Pattern Recognition
and Applications Lab

Dimensionality Reduction via PCA and t-SNE

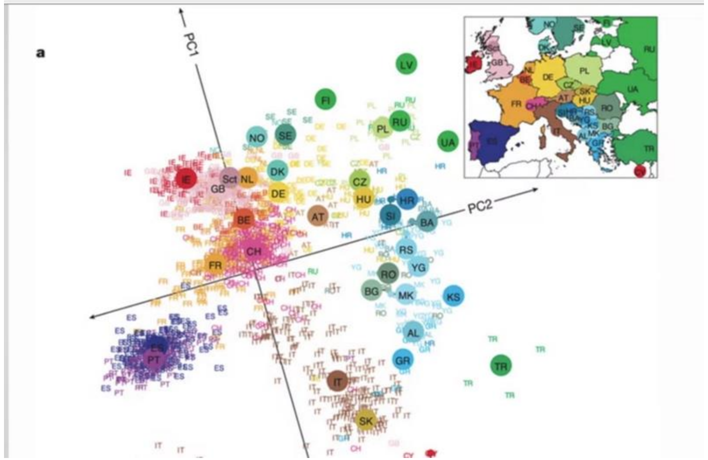
Fabio Brau

Department of Electrical and Electronic Engineering
University of Cagliari, Italy

Data processing and data visualization

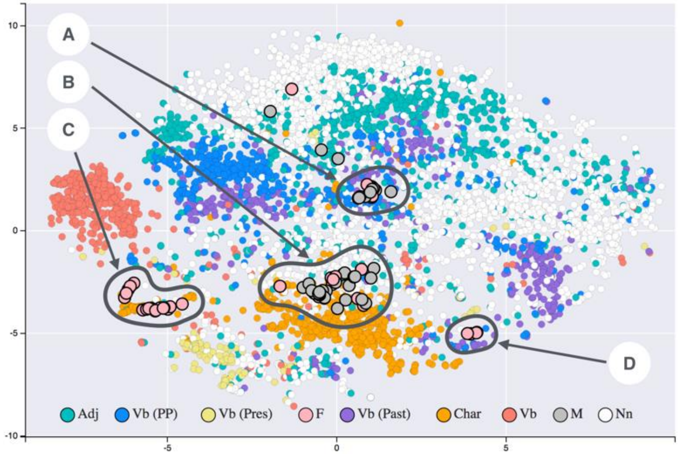
Dimensionality reduction includes a large numbers of algorithm for extracting the "useful" variables from our highly dimensional data. In this lecture we will focus on the two methods

Principal Component Analysis



Representation of the first two principal components of genetic data from a European population

Stochastic Neighbor Embedding



T-SNE representation of word embedding

Principal Component Analysis (PCA)

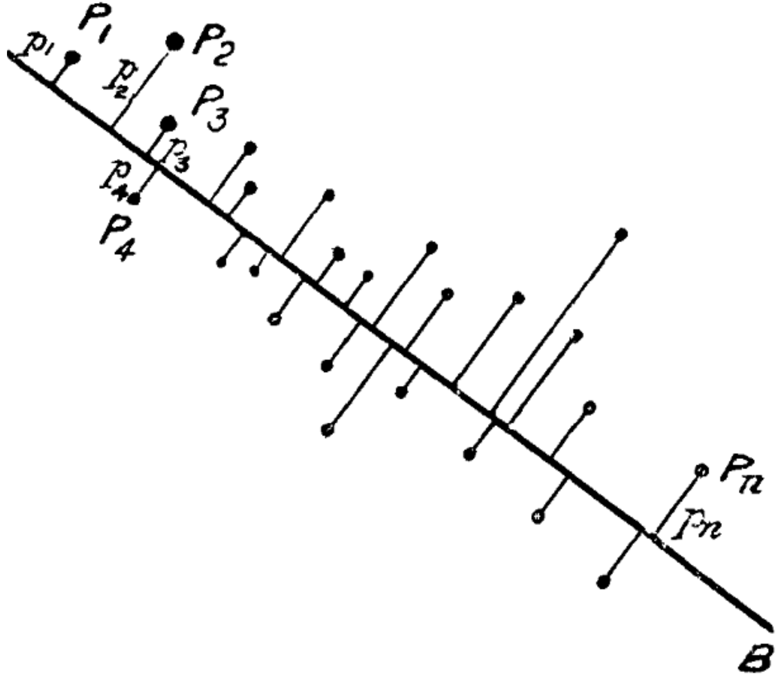
Introduction

- Karl Pearson, *Closest Fit to Systems of Points in Space,* 1901.

  Worked on the problem of finding the closest line to the data (Geometrical View).

- Harold Hotelling, *Relation between two sets of variates,* 1930.

  Introduced the term **principal component** focusing on factor analysis (Statistical View)



The best line which approximates a set of points, Pearson

Background and Notations

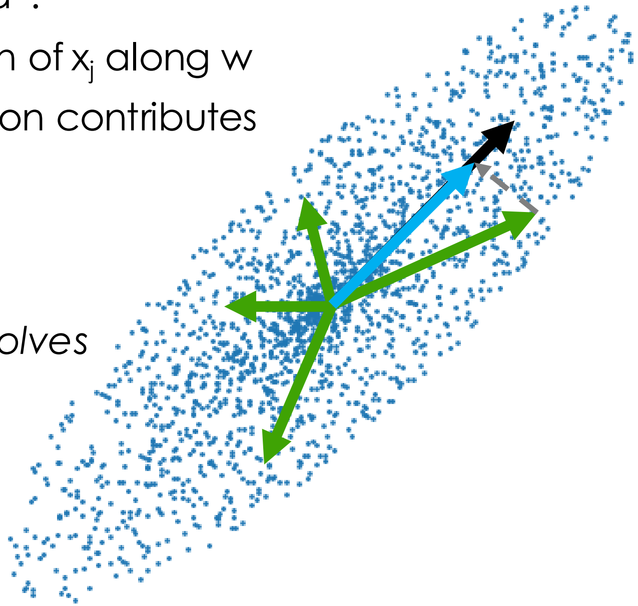
The data will be expressed in term of **observations** including several **variables**.

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} = \begin{bmatrix} x^{(1)} & | & \cdots & | & x^{(n)} \end{bmatrix} \in \mathbb{R}^{N \times n}$$

- Each row of the matrix represents an **observation,** i.e., is a sample in the feature space
- Each column of the matrix represents a **variable,**
- We assume that all the variables are **centered**, i.e., have zero mean $\forall i, \sum_j x_j^{(i)} = x^{(i)T} e = 0$

Finding a good definition of principal direction

- Let's find some direction w that "points along the data".
- Scalar product can be used to measure the projection of x_j along w
- Summation over all the samples to find global projection contributes



Definition. A *principal direction, is an unitary vector that solves*

$$w \in \arg\max_{\|v\|_2=1} \frac{1}{N} \sum_i (x_i \cdot v)^2$$

Finding other directions

Searching for new directions that are orthogonal to previous one

$$w_1 \in arg \max_{\|w\|=1} \frac{1}{N} \sum_{i=1}^{N} (w \cdot x_i)$$

$$w_2 \in arg \max_{\|w\|=1} \frac{1}{N} \sum_{i=1}^{N} (w \cdot x_i) \quad \text{s.t.} \quad w_2 \perp w_1$$

$$\vdots$$

$$w_n \in arg \max_{\|w\|=1} \frac{1}{N} \sum_{i=1}^{N} (w \cdot x_i) \quad \text{s.t.} \quad w_n \perp \{w_1, ..., w_{n-1}\}$$

Observation (Finding coefficient with respect to an ortho-normal base)

For every vector $\forall v, \quad v = \sum_{i=1}^{n} \alpha_i w_i,$ the coefficients are deduced with a dot product

$$\alpha_i = v \cdot w_i$$

Dimensionality reduction by projecting on first directions

Definition (Momentum)

Given a direction w, we define the momentum as $V(w) = \frac{1}{N}\sum_j (w \cdot x_j)^2$

Let's measure the momentum among the directions of the previous example

Projection on the first 2 components

$$\tilde{x}_i = \alpha_{1i} w_1 + \alpha_{2i} w_2$$

Estimated compression error

$$MSE(x, \tilde{x}) = \frac{1}{N}\sum_i \|x_i - \tilde{x}_i\|^2 = V(w_3)$$

```
Momentum along v1: 0.1670, Percentage: 89.55%
Momentum along v2: 0.0144, Percentage: 7.75%
Momentum along v3: 0.0050, Percentage: 2.71%

MSE considering only v1:         0.0195
MSE considering v1 and v2:       0.0050
MSE considering v1, v2, and v3: 0.0000
```

Open in Colab

Take out from the geometrical introduction

- Given a set of data , we can find the directions that maximize the momentum $w_1, \ldots, w_n$

- Directions should be deduced iteratively, having $V(w_1) > \cdots > V(w_n)$

- **(Conjecture)** The mean square error obtained by projecting the data onto the first k directions satisfies

$$MSE(X, \tilde{X}) \,?\, V(w_{k+1}), \cdots, V(w_n)$$

PCA as an eigen-value problem

Theorem. The principal component is the (unitary) eigen-vector corresponding to the largest eigen-value of the matrix $\frac{1}{N}X^TX$. Furthermore, the largest eigen-value coincides with the first momentum,

$$V(w_1) = \lambda_{max}\left(\frac{1}{N}X^TX\right)$$

Proof.

– Rewrite the maximum problem, in matrix form, and observe that Rayleigh coefficient appears.

– Show that Rayleigh coefficient takes its maximum in the highest eigen value of the matrix.

Part I: Matrix formulation of the momentum

By considering u = Xw, we can deduce that

$$\max_{\|w\|=1} \frac{1}{N} \sum_i (w \cdot x_i)^2 = \max_{\|w\|=1} \frac{1}{N} w^T X^T X w$$

By dividing each term by the norm of w, we get

$$\max_{\|w\|=1} \frac{1}{N} \sum_i (w \cdot x_i) = \max_{w \neq 0} \frac{1}{N} \boxed{\frac{w^T X^T X w}{w^T w}}$$

This value is called Rayleigh coefficient $R(w) = \frac{w^T X^T X w}{w^T w}$

Part II : Maximum of Rayleigh coefficient

Observation

The Rayleigh coefficient takes its maximum in the largest eigenvalue.

$$\max_{v \neq 0} R(v) = R(w_{max}) \quad \text{where} \quad (X^T X) w_{max} = \lambda_{max} w_{max}$$

Proof

- Since the $X^T X$ is symmetric and positive defined, there exists $w_1, \ldots, w_n$ orthogonal eigen-vectors (Spectral theorem). Let's sort them by $\lambda_1 \geq \cdots \geq \lambda_n$

- By definition $R(w_i) = \frac{w_i^T X^T X w_i}{w_i^T w_i} = \lambda_i \leq \max_{v \neq 0} R(v)$

- Note that for every v, $R(v) = \frac{\sum_i \alpha_i^2 \lambda_i}{\sum_i \alpha_i^2} \leq \lambda_1$, where v is written in the ort-base.

Other directions

Theorem (no proof)

Not only the first one, but all the principal directions can be deduced by solving an eigen-value problem. The eigen-values corresponds to the momentum along the principal directions

$$w_1, \ldots, w_n \in \mathbb{R}^n, \quad \text{eigenvectors of} \quad \frac{1}{N} X^T X$$

$$V(w_i) w_i = \frac{1}{N} X^T X w_i, \quad \text{eigenvalues}$$

$$V(w_1) \geq V(w_2) \geq \cdots V(w_n), \quad \text{Decreasing momementum}$$

How to practically compute a PCA

Reminder (Singular Value Decomposition)

Given a matrix X, there exists U,V orthonormal matrices such that

$$X = U\Sigma V^T, \quad U \in \mathbb{R}^{N\times n}, \quad V, \Sigma \in \mathbb{R}^{n\times n}$$

where

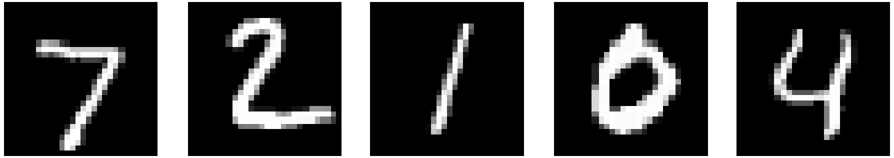
$$U^T U = I, \quad VV^T = V^T V = I, \quad \Sigma = diag\left(\sigma_1^2, \ldots, \sigma_n^2\right)$$

Observation.
The SVD provides the principal directions in the columns of the matrix V

$$\frac{1}{N} X^T X = V \frac{1}{N} \Sigma^2 V^T \quad \Longrightarrow \quad \frac{1}{N} X^T X v_i = \frac{\sigma_i^2}{N} v_i$$

Practically find Principal Components

MNIST, N=10000, n=784



Open in Colab

Centering the data

```python
## Load the data
import torch
from torchvision.datasets import MNIST
import matplotlib.pyplot as plt
import numpy as np

data = MNIST(root='./data', download=True, train=False)
```

```python
## Center the data
X = np.array(data.data.float())/255
mean = X.mean()
print(f"Mean: {mean}")
X = (X - mean)
```

Solving the eigenvalue problem

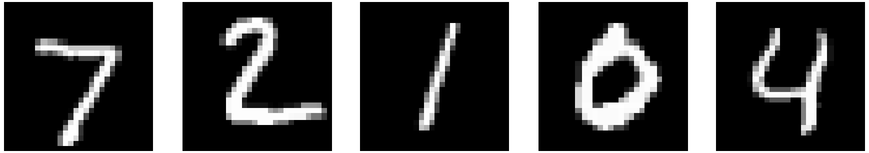
```python
### Finding the principal directions with sklearn (three manners)
# Directly finding the eigen values of the X^TX
M = (X.T @ X)/N
eigenvalues, eigenvectors = np.linalg.eigh(M)
```

Or singular value decomposition

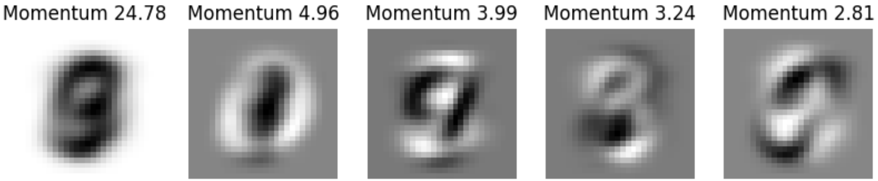
```python
## Second method: Using the singular value decomposition
U, S, V = np.linalg.svd(X, full_matrices=False)
V = V.T
```

Practically find Principal Components

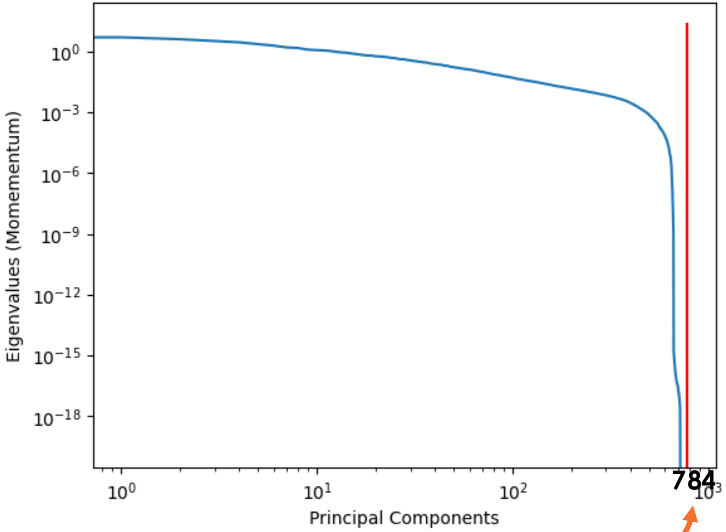
MNIST, N=10000, n=784



First 5 eigen-digits



Momentum computed with eigh



Quiz: Why there is a gap?

Dimensionality Reduction with PCA

Definition (Factors)

Given X the set of centered data, and given $w_1, \dots, w_n$, the principal components we can extract, the *Factors,* i.e the coefficient respect to the new bases.

$$F = XW = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \begin{bmatrix} w_1 | \cdots | w_n \end{bmatrix} = \boxed{\begin{bmatrix} f^{(1)} | \cdots | f^{(n)} \end{bmatrix}}$$

Represents latent (unobserved) variables that can replicate the data through a linear combination.

Quizzz: How many factors can be visualized?

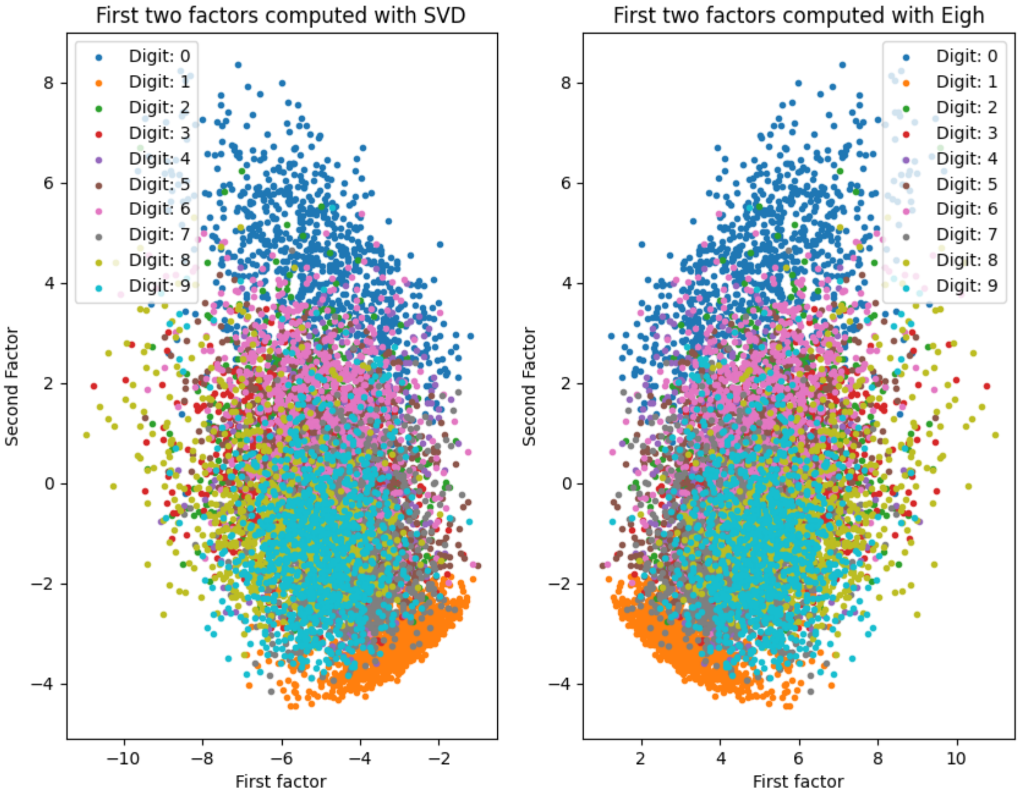
Representing two factors

Comparison of two strategies

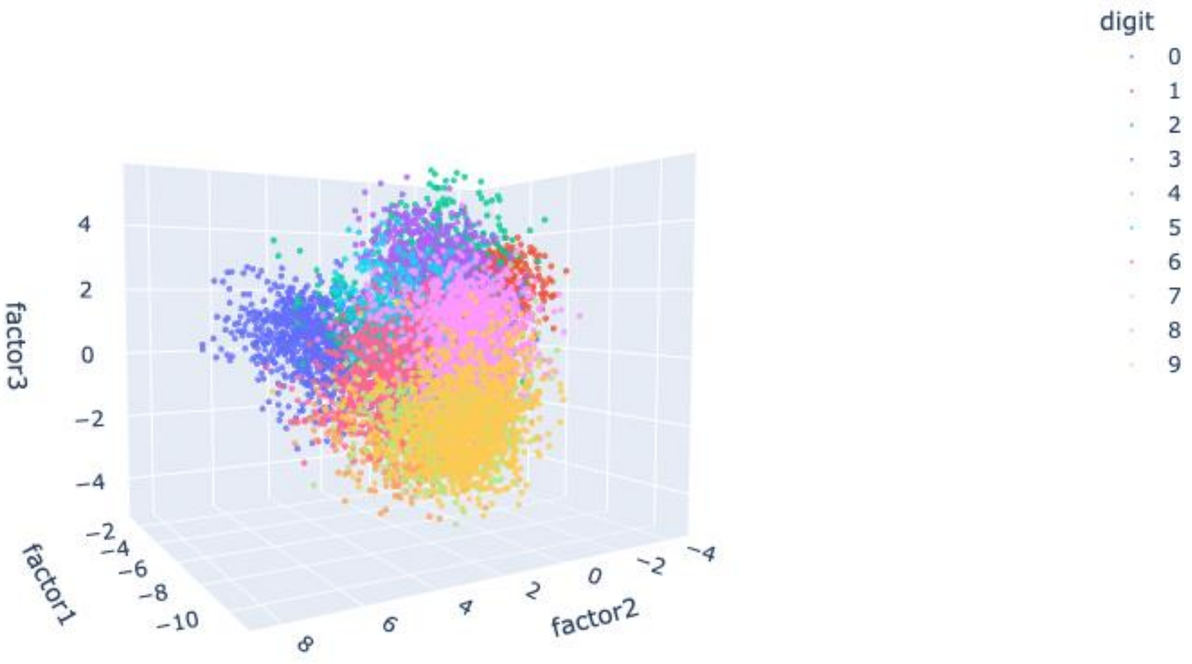
```
### Finding the principal directions with sklearn (three manners)
# Directly finding the eigen values of the X^TX
M = (X.T @ X)/N
eigenvalues, eigenvectors = np.linalg.eigh(M)
```

```
## Second method: Using the singular value decomposition
U, S, V = np.linalg.svd(X, full_matrices=False)
V = V.T
```

Quiz: Why are they different?



Representing “more” factors



Dimensionality Reduction with PCA (Compression Error)

Definition (Projection on the first k-components)

Let k be a number smaller than n, we can project on the first k factors and deducing

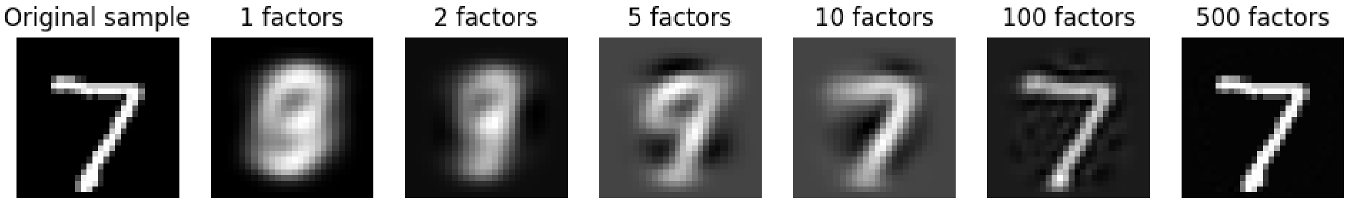
$$X^{(k)} = F_k W_k^T = \begin{bmatrix} f^{(1)} | \cdots | f^{(k)} \end{bmatrix} \begin{bmatrix} w_1^T \\ \vdots \\ w_k^T \end{bmatrix}$$

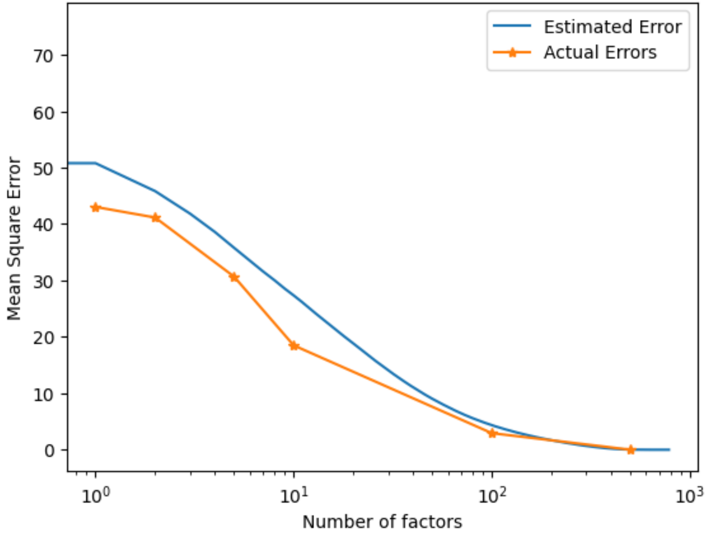
Theorem (Compression Error [Exercise])

Show that , according to the definition of projection provided in advance

1. Projecting on all the components, reproduce the same data, i.e. $X^{(n)} = X$
2. The projection error, depends on the left momenta, i.e. $MSE(X^{(k)}, X) = V(w_{k+1}) + \cdots + V(w_n)$

Compression Error





PCA from a Statistical point of view

Language translation

The PCs are the eigenvectors of the covariance matrix $$\frac{1}{N} X^T X = Cov(X)$$

The momentum among w is the variance of the data projected on w $$V(w) = Var(X\, w)$$

Theorem (Explained Variance)

The compression theorem can be written in terms of the variance as follows

- The $MSE(X^{(k)}, X) = \mathbb{E}\left[(X^{(k)} - X)^2\right] = Var(X\, w_{k+1}) + \cdots Var(X\, w_n)$
- The first k PCs explain the $100 \cdot \left(\frac{\sum_{i=k+1}^{n} Var(X\, w_i)}{\sum_{i=1}^{n} Var(X\, w_i)} \right) \%$ of the whole variance.

Conclusions

What is PCA goot at

- Effective dimensionality reduction while preserving most of the dataset's variance
- Useful for visualizing complex data and as a preprocessing in machine learning
- Principal components help uncover latent patterns between variables

Limitations

- Restricts comparisons to linear relationships between data
- Components are not always easy to interpretSensitive to feature scaling

Stochastic Neighbor Embedding

Introduction

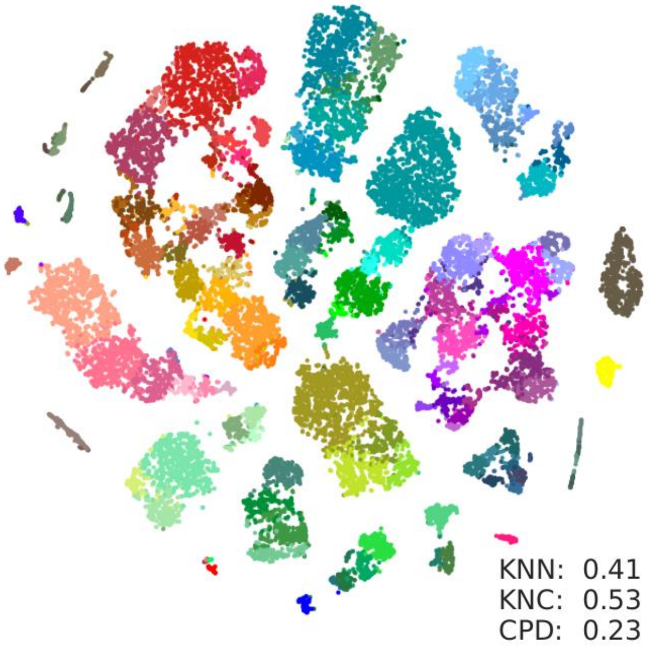
Stochastic Neighbor Embedding (SNE) and its evolution T-SNE

- Goeffrey Hinton, *Stochastic Neighbor Embedding,* 2002.

  Introduced the idea of similarity mapping

- Laurens van der Maaten, *Visualizing Data using t-SNE, 2018.*

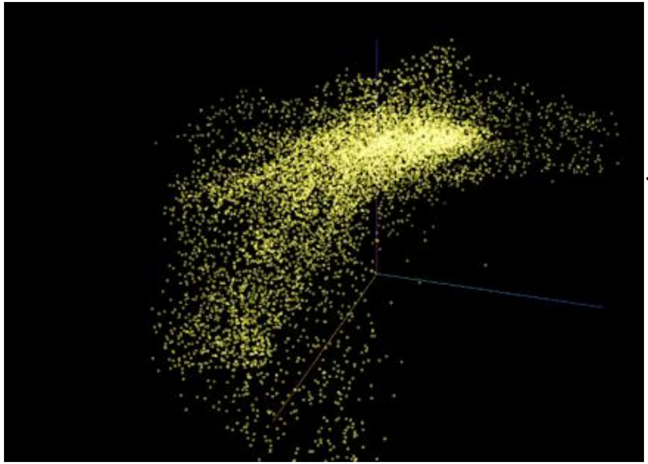
  Refined the method for better data visualization



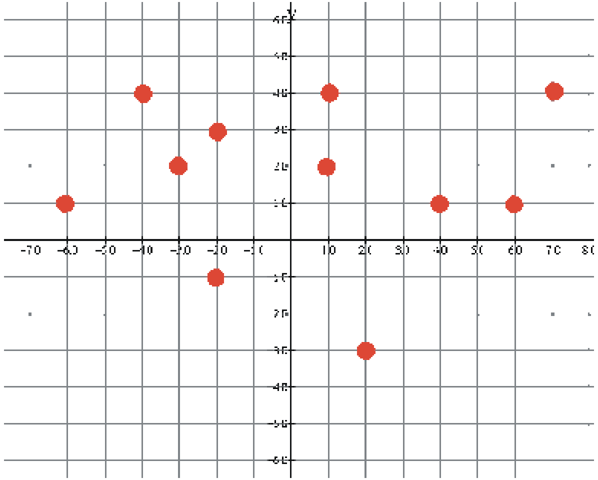
T-SNE for Gene representation (Tasic et al., 2018)

Embedding Map

To visualize high-dimensional data, we need to embed them on a “small” space, i.e., $\mathbb{R}^2$



Φ



$$X = \{x_i \in \mathbb{R}^d \, : \, i = 1, \ldots, N\}$$

$$Y = \{\Phi(x_i) \in \mathbb{R}^2 \, : \, i = 1, \ldots, N\}$$

Asymmetric similarity as a probability distribution

Probability of being neighborhood in the input space

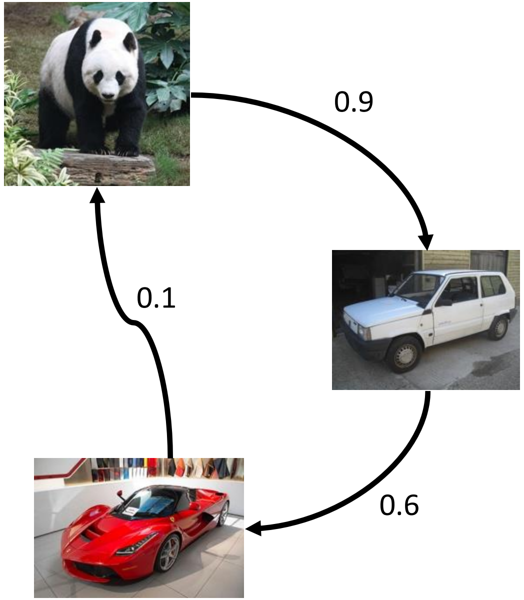
$p_{i \to j}$: *"Probability that samples i and j are neighbors"*

Note. *For each i, p_i is a discrete probability on [N] \ { i }*

Similarity in the representation space of dimension 2

$q_{i \to j}$: *"Probability that representations of i and j are close"*

Note. *For each i, also q_i is a discrete probability on [N] \ { i }*



**p, q are not joint probabilities, furthermore they are not defined in the couples (i,i)-*

How to find a stochastic embedding

Problem space. We have three unknown variables: Φ , $p_{i \to j}$, $q_{i \to j}$

Key Idea. Let's fix the definition of similarity, and let's fine tune the representation map to make the two probabilities distribution closer.

Observation

The embedding map is defined pointwise, i.e., Φ is only defined over the known samples through the definition $\Phi(x_i) = y_i$. Another way to say that the embeddings are the parameters themselves.

Static definition of similarity distribution

Definition (Similarity Distribution via Gaussian Kernel)

Given a value of "dissimilarity" $d_{i,j}$ between input samples x_i and x_j, we consider the following probability distribution for the input

$$p_{i\to j} = \frac{e^{-d_{i,j}^2}}{\sum_{k\neq i} e^{-d_{ik}^2}}$$

While we can consider the following probability distribution in the embedded space

$$q_{i\to j} = \frac{e^{-\|y_i - y_j\|^2}}{\sum_{k\neq i} e^{-\|y_i - y_k\|^2}}$$

Definition (Dissimilarity Map via Scaled Euclidean Distance)

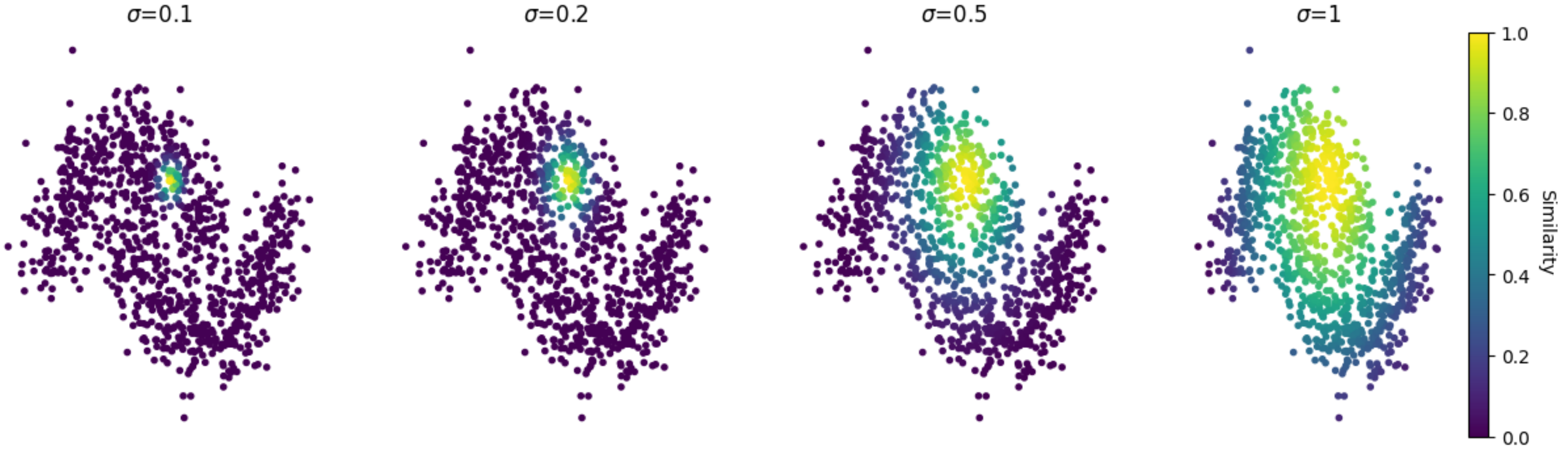
The similarity in the embedded space is assumed symmetric, while in the input space depends on the asymmetric dissimilarity defined as

$$d_{ij}^2 = \frac{\|x_i - x_j\|^2}{2\sigma_i^2}$$

Static definition of similarity distribution

The impact of the coefficient σ_i

Considering larger values, we are including more samples in the neighborhood.



$$p_{i\to j} = \frac{e^{-d_{i,j}^2}}{\sum_{k\neq i} e^{-d_{ik}^2}} \qquad d_{ij}^2 = \frac{\|x_i - x_j\|^2}{2\sigma_i^2}$$

Training the embedding function

Reducing Probabilistic Divergence

The Kullback-Leibler divergence can be used to make the p_i and q_i distributions closer.

$$D_{KL}(p_i||q_i) = \sum_{i \neq j} p_{i \to j} \log \left(\frac{p_{i \to j}}{q_{i \to j}} \right) \qquad \mathcal{C}(y) = \sum_i D_{KL}(p_i||q_i)$$

Observation

The gradient respect to the embedding can be computed explicitly as follows

$$\frac{\partial \mathcal{C}}{\partial y_i} = 2 \sum_{i \neq j} [\overbrace{(p_{i \to j} + p_{j \to i})}^{\text{Depends only on the data}} - \overbrace{(q_{i \to j} + q_{j \to i})}^{\text{Depends on the embeddings}}] (y_i - y_j)$$

Hence stochastic gradient descend can be easily applied to the variables y

Use the perplexity to set the sigma

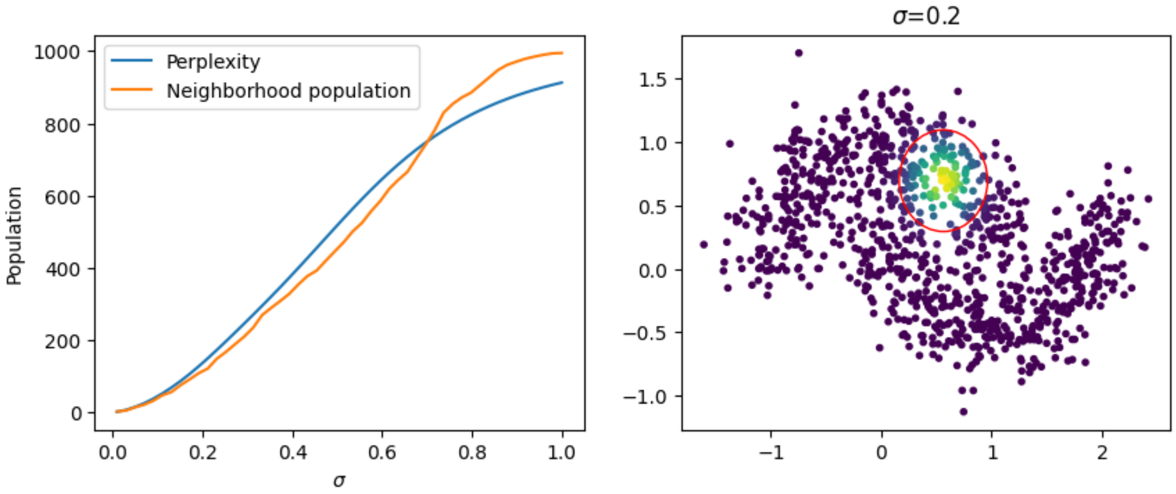
Shannon Entropy

$$\mathbb{H}(p) = -\sum_i p_i \log_2 (p_i)$$

Perplexity

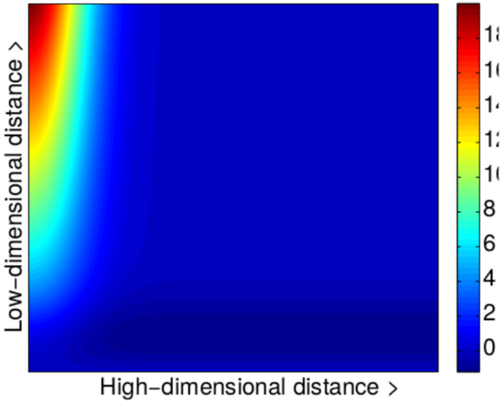
$$Perp(p) = 2^{\mathbb{H}(p)}$$

Observation (No proof). The perplexity estimates the sample in a neighborhood of radius 2 * sigma

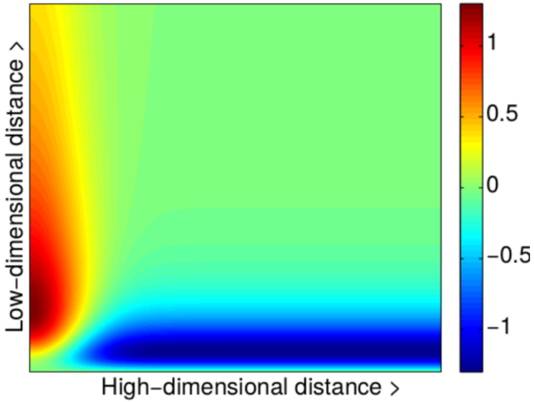


T-SNE is SNE with T-Student Distribution (no details)

T-Student distribution in the embedded space $q_{i \to j} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \|y_i - y_j\|^2\right)^{-1}}$



(a) Gradient of SNE.



(c) Gradient of t-SNE.

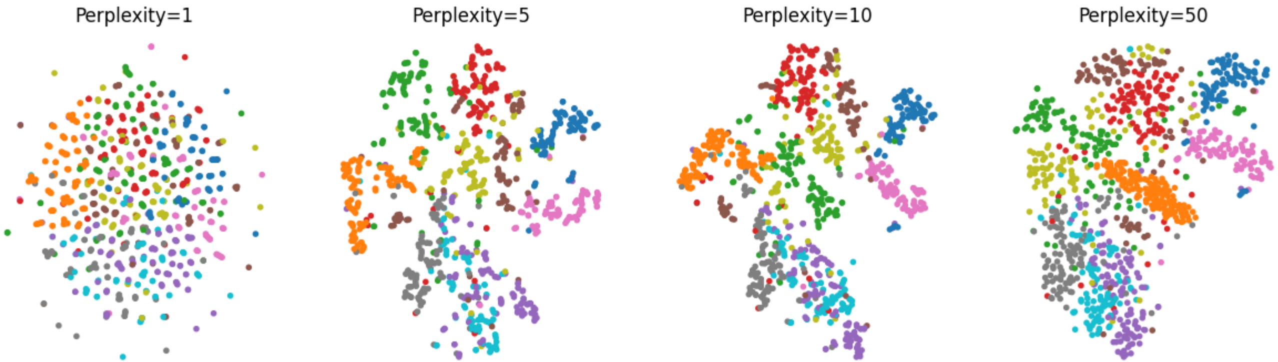
Gradient of t-SNE as a function of the distance in both input end embedded space

T-SNE with scikit-learn library

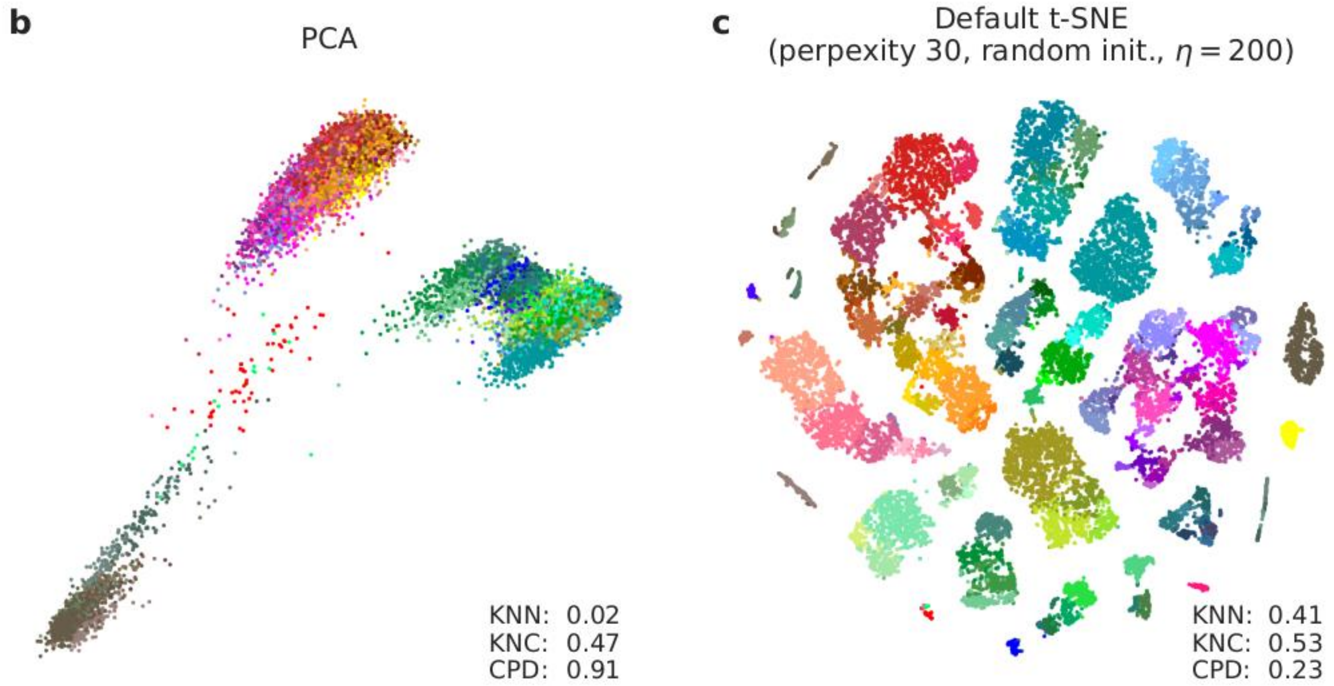
```
perps = [1, 5, 10, 50]
fig, axes = plt.subplots(1,len(perps), figsize=(15,4))

for perp, ax in zip(perps, axes):
  components = TSNE(perplexity=perp).fit_transform(mnist.data.numpy().reshape(-1, 28*28)[:1000,:])
  # show components with colors depending on the digit
  ax.scatter(components[:,0], components[:,1], c=mnist.targets[:1000], marker=".", cmap="tab10")
  ax.axis("off")
  ax.set_title(f"Perplexity={perp}")
plt.plot()
```

Open in Colab

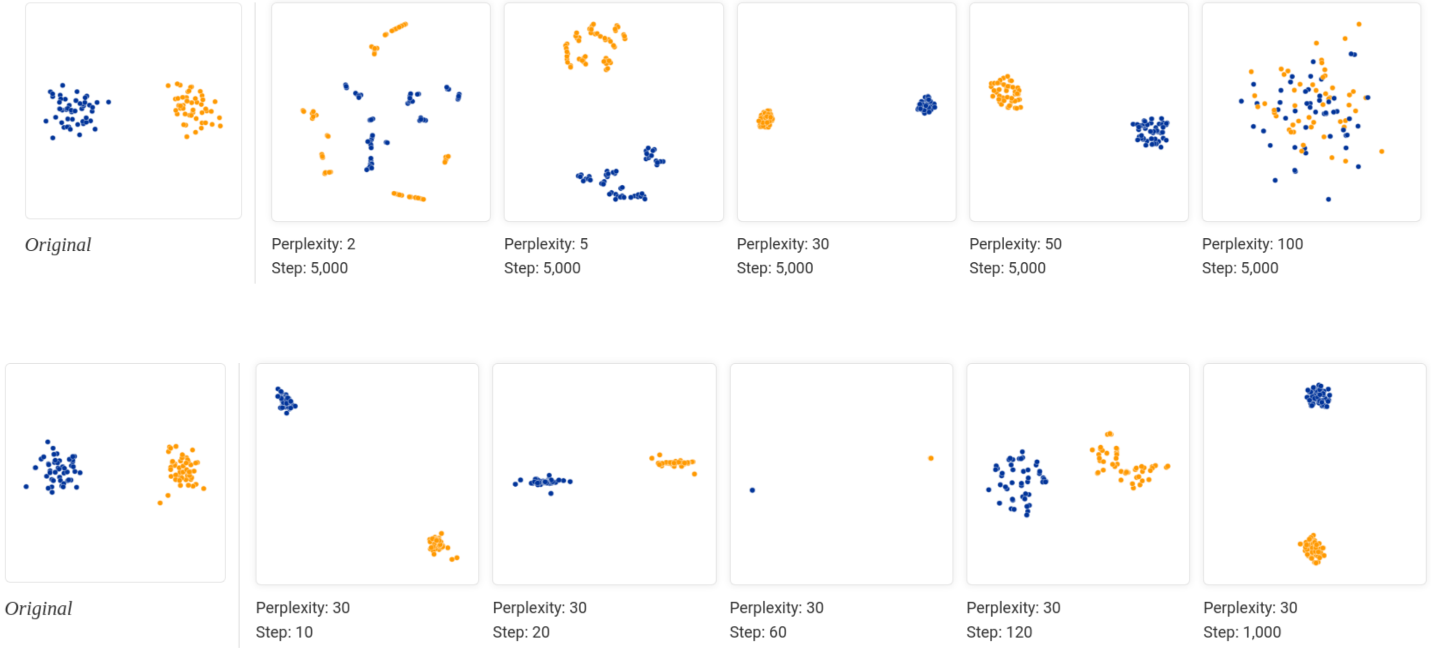


T-SNE on biological data



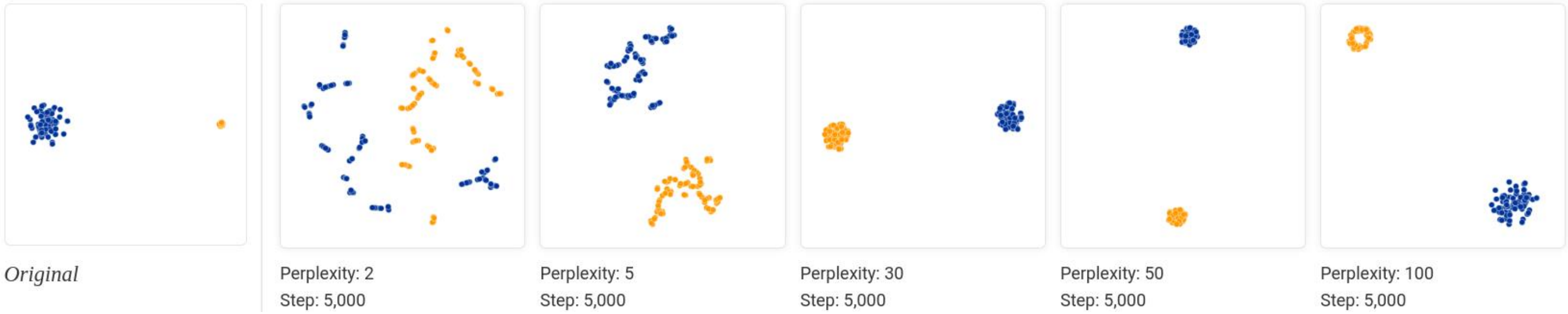
A hand made hierarchical clustering of cortex's cell is visualized through t-SNE. PCA is able to find the main three clusters but not to distinguish the sub-clusters.

Hyper-parameters really matter



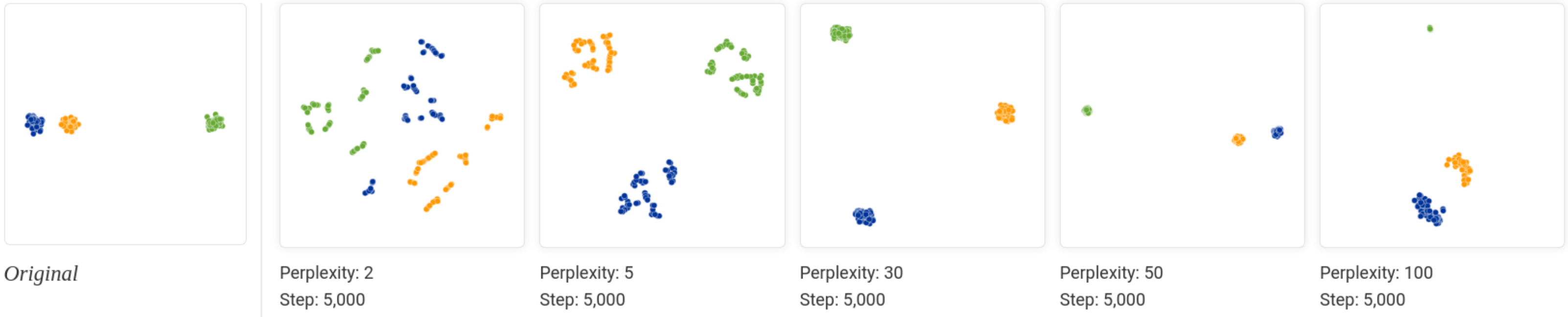
Images taken from https://distill.pub/2016/misread-tsne/

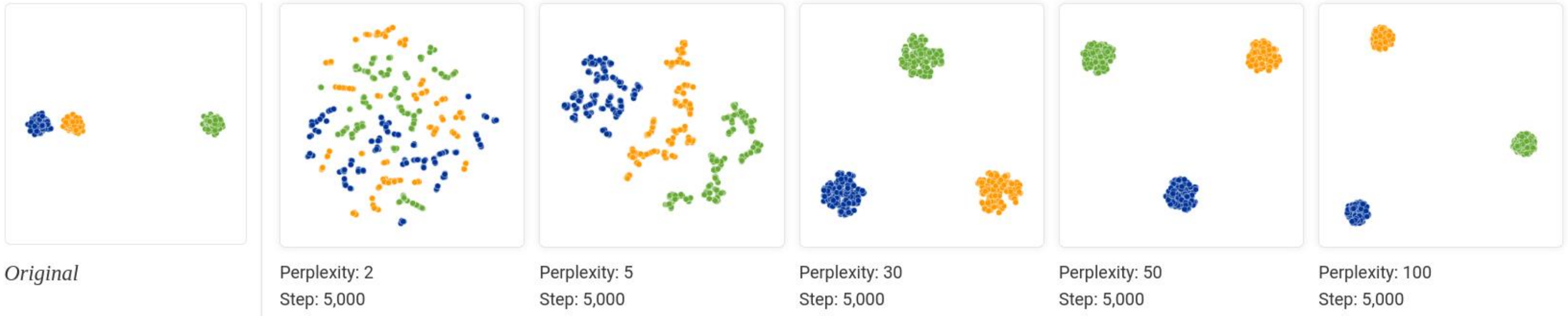
Cluster diameter has no meaning



Images taken from https://distill.pub/2016/misread-tsne/

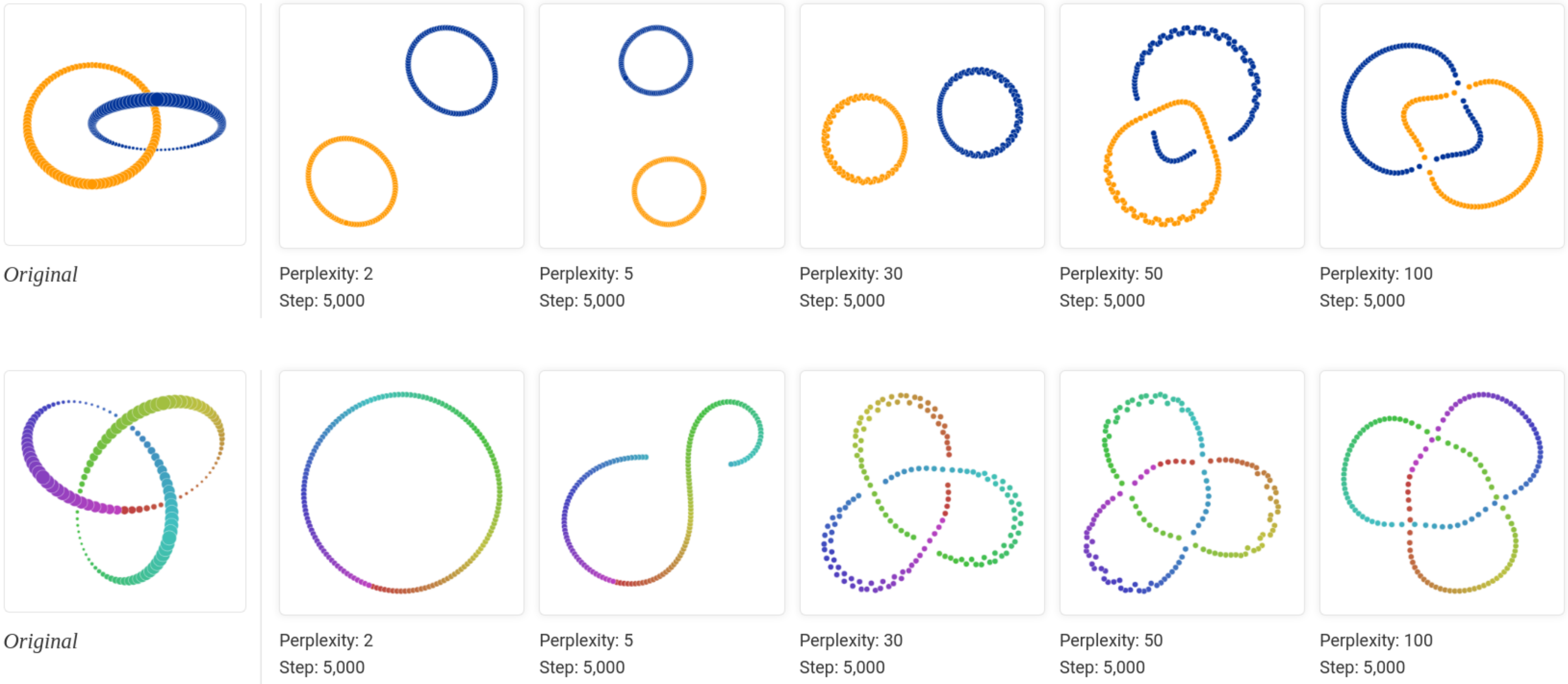
Different Initializations give different inter-cluster distances





Images taken from https://distill.pub/2016/misread-tsne/

Topology is not preserved



Images taken from https://distill.pub/2016/misread-tsne/

Conclusions

What is T-SNE good at

- Visualizing data to confirm intuition based on other clustering techniques
- Aiding in manual clustering when no other techniques can not be used

Do not use T-SNE for

- Evaluate inter-clusters distances
- Evaluate the density of a cluster
- Deducing topological properties on the data