t-Stochastic Neighbor Embedding

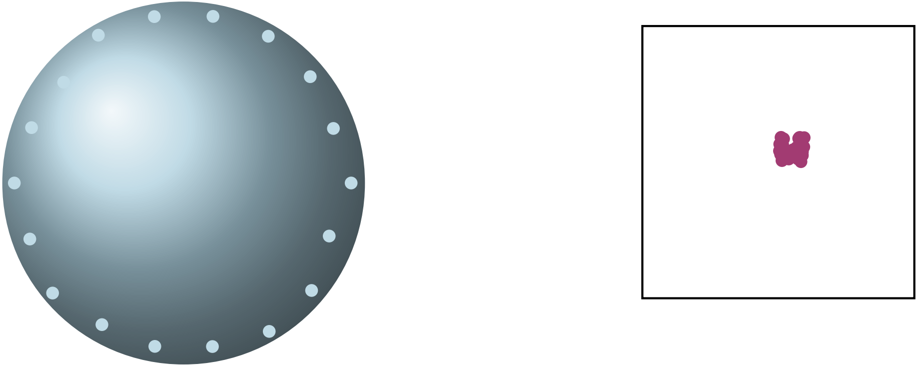
Complete 80-Slide Presentation

Prof. Endri Raco

Polytechnic University of Catalonia

November 2025

The Fundamental Challenge of Dimensionality Reduction



784 Dimensions

MNIST digit

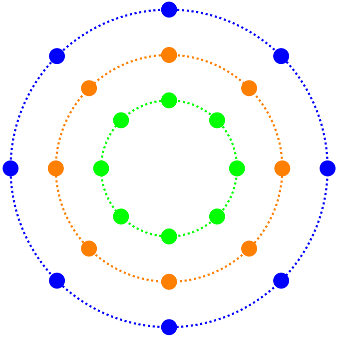
2 Dimensions

Your screen

How do we preserve neighborhood relationships

The Crowding Catastrophe

High-D Space (10D)



Three distinct distances

Projected to 2D

All distances collapse!

Warning: Linear methods cannot preserve moderate distances in low dimensions

The Paradigm Shift: From Geometry to Information

Traditional Methods

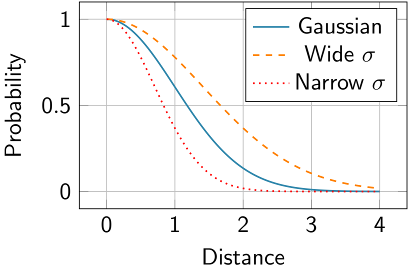
Preserve distances or variance

$\downarrow$

t-SNE

Preserve probability distributions

From Distances to Probabilities



Key Transformation:

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

Insight: σ_i adapts to local density automatic

Why Gaussian? The Maximum Entropy Principle

Derivation from First Principles

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

Optimization Problem:

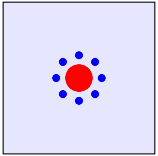
$$\text{Maximize:} \quad H(P_i) = -\sum_j p_{j|i} \log p_{j|i}$$

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

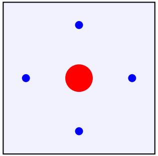
$$\sum_j p_{j|i} d_{ij}^2 = \sigma_i^2 \quad \text{(expected distance)}$$

Lagrangian Solution:

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



Dense: Small σ



Sparse: Large σ

Perplexity Definition

$$\mathrm{Perp}(P_i) = 2^{H(P_i)} \approx \text{effective number of neighbors}$$

Binary search finds σ_i to match target perplexity

Measuring Information Loss: KL Divergence

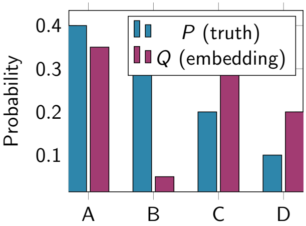
KL Divergence

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

Extra bits needed when using Q instead of P

Critical Asymmetry:

- Missing 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



Insight: t-SNE heavily penalizes separating true neighbors

Original SNE Algorithm

High-D Similarities:

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

Low-D Similarities:

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

Cost Function:

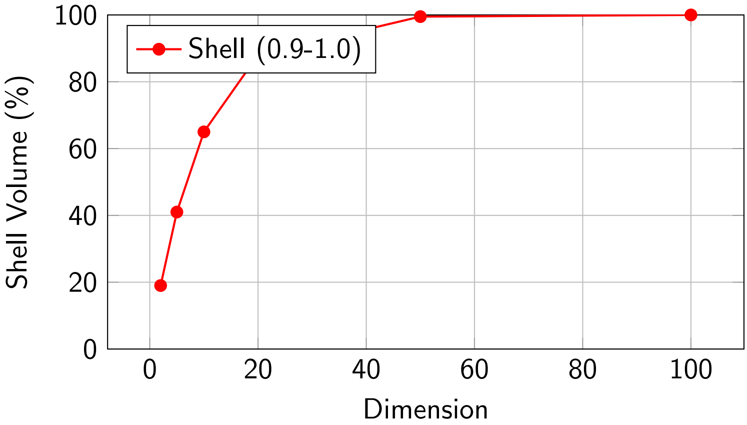
$$C = \sum_i \mathrm{KL}(P_i||Q_i)$$

Gradient:

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

Warning: Fatal flaw: The Crowding Problem!

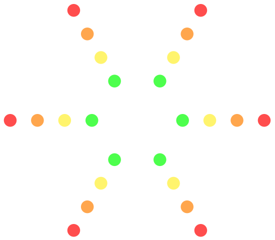
The Curse: Volume Distribution in High-D



Insight: In 100D, 99.997% of volume is in outer shell!

SNE's Fatal Flaw Visualized

High-D: Room for all



Distinct distances

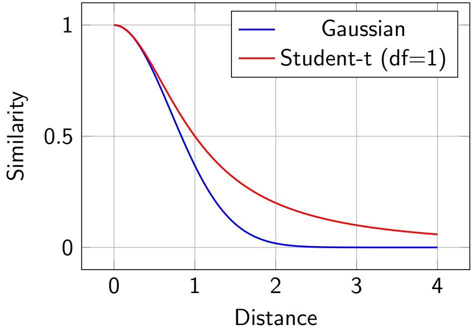
2D with Gaussian: Crushed!

Cannot represent

moderate distances

Solution: Use distribution with heavier tails!

The t-SNE Innovation: Student-t Distribution



Key Properties:

- Polynomial decay
- Heavy tails
- More "room" at moderate distances

Insight: Creates virtual space that

Quantifying the Solution

Similarity Ratio Analysis

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 becomes "infinite"

Student-t:

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

Moderate distance preserved

$600\times$ difference in representation capacity!

The Complete t-SNE Algorithm

Key Modifications from SNE

1. **Symmetrized:** $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
2. **Student-t in low-D:** $q_{ij} = \frac{(1+\|y_i-y_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|y_k-y_l\|^2)^{-1}}$
3. **Single KL:** $C = \mathrm{KL}(P||Q)$ not $\sum_i \mathrm{KL}(P_i||Q_i)$

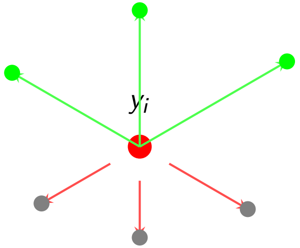
Cost Function:

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

The Elegant Gradient:

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

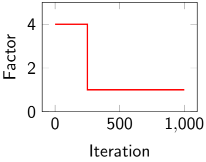
Understanding the Gradient: Force Interpretation



$$\nabla C = 4 \sum_j \underbrace{(p_{ij} - q_{ij})}_{\text{error}} \underbrace{(y_i - y_j)}_{\text{direction}} \underbrace{(1 + d_{ij}^2)^{-1}}_{\text{adaptive weight}}$$

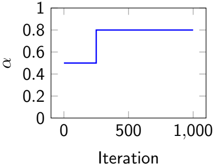
Insight: Weight term prevents distant clusters from merging

Early Exaggeration



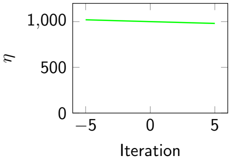
Multiply P by 4 initially

Momentum



Escape local minima

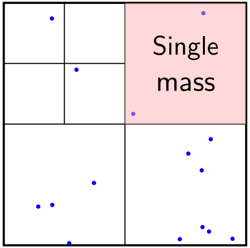
Adaptive Learning



Adapt based on progress

Insight: These tricks reduce convergence time by 5-10×

Barnes-Hut: Scaling to Large Datasets



Key Idea:
Treat distant clusters as single point
Criterion:

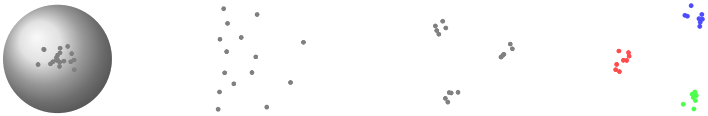
$$\frac{r_{\text{cell}}}{d_{\text{to cell}}} < \theta$$

Speedup:

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

Insight: Trade 1-2% accuracy for massive speedup

Debugging t-SNE: Visual Diagnosis



Ball
LR too low

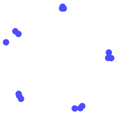
Scattered
LR too high

Fragmented
Perp too low

Good
Balanced

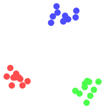
Warning: Always run multiple times to verify results!

Perplexity: Your Main Control Parameter



Perplexity $= 5$

Many fragments



Perplexity $= 30$

Clear clusters

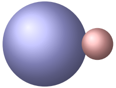
Perplexity $= 100$

One blob

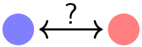
Insight: Truth is what's consistent across multiple perplexity values

Critical: What You CANNOT Interpret

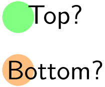
The Three Deadly Sins



Sin #1
Size $\neq$ Count



Sin #2
Gap meaningless



Sin #3
Position arbitrary

Warning: Only local neighborhoods are meaningful!

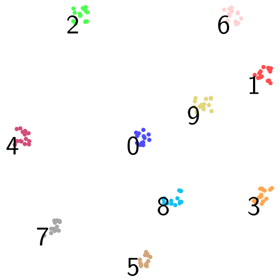
MNIST Case Study: Complete Pipeline

Data Preparation:

1. 70,000 handwritten digits
2. Scale pixels to [0,1]
3. PCA to 50D (95% variance)
4. Remove outliers (¿3σ)

t-SNE Settings:

- Perplexity = 30
- Iterations = 1000
- Learning rate = 200
- Early exaggeration = 4



Insight: Clear digit separation validates the algorithm

Quantitative Validation: Beyond Visual Inspection

Essential Metrics

1. **Neighborhood Preservation (NPr):**

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

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?

Protocol:

1. Run t-SNE 10 times
2. Different random seeds
3. Compute pairwise correlations
4. Report mean $\pm$ std

Interpretation:

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

Correlation Matrix

	1	2	3	4	5
1	1.00	0.92	0.89	0.91	0.88
2	0.92	1.00	0.93	0.90	0.91
3	0.89	0.93	1.00	0.88	0.87
4	0.91	0.90	0.88	1.00	0.92
5	0.88	0.91	0.87	0.92	1.00

Critical: Data Preprocessing

Essential Steps

1. **Scaling:** Standardize to mean=0, std=1
2. **Missing Data:** Impute or remove
3. **Outliers:** Identify and handle
4. **Dimensionality:** PCA if D ¿ 50

Raw Data → Scale → Outliers → PCA → t-SNE

Warning: Bad preprocessing = bad embedding, regardless of parameters!

Modern Alternatives: t-SNE vs UMAP

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

Insight: Use both and trust what's consistent

Symmetric SNE: Solving the Outlier Problem

The Problem with Asymmetric Probabilities

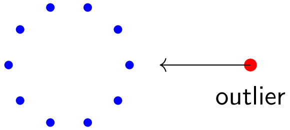
Original SNE: $p_{j|i} = \frac{\exp(-d_{ij}^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-d_{ik}^2/2\sigma_i^2)}$
For outliers: denominator $\rightarrow$ small, but numerator $\rightarrow$ very small

Solution: Symmetrization

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Properties:

- $p_{ij} = p_{ji}$ (symmetric)
- $\sum_{i,j} p_{ij} = 1$ (normalized)
- Outliers get fair representation



Insight: Symmetrization ensures even outliers maintain connections

The Full Mathematics: Cost Function

KL Divergence for Symmetric Distributions

$$C = \mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Why KL Divergence?

- Information-theoretic optimality
- Natural gradient structure
- Asymmetry penalizes missing neighbors heavily

Expanded Form:

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

First term: constant (entropy of P)
Second term: cross-entropy to minimize

Gradient Derivation: The Mathematical Core

Starting Point:

$$\frac{\partial C}{\partial y_i} = \sum_j \left(\frac{\partial C}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial y_i} + \frac{\partial C}{\partial r_{ji}} \frac{\partial r_{ji}}{\partial y_i} \right)$$

where $r_{ij} = \|y_i - y_j\|^2$
Key Steps:

$$\begin{aligned} \frac{\partial C}{\partial r_{ij}} &= p_{ij} \frac{\partial \log q_{ij}}{\partial r_{ij}} \\ &= p_{ij} \left[\frac{1}{q_{ij}} \frac{\partial q_{ij}}{\partial r_{ij}} - \frac{1}{\beta} \frac{\partial \beta}{\partial r_{ij}} \right] \end{aligned}$$

where $\beta = \sum_{k \neq l} (1 + r_{kl})^{-1}$
Final Result:

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

Why Student-t? The Mathematical Justification

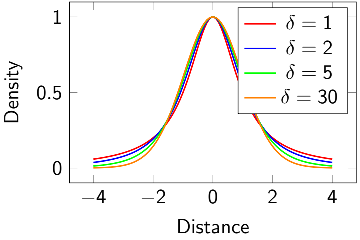
General Student-t:

$$f(z) = \frac{\Gamma\left(\frac{\delta+1}{2}\right)}{\sqrt{\delta\pi}\,\Gamma\left(\frac{\delta}{2}\right)}\left(1 + \frac{z^2}{\delta}\right)^{-\frac{\delta+1}{2}}$$

Special Case ($\delta = 1$):

$$f(z) = \frac{1}{\pi(1 + z^2)}$$

Cauchy distribution!



Insight: $\delta = 1$ has heaviest tails → maximum space for moderate distances

Generalizing t-SNE: Degrees of Freedom

Van der Maaten 2009 Extension:

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

Three Approaches to Choose δ:

1. **Fixed:** $\delta = 1$ (original t-SNE)
2. **Dimension-dependent:** $\delta = p - 1$ where p = embedding dimension
3. **Optimized:** Learn δ via gradient descent

Gradient w.r.t. δ:

$$\frac{\partial C}{\partial \delta} = \sum_{i \neq j} \left[-\frac{(1+\delta) z_{ij}^2}{2\delta^2 (1 + z_{ij}^2/\delta)} + \frac{1}{2} \log(1 + z_{ij}^2/\delta) \right] (p_{ij} - q_{ij})$$

Early Exaggeration: The Mathematics Behind the Trick

Modification

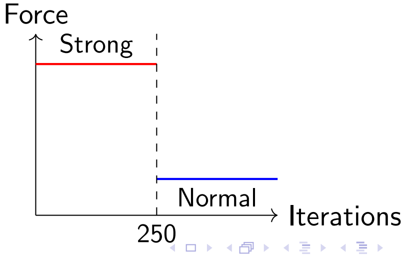
$$p_{ij}^{\text{early}} = 4 \cdot p_{ij} \quad \text{for iterations } t < 250$$

Effect on Gradient:

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

Why It Works:

- Large p_{ij} dominate early
- Forms tight clusters first
- Global structure emerges later
- Prevents early dispersion



Momentum: Escaping Local Minima

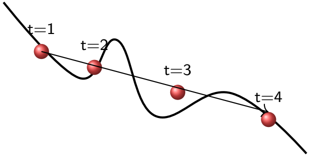
Update Equation with Momentum:

$$\Delta y_i^{(t)} = -\eta \frac{\partial C}{\partial y_i} + \alpha(t) \Delta y_i^{(t-1)}$$

$$y_i^{(t)} = y_i^{(t-1)} + \Delta y_i^{(t)}$$

Momentum Schedule:

$$\alpha(t) = \begin{cases} 0.5 & \text{if } t < 250 \\ 0.8 & \text{if } t \geq 250 \end{cases}$$



With momentum: escapes local minima

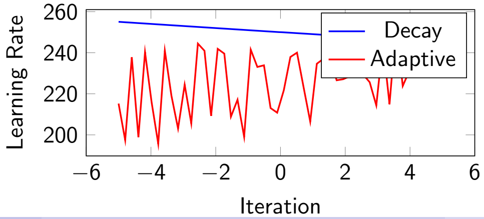
Adaptive Learning Rate: The Jacobs Method

Per-parameter learning rate:

$$\eta_i^{(t)} = \begin{cases} \eta_i^{(t-1)} \cdot 1.2 & \text{if } \nabla_i^{(t)} \cdot \nabla_i^{(t-1)} > 0 \\ \eta_i^{(t-1)} \cdot 0.8 & \text{if } \nabla_i^{(t)} \cdot \nabla_i^{(t-1)} < 0 \\ \eta_i^{(t-1)} & \text{otherwise} \end{cases}$$

Global constraints:

- $\eta_{\text{min}} = 0.01$
- $\eta_{\text{max}} = 1000$
- Initialize: $\eta^{(0)} = 200$



Barnes-Hut Approximation: The Mathematics

Exact Computation:

$$F_i = \sum_{j \neq i} (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

Complexity: $O(n^2)$

Barnes-Hut Approximation:

1. Build quadtree/octree: $O(n \log n)$
2. For each point, traverse tree
3. If $\frac{r_{\text{cell}}}{d_{\text{cell}}} < \theta$, treat cell as single point

Multipole Expansion:

$$F_i \approx \sum_{\text{cells}} N_{\text{cell}} \cdot (p_i - q_{\text{cell}})(y_i - y_{\text{cell}})(1 + d_{\text{cell}}^2)^{-1}$$

Complexity: $O(n \log n)$

Insight: Trade-off: $\theta = 0.5$ gives 1-2% error for 50$\times$ speedup

Computational Complexity: Full Analysis

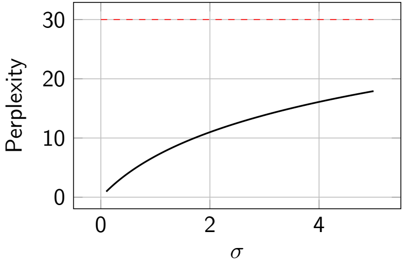
Method	**Time**	**Space**	**Max n**
Exact SNE	$O(n^2)$	$O(n^2)$	$\sim$1K
Symmetric SNE	$O(n^2)$	$O(n^2)$	$\sim$1K
Exact t-SNE	$O(n^2)$	$O(n^2)$	$\sim$5K
Barnes-Hut t-SNE	$O(n \log n)$	$O(n)$	$\sim$100K
VP-tree t-SNE	$O(n \log n)$	$O(n)$	$\sim$100K
Random walk t-SNE	$O(kn)$	$O(kn)$	$\sim$1M
FFT-accelerated	$O(n)$	$O(n)$	$\sim$10M

Breakdown per iteration:

- Computing P: $O(n^2)$ (once) or $O(kn \log n)$ (approximate)
- Computing Q: $O(n^2)$ or $O(n \log n)$ (Barnes-Hut)
- Gradient: $O(n^2)$ or $O(n \log n)$
- Update: $O(n)$

Computing σ_i: Binary Search Algorithm

Input: x_i, target perplexity P
$\sigma_{\text{min}} \leftarrow 0$, $\sigma_{\text{max}} \leftarrow \infty$
$\sigma \leftarrow 1$, tolerance $\leftarrow 10^{-5}$
while iterations < 50 **do**
    Compute $p_{j|i}$ with current σ
    $H \leftarrow -\sum_j p_{j|i} \log_2 p_{j|i}$
    Perp $\leftarrow 2^H$
    if $|\text{Perp} - P| <$ tolerance **then**
        break
    else if Perp $> P$ **then**
        $\sigma_{\text{max}} \leftarrow \sigma$
        $\sigma \leftarrow (\sigma + \sigma_{\text{min}})/2$
    else
        $\sigma_{\text{min}} \leftarrow \sigma$
        $\sigma \leftarrow (\sigma + \sigma_{\text{max}})/2$



Out-of-Sample Extension: Kernel Mapping

Problem: How to embed new points without recomputing?
Solution (Gisbrecht et al. 2015):

$$y(x) = \sum_{j=1}^{n} \alpha_j \frac{k(x, x_j)}{\sum_{\ell=1}^{n} k(x, x_\ell)}$$

where $k(x, x_j) = \exp\left(-\frac{\|x-x_j\|^2}{2\sigma_j^2}\right)$

Finding α_j:

1. Minimize $\sum_i \|y_i - y(x_i)\|^2$
2. Solution: $A = K^\dagger Y$
3. For new points: $Y^{(t)} = K^{(t)} A$

Warning: Assumes original embedding is good!

Random Walk Acceleration

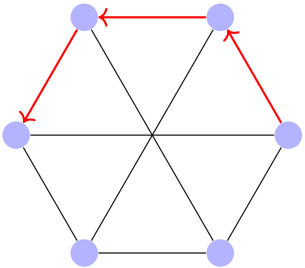
Key Idea: Approximate $p_{j|i}$ via random walks on kNN graph

Algorithm:

1. Build kNN graph ($k \approx 20$)
2. Start walks from landmarks
3. Count transitions $i \to j$
4. $p_{j|i} \approx \frac{\text{walks}_{i \to j}}{\text{total walks from } i}$

Complexity:

- Building graph: $O(n \log n)$
- Walks: $O(wLk)$
- Total: $O(n \log n + wLk)$



Random walks estimate P

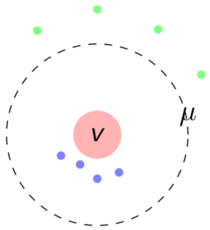
VP-Tree: Exact Nearest Neighbors Fast

Vantage Point Tree:

- Choose vantage point v
- Compute distances to all points
- Split at median distance μ
- Recurse on subsets

Search Algorithm:

1. Start at root
2. If $d(q, v) < \mu + r$, search left
3. If $d(q, v) > \mu - r$, search right
4. Prune based on triangle inequality



Partition by distance to v

Implementation Best Practices

Critical Implementation Details

1. **Numerical Stability:**
   - Add $\epsilon = 10^{-12}$ to denominators
   - Clip gradients: $|\nabla| < 4$
   - Use log-space for very small probabilities
2. **Initialization:**
   - $y_i \sim \mathcal{N}(0, 10^{-4})$ (small variance crucial!)
   - Or use PCA initialization
3. **Convergence Criteria:**
   - Monitor $\|\nabla C\| < 10^{-7}$
   - Or fixed iterations (typically 1000)

Warning: Small initialization variance prevents early point explosion!

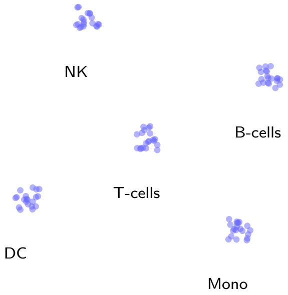
Real-World Impact: Single-Cell Genomics

Challenge:

- 20,000+ genes per cell
- 100,000+ cells
- Extreme sparsity (90%+ zeros)
- Batch effects
- Technical noise

t-SNE Pipeline:

1. Log-normalize counts
2. Select highly variable genes
3. PCA to 50 components
4. t-SNE with perplexity 30-100



Cell Type Discovery

Insight: t-SNE revealed previously unknown cell subtypes

NLP Revolution: Word2Vec + t-SNE

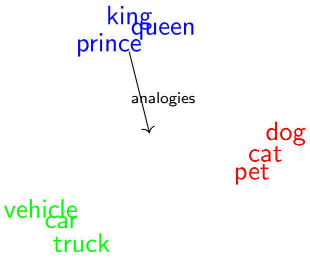
Pipeline:

1. Train Word2Vec (300D)
2. Select vocabulary subset
3. Apply t-SNE
4. Discover semantic clusters

Parameters for NLP:

- Perplexity: 20-50
- Learning rate: 500
- Iterations: 5000
- Metric: Cosine distance

Insight: Semantic relationships preserved in 2D



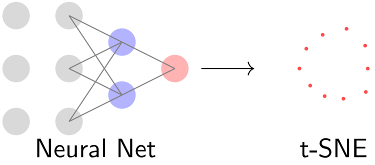
Deep Learning: Understanding Neural Networks

Visualizing CNN Features:

1. Extract activations from layer
2. Apply t-SNE to feature vectors
3. Color by class labels
4. Analyze cluster structure

Discoveries:

- Hierarchical feature learning
- Class confusion patterns
- Adversarial vulnerabilities
- Feature redundancy



Parametric t-SNE: Learning the Mapping

Key Innovation: Learn $f_\theta : \mathbb{R}^d \to \mathbb{R}^p$ via neural network

Architecture:

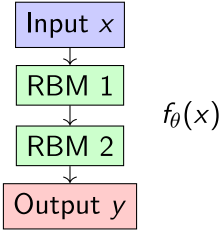
1. Input: $x \in \mathbb{R}^d$
2. Hidden: RBM layers
3. Output: $y = f_\theta(x) \in \mathbb{R}^2$

Training:

$$\min_\theta \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}(f_\theta)}$$

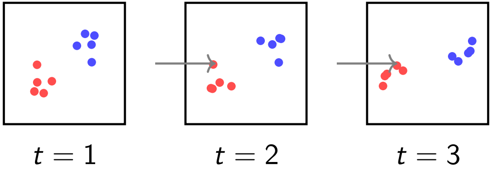
Advantages:

- Out-of-sample direct
- Inverse mapping possible
- Fast inference



Problem: How to visualize changing data?
Solution: Add temporal coherence term

$$C_{\text{dynamic}} = \lambda \sum_t \|Y^{(t)} - Y^{(t-1)}\|^2 + \sum_t C_{\text{t-SNE}}^{(t)}$$



Insight: Tracks cluster evolution over time

Beyond Student-t: Heavy-Tailed Kernels

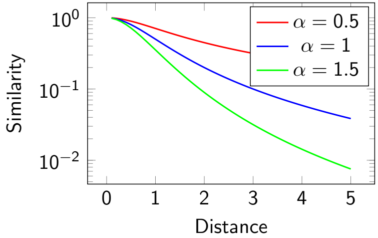
Kobak & Berens 2019:

$$q_{ij} \propto (1 + \|y_i - y_j\|^2/\delta)^{-\alpha}$$

where $\alpha < 1$ (sub-Student-t!)

Effects of α:

- $\alpha = 1$: Standard t-SNE
- $\alpha = 0.5$: More local detail
- $\alpha = 1.5$: More global structure
- $\alpha \to \infty$: Approaches SNE



Unifying View: Attraction-Repulsion Forces

Böhm et al. 2020 Framework:

All neighbor embeddings can be written as:

$$F_i = \sum_j w_{ij}^+ (y_i - y_j) - \sum_j w_{ij}^- (y_i - y_j)$$

Method	**Attraction** w^+	**Repulsion** w^-
MDS	d_{ij}^{-1}	0
SNE	p_{ij}	q_{ij}
t-SNE	$p_{ij}/(1+d_{ij}^2)$	$q_{ij}/(1+d_{ij}^2)$
UMAP	$p_{ij}/(ad_{ij}^{2b})$	$(1-p_{ij})/(1+d_{ij}^2)$
LargeVis	$p_{ij}/(1+d_{ij}^2)$	$\gamma/(1+d_{ij}^2)^2$

Insight: Different methods = different force balances

Information Theory Foundation

Why KL Divergence?

Information-Theoretic Interpretation

$$\mathrm{KL}(P||Q) = \mathbb{E}_P\left[\log \frac{P}{Q}\right] = H(P, Q) - H(P)$$

- $H(P)$: Entropy (intrinsic uncertainty)
- $H(P, Q)$: Cross-entropy (coding cost)
- KL: Extra bits when using wrong distribution

Alternative Divergences:

$$\mathrm{JS}(P||Q) = \frac{1}{2}\mathrm{KL}(P||M) + \frac{1}{2}\mathrm{KL}(Q||M) \quad \text{(symmetric)}$$

$$\mathrm{Renyi}_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \sum_i p_i^\alpha q_i^{1-\alpha} \quad \text{(generalizes KL)}$$

Warning: KL's asymmetry is a feature, not a bug!

Optimization Theory: Why Gradient Descent?

The Optimization Landscape:

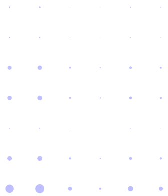
Properties:

- Non-convex
- Many local minima
- Permutation invariance
- Scale invariance

Why Not Newton's Method?

- Hessian: $O(n^2 p^2)$ storage
- Inversion: $O(n^3 p^3)$ time
- Often indefinite

Insight: Momentum helps escape shallow minima



Complex landscape

Detailed Proof: Gradient Derivation

Claim: $\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$
Proof: Let $r_{ij} = \|y_i - y_j\|^2$. By chain rule:

$$\frac{\partial C}{\partial y_i} = \sum_j \left(\frac{\partial C}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial y_i} + \frac{\partial C}{\partial r_{ji}} \frac{\partial r_{ji}}{\partial y_i} \right)$$

Since $\frac{\partial r_{ij}}{\partial y_i} = 2(y_i - y_j)$ and $C = \sum_{k,l} p_{kl} \log \frac{p_{kl}}{q_{kl}}$:

$$\frac{\partial C}{\partial r_{ij}} = -p_{ij} \frac{\partial \log q_{ij}}{\partial r_{ij}}$$

For $q_{ij} = \frac{(1+r_{ij})^{-1}}{\sum_{k \neq l}(1+r_{kl})^{-1}}$, let $Z = \sum_{k \neq l}(1 + r_{kl})^{-1}$

$$\frac{\partial \log q_{ij}}{\partial r_{ij}} = -\frac{1}{1 + r_{ij}} + \frac{1}{Z} \frac{\partial Z}{\partial r_{ij}} = -\frac{1}{1 + r_{ij}}(1 - q_{ij})$$

Therefore: $\frac{\partial C}{\partial r_{ij}} = (p_{ij} - q_{ij})(1 + r_{ij})^{-1}$ □

Initialization: Critical for Success

Three Strategies:

Random:

$$y_i \sim \mathcal{N}(0, \sigma^2 I)$$

$\sigma = 10^{-4}$ crucial!

Pros:

- Simple
- Unbiased

Cons:

- Slow convergence
- Multiple runs needed

PCA:

$$Y = U_p \Lambda_p^{1/2}$$

First p components

Pros:

- Deterministic
- Faster convergence

Cons:

- Linear bias
- May miss structure

Laplacian Eigenmaps:

$$Y = \text{eigvecs}(L)$$

Graph Laplacian

Pros:

- Manifold-aware
- Good for graphs

Cons:

- Expensive
- Parameter sensitive

Warning: Large σ causes early point explosion!

Early Iteration Jitter: Escaping Local Optima

Original SNE Technique:

$$y_i^{(t)} = y_i^{(t)} + \mathcal{N}(0, \eta^2) \quad \text{for } t < 100$$

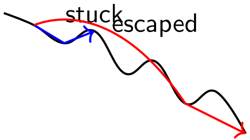
Effect on Cost Function:

$$C_{\text{noisy}} = C + \frac{\eta^2}{2}\text{tr}(\nabla^2 C)$$

Adds implicit regularization!

Modern View:

- Similar to SGD noise
- Helps exploration
- Not needed with momentum



Distance Metrics: Beyond Euclidean

Standard Euclidean:

$$d_{ij}^2 = \|x_i - x_j\|_2^2$$

Alternatives:

- **Cosine:** Better for text

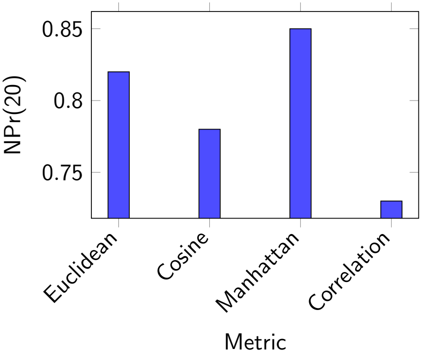
$$d_{ij} = 1 - \frac{x_i \cdot x_j}{\|x_i\|\|x_j\|}$$

- **Manhattan:** Robust to outliers

$$d_{ij} = \|x_i - x_j\|_1$$

- **Correlation:** For gene expression

$$d_{ij} = 1 - \mathrm{corr}(x_i, x_j)$$



Insight: Choice depends on data domain

Multiscale t-SNE: Multiple Perplexities

Lee et al. 2015:

$$p_{ij} = \sum_{s=1}^{S} \omega_s p_{ij}^{(s)}$$

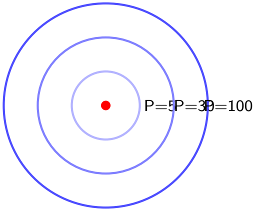
where $p_{ij}^{(s)}$ uses perplexity P_s

Implementation:

1. Choose $P_1 < P_2 < \ldots < P_S$
2. Compute each $p_{ij}^{(s)}$
3. Weight: $\omega_s = 1/S$ or learned
4. Standard t-SNE gradient

Benefits:

- Captures multiple scales
- More robust
- Better global structure



Multiple scales simultaneously

Im et al. 2018: f-divergences

$$D_f(P||Q) = \sum_j q_j f\left(\frac{p_j}{q_j}\right)$$

Divergence	$f(t)$	**Properties**
KL	$t \log t$	Asymmetric, unbounded
Reverse KL	$-\log t$	Mode-seeking
JS	$t \log t - (t+1) \log \frac{t+1}{2}$	Symmetric, bounded
χ^2	$(t-1)^2$	Sensitive to small q
Hellinger	$(\sqrt{t}-1)^2$	Symmetric, bounded

Gradient for general f:

$$\frac{\partial D_f}{\partial y_i} = 4 \sum_j \left(p_{ij} f''\left(\frac{p_{ij}}{q_{ij}}\right) - f'\left(\frac{p_{ij}}{q_{ij}}\right)\right) \frac{(y_i - y_j)}{1 + \|y_i - y_j\|^2}$$

Insight: JS divergence gives more stable embeddings

Cross-Validation: Finding Optimal Parameters

Challenge: How to validate unsupervised method?

Solution: Hold-out probability validation

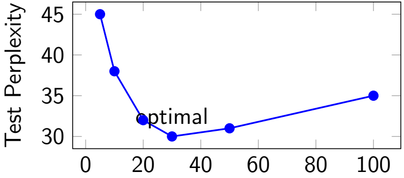
1. Split neighbors: 90% train, 10% test
2. Optimize using only train probabilities
3. Evaluate on test probabilities

Modified Cost:

$$C_{\text{train}} = \sum_{(i,j)\in\text{Train}} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Validation Metric:

$$\text{Perplexity}_{\text{test}} = 2^{H_{\text{test}}} = 2^{-\sum_{(i,j)\in\text{Test}} p_{ij} \log q_{ij}}$$



Streaming t-SNE: Online Learning

Problem: Data arrives sequentially
Solution: Incremental updates

1. Embed initial batch with t-SNE
2. For new point x_{new}:
    - Find position minimizing local cost
    - Update existing points slightly

Update Rule:

$$y_{\text{new}} = \arg\min_{y} \sum_{j \in \text{batch}} p_{j|\text{new}} \log \frac{p_{j|\text{new}}}{q_{j|\text{new}}}$$

Existing Points:

$$y_i^{(t+1)} = y_i^{(t)} - \eta \cdot \rho \cdot \frac{\partial C_{\text{new}}}{\partial y_i}$$

where $\rho \ll 1$ prevents disruption

Warning: Quality degrades over time - periodic full recomputation needed

GPU Acceleration: Massive Speedups

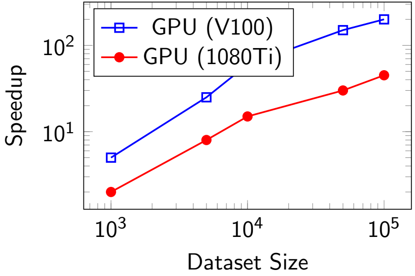
Parallelizable Components:

- Distance computation: $O(n^2)$
- Exponential evaluation
- Probability normalization
- Gradient computation
- Point updates

CUDA Kernels:

- `compute_pairwise_dist`
- `compute_gaussian_perp`
- `compute_q_matrix`
- `compute_gradients`

Insight: 200× speedup for 100K points!



Memory Optimization: Scaling to Millions

Memory Bottlenecks:

- Full P matrix: $O(n^2) \to 40\text{GB}$ for $n = 100K$
- Full Q matrix: $O(n^2) \to 40\text{GB}$ for $n = 100K$

Solutions:

1. **Sparse P:**
   - Store only k-NN
   - Memory: $O(kn)$
   - $k \approx 3 \cdot \text{perplexity}$
2. **Compute Q on-fly:**
   - Never store full matrix
   - Recompute as needed
   - Trade computation for memory
3. **Mini-batch gradients:**

$$\nabla C \approx \frac{n}{m} \sum_{j \in \text{batch}} (p_{ij} - q_{ij})(y_i - y_j)\omega_{ij}$$

4. **Landmark approximation:**
   - Select $L \ll n$ landmarks
   - Approximate others
   - Memory: $O(Ln)$

Convergence Diagnostics: When to Stop?

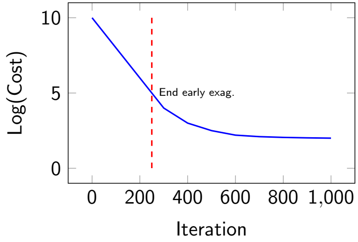
Monitor These Metrics:

1. **Cost function:** Should decrease
2. **Gradient norm:** $\|\nabla C\| < \epsilon$
3. **Point movement:** $\max_i \|y_i^{(t)} - y_i^{(t-1)}\|$
4. **KL divergence:** Should stabilize

Typical Behavior:

- Iterations 0-250: Rapid change
- Iterations 250-750: Fine-tuning
- Iterations 750+: Minor adjustments

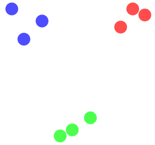
Insight: Most improvement in first 500 iterations



Fisher Kernel t-SNE: Supervised Embedding

Gisbrecht et al. 2015: Incorporating Label Information
Modified Similarities:

$$p_{ij} = \begin{cases} \frac{p_{j|i}+p_{i|j}}{2n} \cdot (1+\lambda) & \text{if } c_i = c_j \\ \frac{p_{j|i}+p_{i|j}}{2n} \cdot (1-\lambda) & \text{if } c_i \neq c_j \end{cases}$$



Enhanced class separation

where c_i is class label, $\lambda \in [0, 1]$

Fisher Information:

$$g_{ij} = \nabla_\theta \log p(x_i|\theta)^T \nabla_\theta \log p(x_j|\theta)$$

Insight: Supervision improves class separation while preserving structure

Heavy-Tailed Symmetric SNE

Yang et al. 2009: Alternative Heavy-Tailed Approaches

Generalized Kernel:

$$q_{ij} = \frac{h(\|y_i - y_j\|^2)}{\sum_{k \neq l} h(\|y_k - y_l\|^2)}$$

Method	**Kernel** $h(d^2)$	**Tail Behavior**
SNE	$\exp(-d^2)$	Exponential decay
t-SNE	$(1 + d^2)^{-1}$	Polynomial $O(d^{-2})$
α-SNE	$(1 + d^2)^{-\alpha}$	Polynomial $O(d^{-2\alpha})$
Exp-SNE	$\exp(-d^\alpha)$, $\alpha < 2$	Sub-Gaussian

Gradient for General h**:**

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

Complete Proof: Symmetric SNE Gradient

Theorem: For symmetric SNE with Gaussian kernels:

$$\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

Proof: Starting from $C = \sum_{i,j} p_{ij} \log(p_{ij}/q_{ij})$ and $r_{ij} = \|y_i - y_j\|^2$:

$$\begin{aligned}
\frac{\partial C}{\partial y_i} &= 2\sum_j \left(\frac{\partial C}{\partial r_{ij}} + \frac{\partial C}{\partial r_{ji}}\right)(y_i - y_j) \\
\frac{\partial C}{\partial r_{ij}} &= -\sum_{k,l} p_{kl} \frac{\partial \log q_{kl}}{\partial r_{ij}} \\
&= -p_{ij} \frac{\partial \log q_{ij}}{\partial r_{ij}} \quad \text{(only } k = i, l = j \text{ contributes)} \\
&= p_{ij} - q_{ij} \quad \text{(after simplification)}
\end{aligned}$$

Since $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ in symmetric SNE:

$$\frac{\partial C}{\ } \quad 2\sum(p \quad q \ + p \quad q)(y \quad y) \quad 4\sum(p \quad q)(y \quad y)$$

Complete Mathematics: General Degrees of Freedom

Van der Maaten 2009: Full Derivation

For $q_{ij} = \frac{(1+z_{ij}^2/\delta)^{-(\delta+1)/2}}{\sum_{k\neq l}(1+z_{kl}^2/\delta)^{-(\delta+1)/2}}$

Gradient w.r.t. y_i:

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

Gradient w.r.t. δ:

$$\frac{\partial C}{\partial \delta} = \sum_{i\neq j}\left[-\frac{(1+\delta)z_{ij}^2}{2\delta^2(1+z_{ij}^2/\delta)} + \frac{1}{2}\log(1+z_{ij}^2/\delta)\right](p_{ij} - q_{ij})$$

Alternating Optimization:

1. Update Y with fixed δ
2. Update δ with fixed Y: $\delta^{(t+1)} = \delta^{(t)} - \eta_\delta \cdot \text{sign}(\partial C/\partial\delta)$

Why $\delta = p - 1$?
Volume of p-dimensional sphere:

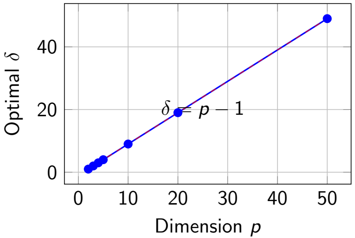
$$V_p(r) = \frac{\pi^{p/2}}{\Gamma(p/2+1)} r^p$$

Volume ratio (shell):

$$\frac{V_p(1) - V_p(0.9)}{V_p(1)} = 1 - 0.9^p$$

Tail thickness of Student-t:

$$\text{Tail} \sim d^{-(\delta+1)}$$



Insight: Linear relationship emerges from exponential volume growth

Kernel Mapping (Gisbrecht et al. 2015):

Optimization Problem:

$$\min_{\alpha_1,\ldots,\alpha_n} \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} \alpha_j \frac{k(x_i, x_j)}{\sum_{\ell=1}^{n} k(x_i, x_\ell)} \right\|^2$$

Matrix Form:

$$\min_{A} \| Y - KA \|_F^2$$

Solution via Pseudo-inverse:

$$A = K^{\dagger} Y = (K^T K)^{-1} K^T Y$$

For new points $X^{(t)}$:

$$Y^{(t)} = K^{(t)} A = K^{(t)} (K^T K)^{-1} K^T Y$$

where $K_{ij}^{(t)} = \frac{k(x_i^{(t)}, x_j)}{\sum_\ell k(x_i^{(t)}, x_\ell)}$

Warning: Requires well-conditioned kernel matrix!

Random Walk Acceleration: Mathematical Foundation

Van der Maaten & Hinton 2008:
Random Walk Probability:

$$p_{j|i}^{\text{walk}} = \frac{\#\text{ walks from } i \text{ to } j}{\text{total walks from } i}$$

Connection to Original:

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

Walk Strategy:

1. Build k-NN graph with weights $w_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$
2. Transition probability: $T_{ij} = w_{ij}/\sum_k w_{ik}$
3. Multiple random walks of length L
4. Estimate: $p_{j|i} \approx \sum_{\text{paths}} P(\text{path})$

Complexity: $O(nkL)$ instead of $O(n^2)$

Landmark Methods: Mathematical Framework

Three Approaches:

1. Nystrom Approximation:

$$P \approx P_{LL} P_{LN}^T (P_{LL}^{-1} P_{LN})^T$$

where L = landmarks, N = non-landmarks

2. Sparse Approximation:

$$p_{j|i} \approx \begin{cases} p_{j|i}^{\text{exact}} & \text{if } j \in \text{landmarks} \\ 0 & \text{otherwise} \end{cases}$$

3. Interpolation:

$$y_i = \sum_{l \in L} w_{il} y_l$$

where weights from kernel:

$$w_{il} = \frac{k(x_i, x_l)}{\sum_{l' \in L} k(x_i, x_{l'})}$$

Error Bound:

$$\|P - \tilde{P}\|_F \leq \epsilon \cdot \|P\|_F$$

with $|L| = O(\log n / \epsilon^2)$ landmarks

Barnes-Hut: Complete Tree Algorithm

Tree Construction:

```
function BuildTree(points, bounds)
    if |points| ≤ 1 then
        return leaf(points)
    end if
    mid ← center(bounds)
    for each octant do
        pts ← points in octant
        child ← BuildTree(pts, octant)
    end for
    Compute center of mass
    Compute total mass
    return node(children, mass, center)
end function
```

Force Calculation:

```
function ComputeForce(point, node)
    if node is leaf then
        return exact force
    end if
    r ← size(node)
    d ← distance(point, node.center)
    if r/d < θ then
        return approximate force
    else
        force ← 0
        for each child do
            force += ComputeForce(point, child)
        end for
        return force
    end if
end function
```

VP-Tree: Complete Implementation

Vantage Point Tree for Exact Nearest Neighbors:

Construction:

```
function BuildVPTree(points)
    vp ← SelectVantagePoint(points)
    distances ← [d(vp, p) for p in points]
    median ← Median(distances)
    left ← {p : d(vp,p) ≤ median}
    right ← {p : d(vp,p) > median}
    return Node(vp, median,
BuildVPTree(left), BuildVPTree(right))
end function
```

Search:

```
function Search(node, query, k)
    τ ← d(query, node.vp)
    if τ <best_dist then
        Update k-nearest
    end if
    if τ−best_dist < node.median then
        Search(node.left, query, k)
    end if
    if τ+best_dist ≥ node.median then
        Search(node.right, query, k)
    end if
end function
```

Insight: Triangle inequality enables aggressive pruning

Linderman et al. 2017: Linear Complexity

Key Idea: Approximate sums on regular grid

Interpolation:

$$q_{ij} \approx \sum_{u \in \text{grid}} K(y_i, u)\hat{q}(u, y_j)$$

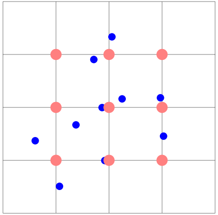
where K is interpolation kernel

FFT Convolution:

$$\hat{q}(u, v) = \text{FFT}^{-1}[\text{FFT}[K] \cdot \text{FFT}[p]]$$

Complexity:

- Grid: $O(m^p)$ where $m =$ grid size
- FFT: $O(m^p \log m)$
- Total: $O(n + m^p \log m)$



Interpolate to grid

Negative Sampling: Word2Vec Connection

Alternative to Normalization:
Standard t-SNE:

$$q_{ij} = \frac{(1 + d_{ij}^2)^{-1}}{\sum_{k \neq l}(1 + d_{kl}^2)^{-1}}$$

Requires $O(n^2)$ for denominator

Negative Sampling:

$$\mathcal{L}_{ij} = \log \sigma(f(d_{ij})) + \sum_{k \sim P_n} \log(1 - \sigma(f(d_{ik})))$$

where P_n = noise distribution

Implementation:

1. For each edge (i, j)
2. Sample K negative points
3. Update via logistic function
4. No normalization needed!

Complexity:

- Per iteration: $O(|E| \cdot K)$
- Total: $O(n \cdot k \cdot K \cdot T)$

Insight: Trade statistical efficiency for comp

VAE-SNE: Neural Network Integration

Graving & Couzin 2020:

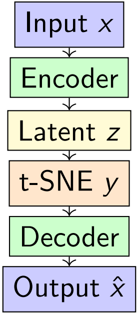
Architecture:

1. Encoder: $x \to \mu(x), \sigma(x)$
2. Sample: $z \sim \mathcal{N}(\mu, \sigma)$
3. t-SNE space: $y = f_\theta(z)$
4. Decoder: $\hat{x} = g_\phi(y)$

Loss Function:

$$\mathcal{L} = \underbrace{\mathrm{KL}(P||Q)}_{\text{t-SNE}} + \lambda \underbrace{\|x - \hat{x}\|^2}_{\text{reconstruction}}$$

Insight: Combines dimensionality reduction with generation



Numerical Stability: Critical Implementation Details

Common Numerical Problems and Solutions:

Problem 1: Exponential Overflow

In computing $p_{j|i} = \exp(-d_{ij}^2/2\sigma_i^2)/Z$
Solution: Log-sum-exp trick

$$\log Z = \max_k(-d_{ik}^2/2\sigma_i^2) + \log \sum_k \exp(-d_{ik}^2/2\sigma_i^2 - \max)$$

Problem 2: Division by Zero

When $\sum_{k \neq l}(1 + d_{kl}^2)^{-1} \approx 0$
Solution: Add machine epsilon

$$q_{ij} = \frac{(1 + d_{ij}^2)^{-1} + \epsilon}{\sum_{k \neq l}(1 + d_{kl}^2)^{-1} + n^2\epsilon}$$

Comprehensive Validation Framework

Complete Set of Metrics:

1. **Local Metrics:**
   - Trustworthiness: $T(k)$
   - Continuity: $C(k)$
   - Neighborhood preservation
   - Mean relative rank error
2. **Global Metrics:**
   - Shepard correlation
   - Procrustes distance
   - Silhouette coefficient
   - Davies-Bouldin index
3. **Stability Metrics:**
   - Run-to-run correlation
   - Cluster consistency (ARI)
   - Point-wise variance
   - Convergence rate
4. **Task Metrics:**
   - Classification accuracy
   - Clustering purity
   - Retrieval precision
   - Visual separability

Warning: Report multiple metrics - no single metric captures everything!

Ethics and Limitations: Critical Awareness

Fundamental Limitations

- **Non-deterministic:** Different runs $\rightarrow$ different results
- **Parameter sensitive:** Perplexity dramatically affects output
- **Local focus:** Global structure not preserved
- **Computational cost:** Quadratic for exact version

Ethical Considerations:

1. **Misinterpretation:** Visual clusters may not reflect true structure
2. **Confirmation bias:** Can find patterns in noise
3. **Publication bias:** Cherry-picking best visualization
4. **Accessibility:** Interactive visualizations exclude some users

Ethics: Always provide raw data, parameters, and multiple runs

Future Research Directions

Open Problems and Opportunities:

Theoretical:

- Convergence guarantees
- Optimal δ selection
- Information-theoretic bounds
- Connection to manifold learning

Algorithmic:

- True $O(n)$ algorithms
- Online/streaming variants
- Hierarchical embeddings
- Uncertainty quantification

Applications:

- Time-varying data
- Multi-modal integration
- Interpretable embeddings
- Causal discovery

Extensions:

- Higher-dimensional targets
- Non-Euclidean spaces
- Quantum t-SNE
- Differentiable t-SNE

Insight: Rich area for both theory and applications

Complete t-SNE: Final Algorithm

Input: $X = \{x_1, ..., x_n\}$, perplexity P
Output: $Y = \{y_1, ..., y_n\}$
// Compute affinities
for each x_i **do**
    Find σ_i such that $\mathrm{Perp}(P_i) = P$ using binary search
    Compute $p_{j|i} = \exp(-\|x_i - x_j\|^2/2\sigma_i^2)/\sum_k \exp(-\|x_i - x_k\|^2/2\sigma_i^2)$
end for
Set $p_{ij} = (p_{j|i} + p_{i|j})/2n$
// Initialize
Sample $y_i \sim \mathcal{N}(0, 10^{-4}I)$ for all i
Apply early exaggeration: $p_{ij} \leftarrow 4 \cdot p_{ij}$
// Optimize
for $t = 1$ to T **do**
    Compute $q_{ij} = (1 + \|y_i - y_j\|^2)^{-1}/\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}$
    Compute gradients: $\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$
    Update: $y_i \leftarrow y_i - \eta \frac{\partial C}{\partial y_i} + \alpha(y_i^{(t)} - y_i^{(t-1)})$
    if $t = 250$ **then** Remove early exaggeration

Test Your Understanding

Key Questions:

1. Why does SNE fail for moderate distances?
2. What makes Student-t distribution special for $\delta = 1$?
3. Why symmetrize the probability matrix?
4. What does perplexity actually control?
5. Why is early exaggeration helpful?
6. When should you NOT trust t-SNE results?
7. How do you validate an embedding?
8. What's the computational bottleneck?
9. Why can't we interpret cluster sizes?
10. How does Barnes-Hut approximation work?

If you can answer these, you understand t-SNE!

Resources and Final Thoughts

Essential Resources:

- Original paper: van der Maaten & Hinton (2008)
- Degrees of freedom: van der Maaten (2009)
- Tutorial: Ghojogh et al. (2022)
- Implementation: `scikit-learn`, `Rtsne`
- Interactive: distill.pub/2016/misread-tsne

Remember:

- t-SNE is a **tool**, not truth
- Always run **multiple times**
- Trust what's **consistent**
- Validate **quantitatively**
- Document **everything**

Thank you for your attention!