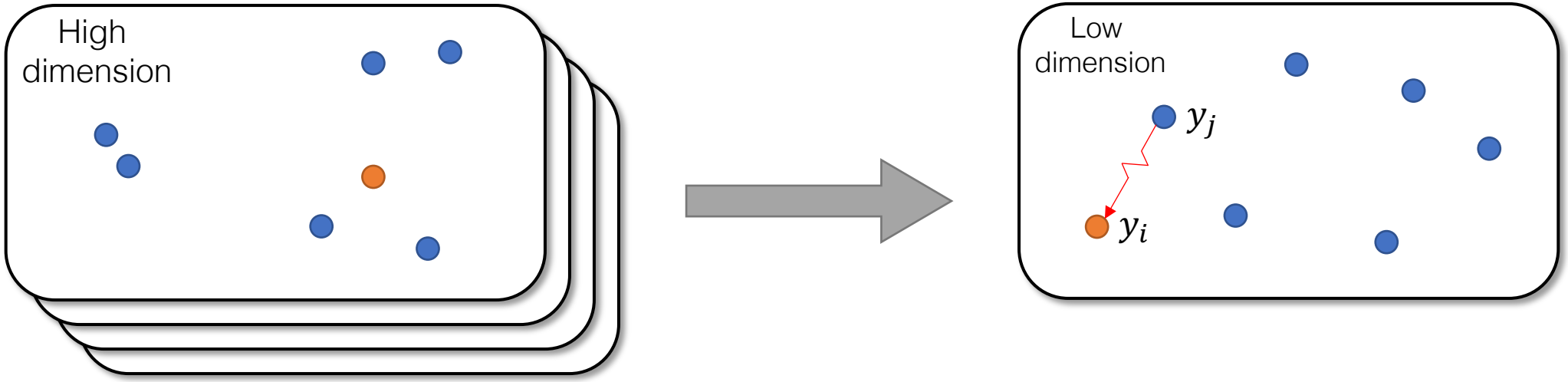
Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})\left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

exertion/compression



Red = attraction between points
Blue = repulsion between points



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

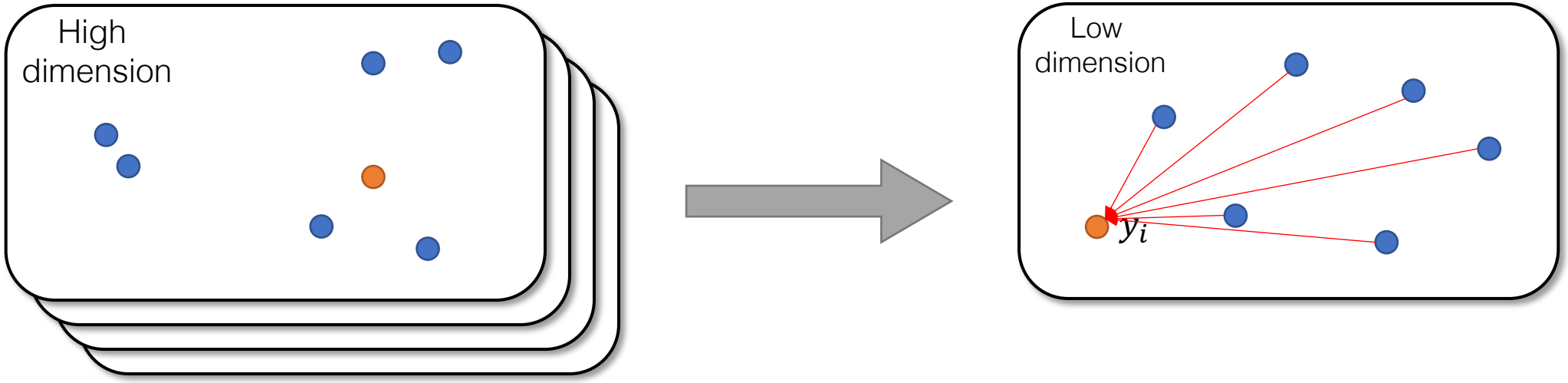
Gradient descent is used to minimize the cost function

- The term $(y_i - y_j)$ represents a spring between a pair of points.
- The term $(p_{ij} - q_{ij})\left(1 + \|y_i - y_j\|^2\right)^{-1}$ represents the exertion or compression of the spring between the pair of points.
- The summation Σ adds up all the forces exerted by each point on y_i.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4\sum_{j \neq i}(p_{ij} - q_{ij})\left(1 + \|y_i - y_j\|^2\right)^{-1}(y_i - y_j)$$

summation



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

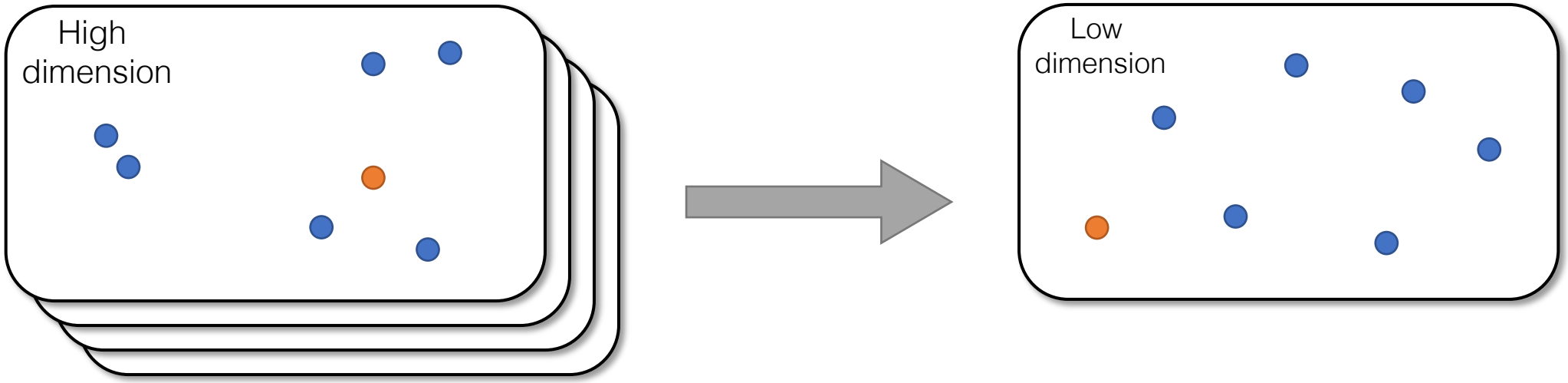
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

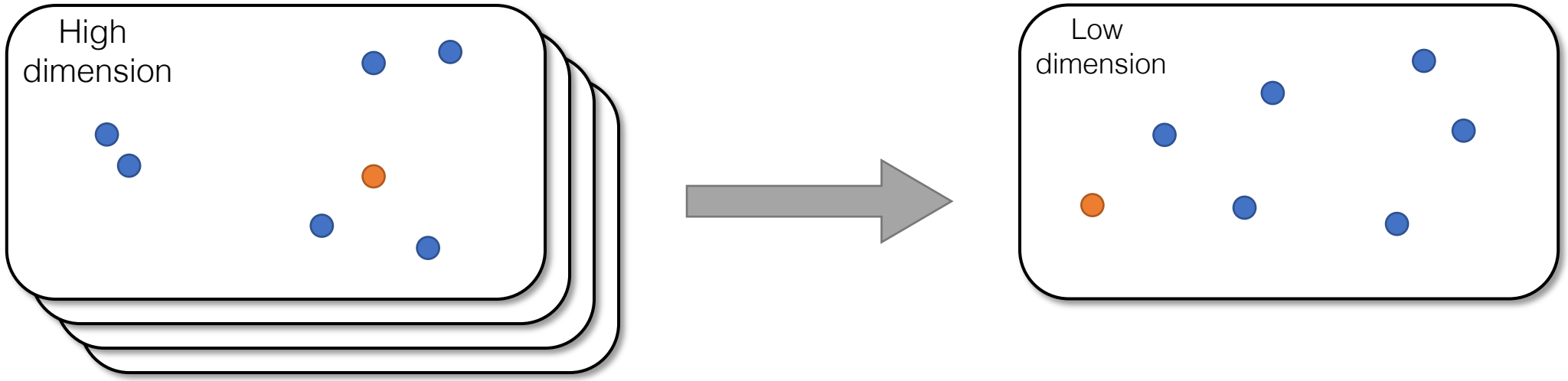
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

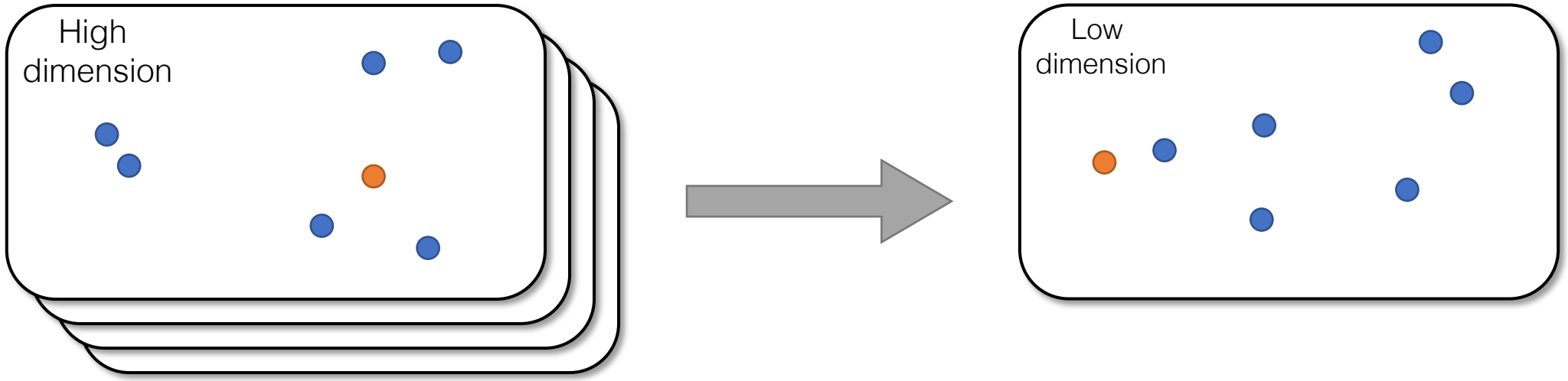
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

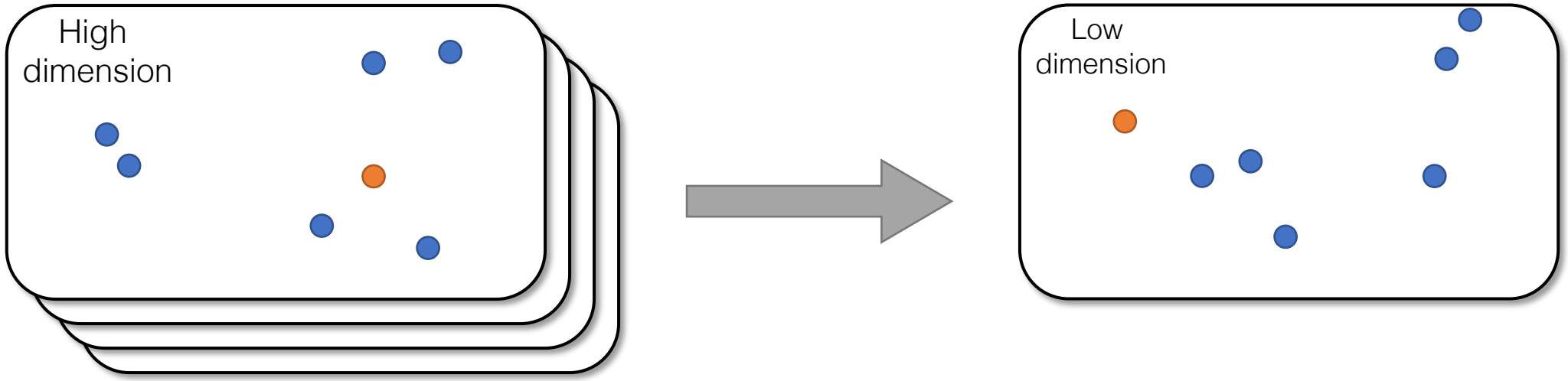
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

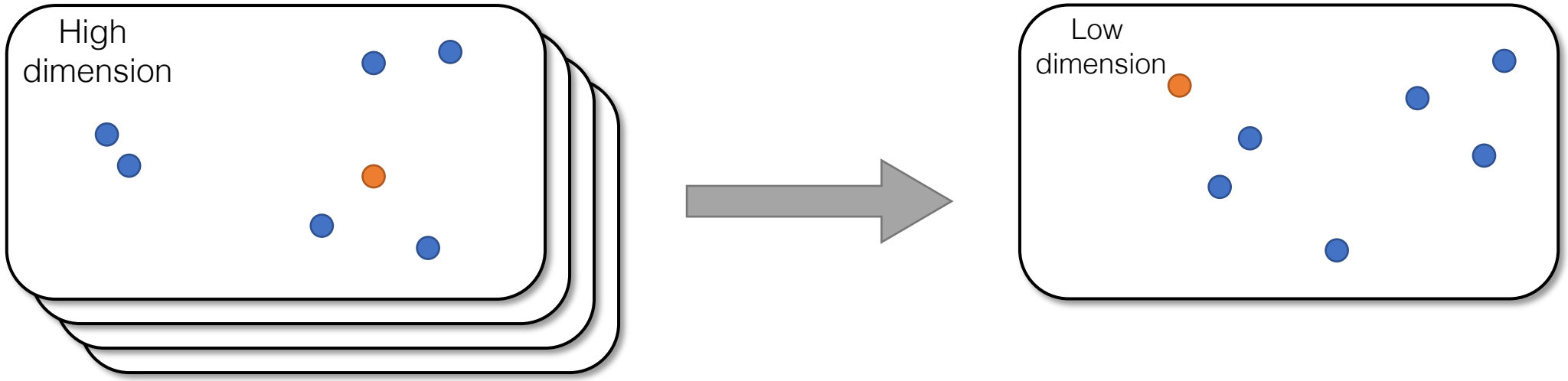
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

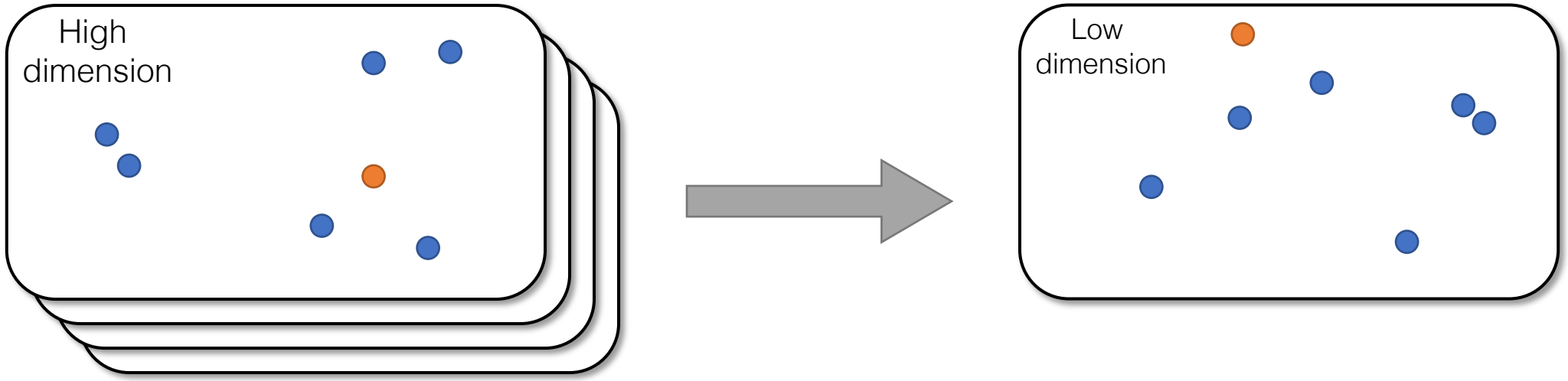
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

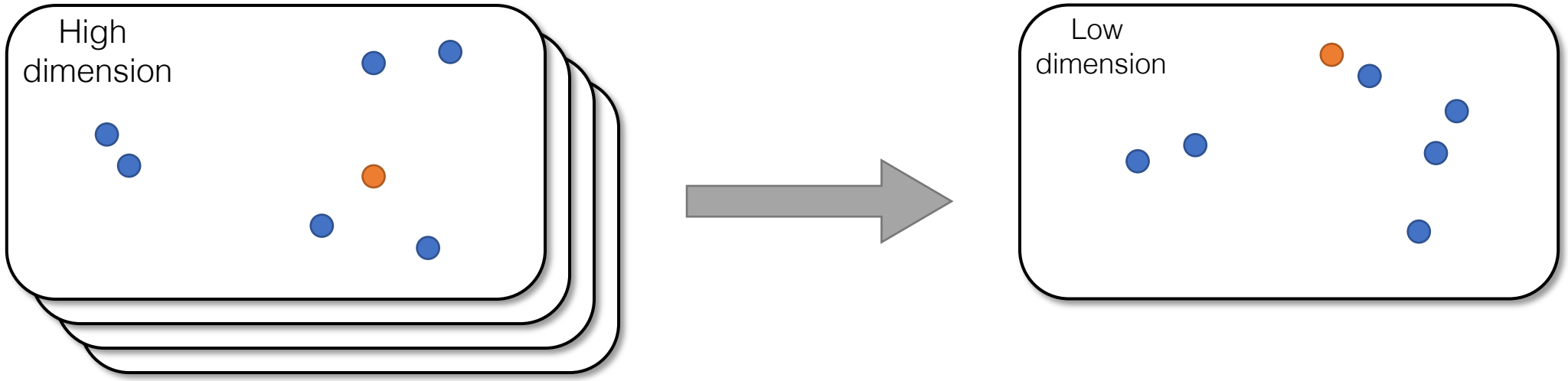
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

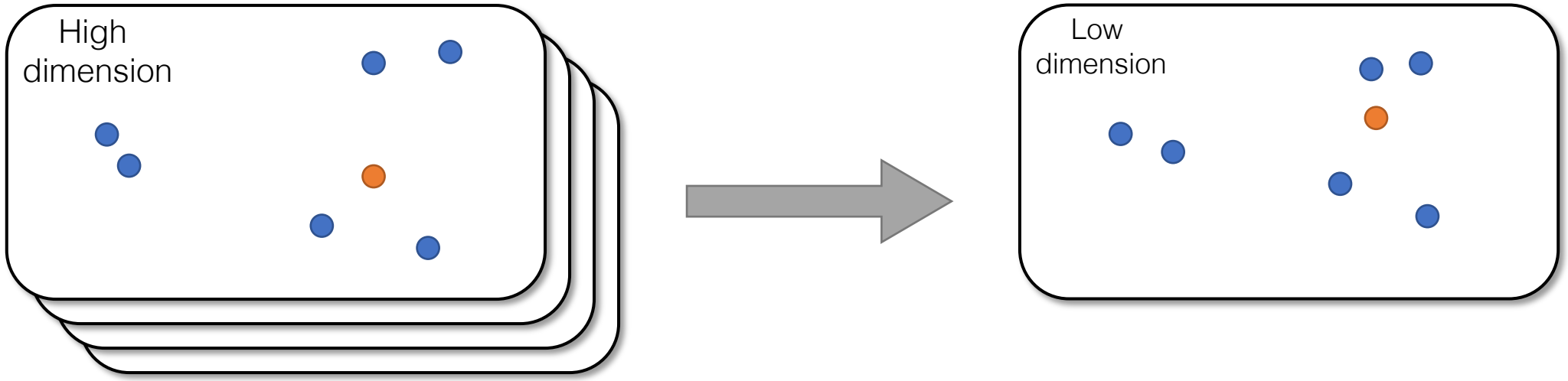
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

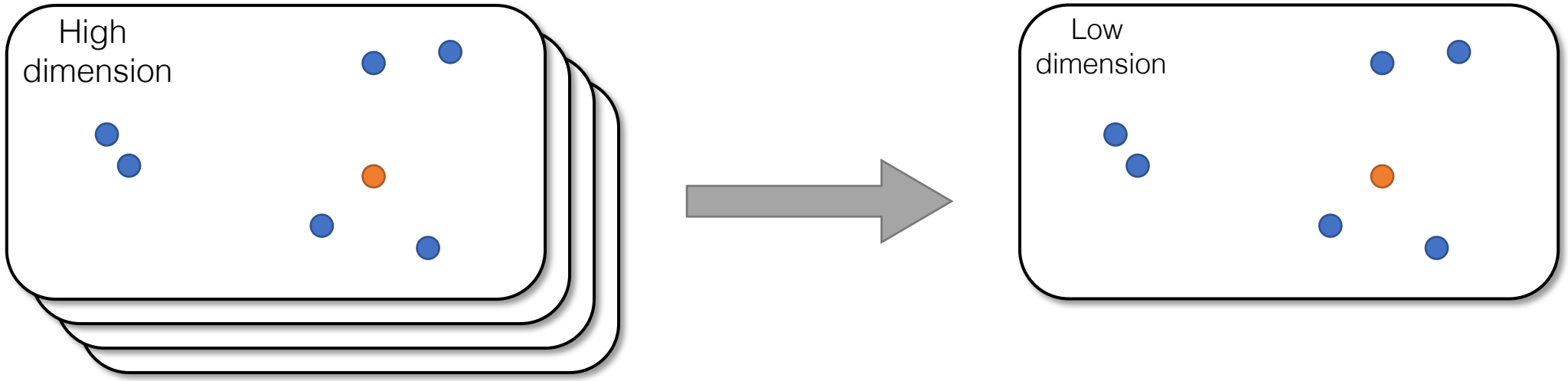
Gradient descent is used to minimize the cost function

- Interpretation of the gradient:
  - Given that each point exerts a force on y_i, calculate the resultant force (via summation) that is required to move y_i so that the cost function is reduced.
- Now, we iteratively adjust data points in the low dimension to minimize the cost function.

Gradient (t-SNE):

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

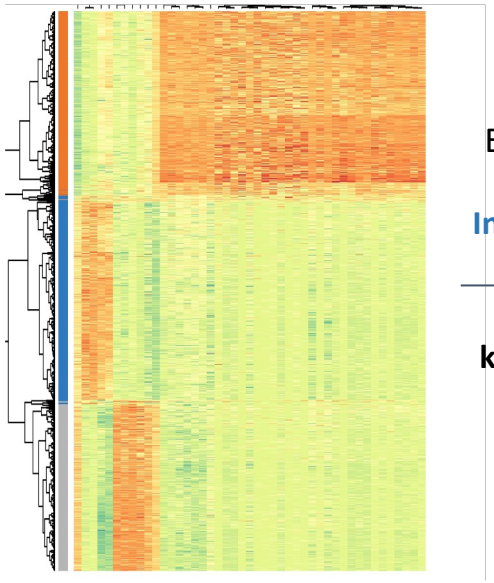
* In the ideal case (in which the similarities in the low-dimensional map perfectly model the similarities in the high-dimensional data), $p_{ij} = q_{ij}$. Therefore, the gradient is zero and points are not moved.



van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
Adapted from van der Maaten, *Google Tech Talk*, 2013

t-SNE Step 1: Calculate affinities

Features (high-dim: p)



n points

(n, p)

Euclidean

Input metric

kNN search

kNN list
(directed, weighted graph)

knn_inds
(n, k)

knn_dists
(n, k)

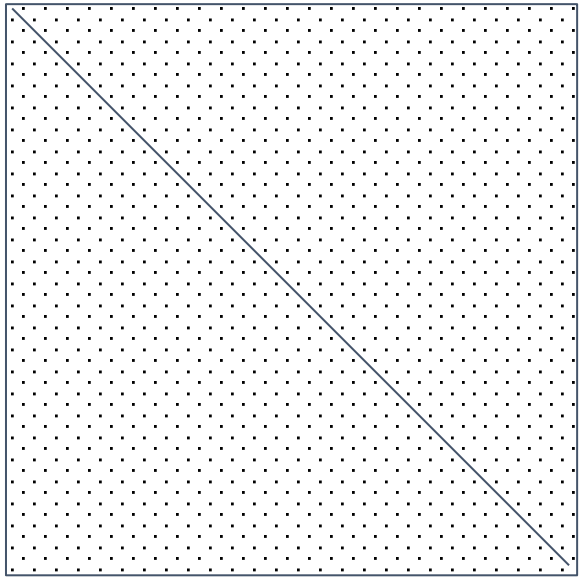
$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$$

Input kernel

Symmetrization

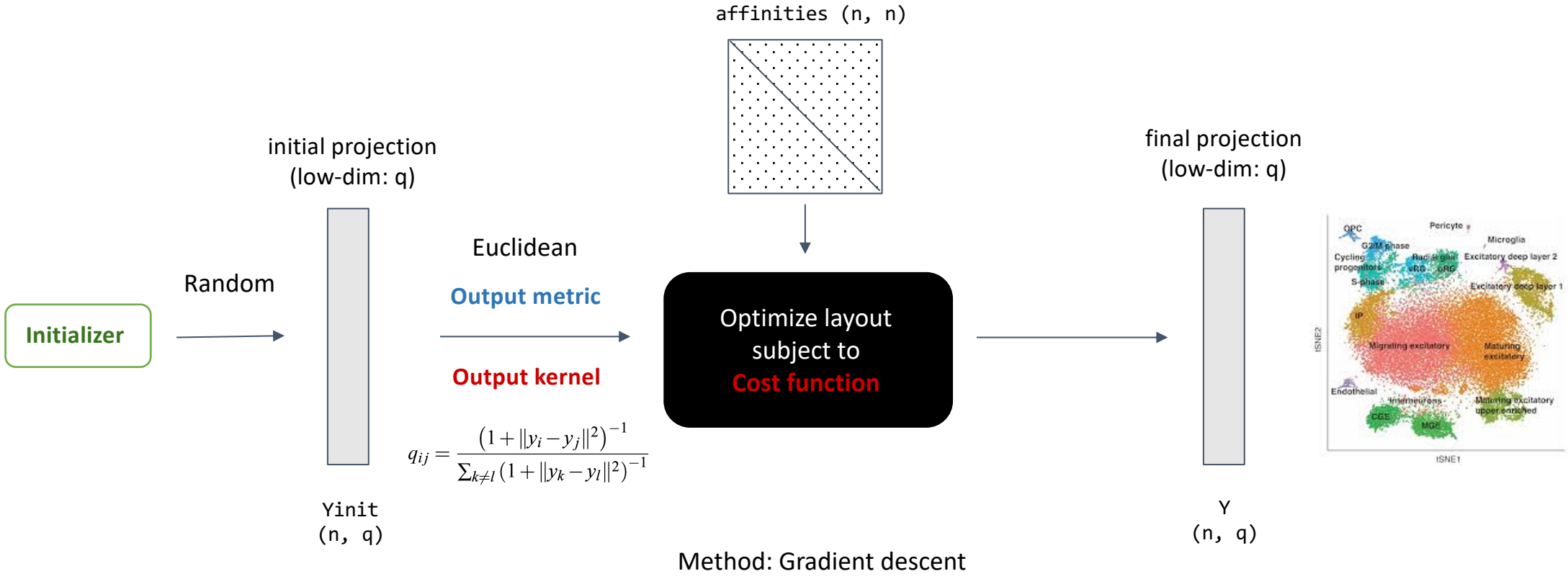
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Sparse, symmetric **affinity** matrix
(undirected, weighted graph)



(n, n)
sparse

t-SNE Step 2: Initialize and optimize the low-dim layout



Cost function (t-SNE):

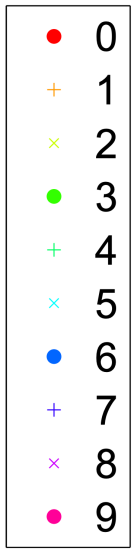
$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Gradient (t-SNE):

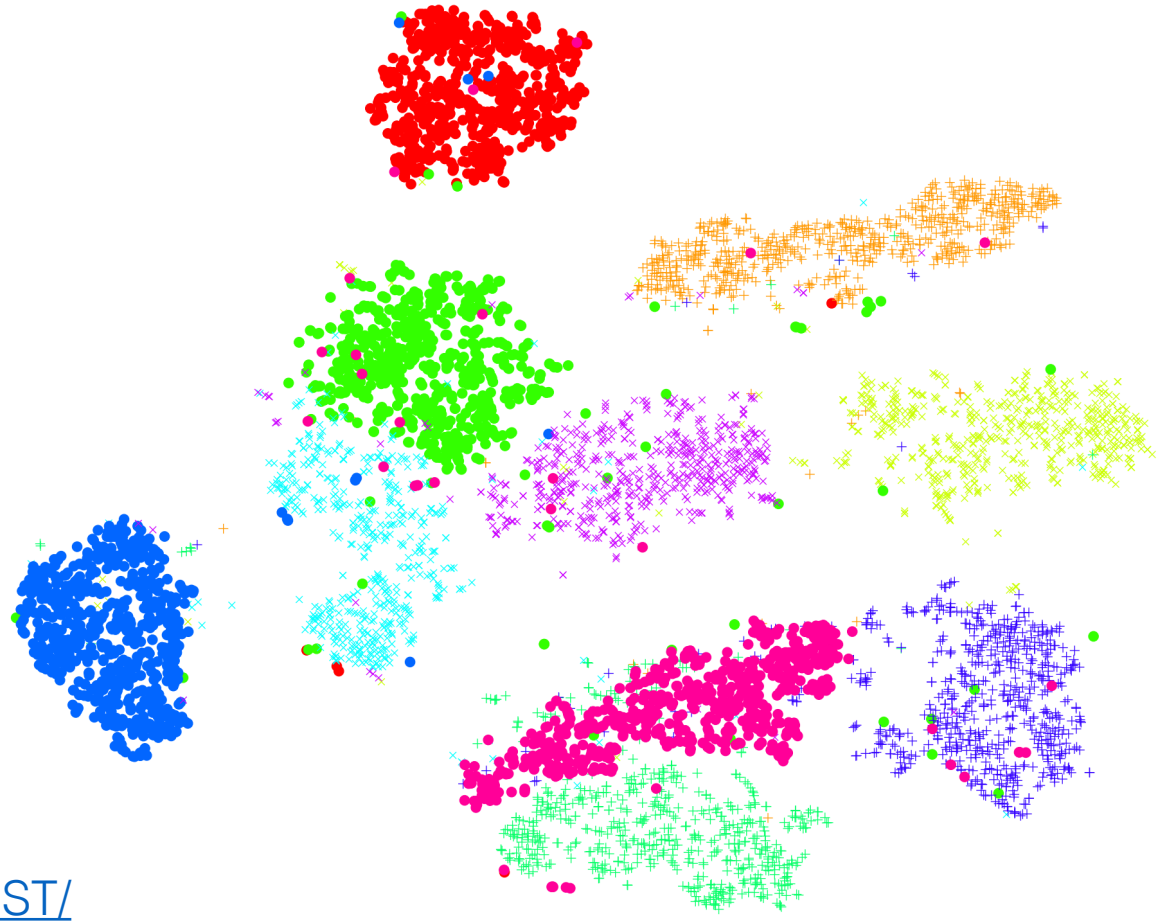
$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) \left(1 + \|y_i - y_j\|^2\right)^{-1} (y_i - y_j)$$

t-SNE performs much better on the MNIST dataset

MNIST dataset



t-SNE, applied to MNIST



t-SNE on MNIST, in action:
http://colah.github.io/posts/2014-10-Visualizing-MNIST/

LeCun et al., *Proc. of the IEEE*, 1998
van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008

Improving performance: calculating repulsive forces is inefficient

$$4\left(\sum_{j\neq i} p_{ij}q_{ij}Z(\mathbf{y}_i-\mathbf{y}_j)-\sum_{j\neq i} q_{ij}^2 Z(\mathbf{y}_i-\mathbf{y}_j)\right)$$

Normalization constant

$$Z=\sum_{k\neq l}(1+\|\mathbf{y}_k-\mathbf{y}_l\|^2)^{-1}$$

Attractive forces:
|Charge(i, j)| ~ (Input affinity) * (Output affinity)
Input affinity = 0 when i & j are not neighbors in input space.
So **attraction forces are efficient** to compute O(kN) thanks to sparsification of the input.

Repulsive forces don't depend on the input affinities.
All points in output space repel all other points!

So this calculation doesn't get sparsified -- **inefficient** $O(N^2)$

Unless we find an **approximation** trick...

Published: 04 December 1986

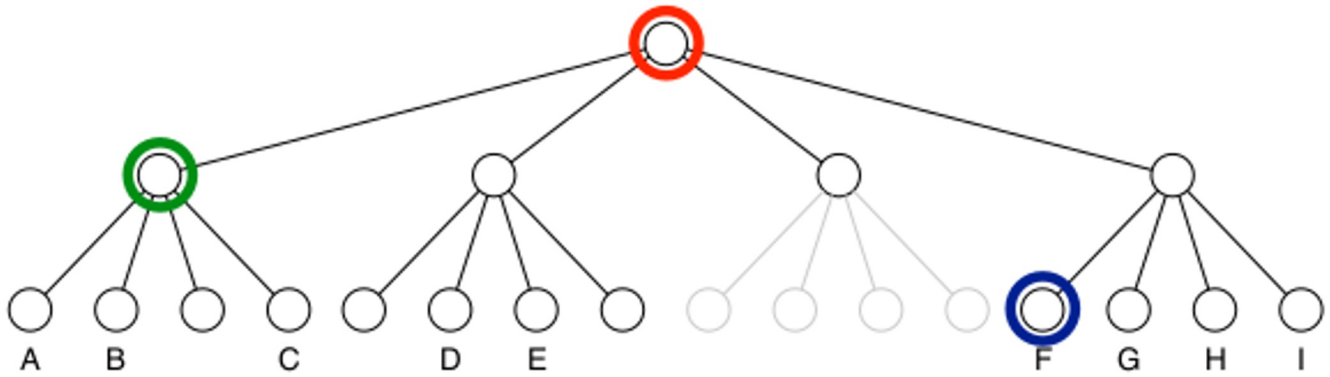
A hierarchical *O(N log N)* force-calculation algorithm

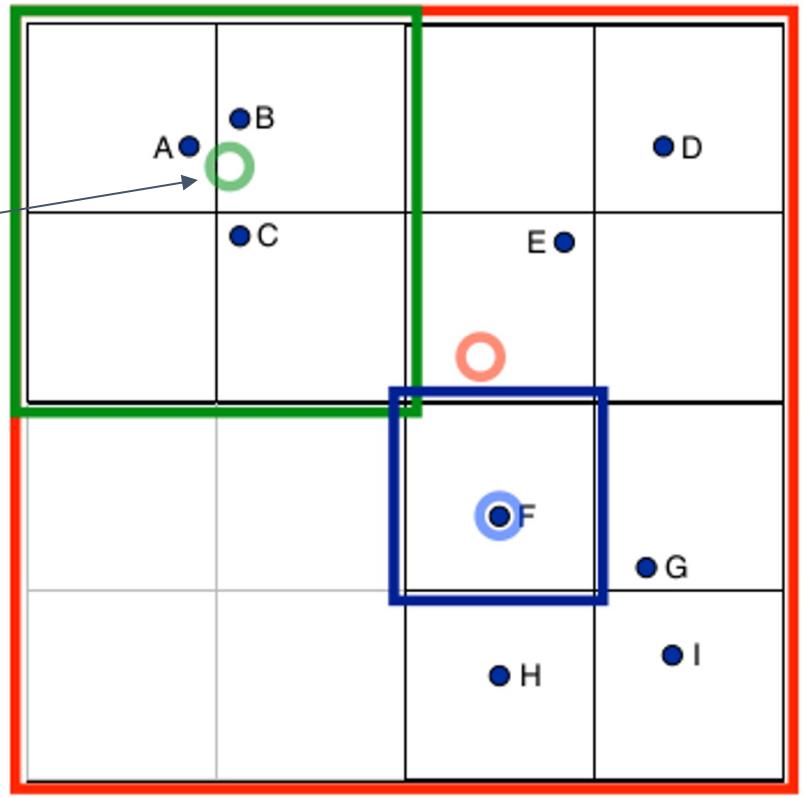
Josh Barnes & Piet Hut

Nature **324**, 446–449(1986) | Cite this article

Abstract

Until recently the gravitational *N*-body problem has been modelled numerically either by direct integration, in which the computation needed increases as N^2, or by an iterative potential method in which the number of operations grows as $N \log N$. Here we describe a novel method of directly calculating the force on *N* bodies that grows only as $N \log N$. The technique uses a tree-structured hierarchical subdivision of space into cubic cells, each of which is recursively divided into eight subcells whenever more than one particle is found to occupy the same cell. This tree is constructed anew at every time step, avoiding ambiguity and tangling. Advantages over potential-solving codes are: accurate local interactions; freedom from geometrical assumptions and restrictions; and applicability to a wide class of systems, including (proto-)planetary, stellar, galactic and cosmological ones. Advantages over previous hierarchical tree-codes include simplicity and the possibility of rigorous analysis of error. Although we concentrate here on stellar dynamical applications, our techniques of efficiently handling a large number of long-range interactions and concentrating computational effort where most needed have potential applications in other areas of astrophysics as well.





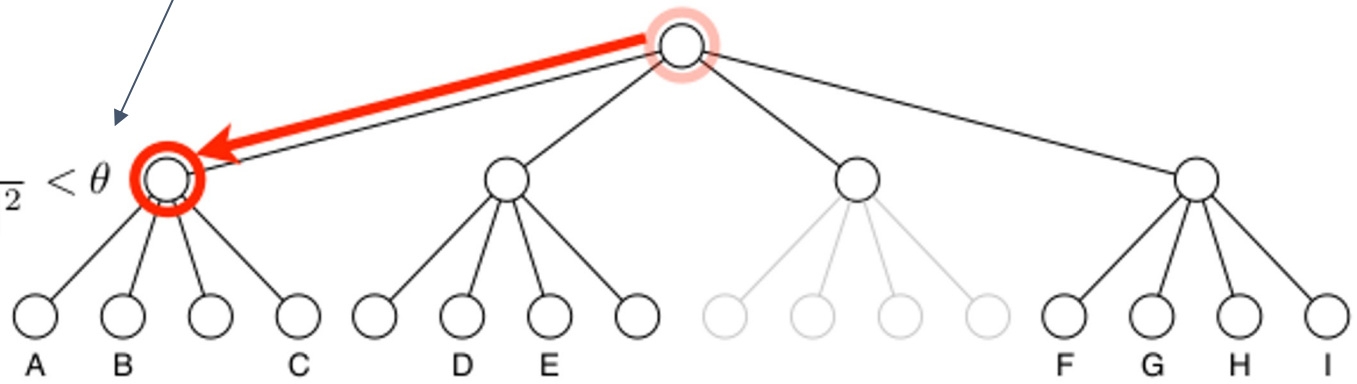
Center of mass and total mass (# points) gets stored in the node

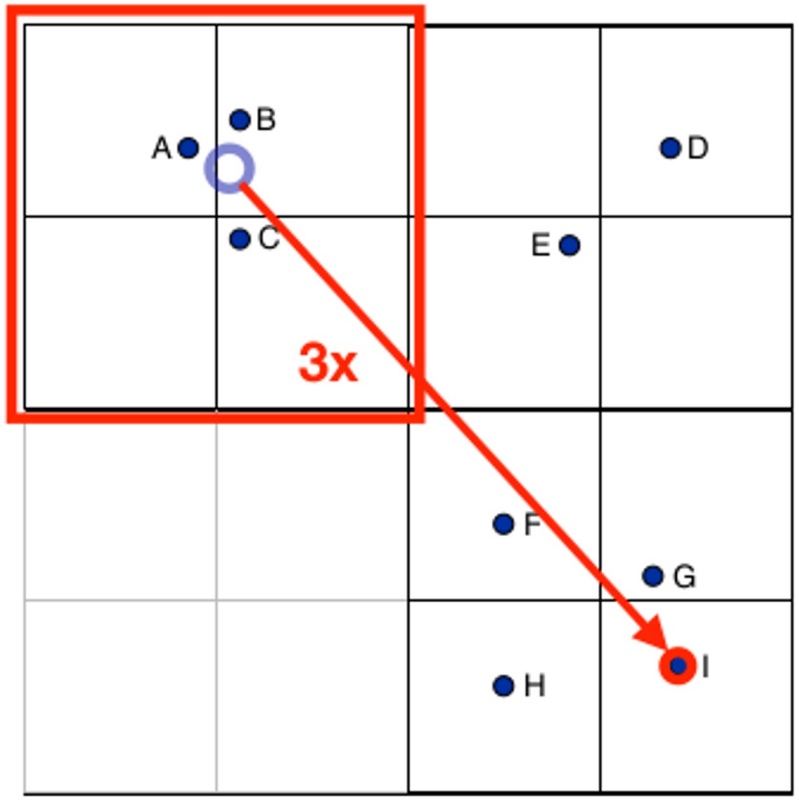
Diagonal length of cell

Coarse graining threshold

$$\frac{r_{cell}}{\|\mathbf{y}_i - \mathbf{y}_{cell}\|^2} < \theta$$

Distance from I to cell's center of mass





To calculate grad for point I:

- Do depth-first search and stop whenever the criterion is satisfied.
- Add the contribution to the overall gradient for I.

Parameters of t-SNE

- Perplexity ($Perp$)
  - Determines the widths of the Gaussian kernels
  - The default number of neighbors used in most implementations is 3 * perplexity.
- Number of iterations (T)
  - The number of times to adjust low-dimensional points and perform updates to the gradient when minimizing the cost function.
- Learning rate (η)
  - The amount by which the gradient is updated during each iteration.
- Momentum ($\alpha(t)$)
  - The degree to which updates to the gradient keep moving in the same direction.

Pseudocode for the t-SNE algorithm

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for *t=1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

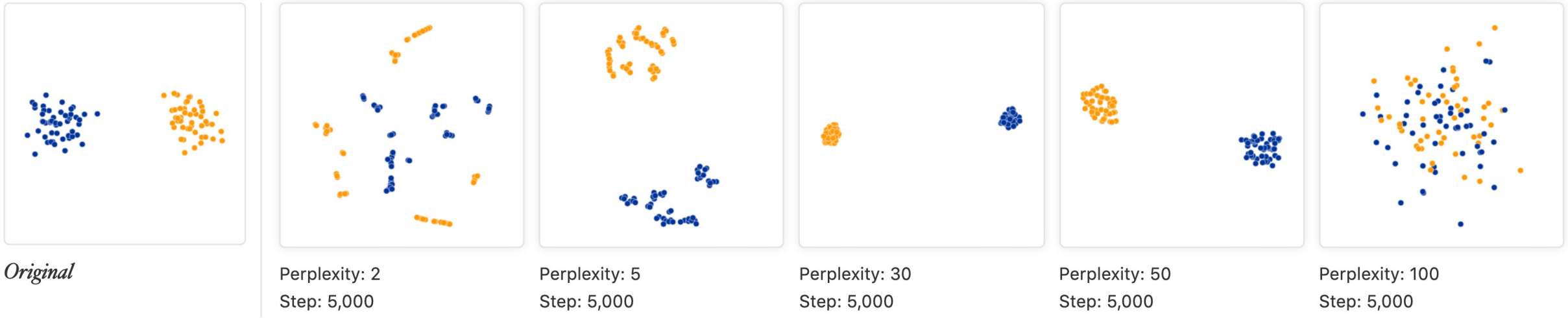
t-SNE parameters, in action:
https://distill.pub/2016/misread-tsne/

van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008
http://v9docs.flowjo.com/html/tsne.html

Considerations when implementing t-SNE

Toy dataset: 2 clusters, 50 points each, color-coded

Parameters: constant number of iterations, increasing perplexity



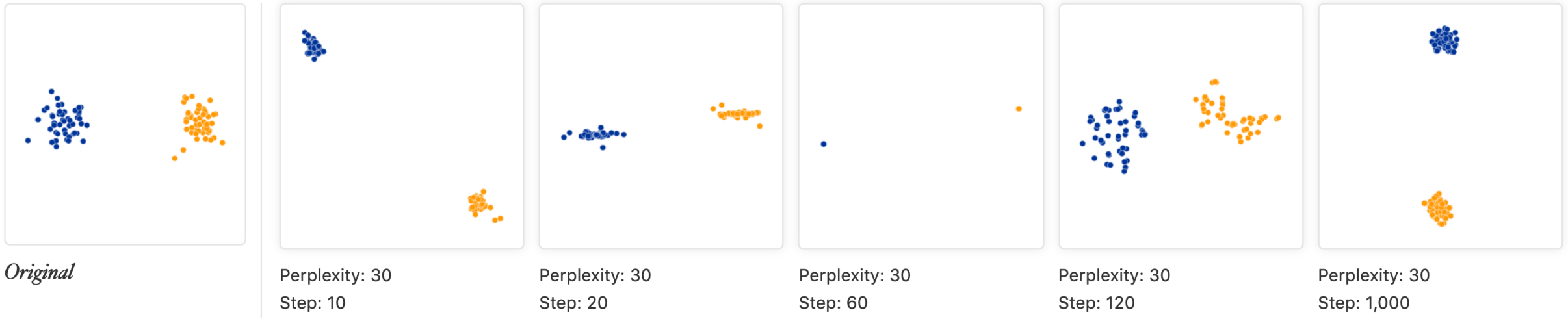
$\Longrightarrow$ Perplexity values should be between 5 and 50 (inclusive), and always less than the number of points in the dataset. Always try running t-SNE with different perplexities to find the optimal value.

https://distill.pub/2016/misread-tsne/

Considerations when implementing t-SNE

Toy dataset: 2 clusters, 50 points each, color-coded

Parameters: constant perplexity, increasing number of iterations



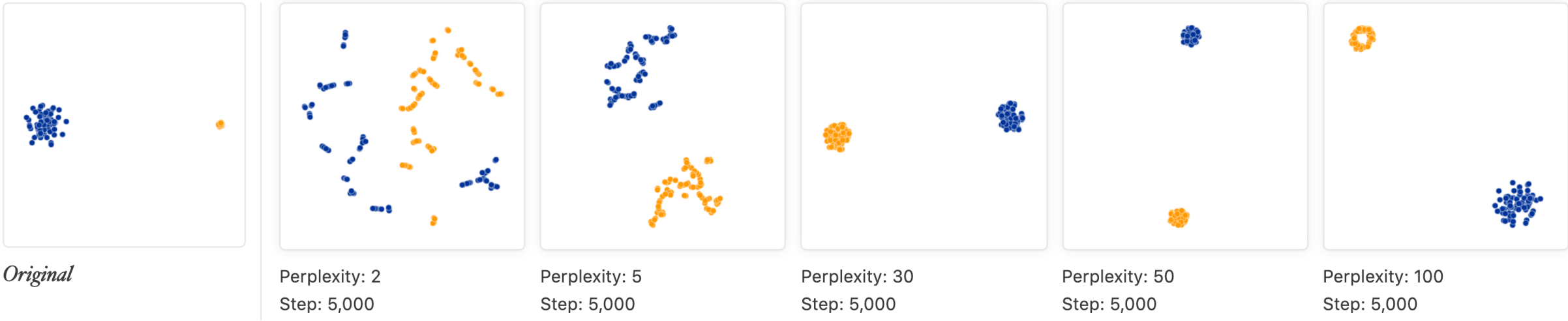
$\Longrightarrow$ Always run t-SNE for enough number of iterations until the resulting shape stabilizes (i.e., reaches convergence) – the number of iterations required may vary for different datasets.

https://distill.pub/2016/misread-tsne/

Considerations when implementing t-SNE

Toy dataset: 2 clusters, 50 points each, color-coded; one cluster 10 times as dense as the other

Parameters: constant number of iterations, increasing perplexity



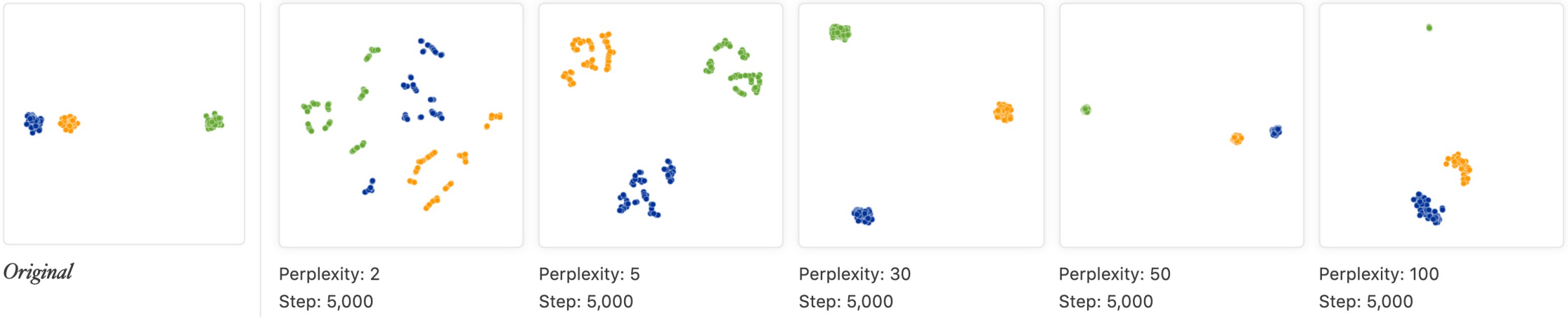
$\Longrightarrow$ Do not make conclusions regarding the relative sizes of clusters from a t-SNE plot – they may not accurately reflect the relative sizes of clusters in the original dataset.

https://distill.pub/2016/misread-tsne/

Considerations when implementing t-SNE

Toy dataset: 3 clusters, 50 points each, color-coded; one pair 5 times as far apart as the other

Parameters: constant number of iterations, increasing perplexity



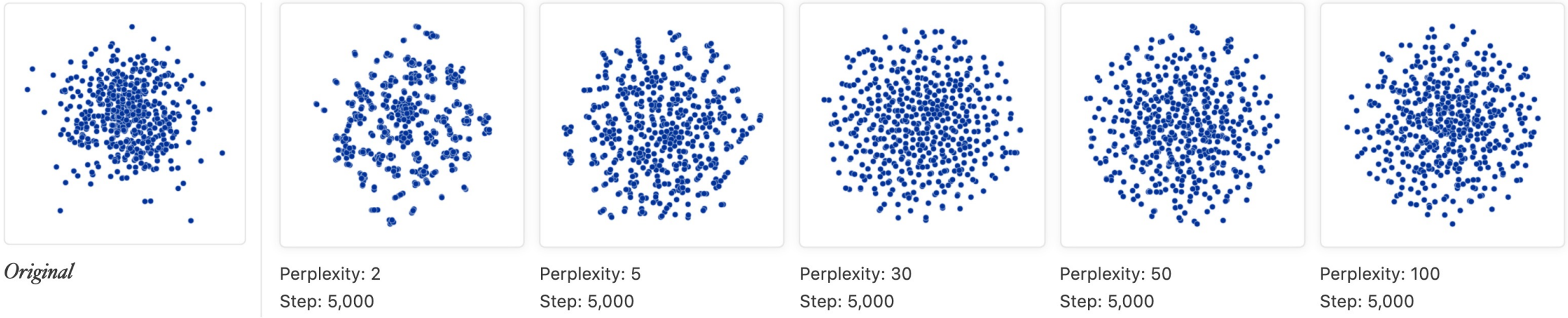
$\Longrightarrow$ Do not make conclusions regarding the relative distances between clusters from a t-SNE plot – they may not accurately reflect the relative distances between clusters in the original dataset.

https://distill.pub/2016/misread-tsne/

Considerations when implementing t-SNE

Toy dataset: random data, 500 points draw from a Gaussian distribution

Parameters: constant perplexity, increasing number of iterations



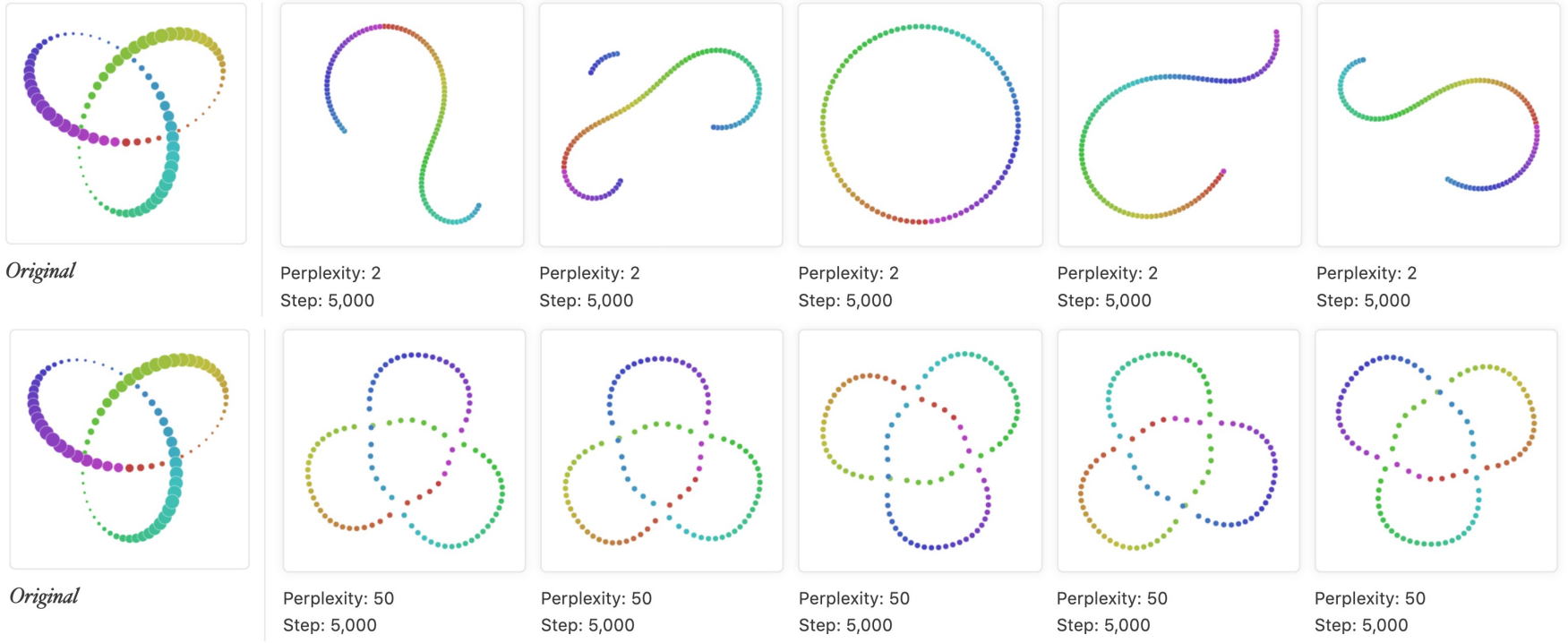
$\Longrightarrow$ Be cautious when interpreting t-SNE plots to avoid concluding that there are patterns in data that is, in reality, just noise.

https://distill.pub/2016/misread-tsne/

Considerations when implementing t-SNE

Toy dataset: trefoil knot

Parameters: constant perplexity, constant number of iterations, run multiple times

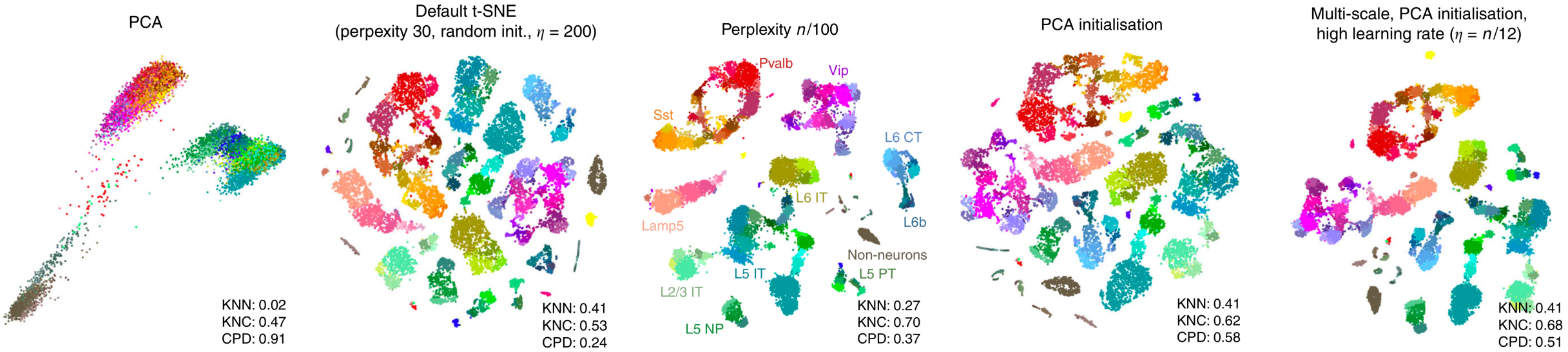


$\Longrightarrow$ Always run t-SNE multiple times with a fixed set of parameters to make sure the output is similar for each run.

https://distill.pub/2016/misread-tsne/

Applications of t-SNE in bioinformatics

Dataset: single-cell RNA-seq (scRNA-seq) of 23,822 (*n*) cells from adult mouse cortex, 133 cell types (Tasic et al., *Nature*, 2018)



Warm colors → inhibitory neurons

Cold colors → excitatory neurons

Brown/gray → non-neural cells

Kobak and Berens, *Nat Comms*, 2019

Additional resources for t-SNE

- **Official t-SNE webpage**:
  - https://lvdmaaten.github.io/tsne/
- **Original paper**: Visualizing Data using t-SNE (van der Maaten and Hinton, *J. Mach. Learn. Res.*, 2008)
  - https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf
- **t-SNE Google Tech Talk**, by the first author Laurens van der Maaten:
  - https://www.youtube.com/watch?v=RJVL80Gg3lA
- **StatQuest by Josh Starmer**: 2-D example of t-SNE
  - https://www.youtube.com/watch?v=NEaUSP4YerM

Part 2: Uniform Manifold Approximation and Projection (UMAP)

Overview of UMAP

- UMAP was developed based on principles from algebraic topology and differential geometry, advanced fields in pure mathematics.

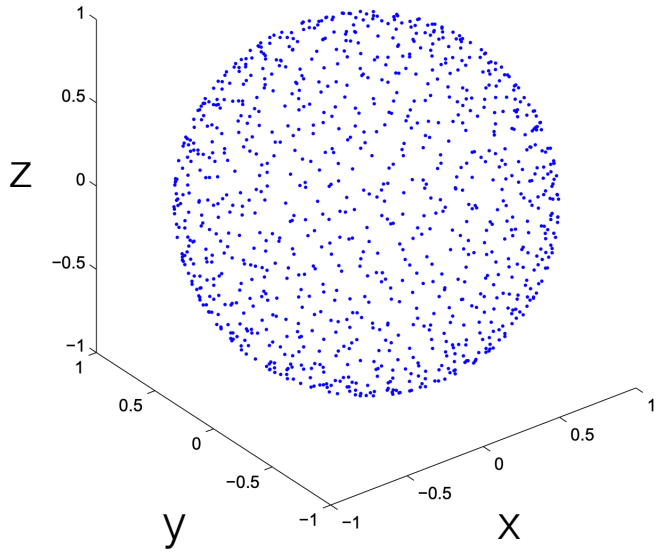
- UMAP does very similar things to t-SNE. While t-SNE is motivated largely by heuristics, UMAP has a stronger theoretical justification behind its choices.

- The general steps of UMAP are as follows:
  - Use data points and their nearest neighbors to estimate the local spatial structure around each point.
  - Use these to generate a weighted graph to approximate the connectivity of the manifold.
  - Project the data onto the lower dimension such that the topology of the projection is similar to that of the original data.

McInnes et al., *arXiv*, 2018

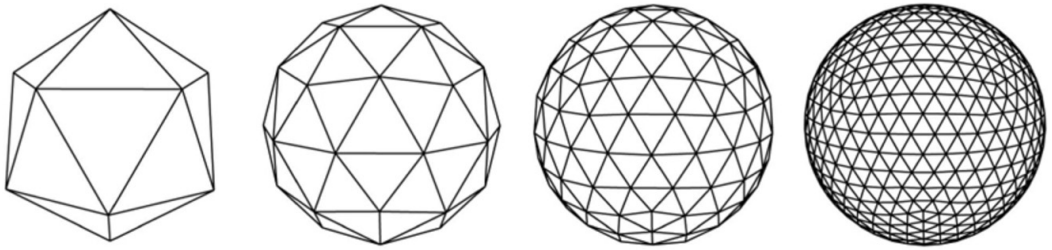
Approximation of a manifold

Suppose we are given the following dataset on a sphere:



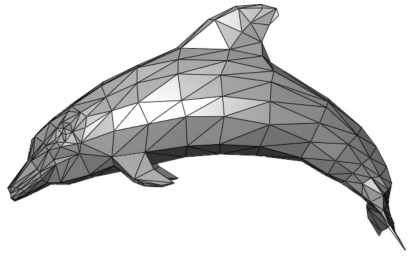
UMAP assumes that the data lies on a manifold. Our goal is *to approximate the manifold on which the manifold resides*. How can we achieve this?

We can use “simplicial complexes” to approximate the underlying topological structure of the manifold.



Note that the accuracy of our approximation is dependent on the size of simplices used.

We can apply the same principles for more complex manifolds.

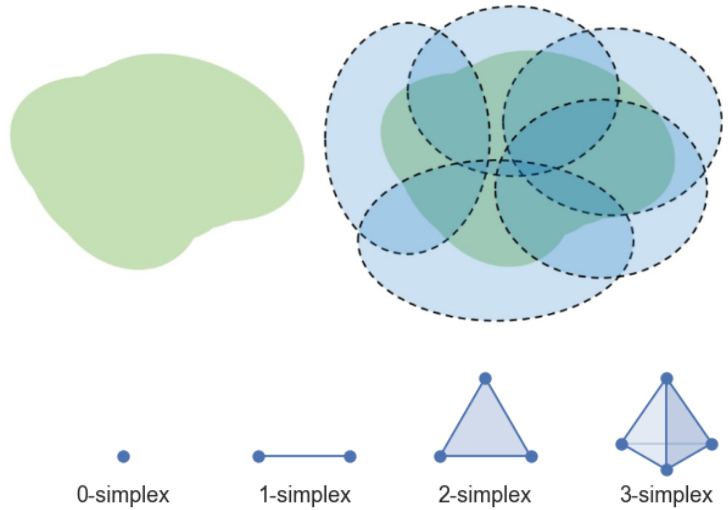


Cavoretto and De Rossi, *J. Comput. Appl. Math*, 2010
http://clivebest.com/blog/?p=8119
https://en.wikipedia.org/wiki/Triangulation_(topology)

Using open covers for manifold approximation

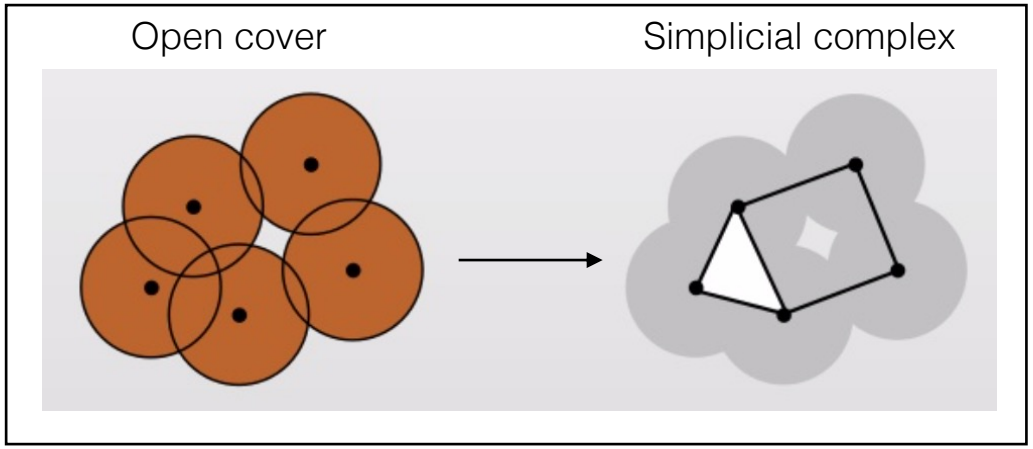
- An open cover of a topological space is a collection of open sets whose union contains that space.
- The quality of the cover will determine how accurate the approximation of the manifold is; the finer the resolution of the cover, the better the approximation will be.

Open cover of a space:



- A simplicial complex can be constructed from an open cover as follows:
  - Each set in the cover is a 0-simplex.
  - If two sets have a non-empty intersection, create a **1-simplex**. i.e. connect the points by an edge
  - If three sets have a non-empty intersection, create a 2-simplex.
  - And so on, for higher dimensions.

Construct a simplicial complex from an open cover:

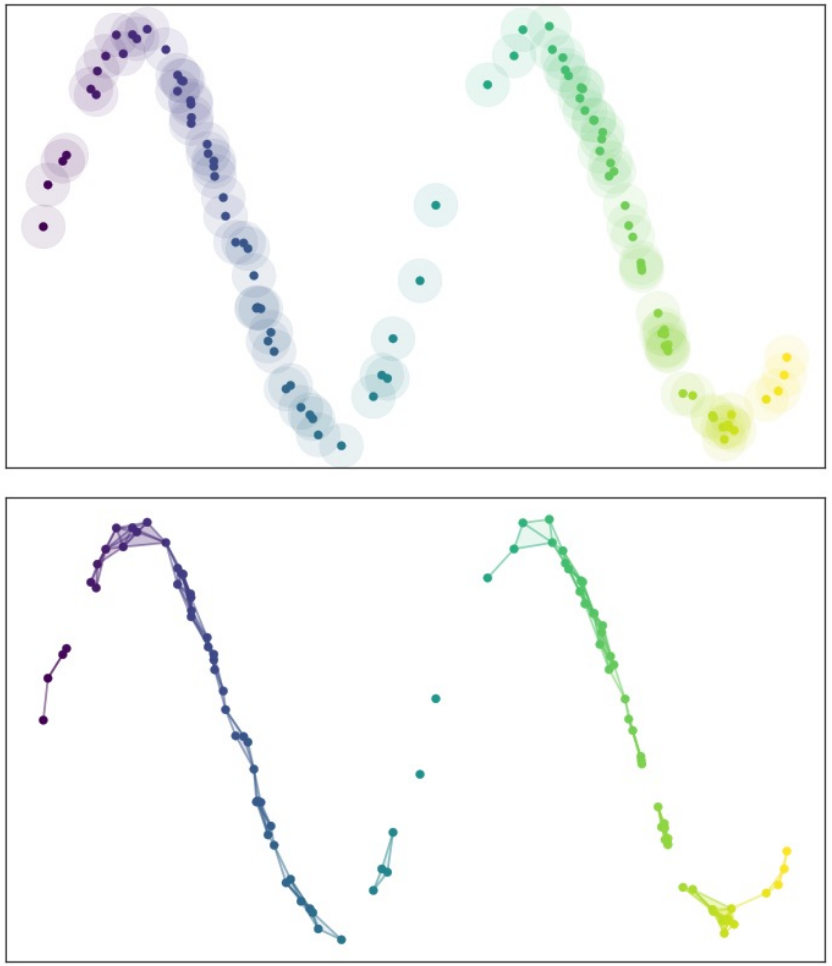


Collins, *Notes on Topology*, 2017
Sheehy, *INRIA Saclay*, 2012

The nerve theorem

- The simplicial complex constructed from an open cover is also called the *nerve* of the cover.
- The nerve theorem states that if we have a "good" open cover of a space and we construct the nerve of that cover, it will capture the important topology of that space.
- In the example to the right, there are "clumps" and "gaps" in the nerve. Therefore, the open cover is not of good quality, and we only capture some of the topology of the space.
- In order to accurately approximate the manifold, we need to construct a better open cover.

Example of a poor open cover and its resulting nerve:



McInnes et al., *arXiv*, 2018

Constructing a better open cover

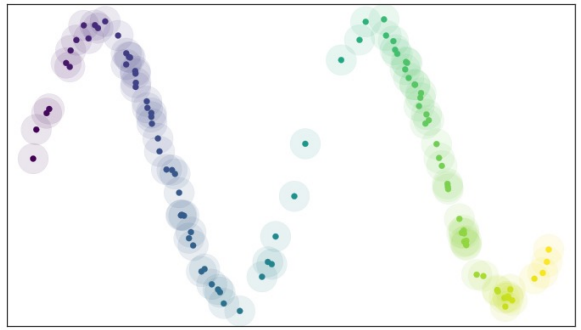
- If we assume data is uniformly distributed on the low-dimensional manifold then every open ball of fixed volume centered at any point p should contain approximately the same number of points.

- Therefore, we use the nearest neighbors around each point p to estimate the size of its local ball.

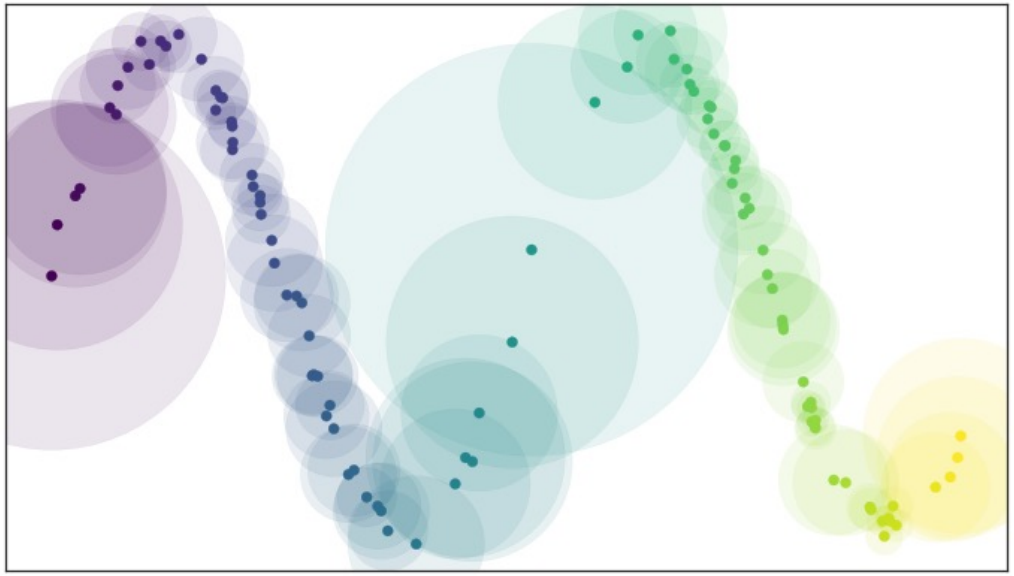
- Though these open balls look different in the ambient space, they are interpreted as having the same radius within the locally-defined geodesic metric around each point.

- If the data appears not to be uniformly distributed, then this can be explained by the fact that the curvature (and therefore, the metric) is varying across the manifold, and this is exactly what we want to capture.

Before:



After:



McInnes et al., *arXiv*, 2018

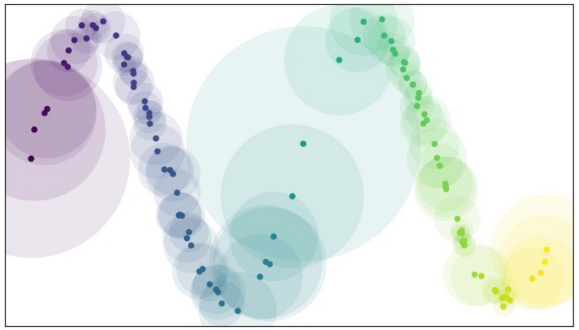
Make the cover “fuzzy”

- We define a weight function that assigns to every neighbor of point *p* a **“membership weight”** to the open ball centered at *p*.

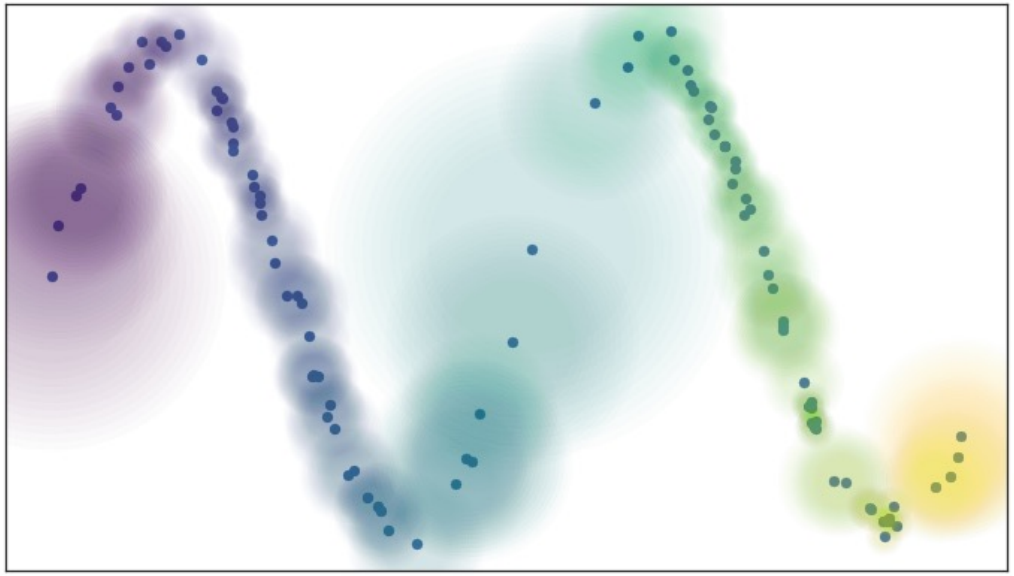
- Each weight is a value in [0, 1].
  - The closer the weight is to 1, the stronger the membership.
  - The closer the weight is to 0, the weaker the membership.

- These fuzzy membership weights serve as our “affinities” in UMAP. As with t-SNE, we calculate them using a kernel function.

Before:



After:



McInnes et al., *arXiv*, 2018

Keep the cover connected

- The second assumption of UMAP is that **no point in the data is isolated**, and thus the manifold is locally connected.

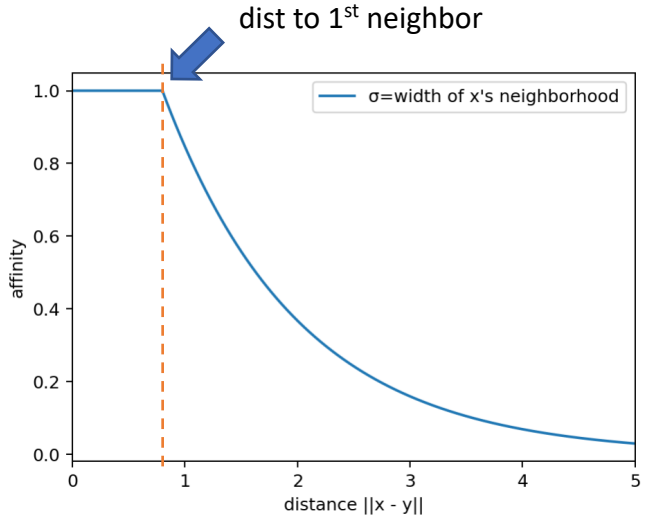
- We realize this assumption by first extending the open set centered at point ρ to its nearest neighbor, then defining the membership decay beyond the nearest neighbor.

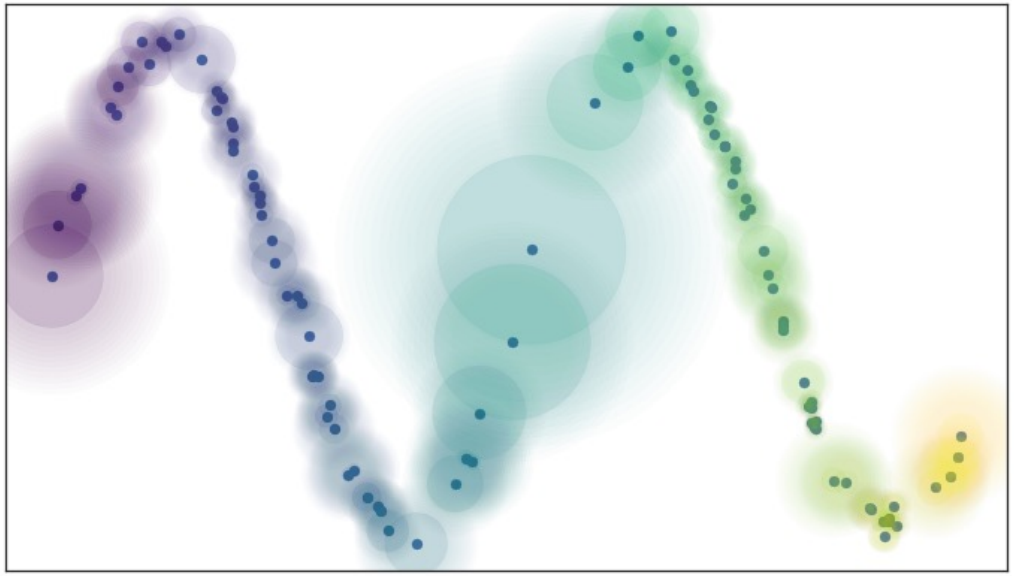
- That just means that our kernel function looks like this:

$$w((x_i, x_{i_j})) = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right)$$

$\rho_i = the\ distance\ between\ x_i\ and\ its\ nearest\ neighbor$

$\sigma_i = the\ diameter\ of\ x_i's\ k - nearest\ neighborhood$

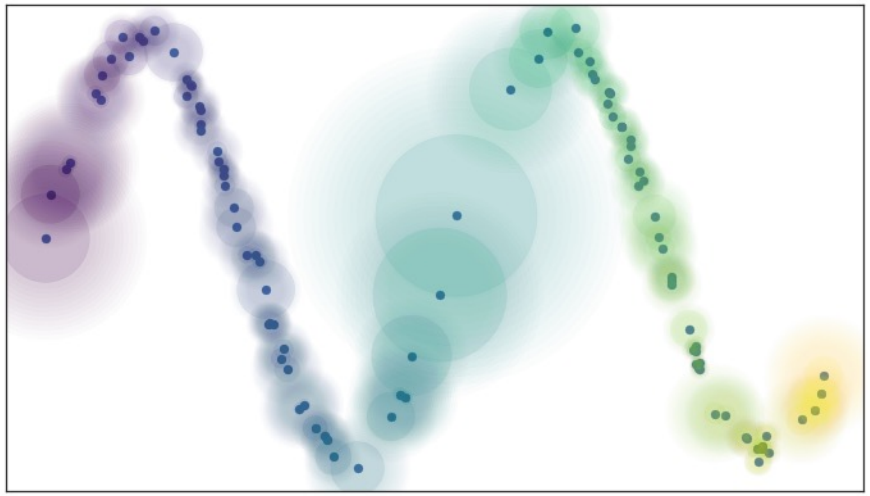


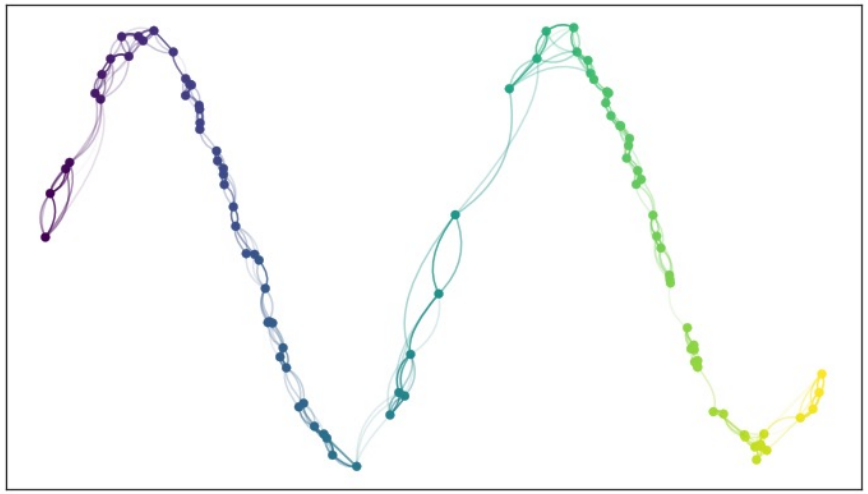


McInnes et al., *arXiv*, 2018
umap-learn.readthedocs.io

Construct the nerve

- We can use the same rules as previously described to construct the simplicial complex from the fuzzy cover.
- However, UMAP as implemented ignores simplices of dimension 2 or higher, leaving a network composed of only 0-simplices and 1-simplices.
- This is also known as a **nearest neighbors graph**!
- The edge weights on this graph are just the membership scores we calculated via the kernel.





McInnes et al., *arXiv*, 2018

Asymmetry due to incompatible local metrics

- Membership weights (affinities) in both directions between two points may not be the same.
  - i.e., the membership strength of point *p* in point *q*'s neighborhood does not equal the membership strength of point *q* in point *p*'s neighborhood, because each value has been calculated using a differently parameterized kernel.

- Solution: we symmetrize the weights using fuzzy set union

$$w(x_i, x_j) = w_i(x_i, x_j) + w_{i_j}(x_j, x_i) - w_i(x_i, x_j)w_{i_j}(x_j, x_i)$$

UMAP cost function: cross-entropy

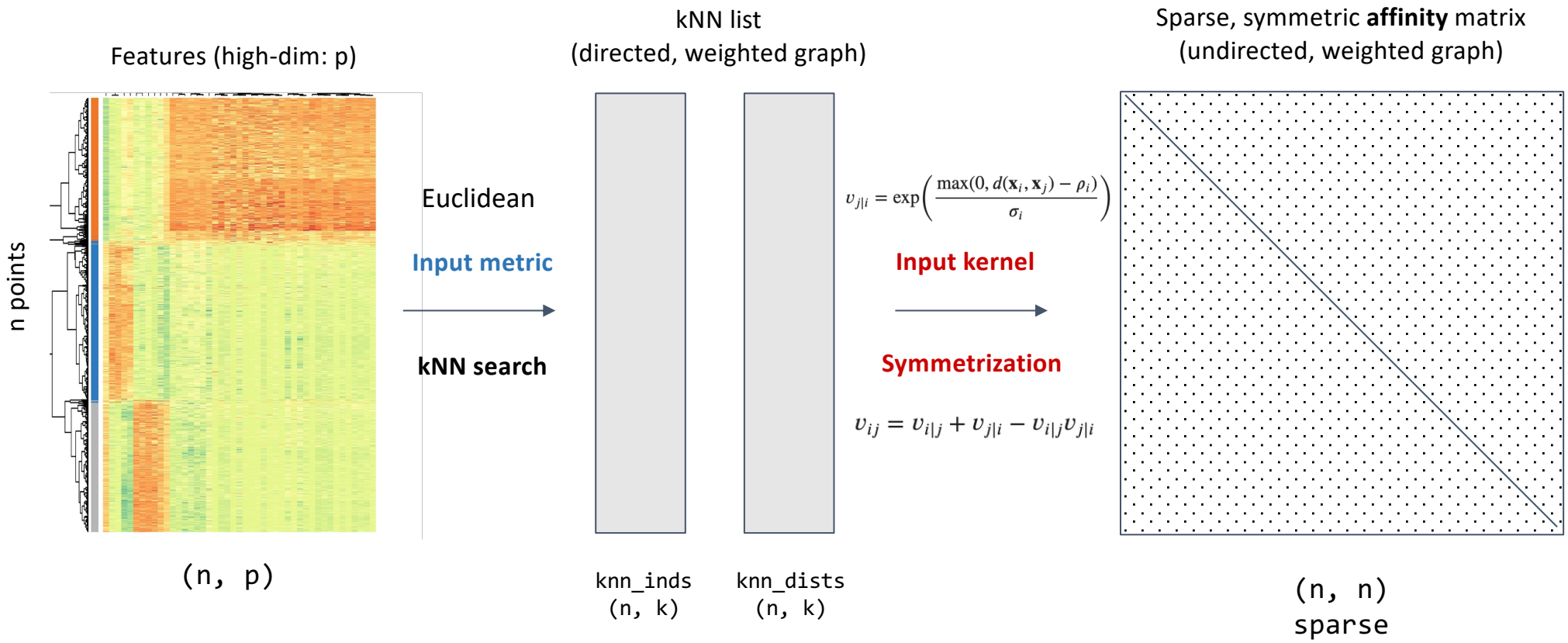
- Minimize the cross-entropy so that the low-dimensional data representation will have a similar fuzzy membership structure to the high-dimensional data

$$C(w, w') = \sum_{i \sim j} \underbrace{w(i,j) \log\left(\frac{w(i,j)}{w'(i,j)}\right)}_{\text{This term addresses the “clumps”.}} + \underbrace{(1 - w(i,j)) \log\left(\frac{1 - w(i,j)}{1 - w'(i,j)}\right)}_{\text{This term addresses the “gaps”.}}$$

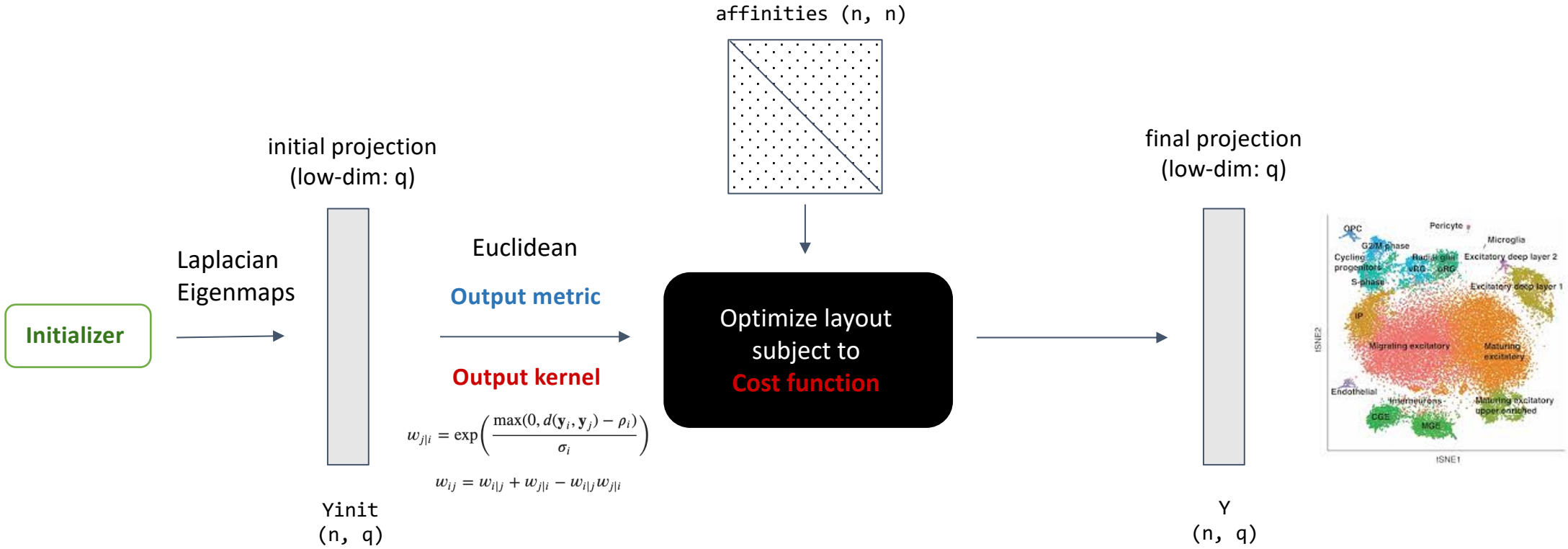
- Instead of standard gradient descent or Barnes-Hut approximation, UMAP uses another optimization algorithm called **Negative Sampling**
  - On each gradient descent iteration, only a small number m (default m=5) of randomly picked repulsive forces are applied to each point for each of the attractive forces that it feels. Other repulsive terms are ignored.

McInnes et al., *SciPy*, 2018
Hansen, *UPenn.edu,* 2018

UMAP Step 1: Calculate affinities



UMAP Step 2: Initialize and optimize the low-dim layout



Method: Gradient Descent by **Negative Sampling**

Cost function (UMAP):

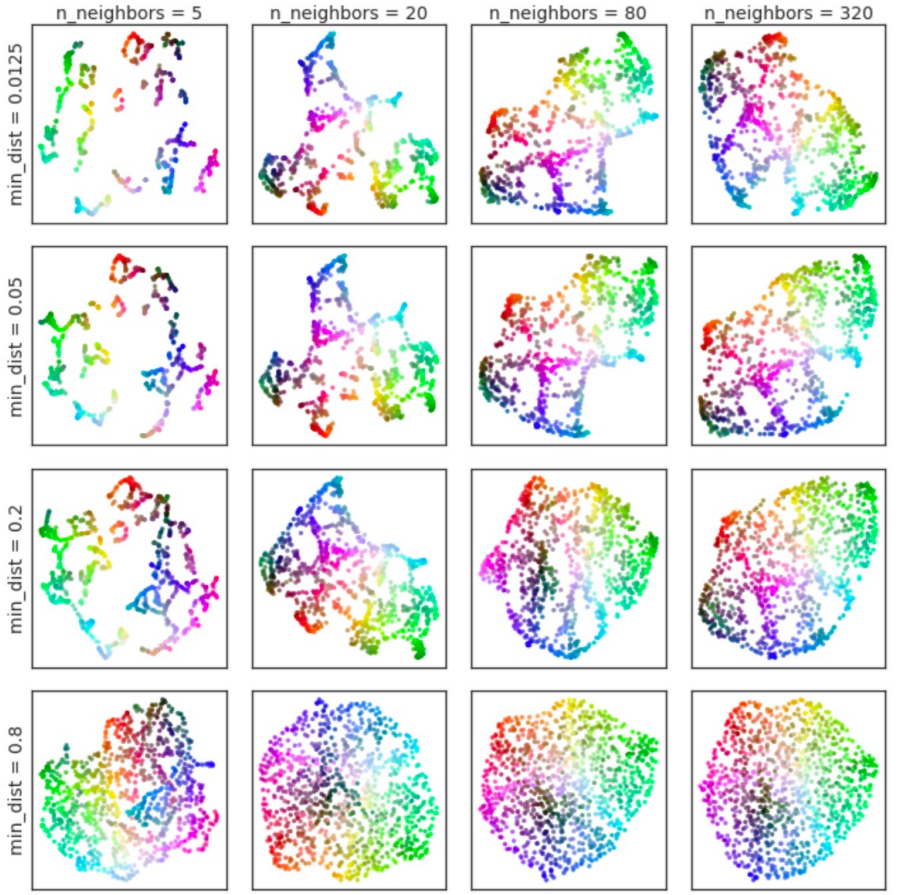
$$C = \sum_{i \sim j} \left[v_{ij} \log \frac{v_{ij}}{w_{ij}} + (1 - v_{ij}) \log \frac{1 - v_{ij}}{1 - w_{ij}} \right]$$

Gradient (UMAP):

$$\frac{\partial C}{\partial \mathbf{y}_i} \sim \sum_{j \neq i} \left[v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \frac{1}{d_{ij}^2 + \epsilon} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) \right]$$

Parameters of UMAP

- Number of nearest neighbors (k)
  - The number of points to be contained within each set of the open cover.
- The target embedding dimension (d)
  - i.e., the lower dimension
- min-dist
  - The desired separation between close points in the embedding space.
  - An aesthetic parameter that assists in visualization.
- n-epochs
  - The number of training epochs to use when optimizing the low dimensional representation.

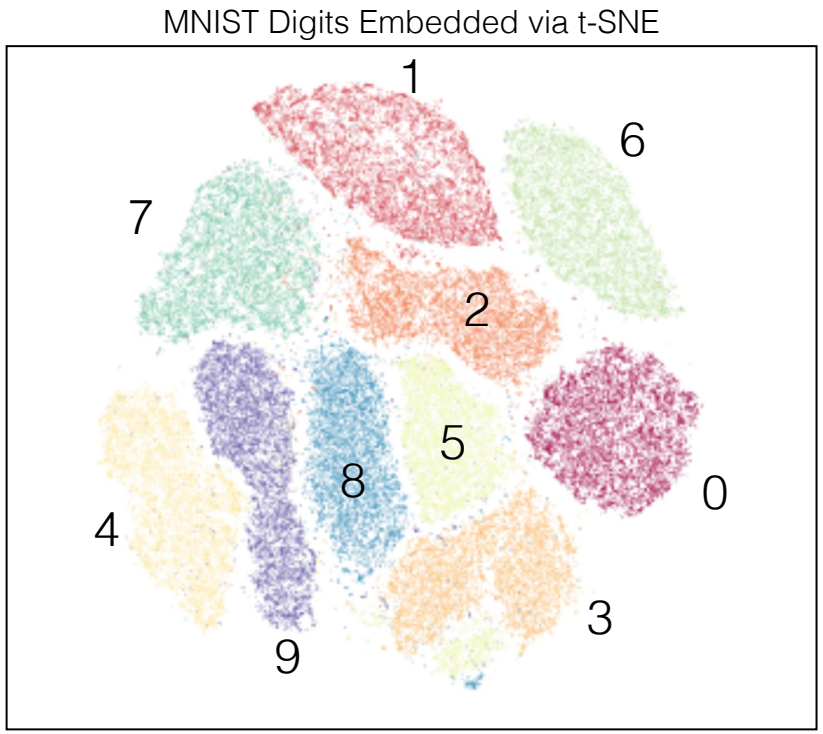


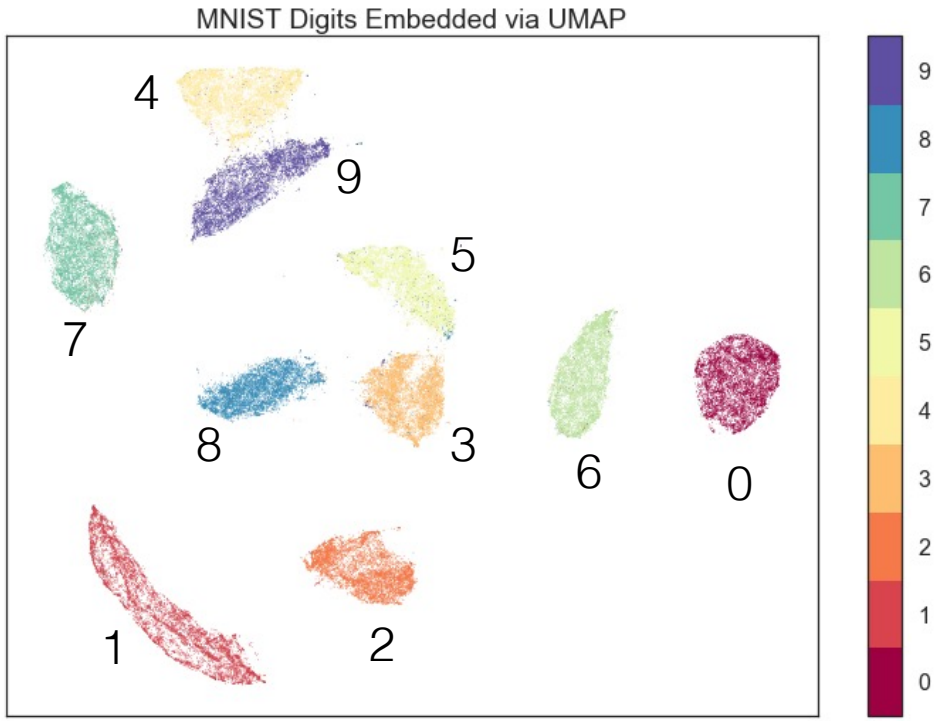
Understanding UMAP parameters:

https://pair-code.github.io/understanding-umap/

McInnes et al., *arXiv*, 2018
umap-learn.readthedocs.io

Comparing t-SNE and UMAP performance on MNIST

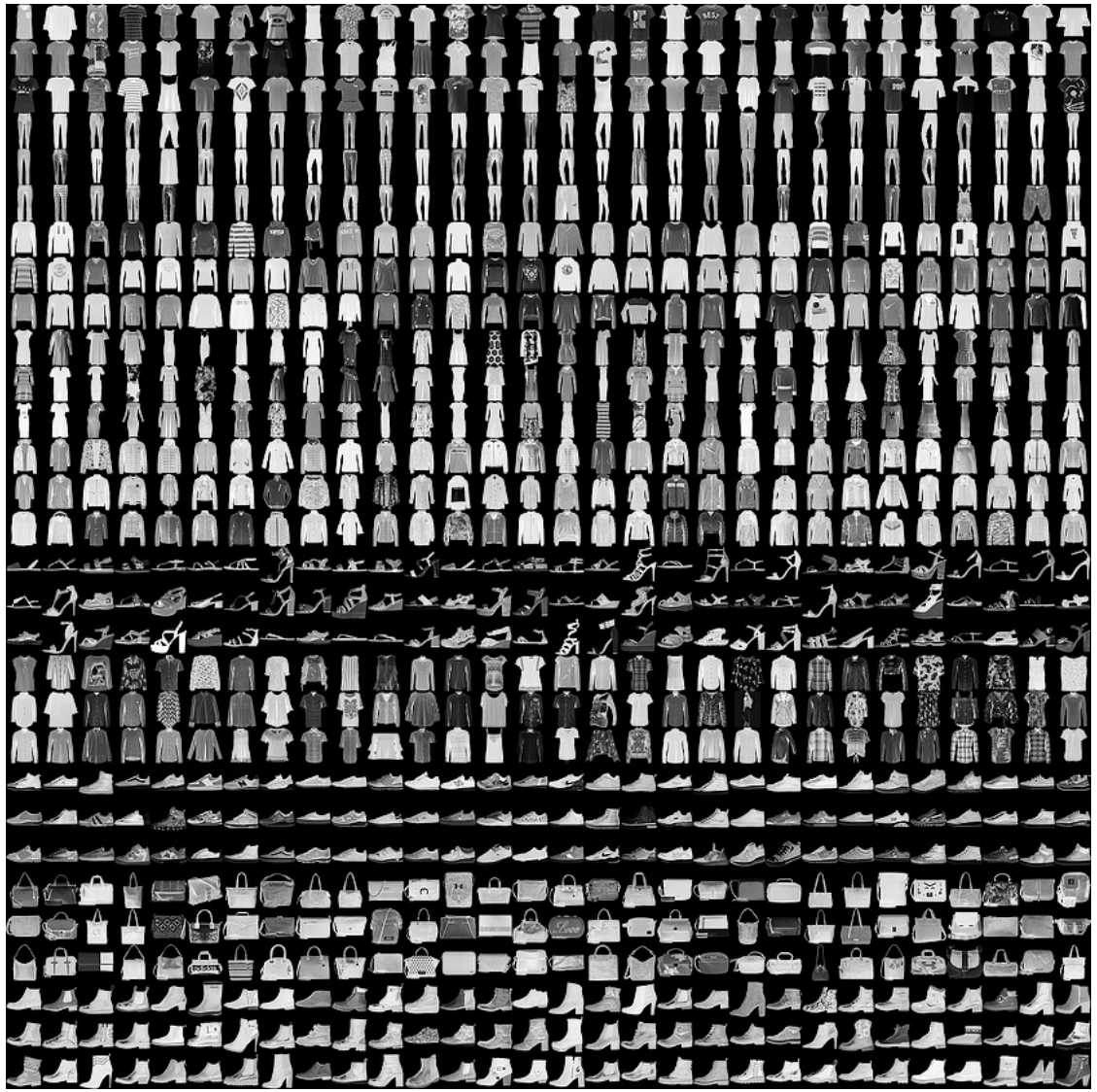




McInnes et al., *arXiv*, 2018
https://github.com/lmcinnes/umap

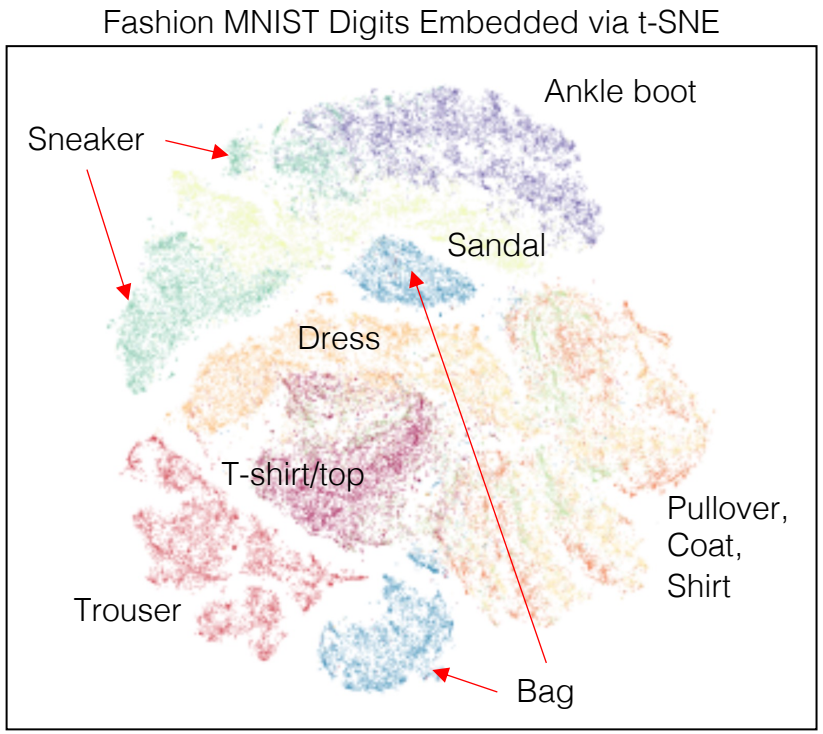
Fashion MNIST

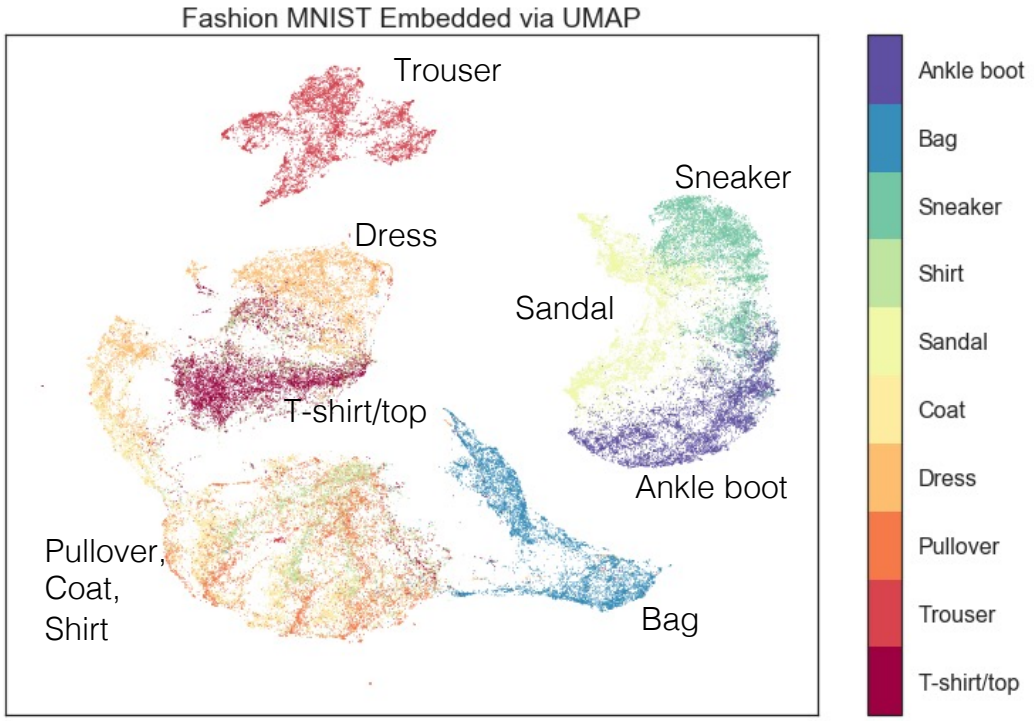
- 28x28 pixel images (784 dimensions)
- 10 classes of different clothing, shoe, and accessory items.
- Training set: 60,000 examples
- Test set: 10,000 examples
- Fashion MNIST is a direct “drop-in replacement” for MNIST.
- Presents a more challenging dataset for benchmarking machine learning algorithms.



Xiao et al., *arXiv*, 2017
https://github.com/zalandoresearch/fashion-mnist

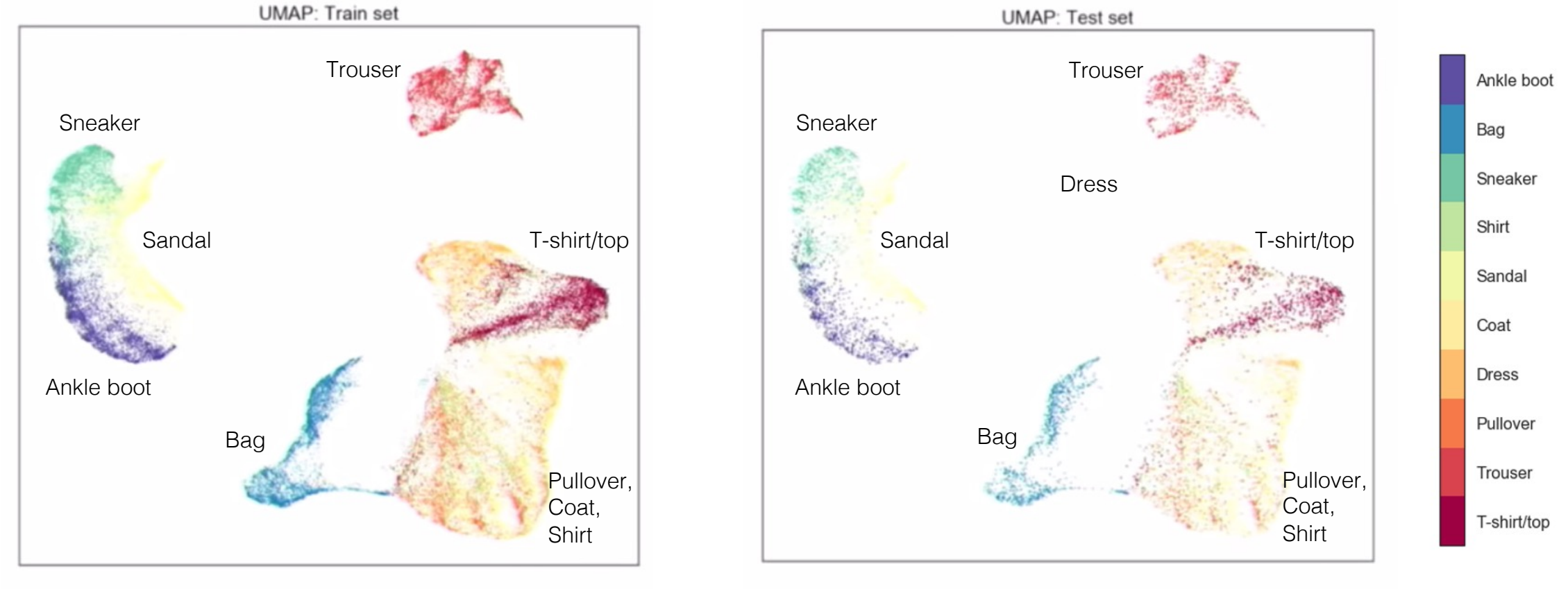
Comparing t-SNE and UMAP performance on Fashion MNIST





McInnes et al., *arXiv*, 2018
https://github.com/lmcinnes/umap

UMAP can project new, unseen data into existing embeddings



McInnes et al., *SciPy*, 2018

Applications of UMAP in bioinformatics

- Dataset:
  - 200 embryonic human DNase datasets from the ENCODE consortium.
  - 2.2 million representative DNase hypersensitive sites (rDHSs)
    - In other words, 2.2 million dimensions!

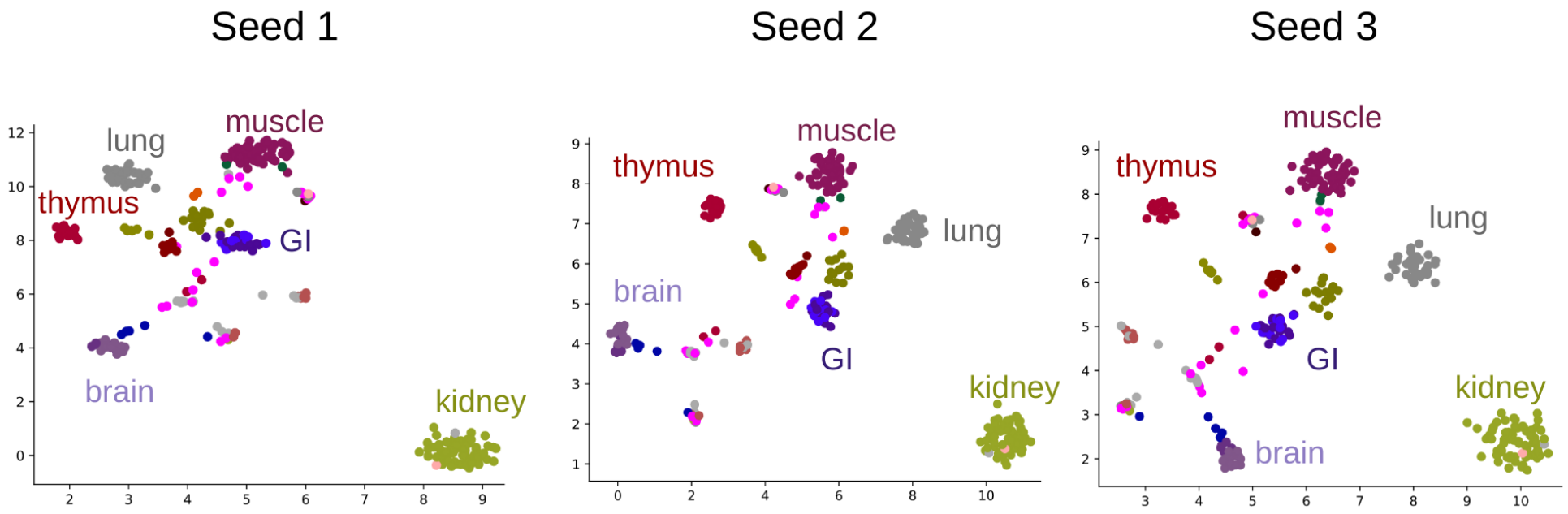
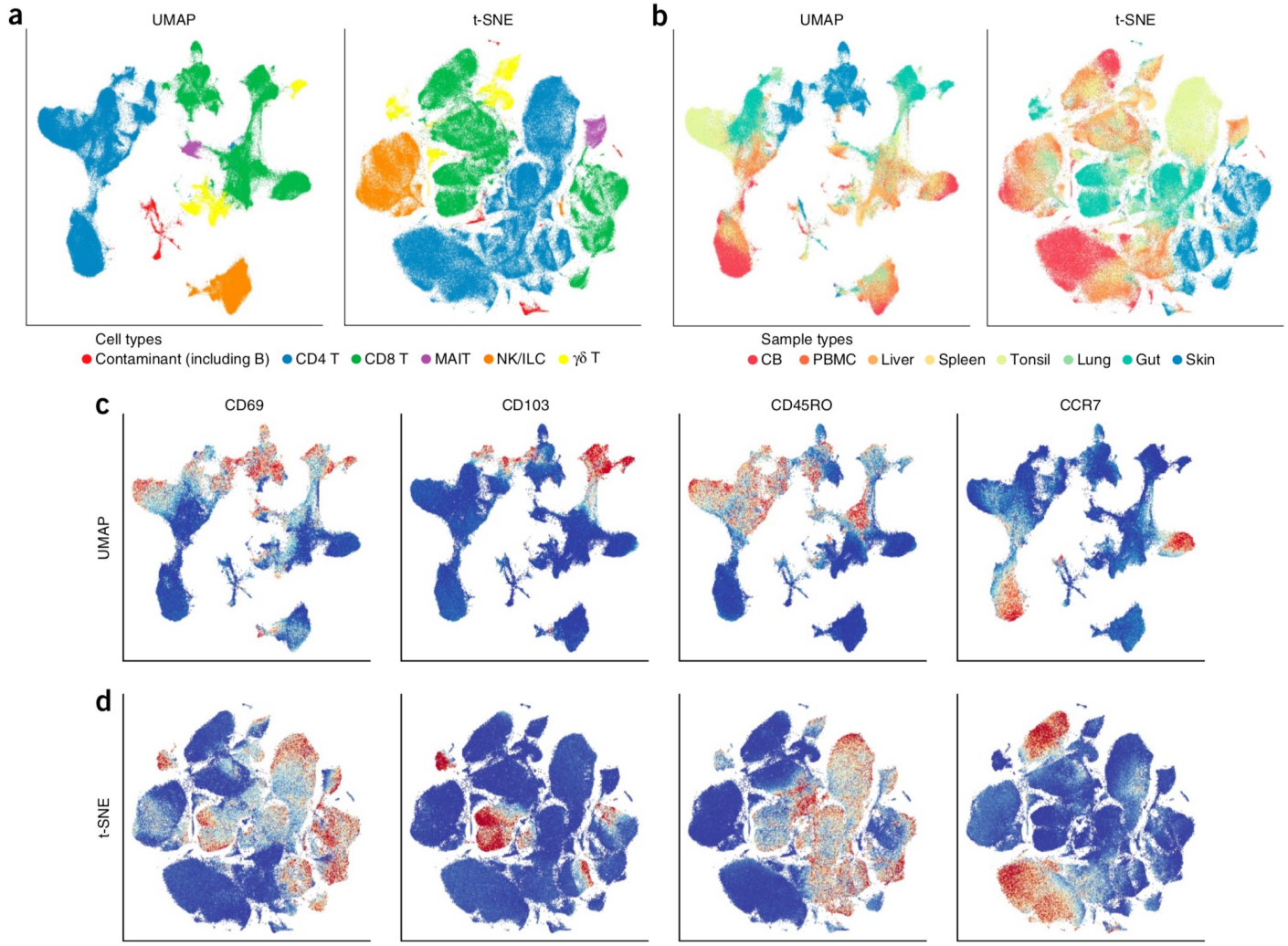


Figure credit: Henry Pratt

Applications of UMAP in bioinformatics

- Dataset (Wong et al., *Immunity*, 2016) :
  - Mass cytometry
  - 35 samples
  - 8 human tissues enriched for T and NK cells
  - > 300,000 events



Becht et al., *Nat Biotech*, 2019

Additional resources for UMAP

- Official UMAP webpage:
  - https://github.com/lmcinnes/umap
- UMAP documentation:
  - https://umap-learn.readthedocs.io/en/latest/index.html
- Original paper: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction (McInnes et al., *arXiv*, 2018)
  - https://arxiv.org/abs/1802.03426
- UMAP SciPy Talk, by the author:
  - https://www.youtube.com/watch?v=nq6iPZVUxZU
- Understanding UMAP:
  - https://pair-code.github.io/understanding-umap/

A Unifying Perspective on Neighbor Embeddings along the Attraction-Repulsion Spectrum

Jan Niklas Böhm
University of Tübingen
Tübingen, Germany
mail@jnboehm.com

Philipp Berens
University of Tübingen
Tübingen, Germany
philipp.berens@
uni-tuebingen.de

Dmitry Kobak
University of Tübingen
Tübingen, Germany
dmitry.kobak@
uni-tuebingen.de

Abstract

Neighbor embeddings are a family of methods for visualizing complex high-dimensional datasets using kNN graphs. To find the low-dimensional embedding, these algorithms combine an attractive force between neighboring pairs of points with a repulsive force between all points. One of the most popular examples of such algorithms is t-SNE. Here we show that changing the balance between the attractive and the repulsive forces in t-SNE yields a spectrum of embeddings, which is characterized by a simple trade-off: stronger attraction can better represent continuous manifold structures, while stronger repulsion can better represent discrete cluster structures. We show that UMAP embeddings correspond to t-SNE with increased attraction; this happens because the negative sampling optimisation strategy employed by UMAP strongly lowers the effective repulsion. Likewise, ForceAtlas2, commonly used for visualizing developmental single-cell transcriptomic data, yields embeddings corresponding to t-SNE with the attraction increased even more. At the extreme of this spectrum lies Laplacian Eigenmaps, corresponding to zero repulsion. Our results demonstrate that many prominent neighbor embedding algorithms can be placed onto this attraction-repulsion spectrum, and highlight the inherent trade-offs between them.

https://www.jmlr.org/papers/volume23/21-0055/21-0055.pdf

Gradient functions for many non-linear embedding methods are similar

Attraction Repulsion

$$\frac{\partial \mathcal{L}_{\text{t-SNE}}}{\partial \mathbf{y}_i} \sim \sum_j v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \frac{n}{Z} \sum_j w_{ij}^2 (\mathbf{y}_i - \mathbf{y}_j).$$

$$\frac{\partial \mathcal{L}_{\text{UMAP}}}{\partial \mathbf{y}_i} \sim \sum_j v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \sum_j \frac{1}{d_{ij}^2 + \epsilon} w_{ij} (\mathbf{y}_i - \mathbf{y}_j),$$

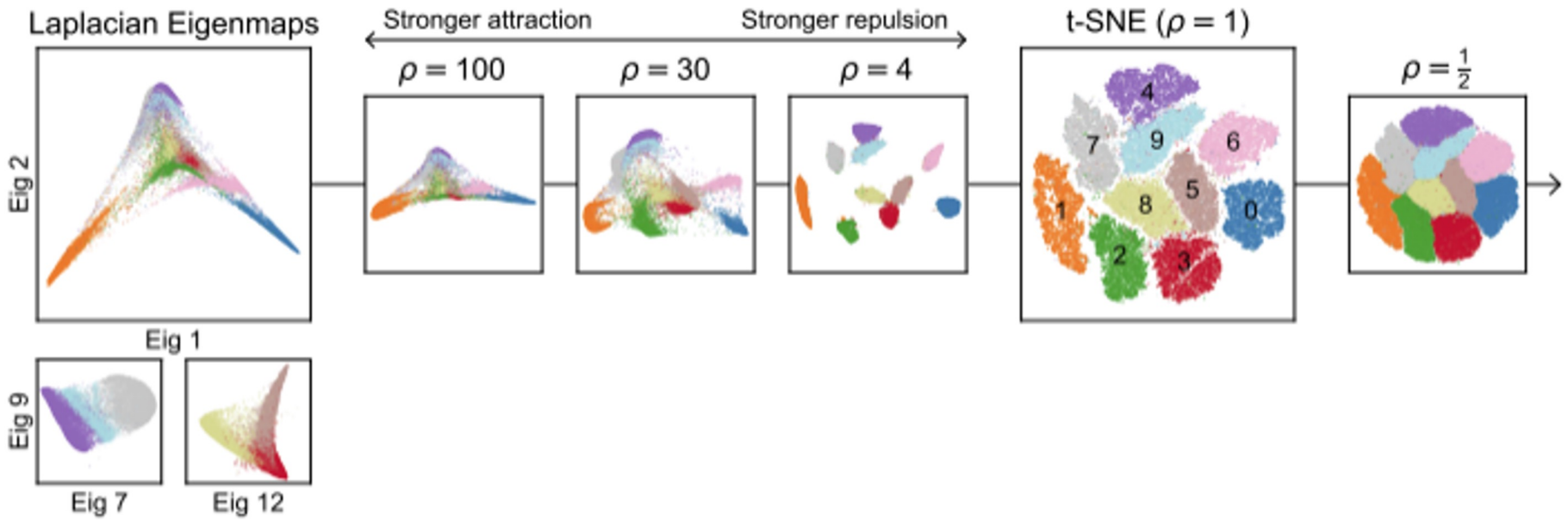
$$\frac{\partial \mathcal{L}_{\text{FA2}}}{\partial \mathbf{y}_i} = \sum_j v_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \sum_j \frac{(h_i + 1)(h_j + 1)}{d_{ij}^2} (\mathbf{y}_i - \mathbf{y}_j),$$

Introduce an exaggeration hyperparameter ρ

- Inspired by a optimization trick used in some implementations of t-SNE called “early exaggeration”
  - Temporarily increase the relative strength of attractive forces during the first few iterations to allow similar points to gather into clusters more effectively
- What if we <u>permanently</u> alter the relative weights of attractive vs repulsive forces in t-SNE throughout the layout optimization?

$$\frac{\partial \mathcal{L}_{\text{t-SNE}}(\rho)}{\partial \mathbf{y}_i} \sim \sum_j v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \frac{n}{\rho Z} \sum_j w_{ij}^2 (\mathbf{y}_i - \mathbf{y}_j)$$

- Low ρ → more repulsion
- High ρ → more attraction



- Stronger **attraction** can better represent **continuous** manifold structures
- Stronger **repulsion** can better represent **discrete** cluster structures
- Increasingly strong repulsion ‘brings out’ information from higher Laplacian eigenvectors into the two embedding dimensions

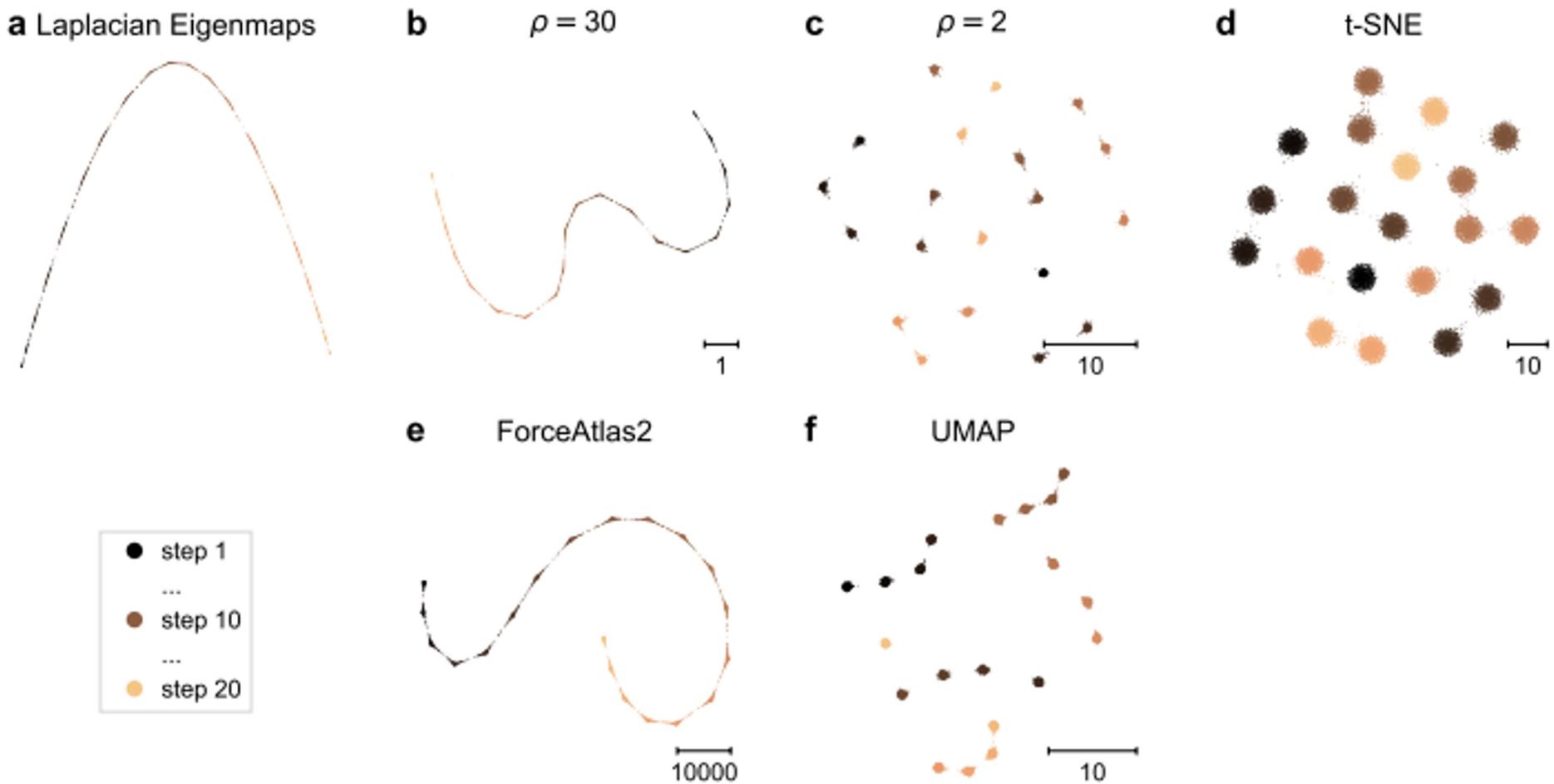


Figure 2: **Neighbor embeddings of simulated data emulating a developmental trajectory.** The points were sampled from 20 isotropic 50-dimensional Gaussians, equally spaced along one axis such that only few inter-cluster edges exist in the kNN graph.

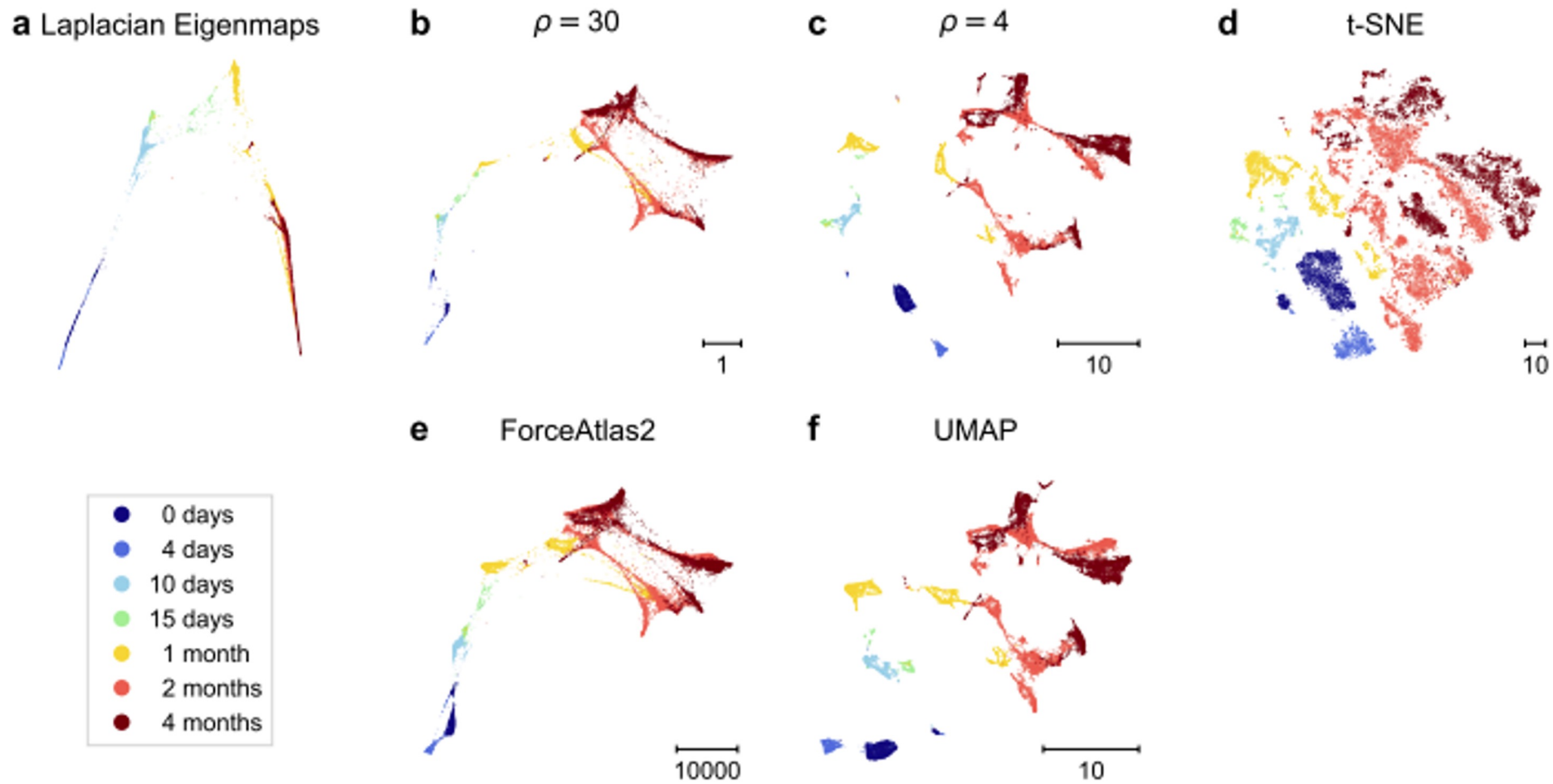


Figure 3: **Neighbor embeddings of the single-cell RNA-seq developmental data.** Cells were sampled from human brain organoids (cell line 409b2) at seven time points between 0 days and 4 months into the development [14]. Sample size $n = 20\,272$. Data were reduced with PCA to 50 dimensions. See Appendix for transcriptomic data preprocessing steps.

But why do UMAP and FA2 lie along the t-SNE attractive-repulsive spectrum?

- FA2 produced an embedding very similar to t-SNE with $\rho \approx 30$, while UMAP produced an embedding very similar to t-SNE with $\rho \approx 4$.
- Increased Attraction in FA2 due to non-decaying attractive forces
- Increased Attraction in UMAP due to… *Negative Sampling* algorithm!
  - "… this mismatch arises because the UMAP implementation is based on negative sampling and **does not in fact optimize its stated loss**. Instead, the negative sampling **decreases the repulsion strength**."

Optimizing UMAP with Barnes-Hut instead of Negative Sampling

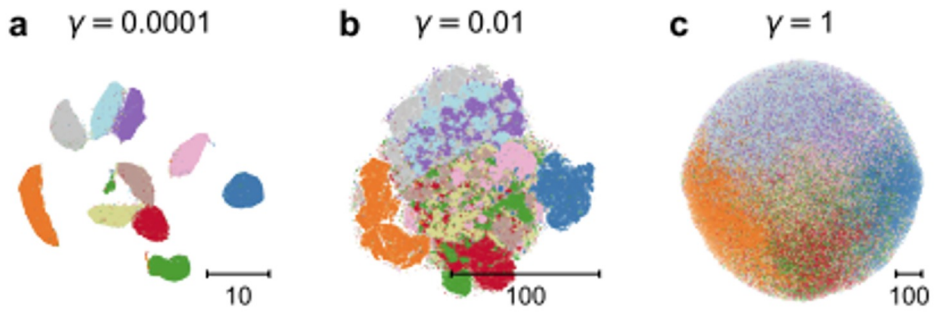
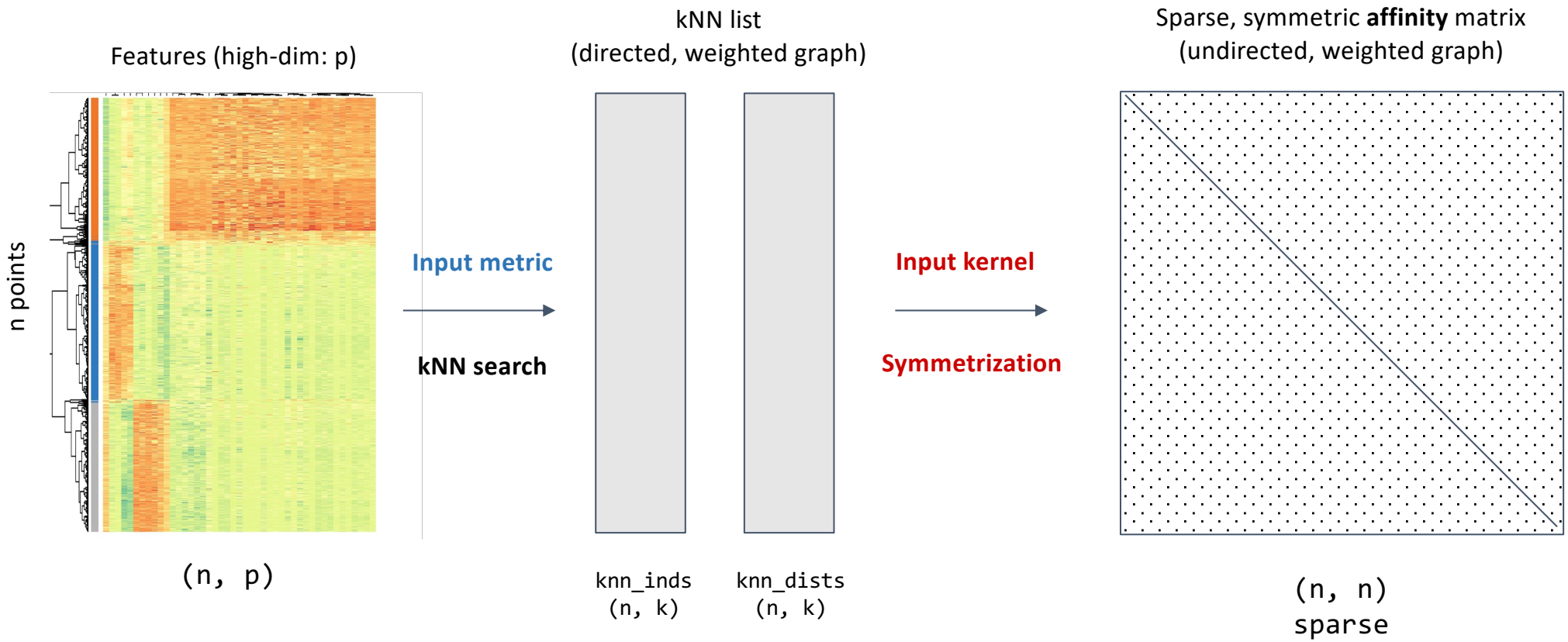


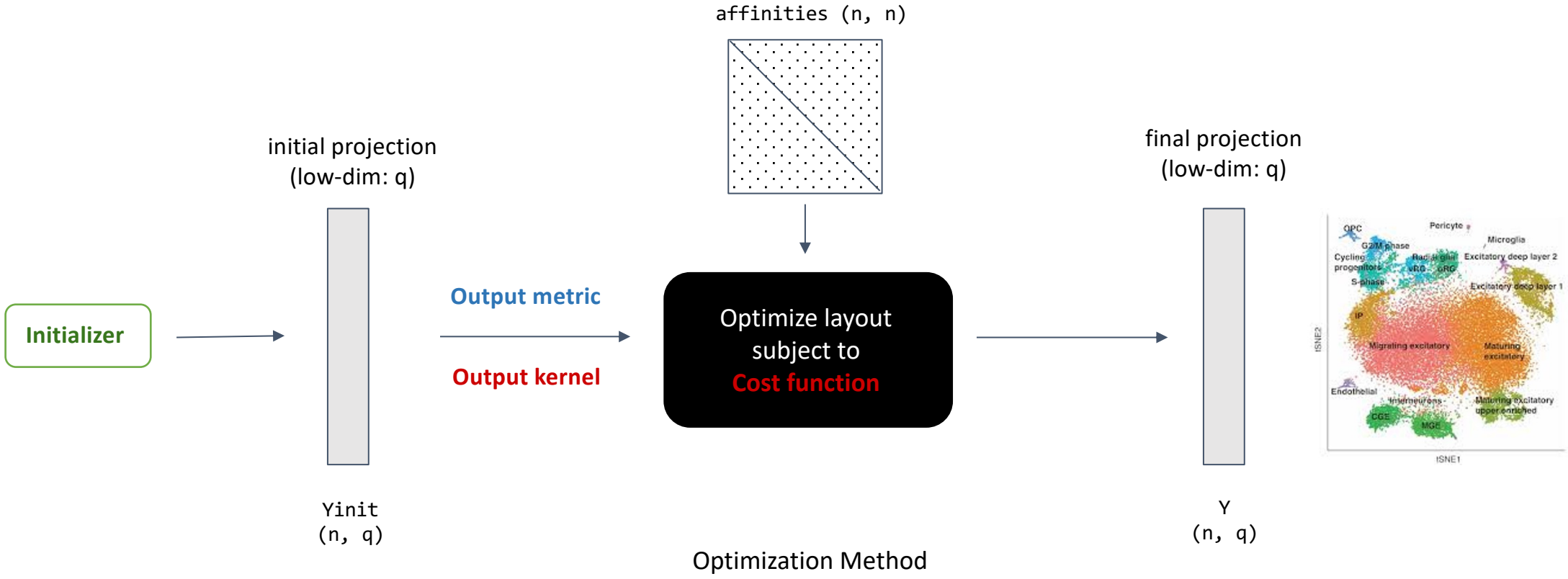
Figure 6: **Barnes-Hut UMAP without negative sampling. (a–c)** Embeddings with gamma values $\gamma \in \{0.0001, 0.01, 1\}$. **(d–e)** Embeddings with gamma values $\gamma \in \{0.01, 1\}$ initialized with the embedding with $\gamma = 0.0001$ [panel (a)], in an analogy to early exaggeration in t-SNE.

$$\frac{\partial \mathcal{L}_{\text{UMAP}}(\gamma)}{\partial \mathbf{y}_i} \sim \sum_j v_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \gamma \sum_j \frac{1}{d_{ij}^2 + \epsilon} w_{ij} (\mathbf{y}_i - \mathbf{y}_j).$$

Step 1: Calculate affinities



Step 2: Initialize and optimize the low-dim layout



	PCA	tSNE / UMAP
Linearity	Linear	Non-linear
Parametric	Yes	No
Similarity measure	Data (and feature) covariance/correlation	Data similarities derived from a non-linear kernel transformation of pairwise point distances
Structure inferred	Global: captures structure of data cloud within ambient space	Local: captures structure on a presumed low-dimensional manifold (neighborhood structure)
Flexibility of dimensions	Limited: only a cut-off on the number of PCs can be chosen based on explained variance	High: any number of dimensions can be chosen (usually 2D or 3D)*
Flexibility of layout	Low: PC projections are unique and fixed	High: many tuning parameters
Visualization	Challenging when explained var spread over many PCs	Easy for 2D and 3D
Interpretability of axes	Good: coordinates are linear combinations of original features; projection is unique, non-distortional	Poor: projection is arbitrarily oriented and embeddings are sensitive to hyperparameters and optimization details
Interpretability of distances	High if all PCs are used. Limited if the PC space is truncated.	Short-range distances are qualitatively interpretable. Long-range distances are not.

*Note: Barnes-Hut algorithm only works in 2D or 3D