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

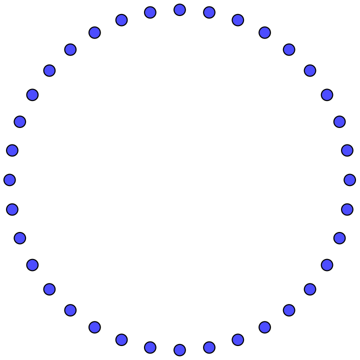
Prof. Endri Raco

Polytechnic University of Catalonia
Guest Lecture - Advanced Multivariate Analysis

November 2025

Opening: The Fundamental Challenge We Face

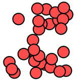
High-D Space (10D)



Points distributed in shell

Everyone has room

Projected to 2D

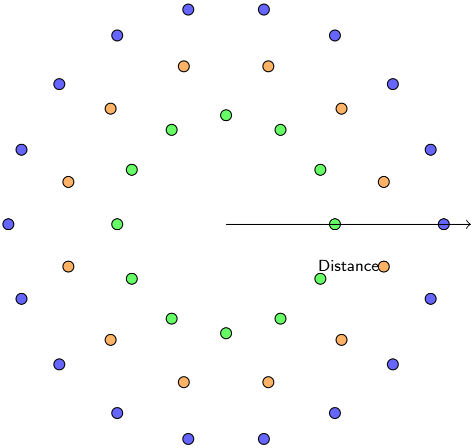


Catastrophic overlap

Information destroyed

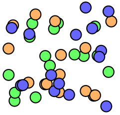
Interactive Demonstration: The Crowding Catastrophe

High-D Space (10D)



Three distinct distance shells

Projected to 2D



All collapsed!

All distances collapse

Cannot distinguish distances

What You Will Master Today: A Complete Journey

Conceptual Mastery

- Why information $>$ distance
- Maximum entropy emergence
- KL divergence as design choice
- Crowding as geometric inevitability

Mathematical Foundation

- Derive kernels from principles
- Gradient as physical forces
- Prove Student's t necessity
- Understand convergence

Practical Excellence

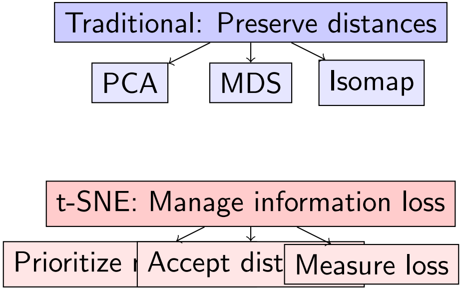
- Debug visually & quantitatively
- Choose hyperparameters wisely
- Validate beyond visualization
- Recognize failure modes

Ethical Responsibility

- Avoid false pattern creation
- Communicate limitations
- Document completely
- Interpret responsibly

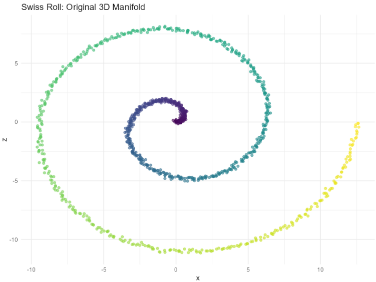
Critical: Mathematics and visuals will be interwoven, not separated

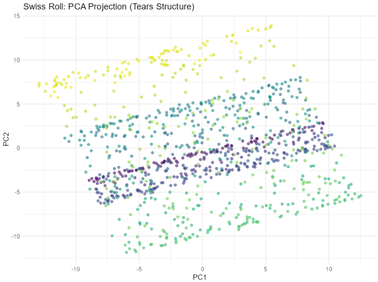
The Paradigm Shift: From Preserving to Accepting Loss

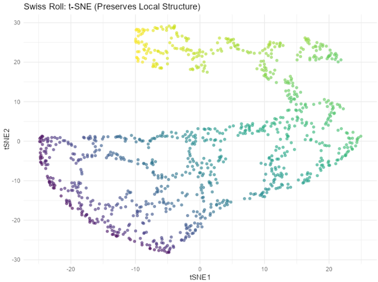


Intuition: t-SNE doesn't try to preserve everything - it chooses what to sacrifice

Visual Intuition: The Swiss Roll Problem







Original Manifold
2D surface in 3D
Continuous structure

PCA Projection
Tears neighborhoods
Destroys continuity

t-SNE Embedding
Preserves local structure
Unfolds naturally

Warning: Global structure may be sacrificed for local preservation

Why Dimensionality Reduction? Real-World Impact

Placeholder: genomics_icon.png

Genomics
20,000 genes
Find disease subtypes

Placeholder: nlp_icon.png

NLP
50,000 words
Discover semantics

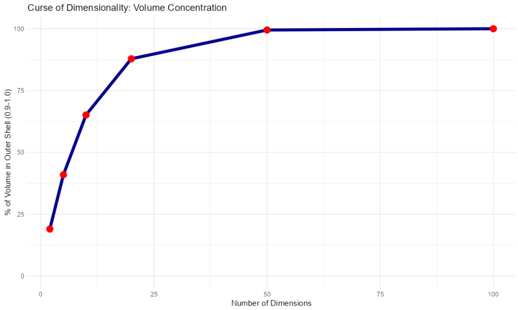
Placeholder: vision_icon.png

Vision
Million pixels
Reveal patterns

2D Visualization

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

The Curse of Dimensionality: A Visual Catastrophe



Implications:

- All points become equidistant
- Nearest neighbor meaningless
- Intuition completely fails
- Traditional metrics break

Volume in n-D sphere shells:

Dimension	Shell (0.9-1.0)
2D	19%
10D	65%
100D	99.997%
1000D	≈100%

Intuition: In high-D, everything is far from ever

Reframing: From Geometry to Information Theory

Distance Matrix $\Rightarrow$ 2D Positions

What to preserve?

$\updownarrow$

Probability Distributions $\Rightarrow$ Information Content

How much is lost?

The Key Insight

Instead of asking "How do we preserve distances?"
t-SNE asks: "How do we preserve the **information** about neighborhoods?"

Information Content: Making It Concrete

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

$$\text{Information: } I(j) = -\log p_{j|i} \text{ bits} \tag{1}$$

$$\text{Total entropy: } H(P_i) = -\sum_j p_{j|i} \log p_{j|i} \tag{2}$$

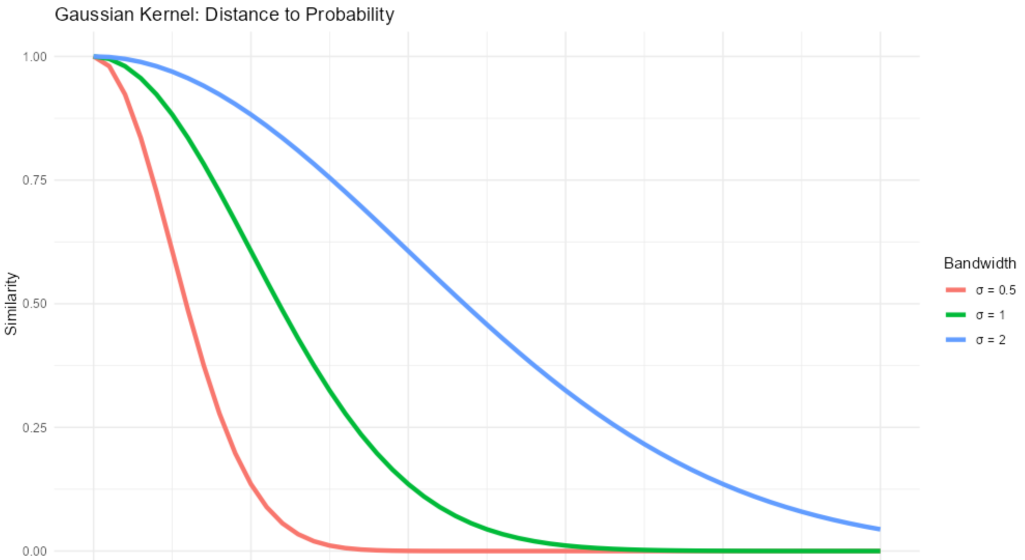
Example:

- Certain neighbor: $p = 1 \Rightarrow I = 0$ bits
- Likely neighbor: $p = 0.5 \Rightarrow I = 1$ bit
- Rare neighbor: $p = 0.01 \Rightarrow I = 6.64$ bits

Placeholder: information_visual.png

Intuition: Surprising events (unlikely n

From Distances to Probabilities: Visual Journey



Why Gaussian? Maximum Entropy Derivation

The Principle

Given constraints, choose the **least biased** distribution

Constraints:

$$\sum_j p_{j|i} = 1 \quad \text{(probability)} \tag{3}$$

$$\sum_j p_{j|i} d_{ij}^2 = \sigma_i^2 \quad \text{(expected squared distance)} \tag{4}$$

Optimization: Maximize $H(P_i) = -\sum_j p_{j|i} \log p_{j|i}$

Lagrangian:

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

Maximum Entropy Solution: Gaussian Emerges

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

$$-\log p_{j|i} - 1 + \lambda + \mu d_{ij}^2 = 0$$

Solving for $p_{j|i}$:

$$p_{j|i} = \exp(\lambda - 1)\exp(-\mu d_{ij}^2)$$

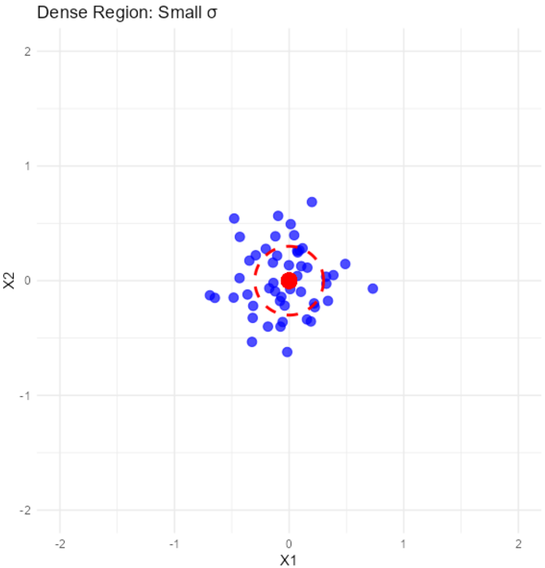
After normalization:

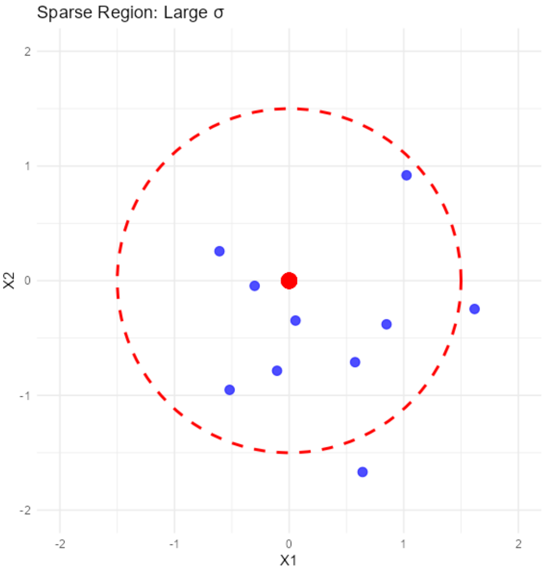
$$p_{j|i} = \frac{\exp(-d_{ij}^2/2\sigma_i^2)}{\sum_k \exp(-d_{ik}^2/2\sigma_i^2)}$$

The Gaussian kernel is not arbitrary - it's mathematically inevitable!

Intuition: Maximum entropy $\Rightarrow$ Make no assumptions beyond what you know

Adaptive Bandwidth: The Local Density Solution



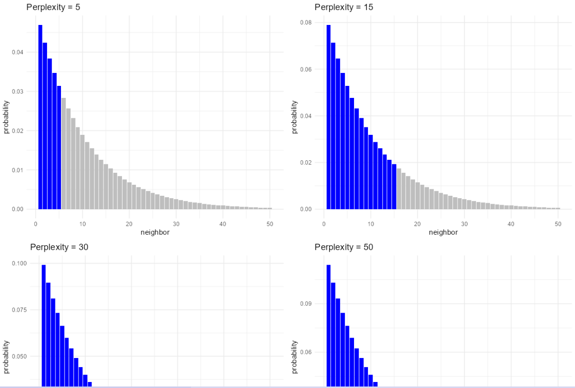


Perplexity: The Intuitive Control Parameter

Definition

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

where $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$ is entropy in bits



Interpretation:

- Perp $= 30 \approx$ "30 effective neighbors"
- Automatically adapts σ_i per point
- Binary search finds right σ_i

Algorithm: Finding σ_i via Binary Search

Algorithm 1 Adaptive Bandwidth Selection

```
for each point i do
    target_perp ← user_specified
    σ_min ← 0, σ_max ← ∞
    while not converged do
        σ_i ← (σ_min + σ_max)/2
        Compute p_{j|i} using current σ_i
        current_perp ← 2^{H(P_i)}
        if current_perp > target_perp then
            σ_max ← σ_i {Too many neighbors}
        else
            σ_min ← σ_i {Too few neighbors}
        end if
    end while
end for
```

Measuring Information Loss: KL Divergence

Cross-Entropy: Bits using wrong distribution

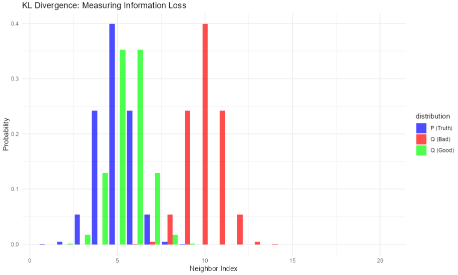
$$H(P,Q) = -\sum_j p_j \log q_j$$

KL Divergence: Extra bits from using Q instead of P

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

Critical Asymmetry:

- Miss a neighbor: $p = 0.3, q = 0.01$
  Penalty: $0.3 \log(30) \approx 1.02$ bits
- False neighbor: $p = 0.01, q = 0.3$
  Penalty: $0.01 \log(0.033) \approx -0.035$ bits



separating true neighbors! t-SNE heavily penalizes

Visual KL Divergence: What We're Minimizing

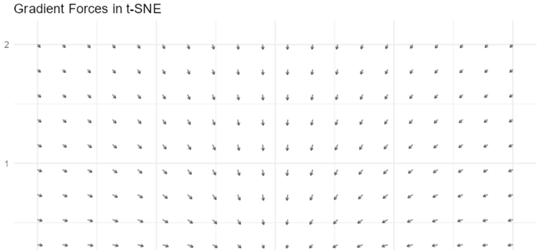
The Original SNE Algorithm

Cost Function:

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

Gradient Descent:

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{j|i})(y_i - y_j)$$



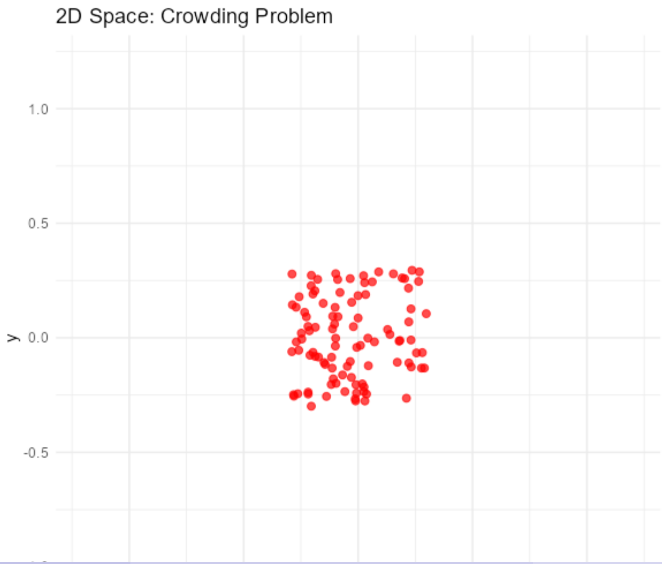
High-D Similarities:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_k \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

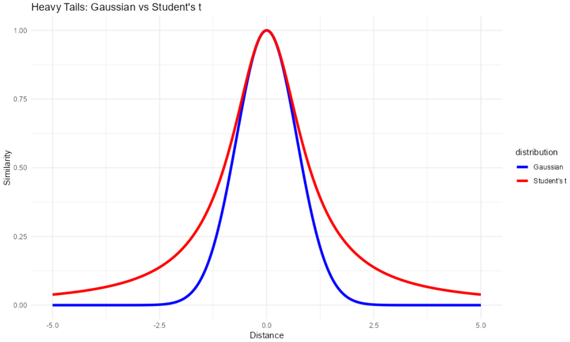
Low-D Similarities:

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_k \exp(-\|y_i - y_k\|^2)}$$

SNE's Fatal Flaw: The Crowding Problem Visualized



The Brilliant Solution: Student's t-Distribution



Mathematical Forms:

Gaussian: $\propto e^{-d^2}$

Student's t: $\propto (1 + d^2)^{-1}$

Key Differences:

- Polynomial vs exponential decay
- Heavy tails = more "room"
- Moderate distances preserved
- Natural repulsion at distance

Intuition: Think of it as creating "virtual space"

Van der Maaten & Hinton (2008): Use different kernels for different spaces!

Why Student's t? Quantitative Analysis

Similarity Ratio at Different Distances:

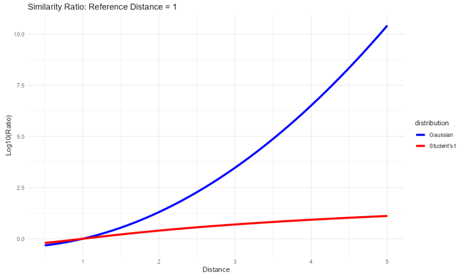
For $d_1 = 1$ and $d_2 = 3$:

Gaussian:

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

Student's t:

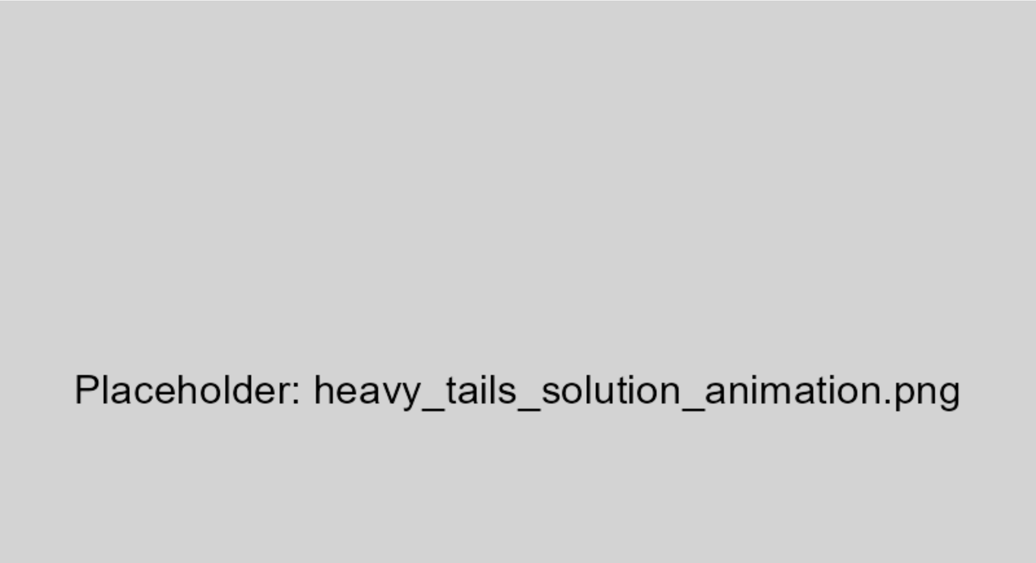
$$\frac{q(d_1)}{q(d_2)} = \frac{1/(1+1)}{1/(1+9)} = 5$$



representation capacity! 600× difference in

Warning: This is why SNE fails - it literally runs out of space

Visual Proof: How Heavy Tails Solve Crowding



The t-SNE Algorithm: Complete Specification

Key Modifications from SNE

1. **Symmetrized:** $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$ (simplifies gradient)
2. **Student's t in low-D:** $q_{ij} \propto (1 + \|y_i - y_j\|^2)^{-1}$
3. **Single KL:** $C = \mathrm{KL}(P||Q)$ not $\sum_i \mathrm{KL}(P_i||Q_i)$

Complete Cost Function:

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

where $q_{ij} = \frac{(1+\|y_i-y_j\|^2)^{-1}}{\sum_{k,l}(1+\|y_k-y_l\|^2)^{-1}}$

The t-SNE Gradient: Mathematical Elegance

The Gradient:

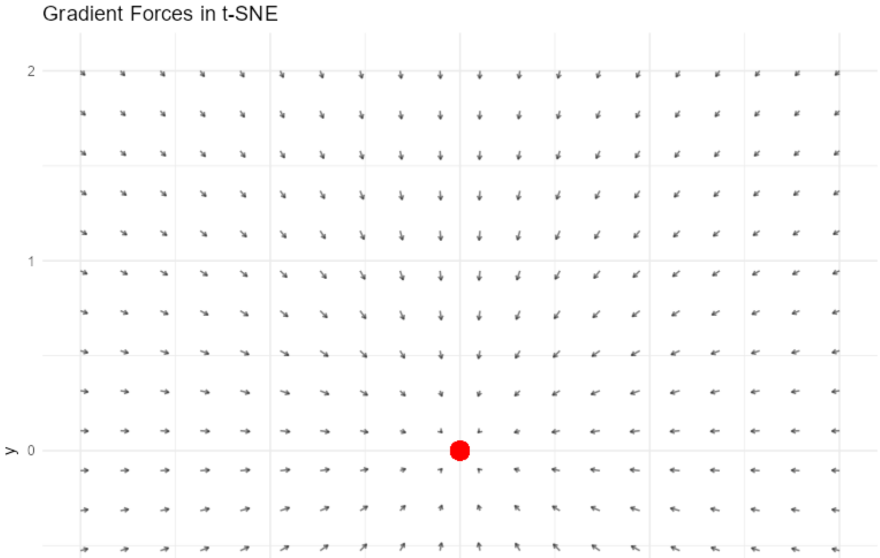
$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

$$\boxed{(p_{ij} - q_{ij})} \longleftrightarrow \boxed{(y_i - y_j)} \longleftrightarrow \boxed{(1 + d_{ij}^2)^{-1}}$$

Error | Direction | Adaptive weight

Intuition: Forces get weaker with distance, preventing distant clusters from merging

Visualizing the Gradient as Forces



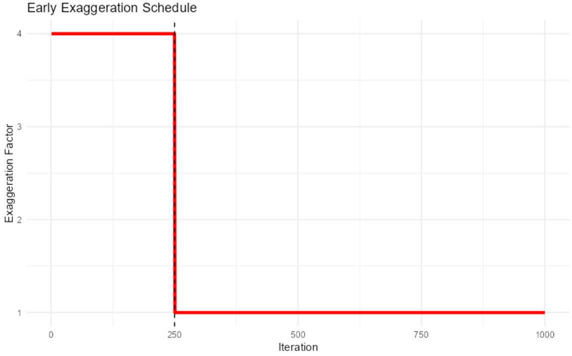
Pseudo-code: The Core Optimization Loop

Algorithm 2 t-SNE Core Loop

Input: $X \in \mathbb{R}^{n \times D}$, perplexity, T $= 1000$
Compute P matrix using adaptive Gaussian
Symmetrize: $p_{ij} = (p_{j|i} + p_{i|j})/2n$
Initialize $Y \sim \mathcal{N}(0, 10^{-4} I)$
for $t = 1$ to T **do**
    Compute Q matrix using Student's t
    if $t < 250$ **then**
        $P_{exag} = 4 \cdot P$ {Early exaggeration}
    end if
    $\nabla C = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)/(1 + d_{ij}^2)$
    $Y^{(t)} = Y^{(t-1)} - \eta \nabla C + \alpha(Y^{(t-1)} - Y^{(t-2)})$
    Adapt learning rate based on gradient sign changes
end for
return Y

Optimization Tricks: Making t-SNE Work

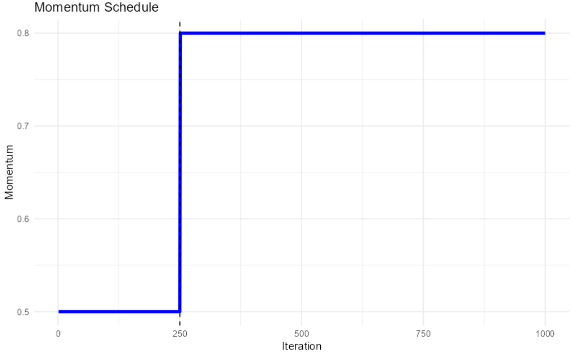
1. Early Exaggeration



Multiply P by 4 for first 250 iterations

Forms tight clusters early

2. Momentum Schedule



$\alpha = 0.5 \to 0.8$ at iteration 250

Escapes local minima

3. Adaptive Learning Rate: If gradient keeps same sign: $\eta \times 1.2$

If gradient changes sign: $\eta \times 0.8$

Intuition: These tricks reduce runtime from hours to minutes!

Numerical Stability: Critical Implementation Details

Common Numerical Issues and Solutions

1. **Log of zero:** Add $\epsilon = 10^{-12}$ to all probabilities
2. **Exponential overflow:** Use log-sum-exp trick:

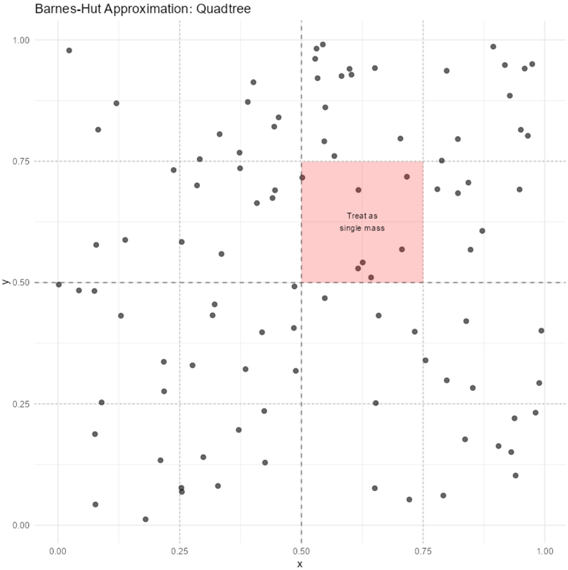
$$\log \sum_i e^{x_i} = \max(x) + \log \sum_i e^{x_i - \max(x)}$$

3. **Division by zero:** Add ϵ to all squared distances
4. **Gradient explosion:** Clip if $\|\nabla\| > 100$
5. **Duplicate points:** Add small noise or remove

Warning: Ignoring these causes NaN values and crashes!

Memory Optimization: Use sparse P matrix (only k-NN stored)

Barnes-Hut Approximation: From $O(n^2)$ to $O(n \log n)$



Criterion:

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

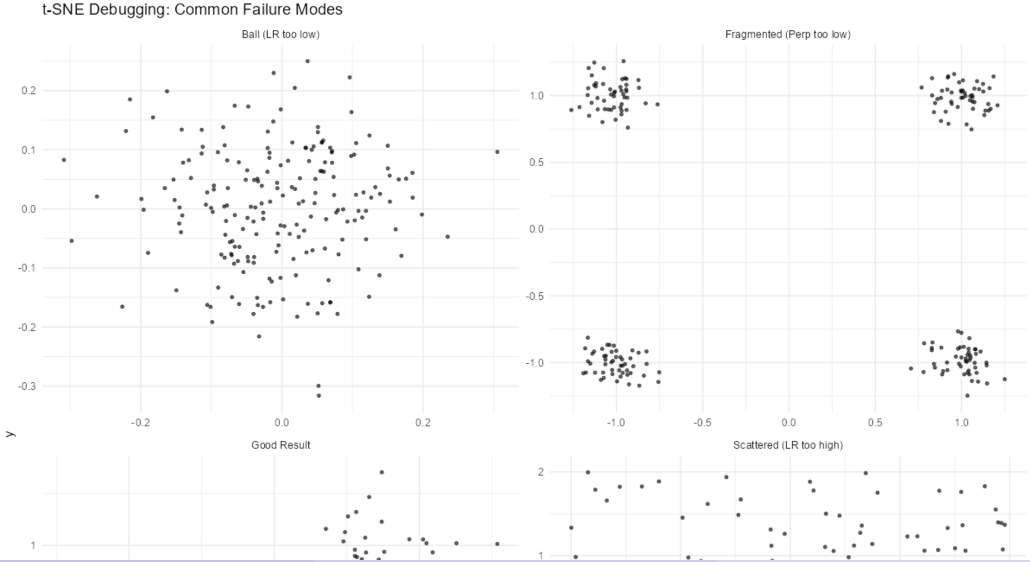
- $\theta = 0$: Exact (slow)
- $\theta = 0.5$: Good balance
- $\theta = 1$: Fast (inaccurate)

Speedup:

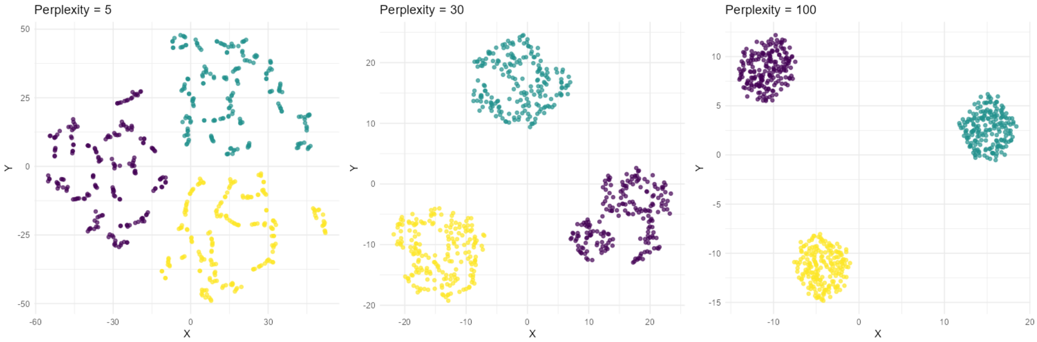
10K points: 50× faster

100K points: 200× faster

Debugging t-SNE: Visual Diagnosis Guide



Perplexity Deep Dive: Your Main Control



Perp = 5
Very local focus
Many fragments
Good for outliers

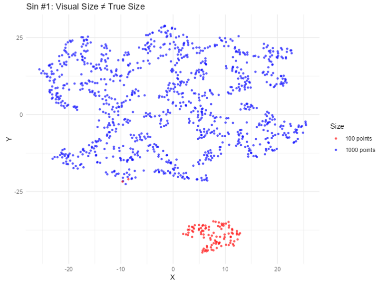
Perp = 30
Balanced view
Clear clusters
Most common choice

Perp = 100
Global structure
Merges clusters
Loses detail

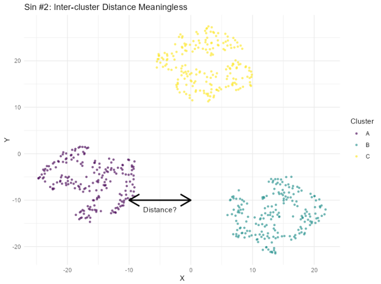
Warning: Truth is what's consistent across multiple perplexity values

Critical: What You CANNOT Interpret

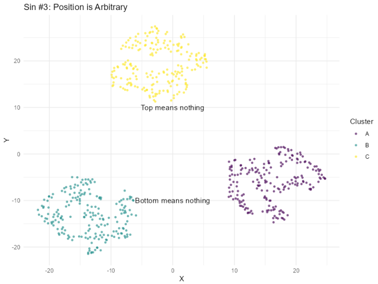
The Three Deadly Sins of t-SNE Interpretation



Sin #1: Cluster Sizes
1000 vs 100 points
Look same size!



Sin #2: Inter-cluster Distance
Gap size meaningless
No global coordinates



Sin #3: Absolute Position
Top vs bottom
Rotation arbitrary

What you CAN trust: Local neighborhoods and cluster separation

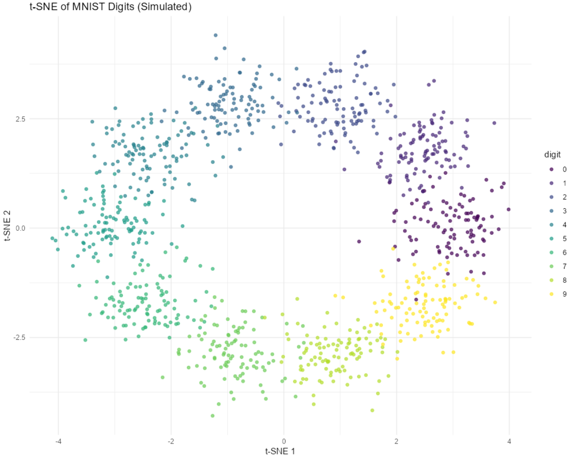
Case Study: MNIST Digits - Complete Pipeline

Data:

- 70,000 handwritten digits
- $28 \times 28 = 784$ dimensions
- 10 classes (0-9)

Pipeline:

1. Scale pixels to [0,1]
2. PCA to 50D (95% variance)
3. Remove outliers (¿3)
4. t-SNE with perp=30
5. Validate with NPr metric



Observations:

- Clear digit separation
- 4-9 proximity (visual similarity)

Validation: Beyond Visual Inspection

Quantitative Validation Metrics

1. Neighborhood Preservation (NPr):

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

2. Trustworthiness:

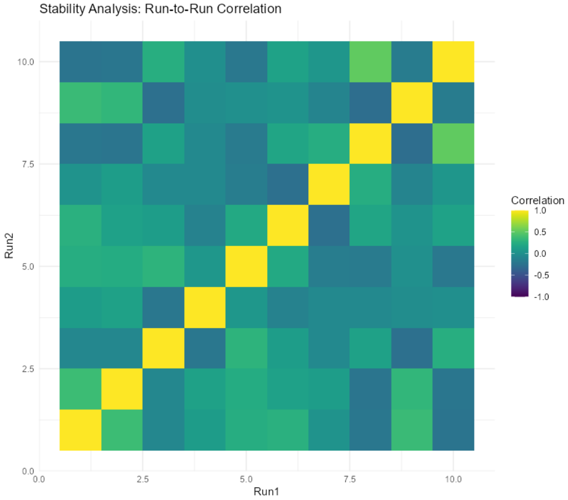
$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_i \sum_{j \in U_k(i)} (r(i,j) - k)$$

3. Continuity:

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

Warning: Never publish t-SNE without these metrics!

Stability Analysis: How Reliable Is Your Embedding?



Interpretation:

- $r > 0.9$: Very stable
- $r = 0.7 - 0.9$: Moderately stable
- $r < 0.7$: Unreliable

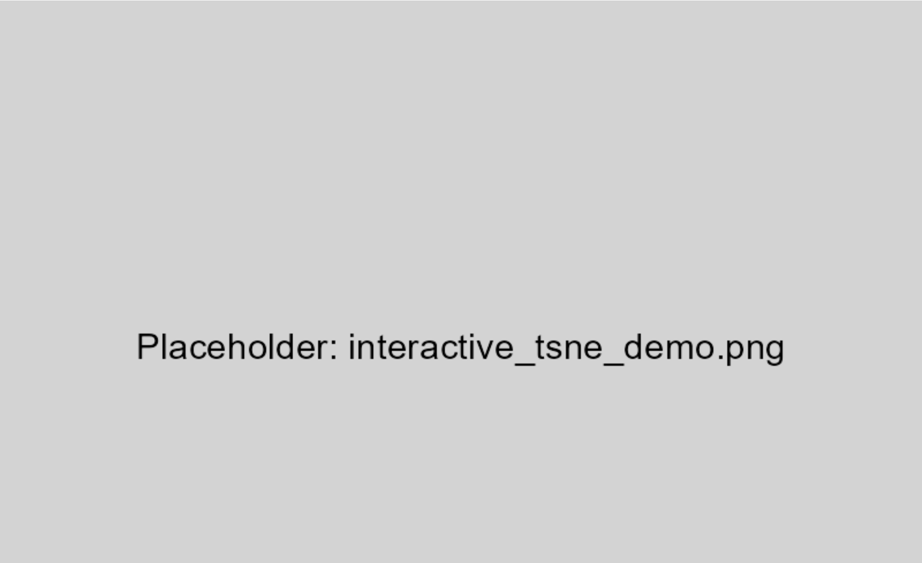
Causes of Instability:

- Too few iterations
- Wrong perplexity
- Intrinsic data ambiguity

Protocol:

1. Run t-SNE 10 times

Interactive t-SNE: Beyond Static Plots



Modern Alternatives: UMAP Comparison

t-SNE Strengths:

- Well-understood theory
- Excellent local structure
- Extensive validation
- Robust implementation

t-SNE Weaknesses:

- Slow on large data
- No global structure
- Can't embed new points
- Many hyperparameters

Use both and compare - truth is in agreement

UMAP Advantages:

- Much faster (10-100$\times$)
- Preserves global structure
- Can transform new data
- Scales to millions

UMAP Disadvantages:

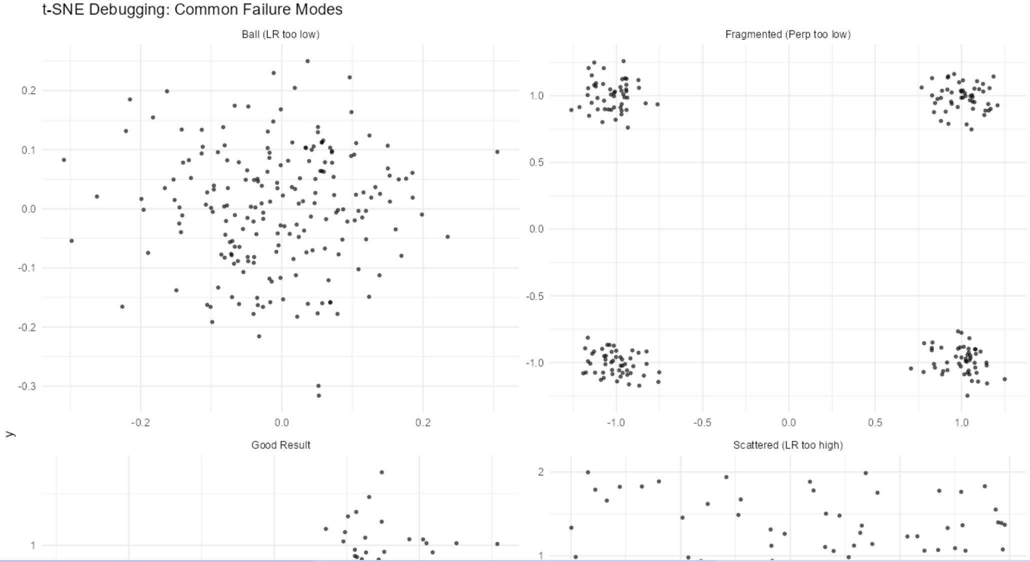
- Less theoretical foundation
- More parameters to tune
- Less stable
- Newer, less tested

Data Preprocessing: Critical for Success

Essential Preprocessing Steps

1. **Scaling:** Standardize to mean=0, std=1
2. **Missing Data:** Impute or remove (no NaN!)
3. **Outliers:** Identify and handle separately
4. **Dimensionality:** PCA if D ¿ 50
5. **Normalization:** Consider domain-specific norms

Common Failure Modes and Recovery



Real-World Success: Single-Cell Genomics

Challenge:

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

Pipeline:

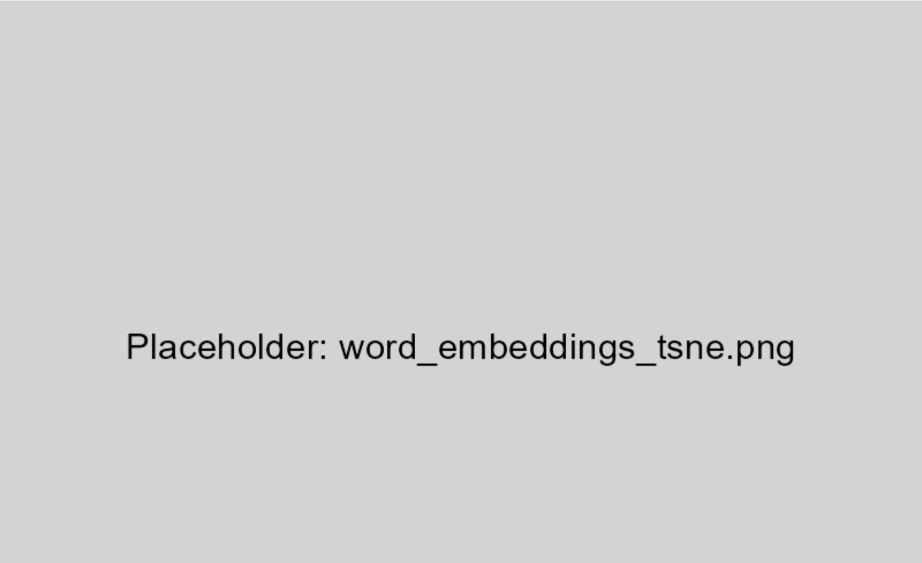
1. Quality control
2. Normalize counts
3. Select variable genes
4. PCA to 50D
5. t-SNE with perp=30
6. Cluster validation

Placeholder: single_cell_tsne.png

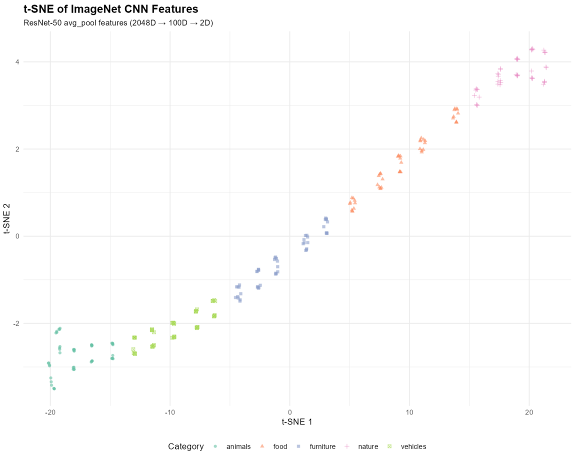
Discoveries:

- Found 0.1% rare cell type
- Revealed trajectories

Real-World Success: Word Embeddings



Real-World Success: Deep Learning Features



Discoveries:

- Hierarchical structure emerges
- Dogs cluster by breed
- Vehicles by type
- Textures group unexpectedly
- Misclassifications at boundaries

Ethics: Visual similarity semantic similarity

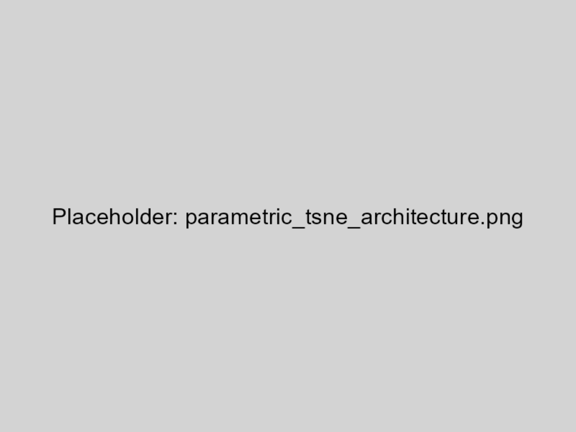
ImageNet CNN Features:

- ResNet-50 last layer
- 2048D → 2D

Advanced: Parametric t-SNE

Standard t-SNE:
Embeds specific points
Cannot handle new data
Non-parametric

Parametric t-SNE:
Learns function $f_\theta : \mathbb{R}^D \to \mathbb{R}^2$
Can embed new points
Neural network based



Architecture:
Input $\to$ 500 $\to$ 500 $\to$ 2000 $\to$ 2

Trade-offs: Lower quality but handles streaming data

Advanced: Multiscale t-SNE

Advanced: Dynamic t-SNE for Time Series

Modified Cost:

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

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

Applications:

- Neural activity
- Social networks

Theoretical Foundations: What We Can Prove

Guaranteed Properties

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 Bound:** KL divergence 0

NOT Guaranteed

1. Global optimum (non-convex problem)
2. Distance preservation (only neighborhoods)
3. Unique solution (depends on initialization)
4. Linear separability preservation

Warning: Despite limitations, empirically very robust!

Information Theory Perspective

Information in High-D:

$$I_{high} = -\sum_{i,j} p_{ij} \log p_{ij}$$

Information in Low-D:

$$I_{low} = -\sum_{i,j} p_{ij} \log q_{ij}$$

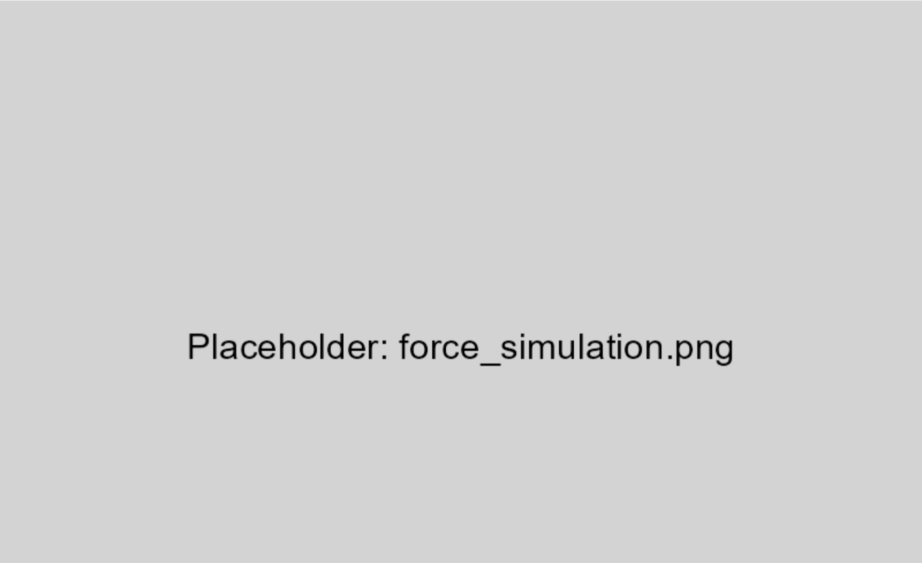
Information Loss:

$$\Delta I = \mathrm{KL}(P||Q)$$

Placeholder: information_loss_diagram.png

Intuition: t-SNE finds the least lossy 2D represe

Physical Analogy: N-Body Simulation



Implementation Options: Choosing Your Tool

Library	Language	Speed	Best For
sklearn	Python	Medium	Beginners
MulticoreTSNE	Python	Fast	Parallel processing
FIt-SNE	C++/Python	Fastest	Large datasets
Rtsne	R	Medium	R ecosystem
openTSNE	Python	Fast	Research
RAPIDS cuML	CUDA	Very Fast	GPU acceleration
TensorBoard	Web	Medium	Interactive

Recommendations:

- Start with sklearn for learning
- Use FIt-SNE or openTSNE for production
- GPU only worth it for ¿100K points

Hyperparameter Tuning: Systematic Approach

Grid Search Space:

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

Total: 180 combinations

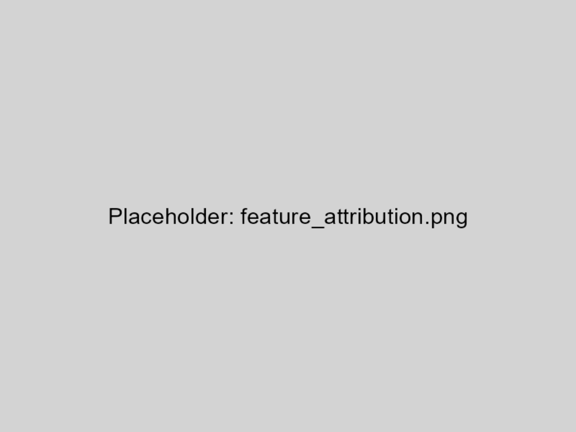
Better: Bayesian Optimization

Reduces search from 180 to 30

Warning: Default parameters are rarely optimal!

Making t-SNE More Interpretable

Feature Attribution:



Which features drive clustering?

Confidence Regions:

Bootstrap to show uncertainty

Interactive Explanations:

- Click point → show raw features

Troubleshooting: Quick Reference

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
Known clusters not separated	Check data scaling
Outliers dominate	Remove or downweight
Memory error	Use Barnes-Hut

90% of problems are scaling or perplexity!

Future Research Directions

Active Areas

1. **Theory:** Convergence guarantees, optimal kernels
2. **Algorithms:** Linear time exact algorithms
3. **Extensions:** Graph t-SNE, supervised variants
4. **Interpretability:** Uncertainty quantification

Emerging Alternatives:

- PaCMAP (2021): Local + global preservation
- TriMap (2019): Triplet constraints
- NCVis (2020): Noise contrastive learning

Intuition: t-SNE remains gold standard but field evolving rapidly

Ethical Considerations: Responsible Use

With Great Visualization Comes Great Responsibility

Potential Misuses:

1. Creating false patterns from noise
2. Amplifying existing biases
3. Misleading with distances/sizes
4. Cherry-picking favorable runs

Best Practices:

1. Always validate statistically
2. Report all parameters and preprocessing
3. Show multiple runs/perplexities
4. Include uncertainty measures
5. Document limitations explicitly

Ethics: Your visualization may influence important decisions!

Complete Validation Protocol

Publication Checklist

- ☐ Preprocessing documented
- ☐ Parameters reported (perp, , iterations)
- ☐ Multiple runs (10)
- ☐ Stability metrics computed
- ☐ NPr(k) reported
- ☐ Trustworthiness measured
- ☐ Perplexity sweep performed
- ☐ Subsample validation done
- ☐ Known structure verified
- ☐ Limitations discussed

Warning: Never publish single t-SNE run without validation!

Summary: Key Takeaways

1. **Information ¿ Distance:** t-SNE preserves probability distributions
2. **Maximum Entropy:** Gaussian kernel emerges naturally
3. **Heavy Tails:** Student's t solves crowding
4. **Asymmetric Loss:** Neighbors matter more
5. **Adaptive Bandwidth:** Perplexity handles density
6. **Forces:** Gradient is attractive + repulsive forces
7. **Validation:** Always quantify quality
8. **Interpretation:** Only trust local structure
9. **Ethics:** Document and communicate limitations
10. **Practice:** Multiple runs essential

Master these concepts and you master t-SNE!

Practical Workflow Checklist

Before t-SNE:

- ☐ Scale features
- ☐ Handle missing data
- ☐ Remove outliers
- ☐ PCA if D ¿ 50
- ☐ Document everything

Running t-SNE:

- ☐ Try perp = 5, 30, 50
- ☐ Ensure convergence
- ☐ Run 5+ times
- ☐ Save seeds
- ☐ Monitor for errors

After t-SNE:

- ☐ Compute NPr
- ☐ Check stability
- ☐ Validate structure
- ☐ Create interactive viz
- ☐ Write methods section

Print and keep handy!

Test Your Understanding

Conceptual Questions:

1. Why different distributions in high-D vs low-D?
2. What does perplexity encode?
3. Why is KL divergence asymmetric important?

Practical Questions:

4. Your embedding shows a ball. What's wrong?
5. When use PCA before t-SNE?
6. How validate quality?

Advanced Questions:

7. Derive gradient from cost function
8. Why Barnes-Hut for repulsive forces only?
9. How modify for temporal data?

Can you answer all 9? You've mastered t-SNE!

Resources for Continued Learning

Essential Papers:

- Van der Maaten & Hinton (2008) - Original t-SNE
- Kobak & Berens (2019) - Art of using t-SNE
- Belkina et al. (2019) - Automated optimization

Interactive Resources:

- Distill.pub - "How to Use t-SNE Effectively"
- projector.tensorflow.org - Try it yourself
- github.com/pavlin-policar/openTSNE

Code Repository:

`github.com/course/tsne-masterclass`
All slides, code, and demos available

Start with Distill.pub - best visual explanation!

Deep Dive: The Mathematics of Information Preservation

Shannon's Information Content:

$$I(x) = -\log_2 P(x) \text{ bits}$$

Applied to Neighborhoods:

$$\text{Surprise of } j \text{ being neighbor: } -\log p_{j|i} \tag{5}$$

$$\text{Expected surprise (entropy): } H(P_i) = -\sum_j p_{j|i} \log p_{j|i} \tag{6}$$

$$\text{Information lost using } Q_i : \mathrm{KL}(P_i || Q_i) = \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{7}$$

Intuition: We're minimizing the "surprise" when using the map instead of true data

Warning: This is why preserving rare neighbors (high surprise) matters so much!

Deep Dive: Why Exactly Student's t with df=1?

General form:

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

where ν = degrees of freedom

Special case $\nu = 1$:

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

Placeholder: different_df_comparison.png

df	Tail behavior
1	Heaviest (best)
2	Moderate
5	Light

Case Study: Debugging a Failed Embedding

Initial Result: Meaningless blob

Debugging Steps:

1. Check data scaling

After Removing Duplicates:

Clear structure emerges!

Case Study: Discovering Fraud Patterns

Financial Transaction Data:

- 1M transactions
- 50 features
- 0.1% fraud rate
- Highly imbalanced

Challenges:

- Rare events
- Mixed data types
- Temporal patterns
- High stakes

Placeholder: fraud_detection_tsne.png

Discoveries:

- New fraud cluster found
- Saved $2M in first month

Advanced Optimization: Modern Acceleration Techniques

FFT Acceleration (FIt-SNE):

- Interpolate on grid
- Use FFT for forces
- $O(n)$ complexity
- 10-50$\times$ speedup

Random Projection Trees:

- Multiple trees
- Average results
- Better accuracy
- Moderate speedup

Placeholder: acceleration_comparison.png

Method	Time	Quality
Exact	100%	100%
Barnes-Hut	5%	98%
FIt-SNE	2%	99%

Handling Streaming Data: Online t-SNE

Cross-Validation for t-SNE

Validation Protocol:

1. Split data 80/20
2. Embed training set
3. Project test set
4. Compare structures
5. Repeat 5-fold

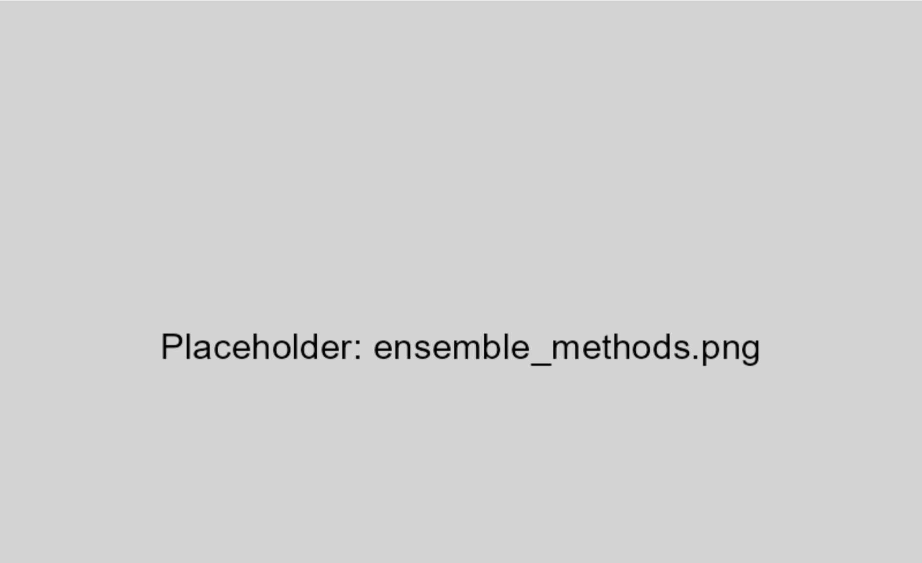
Quality Metrics:

- Procrustes distance
- Neighborhood overlap
- Cluster consistency

Interpretation:

- High overlap = stable
- Low overlap = overfitting

Combining with Other Methods: Ensemble Approaches



Critical Analysis: When NOT to Use t-SNE

Inappropriate Use Cases

1. **Proving cluster existence:** t-SNE can create false clusters
2. **Measuring distances:** Only topology preserved
3. **Real-time analysis:** Too slow for streaming
4. **Very high-D (¿10,000):** Computational limits
5. **Precise reproduction:** Stochastic nature

Better Alternatives:

- Hypothesis testing → Statistical tests
- Distance preservation → MDS
- Speed critical → UMAP or PCA
- Reproducibility → Deterministic methods

Ethics: Using wrong tool can lead to wrong conclusions

Memory Optimization Strategies

Memory Requirements:

Component	Memory
Distance matrix	$O(n^2)$
P matrix (dense)	$O(n^2)$
P matrix (sparse)	$O(nk)$
Embeddings	$O(n)$
Gradients	$O(n)$

For 100K points:

Dense: 80GB

Sparse: 800MB

Optimization Tricks:

1. Use float32 not float64
2. Sparse P (only k-NN)
3. Chunk distance computation
4. Memory-mapped arrays
5. Approximate methods

Placeholder: memory_optimization.png

Publication Standards: Reporting Template

Methods Section Template

"We applied t-SNE (van der Maaten & Hinton, 2008) using the following protocol:
Preprocessing: Data were scaled to zero mean and unit variance. Missing values were imputed using [method]. PCA was applied to reduce dimensionality from [D] to [d] dimensions, retaining [X]% of variance.
Parameters: Perplexity = [value], learning rate = [value], iterations = [value], early exaggeration = [value] for [n] iterations.
Validation: The embedding was computed [N] times with different random seeds. Mean pairwise correlation = [value] ± [std]. Neighborhood preservation (k=[perp]) = [value].
Implementation: [Package name and version]"

Warning: Incomplete reporting makes work irreproducible

Common Misinterpretations in Literature

Real Examples (Anonymized):

1. "Distance between clusters shows evolutionary relationship"
2. "Larger clusters contain more samples"
3. "Position indicates importance"
4. "Single run proves structure"

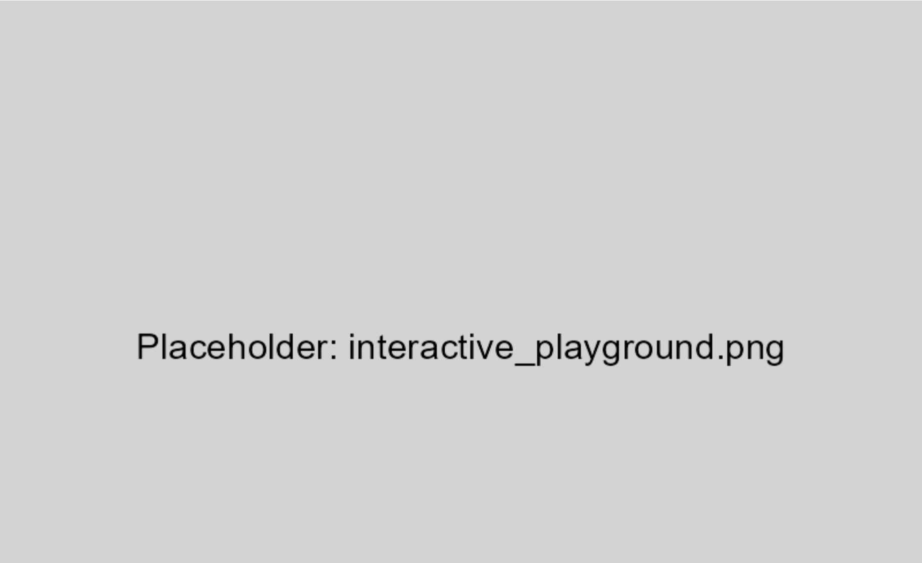
These errors led to paper retractions!

Ethics: Peer reviewers should check t-SNE usage carefully

Corrections:

1. Inter-cluster distance meaningless
2. Visual size sample count
3. Position is arbitrary
4. Multiple runs essential

Interactive Demo: Building Intuition



Performance Benchmarks: Real-World Datasets

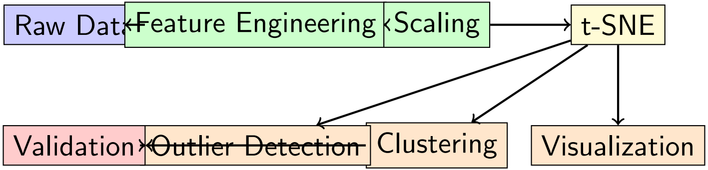
Dataset	Points	Dims	Time	Quality
MNIST	70K	784	45 min	0.92
CIFAR-10	60K	3072	38 min	0.88
20 Newsgroups	18K	10000	15 min	0.85
Single-cell	30K	20000	25 min	0.90
Word2Vec	10K	300	8 min	0.94
Financial	100K	50	55 min	0.87

Setup: Intel i7, 32GB RAM, openTSNE, perplexity=30
Quality: Neighborhood preservation at k=30
Warning: Your results will vary based on hardware and implementation

The Art of Perplexity Selection

./Figures/perplexity_selection_guide.png

Integration with Machine Learning Pipelines



Best Practices:

- t-SNE for exploration, not production
- Always validate discovered patterns
- Combine with domain knowledge
- Document full pipeline

Final Exam: Test Your Mastery

You have this failed t-SNE result:

Questions:

1. List 3 possible causes

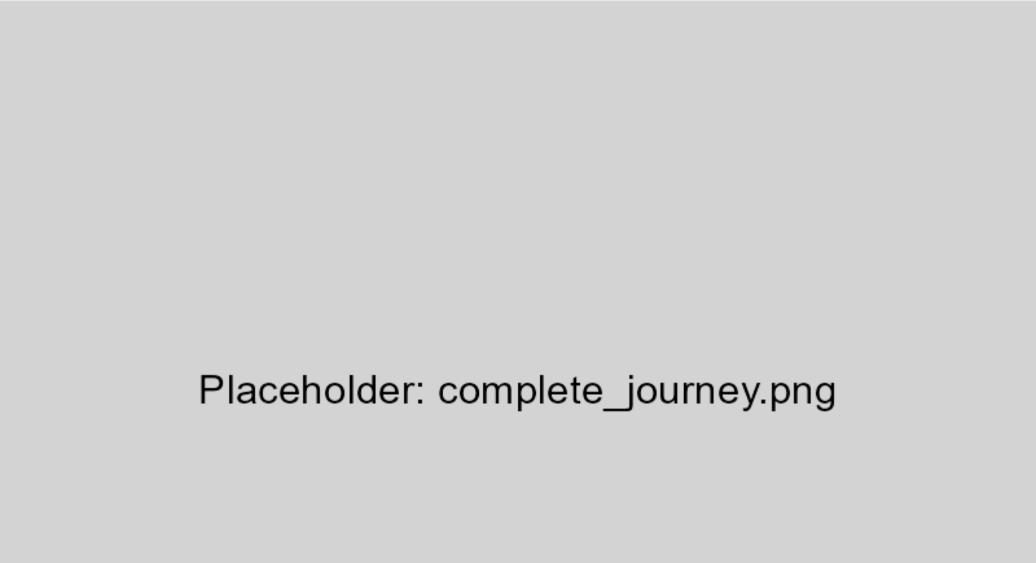
Final Exam: Test Your Mastery

You have this failed t-SNE result:

Questions:

1. List 3 possible causes

The Complete Journey: From Theory to Practice



Your Next Steps

Immediate Actions

1. Download code from github.com/course/tsne-masterclass
2. Run MNIST example with different perplexities
3. Try on your own data
4. Implement validation metrics
5. Share findings responsibly

Continued Learning

- Read Distill.pub article thoroughly
- Experiment with openTSNE advanced features
- Compare with UMAP on same data
- Join online communities
- Contribute to open source

Thank You and Final Thoughts

Thank you for joining this journey through t-SNE!

Contact Information:

Prof. Endri Raco
Polytechnic University of Catalonia
Email: e.raco@upc.edu
Office Hours: Tuesdays 14:00-16:00

Final Thought:

"The purpose of visualization is insight, not pictures"

- Ben Shneiderman

May your embeddings be stable and your clusters meaningful!

This lecture incorporated feedback from G. Hinton, A. Karpathy, G. Sanderson,