Manifold learning

t-SNE, UMAP, LLE, etc

Readings

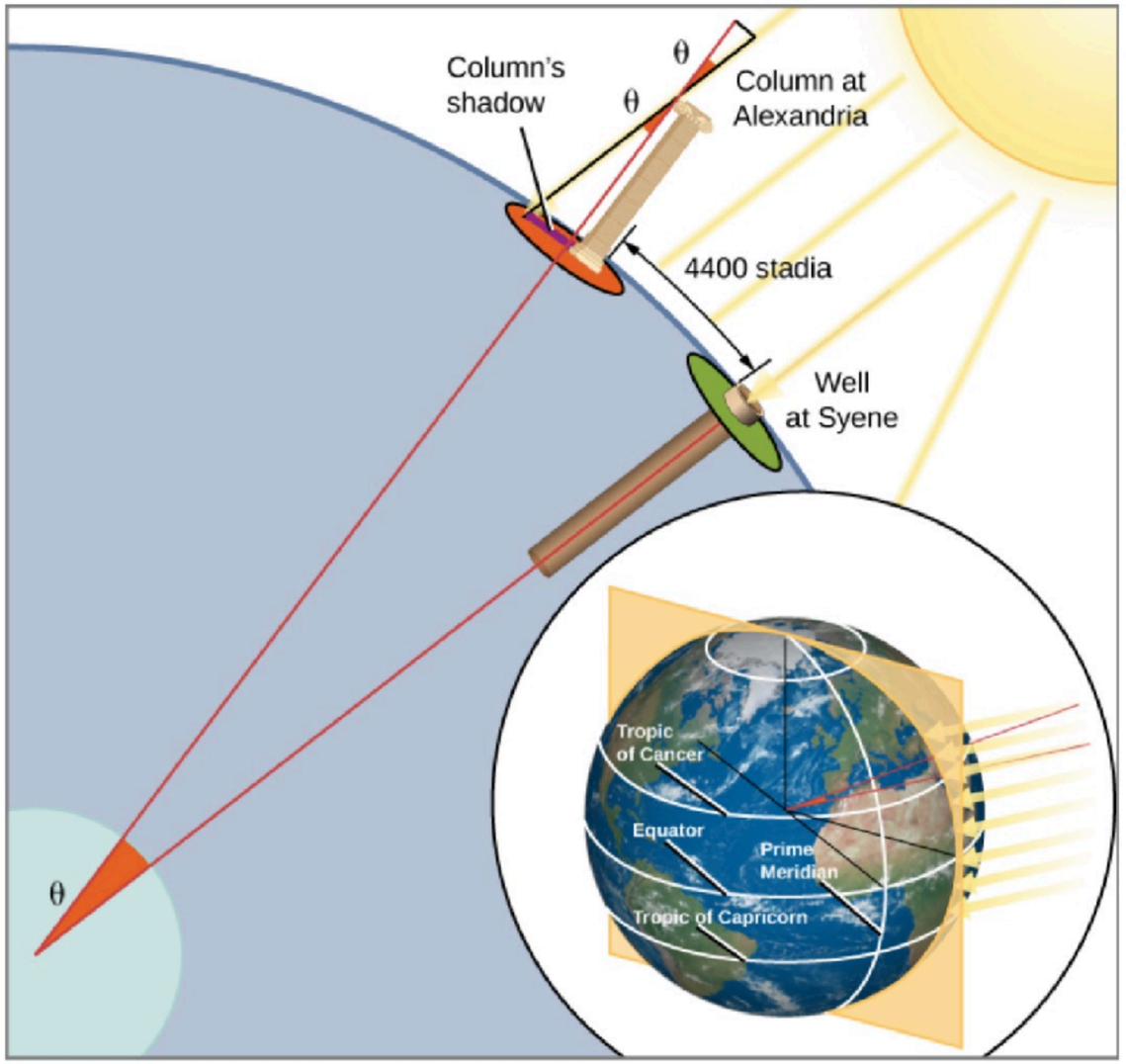
- https://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
- https://distill.pub/2016/misread-tsne/
- https://pair-code.github.io/understanding-umap/index.html
- https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

Manifolds

- A manifold is a topological space that is locally Euclidean
- Consider the existence of FLAT EARTHERS
- On the small scales that we see daily the Earth does indeed look flat.
- Any object that is nearly "flat" on small scales is a manifold, and so manifolds constitute a generalization of objects we could live on :)

Manifolds
SIDE BAR

- A manifold is a topological space that is locally Euclidean
- Flat earth is not some ancient belief… the ancient Greeks and others were generally aware of the curvature of the Earth.
- Eratosthenes of Cyrene (3rd century BCE) was chief librarian at the Library of Alexandria & calculated the Earth's circumference with an error of ~2%



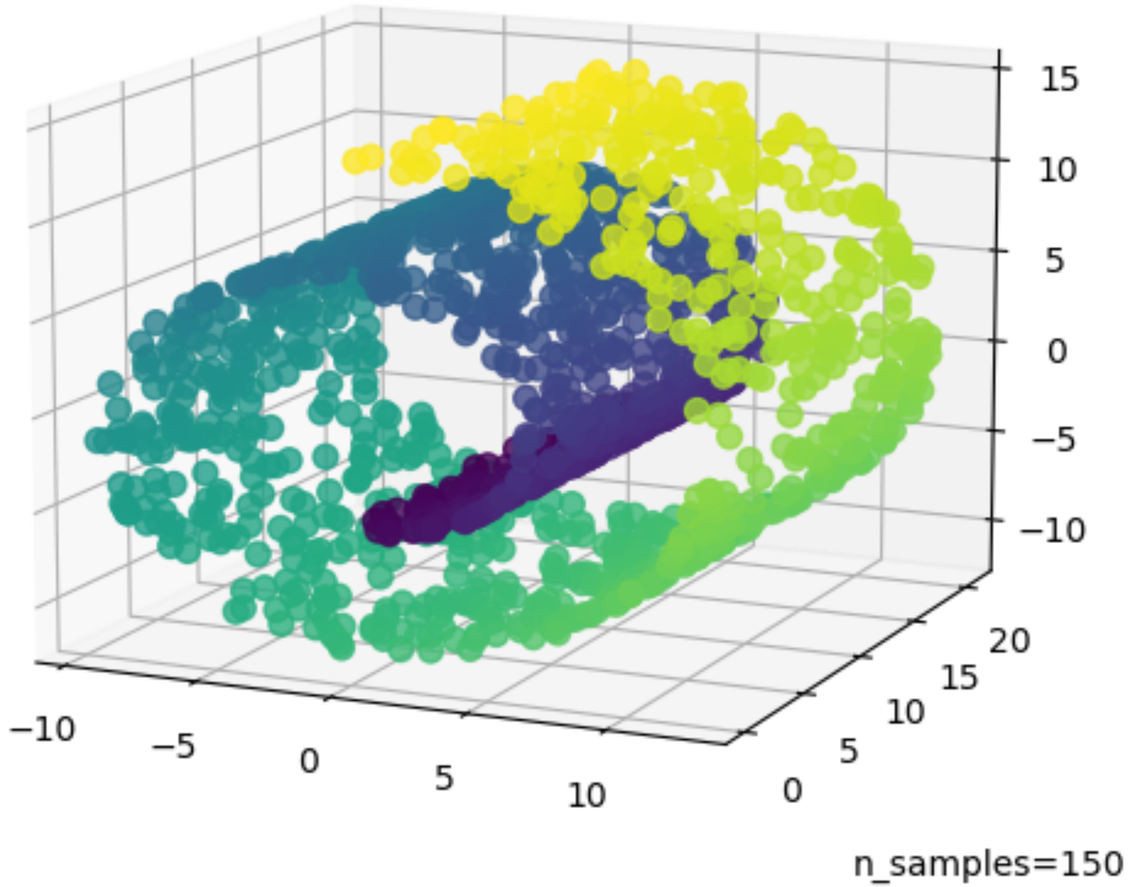
Topology
SIDE BAR

- One of the goals of topology is to find ways of distinguishing manifolds, e.g….
  - A circle is topologically the same as any closed loop
  - The surface of a coffee mug with a handle is topologically the same as the surface of the donut

Manifold learning

- Manifold learning is an approach to dimensionality reduction where the problem is only locally linear/Euclidean
- Algorithms for this task are based on the idea that the dimensionality of many data sets is only artificially high if you allow curved manifolds

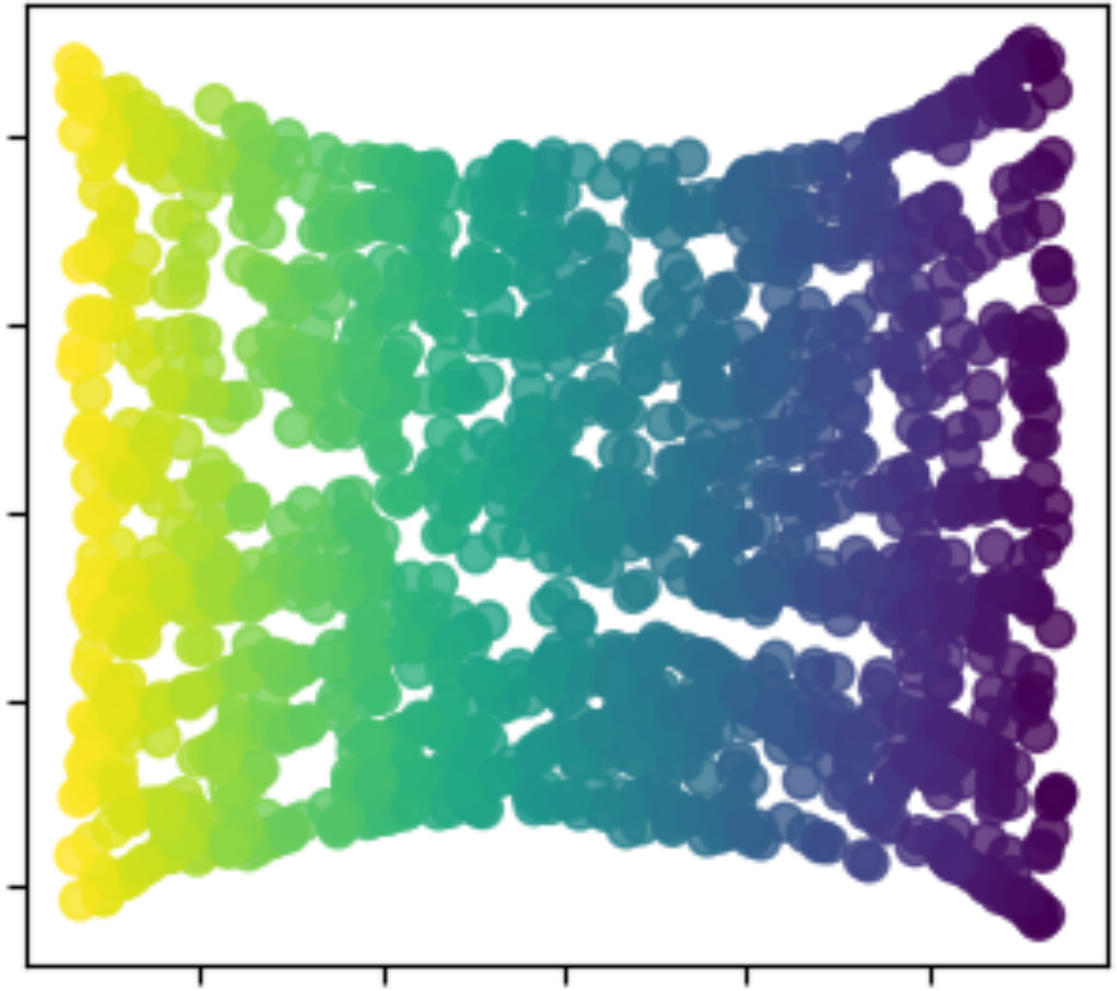
Swiss roll dataset



Manifold learning

- Manifold learning is an approach to dimensionality reduction where the problem is only locally linear/Euclidean
- Algorithms for this task are based on the idea that the dimensionality of many data sets is only artificially high if you allow curved manifolds

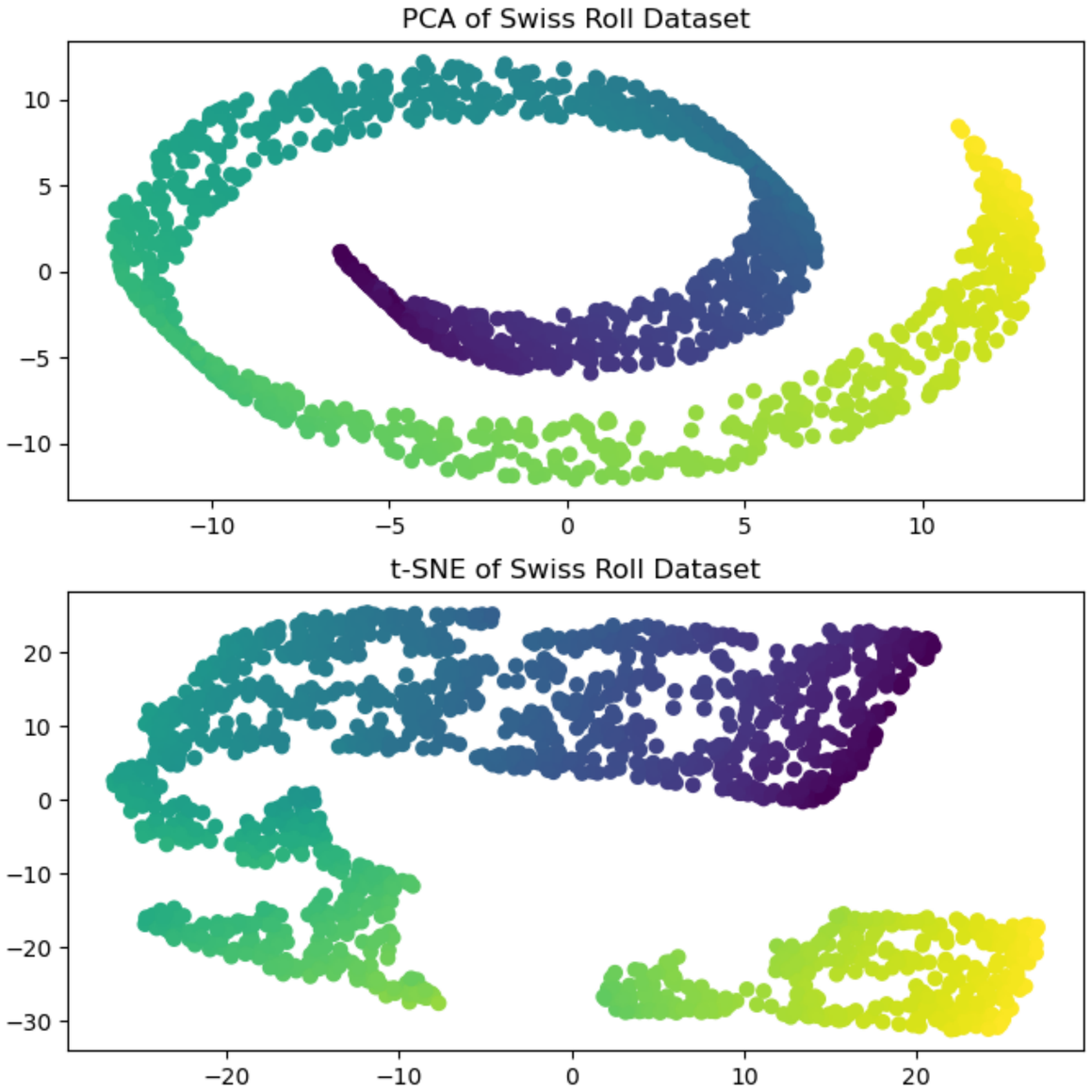
"Idealized" 2D manifold for Swiss Roll

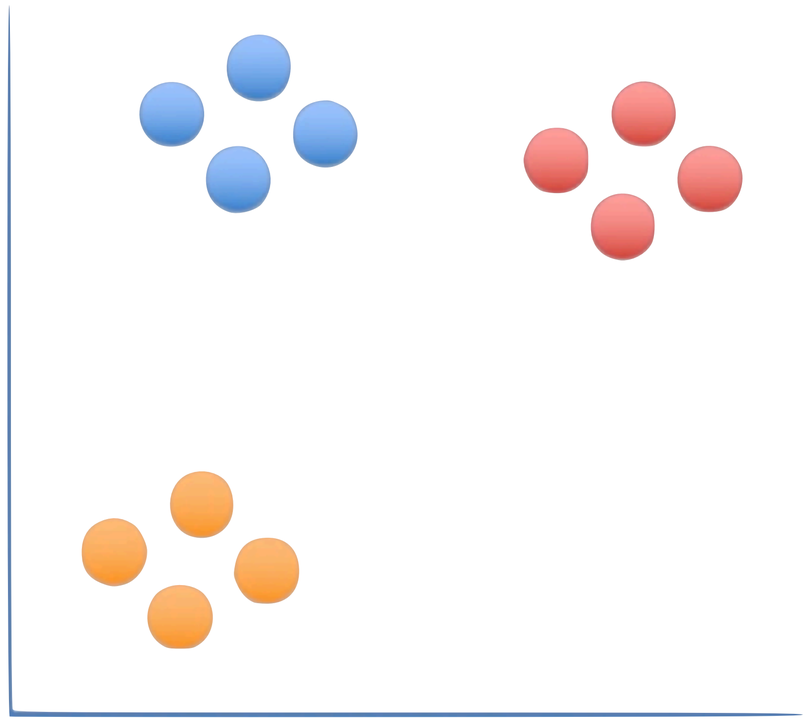


t-SNE

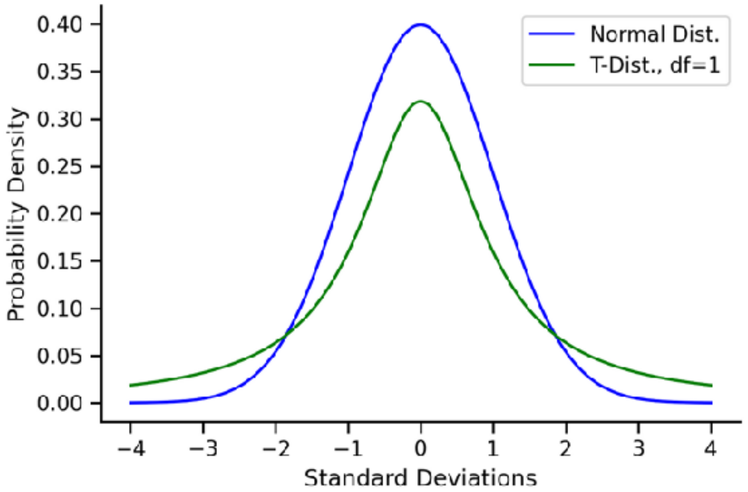
t-distributed Stochastic Neighbor Embedding

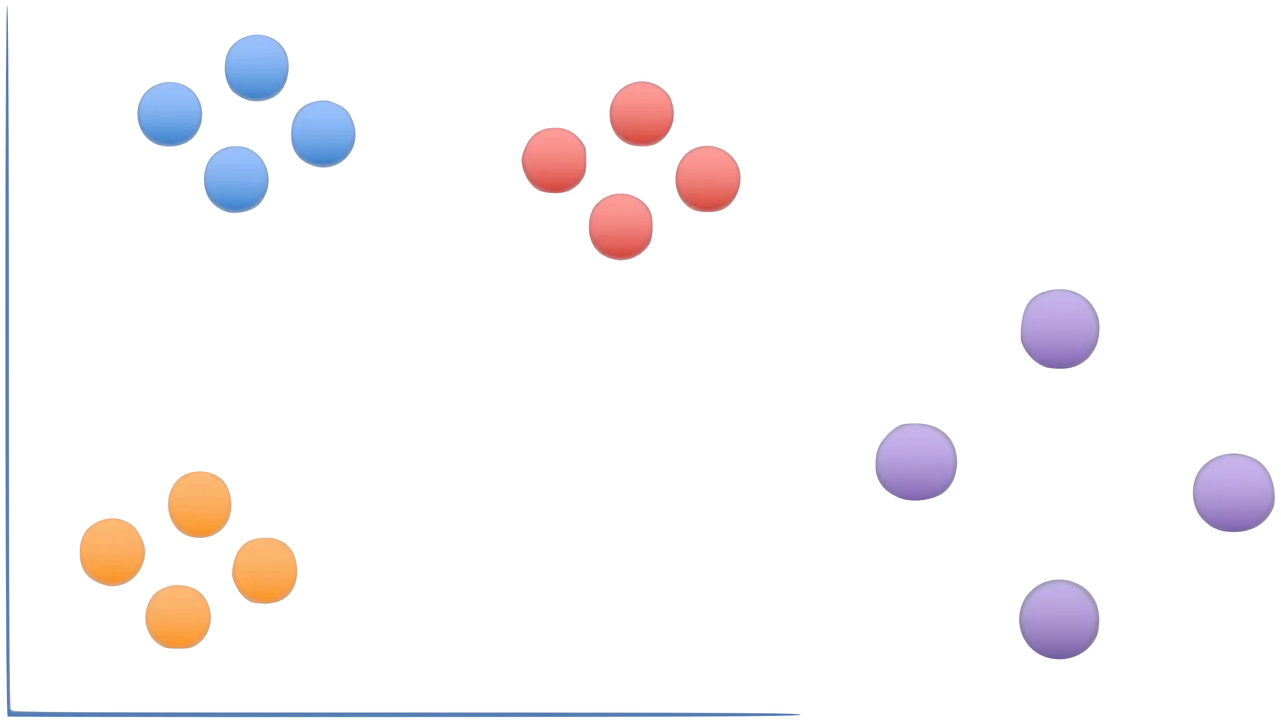
- PCA tries to find a global structure
  - Can lead to local inconsistencies in a subspace…far away points can become nearest neighbors
- t-SNE tries to preserve local structure
  - “For high-dimensional data that lies on or near a low-dimensional, non-linear manifold it is usually more important to keep the low-dimensional representations of very similar datapoints close together, which is typically not possible with a linear mapping”





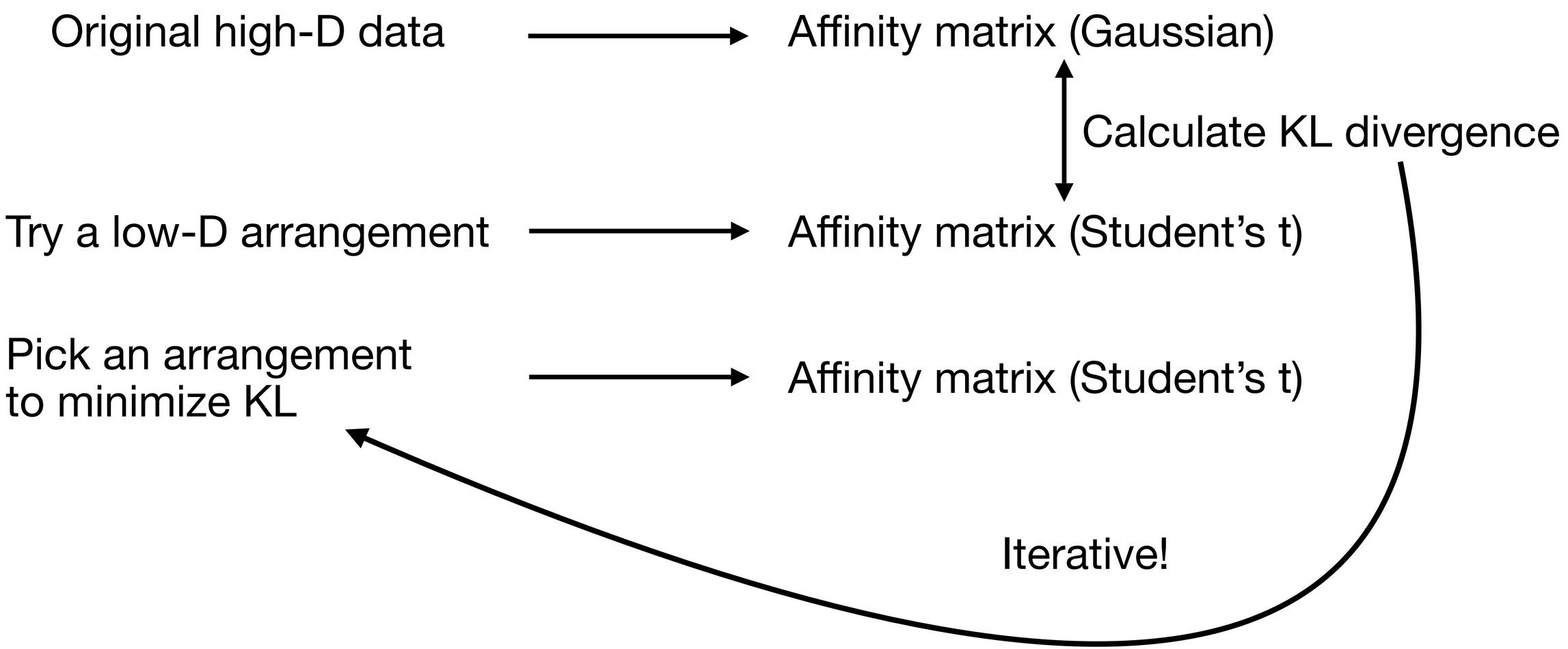
https://www.youtube.com/watch?v=NEaUSP4YerM





https://www.youtube.com/watch?v=NEaUSP4YerM

t-SNE concepts



https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf

t-SNE algorithm

points in the embedding. Assume we are given a data set of (high-dimensional) input objects $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and a function $d(\mathbf{x}_i, \mathbf{x}_j)$ that computes a distance between a pair of objects, *e.g.*, the Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|$. Our aim is to learn an s-dimensional embedding in which each object is represented by a point, $\mathcal{E} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ with $\mathbf{y}_i \in \mathbb{R}^s$ (typical values for s are 2 or 3). To this end, t-SNE defines joint probabilities

$$p_{j|i} = \frac{\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-d(\mathbf{x}_i, \mathbf{x}_k)^2/2\sigma_i^2)}, \qquad p_{i|i} = 0$$

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

https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf

t-SNE algorithm

In the s-dimensional embedding $\mathcal{E}$, the similarities between two points $\mathbf{y}_i$ and $\mathbf{y}_j$ (*i.e.*, the low-dimensional models of $\mathbf{x}_i$ and $\mathbf{x}_j$) are measured using a normalized heavy-tailed kernel. Specifically, the embedding similarity q_{ij} between the two points $\mathbf{y}_i$ and $\mathbf{y}_j$ is computed as a normalized Student-t kernel with a single degree of freedom:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}, \qquad q_{ii} = 0. \tag{3}$$

The heavy tails of the normalized Student-t kernel allow dissimilar input objects $\mathbf{x}_i$ and $\mathbf{x}_j$ to be modeled by low-dimensional counterparts $\mathbf{y}_i$ and $\mathbf{y}_j$ that are too far apart. This is desirable because it creates more space to accurately model the small pairwise distances (*i.e.*, the local data structure) in the low-dimensional embedding.

The locations of the embedding points $\mathbf{y}_i$ are determined by minimizing the Kullback-Leibler divergence between the joint distributions P and Q:

$$C(\mathcal{E}) = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{4}$$

https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf

t-SNE algorithm

Due to the asymmetry of the Kullback-Leibler divergence, the objective function focuses on modeling high values of p_{ij} (similar objects) by high values of q_{ij} (nearby points in the embedding space). The objective function is non-convex in the embedding $\mathcal{E}$. It is typically minimized by descending along the gradient:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) q_{ij} Z (\mathbf{y}_i - \mathbf{y}_j), \tag{5}$$

where we defined the normalization term $Z = \sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}$.

It is straightforward to see that the evaluation of the joint distributions P and Q is $\mathcal{O}(N^2)$, because both distributions involve a normalization term that sum over all $N(N-1)$ pairs of unique objects. Since t-SNE scales quadratically in the number of objects N, its applicability is limited to data sets with only a few thousand input objects; beyond that, learning becomes too slow to be practical (and the memory requirements become too large).

https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf

t-SNE algorithm

One last thing we have not yet mentioned how to set the bandwidths σ_i for the Gaussian kernels centered over each data point in the input space. It is unlikely that one single value of σ_i is optimal for all data points because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Perplexity is defined as

$$\text{Perplexity}(\mathbf{p}_i) = 2^{H(\mathbf{p}_i)}$$

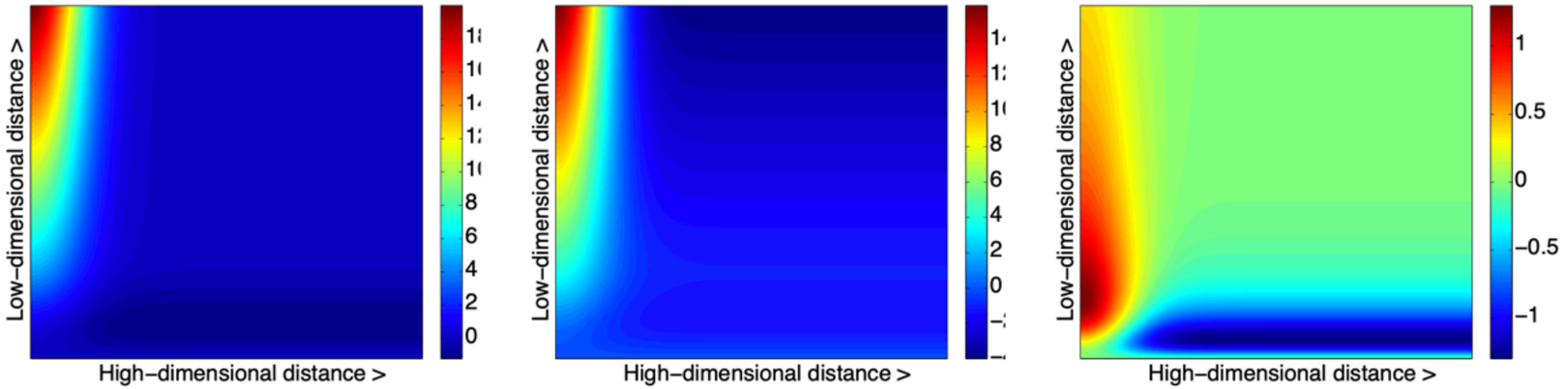
where H is the Shannon entropy of a discrete distribution

$$H(\mathbf{p}_i) = -\sum_i p_{j|i} \log_2(p_{j|i})$$

Perplexity can be thought of as a continuous analogue to the k nearest neighbours, to which t-SNE will attempt to preserve distances. More concretely, the bandwidths σ_i are set such that each Gaussian kernel fits k nearest neighbors within one standard deviation of the probability density.

- Low Perplexity is small σ_i
- High Perplexity is large σ_i

Mismatched Tails can Compensate for Mismatched Dimensionalities



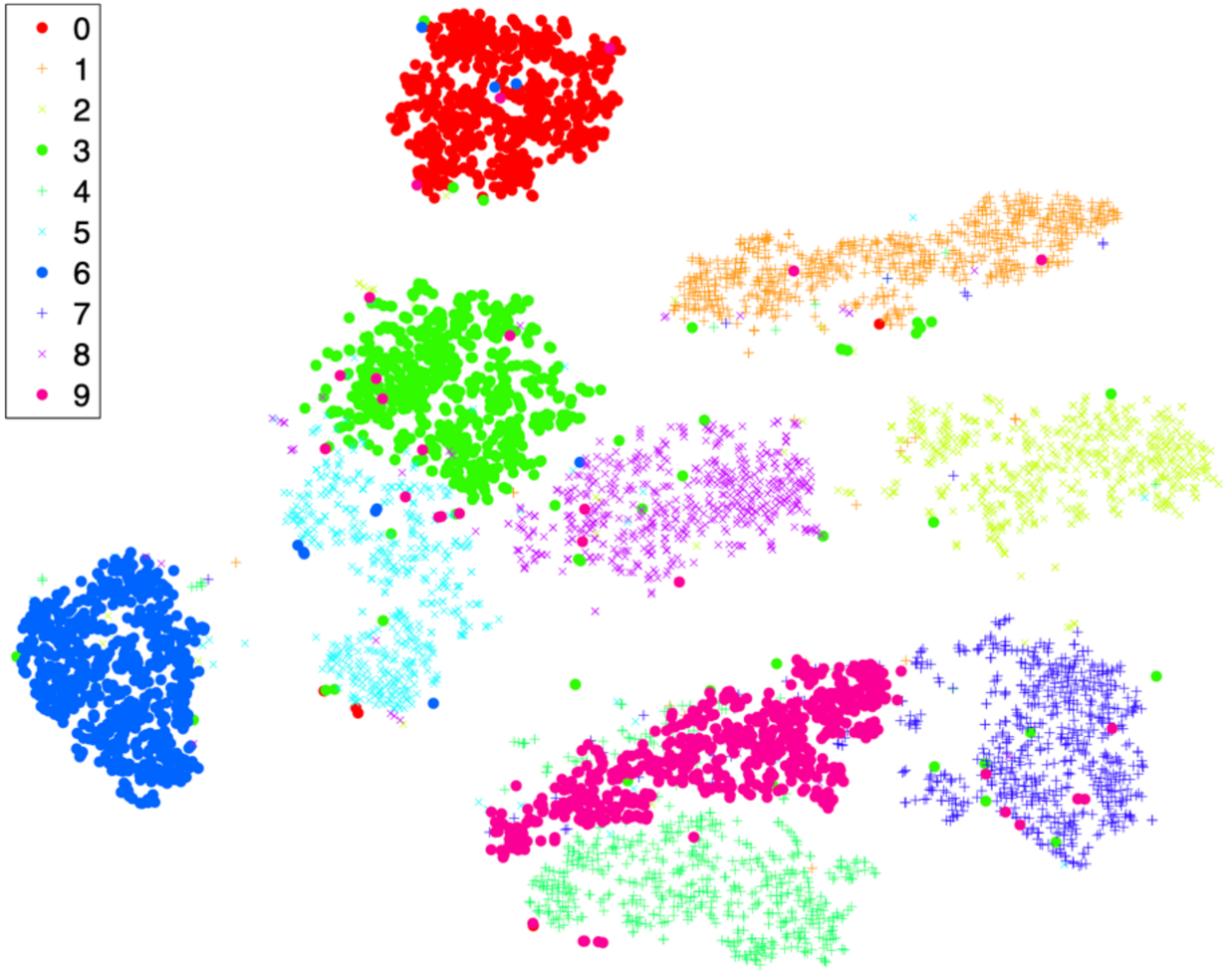
(a) Gradient of SNE. (b) Gradient of UNI-SNE. (c) Gradient of t-SNE.

Figure 1: Gradients of three types of SNE as a function of the pairwise Euclidean distance between two points in the high-dimensional and the pairwise distance between the points in the low-dimensional data representation.

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for *t=1* **to** T **do**
        compute low-dimensional affinities q_{ij} (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
    end
end

MNIST



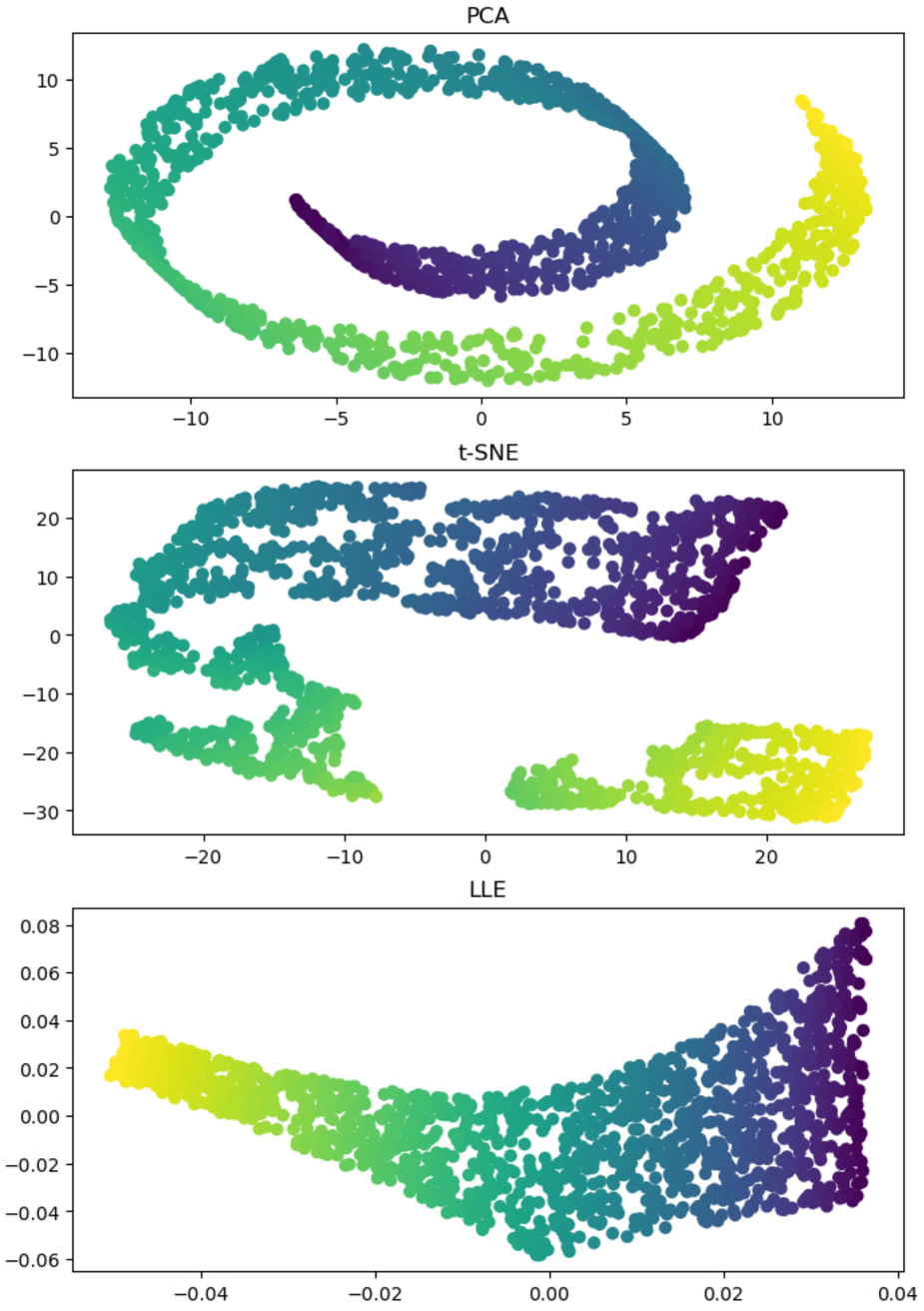
(a) Visualization by t-SNE.

t-SNE limitations

- Computationally expensive!
  - $\mathcal{O}(N^2)$ because of affinity matrix calculations… and worse than that because we have to do this iteratively on the lower-d embedding
  - Barnes-Hut approximation to gradient descent on KL (a spanning tree style algorithm) compares points to “cells” in $\mathcal{O}(N \log N)$.
    - Can only go up to 3D! Only works with dense input data!
    - Can go up to ~10^5 to 10^6 samples
- Only has .fit_transform() because there’s no way to .transform() separately from a .fit()! You’d have to redo from scratch to calculate where a new data point should go!
  - This technique is for visualization, not for building cross-validated classifiers on top of

Procrustes alignment

- Can measure the stability of the overall structure of the embedding.
- Considering the normalized Procrustes distance between the embedding of a sub-sample, and the corresponding sub-sample of an embedding of the full dataset
- As the size of the sub-sample increases the average distance per point between the sub- sampled embeddings should decrease, potentially toward some asymptote of maximal agreement under repeated runs. Ideally this asymptotic value would be zero error, but for stochastic embeddings such as UMAP and t- SNE this is not achievable.


PCA
10
5
0
−5
−10
−10
−5
0
5
10
t-SNE
20
10
0
−10
−20
−30
−20
−10
0
10
20
LLE
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.04
−0.02
0.00
0.02
0.04