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

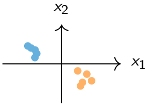
- **2-hour deep dive** into t-SNE
- **Mathematical foundations** to practical insights
- **Three key parts:**
  1. SNE - The original idea
  2. t-SNE - Solving the crowding problem
  3. Hyperparameters & interpretation

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

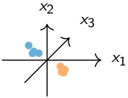
Core t-SNE Formula (We'll derive this today)

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

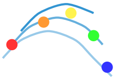
Benefits:

- Noise reduction
- Efficiency
- Better ML

Challenge: Preserve relationships while reducing dimensions

When Linear Methods Falter

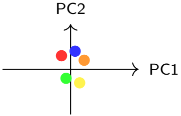
Swiss Roll Dataset



3D Manifold

True Structure:
2D manifold in 3D space

PCA Projection



PCA Result

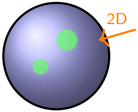
Problem:
Preserves variance,
destroys local structure

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

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

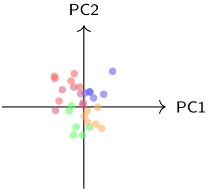
$$\mathbf{x}_i \approx f(\mathbf{z}_i) + \epsilon_i$$

- $\mathbf{z}_i \in \mathbb{R}^d$ (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

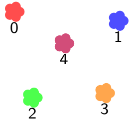


PCA: Mixed Clusters

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 → 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

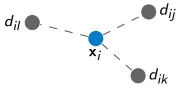
Part 3: Practice
(Application)
Perplexity & LR

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

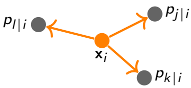
Traditional Approach



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

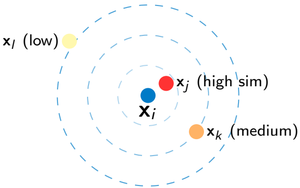
SNE → t-SNE

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

High-Dimensional Similarity: Gaussian Kernels

The Core Question: How similar are two points in high-dimensional space?

Gaussian Similarity



Similarity decreases with distance following Gaussian decay

Mathematical Form

Unnormalized similarity:

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

Key Properties:

- Range: $(0, 1]$
- Max when $i = j$
- Smooth decay
- Point-specific σ_i

Critical: σ_i controls the "neighborhood size" for point i

The Conditional Probability $p_{j|i}$

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

Breaking Down the Components:

Numerator:

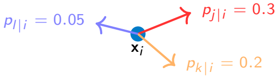
$$\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right)$$

- $\|\mathbf{x}_i - \mathbf{x}_j\|^2$: Squared distance
- σ_i^2: Variance for point i
- Gaussian kernel at $\mathbf{x}_i$

Denominator:

$$\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}\right)$$

- Sum over all other points
- Ensures $\sum_j p_{j|i} = 1$
- Makes it a probability



Tuning Neighborhood Size: Perplexity

The Problem: How to choose σ_i for each point?

Solution: Use *Perplexity* - a user-specified parameter that indirectly sets σ_i through binary search

Definition

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

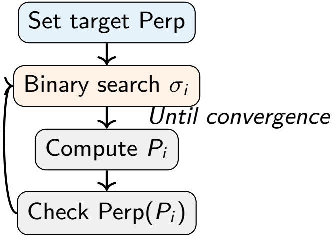
where Shannon entropy:

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

Interpretation:
Perplexity = effective number of neighbors

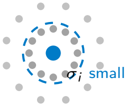
Typical values: 5-50

Binary Search for σ_i



How Perplexity Affects Point Neighborhoods

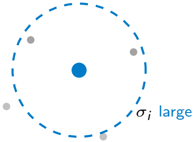
Dense Region



Small σ_i needed

Many nearby points
→ Narrow Gaussian
→ Small σ_i achieves target perplexity

Sparse Region



Large σ_i needed

Few nearby points
→ Wide Gaussian
→ Large σ_i achieves target perplexity

Key Insight: SNE automatically adapts to local data density
Dense regions get small σ_i, sparse regions get large σ_i

σ_i

Low-Dimensional Similarity ($q_{j|i}$): Standard Gaussians

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

Key Differences from High-D:

Fixed Variance

$\sigma^2 = 0.5$ (fixed)

No point-specific σ_i
Often $\sigma^2 = 0.5$
Or absorbed into gradient

Why Fixed?

1. **Equal density:**
Points equally dense in low-D
2. **Scale arbitrary:**
Can rescale embedding
3. **Simplification:**
Fewer parameters

Result: Simpler optimization

High-D Space $\xrightarrow{\text{map}}$ **Low-D Space**

varying σ_i fixed σ

Measuring Mismatch: Kullback-Leibler Divergence

Intuition: The cost of encoding data from distribution P using a code optimized for distribution Q.

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

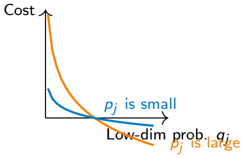
Key Properties

- Always non-negative: $\mathrm{KL}(P||Q) \geq 0$
- Zero if and only if $P = Q$
- **Not a distance metric** because it's **asymmetric**:

$$\mathrm{KL}(P||Q) \neq \mathrm{KL}(Q||P)$$

SNE's Goal: Minimize $\sum_i \mathrm{KL}(P_i||Q_i)$, where P_i are high-dim probabilities and Q_i are low-dim probabilities.

Asymmetric Penalty



Focus: A large penalty is incurred for representing nearby points (p_j large) with distant map points (q_j small).

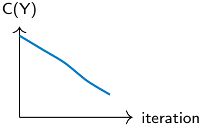
SNE Cost Function: C(Y)

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

Understanding the Cost:

What We Minimize

Sum over all points i:
Each i has its own distribution P_i
We want Q_i to match P_i



Decreasing cost

Asymmetric Penalties

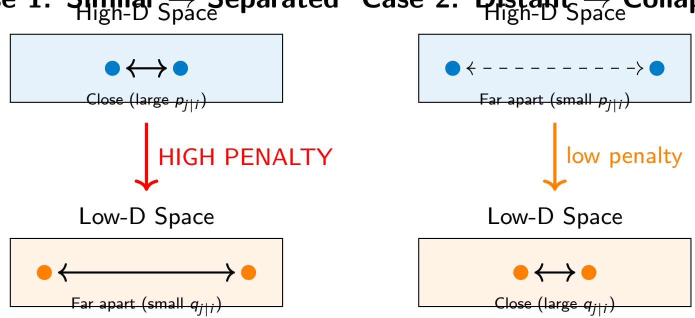
High penalty:
Similar points mapped far
Low penalty:
Distant points mapped close

Result: SNE preserves local neighborhoods at the expense of global

What Does SNE Prioritize?

The Asymmetry of KL Divergence in Action

Case 1: Similar → Separated **Case 2: Distant → Collapsed**



Key Insight: SNE strongly preserves local neighborhoods (Case 1) but allows distant points to collapse (Case 2)

Optimizing SNE: The Gradient $\frac{\partial C}{\partial \mathbf{y}_i}$

Gradient Descent: Update positions to minimize cost

$$\mathbf{y}_i^{(t+1)} = \mathbf{y}_i^{(t)} - \eta \frac{\partial C}{\partial \mathbf{y}_i}$$

The SNE Gradient Formula:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 2 \sum_{j=1}^{n} (p_{ij} - q_{ij} + p_{ji} - q_{ji})(\mathbf{y}_i - \mathbf{y}_j)$$

Key Components

Force terms:
$(p_{ij} - q_{ij})$: Mismatch from i to j
$(p_{ji} - q_{ji})$: Mismatch from j to i

Direction:
$(\mathbf{y}_i - \mathbf{y}_j)$: Vector from j to i

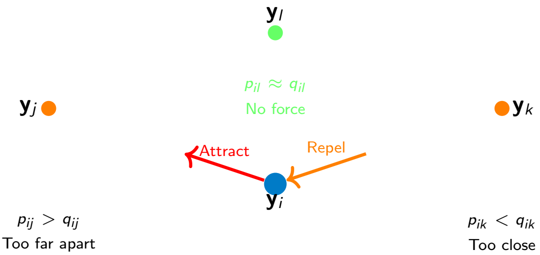
Derivation Steps

1. Start with KL divergence
2. Apply chain rule
3. Differentiate w.r.t. $\mathbf{y}_i$
4. Account for asymmetry
5. Simplify terms

Full derivation in van der Maaten & Hinton (2008), Section 2

Gradient Intuition: Forces at Play

The Gradient as a System of Forces



Attractive Force

When $p_{ij} > q_{ij}$:
Points should be closer
$\Rightarrow$ Pull together

Repulsive Force

When $p_{ij} < q_{ij}$:
Points are too close
$\Rightarrow$ Push apart

Equilibrium: Forces balance when $p_{ij} = q_{ij}$ for all pairs