t-Stochastic Neighbor Embedding

A Mathematical Journey

Advanced Multivariate Analysis

UPC Barcelona

October 2025

The Fundamental Problem

Given:

- $X = \{x_1, \ldots, x_n\}$
- $x_i \in \mathbb{R}^d$, $d \gg 2$

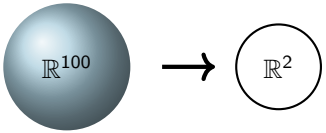
Find:

- $Y = \{y_1, \ldots, y_n\}$
- $y_i \in \mathbb{R}^2$

Information Loss:

$$\text{Loss} = \frac{d-2}{d} \times 100\%$$

$d = 100 \Rightarrow 98\%$ loss!



Key Question:
Which 2% of structure should we preserve?

Distance Matrices

Distance Matrix:

$$D_{ij} = \|x_i - x_j\|_2$$

Properties:

- Symmetric: $D_{ij} = D_{ji}$
- Non-negative: $D_{ij} \geq 0$
- Zero diagonal: $D_{ii} = 0$

Can we embed exactly in $\mathbb{R}^2$?
Usually NO!

Example: 4 points

	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

This is a tetrahedron!

- Needs $\mathbb{R}^3$ minimum
- Cannot draw in $\mathbb{R}^2$

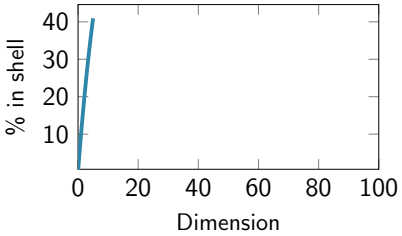
The Curse of Dimensionality

Volume of d-sphere:

$$V_d(r) \propto r^d$$

Shell volume ratio:
Inner radius $= 0.9$, Outer $= 1.0$

Dim	In shell
2	19%
10	65%
100	99.997%



In high-D: All points on boundary!

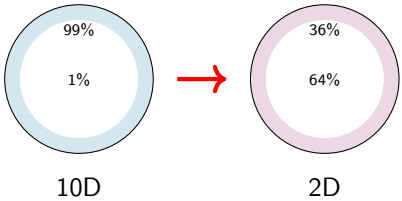
The Crowding Problem

1000 points in $\mathbb{R}^{10}$

- 99% in shell $r \in [0.8, 1.0]$
- Average distance ≈ 1.13

Project to $\mathbb{R}^2$:

- Area ratio: $(0.8)^2 = 0.64$
- 99% must fit in 36% area
- Crowding!



From Distances to Probabilities

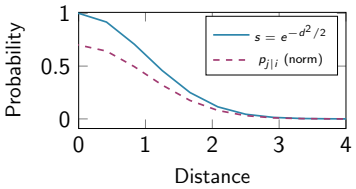
Why Probabilities?

- Distances: absolute
- Probabilities: relative
- Focus on local structure

Transform:

$$s_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right)$$

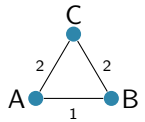
$$p_{j|i} = \frac{s_{ij}}{\sum_{k \neq i} s_{ik}}$$



$d = 1$: $s \approx 0.606$
$d = 2$: $s \approx 0.135$

Computing Conditional Probabilities

Example: 3 points



With $\sigma_A = 1$:

$s_{AB} = e^{-1/2} \approx 0.606$

$s_{AC} = e^{-2} \approx 0.135$

Sum $= 0.741$

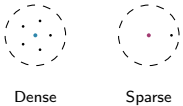
$p_{B|A} = 0.606/0.741 \approx 0.82$

$p_{C|A} = 0.135/0.741 \approx 0.18$

Key Properties:

- $\sum_j p_{j|i} = 1$ (normalized)
- $p_{j|i} \neq p_{i|j}$ (asymmetric)
- Close points: high probability
- Far points: nearly zero

Adaptive σ:



Perplexity: Setting σ_i

Definition:
Perplexity $= 2^{H(P_i)}$
where $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$

Interpretation:
Effective number of neighbors

Algorithm:

- User sets perplexity (5-50)
- Binary search finds σ_i
- Each point gets its own σ_i

Example: Perplexity = 3

3 effective neighbors

Typical values:
Small data: perplexity = 5-30
Large data: perplexity = 30-50

SNE Objective Function

Kullback-Leibler divergence:

$$C = \sum_i KL(P_i||Q_i)$$

$$= \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

What KL measures:

- Info lost using Q for P
- Asymmetric penalty
- Focus on preserving neighbors

Penalty structure:

$p_{j\|i}$	$q_{j\|i}$	Cost
Large	Small	High
Small	Large	Low

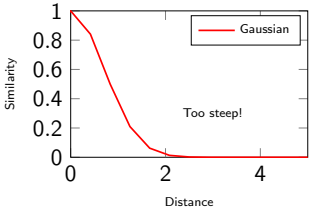
Optimization:
Gradient descent on Y

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

Why SNE Fails: Crowding in Optimization

The Problem:

- Gaussian decay: e^{-d^2}
- Too steep for moderate distances
- Forces all points together



Example:
Distance 2 vs 4:
$e^{-4}/e^{-16} = e^{12} \approx 162,754$
Ratio too extreme!

Result: Points collapse to center

The t-Distribution Solution

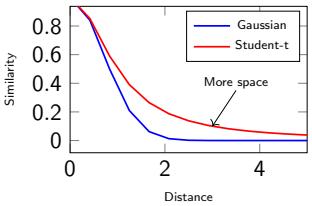
Key Innovation:
Replace Gaussian with Student-t (df=1)

In low-D space:

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

Heavy tails:

- More room for moderate distances
- Alleviates crowding
- Better separation



Ratio at d=2 vs d=4:
t-dist: $(1 + 4)/(1 + 16) \approx 0.29$
Gaussian: $e^{-4}/e^{-16} \approx 162,754$

Symmetric t-SNE: The Final Form

Symmetrization:

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

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

Benefits:

- Simpler gradients
- Better outlier handling
- Single KL divergence

Final objective:

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

Gradient:

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

where $F_{ij} = (1 + \|y_i - y_j\|^2)^{-1}$

This is t-SNE!

Understanding the Gradient: Forces

Gradient as forces:

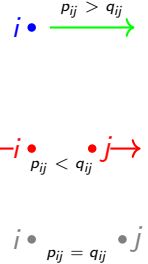
$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij}) F_{ij}(y_i - y_j)$$

Two components:

- $(p_{ij} - q_{ij})$: strength
- $(y_i - y_j)$: direction

Interpretation:

- $p_{ij} > q_{ij}$: attract
- $p_{ij} < q_{ij}$: repel



Equilibrium: Forces balance

Optimization Algorithm

Gradient Descent with Momentum:

- Initialize: $Y \sim \mathcal{N}(0, 10^{-4}I)$
- Learning rate: $\eta = 200$
- Momentum: $\alpha = 0.5$ (then 0.8)

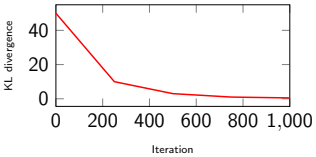
Update rule:

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$$

Early exaggeration:
Multiply all p_{ij} by 4 for first 250 iterations

Typical schedule:

Iterations	Setting
1-250	Exaggeration $= 4$ $\alpha = 0.5$
251-1000	No exaggeration $\alpha = 0.8$



Computational Complexity

Naive implementation:

- Computing all q_{ij}: $O(n^2)$
- Per iteration: $O(n^2)$
- 1000 iterations: $O(1000n^2)$

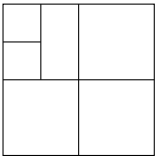
Memory: $O(n^2)$ **for** P **matrix**

Example times:

n	Time
1,000	30 sec
10,000	50 min
100,000	3+ days

Barnes-Hut t-SNE:

- Uses quad/oct-trees
- Approximates far interactions
- Complexity: $O(n \log n)$
- Accuracy parameter θ



Quadtree

Speedup: 100x for large n

Practical Implementation Tips

Preprocessing:

1. Center data: mean $= 0$
2. Scale: unit variance
3. PCA to 50 dims (if $d > 50$)
4. Remove duplicates

Parameter selection:

- Perplexity: try 5, 30, 50
- Learning rate: $n/12$ or 200
- Iterations: min 1000

Common pitfalls:

- Too few iterations
- Single perplexity value
- Not checking convergence
- Interpreting distances

Best practice:
Run 5 times with different seeds
Check stability of clusters

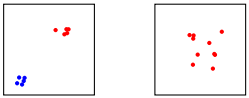
Convergence and Validation

Monitor convergence:

- KL divergence decrease
- Gradient norm $< 10^{-7}$
- Visual stability

Validation strategies:

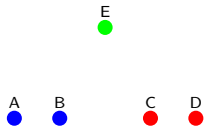
1. Known labels: check separation
2. Multiple runs: consistency
3. Different perplexities
4. Subset analysis



Quality indicators:
Clear clusters, stable across runs

Complete Example: 5 Points

High-D data (simplified):



Distance matrix:

	A	B	C	D	E
A	0	1	3	4	2.2
B	1	0	2	3	1.4
C	3	2	0	1	1.4
D	4	3	1	0	2.2
E	2.2	1.4	1.4	2.2	0

Goal:

Map to 2D preserving structure

Expected result:

- A-B close (blue cluster)
- C-D close (red cluster)
- E somewhat separate

Parameters:

- Perplexity $= 2$
- Will find σ_i for each point

Computing the P Matrix

Step 1: Find σ_A (perp = 2)

With $\sigma_A = 1.0$:

- $s_{AB} = e^{-0.5} = 0.606$
- $s_{AC} = e^{-4.5} = 0.011$
- $s_{AD} = e^{-8} = 0.0003$
- $s_{AE} = e^{-2.42} = 0.089$

Perplexity check: $2^H \approx 1.8$ ✓

Normalize:

- $p_{B|A} = 0.606/0.706 = 0.858$
- $p_{C|A} = 0.011/0.706 = 0.016$
- $p_{D|A} = 0.0003/0.706 = 0.0004$
- $p_{E|A} = 0.089/0.706 = 0.126$

Step 2: Symmetrize

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

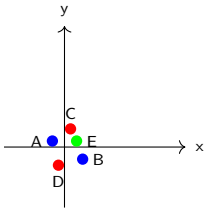
Final P matrix ($\times$1000):

	A	B	C	D	E
A	0	86	2	0.1	15
B	86	0	12	2	40
C	2	12	0	86	40
D	0.1	2	86	0	15
E	15	40	40	15	0

Note: A-B and C-D have high p_{ij}

Initial Embedding and Q Matrix

Random initialization:



All points near origin
(random $\sim \mathcal{N}(0, 10^{-4})$)

Compute Q matrix:

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

Initial Q ($\times 1000$):

	A	B	C	D	E
A	0	18	22	21	19
B	18	0	19	23	20
C	22	19	0	18	21
D	21	23	18	0	19
E	19	20	21	19	0

Nearly uniform! (all ~ 20)

First Optimization Step

Compute gradients:
For point A:

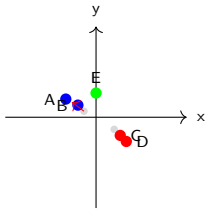
$$\frac{\partial C}{\partial y_A} = 4 \sum_j (p_{Aj} - q_{Aj}) F_{Aj} (y_A - y_j)$$

Forces on A:

- From B: $(86 - 18)\times$ attract
- From C: $(2 - 22)\times$ repel
- From D: $(0.1 - 21)\times$ repel
- From E: $(15 - 19)\times$ repel

Result: A moves toward B

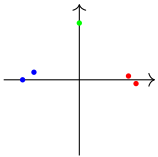
After 1 iteration:



Clusters beginning to form!

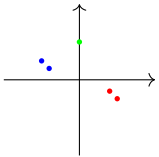
Effect of Perplexity

Perplexity = 1



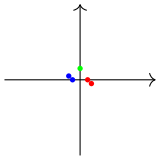
Too local

Perplexity = 2



Balanced

Perplexity = 4



Too global

Rule: $5 \leq \text{Perplexity} \leq 50$

Real Application: MNIST Digits

Dataset:

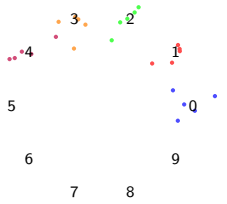
- 60,000 handwritten digits
- 28×28 pixels $= 784$ dimensions
- 10 classes (0-9)

t-SNE settings:

- PCA to 50D first
- Perplexity $= 30$
- 1000 iterations

What we see:

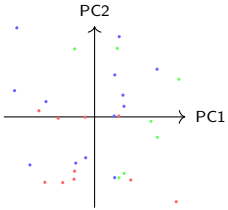
- Clear digit clusters
- Similar digits nearby
- 4-9-7 often close
- 0-6 sometimes overlap



t-SNE embedding

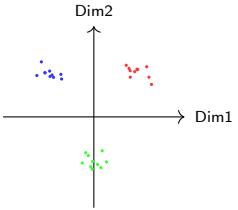
t-SNE vs PCA: Same Data

PCA (linear):



- Maximizes variance
- Linear projection
- Overlapping clusters
- Fast computation

t-SNE (nonlinear):

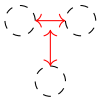


- Preserves neighborhoods
- Nonlinear mapping
- Clear clusters
- Slow computation

Interpretation Pitfalls

What NOT to trust:

1. Distances between clusters



Meaningless!

2. Cluster sizes

Not density!

What TO trust:

1. Local structure

- Points close $\rightarrow$ similar
- Within-cluster relationships

2. Cluster existence

- Clear separations meaningful
- But validate with other methods

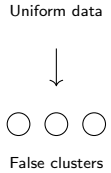
Remember:

t-SNE creates a useful map,
not a perfect representation

When t-SNE Can Mislead

Scenario 1: No real clusters

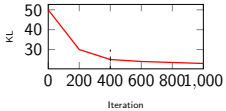
Random uniform data can show fake clusters!



Scenario 2: Wrong perplexity

- Too low: fragments clusters
- Too high: merges clusters

Scenario 3: Insufficient iterations



Solution:

- Multiple perplexities
- Run to convergence
- Multiple random seeds

Validation Strategies

1. Stability check:

- Run 5-10 times
- Different random seeds
- Compare results

2. Parameter sweep:

- Perplexity: 5, 10, 30, 50, 100
- Look for consistent patterns

3. Known structure:

- If labels exist, check separation
- Silhouette score
- Nearest neighbor preservation

4. Complementary methods:

Method	Check
PCA	Global structure
UMAP	Alternative view
Clustering	Group validity

Quality metrics:

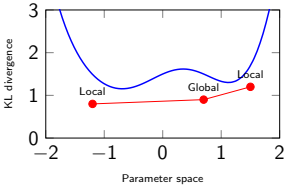
- Trustworthiness: $T(k)$
- Continuity: $C(k)$
- KL divergence convergence

Golden rule:
Never trust a single t-SNE run!

Non-convex Optimization Landscape

The Challenge:

- KL divergence is non-convex in Y
- Multiple local minima
- Different initializations $\rightarrow$ different solutions



Mathematical reason:
The function

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

is non-convex in y_i

Solution: Multiple runs

Consequence:
No guarantee of global optimum

Barnes-Hut Approximation

The Idea:

- Group distant points
- Treat as single point
- Use tree structure

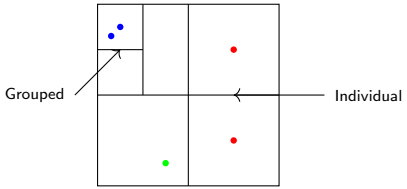
Criterion:

$$\frac{r_{\text{cell}}}{d_{\text{cell},i}} < \theta$$

where $\theta \approx 0.5$ (accuracy)

Complexity:

- Naive: $O(n^2)$
- Barnes-Hut: $O(n \log n)$



Error bound:

$|F_{\text{exact}} - F_{\text{approx}}| \leq \epsilon F_{\text{exact}}$

Mathematical Proof: Why Student-t?

Volume preservation:
In d dimensions, volume element:

$$dV_d = r^{d-1} dr \, d\Omega_{d-1}$$

For uniform distribution:

$$P(r) \propto r^{d-1}$$

In high-D:
Most volume at $r \approx \sqrt{d}$

Student-t tail:

$$t(r) \sim r^{-(d+1)}$$

Exactly compensates!

Mathematical argument:
Gaussian: $e^{-r^2} \cdot r^{d-1}$ peaks at $r = \sqrt{(d-1)/2}$
Student-t: $(1 + r^2)^{-1} \cdot r$ is flatter

Key ratio:

$$\frac{q_{ij}}{q_{kl}} = \frac{1 + \|y_k - y_l\|^2}{1 + \|y_i - y_j\|^2}$$

Linear in squared distance!
(Gaussian is exponential)

Connection to Manifold Learning

Manifold hypothesis:
High-D data lies on low-D manifold

Methods comparison:

Method	Preserves
Isomap	Geodesic distances
LLE	Local linear structure
Laplacian	Graph structure
t-SNE	Local probabilities

t-SNE advantage:
Adaptive bandwidth handles varying density

Swiss roll Unrolled

Graph Laplacian view:

$$\mathcal{L} = D - W$$

where $W_{ij} = p_{ij}$ (similarities)

Theoretical Guarantees and Limitations

What we DON'T have:

- Global convergence guarantee
- Unique solution
- Distance preservation bounds
- Consistency as $n \to \infty$

What we DO have:

- Local neighborhood preservation
- Gradient convergence to local min
- Empirical success

Open problems:

1. Optimal perplexity selection
2. Convergence rate analysis
3. Approximation quality bounds
4. Statistical properties

Key limitation:
No embedding quality guarantee
unlike PCA's variance explanation

Bottom line:
Powerful tool, but use carefully

t-SNE vs UMAP

UMAP (2018):

- Based on topology theory
- Fuzzy simplicial sets
- Cross-entropy objective
- Faster: $O(n^{1.14})$

Mathematical difference:

UMAP: $q_{ij} = (1 + a\|y_i - y_j\|^{2b})^{-1}$

t-SNE: $q_{ij} = (1 + \|y_i - y_j\|^2)^{-1}$

Parameters a, b learned from data

Feature	t-SNE	UMAP
Speed	Slow	Fast
Global	No	Partial
Theory	Prob	Topo
Params	1	4+
Stable	No	More

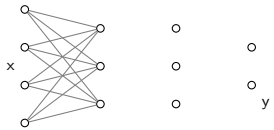
When to use which:

- t-SNE: Final visualization
- UMAP: Exploration, large data

Parametric t-SNE: Neural Network Approach

The Idea:
Learn function $f_\theta : \mathbb{R}^d \to \mathbb{R}^2$

Neural network:



Loss function:

$$\mathcal{L} = KL(P||Q) + \lambda\|\theta\|^2$$

Advantages:

- Out-of-sample predictions
- No need to store Y
- Can use mini-batches

Training:

1. Sample batch of points
2. Compute p_{ij} for batch
3. Forward pass: $y_i = f_\theta(x_i)$
4. Compute q_{ij}
5. Backprop KL gradient

Limitation:
Less flexible than non-parametric

Embedding Quality Metrics

1. Trustworthiness:

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_i \sum_{j \in U_i^k} (r_{ij} - k)$$

Measures false neighbors

2. Continuity:

$$C(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_i \sum_{j \in V_i^k} (\hat{r}_{ij} - k)$$

Measures torn neighborhoods

3. Neighborhood preservation:

$$NPR = \frac{1}{n} \sum_i \frac{|N_k(\mathrm{x}_i) \cap N_k(\mathrm{y}_i)|}{k}$$

Example values:

Metric	Good
$T(10)$	> 0.95
$C(10)$	> 0.95
NPR	> 0.7

Visual check:
Still most reliable!

Recent Advances (2020-2025)

1. FIt-SNE:

- FFT acceleration
- Interpolation-based
- 100x speedup

2. openTSNE:

- Modular implementation
- Custom objectives
- Initialization options

3. GPU-SNE:

- CUDA implementation
- Millions of points
- Real-time interaction

4. Dynamic t-SNE:

For time-series embeddings

5. Hierarchical t-SNE:

- Multi-scale visualization
- Zoom in/out of clusters
- Interactive exploration

Software and Implementation

Python:

- `sklearn.manifold.TSNE`
- `openTSNE` (recommended)
- `FIt-SNE`
- `RAPIDS cuML` (GPU)

R:

- `Rtsne` package
- `tsne` package

Best practices:

- Use PCA preprocessing
- Set verbose=True
- Save intermediate results

Typical workflow:

1. Load data
2. Standardize features
3. PCA to 50D
4. t-SNE with perp=30
5. Try perp=$\{5,50,100\}$
6. Validate clusters

Performance tips:

- Use float32, not float64
- Barnes-Hut for $n > 5000$
- GPU for $n > 50000$
- Subsample if needed

Application: Single-Cell RNA Sequencing

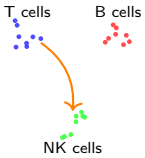
The Challenge:

- 20,000+ genes (dimensions)
- 10,000+ cells (points)
- Sparse data (90% zeros)
- Technical noise

Preprocessing:

1. Log normalization
2. Highly variable genes
3. PCA to 50 components
4. t-SNE on PCA space

Reveals: Cell types, states, trajectories



Cell differentiation

Key parameters:
Perplexity = 30-50
Learning rate = $n/12$

Application: Word Embeddings

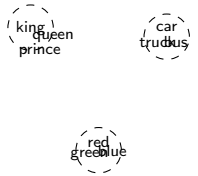
Word vectors:

- Word2Vec, GloVe, BERT
- 100-768 dimensions
- Semantic relationships

t-SNE reveals:

- Semantic clusters
- Analogies
- Language structure

Example distances:
king - man + woman $\sim$ queen
Preserved in 2D!



Semantic clusters

Special consideration:
Cosine distance often better than Euclidean for text

Application: Image Feature Analysis

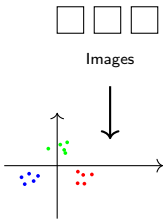
Deep learning features:

- CNN embeddings
- Last layer before classification
- 512-2048 dimensions

Use cases:

- Dataset exploration
- Outlier detection
- Class balance check
- Model debugging

Workflow:
Extract features $\rightarrow$ PCA $\rightarrow$ t-SNE



Feature clusters

Note: Similar images cluster even without labels!

Application: Time-Series and Trajectories

Dynamic t-SNE:

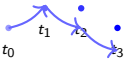
- Sliding windows
- Temporal regularization
- Preserve trajectories

Applications:

- Patient monitoring
- Financial markets
- System states
- Behavior patterns

Modified objective:

$$C = KL(P||Q) + \lambda \sum_t \|Y^t - Y^{t-1}\|^2$$



State evolution

Challenge:
Balance stability vs. change

Domain-Specific Considerations

Biology:

- Handle sparsity
- Log transform counts
- Remove batch effects
- Perplexity = 30-100

Text:

- TF-IDF preprocessing
- Cosine distance
- Remove stop words
- Perplexity = 5-50

Images:

- Use pretrained features
- Data augmentation aware

Common pitfalls by domain:

Domain	Pitfall
Biology	Batch effects
Text	Rare words
Images	Class imbalance
Finance	Temporal leakage

Universal tip:
Always validate with domain knowledge!

Remember:
t-SNE is exploratory, not definitive

Mathematical Deep Dive: Gradient Derivation (1/2)

Objective function:

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

Step 1: Derivative of $\log q_{ij}$

$$q_{ij} = \frac{(1 + d_{ij}^2)^{-1}}{Z}$$

where $Z = \sum_{k<l} (1 + d_{kl}^2)^{-1}$

$$\frac{\partial \log q_{ij}}{\partial y_i} = \frac{1}{q_{ij}} \frac{\partial q_{ij}}{\partial y_i}$$

Step 2: Chain rule

$$\frac{\partial q_{ij}}{\partial y_i} = \frac{\partial}{\partial y_i} \left[\frac{(1 + d_{ij}^2)^{-1}}{Z} \right]$$

Using quotient rule:

$$= \frac{-2(y_i - y_j)(1 + d_{ij}^2)^{-2} Z + (1 + d_{ij}^2)^{-1} \frac{\partial Z}{\partial y_i}}{Z^2}$$

Mathematical Deep Dive: Gradient Derivation (2/2)

Step 3: Simplify
After algebra:

$$\frac{\partial C}{\partial \mathrm{y}_i} = 4 \sum_j (p_{ij} - q_{ij}) F_{ij} (\mathrm{y}_i - \mathrm{y}_j)$$

where $F_{ij} = (1 + d_{ij}^2)^{-1}$

Interpretation:

- $(p_{ij} - q_{ij})$: error signal
- F_{ij}: weight function
- $(\mathrm{y}_i - \mathrm{y}_j)$: direction

Beautiful symmetry:
Force on point i from j:

$$\mathrm{F}_{ij} = 4(p_{ij} - q_{ij}) \frac{\mathrm{y}_i - \mathrm{y}_j}{1 + \|\mathrm{y}_i - \mathrm{y}_j\|^2}$$

Note: Student-t kernel appears naturally in gradient!

Initialization Strategies

1. Random initialization:

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

- Classic: $\sigma = 10^{-4}$
- Spread out: $\sigma = 0.01$
- Risk: poor local minima

2. PCA initialization:

$$Y = [v_1, v_2]^T X$$

- First 2 principal components
- Preserves global structure

3. Spectral initialization:
Graph Laplacian eigenvectors

Impact on convergence:
PCA: 500 iterations
Random: 1000 iterations

Advanced Initialization: Smart Strategies

Scaled PCA init:

1. Compute PCA
2. Scale to match t-dist variance
3. $Y = \alpha[v_1, v_2]^T X$
4. $\alpha = 0.0001 \times \max(|X|)$

Why scaling matters:
Initial q_{ij} should be uniform
Prevents early clustering

Experimental results:

Init	KL final	Time
Random	1.82	45s
PCA	1.65	30s
Scaled PCA	1.58	25s

Recommendation:
Always use PCA initialization
unless specific reason not to

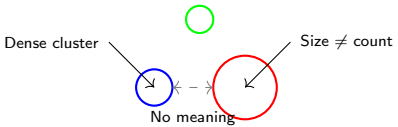
How to Read t-SNE Plots

DO interpret:

- ✓ Clusters existence
- ✓ Within-cluster structure
- ✓ Local neighborhoods
- ✓ Relative density

DON'T interpret:

- ✗ Cluster sizes
- ✗ Between-cluster distances
- ✗ Absolute positions
- ✗ Empty space meaning



Golden rule:
Confirm patterns with other methods

Common Visual Artifacts

1. Clumping:

Grid artifact

Cause: Optimization stuck

2. Outlier chains:

Chain

Cause: Early stopping

3. Pinched clusters:

Pinch

Cause: Perplexity too high

Solutions:

- Run longer
- Adjust perplexity
- Check convergence
- Multiple runs

Statistical Validation of Clusters

Silhouette score:

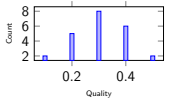
$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- a_i: mean intra-cluster dist
- b_i: mean nearest-cluster dist
- Range: [-1, 1]

Hopkins statistic:
Tests clustering tendency
$H > 0.75$: clusterable

Permutation test:

1. Run t-SNE on real data
2. Measure cluster quality Q
3. Shuffle labels 100 times
4. Run t-SNE on shuffled
5. Compare Q to null distribution



Cross-validation for t-SNE

Problem:
Can't embed test points directly

Solution 1: Parametric t-SNE
Learn mapping function

Solution 2: Out-of-sample

1. Run t-SNE on training
2. Fix training positions
3. Optimize test positions only

k-NN classification test:

1. Embed training data
2. Place test points
3. k-NN in 2D space
4. Measure accuracy

Results:

Space	Accuracy
Original	95%
t-SNE 2D	88%
Random 2D	45%

Troubleshooting Guide

Problem: One big blob

- Increase perplexity
- Run longer
- Check data scale

Problem: Too many clusters

- Decrease perplexity
- Check duplicates
- PCA preprocessing

Problem: Strange shapes

- Increase iterations
- Check for outliers
- Try different init

Diagnostic checklist:

1. KL decreasing? ✓
2. Gradient $\to 0$? ✓
3. Multiple runs similar? ✓
4. Different perplexities? ✓
5. PCA looks reasonable? ✓

Emergency fixes:

- Subsample to debug
- Remove outliers
- Try UMAP instead

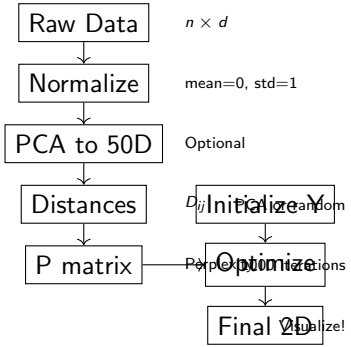
Top 10 Common Mistakes

1. Not standardizing data
2. Single perplexity value
3. Too few iterations
4. Interpreting distances
5. Ignoring convergence
6. No PCA preprocessing
7. Wrong distance metric
8. Single random seed
9. Over-interpreting
10. Not validating

How to avoid:

- Always standardize
- Try 3+ perplexities
- Min 1000 iterations
- Focus on local structure
- Check KL curve
- PCA to 50D first
- Match metric to data
- Run 5+ times
- Confirm with other methods
- Use validation metrics

The Complete t-SNE Pipeline



Mathematical Journey: Synthesis

From distances to geometry:

1. Distances: $D_{ij} = \|x_i - x_j\|$
2. Similarities: $s_{ij} = e^{-D_{ij}^2/2\sigma_i^2}$
3. Probabilities: $p_{ij} = s_{ij} / \sum_{kl} s_{kl}$
4. Embedding: $\min_Y KL(P||Q)$
5. Forces: $\nabla = 4(P - Q) \odot F \odot D$

Key insights:

- Local $>$ global
- Relative $>$ absolute
- Adaptive $>$ fixed

The mathematics that matters:

- **Information:** KL divergence
- **Probability:** Conditional $\rightarrow$ Joint
- **Optimization:** Non-convex
- **Geometry:** Manifold learning
- **Statistics:** Heavy tails

Unified view:

t-SNE = Information geometry
+ Stochastic neighbors
+ Heavy-tailed embeddings

The t-SNE Recipe: Your Go-To Guide

Ingredients:

- Data matrix: $n \times d$
- Perplexity: 30 (default)
- Iterations: 1000
- Learning rate: 200 or $n/12$

Instructions:

1. Center & scale data
2. PCA to 50D (if $d > 50$)
3. Set perplexity $\in [5, 50]$
4. Initialize with PCA
5. Run with early exaggeration
6. Check convergence
7. Try 3 perplexities
8. Run 5 times

Quality control:

- ✓ KL decreasing?
- ✓ Clusters stable?
- ✓ Known structure visible?
- ✓ No artifacts?

Serving suggestions:

- Color by known labels
- Interactive plot (plotly)
- Compare with PCA
- Validate with clustering

Warning:

Never trust distances!
Always run multiple times!

Open Problems and Future Directions

Theoretical challenges:

1. Convergence guarantees
2. Optimal perplexity theory
3. Embedding quality bounds
4. Statistical consistency
5. Manifold reconstruction

Algorithmic advances:

1. Linear time algorithms
2. Online/streaming t-SNE
3. Automatic hyperparameters
4. Interpretable embeddings

Applications frontier:

- Million-point datasets
- Real-time visualization
- 3D and VR embedding
- Temporal dynamics
- Multi-modal data

Your contribution?

Many problems remain open!
Combine theory + practice

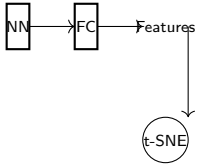
t-SNE in the Deep Learning Era

Visualizing deep networks:

- Layer activations
- Learned representations
- Attention patterns
- Embedding spaces

Model debugging:

- Feature collapse
- Class separation
- Adversarial examples
- Transfer learning



Integration approaches:

1. Extract $\rightarrow$ t-SNE $\rightarrow$ Visualize
2. Joint training (parametric)
3. Regularization with t-SNE

Complete Example: Iris Dataset

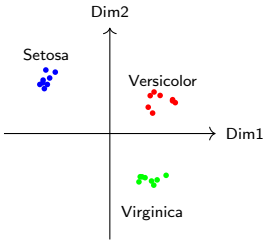
Data:

- 150 samples, 4 features
- 3 species (50 each)
- Classic test case

Steps:

1. Load: (150×4) matrix
2. Scale: mean=0, std=1
3. Distances: Euclidean
4. Perplexity: 30
5. σ search: $[0.1, 10]$
6. P matrix: symmetrize
7. Init: PCA $\times$ 0.0001
8. Optimize: 1000 iterations

Results:



Perfect separation achieved!

Key Takeaways: What to Remember

Core concepts:

1. Preserves neighborhoods, not distances
2. Probabilities ¿ distances
3. Heavy tails solve crowding
4. Non-convex optimization
5. Multiple runs essential

Practical wisdom:

- Always standardize
- Try multiple perplexities
- PCA preprocessing helps
- Validate findings

Mathematical insights:

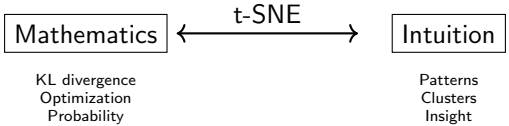
- KL divergence for matching
- Student-t for heavy tails
- Gradient = attractive/repulsive forces
- Adaptive bandwidth crucial

Remember:
t-SNE is a tool for exploration,
not proof!

Use wisely, interpret carefully

Final Thoughts: The Art and Science of t-SNE

t-SNE bridges mathematics and intuition



You now have:

- Mathematical understanding
- Practical skills
- Critical perspective

Go forth and visualize responsibly!