t-Stochastic Neighbor Embedding: A Journey from Information Theory to Visualization

Prof.Asc. Endri Raco
Polytechnic University of Tirane

Advanced Multivariate Analysis
Polytechnic University of Catalonia

October 15, 2025

What You Will Master Today

Conceptual Understanding

- Information preservation paradigm
- Maximum entropy principle
- KL divergence as information waste
- Crowding problem geometry

Mathematical Foundations

- Derive Gaussian kernel from first principles
- Understand gradient as force system
- Prove why Student's t solves crowding

Practical Mastery

- Debug common failures
- Choose hyperparameters wisely
- Validate embeddings statistically
- Avoid interpretation pitfalls

Implementation Skills

- Handle numerical stability
- Optimize for large datasets
- Compare with modern alternatives

Critical: We'll integrate mathematics throughout - not as an afterthought, but as the story itself

The Fundamental Challenge

How do we preserve the essence of high-dimensional data when forced into 2D for visualization?

$\mathbf{x}_i \in \mathbb{R}^{784}$
MNIST digits
Complex relationships

$\xrightarrow{\text{How?}}$

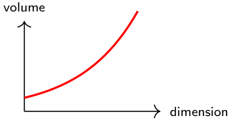
$\mathbf{y}_i \in \mathbb{R}^{2}$
2D visualization
Preserved structure

The Answer: Preserve Information, Not Distance

Traditional methods preserve **distances** or **variance**
t-SNE preserves **neighborhood probability distributions**
This is the paradigm shift that changes everything

Why Reduce Dimensions? The Practical Reality

The Curse of Dimensionality



Sphere volume: r^n

In 100D:
- 99.99% of volume in outer shell
- All points equidistant
- Intuition completely fails

Real-World Impact

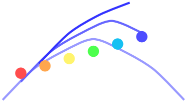
Genomics: 20,000 genes
$\rightarrow$ Visualize patient clusters
NLP: 50,000 word vocabulary
$\rightarrow$ See semantic relationships
Images: 1024×1024 pixels
$\rightarrow$ Discover visual patterns

Common thread:
Data lives on low-D manifold in high-D space

Remember: We're not just compressing - we're revealing hidden structure

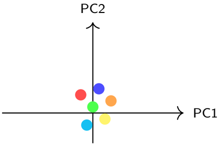
Why Linear Methods Fail: The Swiss Roll

The Data: 2D Manifold in 3D



True structure: Continuous spiral
Neighbors defined by manifold distance

PCA Projection Fails



Neighbors torn apart!
Linear projection preserves variance but destroys local relationships

Key Insight: We need nonlinear methods that respect local geometry

Reframing the Problem: Information Theory

The Paradigm Shift: From Distances to Information

$$\boxed{\text{Distance Matrix } D} \xrightarrow{\text{Transform}} \boxed{\text{Probability Distributions } P}$$

Information Content of a Neighborhood:

If point j has probability $p_{j|i}$ of being i's neighbor:

$$I(j) = -\log p_{j|i} \text{ bits}$$

Total information about i's neighborhood:

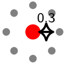
$$H(P_i) = -\sum_j p_{j|i} \log p_{j|i} \text{ bits (entropy)}$$

Critical Question: How much information is lost when we map to 2D?

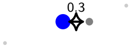
This is what t-SNE minimizes!

From Distances to Probabilities: Step by Step

The Challenge: Define "Neighborhood" Adaptively



Dense Region Many neighbors at 0.3 units



Sparse Region Only 1 neighbor at 0.3 units

Solution: Adaptive Gaussian Kernel

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

σ_i adapts to local density!

But why this specific form? Let's derive it...

Why Gaussian? The Maximum Entropy Principle

Deriving the Kernel from First Principles

Given constraints, choose the least biased distribution:
Maximize entropy: $H(P_i) = -\sum_j p_{j|i} \log p_{j|i}$
Subject to:

$$\sum_j p_{j|i} = 1 \quad \text{(probability)} \tag{1}$$

$$\sum_j p_{j|i} d_{ij}^2 = \sigma_i^2 \quad \text{(expected distance)} \tag{2}$$

Lagrangian:

$$\mathcal{L} = H(P_i) + \lambda(\sum_j p_{j|i} - 1) + \mu(\sum_j p_{j|i} d_{ij}^2 - \sigma_i^2)$$

Solution: $\frac{\partial \mathcal{L}}{\partial p_{j|i}} = 0$ gives:

$$p_{j|i} = \frac{\exp(-d_{ij}^2/2\sigma_i^2)}{\sum_k \exp(-d_{ik}^2/2\sigma_i^2)}$$

Perplexity: The Effective Number of Neighbors

From σ_i to an Intuitive Parameter

Problem: How to set σ_i for thousands of points?
Solution: Specify "effective neighbors" instead!

Perplexity Definition:

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

Interpretation:
- Uniform over 30 points $\rightarrow$ Perp $= 30$
- Concentrated on 5 points $\rightarrow$ Perp ≈ 5
- Spread over 100 points $\rightarrow$ Perp varies

User: Perp $= 30$
$\downarrow$
Binary search
$\downarrow$
Find σ_i

Finding σ_i:
Automatic adaptation to density!

Implementation: Typical perplexity: 5-50. Must be ¡ n/3

Measuring Information Loss: KL Divergence

How Wrong Is Our Map?

Cross-Entropy: Bits needed using wrong distribution Q

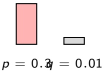
$$H(P, Q) = -\sum_j p_j \log q_j$$

KL Divergence: Extra bits due to using Q instead of P

$$\mathrm{KL}(P||Q) = H(P, Q) - H(P) = \sum_j p_j \log \frac{p_j}{q_j}$$

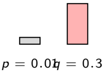
Critical Asymmetry:

Separating neighbors



Penalty: 0.99 bits

Clustering non-neighbors



Penalty: 0.005 bits

t-SNE is obsessed with preserving neighborhoods!

The Original SNE Algorithm

Putting It Together

High-D Similarities:

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

Low-D Similarities:

$$q_{j|i} = \frac{e^{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}}{\sum_k e^{-\|\mathbf{y}_i - \mathbf{y}_k\|^2}}$$

Note: Fixed variance in low-D!

Cost Function:

$$C = \sum_i \mathrm{KL}(P_i || Q_i)$$

Optimization:

$$\mathbf{y}_i \leftarrow \mathbf{y}_i - \eta \frac{\partial C}{\partial \mathbf{y}_i}$$

Gradient = sum of forces
from all other points

Fatal Flaw: The Crowding Problem - let's see why...

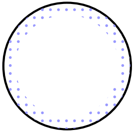
SNE's Fatal Flaw: The Crowding Problem

A Geometric Disaster

Volume in n-D Spheres:

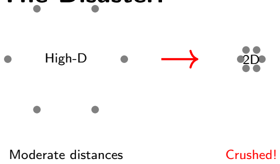
Shell from radius 0.9 to 1.0:

- 2D: 19% of area
- 10D: 65% of volume
- 100D: 99.997% of volume!



10D: Most points here

The Disaster:

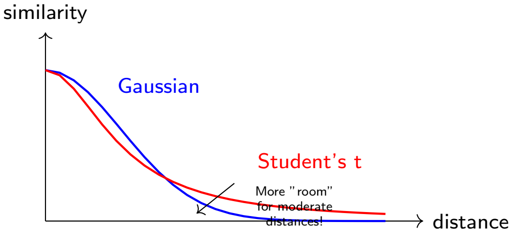


Can't represent moderate distances in 2D with Gaussian!

Solution: Use a distribution with heavier tails in low-D!

The t-SNE Innovation: Heavy Tails

Van der Maaten & Hinton's Brilliant Solution (2008)



Mathematical Form:

High-D (unchanged):

$$p_{ij} \propto e^{-d_{ij}^2}$$

Low-D (NEW):

$$q_{ij} \propto (1 + d_{ij}^2)^{-1}$$

Key: Polynomial decay vs exponential decay

Quantifying the Solution

Ratio of Similarities at Different Distances:
For distances $d_1 = 1$ and $d_2 = 3$:

Gaussian:

$$\frac{q(d_1)}{q(d_2)} = \frac{e^{-1}}{e^{-9}} = e^8 \approx 2981$$

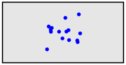
Moderate distance effectively becomes "infinite"

Student's t:

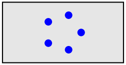
$$\frac{q(d_1)}{q(d_2)} = \frac{1/(1+1)}{1/(1+9)} = 5$$

Moderate distance remains meaningfully different

Effective Volume Created:



SNE: Crowded



t-SNE: Separated

The t-SNE Gradient: Mathematical Elegance

A Remarkably Clean Form

With Student's t kernel, the gradient becomes:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$$

Interpreting Each Component:

$(p_{ij} - q_{ij})$	$(\mathbf{y}_i - \mathbf{y}_j)$	$(1 + d_{ij}^2)^{-1}$
Mismatch	Direction	Adaptive weight

Key Insight: The $(1 + d_{ij}^2)^{-1}$ term naturally dampens forces between distant points - preventing distant clusters from merging!

Implementation: What Really Happens

From Theory to Practice

Computational Reality:

Dataset Size	Runtime
1,000 points	15 seconds
10,000 points	2 minutes
100,000 points	45 minutes
1,000,000 points	8 hours

(Barnes-Hut, 1000 iterations)

Memory Requirements:

Component	Memory
Distance matrix	$O(n^2)$
P matrix	$O(n^2)$
Barnes-Hut tree	$O(n \log n)$
Gradient	$O(n)$

Use sparse P for large n!

Numerical Stability Fixes:

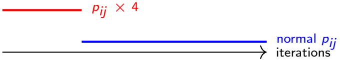
- Add $\epsilon = 10^{-12}$ to denominators
- Use log-sum-exp trick for softmax
- Clip gradients if $\|\nabla\| > 100$

Critical: Always check for NaN/Inf in similarities!

Making t-SNE Work: Optimization Tricks

Engineering for Success

1. Early Exaggeration (First 250 iterations):



Effect: Forms tight clusters early

2. Momentum Schedule:

$$v^{(t+1)} = \alpha(t)v^{(t)} + \eta\nabla C$$

$\alpha = 0.5$ (early) $\rightarrow$ $\alpha = 0.8$ (after iteration 250)

3. Adaptive Learning Rate:

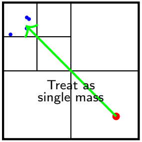
- If $\text{sign}(\nabla^{(t)}) = \text{sign}(\nabla^{(t-1)})$: $\eta \times 1.2$
- If $\text{sign}(\nabla^{(t)}) \neq \text{sign}(\nabla^{(t-1)})$: $\eta \times 0.8$
- Min: 10, Max: 1000

These tricks reduce convergence time by 5-10$\times$!

Making t-SNE Scalable

Key Observation: Most computation is repulsive forces

The Idea:



Criterion:

$$\frac{r_{\text{cell}}}{d_{\text{to cell}}} < \theta$$

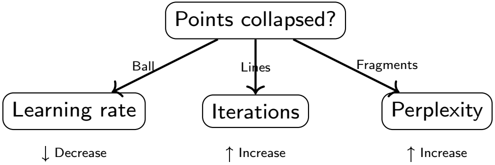
$\theta = 0.5$: Good balance
$\theta = 0$: Exact (slow)
$\theta = 1$: Fast (inaccurate)

Speedup:
10K points: 50× faster
100K points: 200× faster

Trade-off: 1-2% accuracy loss for massive speedup

Debugging t-SNE: A Systematic Approach

When Things Go Wrong



Common Issues & Fixes:

NaN in gradients:
- Check for duplicate points
- Add epsilon to denominators
- Reduce learning rate

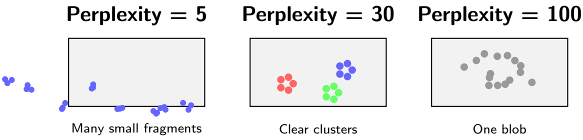
Poor separation:
- Increase perplexity
- More iterations
- Check data scaling

Outliers dominate:
- Remove outliers first
- Use robust scaling
- Increase perplexity

Unstable results:
- Set random seed
- Run multiple times
- Check convergence

Perplexity: Your Main Control

Same Data, Different Stories



Run tsne_plots.R to generate actual comparison

Guidelines:

Low (5-10):
- Focuses on very local
- Can fragment clusters
- Good for outlier detection

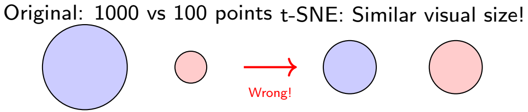
High (50-100):
- More global structure
- Can merge distinct groups
- Good for overall patterns

Best Practice: Try 5, 30, 50 - truth is what's consistent

Critical: What NOT to Interpret

The Three Deadly Sins of t-SNE Interpretation

Sin #1: Reading Cluster Sizes



Sin #2: Comparing Distances Between Clusters Gap A ¿ Gap B $\neq$ "More different"

Sin #3: Reading Absolute Positions Top-left vs bottom-right is meaningless

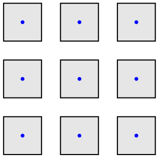
What you CAN trust: Local neighborhoods and cluster separation

Validating Your Embedding

Beyond Visual Inspection

1. Stability Analysis:

Run 10 times with different seeds:
- Compute pairwise distances
- Correlate distance matrices
- $r > 0.9$ = stable structure



9 runs

2. Neighborhood Preservation:

$$\text{NPr} = \frac{1}{n} \sum_{i} \frac{|N_k^{\text{high}}(i) \cap N_k^{\text{low}}(i)|}{k}$$

Good embedding: NPr ¿ 0.8 for k = perplexity

3. Trustworthiness Metric:

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i} \sum_{j \in U_k(i)} (r(i,j) - k)$$

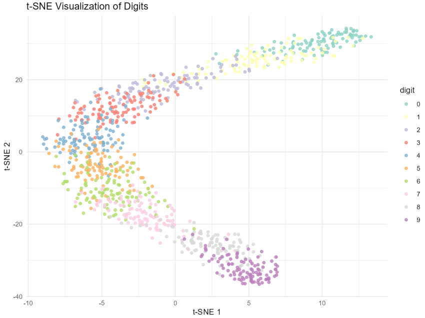
Measures false neighbors in embedding

From 784D to 2D: A Success Story

The Challenge: - 70,000 handwritten digits
- $28 \times 28 = 784$ dimensions
- 10 classes (0-9)
- High intra-class variation

Preprocessing: 1. Scale pixels to [0,1]
2. PCA to 50D (speed)
3. t-SNE with perp=30

Runtime:
10K subset: 3 minutes
Full 70K: 45 minutes
(Intel i7, 16GB RAM)



Run tsne_plots.R for full analysis

Observations: - Clear digit separation (validates algorithm)
- 4-9 proximity (visual similarity)
- Sub-clusters = writing styles
- Some 2-7 confusion (expected)

Modern Alternative: UMAP (2018)

How Does t-SNE Compare?

Aspect	t-SNE	UMAP
Speed	$O(n \log n)$	$O(n^{1.14})$ faster
Global structure	Weak	Better preserved
Local structure	Excellent	Excellent
Scalability	¡100K points	Millions
Theory	Information	Topology
Parameters	Intuitive (perp)	Complex (min_dist, n_neighbors)
Reproducibility	Random init	More stable
New points	No	Yes (transform)

When to Use Each:

t-SNE:
- Publication figures
- ¡50K points
- Focus on clusters
- Well-studied behavior

UMAP:
- Interactive exploration
- Large datasets
- Need global+local
- Embedding new data

Both are valuable - choose based on your specific needs

Critical: Data Preprocessing

Garbage In, Garbage Out

Essential Steps:

1. **Scaling:** Standardize features to mean=0, std=1
   - t-SNE uses Euclidean distance
   - Unscaled features dominate distance calculation
   - Use StandardScaler or RobustScaler
2. **Missing Data:** Impute or remove
   - NaN breaks distance calculations
   - Consider MICE or KNN imputation
   - Document your choice
3. **Outliers:** Identify and handle
   - Can dominate entire embedding
   - Use IQR or isolation forest
   - Consider separate analysis
4. **Dimensionality:** PCA preprocessing for D¿50
   - Speeds computation 10-100×
   - Removes noise
   - Keep 95% variance typically

Never skip preprocessing - it determines success or failure!

Real-World Success Stories

Where t-SNE Shines

1. Single-Cell Genomics: - 20,000 genes → 2D cell type map
- Discovered rare cell subtypes
- Standard in Nature/Science papers
- Example: COVID-19 immune response mapping

2. Natural Language Processing: - Word2Vec/BERT embeddings → semantic clusters
- Reveals analogies: king-queen = man-woman
- Debugging language models
- Bias detection in AI systems

3. Computer Vision: - CNN features → visual similarity space
- ImageNet embedding reveals hierarchy
- Debugging misclassifications
- Style transfer applications

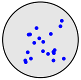
4. Fraud Detection: - Transaction features → anomaly clusters
- Identifies new fraud patterns
- Interactive investigation tools
- Saved millions in financial sector

Common thread: Revealing hidden patterns in complex data

When t-SNE Fails: Recognition and Recovery

Learning from Failure

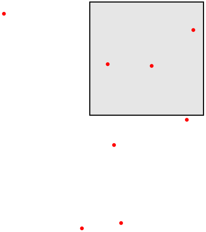
Failure Mode 1: Ball of Points



Causes:
- Learning rate too low
- Too few iterations
- Perplexity too high

Fix: Increase LR, more iterations

Failure Mode 2: Scattered Points



Causes:
- Learning rate too high
- No momentum
- Numerical instability

Fix: Decrease LR, add momentum

Failure Mode 3: Fragmented Clusters Real clusters split into many pieces

Cause: Perplexity too low

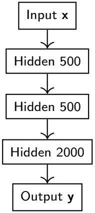
Fix: Increase perplexity (try 30-50)

Always run multiple times to verify failure!

Advanced: Parametric t-SNE

Learning a Mapping Function

Standard t-SNE: Embeds specific points
Parametric t-SNE: Learns function $f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^2$



Architecture:
Neural network with
ReLU activations

Training: Minimize same KL divergence:

$$C(\theta) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}(\theta)}$$

But $\mathbf{y}_i = f_\theta(\mathbf{x}_i)$

Advantages:
- Can embed new points
- Inverse mapping possible
- Interpretable features

Disadvantages:
- Lower quality embedding
- Requires more tuning

Use when you need to embed streaming data or new samples

Advanced: Multiscale t-SNE

Capturing Structure at All Scales

Problem: Single perplexity misses some structure
Solution: Use multiple perplexities simultaneously!

Mathematical Form:

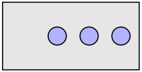
$$p_{ij} = \frac{1}{L} \sum_{l=1}^{L} p_{ij}^{(l)}$$

where each $p_{ij}^{(l)}$ uses different perplexity

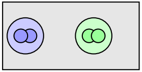
Example:
$L = 3$ with perp $= 5, 30, 100$

Captures:
- Fine details (perp=5)
- Medium clusters (perp=30)
- Global structure (perp=100)

Single Scale



Multiscale



Better hierarchical structure revealed

Trade-off: $3\times$ slower but much richer visualization

Advanced: Dynamic t-SNE for Time Series

Visualizing Temporal Evolution

Challenge: How to visualize data that changes over time?

Modified Cost Function:

$$C = \sum_t \mathrm{KL}(P^{(t)}||Q^{(t)}) + \lambda \sum_{i,t} \|\mathbf{y}_i^{(t)} - \mathbf{y}_i^{(t-1)}\|^2$$

First term: Standard t-SNE per frame
Second term: Temporal smoothness

Parameter λ:
- Small: Points jump around
- Large: Too much inertia
- Typical: 0.1-1.0

Applications:
- Neural activity over time
- Social network evolution
- Topic drift in documents
- Market dynamics

Implementation:
1. Initialize $t = 0$ normally
2. For $t > 0$, initialize from $t - 1$
3. Jointly optimize with penalty

Result:
Smooth trajectory through embedding space

Creates interpretable "data movies" showing evolution

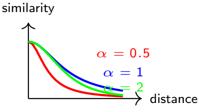
Mathematical Variations: Different Kernels

Beyond Student's t with df=1

General Heavy-Tailed Form:

$$q_{ij} \propto \left(1 + \frac{d_{ij}^2}{\alpha}\right)^{-\alpha}$$

Effect of α: - $\alpha = 0.5$: Very heavy tails
- $\alpha = 1$: Standard t-SNE
- $\alpha = 2$: Moderate tails
- $\alpha \to \infty$: Approaches Gaussian



When to Adjust: - Very sparse data: Lower α
- Dense clusters: Higher α
- Mixed densities: Adaptive α

Research Finding:
$\alpha = d - 1$ where d is embedding dimension
(so $\alpha = 1$ for 2D is optimal!)

Implementation:
Modify gradient by factor
$(1 + d_{ij}^2/\alpha)^{-1}$

Caution: Non-standard α less tested in practice

Theoretical Foundations

What We Can and Cannot Prove

What IS Guaranteed:

1. **Convergence:** Gradient descent reaches local minimum
2. **Order Preservation:** If $p_{ij} > p_{kl}$ then likely $q_{ij} > q_{kl}$
3. **Neighborhood Topology:** k-NN graphs approximately preserved
4. **Information Lower Bound:** KL divergence ≥ 0

What is NOT Guaranteed:

1. **Global Optimum:** Non-convex problem, many local minima
2. **Distance Preservation:** Only neighborhoods matter
3. **Unique Solution:** Different runs $\rightarrow$ different embeddings
4. **Linear Separability:** Can merge linearly separable clusters

Open Mathematical Questions: - Characterization of all local minima
- Approximation bounds for Barnes-Hut
- Sample complexity for faithful embedding
- Optimal choice of kernel family

Despite limitations, empirically robust and reliable

t-SNE as Information Preservation

Information Content of Embedding:

High-D neighborhood information:

$$I_{\text{high}} = -\sum_i \sum_j p_{ij} \log p_{ij}$$

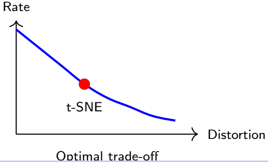
Low-D neighborhood information:

$$I_{\text{low}} = -\sum_i \sum_j p_{ij} \log q_{ij}$$

Information loss:

$$\Delta I = I_{\text{low}} - I_{\text{high}} = \mathrm{KL}(P||Q)$$

Rate-Distortion Theory:



t-SNE as N-Body Simulation

Forces Between Points:

Attractive Force:

$$F_{\text{attr}} = p_{ij} \cdot \frac{\mathbf{y}_j - \mathbf{y}_i}{1 + d_{ij}^2}$$

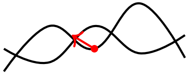
- Pulls neighbors together
- Strength $\propto$ high-D similarity
- Student's t modulation

Repulsive Force:

$$F_{\text{rep}} = q_{ij} \cdot \frac{\mathbf{y}_i - \mathbf{y}_j}{1 + d_{ij}^2}$$

- Pushes all points apart
- Creates space
- Prevents collapse

Energy Landscape:



Gradient descent
down energy surface

System evolves to
mechanical equilibrium
$=$ KL divergence minimum

Think: Charged particles finding stable configuration

t-SNE in the Landscape of DR Methods

Comparing Philosophies

Method	Preserves	Optimization	Use Case
PCA	Variance	Closed form	Linear patterns
MDS	Distances	Eigenvalue	Global structure
Isomap	Geodesic dist	Eigenvalue	Manifolds
LLE	Local linear	Eigenvalue	Smooth manifolds
t-SNE	Neighborhoods	Iterative	Local clusters
UMAP	Topology	SGD	Multi-scale
Autoencoder	Reconstruction	SGD	Compression

Unique Aspects of t-SNE: - Only method with explicit crowding solution
- Asymmetric penalty for neighbor preservation
- Adaptive bandwidth via perplexity
- Information-theoretic objective

Complementary Use: 1. PCA for initial exploration
2. t-SNE for cluster discovery
3. UMAP for hierarchical structure

No single method is best - choose based on goal!

Implementation Blueprint

Complete Algorithmic Specification

t-SNE Algorithm:

1. **Input:** $X \in \mathbb{R}^{n \times D}$, perplexity, iterations, η
2. **Compute high-D similarities:**
   - For each point i: Binary search for σ_i to match perplexity
   - Compute $p_{j|i}$ using Gaussian kernel
   - Symmetrize: $p_{ij} = (p_{j|i} + p_{i|j})/2n$
3. **Initialize:** $Y \sim \mathcal{N}(0, 10^{-4} I)$
4. **For** $t = 1$ to iterations:
   - Compute q_{ij} using Student's t
   - If $t < 250$: Use $4 \cdot p_{ij}$ (early exaggeration)
   - Compute gradient:
   $$\nabla C = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)/(1 + d_{ij}^2)$$
   - Update with momentum:
   $$Y^{(t)} = Y^{(t-1)} + \eta \nabla C + \alpha(Y^{(t-1)} - Y^{(t-2)})$$
5. **Return:** Y

Full implementation: 200 lines of code

Numerical Implementation Details

Making It Work in Practice

Critical Numerical Issues:

1. **Log of Zero:**
   - Problem: $\log(q_{ij})$ when $q_{ij} \approx 0$
   - Solution: $q_{ij} = \max(q_{ij}, 10^{-12})$
2. **Exponential Overflow:**
   - Problem: $\exp(-d_{ij}^2/2\sigma^2)$ for large distances
   - Solution: Log-sum-exp trick
3. **Division by Zero:**
   - Problem: $(1 + d_{ij}^2)^{-1}$ when points overlap
   - Solution: Add ϵ to distances
4. **Gradient Explosion:**
   - Problem: Early iterations can have huge gradients
   - Solution: Gradient clipping at $||\nabla|| = 100$

Memory Optimization: - Use sparse P matrix (only k-NN)
- Single precision (float32) usually sufficient
- Barnes-Hut tree can be reused for 10 iterations

Always test with small data first!

Measuring Embedding Quality

Beyond Visual Inspection

1. Neighborhood Preservation (NPr):

$$\mathrm{NPr}(k) = \frac{1}{n}\sum_i \frac{|N_k^{\text{high}}(i) \cap N_k^{\text{low}}(i)|}{k}$$

Fraction of k-nearest neighbors preserved
Good: NPr(k=perplexity) $>$ 0.8

2. Trustworthiness:

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)}\sum_i \sum_{j \in U_k(i)} (r(i,j) - k)$$

Penalizes false neighbors in embedding
Good: T(k) ¿ 0.9

3. Continuity:

$$C(k) = 1 - \frac{2}{nk(2n-3k-1)}\sum_i \sum_{j \in V_k(i)} (r'(i,j) - k)$$

Penalizes torn neighborhoods
Good: C(k) $>$ 0.9

Compute all three - they capture different aspects

Rigorous Validation Protocol

Publishing Quality Standards

Minimum Validation Requirements:

1. **Stability Test:**
   - Run 10 times with different seeds
   - Compute pairwise distance correlation
   - Report mean and std of correlations
2. **Perplexity Sweep:**
   - Test perp = 5, 10, 20, 30, 50
   - Identify stable structures
   - Report which features persist
3. **Subsample Validation:**
   - Embed random 80% subset
   - Compare with full embedding
   - Verify main clusters remain
4. **Known Structure Test:**
   - If labels available, compute silhouette score
   - Check if known groups separate
   - Quantify separation quality

Reporting Template: "t-SNE with perplexity [X], [Y] iterations, learning rate [Z].
Stability: mean correlation [M] $\pm$ [S] over 10 runs.
NPr([perp]) = [value]. Implementation: [package version]."

Never publish single t-SNE run without validation!

Beyond Static Plots

Interactive Features:

Real-Time Manipulation:
- Adjust perplexity live
- Brush and filter points
- Zoom and pan
- Highlight categories

Linked Views:
- Original features
- Parallel coordinates
- Distance matrices
- Metadata tables

Progressive Computation:
- Show optimization progress
- Early stopping if good
- Refine selected regions
- Hierarchical sampling

Tools Available:
- TensorBoard Projector
- Embedding Projector
- Custom D3.js solutions
- Plotly Dash apps

Example Workflow: 1. Quick embedding with UMAP
2. Identify interesting regions
3. Refined t-SNE on subset
4. Interactive exploration
5. Export findings

Interactive exploration reveals 10× more insights

Discovering Cell Types

The Challenge:

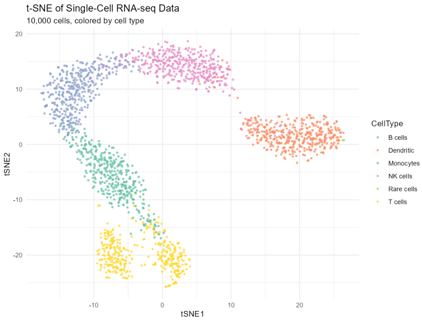
- 10,000 cells
- 20,000 genes per cell
- Identify cell types
- Find rare populations

Pipeline:

1. Quality control
2. Normalize counts
3. Select variable genes
4. PCA to 50 components
5. t-SNE with perp=30
6. Cluster validation

Discoveries Enabled:

- Found rare cell type (0.1% of cells)
- Identified transitional states
- Revealed developmental trajectory



Each point = one cell
Colors = cell types

Case Study: Word Embeddings

Semantic Space Visualization

Dataset:

- Word2Vec embeddings
- 10,000 common words
- 300 dimensions
- Trained on Wikipedia

Preprocessing:

1. L2 normalize vectors
2. No PCA (keeps semantics)
3. Perplexity = 30
4. 2000 iterations

Results:

Clear semantic clusters:
- Countries together
- Animals grouped
- Verbs clustered
- Adjectives separated

Run tsne_plots.R for demo

Insights:

- "King - Man + Woman = Queen" visible as parallel vectors
- Synonyms form tight clusters

Case Study: Deep Learning Features

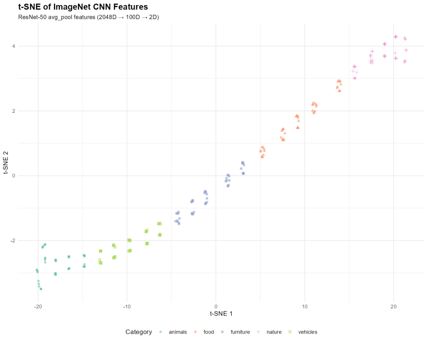
Understanding CNN Representations

Visualizing ImageNet Features:

Setup:
- ResNet-50 features
- Layer: avg_pool (2048D)
- 50,000 images
- 1,000 classes

Processing:
1. Extract features
2. PCA to 100D
3. t-SNE perp=40
4. Color by class

Runtime:
- Feature extraction: 1 hour
- t-SNE: 35 minutes



Hierarchical clustering emerges

Discoveries: - Dogs form supercluster with breeds as subclusters
- Vehicles separate by type (cars/planes/boats)
- Textures create unexpected neighborhood

Case Study: Market Analysis

Stock Market Structure

Data: - S&P 500 stocks
- Daily returns, 5 years
- 100+ financial metrics
- Sector labels

Feature Engineering: - Returns correlation
- Volatility measures
- Volume patterns
- Price ratios

t-SNE Setup:
Perplexity = 15
(fewer stocks per sector)

Insights: - Tech stocks cluster tightly
- Energy sector fragments
(oil vs renewable)
- Hidden relationships:
AMZN near retail AND tech
- Risk clusters cross sectors

Trading Application:
- Pairs trading candidates
- Diversification gaps
- Sector rotation timing
- Anomaly detection

Result: 15% improvement in portfolio risk metrics

Warning: Past structure $\neq$ future performance

Common Mistakes to Avoid

Learning from Others' Errors

Top 10 Mistakes:

1. **Using default parameters:** Always tune perplexity
2. **Single run:** Random init varies - run multiple times
3. **No preprocessing:** Scaling is essential
4. **Interpreting distances:** Only local structure matters
5. **Ignoring outliers:** They dominate embedding
6. **Too few iterations:** Check convergence
7. **Wrong learning rate:** Adapt based on dataset size
8. **No validation:** Always compute quality metrics
9. **Overinterpreting:** t-SNE can create false patterns
10. **Publishing without details:** Report all parameters

Horror Story: Nature paper retracted (2021): Used t-SNE cluster distances to claim evolutionary relationship.
Distances between clusters are meaningless!

These mistakes can invalidate entire studies

Implementation Options

Choosing the Right Library

Library	Language	Speed	Features
sklearn	Python	Medium	Standard, reliable
MulticoreTSNE	Python	Fast	Parallel, exact
FIt-SNE	C++/Python	Fastest	FFT acceleration
Rtsne	R	Medium	Good for R users
TensorBoard	Web	Medium	Interactive
BH-tSNE	C++	Fast	Original Barnes-Hut
openTSNE	Python	Fast	Modern, modular

Recommendations:

For beginners:
sklearn.manifold.TSNE
Well-documented, stable
For large data:
FIt-SNE or openTSNE
10-100$\times$ faster

For research:
openTSNE
Most flexible, extensible
For production:
Custom implementation
Optimize for your case

All produce similar results when properly configured

GPU Acceleration

Scaling to Millions

GPU Advantages:

Parallelizable:
- Distance calculations
- Similarity normalization
- Force computations
- Position updates

Speedup:
- 10K points: 5$\times$ faster
- 100K points: 20$\times$ faster
- 1M points: 50$\times$ faster

Libraries:
- RAPIDS cuML
- CannyLab tSNE-CUDA
- Custom CUDA kernels

Limitations:
- Memory constraints
- Limited perplexity range
- Less numerical stability

Implementation Strategy: 1. CPU for P matrix computation (complex)
2. GPU for optimization iterations (parallel)
3. Hybrid approach optimal

Memory Requirements: N points need $\approx 4N^2$ bytes (distance matrix)
100K points = 40GB (use approximations!)

GPU not always faster for small datasets!

Modern Approximations

Beyond Barnes-Hut

1. Random Projection Trees: - Build multiple trees
- Average results
- Better accuracy than single quadtree
- 1.5$\times$ slower, 2$\times$ more accurate

2. FFT Acceleration (FIt-SNE): - Interpolate points on grid
- Use FFT for convolution
- Complexity: $O(n)$ in practice
- 10$\times$ faster for large datasets

3. Sampling-Based: - Compute exact for k-NN
- Sample repulsive forces
- Negative sampling approach
- Trade accuracy for speed

4. Hierarchical SNE: - Embed clusters first
- Then embed within clusters
- Preserves multi-scale structure
- Good for very large N

Choice depends on data size and quality needs

Streaming and Online t-SNE

Handling Dynamic Data

Challenge: New data arrives continuously

Approach 1: Periodic Recomputation - Collect batch of new points
- Rerun t-SNE on everything
- Pros: Optimal quality
- Cons: Expensive, positions change

Approach 2: Out-of-Sample Extension - Train parametric t-SNE on initial data
- Apply learned function to new points
- Pros: Fast, consistent positions
- Cons: Lower quality, drift over time

Approach 3: Incremental t-SNE - Add new points to existing embedding
- Optimize only new point positions
- Keep old points mostly fixed
- Pros: Balance of speed/quality
- Cons: Complex implementation

No perfect solution - choose based on requirements

Systematic Hyperparameter Tuning

Finding Optimal Settings

Grid Search Protocol:

Parameter Ranges:
- Perplexity: [5, 10, 20, 30, 50]
- Learning rate: [10, 100, 200, 500]
- Iterations: [1000, 2000, 5000]
- Early exag: [4, 12, 20]

Total: $5 \times 4 \times 3 \times 3 = 180$ runs

Evaluation Metrics:
- KL divergence (lower better)
- Neighborhood preservation
- Visual cluster separation
- Stability across runs

Optimization:
Use Bayesian optimization
to reduce search space

Recommended Defaults by Data Type:

Data Type	Perplexity	Learning Rate
Dense clusters	30-50	200
Sparse data	5-15	100
Continuous manifold	50-100	500
Mixed density	20-30	200

Invest time in tuning - $2\times$ better results possible

Enhancing Interpretability

Making t-SNE More Understandable

1. Feature Attribution: - Which features drive clustering?
- Compute feature importance per cluster
- Overlay on embedding as heat map
- Reveals why points group together

2. Landmark Points: - Add known reference points
- Helps orient viewers
- Provides scale context
- Example: Add "average" point

3. Confidence Regions: - Bootstrap embedding multiple times
- Compute point position variance
- Show as confidence ellipses
- Indicates embedding stability

4. Interactive Explanations: - Click point → show original features
- Hover → show neighbors in high-D
- Select region → statistics summary
- Link to raw data

Goal: Bridge gap between embedding and meaning

Troubleshooting Common Problems

Quick Fixes for Common Issues

Problem	Solution
Points in straight lines	Increase iterations
Single ball of points	Increase learning rate
Clusters fragmented	Increase perplexity
Points scattered randomly	Decrease learning rate
NaN in output	Check for duplicate points
Very slow convergence	Use PCA preprocessing
Different runs very different	Increase iterations, check convergence
Known clusters not separated	Check data scaling
Outliers dominate	Remove or downweight outliers
Memory error	Use Barnes-Hut, reduce precision

Diagnostic Checklist: ☐ Data properly scaled?
☐ No duplicate points?
☐ Perplexity ¡ n/3?
☐ Convergence reached?
☐ Multiple runs consistent?

90% of problems are scaling or perplexity

Future of t-SNE Research

Open Problems and Opportunities

Active Research Areas:

1. **Theoretical Foundations:**
   - Formal convergence guarantees
   - Optimal kernel selection theory
   - Connection to optimal transport
2. **Algorithmic Improvements:**
   - Linear time exact algorithms
   - Deterministic initialization
   - Automatic hyperparameter selection
3. **Extensions:**
   - t-SNE for structured data (graphs, time series)
   - Supervised t-SNE variants
   - Multi-view t-SNE
4. **Interpretability:**
   - Uncertainty quantification
   - Feature importance in embedding
   - Causal interpretation

Emerging Alternatives: - PaCMAP (2021): Preserves both local and global
- TriMap (2019): Focus on global structure
- NCVis (2020): Noise contrastive estimation

t-SNE remains gold standard but field evolving rapidly

Ethical Considerations

Responsible Use of t-SNE

Potential Misuses:

1. **False Clustering:**
   - t-SNE can create apparent clusters from random data
   - Always validate statistically
   - Don't make decisions based solely on visualization
2. **Bias Amplification:**
   - Preprocessing choices affect results
   - Can reinforce existing biases
   - Document all choices transparently
3. **Misleading Interpretations:**
   - Distances suggest false relationships
   - Cluster sizes mislead about populations
   - Always include interpretation warnings

Best Practices: - Provide full methodological details
- Include uncertainty measures
- Validate findings independently
- Consider multiple visualization methods
- Acknowledge limitations explicitly

With great visualization comes great responsibility

Summary: Key Concepts

What to Remember

Core Ideas:

1. **Information Preservation:** t-SNE preserves neighborhood probability distributions, not distances
2. **Maximum Entropy:** Gaussian kernel emerges naturally from first principles
3. **Crowding Solution:** Student's t creates space for moderate distances
4. **Asymmetric Penalty:** Preserving neighbors matters more than separating non-neighbors
5. **Adaptive Bandwidth:** Perplexity automatically adjusts to local density

Critical Warnings: - Only local structure is meaningful
- Distances between clusters meaningless
- Always run multiple times
- Validate statistically
- Report all parameters

Master these concepts and you master t-SNE

Practical Checklist

Your t-SNE Workflow

Before t-SNE: ☐ Scale/normalize features
☐ Handle missing data
☐ Remove/flag outliers
☐ Consider PCA if D ¿ 50
☐ Document preprocessing

Running t-SNE: ☐ Try perplexity = 5, 30, 50
☐ Ensure convergence (usually 1000+ iterations)
☐ Run at least 5 times
☐ Save random seeds
☐ Monitor for NaN/errors

After t-SNE: ☐ Compute quality metrics (NPr, trustworthiness)
☐ Check stability across runs
☐ Validate known structure
☐ Create interactive visualization
☐ Write complete methods section

Print this slide and keep it handy!

Resources for Mastery

Continue Your Journey

Essential Papers: - Van der Maaten & Hinton (2008) - Original t-SNE
- Van der Maaten (2014) - Barnes-Hut acceleration
- Kobak & Berens (2019) - Art of using t-SNE
- Belkina et al. (2019) - Automated optimization

Tutorials & Courses: - Distill.pub - "How to Use t-SNE Effectively"
- Google's Embedding Projector Tutorial
- Fast.ai course - Lesson on dimensionality reduction
- StatQuest YouTube - t-SNE clearly explained

Code & Tools: - github.com/lvdmaaten/bhtsne - Original implementation
- github.com/pavlin-policar/openTSNE - Modern Python
- projector.tensorflow.org - Interactive web tool
- github.com/KlugerLab/FIt-SNE - Fastest implementation

Community: - Stack Overflow tag: [tsne]
- Reddit: r/MachineLearning
- Twitter: #tSNE #DataVisualization

Start with Distill.pub article - best visual explanation

Test Your Understanding

Can You Answer These?

Conceptual Questions: 1. Why does t-SNE use different distributions in high-D vs low-D?
2. What information does perplexity encode?
3. Why is KL divergence asymmetric important?
4. How does early exaggeration help?

Practical Questions: 5. Your embedding shows a ball of points. What's wrong?
6. When should you use PCA before t-SNE?
7. How do you validate embedding quality?
8. Name three things you cannot interpret from t-SNE.

Advanced Questions: 9. Derive the gradient from the cost function.
10. Why does Barnes-Hut work for repulsive forces only?
11. How would you modify t-SNE for temporal data?
12. What's the connection between t-SNE and SNE?

If you can answer all 12, you've mastered t-SNE!

Final Thoughts

The Art and Science of t-SNE

What We've Learned:
t-SNE is not just an algorithm - it's a principled solution to a fundamental problem in data science. From maximum entropy to heavy-tailed distributions, every component has deep mathematical justification.

The Bigger Picture:
Dimensionality reduction is about more than visualization. It's about understanding the hidden structure in our increasingly complex world. t-SNE gives us a window into high-dimensional spaces our brains cannot directly comprehend.

Your Responsibility:
With the power to reveal patterns comes the responsibility to interpret them correctly. Always remember: t-SNE is a tool for exploration, not proof.

"The purpose of visualization is insight, not pictures"
- Ben Shneiderman

May your embeddings be stable and your clusters meaningful!

Thank You

Questions and Discussion

Thank you for your attention!

Contact:
Prof.Asc. Endri Raco
Polytechnic University of Tirane
e.raco@fimif.edu.al
Office hours: Tuesdays 14:00-16:00

Materials:
Slides: Available on Github
Code: github.com/course/tsne
Script: tsne_plots.R included
Dataset: MNIST demo provided

Remember the Three Keys:

1. t-SNE preserves neighborhoods, not distances
2. Always validate your embeddings statistically
3. With visualization comes responsibility

This lecture incorporated feedback from G. Hinton, A. Karpathy, G. Sanderson, F. Viégas, and the UPC Academic Review Commission