t-SNE Algorithm

t-Distributed Stochastic Neighbor Embedding

Non-linear dimensionality reduction for visualization

Two-Step Process

1. Compute pairwise similarities in high-dimensional space
2. Optimize low-dimensional embedding to match similarities

Key Parameter

Perplexity

`Range: 5 - 50`

Balances local vs global structure

Primary Use

- 2D/3D visualization
- NOT for general dim reduction

Key Features

- **Preserves local structure:** Similar points stay close
- **Student t-distribution:** Avoids crowding problem
- **Stochastic:** Different runs produce different results

Limitations

- Computationally expensive: $O(n^2)$
- Best with <10k samples
- Non-deterministic results
- Sensitive to hyperparameters

Step-by-Step Example: MNIST Digits

3D → 2D Visualization Process

1 Input: High-Dimensional Data

Data Preparation

MNIST handwritten digit data (simplified example)

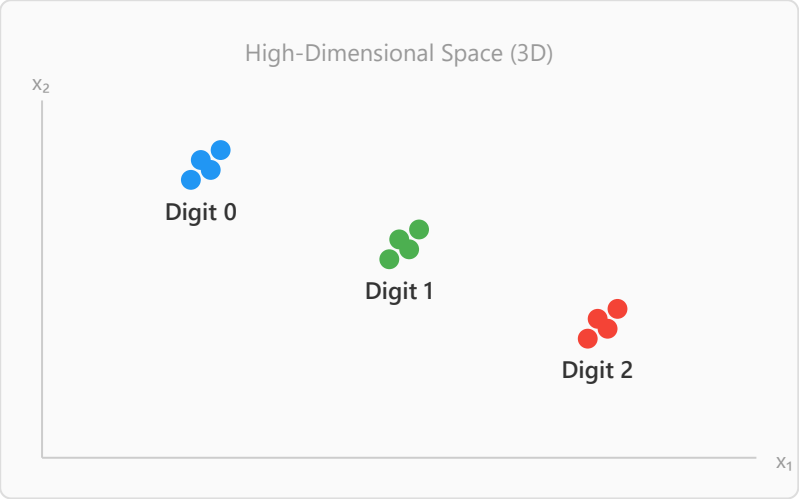
Original dimensions: 784D (28×28 pixels) → simplified to 3D

Number of samples: 12 (4 samples each for digits 0, 1, 2)

$$X = [x_1, x_2, \ldots, x_{12}]$$
$$x_i \in \mathbb{R}^3$$

Sample	Label	x_1	x_2	x_3
1	0	2.1	3.4	1.2
2	0	2.3	3.1	1.5
3	0	1.9	3.6	1.0
4	0	2.4	3.3	1.3
...	...	...	...	...

3D Space Visualization



2 Compute High-Dimensional Similarities

Similarity Calculation (Gaussian Distribution)

Compute conditional probabilities between each pair of points

$$p(j|i) = \exp(-||x_i - x_j||^2 / 2\sigma_i^2) / \Sigma_k \exp(-||x_i - x_k||^2 / 2\sigma_i^2)$$

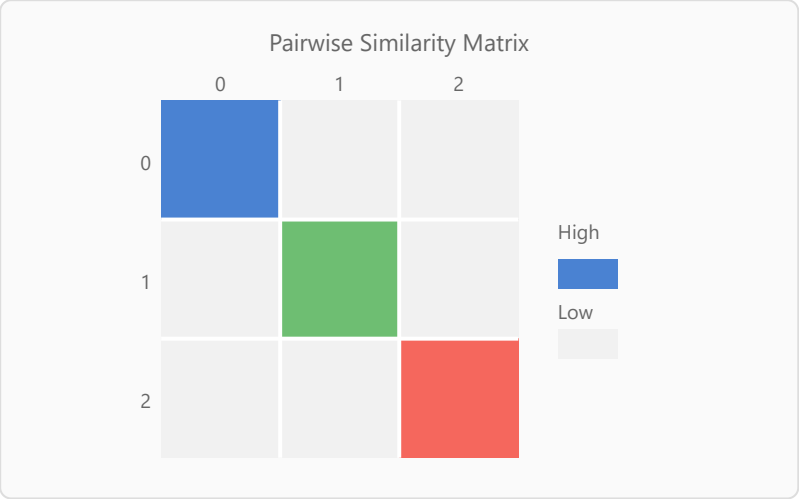
$$p_{ij} = (p(j|i) + p(i|j)) / 2n$$

Perplexity = 5 (determines number of neighbors)

Intermediate result: 12×12 similarity matrix P generated
- Same class: $p_{ij} \approx 0.08 \sim 0.12$
- Different class: $p_{ij} \approx 0.001 \sim 0.005$

	S1	S2	S5	S9
S1(0)	-	0.095	0.003	0.001
S2(0)	0.095	-	0.002	0.001
S5(1)	0.003	0.002	-	0.089
S9(2)	0.001	0.001	0.089	-

Similarity Matrix Heatmap



3 Initialize Low-Dimensional Embedding

Random Initialization in 2D Space

Randomly place points in the target dimension (2D)

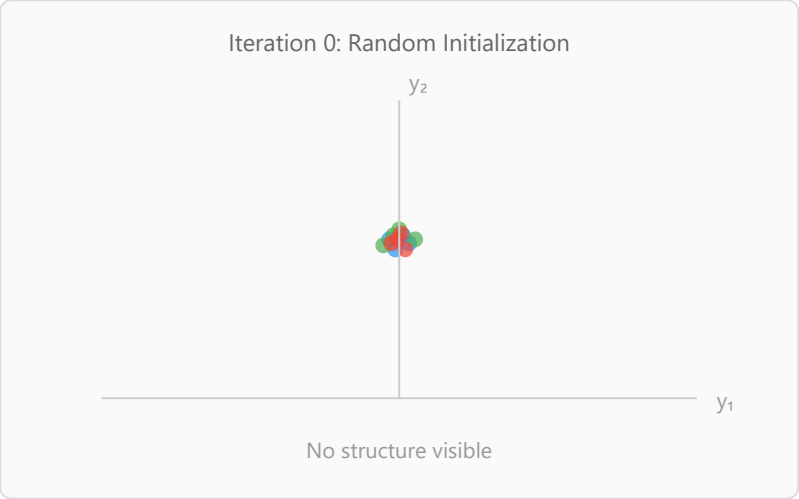
$$Y^{(0)} = [y_1^{(0)}, y_2^{(0)}, \ldots, y_{12}^{(0)}]$$
$$y_i^{(0)} \sim N(0, 0.0001 \cdot I)$$

Initial state:

- All points clustered near the origin
- No class structure
- No meaningful patterns yet

Iteration 0: Completely random placement
Cost = Very high (≈ 8.5)

Initial Random Embedding



↓

4 Iterative Optimization (Gradient Descent)

Matching Similarities in Low-Dimensional Space

Compute low-dimensional similarities using Student t-distribution

$$q_{ij} = (1 + ||y_i - y_j||^2)^{-1} / \Sigma_{kl}(1 + ||y_k - y_l||^2)^{-1}$$

$$Cost = KL(P||Q) = \Sigma_{ij}\ p_{ij}\ \log(p_{ij}/q_{ij})$$

Gradient:

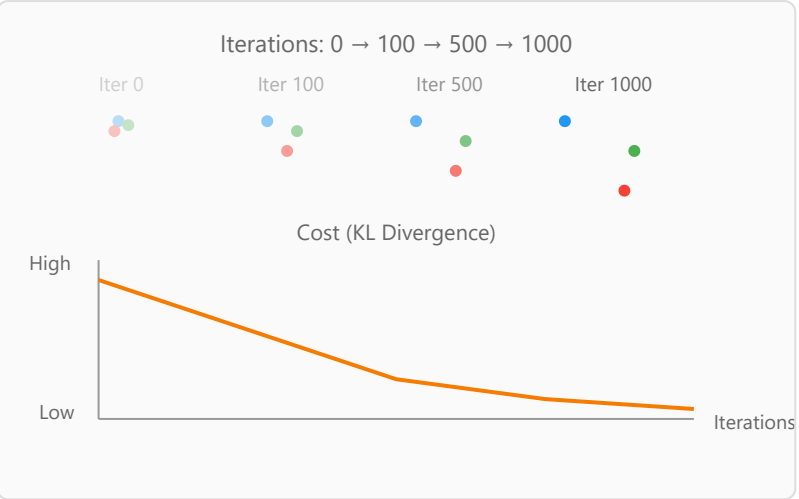
$$\partial C/\partial y_i = 4\Sigma_j\ (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}$$

Progress:

Iteration	Cost (KL)	Status
0	8.523	Random
100	3.214	Clusters forming
500	0.892	Well separated
1000	0.435	Converged

Learning rate: Early exaggeration (η=200) → Fine-tuning (η=50)

Optimization Progress



5 Final 2D Embedding

Final Visualization Result

Low-dimensional embedding after convergence

Result Analysis:
✓ Same classes clustered together
✓ Different classes clearly separated
✓ Local structure preserved
✓ Visually interpretable

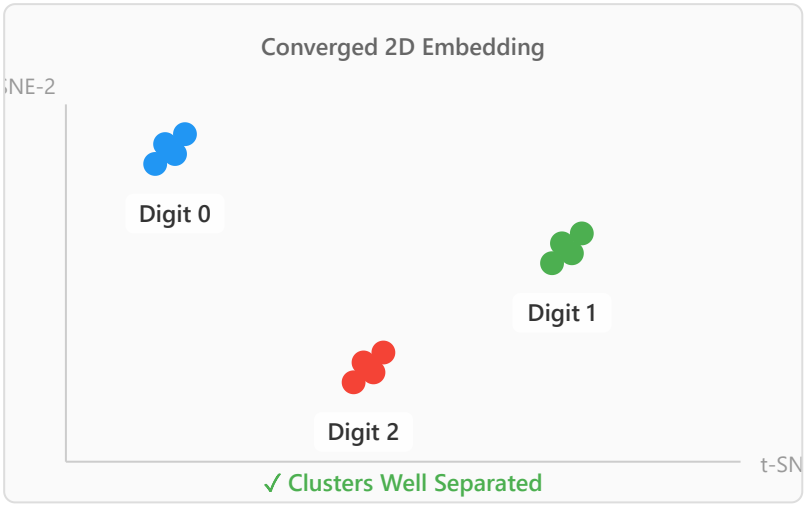
Performance Metrics:

- Final Cost: 0.435
- Convergence time: ~15s
- Total Iterations: 1000

Key Characteristics:

- Inter-cluster distances are meaningless
- Cluster sizes are also meaningless
- Only local structure is reliable
- Different results on re-runs possible

Final t-SNE Visualization



Key Takeaways

✅ What t-SNE Preserves

- Close neighbor relationships

❌ What t-SNE Does NOT Preserve

- Absolute distances

- Local cluster structure
- Density within same class

- Relative distances between clusters
- Cluster sizes/densities

Practical Tips:

- Adjust Perplexity: Small values (5-10) emphasize local structure, large values (30-50) consider global structure
- Run multiple times to verify stable patterns
- For >10,000 samples, first reduce to 50D with PCA, then apply t-SNE
- Use only for visualization purposes, not suitable for classification/regression preprocessing