

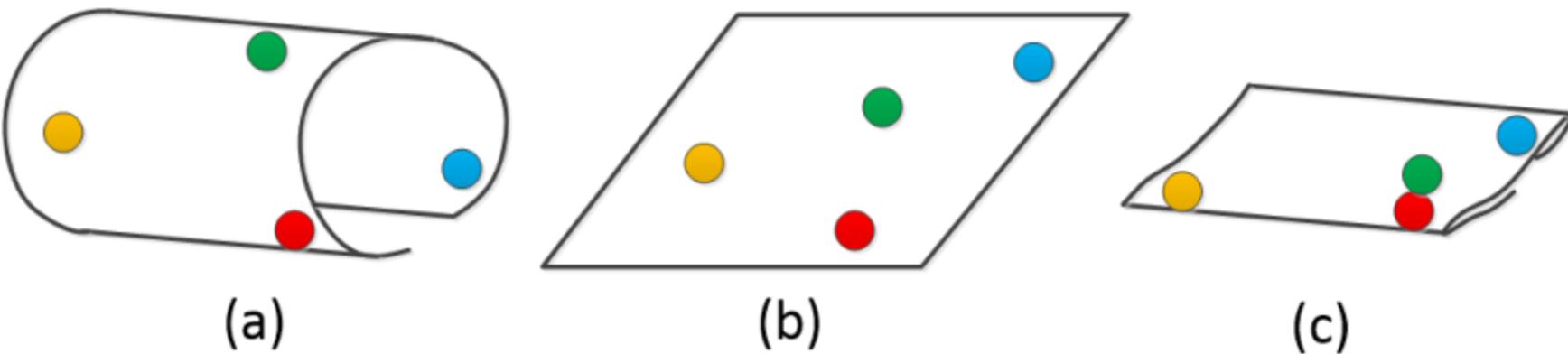


Dimensionality Reduction

- ▶ Locally Linear Embedding (LLE)
- ▶ Stochastic Neighbor Embedding (SNE, T-SNE)

Based on : Slides of Prof Mark Crowley, University of Waterloo
And Slides of Prof Dmitry Kobak , Tubingen University

Why Need Nonlinear Method?

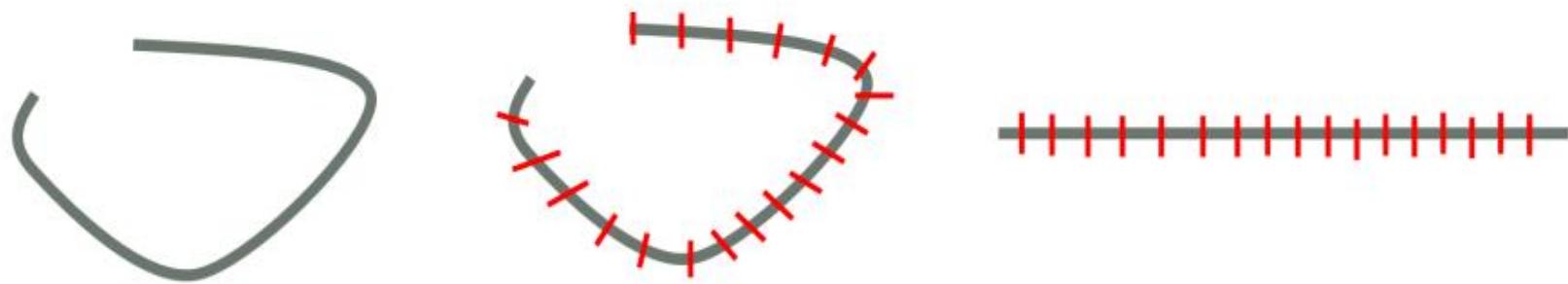


(a) A 2D nonlinear manifold where the data exist on in the 3D original space. As the manifold is nonlinear, the geodesic distances of points on the manifold are different from their Euclidean distances. (b) The correct unfolded manifold where the geodesic distances of points on the manifold have been preserved. (c) Applying the linear PCA, which takes Euclidean distances into account, on the nonlinear data where the found subspace has ruined the manifold so the far away red and green points have fallen next to each other.

The credit of this example is for Prof. Ali Ghodsi.

Goal of LLE: Piece-wise Local Embedding

LLE unfolds the nonlinear manifold in a piece-wise manner. Each piece is unfolded and the unfolded pieces are put together to have the entire unfolded manifold.



If two points are similar, they should be similar in the subspace. If they are dissimilar, do not care. So, it is a local fitting.

LLE was first proposed in [1, 2].

Locally Linear Embedding (LLE)

Assumption:

- ▶ data lies on a manifold: each sample and its neighbors lie on a approximately linear subspace

Approach:

1. Approximate data cloud by a set of linear patches
2. Glue these patches together on a low-dimensional subspace in such a way that the neighborhood relationships between patches are preserved.

Properties:

1. can obtain highly nonlinear embeddings
2. not prone to get stuck at local minima
3. sparse graphs lead to sparse problems, hence scalable

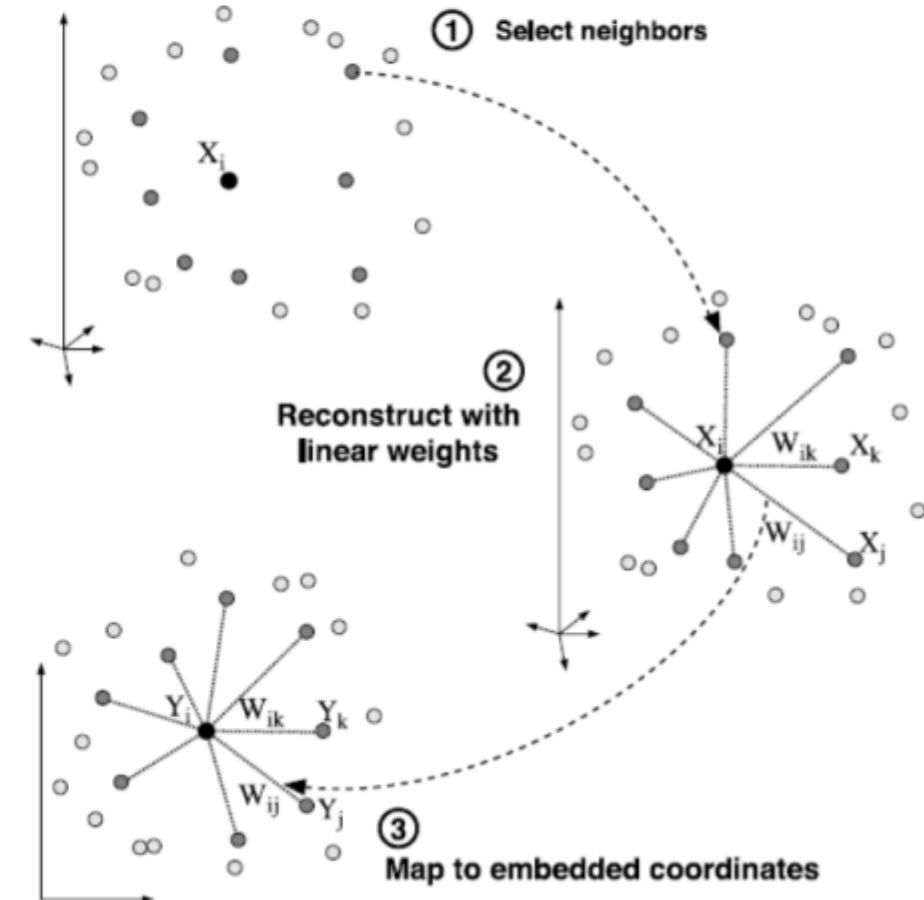
<https://cs.nyu.edu/~roweis/lle/algorithm.html>

- Roweis, Sam T., and Lawrence K. Saul. *Nonlinear dimensionality reduction by locally linear embedding*, Science 290.5500 (2000): 2323-2326. Google Scholar citations: 14,957(2020), 16,416 (2021)

Three Steps in LLE

Steps of LLE:

- ① Construct kNN graph
- ② Calculate the reconstruction of weights for reconstructing every point by its neighbors
- ③ Use the obtained weights to embed the points in the low dimensional subspace



Dataset Notations

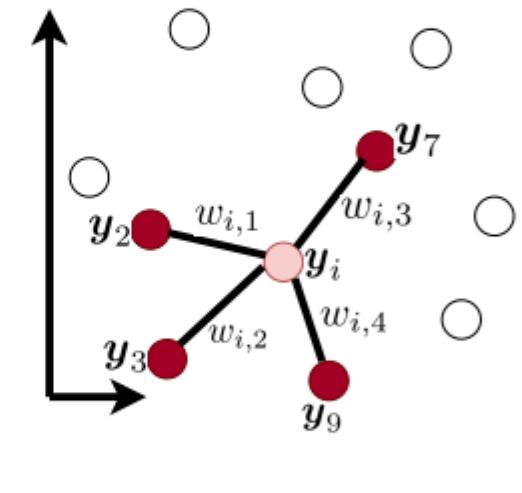
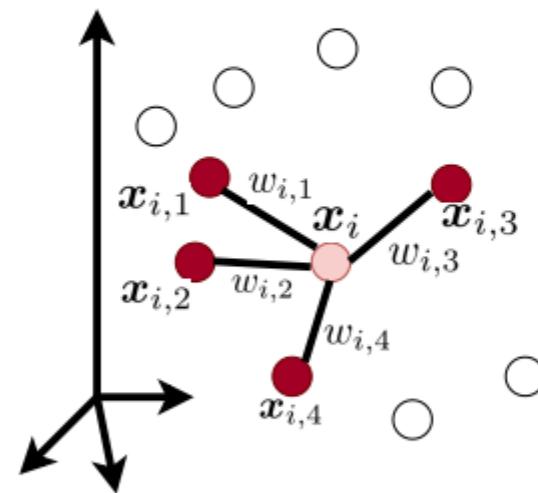
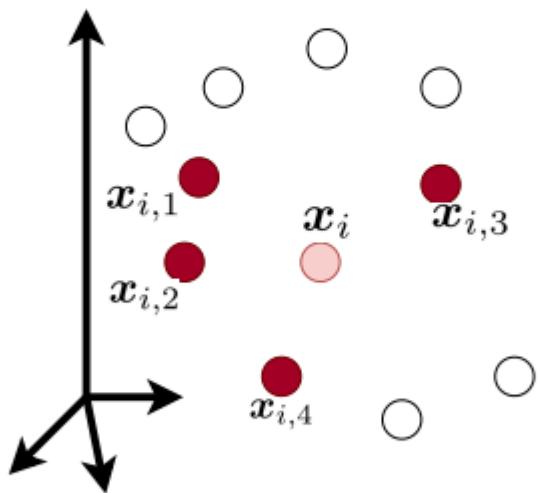
training dataset: $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n, \mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ (1)

j -th neighbor of \mathbf{x}_i : $\mathbf{x}_{ij} \in \mathbb{R}^d, \quad \mathbf{X}_i := [\mathbf{x}_{i1}, \dots, \mathbf{x}_{ik}] \in \mathbb{R}^{d \times k}$ (2)

test dataset: $\{\mathbf{x}_i^{(t)} \in \mathbb{R}^d\}_{i=1}^{n_t}, \mathbf{X}^{(t)} := [\mathbf{x}_1^{(t)}, \dots, \mathbf{x}_{n_t}^{(t)}] \in \mathbb{R}^{d \times n_t}$ (3)

training neighbors of $\mathbf{x}_i^{(t)}$: $\mathbf{X}_i^{(t)} := [\mathbf{x}_{i1}^{(t)}, \dots, \mathbf{x}_{ik}^{(t)}] \in \mathbb{R}^{d \times k}$ (4)

Three Steps in LLE (in our notation)



k-Nearest Neighbors

A *kNN* graph is formed using pairwise Euclidean distance between the data points. Therefore, every data point has *k* neighbors.

j-th neighbor of \mathbf{x}_i : $\mathbf{x}_{ij} \in \mathbb{R}^d$ (5)

$\mathbf{X}_i := [\mathbf{x}_{i1}, \dots, \mathbf{x}_{ik}] \in \mathbb{R}^{d \times k}$ (6)

Step 1: nearest neighbour search

For each node $X_i, i = 1, \dots, n$

- ▶ compute the distance from X_i to every other point X_j
- ▶ find the K smallest distances
- ▶ assign the corresponding points to be neighbours of X_i

More efficiently computationally:

- ▶ use standard algorithms for k -nearest neighbor (**k-nn**) search
- ▶ or even settle for **approximation algorithms**, that compute k -nearest neighbors
- ▶ *Randomized approximate nearest neighbors algorithm*, Peter Wilcox Jones, Andrei Osipov, and Vladimir Rokhlin, PNAS 2011
<https://www.pnas.org/content/108/38/15679.full>
- ▶ **nearest neighbor search** is an established area in theoretical computer science.

Linear Reconstruction by Neighbors

$$\mathbb{R}^k \ni \tilde{\mathbf{w}}_i := [\tilde{w}_{i1}, \dots, \tilde{w}_{ik}]^\top, \quad \mathbb{R}^{n \times k} \ni \tilde{\mathbf{W}} := [\tilde{\mathbf{w}}_1, \dots, \tilde{\mathbf{w}}_n]^\top \quad (7)$$

$$\begin{aligned} & \underset{\tilde{\mathbf{w}}}{\text{minimize}} \quad \varepsilon(\tilde{\mathbf{W}}) := \sum_{i=1}^n \left\| \mathbf{x}_i - \sum_{j=1}^k \tilde{w}_{ij} \mathbf{x}_{ij} \right\|_2^2, \\ & \text{subject to} \quad \boxed{\sum_{j=1}^k \tilde{w}_{ij} = 1}, \quad \forall i \in \{1, \dots, n\}, \end{aligned} \quad (8)$$

$$\varepsilon(\tilde{\mathbf{W}}) = \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{X}_i \tilde{\mathbf{w}}_i\|_2^2 \quad (9)$$

Linear Reconstruction by Neighbors

$$\begin{aligned} \|\mathbf{x}_i - \mathbf{X}_i \tilde{\mathbf{w}}_i\|_2^2 &= \|\mathbf{x}_i \mathbf{1}^\top \tilde{\mathbf{w}}_i - \mathbf{X}_i \tilde{\mathbf{w}}_i\|_2^2 = \|(\mathbf{x}_i \mathbf{1}^\top - \mathbf{X}_i) \tilde{\mathbf{w}}_i\|_2^2 \\ &= \tilde{\mathbf{w}}_i^\top (\mathbf{x}_i \mathbf{1}^\top - \mathbf{X}_i)^\top (\mathbf{x}_i \mathbf{1}^\top - \mathbf{X}_i) \tilde{\mathbf{w}}_i = \tilde{\mathbf{w}}_i^\top \mathbf{G}_i \tilde{\mathbf{w}}_i \end{aligned} \quad (10)$$

$$\mathbb{R}^{k \times k} \ni \mathbf{G}_i := (\mathbf{x}_i \mathbf{1}^\top - \mathbf{X}_i)^\top (\mathbf{x}_i \mathbf{1}^\top - \mathbf{X}_i) \quad (11)$$

$$\begin{aligned} &\underset{\{\tilde{\mathbf{w}}_i\}_{i=1}^n}{\text{minimize}} \quad \sum_{i=1}^n \tilde{\mathbf{w}}_i^\top \mathbf{G}_i \tilde{\mathbf{w}}_i, \\ &\text{subject to} \quad \mathbf{1}^\top \tilde{\mathbf{w}}_i = 1, \quad \forall i \in \{1, \dots, n\}. \end{aligned} \quad (12)$$

Linear Reconstruction by Neighbors

$$\mathcal{L} = \sum_{i=1}^n \tilde{\mathbf{w}}_i^\top \mathbf{G}_i \tilde{\mathbf{w}}_i - \sum_{i=1}^n \lambda_i (1^\top \tilde{\mathbf{w}}_i - 1) \quad (13)$$

$$\begin{aligned} \mathbb{R}^k \ni \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{w}}_i} &= 2\mathbf{G}_i \tilde{\mathbf{w}}_i - \lambda_i 1 \stackrel{\text{set}}{=} 0, \\ \implies \tilde{\mathbf{w}}_i &= \frac{1}{2} \mathbf{G}_i^{-1} \lambda_i 1 = \frac{\lambda_i}{2} \mathbf{G}_i^{-1} 1. \end{aligned} \quad (14)$$

$$\mathbb{R} \ni \frac{\partial \mathcal{L}}{\partial \lambda} = 1^\top \tilde{\mathbf{w}}_i - 1 \stackrel{\text{set}}{=} 0 \implies 1^\top \tilde{\mathbf{w}}_i = 1 \quad (15)$$

$$\therefore \frac{\lambda_i}{2} 1^\top \mathbf{G}_i^{-1} 1 = 1 \implies \lambda_i = \frac{2}{1^\top \mathbf{G}_i^{-1} 1} \quad (16)$$

$$\therefore \tilde{\mathbf{w}}_i = \frac{\lambda_i}{2} \mathbf{G}_i^{-1} 1 = \boxed{\frac{\mathbf{G}_i^{-1} 1}{1^\top \mathbf{G}_i^{-1} 1}} \quad (17)$$

Linear Embedding

$$\begin{aligned} & \underset{\mathbf{Y}}{\text{minimize}} && \sum_{i=1}^n \left\| \mathbf{y}_i - \sum_{j=1}^n w_{ij} \mathbf{y}_j \right\|_2^2, \\ & \text{subject to} && \frac{1}{n} \sum_{i=1}^n \mathbf{y}_i \mathbf{y}_i^\top = \mathbf{I}, \end{aligned} \tag{18}$$

$$\sum_{i=1}^n \mathbf{y}_i = 0,$$

$$w_{ij} := \begin{cases} \tilde{w}_{ij} & \text{if } \mathbf{x}_j \in k\text{NN}(\mathbf{x}_i) \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Linear Embedding

$$\sum_{i=1}^n \left\| \mathbf{y}_i - \sum_{j=1}^n w_{ij} \mathbf{y}_j \right\|_2^2 = \sum_{i=1}^n \| \mathbf{Y}^\top \mathbf{1}_i - \mathbf{Y}^\top \mathbf{w}_i \|_2^2 \quad (20)$$

$$\sum_{i=1}^n \| \mathbf{Y}^\top \mathbf{1}_i - \mathbf{Y}^\top \mathbf{w}_i \|_2^2 = \| \mathbf{Y}^\top \mathbf{I} - \mathbf{Y}^\top \mathbf{W}^\top \|_F^2$$

$$= \| \mathbf{Y}^\top (\mathbf{I} - \mathbf{W})^\top \|_F^2 = \text{tr}((\mathbf{I} - \mathbf{W}) \mathbf{Y} \mathbf{Y}^\top (\mathbf{I} - \mathbf{W})^\top) \quad (21)$$

Cyclic property of trace $\leftarrow \text{tr}(\mathbf{Y}^\top (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W}) \mathbf{Y}) = \text{tr}(\mathbf{Y}^\top \mathbf{M} \mathbf{Y})$

$$\mathbb{R}^{n \times n} \ni \mathbf{M} := (\mathbf{I} - \mathbf{W})^\top (\mathbf{I} - \mathbf{W}). \quad (22)$$

The Frobenius norm of A denoted by $|A|_F$ is defined by

$$\|A\|_F = \left(\sum_{i=1}^n \sum_{j=1}^n a_{i,j}^2 \right)^{\frac{1}{2}} = (\text{trace}[A^T A])^{\frac{1}{2}},$$

where $\text{trace}[A^T A]$ is the sum of the diagonal elements of $A^T A$ and A^T is the transpose of A .

Linear Embedding

$$\begin{aligned} & \underset{\mathbf{Y}}{\text{minimize}} && \text{tr}(\mathbf{Y}^\top \mathbf{M} \mathbf{Y}), \\ & \text{subject to} && \frac{1}{n} \mathbf{Y}^\top \mathbf{Y} = \mathbf{I}, \\ & && \mathbf{Y}^\top \mathbf{1} = 0, \end{aligned} \tag{23}$$

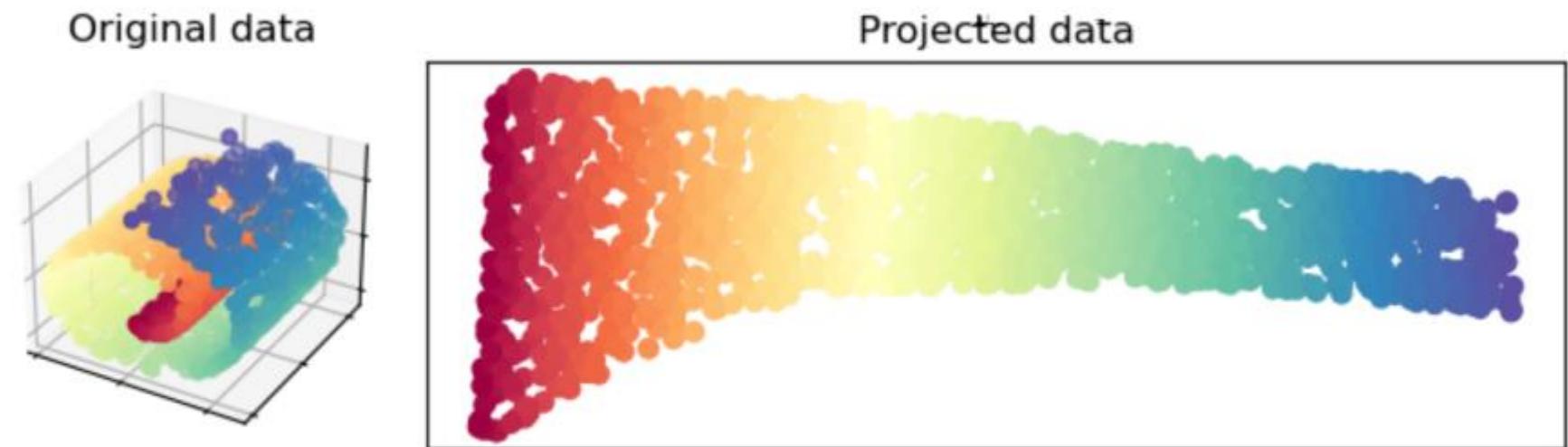
The fact that we have the eigenvector $\mathbf{1}$ with zero eigenvalue implicitly ensures that $\sum_{i=1}^n \mathbf{y}_i = \mathbf{Y}^\top \mathbf{1} = 0$ which was the second constraint. (See [3] for proof.)

$$\mathcal{L} = \text{tr}(\mathbf{Y}^\top \mathbf{M} \mathbf{Y}) - \text{tr}\left(\Lambda^\top \left(\frac{1}{n} \mathbf{Y}^\top \mathbf{Y} - \mathbf{I}\right)\right) \tag{24}$$

$$\mathbb{R}^{n \times p} \ni \frac{\partial \mathcal{L}}{\partial \mathbf{Y}} = 2\mathbf{M} \mathbf{Y} - \frac{2}{n} \mathbf{Y} \Lambda \stackrel{\text{set}}{=} 0$$

$$\Rightarrow \boxed{\mathbf{M} \mathbf{Y} = \mathbf{Y} \left(\frac{1}{n} \Lambda\right)} \tag{25}$$

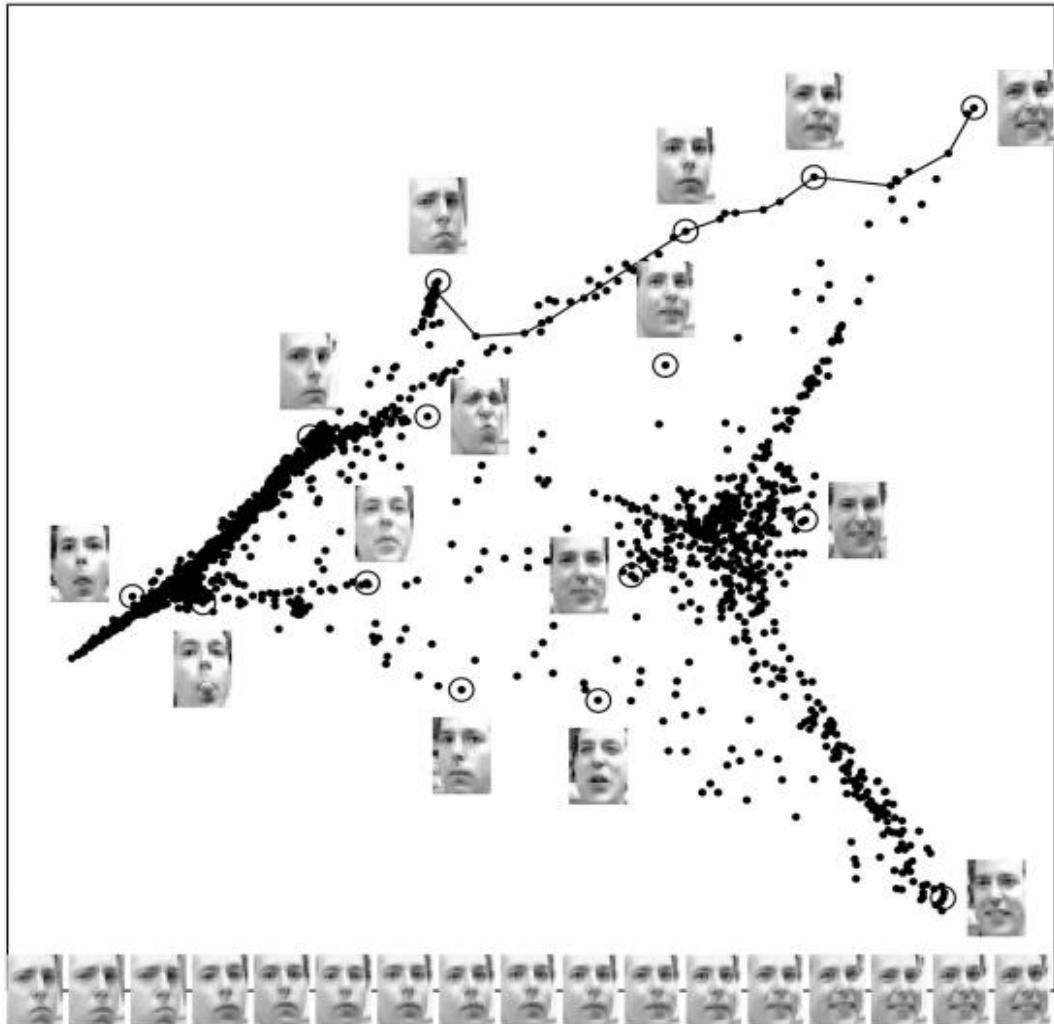
Unfolding Swiss roll by LLE



Note the covariance and mean of embedding.

Credit of image is for https://scikit-learn.org/stable/auto_examples/manifold/plot_swissroll.html

LLE Embedding of Frey Face Dataset



Note the covariance and mean of embedding.

Credit of image is for [1].

- Tutorial paper: “Locally Linear Embedding and its Variants: Tutorial and Survey” [3]
- Tutorial YouTube videos by Prof. Ali Ghodsi at University of Waterloo: [\[Click here\]](#) and [\[Click here\]](#)

- [1] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [2] L. K. Saul and S. T. Roweis, “Think globally, fit locally: unsupervised learning of low dimensional manifolds,” *Journal of machine learning research*, vol. 4, no. Jun, pp. 119–155, 2003.
- [3] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, “Locally linear embedding and its variants: Tutorial and survey,” *arXiv preprint arXiv:2011.10925*, 2020.

Stochastic Neighbor Embedding (**SNE**) t-distributed Stochastic Neighbor Embedding (**t-SNE**)

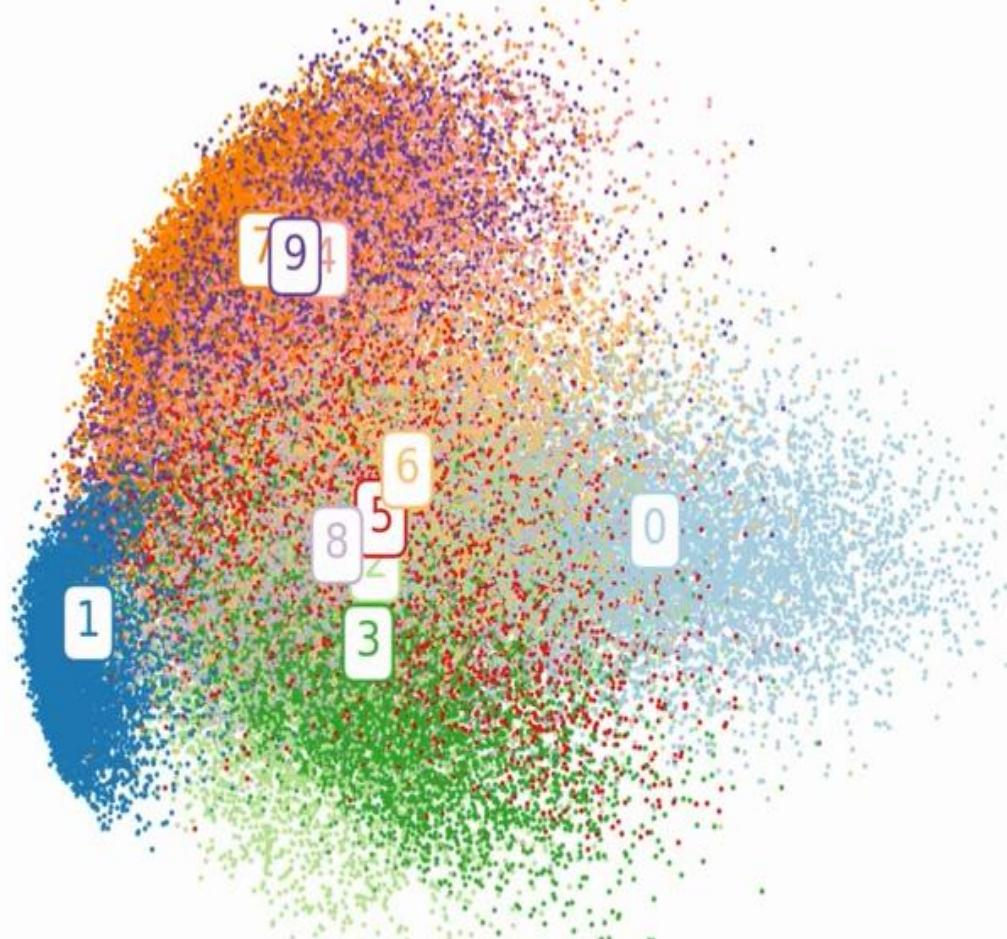
Dataset Notations

training dataset: $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n, \mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ (1)

embedding of training data: $\{\mathbf{y}_i \in \mathbb{R}^h\}_{i=1}^n, \mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n] \in \mathbb{R}^{h \times n}$ (2)

$h \leq d$, usually: $h \ll d$ (3)

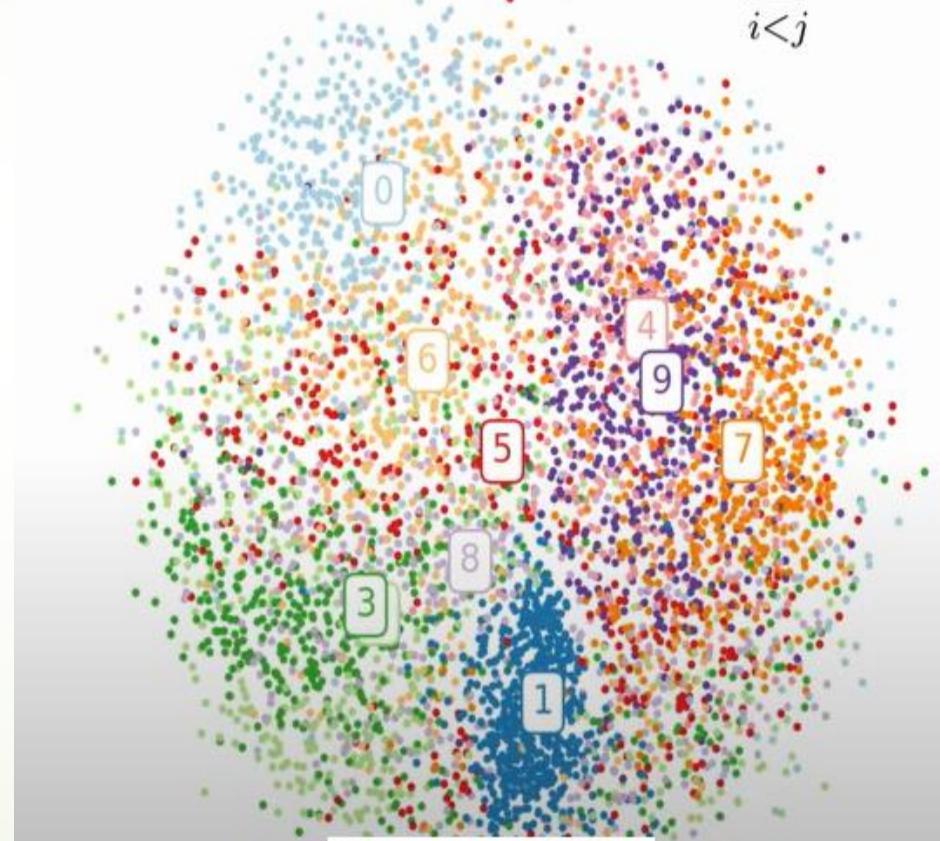
SNE and t-SNE are usually used for data visualization so we usually have $h = 2$ or $h = 3$.



4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

Multidimensional scaling: arrange points in 2D to approximate high-dimensional pairwise distances
(1950s–1960s; Kruskal, Torgerson, etc.).

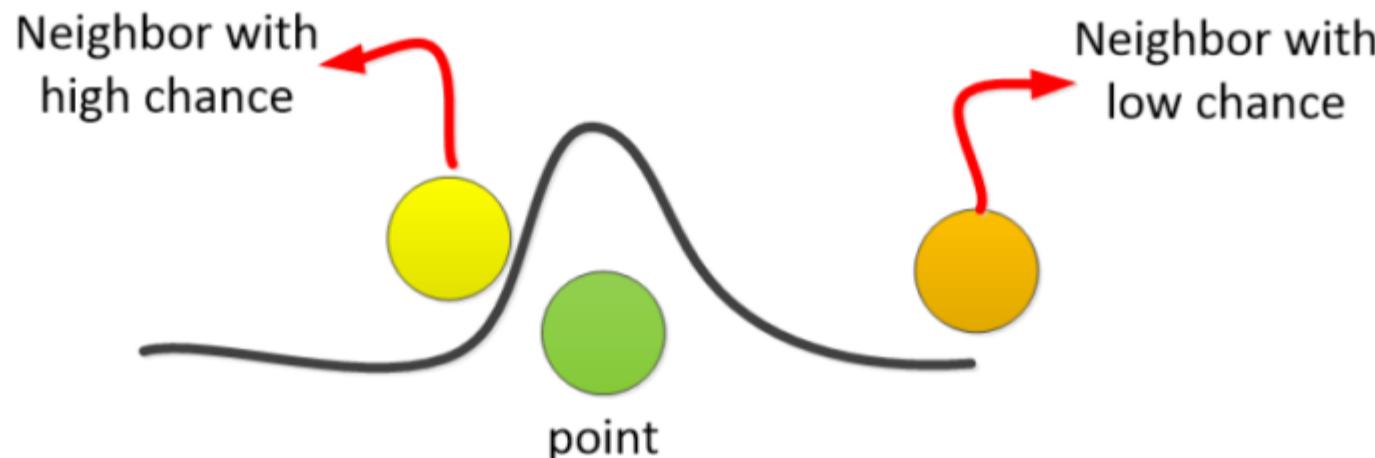
$$\mathcal{L} = \sum_{i < j} (d_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2$$



Here $n = 5,000$.

Probabilistic Approach of Embedding

- Rather than saying that this point is neighbor of that point but the other point is not a neighbor, we can have a **probabilistic** approach.
- We say, all points are neighbors of a point with some probability. A very **similar/dissimilar** point to some other point is its neighbor with **high/low probability**.
- SNE uses **Gaussian** distribution for probability of neighborhood.



Stochastic Neighbor Embedding

$$f(d) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d^2}{2\sigma^2}\right) \propto \exp\left(-\frac{d^2}{2\sigma^2}\right) \quad (4)$$

$$\mathbb{R} \ni p_{ij} := \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)} \quad (5)$$

$$\mathbb{R} \ni d_{ij}^2 := \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_i^2} \quad (6)$$

The σ_i^2 is the variance which we consider for the Gaussian distribution used for the \mathbf{x}_i . It can be set to a fixed number or by a binary search to make the entropy of distribution some specific value [1].

$$\mathbb{R} \ni q_{ij} := \frac{\exp(-z_{ij}^2)}{\sum_{k \neq i} \exp(-z_{ik}^2)} \quad (7)$$

$$\mathbb{R} \ni z_{ij}^2 := \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \quad (8)$$

Stochastic Neighbor Embedding

$$\mathbb{R} \ni c_1 := \sum_{i=1}^n \text{KL}(P_i || Q_i) = \sum_{i=1}^n \sum_{j=1, j \neq i}^n p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \quad (9)$$

$$\mathbb{R}^h \ni \frac{\partial c_1}{\partial \mathbf{y}_i} = 2 \sum_{j=1}^n (p_{ij} - q_{ij} + p_{ji} - q_{ji})(\mathbf{y}_i - \mathbf{y}_j) \quad (10)$$

For derivation of gradient, see [2].

Optimization using gradient descent with momentum:

$$\Delta \mathbf{y}_i^{(t)} := -\eta \frac{\partial c_1}{\partial \mathbf{y}_i} + \alpha(t) \Delta \mathbf{y}_i^{(t-1)} \quad (11)$$

$$\mathbf{y}_i^{(t)} := \mathbf{y}_i^{(t-1)} + \Delta \mathbf{y}_i^{(t)} \quad (12)$$

The momentum parameter:

$$\alpha(t) := \begin{cases} 0.5 & t < 250, \\ 0.8 & t \geq 250. \end{cases} \quad (13)$$

In SNE, we add jitter to the solution of initial iterations for better convergence of embedding.

Symmetric Stochastic Neighbor Embedding

$$\mathbb{R} \ni p_{ij} := \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ki}^2)} \quad \text{vs.} \quad \mathbb{R} \ni p_{ij} := \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)} \quad (14)$$

$$\mathbb{R} \ni d_{ij}^2 := \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_i^2} \quad (15)$$

The first p_{ij} has a problem with outliers. If the point \mathbf{x}_i is an outlier, its p_{ij} will be extremely small because the denominator is fixed for every point and numerator will be small for the outlier. However, If we use the second p_{ij} , the denominator for all the points is not the same and therefore, the effect of denominator waives out the effect of numerator. So we use instead:

$$\mathbb{R} \ni p_{j|i} := \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)} \quad (16)$$

$$\mathbb{R} \ni p_{ij} := \frac{p_{i|j} + p_{j|i}}{2n} \quad (17)$$

Symmetric Stochastic Neighbor Embedding

$$\mathbb{R} \ni p_{j|i} := \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)} \quad (18)$$

$$\mathbb{R} \ni p_{ij} := \frac{p_{i|j} + p_{j|i}}{2n} \quad (19)$$

But that problem does not exist in the embedding space because even for an outlier, the embedded points are initialized close together and not far.

$$\boxed{\mathbb{R} \ni q_{ij} := \frac{\exp(-z_{ij}^2)}{\sum_{k \neq I} \exp(-z_{kl}^2)}, \quad \text{vs.} \quad \mathbb{R} \ni q_{ij} := \frac{\exp(-z_{ij}^2)}{\sum_{k \neq i} \exp(-z_{ik}^2)}} \quad (20)$$

$$\mathbb{R} \ni z_{ij}^2 := \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \quad (21)$$

Symmetric Stochastic Neighbor Embedding

$$\mathbb{R} \ni c_2 := \sum_{i=1}^n \text{KL}(P_i || Q_i) = \sum_{i=1}^n \sum_{j=1, j \neq i}^n p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \quad (22)$$

$$\mathbb{R}^h \ni \frac{\partial c_2}{\partial \mathbf{y}_i} = 4 \sum_{j=1}^n (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j) \quad (23)$$

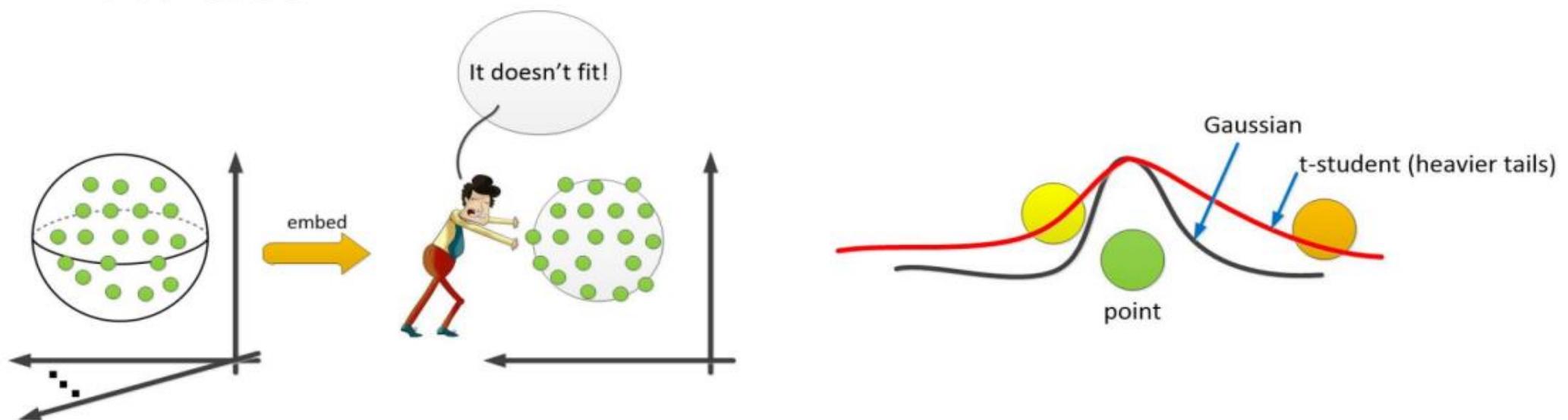
For derivation of gradient, see [2].

As before, optimization is done using gradient descent with momentum.

In symmetric SNE, we add jitter to the solution of initial iterations for better convergence of embedding.

The Crowding Problem

- If we want to fit one million people in a room, they don't fit. So, let's enlarge the room!
- **Crowding problem:** If we want to fit the large information of high dimensional data into low dimensional subspace, we should enlarge the distribution!
- **Student-t distribution** has heavier tails than the Gaussian distribution.



t-Stochastic Neighbor Embedding

$$\mathbb{R} \ni p_{j|i} := \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}, \quad \mathbb{R} \ni p_{ij} := \frac{p_{i|j} + p_{j|i}}{2n} \quad (24)$$

$$\mathbb{R} \ni d_{ij}^2 := \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_i^2} \quad (25)$$

$$q_{ij} = \frac{(1 + z_{ij}^2)^{-1}}{\sum_{k \neq i} (1 + z_{kl}^2)^{-1}} \quad (26)$$

$$\mathbb{R} \ni z_{ij}^2 := \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \quad (27)$$

In the embedding space, we use Student-t distribution (with one degree of freedom) which is the standard Cauchy distribution:

$$f(z) = \frac{1}{\pi(1 + z^2)} \quad (28)$$

t-Stochastic Neighbor Embedding

$$\mathbb{R} \ni c_3 := \sum_{i=1}^n \text{KL}(P_i || Q_i) = \sum_{i=1}^n \sum_{j=1, j \neq i}^n p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \quad (29)$$

$$\frac{\partial c_3}{\partial \mathbf{y}_i} = 4 \sum_{j=1}^n (p_{ij} - q_{ij}) (1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^{-1} (\mathbf{y}_i - \mathbf{y}_j) \quad (30)$$

For derivation of gradient, see [2].

As before, optimization is done using gradient descent with momentum.

For t-SNE, there is no need to add jitter to the solution of initial iterations because it is more robust than SNE.

Early Exaggeration

In t-SNE, it is better to multiply all p_{ij} 's by a constant (e.g., 4) in the initial iterations:

$$p_{ij} := p_{ij} \times 4, \quad (31)$$

which is called **early exaggeration**.

- This heuristic helps the optimization to focus on the large p_{ij} 's (close neighbors) more in the early iterations.
- This is because large p_{ij} 's are affected more by multiplying by 4 than the small p_{ij} 's.

After the neighbors are embedded close to one another, we are free not to do it and let far away points be handled as well. Note that the early exaggeration is optional and not mandatory.

General Degrees of Freedom in t-SNE

$$\text{Cauchy distribution: } f(z) = \frac{1}{\pi(1 + z^2)} \quad (32)$$

$$\text{Student-t distribution: } f(z) = \frac{\Gamma(\frac{\delta+1}{2})}{\sqrt{\delta \times \pi} \Gamma(\frac{\delta}{2})} \left(1 + \frac{z^2}{\delta}\right)^{-\frac{\delta+1}{2}} \quad (33)$$

$$q_{ij} = \frac{(1 + z_{ij}^2/\delta)^{-(\delta+1)/2}}{\sum_{k \neq i} (1 + z_{kl}^2/\delta)^{-(\delta+1)/2}} \quad (34)$$

$$\mathbb{R} \ni p_{j|i} := \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}, \quad \mathbb{R} \ni p_{ij} := \frac{p_{i|j} + p_{j|i}}{2n} \quad (35)$$

$$\mathbb{R} \ni d_{ij}^2 := \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_i^2} \quad (36)$$

General Degrees of Freedom in t-SNE

For determining the degrees of freedom, we can do one the following items:

- ① Set δ to be fixed
- ② $\delta = h - 1$
 - ▶ Volume $\propto \exp(h)$ [example: πr^2 and $(4/3)\pi r^3$]
 - ▶ In Eq. (33): $f(z) \propto \exp(-\delta)$
 - ▶ $\delta \propto h$, special case: $\delta = 1, h = 2 \implies \delta = h - 1$
- ③ Updating both variables δ and $\{\mathbf{y}_i\}_{i=1}^n$ by restricted Boltzmann machine [3] or backpropagation
- ④ Alternating optimization approach [4]: alternate between δ and $\{\mathbf{y}_i\}_{i=1}^n$ in optimization

General Degrees of Freedom in t-SNE

$$\mathbb{R} \ni c_3 := \sum_i \text{KL}(P_i || Q_i) = \sum_i \sum_{j \neq i} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \quad (37)$$

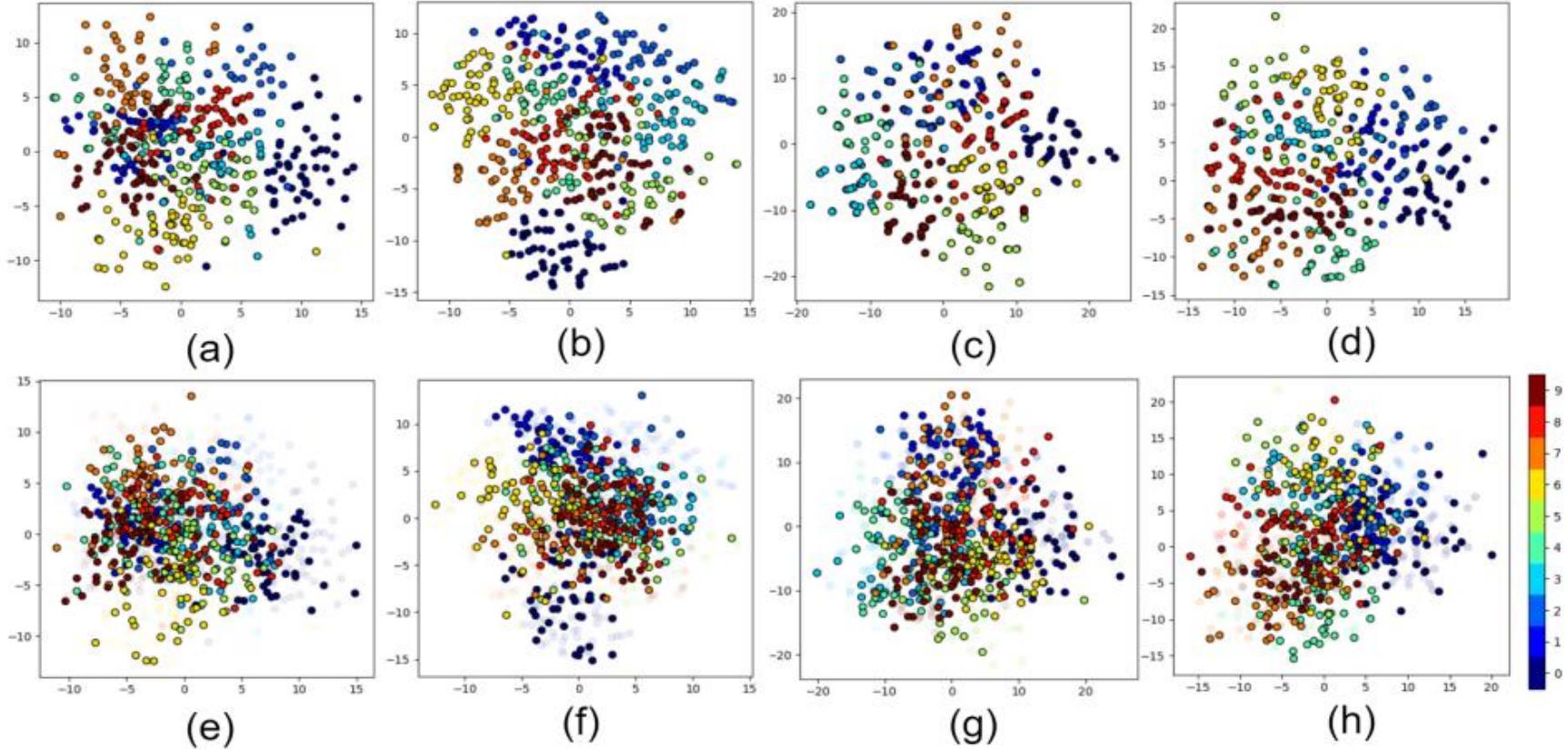
$$\frac{\partial c_3}{\partial \delta} = \sum_{i \neq j} \left(\frac{-(1 + \delta)z_{ij}^2}{2\delta^2(1 + \frac{z_{ij}^2}{\delta})} + \frac{1}{2} \log\left(1 + \frac{z_{ij}^2}{\delta}\right) \right) (p_{ij} - q_{ij}) \quad (38)$$

$$\delta := \delta - \text{sign}\left(\frac{\partial c_3}{\partial \delta}\right) \quad (39)$$

$$\frac{\partial c_3}{\partial \mathbf{y}_i} = \frac{2\delta + 2}{\delta} \times \sum_j (p_{ij} - q_{ij}) \left(1 + \frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2^2}{\delta}\right)^{-1} (\mathbf{y}_i - \mathbf{y}_j) \quad (40)$$

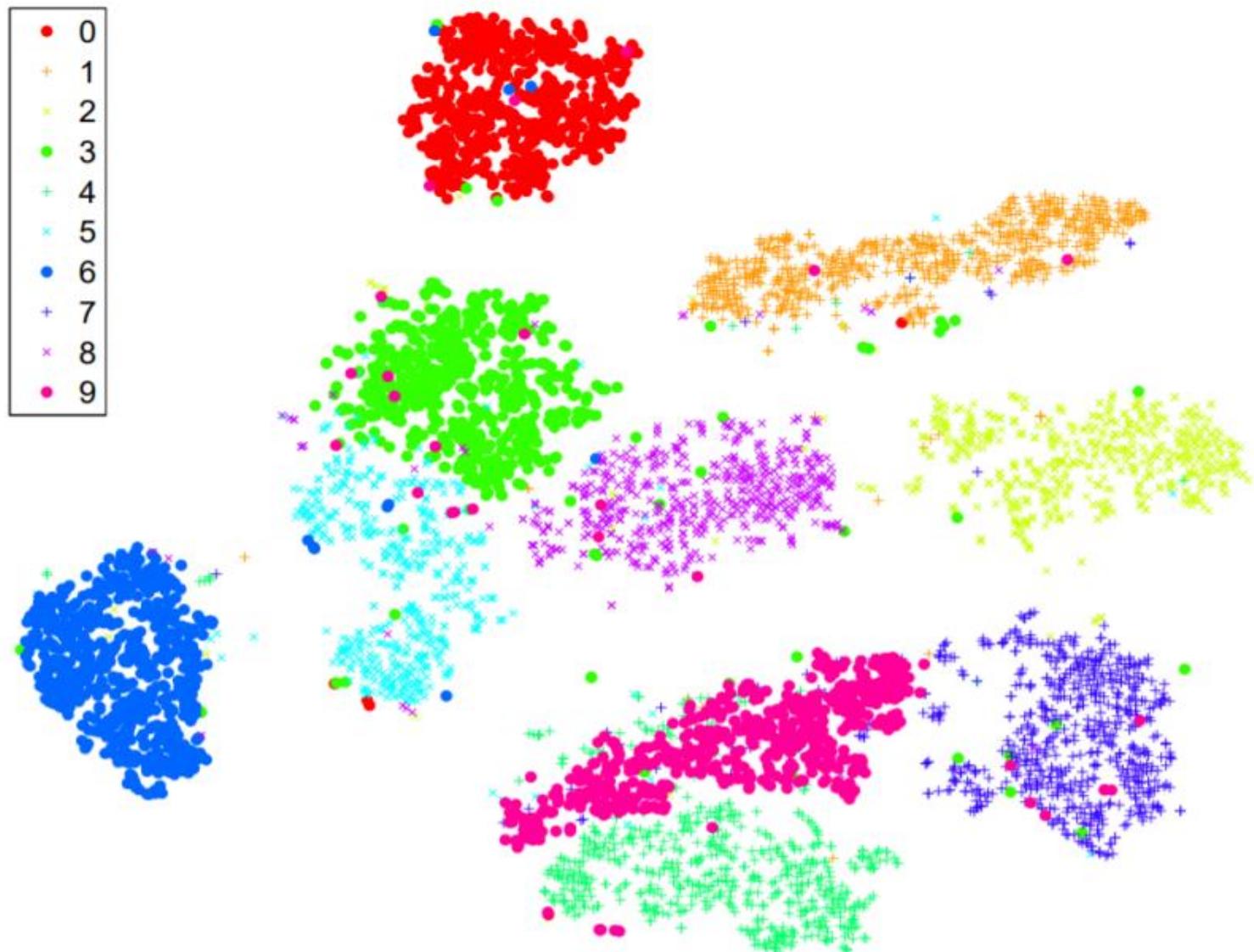
For derivation of gradients, see [2].

Examples



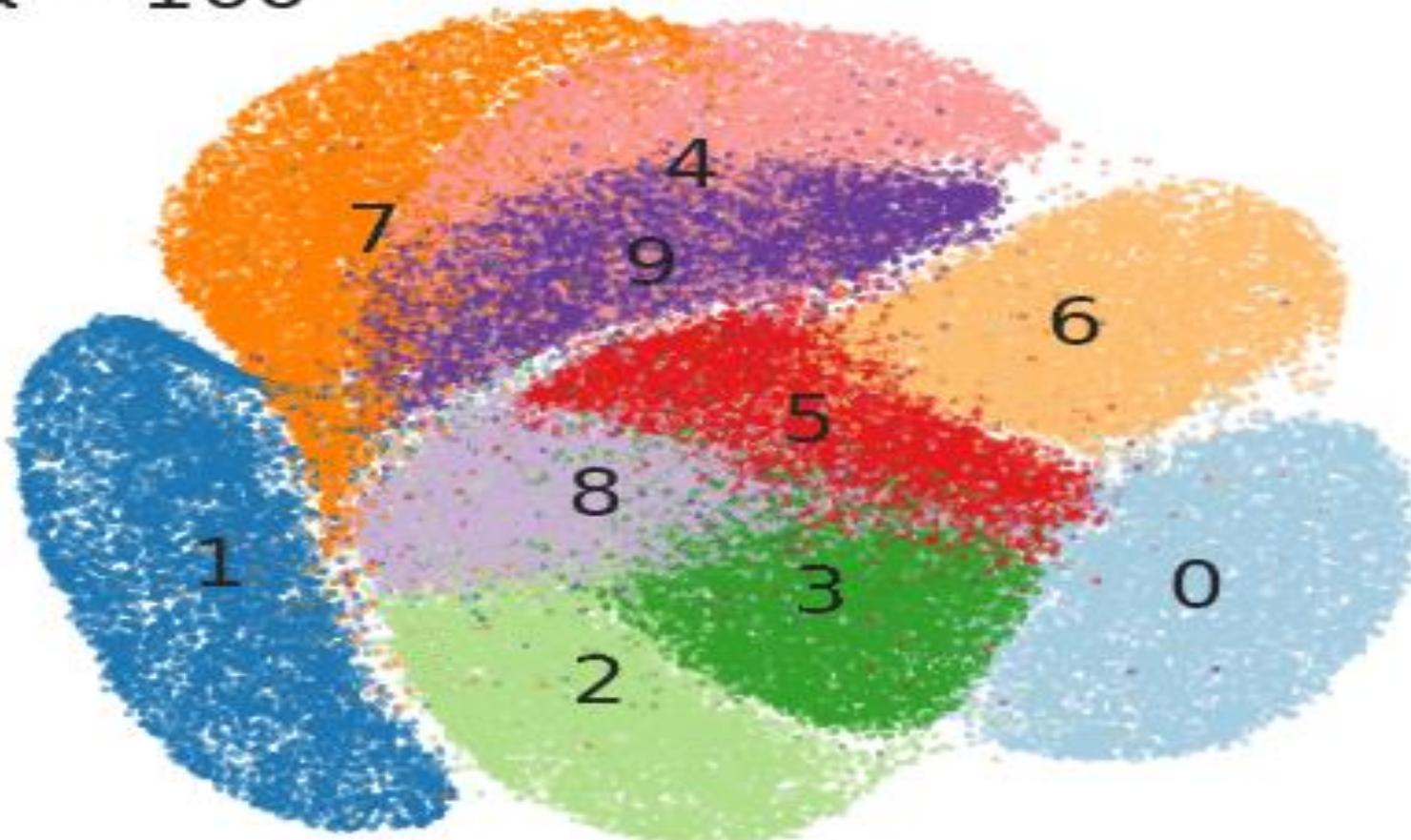
The embeddings of training data are shown in (a) SNE, (b) symmetric SNE, (c) t-SNE (Cauchy-SNE), and (d) t-SNE with general degrees of freedom. The out-of-sample embeddings are shown in (e) SNE, (f) symmetric SNE, (g) t-SNE (Cauchy-SNE), and (h) t-SNE with general degrees of freedom.

Examples on MNIST dataset



The credit of this image is for [5].

$\alpha = 100$



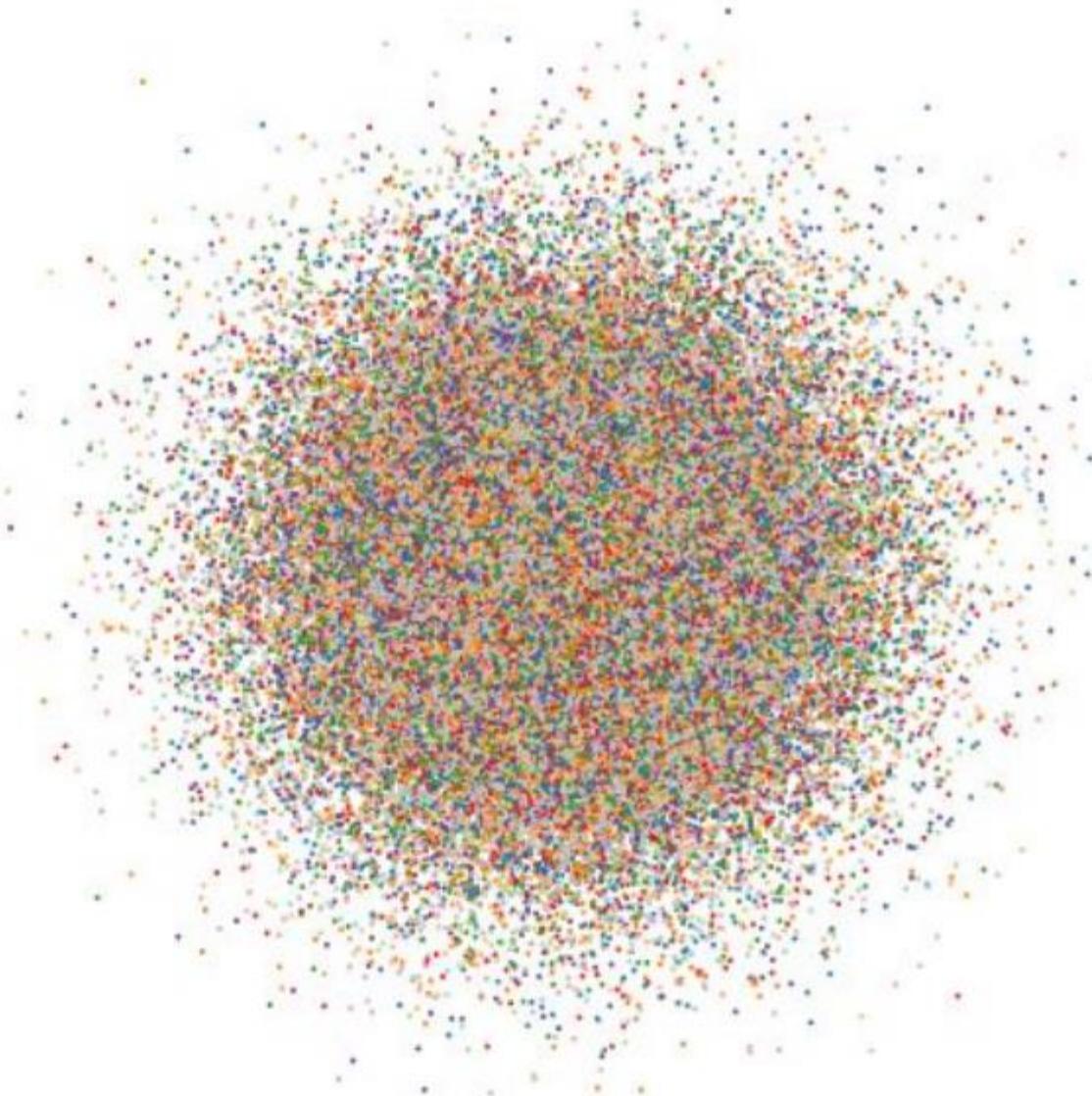
Early exaggeration

Multiply all
attractive forces
by 12 for 250
iterations.

Note that the
learning rate
should be high
enough for this
to work:

$$\eta = n/12$$

(Belkina et al., 2019)



- Tutorial paper: “Stochastic Neighbor Embedding with Gaussian and Student-t Distributions: Tutorial and Survey” [2]
- Tutorial YouTube videos by Prof. Ali Ghodsi at University of Waterloo: [\[Click here\]](#)

- [1] G. E. Hinton and S. T. Roweis, “Stochastic neighbor embedding,” in *Advances in neural information processing systems*, pp. 857–864, 2003.
- [2] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, “Stochastic neighbor embedding with Gaussian and Student-t distributions: Tutorial and survey,” *arXiv preprint arXiv:2009.10301*, 2020.
- [3] L. van der Maaten, “Learning a parametric embedding by preserving local structure,” in *Artificial Intelligence and Statistics*, pp. 384–391, 2009.
- [4] P. Jain and P. Kar, “Non-convex optimization for machine learning,” *arXiv preprint arXiv:1712.07897*, 2017.
- [5] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Fast approximate implementations

Vanilla t-SNE has $O(n^2)$ attractive and repulsive forces. To speed it up, we need to deal with both.

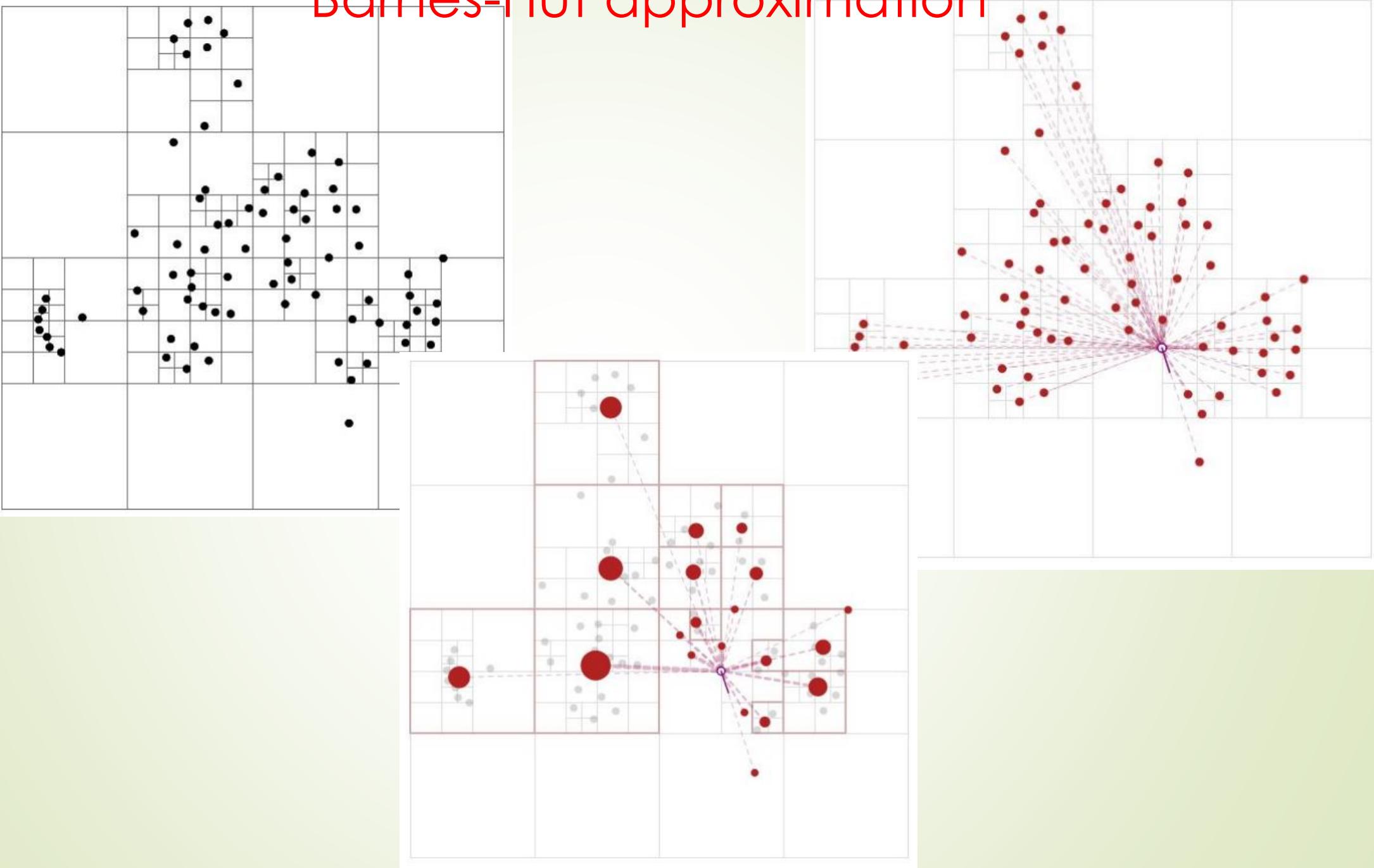
Attractive forces:

- Only use a small number of non-zero affinities, i.e. a sparse k-nearest-neighbour (kNN) graph. This reduces the number of forces. (Standard heuristic: $k = 3P$. For $P = 30$, this gives $k = 90$.)
- Use approximate kNN graphs. This speeds up graph construction.

Repulsive forces:

- Barnes-Hut t-SNE (BH t-SNE, 2013): $O(n \log(n))$
- FFT-accelerated interpolation-based t-SNE (FIt-SNE, 2019): $O(n)$
- Noise contrastive estimation / negative sampling (NCVis, 2020): $O(n)$

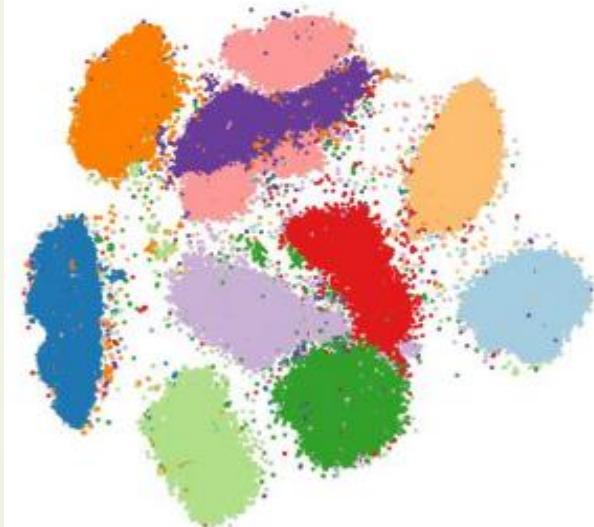
Barnes-Hut approximation



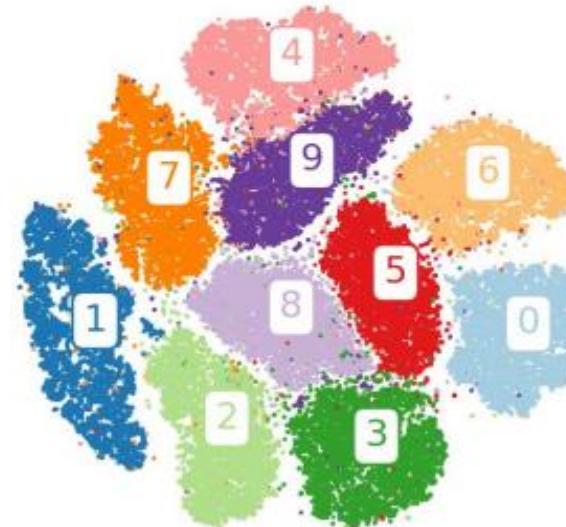
Perplexity and the number of neighbours

Perplexity can be seen as the ‘effective’ number of neighbours that enter the loss function. Default perplexity is 30.

Perplexity 300



Perplexity 30



Perplexity 3

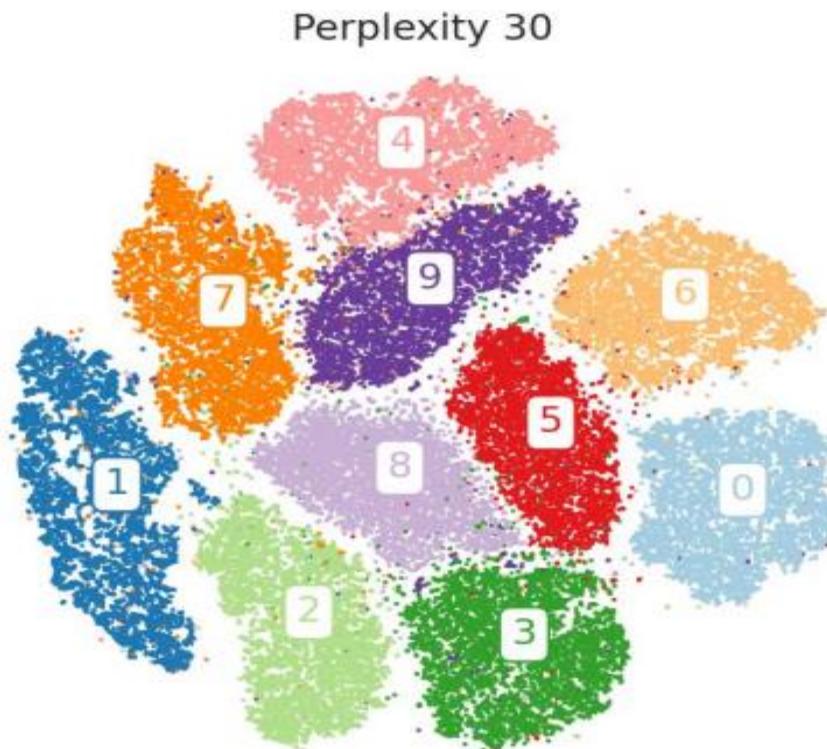


Much smaller values are rarely useful.

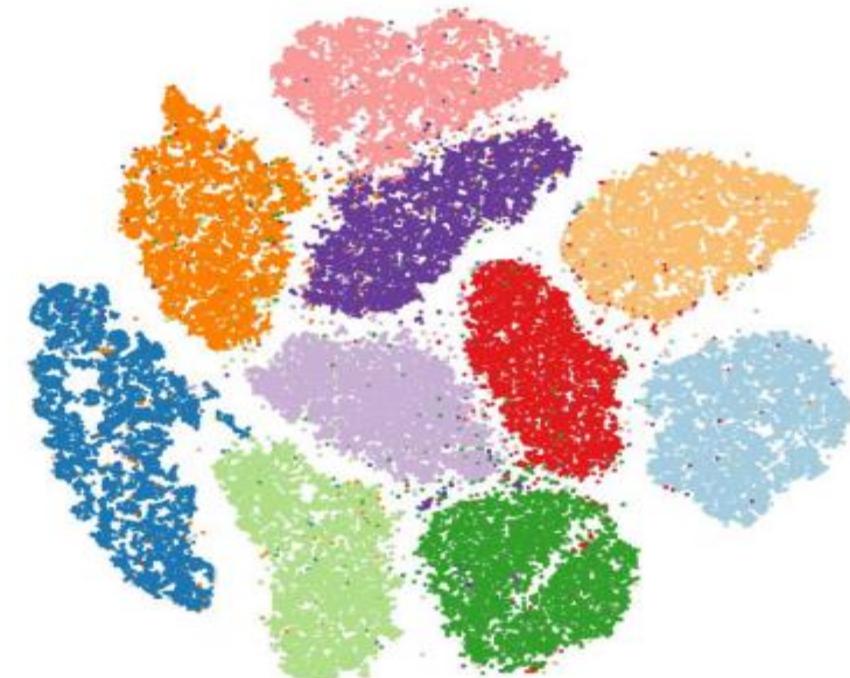
Much larger values are impractical or even computationally prohibitive.

Uniform affinity

Gaussian affinities with perplexity P can usually be replaced by the uniform affinities with $k \approx P/2$.



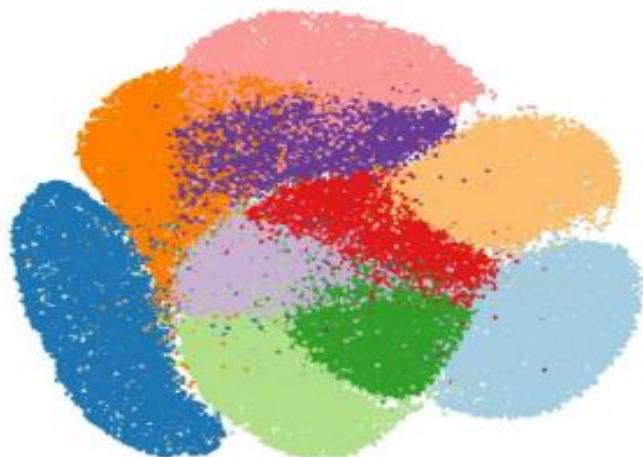
Uniform affinity kernel, k=15



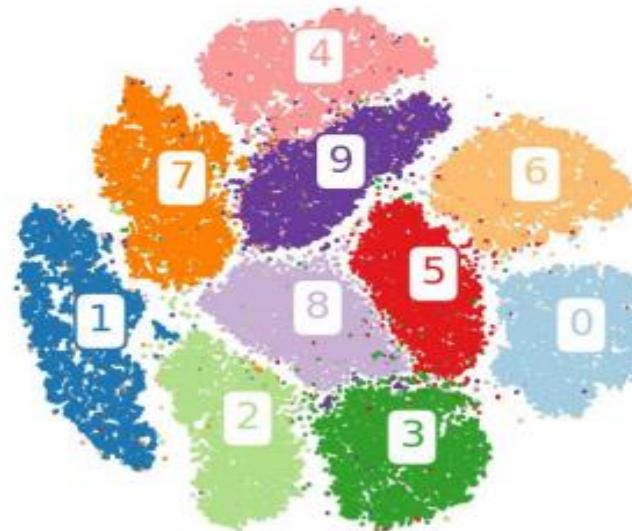
Low-dimensional similarity kernel

The main innovation of t-SNE compared to SNE was the Cauchy kernel, addressing the ‘crowding problem’ of SNE.

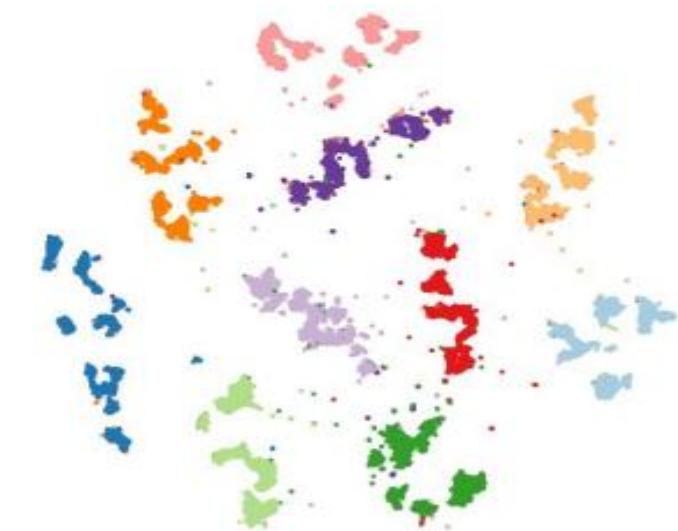
Gaussian kernel



Cauchy kernel



Heavier-tailed kernel

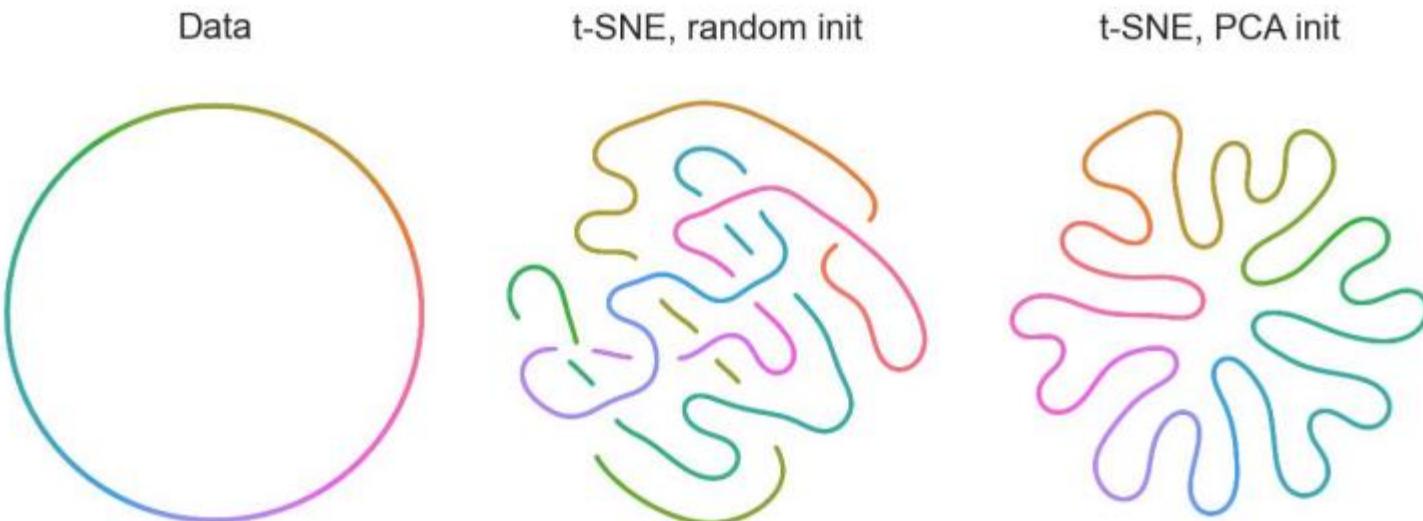


Even heavier-tailed kernels can bring out even finer cluster structure.

(Kobak et al., 2020)

The role of initialization

t-SNE preserves local structure (neighbours) but often struggles to preserve global structure. The loss function has many local minima and initialization can play a large role.



Always use informative initialization, e.g. PCA.

(Kobak and Linderman, 2021)