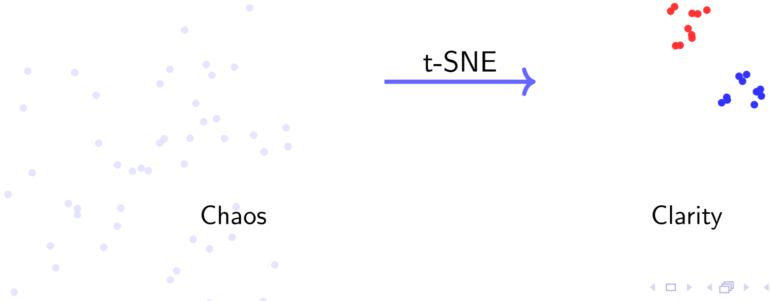
t-SNE: Seeing the Invisible Structure

Advanced Multivariate Analysis

Associate Professor Endri Raco
Polytechnic University of Catalonia



Our Journey Together

This Session:

- Exchange of ideas
- Build intuition first
- Then mathematics
- Your questions welcome

You'll Master:

- Why t-SNE works
- When to use it
- How to implement
- Pitfalls to avoid

"By the end, you'll see data differently"

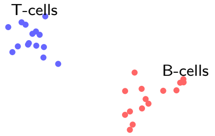
The Data Visualization Challenge

Single-cell RNA
20,000 dimensions!

Cell	Gene1	Gene2	...
1	0.23	1.45	...
2	0.67	0.89	...
3	1.23	0.02	...
...	...	...	...

How find cell types?

After t-SNE:



Cell types visible!

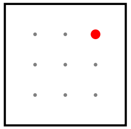
The Curse: A Thought Experiment

Finding Your Friend at a Concert

1D: Line

10 people

2D: Field



100 people

3D: Stadium

1000 people

In 20,000 dimensions?

Your friend is *everywhere and nowhere*

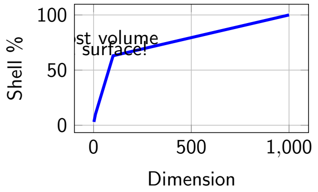
The Curse: Mathematics

Volume in n-dim sphere:

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

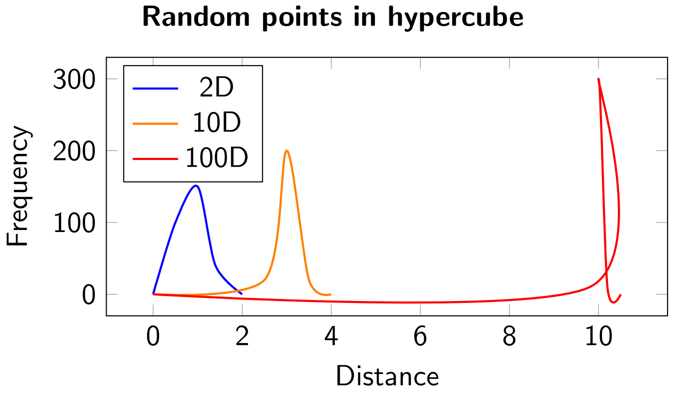
Shell vs Core:

$$\frac{V_{shell}}{V_{total}} = 1 - (0.99)^n$$



Key Insight: All points become equally distant!

Distance Collapse



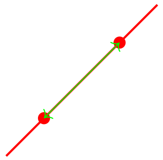
Problem: No meaningful neighborhoods!

Why PCA Fails

The Swiss Roll Problem

True Structure

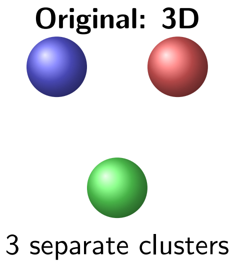
PCA Result

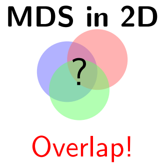


Wrong distance!

PCA assumes linear subspace - misses manifold structure

MDS: The Crowding Problem



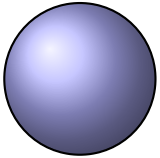


Not enough "room" in 2D for all distances

The Manifold Hypothesis

High-D data lies on low-D manifolds

Earth

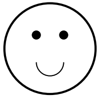


3D sphere

$\downarrow$

2D surface

Faces

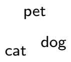


10K pixels

$\downarrow$

50 dimensions

Words



Vocabulary

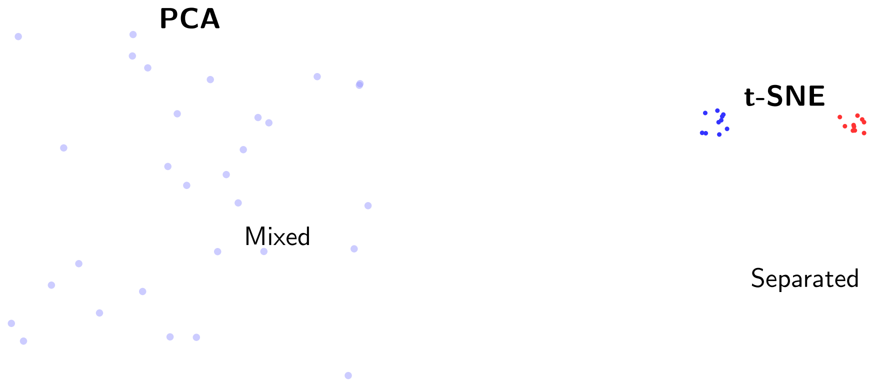
$\downarrow$

300D semantic

Key: True complexity $\ll$ apparent dimensions

t-SNE's Revolutionary Promise

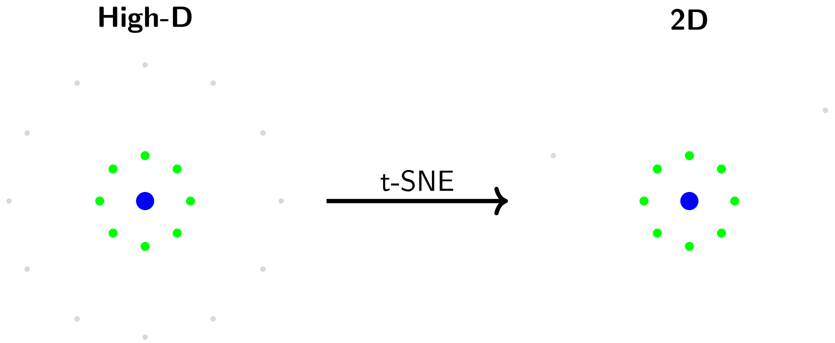
MNIST Digits: 784D $\to$ 2D



Key: Preserve neighborhoods, not distances

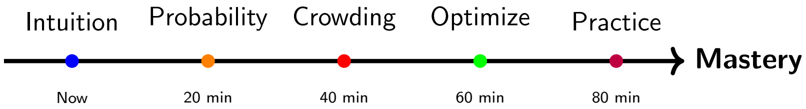
The Fundamental Insight

Local vs Global



- ✓Keep nearby points together
- ○ Let distant points reorganize

Our Learning Journey



Questions welcome at each checkpoint!

Mathematical Prerequisites Check

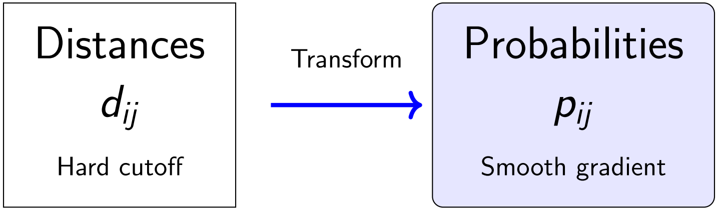
Essential Concepts You'll Need

- **Entropy:** $H(p) = -\sum_i p_i \log(p_i)$
  - Fair coin: $H = -0.5\log(0.5) - 0.5\log(0.5) = 1$ bit
  - Loaded coin $(0.9, 0.1)$: $H = 0.47$ bits $\rightarrow$ less uncertainty
- **KL Divergence:** $KL(P||Q) = \sum_i p_i \log(p_i/q_i)$
  - Example: $P = [0.7, 0.3]$, $Q = [0.5, 0.5]$
  - $KL = 0.7\log(0.7/0.5) + 0.3\log(0.3/0.5) = 0.08$
- **Gradient Descent:** $x_{new} = x_{old} - \eta \cdot \nabla f(x)$
  - Example: $f(x) = x^2$, $\nabla f(4) = 8$, $\eta = 0.1$
  - $x_{new} = 4 - 0.1 \cdot 8 = 3.2 \rightarrow$ moving toward minimum

Don't worry – we'll review these as we use them!

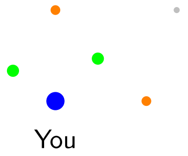
Part 2: From Distances to Probabilities

The Core Innovation



The "Friends" Analogy

Your Social Network



Picking Probability

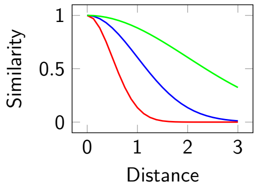
Best friend:	40%
Close friends:	30%
Acquaintances:	25%
Strangers:	5%

Closer = Higher probability

t-SNE does this for **every** point!

The Gaussian Kernel

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



Effect of σ_i:

- Small: Very local
- Medium: Balanced
- Large: Global view

Each point gets its own σ_i!

Building the Probabilities

1. **Compute distances**

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

2. **Apply Gaussian**

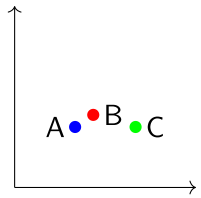
$$\tilde{p}_{j|i} = \exp(-d_{ij}^2/2\sigma_i^2)$$

3. **Normalize**

$$p_{j|i} = \frac{\tilde{p}_{j|i}}{\sum_{k \neq i} \tilde{p}_{k|i}}$$

4. **Interpretation** Probability that i picks j as neighbor

Example: 5 Points



From A's perspective:

Point	Distance	$p_{j\|A}$
B	0.36	51%
C	1.00	9%
Others	¿1.5	¡1%

B is A's primary neighbor

Worked Example: Computing p_{ij} Step-by-Step

Given: 3 points in 2D, perplexity $= 2$
Points: $A(0,0)$, $B(1,0)$, $C(3,0)$

1. **Compute distances from A**
   - $d_{AB} = 1$, $d_{AC} = 3$
2. **Find σ_A via binary search**
   Target: $H(P_A) = \log(2) = 0.693$

Try $\sigma = 0.5$:

- $\tilde{p}_{B|A} = \exp(-1^2/2 \cdot 0.25) = 0.135$
- $\tilde{p}_{C|A} = \exp(-9/2 \cdot 0.25) \approx 0$
- $H = 0.41$ (too low)

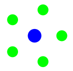
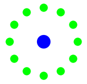
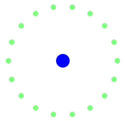
Try $\sigma = 1.0$:

- $\tilde{p}_{B|A} = \exp(-0.5) = 0.606$
- $\tilde{p}_{C|A} = \exp(-4.5) = 0.011$
- $H = 0.69$ ✓

3. **Normalize**
   - $p_{B|A} = 0.606/(0.606 + 0.011) = 0.982$
   - $p_{C|A} = 0.018$

Perplexity: The Key Parameter

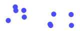
Perplexity = "How many neighbors?"

Perp = 5	**Perp = 30**	**Perp = 100**
		
Very local	Balanced	Global

Perplexity in Action

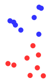
Same data, different perplexity

Perp = 5



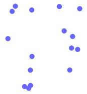
Fragmented

Perp = 30



Just right

Perp = 100

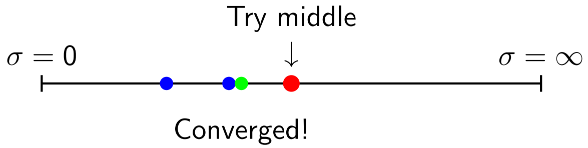


Merged

Rule: Perplexity between 5 and 50

Finding the Right σ_i

Binary Search for Each Point



- Target: perplexity $\rightarrow$ entropy
- Adjust σ until match
- Converges in 10 iterations
- Do this for ALL n points!

Perplexity: The Math

Definition (Perplexity)

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

where entropy:

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

Interpretation:

- Effective number of neighbors
- Perp $= 30 \approx 30$ equally likely neighbors
- Controls local vs global focus

Binary search finds σ_i such that:

$$\text{Perp}(P_i) = \text{user specified perplexity}$$

Perplexity Pitfalls

Common Mistakes

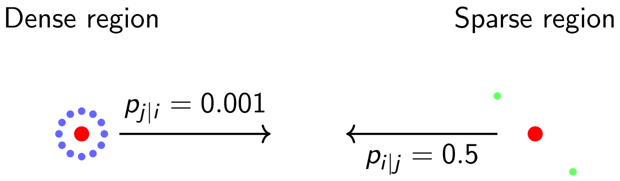
Setting	**Result**	**Fix**
Perp $= 2$	Islands	Increase to $15+$
Perp $= 200$	Blob	Decrease to 50
Perp ¿ n/3	Unstable	Use 5-50 range

Best Practice:

- Try multiple values (15, 30, 50)
- Look for stable patterns
- Consider data size: larger n $\rightarrow$ larger perp OK

The Asymmetry Problem

$$p_{j|i} \neq p_{i|j}$$



Problem: Outliers pull but aren't pulled!

Symmetrization Solution

Simple Fix:

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

Properties:

- Symmetric
- Sum to 1
- Fair to all points

Example:

Before:
$p_{j|i} = 0.001$
$p_{i|j} = 0.500$

After:
$p_{ij} = 0.250/n$
$p_{ji} = 0.250/n$

Balanced!

Mathematical Check

Theorem (Joint Distribution)

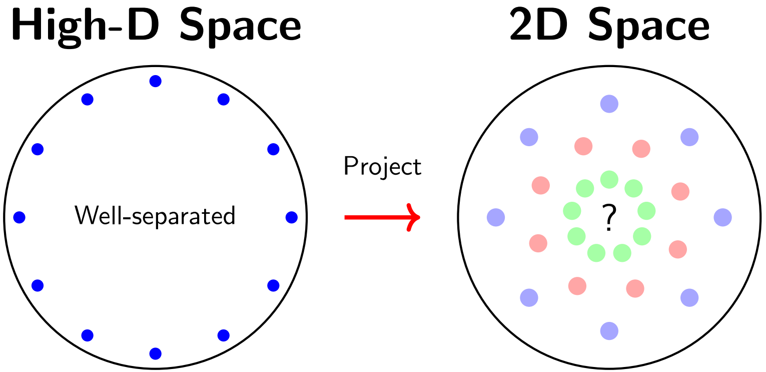
With $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$*:*

$$\sum_{i,j} p_{ij} = 1$$

Proof:

$$\begin{aligned}\sum_{i,j} p_{ij} &= \sum_{i,j} \frac{p_{j|i} + p_{i|j}}{2n} \\ &= \frac{1}{2n}\left[\sum_{i,j} p_{j|i} + \sum_{i,j} p_{i|j}\right] \\ &= \frac{1}{2n}[n+n] = 1 \quad \checkmark\end{aligned}$$

Why Gaussians Fail



Not enough "room" in 2D!

Why PCA to 50D First?

Computational Benefits

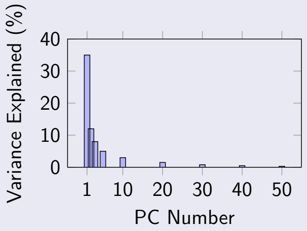
Original: 20,000D $\times$ 10,000 points

- Distance calc: $O(20,000)$
- Memory: 1.6 GB (float64)

After PCA to 50D:

- Distance calc: $O(50)$
- 400$\times$ faster!
- Memory: 4 MB
- 400$\times$ less!

Quality Preservation



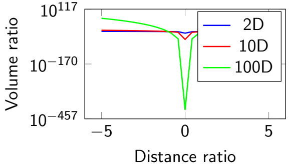
Example with RNA-seq data

- Original: 20,000 genes
- PCA: 50 components capture 92% variance

Volume Scaling Problem

Volume ratio:

$$\frac{V_n(2r)}{V_n(r)} = 2^n$$



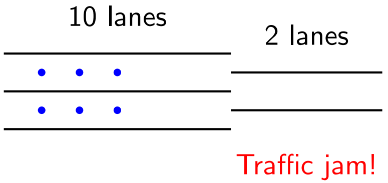
Examples:

- 2D: $2^2 = 4\times$
- 10D: $2^{10} = 1024\times$
- 100D: $2^{100} \approx 10^{30}\times$

Problem: Can't preserve moderate distances in 2D!

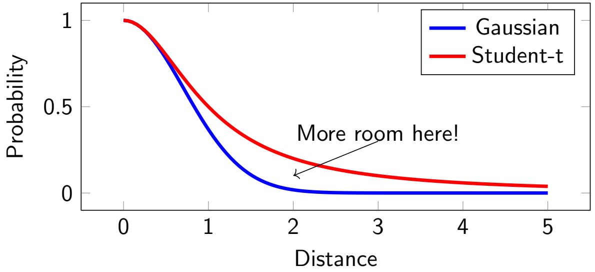
The Traffic Jam

Trying to fit 10D structure in 2D



Solution needed: Different distance function in 2D

The Solution: Heavy Tails



Heavy tails = more probability at moderate distances

Why Student-t with df=1?

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

Why df=1?

- Heaviest tails
- Simple gradient
- Fast computation
- Works best!

Tail Comparison:

Distribution	Decay
Gaussian	e^{-d^2}
t(df=1)	d^{-2}
t(df=5)	d^{-6}

df=1 gives most room!

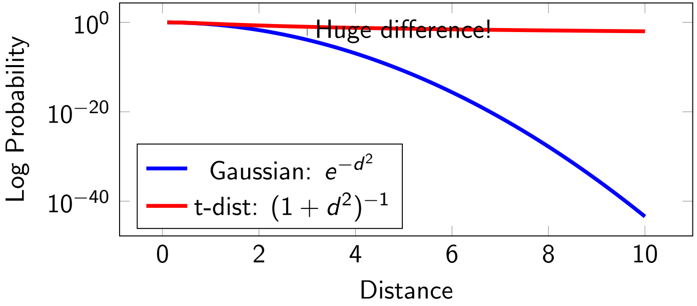
Student-t Distribution: The Mathematics

Definition (Student-t with 1 degree of freedom)

In low dimensions, we use: $q_{ij} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1+||y_k - y_l||^2)^{-1}}$

This is the Cauchy distribution: $f(x) = \frac{1}{\pi(1+x^2)}$
Key property: Polynomial decay vs exponential

Tail Behavior Analysis



At $d = 3$: Gaussian $\approx 10^{-4}$, Student-t ≈ 0.1

The Elegant Gradient

Theorem (t-SNE Gradient)

With Student-t in low dimensions: $\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$

Compare complexity:

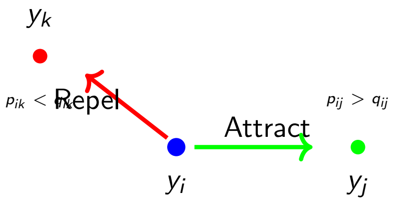
Gaussian:

- Compute $e^{-||y_i - y_j||^2}$
- Expensive exp()
- Numerical issues

Student-t:

- Compute $(1 + ||y_i - y_j||^2)^{-1}$
- Simple division
- Stable

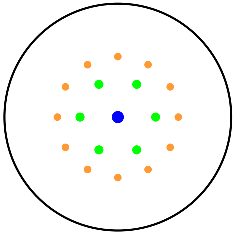
Forces in t-SNE



Spring analogy: $(p_{ij} - q_{ij})$ = spring tension

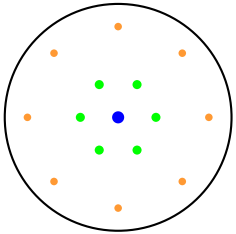
How t-Distribution Solves Crowding

Gaussian



Crowded at moderate

Student-t



Room at moderate

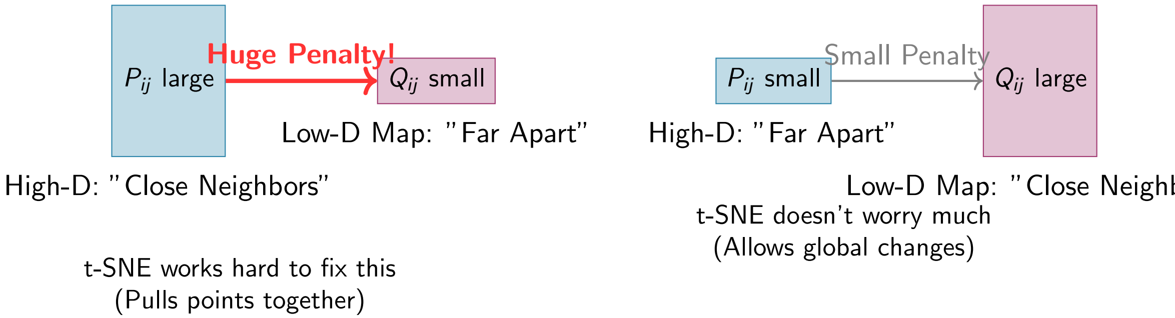
Making t-SNE Work

Key Components:

1. KL Divergence objective
2. Gradient descent
3. Momentum
4. Early exaggeration
5. Learning rate annealing

The Objective: The "Cost of a Bad Map"

The goal is to make the low-D map (Q) reflect the high-D reality (P). The KL Divergence measures the "cost" or "penalty" for every point where the map is wrong.

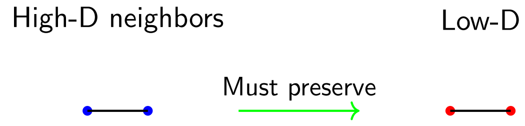


Insight: KL Divergence cares much more about keeping close points together than push

Asymmetry in KL Divergence

Situation	Penalty	Effect
Large p_{ij}, small q_{ij}	HIGH	Preserves local
Small p_{ij}, large q_{ij}	low	Allows global flex

Visual consequence:



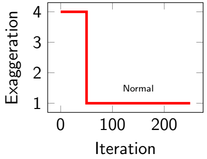
Computing the Gradient

Starting from: $C = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}$
Taking derivative w.r.t. y_i: $\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij}) \cdot F_{ij}$
where: $F_{ij} = \frac{(y_i - y_j)}{1 + ||y_i - y_j||^2}$
Interpretation: Weighted sum of forces from all points

Early Exaggeration Trick

Method:

- Multiply all p_{ij} by 4
- For first 50 iterations
- Creates tight clusters
- Separates clusters early



Why it works: Forces cluster formation before fine-tuning

The Complete Algorithm

```
1: **Input:** $X \in \mathbb{R}^{n \times d}$, perplexity
2: **Output:** $Y \in \mathbb{R}^{n \times 2}$
3:
4: Compute all $p_{ij}$ from $X$
5: Initialize $Y \sim \mathcal{N}(0, 10^{-4} I)$
6:
7: **for** iteration $t = 1$ to $T$ **do**
8:     Compute all $q_{ij}$ from $Y$
9:     Compute gradients $\frac{\partial C}{\partial Y}$
10:    Update with momentum:
11:        $Y^{(t)} = Y^{(t-1)} - \eta \frac{\partial C}{\partial Y} + \alpha \Delta Y^{(t-1)}$
12: **end for**
```

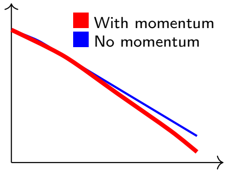
Momentum: Faster Convergence

Update rule:

$Y^{(t)} = Y^{(t-1)} - \eta\nabla + \alpha\Delta Y^{(t-1)}$

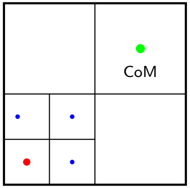
where:

- η: learning rate
- α: momentum (0.5→0.8)
- ΔY: previous update



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

Idea: Group distant points

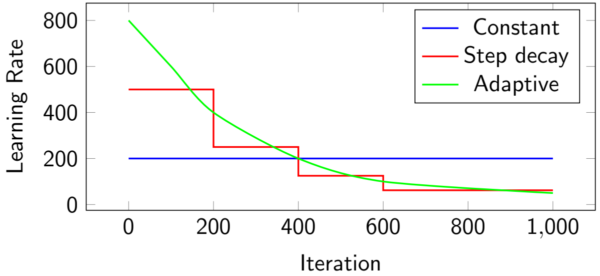


Approximation:

- Build quadtree
- Compute centers of mass
- If cell far: treat as one point
- Threshold: $\theta = 0.5$

Speedup: $100\times$ for $n = 10,000$

Learning Rate Strategies



Recommendation: Start high (500-1000), decrease if needed

Smart Initialization

Method	Pros	Cons
Random small	No bias	Slow start
PCA	Fast convergence	May bias
Previous run	Reproducible	Local minimum

Best practice: $Y_i \sim \mathcal{N}(0, 10^{-4} I)$
Small variance prevents early numerical issues

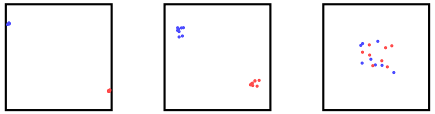
Hyperparameter Impact

Parameter	**Low**	**Default**	**High**
Perplexity	5-15	30	50-100
Learning rate	10-100	200	500-1000
Iterations	250	1000	5000
Momentum	0.5	0.8	0.9
Early exag.	4	12	20

Grid search often needed for optimal results

Perplexity: Detailed Effects

$\text{Perp} = 5$ $\text{Perp} = 30$ $\text{Perp} = 100$



- **Too low:** Breaks clusters into fragments
- **Too high:** Merges distinct clusters
- **Sweet spot:** Usually 5-50, dataset dependent

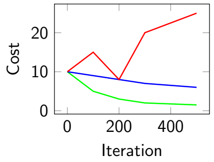
Learning Rate: Finding Balance

Too low ($\eta < 10$):

- Stuck in bad minimum
- Slow convergence
- Poor separation

Too high ($\eta > 1000$):

- Points explode
- Oscillations
- Never converges



t-SNE Troubleshooting Guide

Points explode/diverge? → Learning rate too high → Reduce η by $10\times$

Clusters won't separate? → Check iterations
Check perplexity
Check learning rate → Run $2000+$ iter
Try perp $=$ 15, 30
Increase to 200-500

See "islands" of points? → Perplexity too low → Minimum perp $=$ 15

Everything merges? → Perplexity too high → Maximum perp $=$ n/3

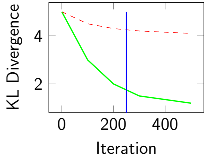
Monitoring Convergence

Watch for:

- KL divergence decrease
- Gradient norm $\rightarrow 0$
- Stable embedding

Warning signs:

- Increasing cost
- Points at infinity
- Oscillations



Interactive Parameter Exploration

Recommended workflow:

1. Start with defaults (perp=30, lr=200)
2. Run 5 times with different seeds
3. If inconsistent: adjust perplexity
4. If slow: increase learning rate
5. If unstable: decrease learning rate
6. Compare multiple perplexity values

Always run multiple times - t-SNE is stochastic!

Reading t-SNE Correctly

Critical questions:

- What can we trust?
- What is meaningless?
- How to validate?

What You Can Trust

Can Trust:

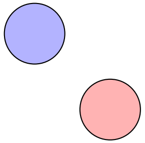
- Local neighborhoods
- Cluster existence
- Within-cluster structure
- Relative densities (roughly)

Cannot Trust:

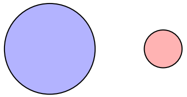
- Cluster sizes
- Between-cluster distances
- Global structure
- Absolute positions

t-SNE is for exploration, not measurement!

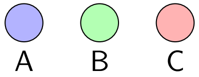
Interpretation Exercise: Real or Artifact?



Q1: Are these groups meaningfully different?



Q2: Is the left cluster actually bigger?



Q3: Is B intermediate between A and C?

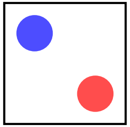
Answers

1. Check consistency: Run 5 times with different perplexities
2. NO! t-SNE doesn't preserve sizes. Check original data.
3. NO! Inter-cluster distances meaningless. B might be closer to A or C in high-D.

Key Test: Run 5 times. Pattern persists → likely real. Changes → artifact.

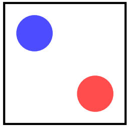
Cluster Separation: Real or Artifact?

Real



Consistent

Artifact

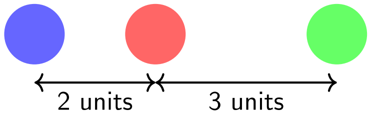


Random seed

Validation: Run multiple times, check if stable

Distance Interpretation Warnings

Between-cluster distances are meaningless!



These distances mean nothing!

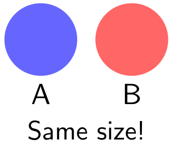
In high-D, all three might be equidistant

Cluster Size Non-Preservation

High-D Reality:

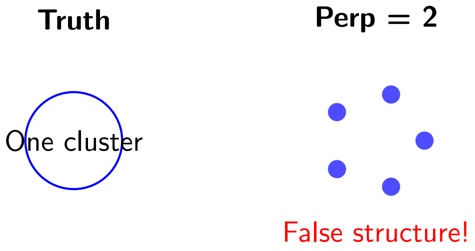
- Cluster A: 1000 points
- Cluster B: 100 points
- Ratio: 10:1

t-SNE Display:



Why: Optimization doesnt preserve density

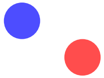
Pitfall: Perplexity Too Low



Solution: Increase perplexity to 15+

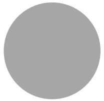
Pitfall: Perplexity Too High

Truth



Two clusters

Perp = 200



Lost structure!

Solution: Decrease perplexity to 30-50

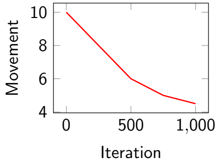
Pitfall: Non-Convergence

Symptoms:

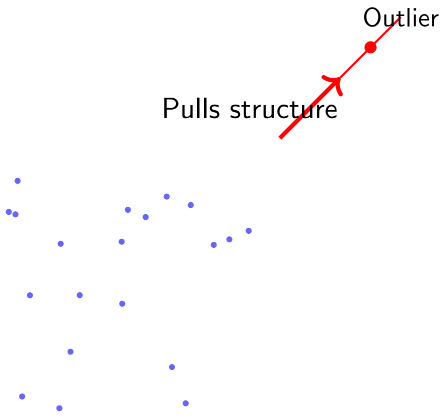
- Points still moving
- Cost oscillating
- Clusters not separated

Solutions:

- More iterations (2000+)
- Adjust learning rate
- Check for outliers



Outlier Effects



Solutions:

- Remove extreme outliers first
- Use robust preprocessing
- Check with and without outliers

Ensuring Reproducibility

1. **Set random seed**
    - For initialization
    - For algorithm
2. **Document parameters**
    - Perplexity
    - Learning rate
    - Iterations
3. **Save intermediate states**
    - Every 100 iterations
    - For debugging

Always report: "t-SNE with perp=X, lr=Y, iter=Z"

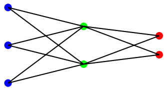
Advanced: Parametric t-SNE

Idea: Learn a function $f : \mathbb{R}^d \to \mathbb{R}^2$

Advantages:

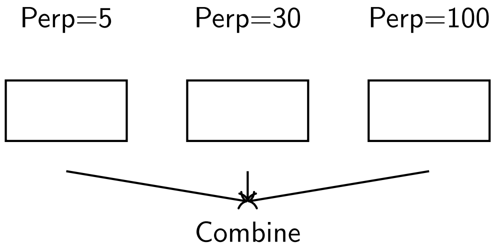
- Can embed new points
- Inverse mapping possible
- Interpretable features

Neural Network:



Advanced: Multi-Scale t-SNE

Use multiple perplexities simultaneously

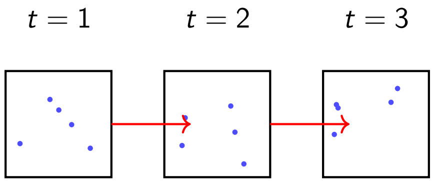


$p_{ij} = \sum_k w_k \cdot p_{ij}^{(perp_k)}$

Captures both local and global structure

Advanced: Dynamic t-SNE

For temporal data: Preserve structure over time



Add temporal regularization: $\lambda||Y_t - Y_{t-1}||^2$

t-SNE vs UMAP

Aspect	**t-SNE**	**UMAP**
Speed	$O(n \log n)$	$O(n^{1.14})$
Theory	Probability	Topology
Global structure	Poor	Better
Parameters	Perplexity	n_neighbors
Reproducibility	Random	More stable
Scalability	¡50K points	Millions

When to use each:

- t-SNE: Exploring clusters, publication figures
- UMAP: Large data, need global structure

When to Use Which Method?

Dataset Size & Method Choice

Size	Method
$<$ 1K	t-SNE
1K–50K	t-SNE + Barnes-Hut
50K-500K	UMAP
$>$ 500K	FIt-SNE or UMAP
$>$ 1M	RAPIDS-UMAP (GPU)

Code Comparison

```
# t-SNE (5 min)
tsne = TSNE(perplexity=30)
Y_tsne = tsne.fit_transform(X)

# UMAP (30 sec)
umap = UMAP(n_neighbors= 30)
Y_umap = umap.fit_transform(X)
```

Specific Scenarios

- **Publication figure (5K cells):** t-SNE – beautiful, trusted
- **Real-time embedding:** UMAP – supports transform()
- **Global structure:** UMAP – better topology
- **Rare cell types:** t-SNE – better local structure

Research Frontiers

1. **Initialization:** PaCMAP, TriMAP
2. **Speed:** FIt-SNE, openTSNE
3. **Theory:** Heavy-tailed embeddings
4. **Interpretability:** Attribution methods
5. **Uncertainty:** Probabilistic embeddings

Active research area: 100+ papers/year

From Theory to Code

Ready to implement t-SNE!

R: Rtsne Package

```
library(Rtsne)

# Prepare data
X <- as.matrix(your_data)

# Run t-SNE
tsne_out <- Rtsne(X,
                  dims = 2,
                  perplexity = 30,
                  theta = 0.5,
                  max_iter = 1000)

# Extract embedding
Y <- tsne_out$Y (# Plot plot(Y, col = labels))
```

Key Parameters Explained

Parameter	Meaning	Guidance
perplexity	Neighborhood size	5-50
learning_rate	Step size	10-1000
n_iter	Iterations	250-5000
theta	Barnes-Hut accuracy	0.5 default
metric	Distance function	euclidean
init	Initialization	pca or random

Most important: perplexity and learning_rate

Performance Optimization

1. **Preprocessing:**
   - PCA to 50D first
   - Normalize features
   - Remove duplicates
2. **Computation:**
   - Use float32 not float64
   - Enable multicore
   - GPU versions available
3. **Large datasets:**
   - Sample first, then embed
   - Use UMAP for ¿100K points
   - Consider parametric t-SNE

Hands-On Exercise: Understanding t-SNE Computations

Exercise: Compute t-SNE Components for 3 Points

Given points: $A(0,0)$, $B(2,0)$, $C(1,1)$ with perplexity $= 2$

Task 1: Compute P matrix

1. **Distances:**
   - $d_{AB} = 2$, $d_{AC} = \sqrt{2}$, $d_{BC} = \sqrt{2}$
2. **Find σ_A:** (binary search)
   - Target: $H(P_A) = \log(2) = 0.693$
   - Result: $\sigma_A \approx 1.2$
3. **Compute $p_{j|A}$:**
   - $p_{B|A} = 0.35$, $p_{C|A} = 0.65$
4. **Symmetrize:**
   - $p_{AB} = (p_{B|A} + p_{A|B})/2n$

Task 2: Compute Gradient

1. **Current positions:**
   - $y_A(0.1, 0.2)$, $y_B(0.8, 0.3)$, $y_C(0.4, 0.9)$
2. **Compute Q matrix:**
   - $q_{AB} = \frac{(1+||y_A - y_B||^2)^{-1}}{\sum_{k\neq l}(1+||y_k - y_l||^2)^{-1}}$
   - $q_{AB} = 0.28$, $q_{AC} = 0.31$, $q_{BC} = 0.41$
3. **Gradient for point A:**
   - $\frac{\partial C}{\partial y_A} = 4\sum_j (p_{Aj} - q_{Aj})F_{Aj}$
   - Result: $\nabla_A = (0.12, -0.08)$

Your turn: Calculate gradient for point B using the values above

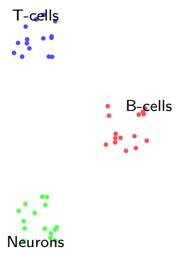
Case Study: Single-Cell RNA-seq

Dataset:

- 10,000 cells
- 20,000 genes
- Goal: Find cell types

Pipeline:

1. Filter genes (variance)
2. Log transform
3. PCA to 50D
4. t-SNE with perp=30



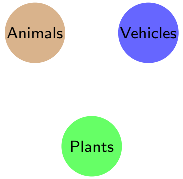
Case Study: ImageNet Features

Setup:

- CNN features (2048D)
- 50,000 images
- 1000 classes

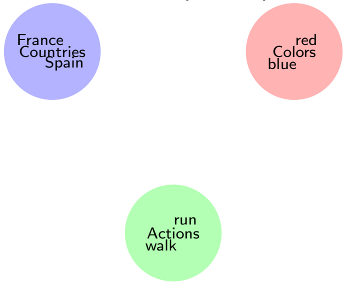
Results:

- Similar objects cluster
- Hierarchical structure
- Visual similarity preserved



Case Study: Word Embeddings

Word2Vec embeddings (300D) $\rightarrow$ t-SNE (2D)



Semantic relationships preserved!

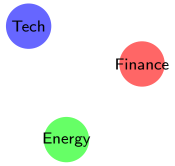
Case Study: Financial Time Series

Data:

- Stock returns
- 500 companies
- 252 trading days

Preprocessing:

- Correlation matrix
- t-SNE embedding
- Color by sector



Sectors naturally separate!

Apply to Your Research

t-SNE Checklist:

1. Is your data high-dimensional? (d ¿ 10)
2. Do you want to explore structure?
3. Is $n < 50,000$?
4. Can you validate clusters independently?

If yes to all $\rightarrow$ t-SNE is perfect!

Remember:

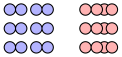
- Try multiple perplexities
- Run multiple times
- Validate findings

Same Data, Different Perplexities: Full Pipeline

Dataset: Iris (150 samples, 4 features)

Perplexity = 5

```
tsne = TSNE(
 perplexity=5,
random_state=42)
Y = tsne.fit_transform(X)
```

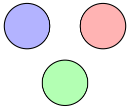


Fragmented

Over-interprets noise

Perplexity = 30

```
tsne = TSNE(
 perplexity=30,
random_state=42)
Y = tsne.fit_transform(X)
```

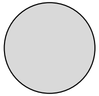


Well-separated

3 clear species

Perplexity = 100

```
tsne = TSNE(
 perplexity=100,
random_state=42)
Y = tsne.fit_transform(X)
```



Merged blob

Structure lost

Key: Perplexity controls the balance between local and global focus

Part 7: Synthesis

Key Takeaways

What you have mastered today

When to Use t-SNE

Use t-SNE:

- Exploring clusters
- Validating features
- Finding outliers
- Publication figures
- Quality over speed

Dont use t-SNE:

- Measuring distances
- $>$ 100K points
- Real-time analysis
- Definitive proof
- Production systems

t-SNE is for exploration and insight

The Interpretation Checklist

Before publishing t-SNE results:

1. ☐ Tried perplexity: 5, 15, 30, 50
2. ☐ Ran 5+ random initializations
3. ☐ Checked convergence (1000+ iterations)
4. ☐ Validated clusters independently
5. ☐ Stated all parameters clearly
6. ☐ Acknowledged limitations
7. ☐ Compared with PCA/other methods

Never interpret distances or sizes!

Resources for Deeper Study

Essential Papers:

- Original: van der Maaten & Hinton (2008)
- Barnes-Hut: van der Maaten (2014)
- Theory: Linderman & Steinerberger (2017)

Software:

- Python: scikit-learn, openTSNE
- R: Rtsne, tsne
- Fast: FIt-SNE, RAPIDS

Tutorials:

- Distill.pub interactive guide
- Google embedding projector

Discussion & Questions

Your Questions?

Theory?
Implementation?
Your data?

Thank you for your attention!
Now lets explore your data with t-SNE!