Reducción de dimensión

Aprendizaje automático

Juan David Martínez
jdmartinev@eafit.edu.co

Agenda

- Reducción de dimensión
- Análisis de componentes principales (PCA)
- Otras formas de reducción de dimensión

Reducción de dimensión

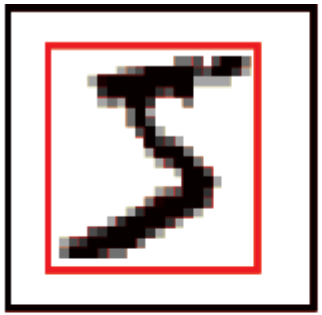
Muchos problemas de **Machine Learning** implican **millones de características** para cada ejemplo (instancia de entrenamiento)

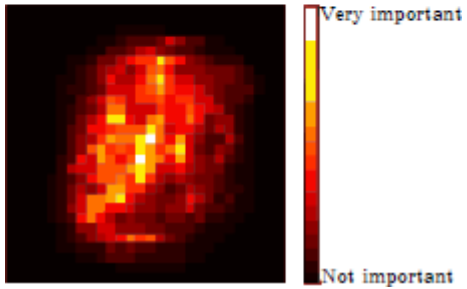
- Extremadamente lento
- Difícil de encontrar una buena solución

Este problema se conoce como la maldición de la dimensionalidad

Reducción de dimensión

Es posible reducir considerablemente el número de características

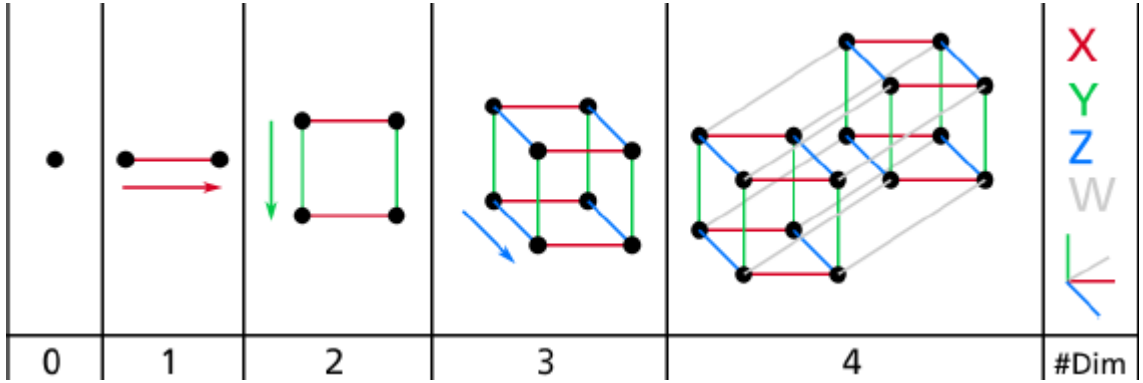




- Los píxeles en el borde generalmente son blancos
- Se pueden eliminar sin perder mucha información
- En el proceso de reducción de dimensión **se pierde algo de información**
- Filtra ruidos y detalles innecesarios y puede mejorar el rendimiento
- Acelera el entrenamiento y facilita la visualización

La maldición de la dimensión

Estamos acostumbrados a vivir en **tres dimensiones,** la intuición nos falla cuando intentamos imaginarnos un espacio de alta dimensión

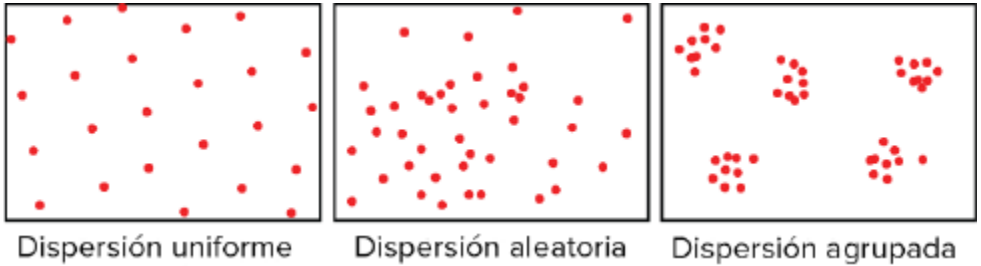


La maldición de la dimensión

- En un cuadrado de lado 1, la distancia entre dos puntos será en promedio 0,52.
- En un cubo de lado 1, la distancia promedio será 0,66
- En un hipercubo de lado 1 de 1 millón de dimensiones, la distancia promedio será de 408,25. Es decir, la mayoría de muestras están muy lejos una de la otra
- ¡Entre más dimensiones, mayor riesgo de sobre-entrenamiento!
- Solución: Aumenta el tamaño de la muestra.
- Problema: El número de muestras requeridas para alcanzar una densidad dada crece exponencialmente con el número de dimensiones.
- Con 100 características, se necesitarían más instancias que átomos en el universo
- Solución: Reducir el número de características

Enfoques principales: Proyección

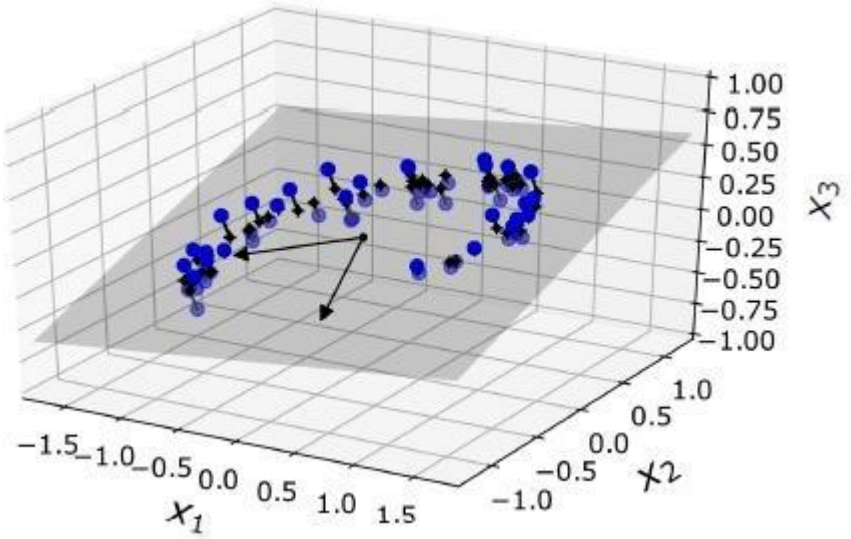
- Generalmente los datos no se distribuyen uniformemente en todas las dimensiones

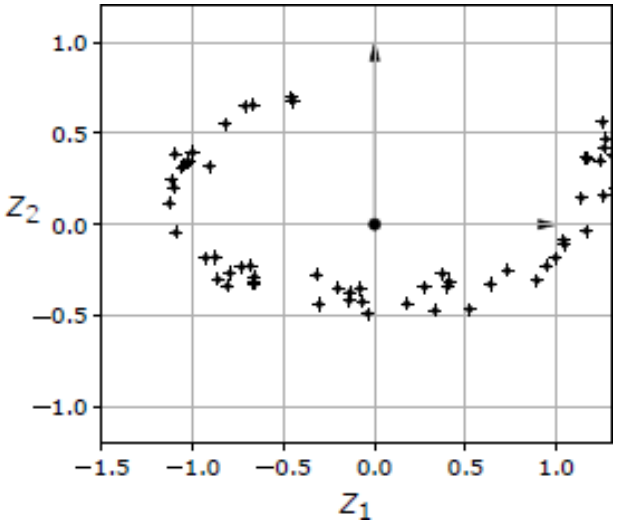


- Muchas características son casi constantes mientras que otras están altamente correlacionadas
- Las muestras se encuentran dentro o cerca de un subespacio de dimensiones mucho más bajas

Proyección: Ejemplo 1

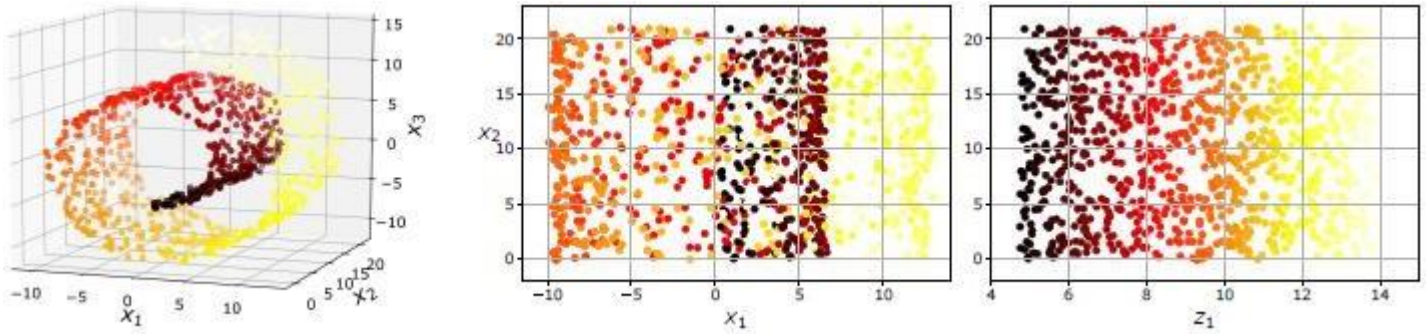
Todas las muestras se encuentran cerca de un plano de menor dimensión (2D)





Proyectamos todas las muestras al nuevo espacio de representación z_1 y z_2

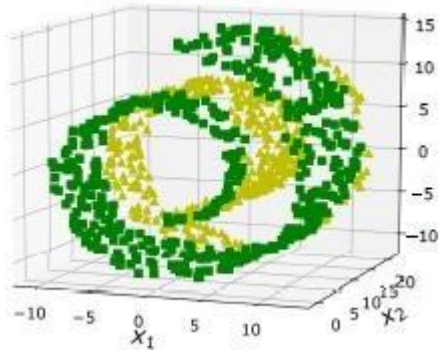
Proyección: Ejemplo 2

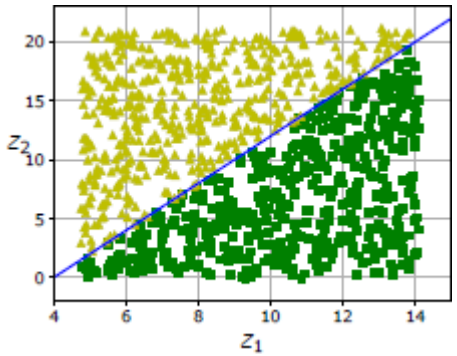


- La proyección en un plano (por ejemplo quitando x_3) aplastaría capas del rollo suizo (centro)
- Lo que deseamos es desenrollar el rollo (derecha)

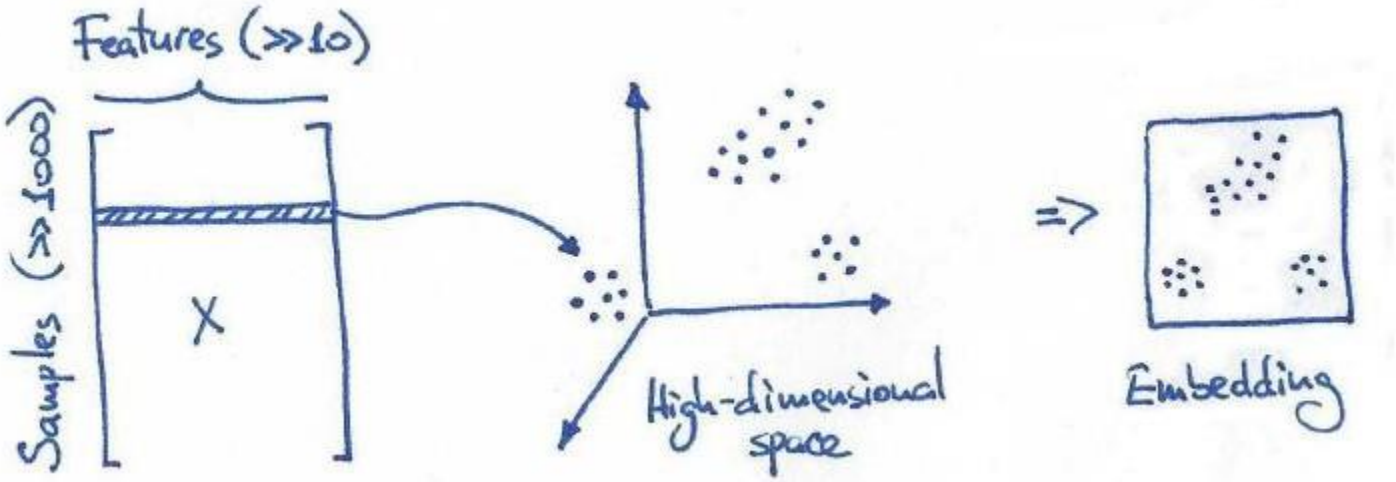
Enfoques principales: Manifold learning

- El rollo suizo es un ejemplo de una variedad 2D
- Una variedad d −dimensional es una parte de un espacio n −dimensional ($d << n$) que localmente se parece a un hiperplano d −dimensional
- Manifold assumption: La mayoría de los conjuntos de datos de alta dimensión se encuentran cerca de un manifold de mucha más baja dimensión
- La tarea a resolver (clasificación, regresión) será más simple en este nuevo espacio





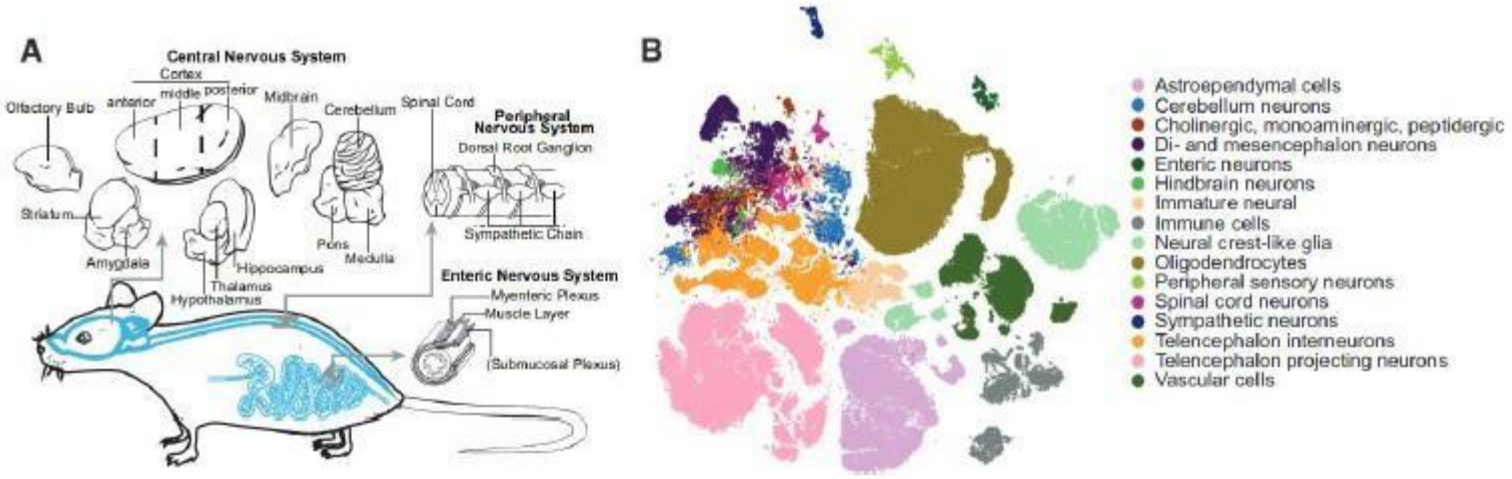
Visualización de datos t-SNE



- Unsupervised non-parametric methods for dimensionality reduction (non-linear methods)

Visualización de datos t-SNE

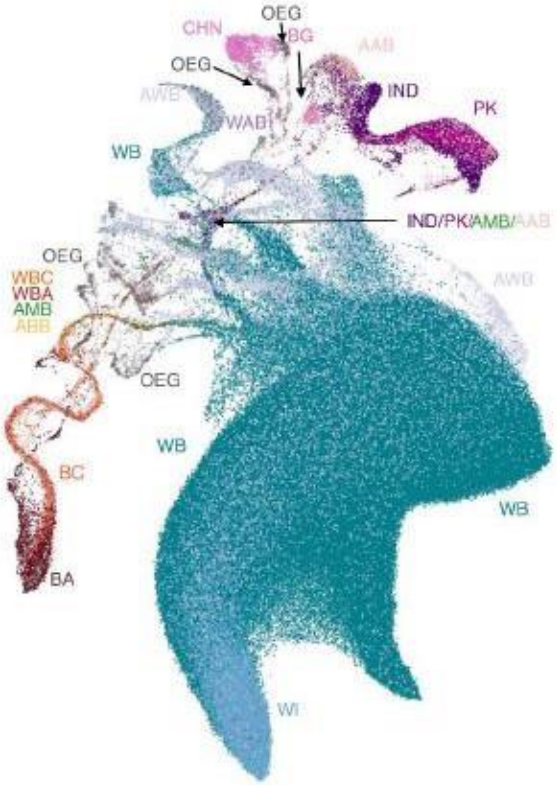
Single-cell transcriptomics (single-cell RNA sequencing): samples are cells, features are genes.



Zeisel et al. (2018)
$n \approx 500{,}000$

Visualización de datos t-SNE

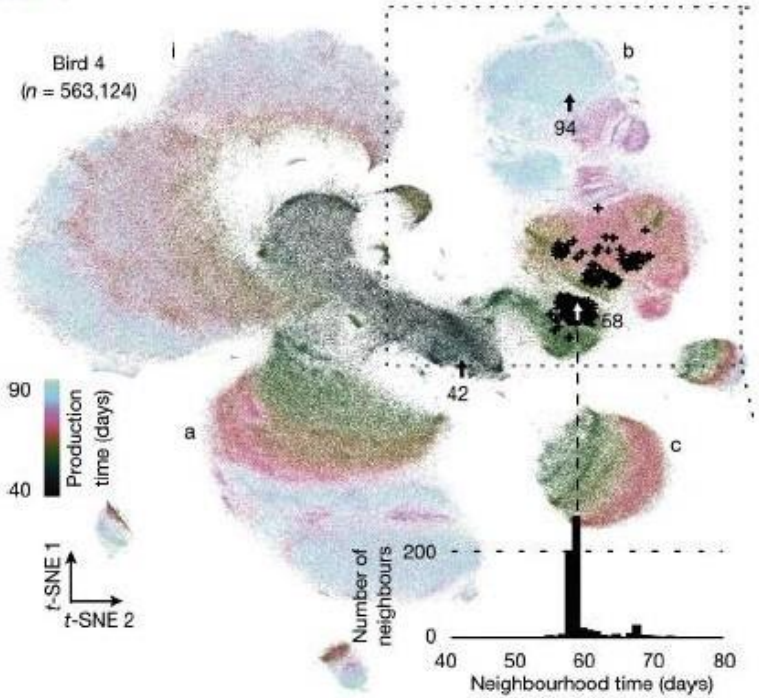
Population genomics: samples are people, features are single-nucleotide polymorphims.



Diaz-Papkovich et al. (2019)
$n \approx 500{,}000$

Visualización de datos t-SNE

Behavioural physiology: samples are syllable renditions, features are spectrogram bins.



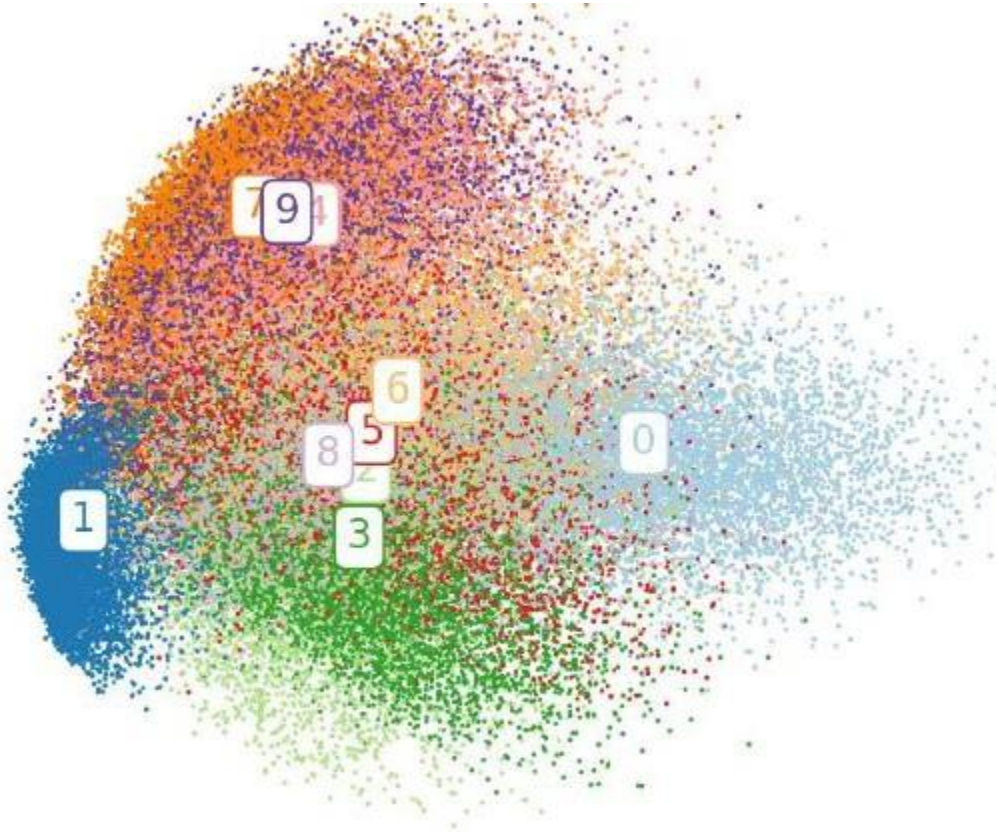
Kollmorgen et al. (2020)
$n \approx 600{,}000$

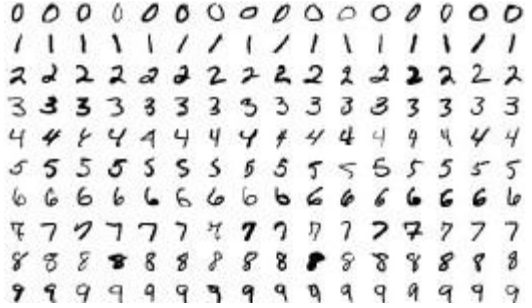
t-SNE MNIST

$n = 70{,}000$

28×28 images $= 784$ pixels

t-SNE MNIST





Multidimensional scaling MDS

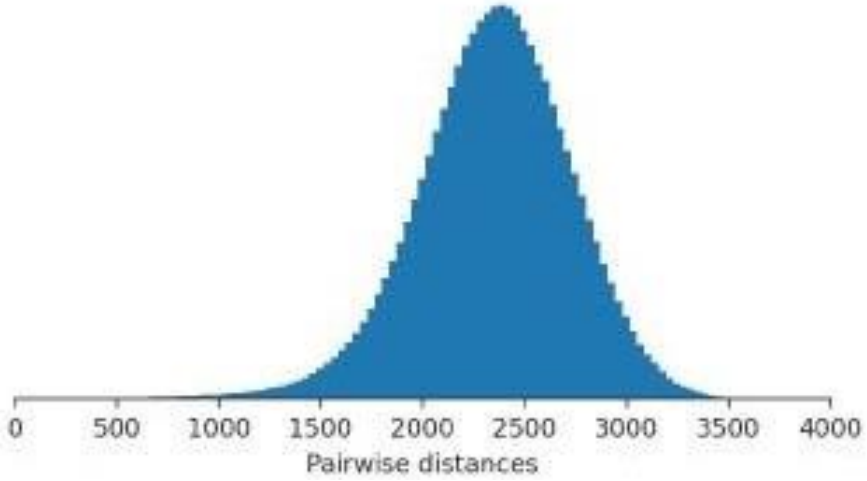
- MSD: Arrange points in 2D to approximate high-dimensional pairwise distances

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

Here $n = 5{,}000$.

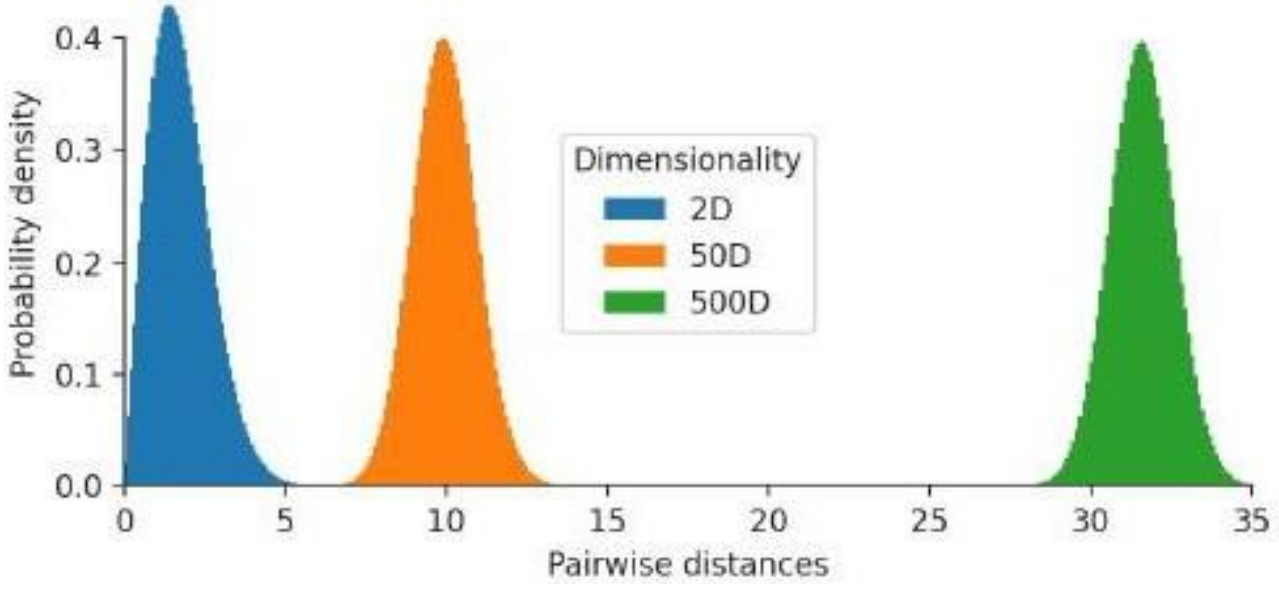
¿Por qué MSD falla?

- Preserving high-dimensional distances is usually a bad idea because it is not posible to preserve them (curse of dimensionality)



¿Por qué MSD falla?

- Pairwise distances between points in a standard Gaussian:



Neighbour embeddings

- Idea: Preserve nearest neighbours instead of preserving distances

[PDF] **Stochastic neighbor embedding**

G Hinton, ST Roweis - NIPS, 2002 - Citeseer

We describe a probabilistic approach to the task of placing objects, described by high-dimensional vectors or by pairwise dissimilarities, in a low-dimensional space in a way that preserves neighbor identities. A Gaussian is centered on each object in the high ...

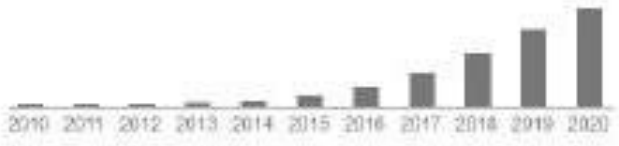
☆ 99 Cited by 1464 Related articles All 17 versions

[PDF] Visualizing data using **t-SNE**.

L Van der Maaten, G Hinton - Journal of machine learning research, 2008 - jmlr.org

We present a new technique called "**t-SNE**" that visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map. The technique is a variation of Stochastic Neighbor Embedding (Hinton and Roweis, 2002) that is much easier to optimize ...

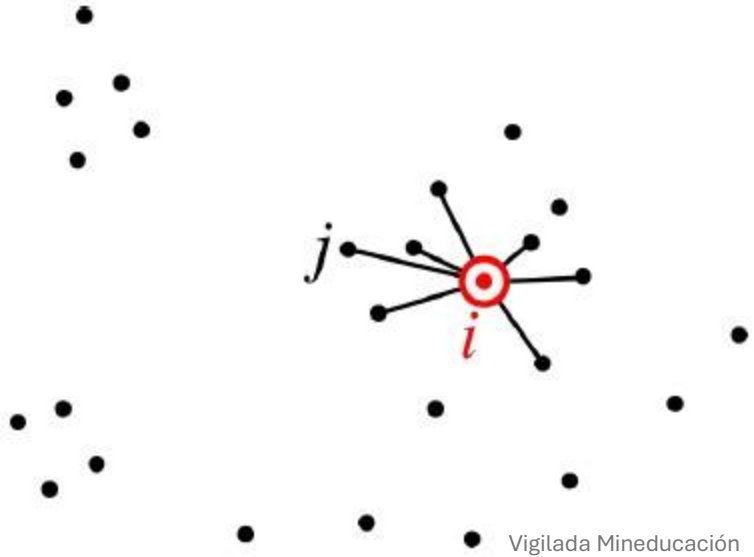
☆ 99 Cited by 18533 Related articles All 52 versions



MNIST t-SNE

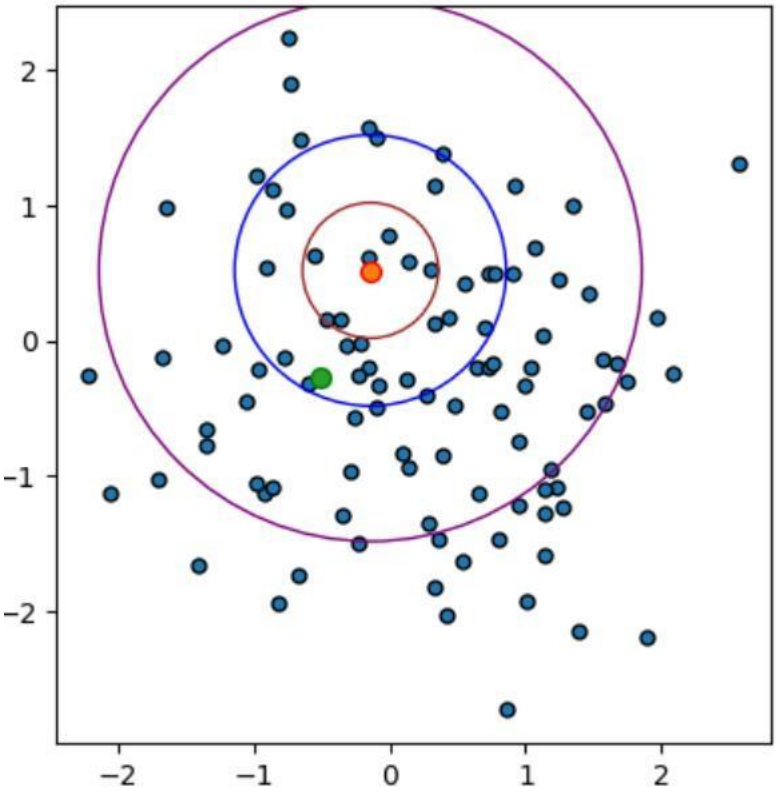
Stochastic neighbour embedding

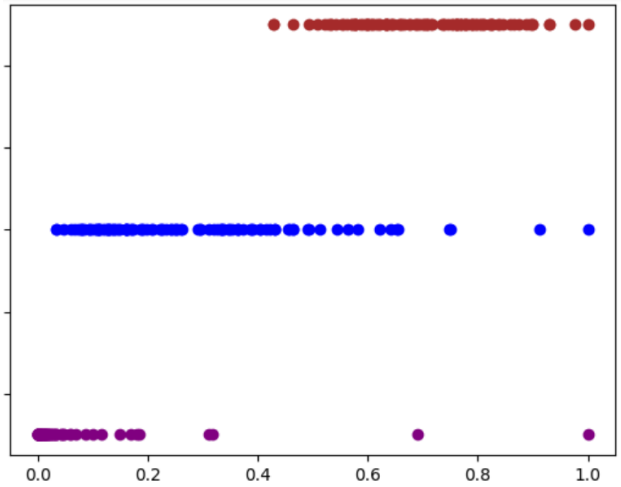
- Loss function: Kullback-Leibler divergence between pairwise similarities (affinities) in the high-dimensional and in the low-dimensional spaces. Similarities are defined such that they sum to 1



$$\mathcal{L} = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Similarity





$$\text{sim}_{i,j} = \exp\left(-\frac{1}{2\sigma^2}||\mathbf{x}_i - \mathbf{x}_j||^2\right)$$

```
s^2 = 1, d = 0.8780828489462678
s^2 = 2, d = 0.21952071223656694
s^2 = 0.5, d = 3.512331395785071
```

Stochastic neighbour embedding

- High dimensional similarities

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

- Kernel width is adaptively chosen to achieve de desided perpexlity (default: 30)

$$\mathcal{P} = 2^{\mathcal{H}}, \text{ where } \mathcal{H} = -\sum_{j\neq i} p_{j|i}\log_2 p_{j|i}$$

- Symmetrize and normalize to sum 1

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

- This defines similarities as non-zero, but most will be ≈ 0 and can be set to 0 without affecting the result. Uniform similarities can be also used

$$p_{j|i} = 1/k \text{ for } k \text{ nearest neighbours}$$

Stochastic neighbour embedding

- Low dimensional similarities

$$q_{ij} = \frac{w_{ij}}{Z}, \quad w_{ij} = k(\|\mathbf{y}_i - \mathbf{y}_j\|), \quad Z = \sum_{k \neq l} w_{kl}$$

- Similarity kernel in SNE:

$$k(d) = \exp(-d^2)$$

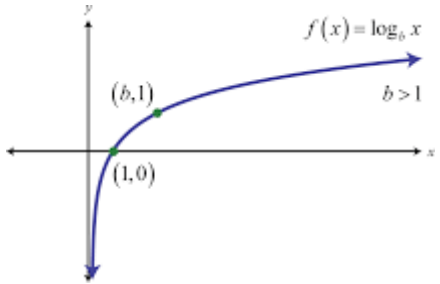
- Similarity kernel in t-SNE

$$k(d) = 1/(1 + d^2)$$

Gradient descent

- The los is optimized via gradient descent starting from a random set of points

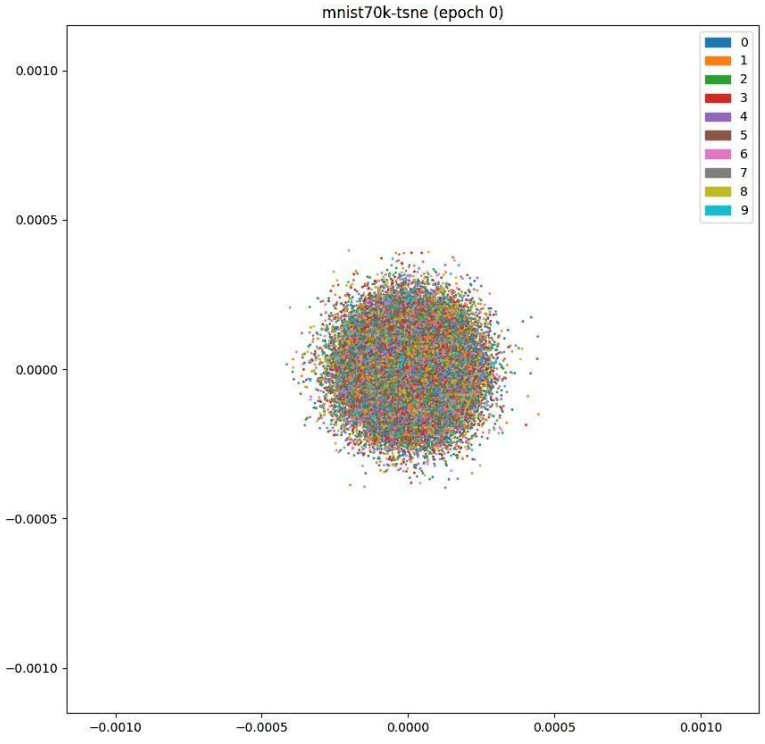
$$\mathcal{L} = -\sum_{i,j} p_{ij} \log q_{ij} = -\sum_{i,j} p_{ij} \log \frac{w_{ij}}{Z}$$
$$= -\sum_{i,j} p_{ij} \log w_{ij} + \log \sum_{i,j} w_{ij},$$

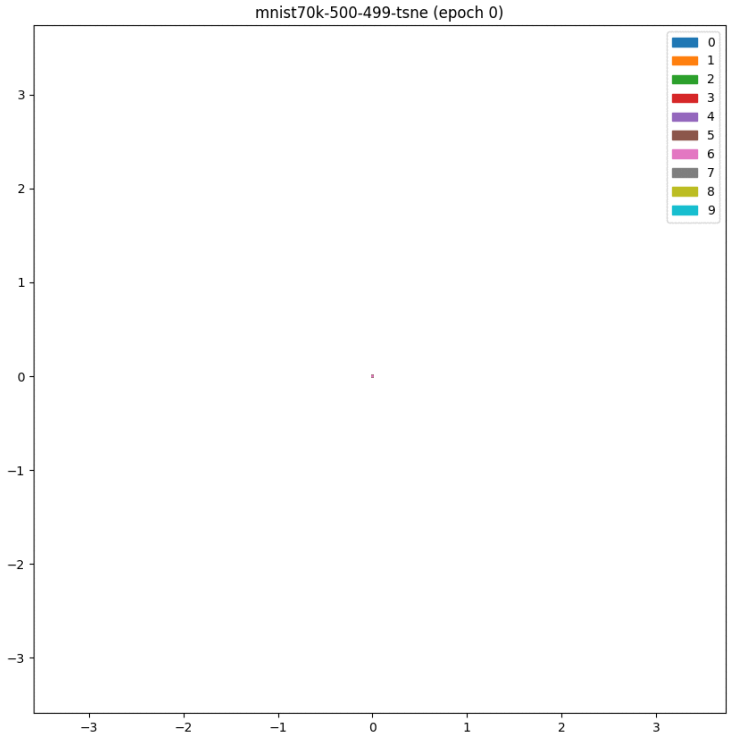


- This works as a many-body simulation: close neighbours attract each other while all points refuse each other

$$\frac{\partial \mathcal{L}_{\text{t-SNE}}}{\partial \mathbf{y}_i} = -2 \sum_j p_{ij} \frac{1}{w_{ij}} \frac{\partial w_{ij}}{\partial \mathbf{y}_i} + 2 \frac{1}{Z} \sum_j \frac{\partial w_{ij}}{\partial \mathbf{y}_i}$$
$$\sim \sum_j p_{ij} w_{ij} (\mathbf{y}_i - \mathbf{y}_j) - \frac{1}{Z} \sum_j w_{ij}^2 (\mathbf{y}_i - \mathbf{y}_j)$$

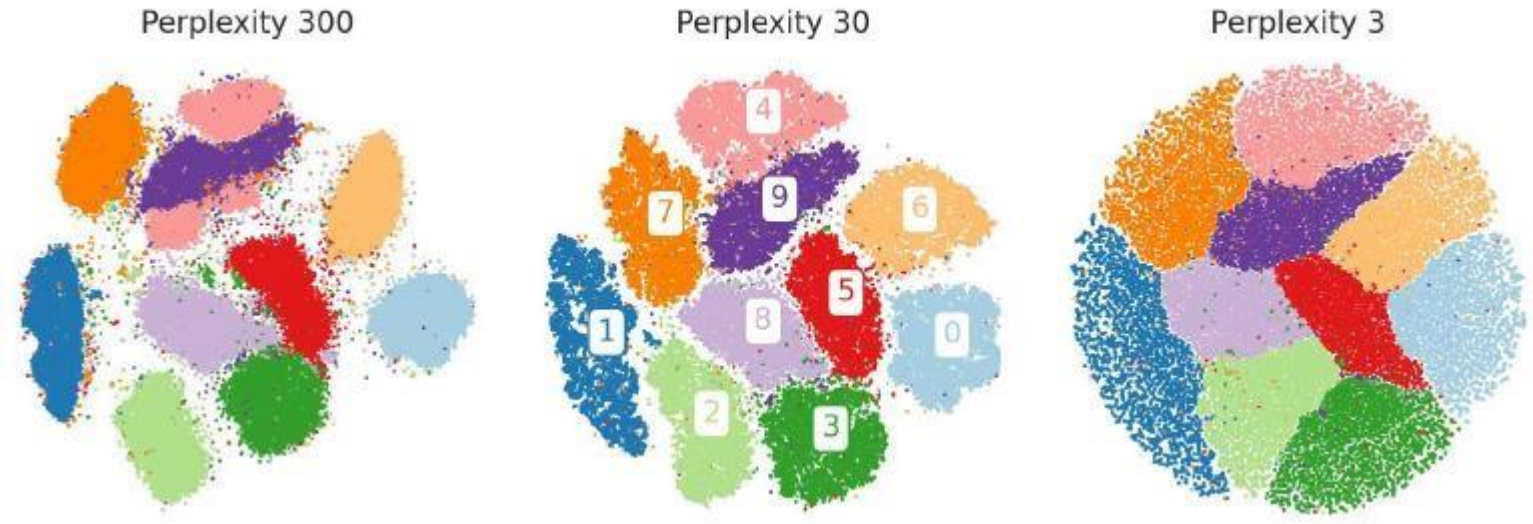
Early exageration





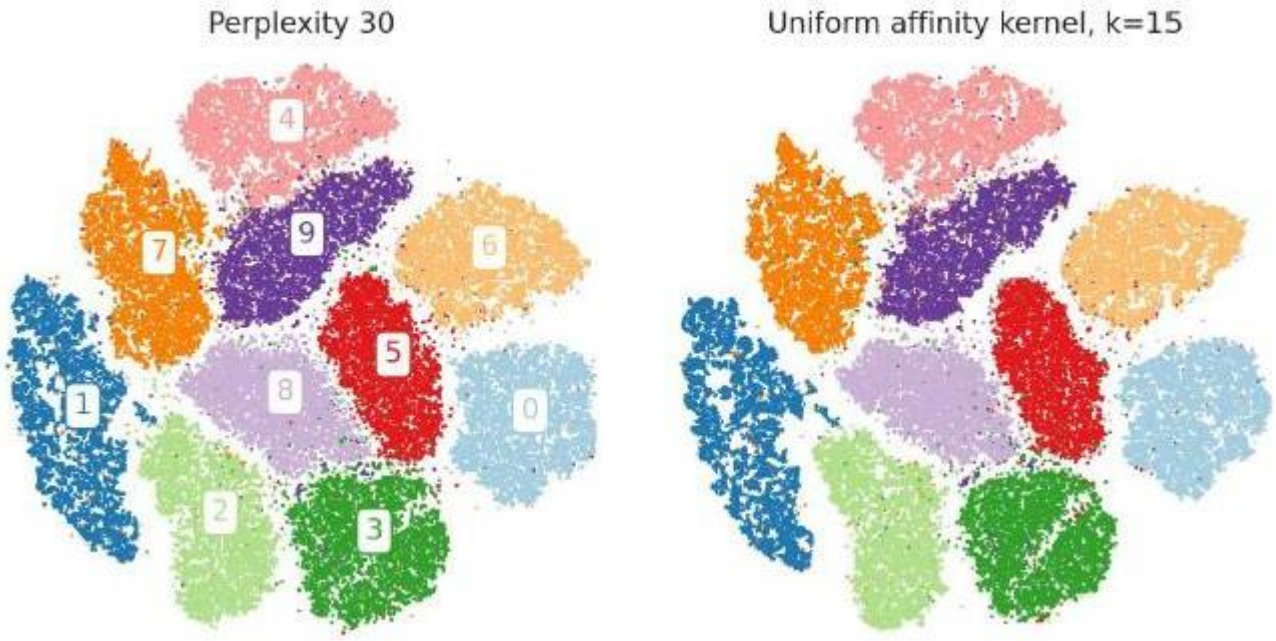
Perplexity and the number of neighbours

- Perplexity can be seen as the ‘efective’ number of neighbours that enter the los function. Default is 30
- Smaller values are rarely useful, higher values are computationally unfeasible



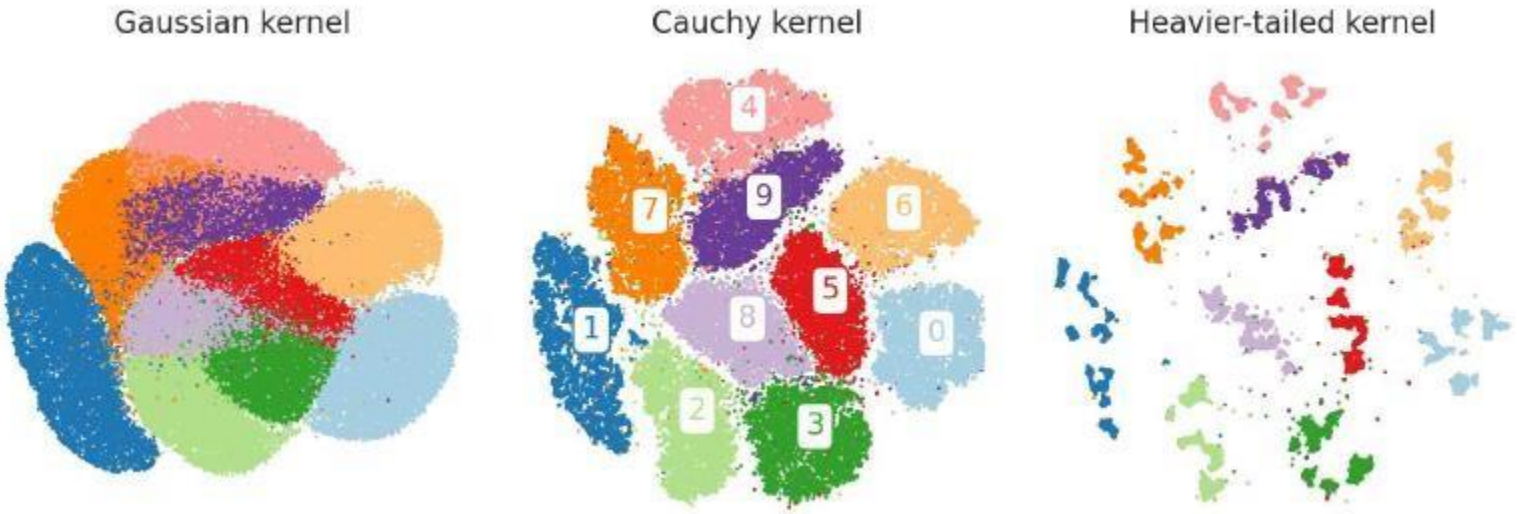
Uniform affinities

- Gaussian affinities with perplexity P can be replaced with uniform affinities with k = P/2



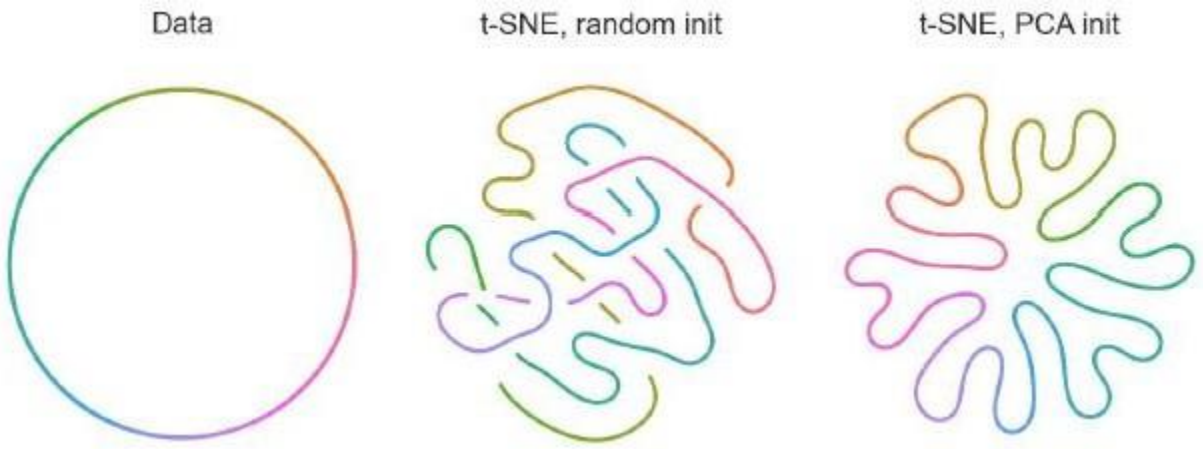
Low-dimensional similarity kernel

- The main innovation of tSNE with respect to SNE was the Cauccy kernel, addresing the crowding problema of SNE
- Even heavier-tailed kernels can bring out even finer cluster structure.



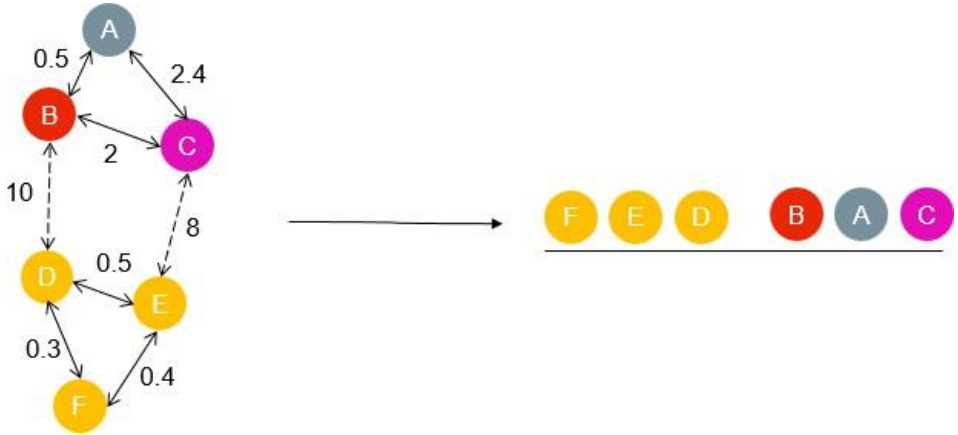
The role of initialization

- tSNE preservers local structure but fails preserving global structure. The cost function has a lot of local mínima and initialization plays an important role.
- Always use informative initialization like PCA



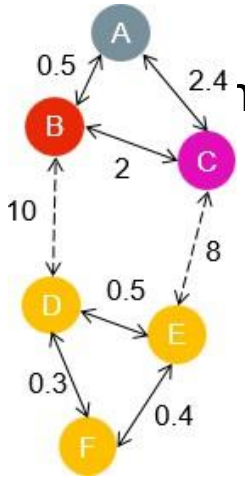
UMAP

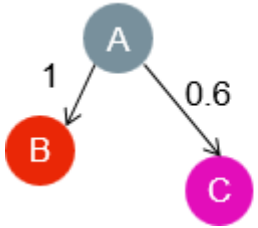
- UMAP is a dimensionality reduction technique that preserves as much of the data's local and global structure as possible.

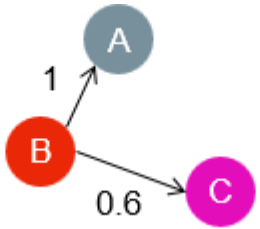


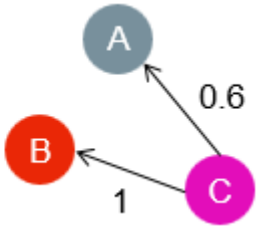
UMAP

- **High-Dimensional Graph Construction:** UMAP begins by constructing a weighted graph in the high-dimensional space. Each data point is connected to its nearest neighbors, creating a neighborhood graph. The connections (edges) are weighted $v_{j|i} = \exp[(-d(x_i, x_j) - \rho_i)/\sigma_i]$ nilarity, with closer points having stronger connections.







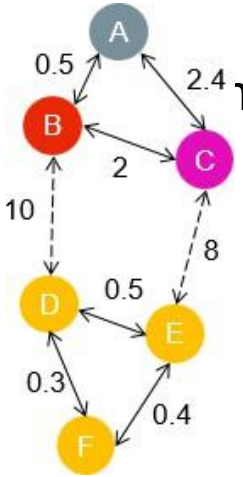


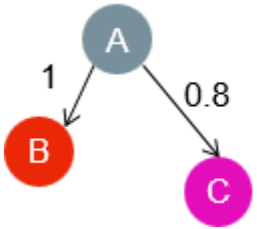
UMAP

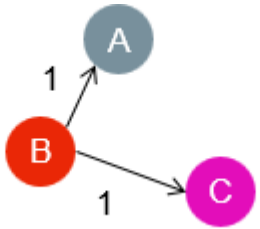
- **High-Dimensional Graph Construction:** UMAP begins by constructing a weighted graph in the high-dimensional space. Each data point is connected to its nearest neighbors, creating a neighborhood graph. The connections (edges) are weighted nilarity, with closer points having stronger connections.

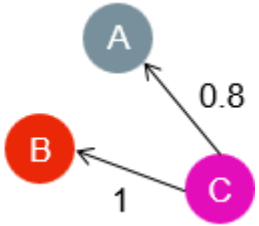
$$v_{j|i} = \exp[(-d(x_i, x_j) - \rho_i)/\sigma_i]$$

$$v_{ij} = (v_{j|i} + v_{i|j}) - v_{j|i}v_{i|j}$$









UMAP

- **Number of neighbors and σ**

Algorithm 3 Compute the normalizing factor for distances σ

function SmoothKNNDist(knn-dists, n, ρ)

Binary search for σ such that $\sum_{i=1}^{n} \exp(-(\text{knn-dists}_i - \rho)/\sigma) = \log_2(n)$

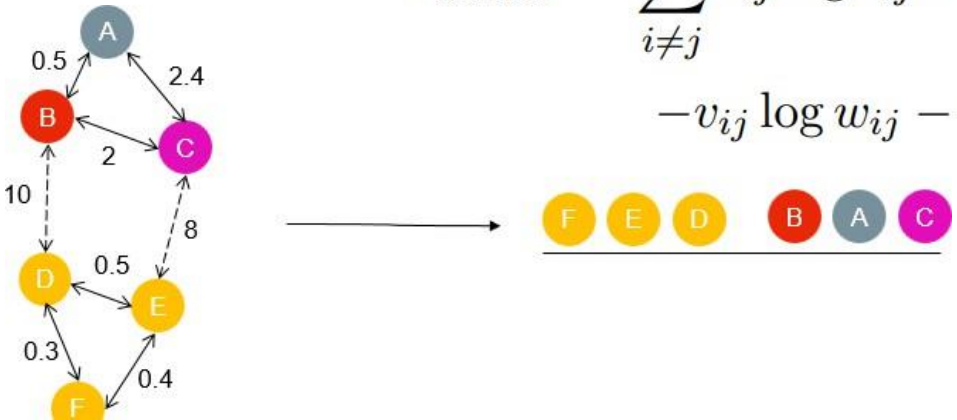
return σ

UMAP

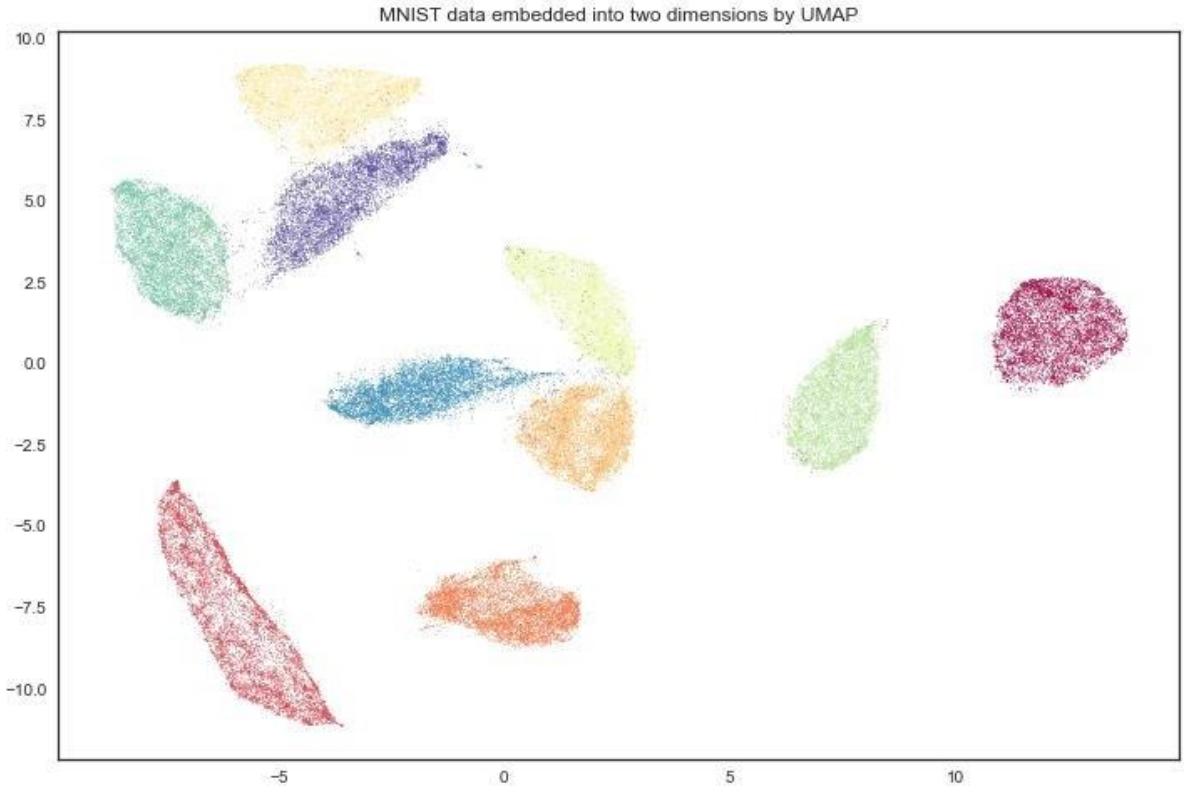
Optimization: UMAP then uses a force-directed graph layout algorithm in lower-dimensional space to find a representation that is as structurally similar as possible to the high-dimensional graph. The low dimensional similarities are given by:

$$w_{ij} = \left(1 + a \left\| y_i - y_j \right\|_2^{2b}\right)^{-1} \qquad a \approx 1.929 \text{ and } b \approx 0.7915$$

$$C_{UMAP} = \sum_{i \neq j} v_{ij} \log v_{ij} + (1 - v_{ij}) \log (1 - v_{ij})$$
$$- v_{ij} \log w_{ij} - (1 - v_{ij}) \log (1 - w_{ij})$$



UMAP - MNIST


MNIST data embedded into two dimensions by UMAP
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−5
0
5
10