t-SNE: A Visual Deep Dive

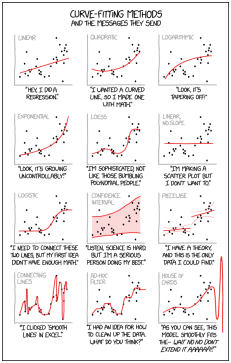
Understanding Neighbor Embedding Without Formulas

Prof. Dr. Jörg Osterrieder

Methods and Algorithms — MSc Data Science

Spring 2026

When You Need to Keep Neighbors Close



You have 50 features.

You need 2 axes.

But you also need to keep the neighborhoods intact.

How?

XKCD #2048 by Randall Munroe (CC BY-NC 2.5)

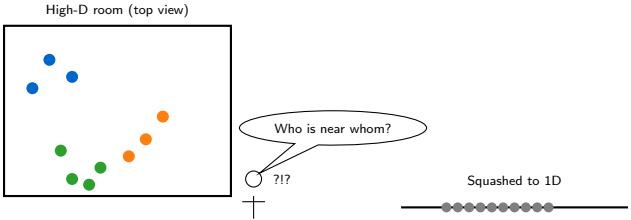
What You Will Learn Today

1. Explain what t-SNE does and why it exists
2. Describe how t-SNE preserves neighborhoods (visually)
3. Recognize when t-SNE results are trustworthy vs misleading

---

Everything is explained with pictures, analogies, and visual diagrams — zero formulas.

The Problem: Who Is Near Whom?



Squashing to fewer dimensions loses the neighborhood structure.
Dimensionality reduction must preserve which points are neighbors.

---

t-SNE was invented specifically to solve this: preserve local neighborhoods when flattening to 2D.

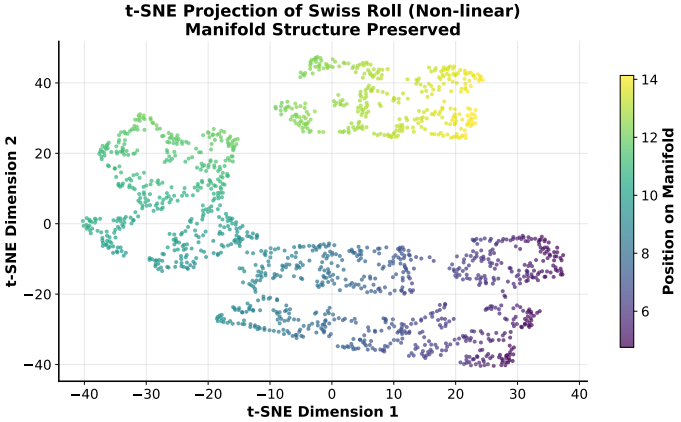
What t-SNE Does (One Sentence)

Aspect	Linear methods	t-SNE
Preserves	global distances	local neighborhoods
Speed	fast	slow
Best for	preprocessing	visualization

t-SNE keeps nearby points nearby. That is the entire idea.

van der Maaten & Hinton, 2008. The "t" is for Student's t-distribution; "SNE" = Stochastic Neighbor Embedding.

t-SNE Unrolls What Linear Methods Cannot

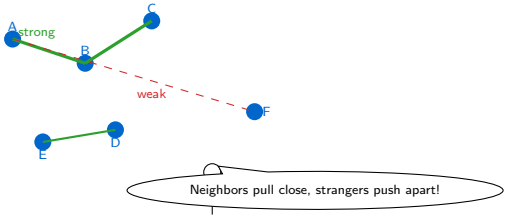


https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L05_PCA_tSNE/05b_tsne_swiss_roll

- Data spirals in 3D like a rolled carpet
- t-SNE preserves the neighborhood order along the spiral

Compare with PCA on the same data: PCA smashes the spiral flat, mixing distant points.

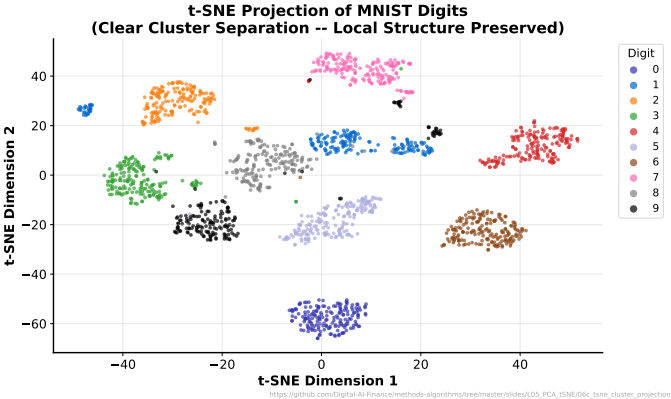
How t-SNE Works: A Spring System



t-SNE connects every pair of points with a spring. Neighbors get strong springs. Strangers get weak ones.

The "springs" are probability-based: high probability = strong pull, low probability = weak push.

The Payoff: Clean Cluster Separation

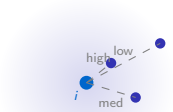


- Each color is a different group in the original high-D data
- t-SNE pulls clusters apart so you can see them
- Warning: distances BETWEEN clusters are not meaningful

t-SNE excels at revealing IF clusters exist. It does NOT tell you how far apart they are.

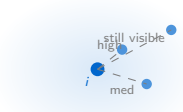
Two Different Lenses for Measuring Similarity

In High-D: Gaussian Lens



Narrow bubble

In 2D: Student-t Lens

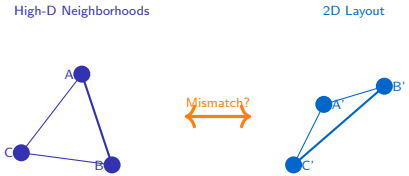


heavy tails = room to spread

- High-D uses a narrow Gaussian bubble — only close neighbors matter
- 2D uses a wider Student-t bubble — far points get pushed much farther apart

The wider t-distribution lens in 2D solves the "crowding problem": it gives clusters room to breathe.

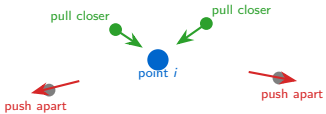
The Goal: Make Both Pictures Match



- Left = neighborhoods in the original data. Right = layout in 2D.
- t-SNE moves points until the 2D layout matches the original neighborhoods as closely as possible.

Technically this is KL divergence minimization, but visually it is just "make these two pictures agree."

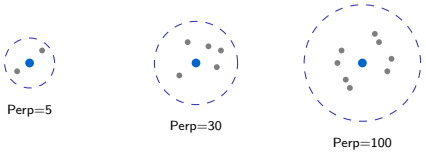
The Engine: Attract Neighbors, Repel Strangers

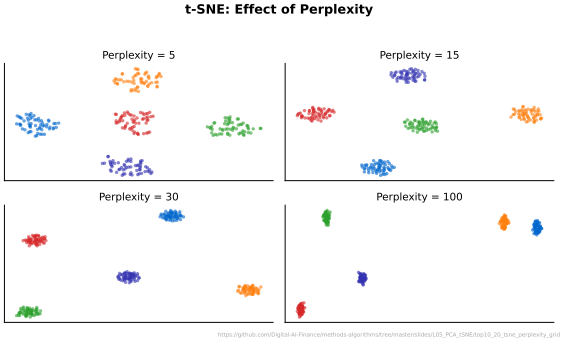


- Green arrows: neighbors that are too far apart get pulled closer
- Red arrows: non-neighbors that are too close get pushed apart
- Every point feels these forces from every other point, every iteration

This is gradient descent in disguise: the forces ARE the gradient. Hundreds of iterations until equilibrium.

The One Knob You Must Tune: Perplexity

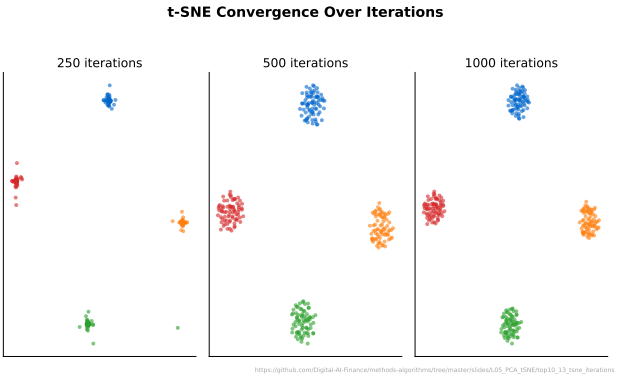




- Perplexity controls how many neighbors each point considers. Always try multiple values.

Default perplexity = 30 in sklearn. Try 5, 30, and 100 to see if your clusters are robust.

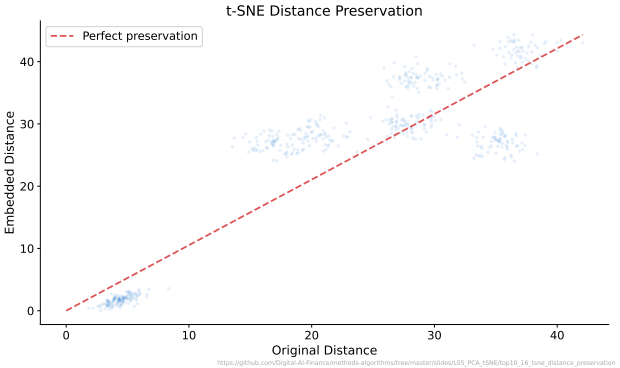
Watch It Converge — and Three Gotchas



1. ! Distances between clusters mean nothing — only within-cluster structure is real
2. ! Different random seeds produce different layouts — run it multiple times
3. ! Slow on big data — use PCA to 50 dims first, then t-SNE

If your clusters disappear when you change the random seed, they are probably not real.

What t-SNE Preserves — and Where It Shines



Finance Applications

- Fraud detection: t-SNE reveals anomalous transaction clusters
- Credit risk: visualize borrower segments in high-dimensional feature space
- Market regimes: identify hidden market states from return data

t-SNE is for exploration and visualization. Never feed t-SNE coordinates into a classifier.

Try It Yourself: 4 Lines of Python

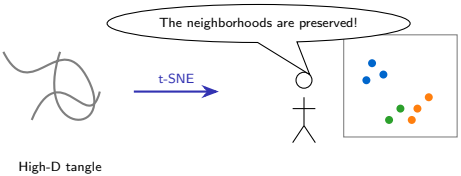
t-SNE in Python

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, perplexity=30)
X_2d = tsne.fit_transform(X)
plt.scatter(X_2d[:, 0], X_2d[:, 1])
```

That is it. Four lines. sklearn does the heavy lifting.

Always scale your data first with StandardScaler. For large datasets, reduce to 50 dims with PCA before t-SNE.

The Cartographer's Map



t-SNE turns high-dimensional tangles into readable maps — if you tune it right.

---

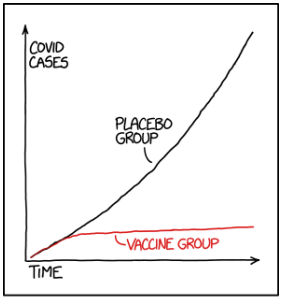
t-SNE: your best tool for seeing if clusters exist in high-dimensional data.

Three Things to Remember

1. t-SNE preserves local neighborhoods when flattening high-D data to 2D
2. It works like a spring system: neighbors attract, strangers repel
3. Always tune perplexity and run multiple seeds before trusting the layout

For preprocessing and speed, use PCA first. For visualization and cluster discovery, use t-SNE.

Go Find the Hidden Clusters



XKCD #2400 by Randall Munroe (CC BY-NC 2.5). Next: try the Jupyter notebook with real financial data!