Nonlinear Dimensionality Reduction: t-Stochastic Neighbor Embedding (t-SNE)

Prof. Endri Raco

Polytechnic University of Tirana

Erasmus+ Exchange Professor

Advanced Multivariate Analysis

Polytechnic University of Catalonia (UPC)

October 15, 2025

Welcome to Advanced Multivariate Analysis

Today's Journey

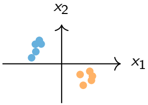
- **2-hour deep dive** into t-SNE
- **Mathematical foundations** to practical insights
- **Three key parts:**
  1. SNE - The original idea
  2. t-SNE - Solving the crowding problem
  3. Hyperparameters & interpretation

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

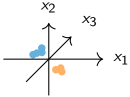
Core t-SNE Formula (We'll derive this today)

The Curse of Dimensionality

Our Intuition Works Here:



2D: Simple



3D: Manageable

But Not Here:

?

100D? 1000D?

Problems:

- Distance concentration
- Volume: $V_n(r) \propto r^n$
- Sparse data

"Geometric intuition fails"

Goals of Dimensionality Reduction

Goal 1: Visualization

$\mathbf{x} \in \mathbb{R}^{784}$
(MNIST)

$\downarrow$ compress

$\mathbf{y} \in \mathbb{R}^{2}$
(2D plot)

Key: "See" hidden structure

Goal 2: Feature Extraction

Noisy features

$\downarrow$ extract

Essential structure

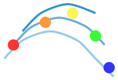
Benefits:

- Noise reduction
- Efficiency
- Better ML

Challenge: Preserve relationships while reducing dimensions

When Linear Methods Falter

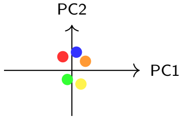
Swiss Roll Dataset



3D Manifold

True Structure:
2D manifold in 3D space

PCA Projection



PCA Result

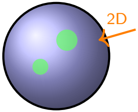
Problem:
Preserves variance,
destroys local structure

Need methods that preserve **local relationships**

The Manifold Hypothesis

"High-dimensional data often lies on or near a much lower-dimensional manifold"

Example: Earth's Surface



2D surface in 3D

Mathematical Form

Data: $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$
where $\mathbf{x}_i \in \mathbb{R}^D$

Assumption:
$\exists$ manifold $\mathcal{M}$ with dim $d \ll D$:

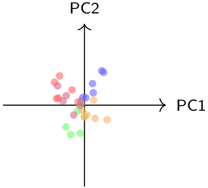
$$\mathbf{x}_i \approx f(\mathbf{z}_i) + \epsilon_i$$

- $\mathbf{z}_i \in \mathbb{R}^d$ (low-dim)
- $f : \mathbb{R}^d \to \mathbb{R}^D$
- ϵ_i (noise)

Goal: Uncover this hidden low-dimensional structure

Preserving Neighborhoods: t-SNE in Action

PCA on MNIST Digits

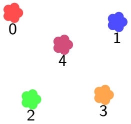


PCA: Mixed Clusters

Problems:

- Classes overlap significantly
- Linear projection limitations
- Poor cluster separation

t-SNE on MNIST Digits



t-SNE: Clear Separation

Advantages:

- Distinct clusters
- Preserves local structure
- Reveals true relationships

"t-SNE focuses on preserving **local neighborhood structure***"*
Similar points in high-D $\rightarrow$ nearby points in low-D

Today's Journey: From Theory to Mastery

Part 1: SNE
(Foundation)
Gaussian Kernels

KL Divergence
Symmetric SNE
Limitations

Part 2: t-SNE
(Innovation)
Crowding Problem

Student's t-dist.
Heavy Tails
Optimization

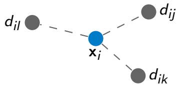
Part 3: Practice
(Application)
Perplexity & LR

Interpretation
Common Pitfalls
t-SNE vs UMAP

From Distances to Probabilities: The Core Idea

Central Insight: Convert distances between points into probabilities
that represent neighborhood relationships

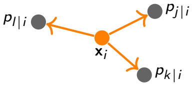
Traditional Approach



Euclidean Distances

Problem: How to weight different distances?

SNE/t-SNE Approach



Probabilities

Solution: Probabilities naturally normalize!

"What is the probability that point i would pick point j as its neighbor?"

Part 1: Stochastic Neighbor Embedding (SNE)

The Foundation: Understanding the original SNE algorithm before moving to t-SNE improvements

What We'll Cover

1. High-dimensional similarities
2. Gaussian kernels
3. Conditional probabilities
4. Perplexity parameter
5. Low-dimensional mapping
6. KL divergence objective
7. Gradient computation

Key References

Original Paper:
Hinton & Roweis (2002)
"Stochastic Neighbor Embedding"
NIPS 2002

Mathematical Framework:
Building on MDS, Isomap, LLE
but with probabilistic approach

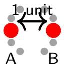
SNE → t-SNE

"SNE converts high-dimensional Euclidean distances into conditional probabilities that represent similarities"

The Challenge of Defining 'Neighborhood'

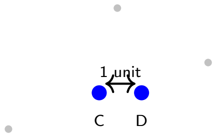
A Fundamental Question: When are two points "neighbors"?

Dense Region



Many neighbors nearby

Sparse Region



Few neighbors nearby

The Problem: Should A-B have the same "similarity" as C-D?

Both are 1 unit apart, but contexts are completely different!

We need similarity to be **relative to local density**

Tool 1: The Gaussian Kernel

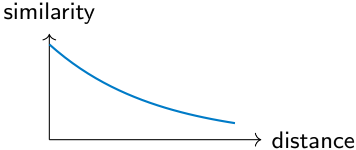
Building Our Similarity Measure Step by Step

Step 1: Start with squared Euclidean distance

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

Step 2: Convert distance to similarity (closer = more similar)

$$\text{sim} = \exp(-d_{ij}^2)$$



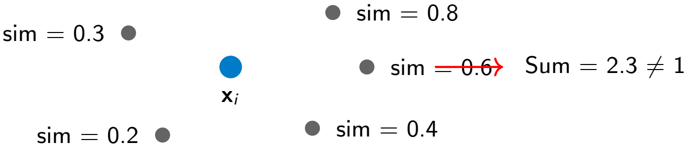
Exponential decay

Step 3: Add bandwidth control with σ_i

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

From Similarity Scores to Probabilities

The Next Problem: Our similarity scores aren't probabilities!



Why is this a problem?

- We want to interpret values as probabilities
- Probabilities must sum to 1
- Need consistent scale across all points

Question: How do we normalize these scores?

Tool 2: Softmax for Normalization

The Solution: Divide by the sum of all similarities!

$$p_{j|i} = \frac{\text{similarity to } j}{\text{sum of all similarities}}$$

The Complete Formula:

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

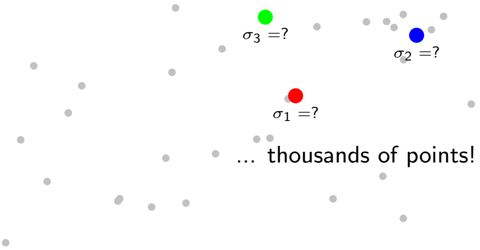
This is the Softmax! - Standard tool in ML
- Converts scores to probabilities
- Preserves relative magnitudes
- Ensures $\sum_j p_{j|i} = 1$

Interpretation: $p_{j|i}$ = probability that point i
would pick point j as its neighbor
if picking based on similarity

Now we have valid probability distributions, but still need to set each σ_i!

An Impossible Task: Setting Each σ_i

The Remaining Challenge: How to choose σ_i for every point?



Problems with manual setting:

- Need different σ_i for each point (adapts to density)
- Dataset might have thousands of points
- No intuitive way to choose values
- Trial and error is impractical

We need a more intuitive, global parameter!

A Better Knob: Perplexity

The Elegant Solution: Specify the "effective number of neighbors"!

Perplexity Definition:

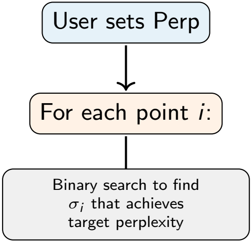
$$\mathrm{Perp}(P_i) = 2^{H(P_i)}$$

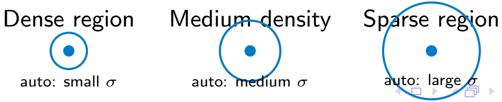
where entropy:

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$$

Intuition:
"I want each point to have about 30 effective neighbors"
$\Rightarrow$ Set Perplexity $= 30$

The Algorithm:





Mapping to the Low-D Space: q(j—i)

The Next Step: Define similarities in the map space

Similar formula, but with a crucial difference:

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

Key Difference: Fixed variance!

High-D Space

Varying σ_i

Low-D Space

Fixed $\sigma = 1/\sqrt{2}$

Why fixed variance?

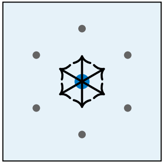
- Encourages uniform distribution in visualization
- Prevents all points from collapsing together
- Scale of map is arbitrary anyway

The Objective: Aligning P and Q

Where We Are:

High-D Space

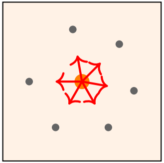
Data points $\mathbf{x}_i$



Distributions P_i
(fixed by data)

Low-D Space

Map points $\mathbf{y}_i$



Distributions Q_i
(we optimize)

The Goal: Arrange points $\mathbf{y}_i$ so that $Q_i \approx P_i$ for all i

Question: How do we measure the total "mismatch"?

Tool 3: Kullback-Leibler Divergence

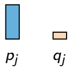
Measuring Distribution Mismatch: KL Divergence

$$\mathrm{KL}(P||Q) = \sum_j p_j \log\left(\frac{p_j}{q_j}\right)$$

"The cost of encoding data from P using a code optimized for Q"

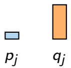
Critical Property: Asymmetry!

Case 1: p_j large, q_j small



$\log(p_j/q_j)$ is large positive
$\Rightarrow$ High penalty!

Case 2: p_j small, q_j large



$\log(p_j/q_j)$ is negative
$\Rightarrow$ Small penalty

Implication: Algorithm focuses on preserving local structure (high penalty for separating neighbors, low penalty for collapsing non-neighbors)

The SNE Cost Function: C

Putting It All Together: The Complete Objective

$$C = \sum_{i=1}^{n} \mathrm{KL}(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

What This Means:

Random initial positions $\mathbf{y}_i$ → Gradient descent → Optimized map

Minimize C by adjusting $\mathbf{y}_i$ positions

The asymmetric penalty means:
- Similar points (high $p_{j|i}$) mapped far apart $\Rightarrow$ Large cost
- Dissimilar points (low $p_{j|i}$) mapped close $\Rightarrow$ Small cost

$\therefore$ SNE preserves local neighborhoods!