3011979 Intro to Deep Learning for Medical Imaging

L4: Unsupervised learning – t-SNE and UMAP

Feb 19th, 2021

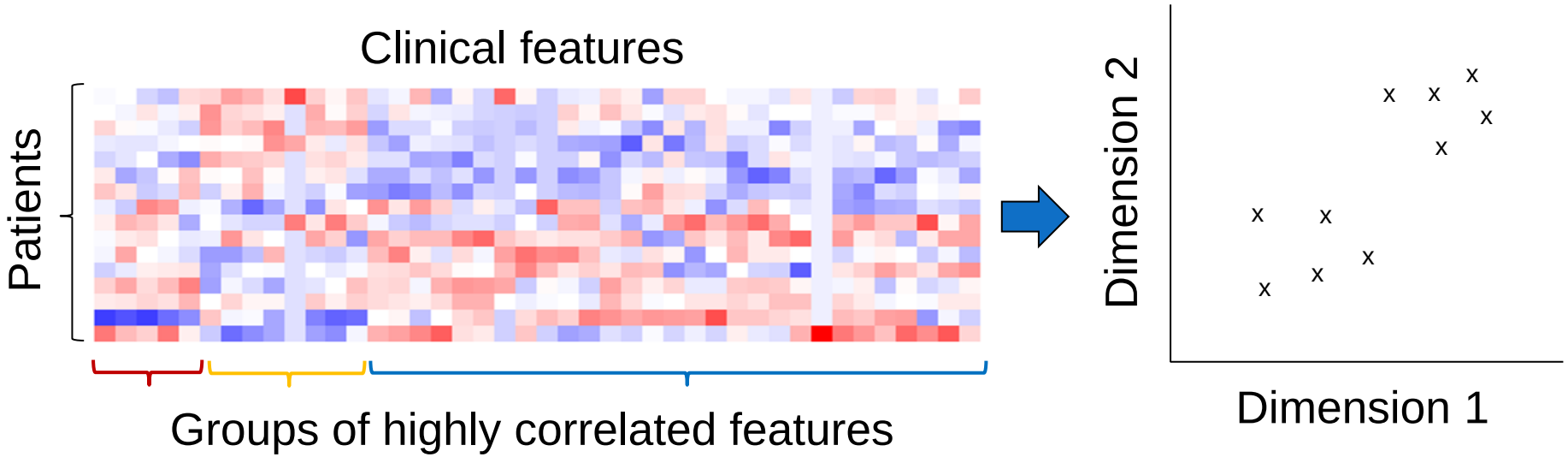
Sira Sriswasdi, Ph.D.
Research Affairs, Faculty of Medicine
Chulalongkorn University

Today’s objectives

- Recap of dimensionality reduction concepts
- t-distribution stochastic neighbor embedding (t-SNE)
- Uniform manifold approximation and projection (UMAP)

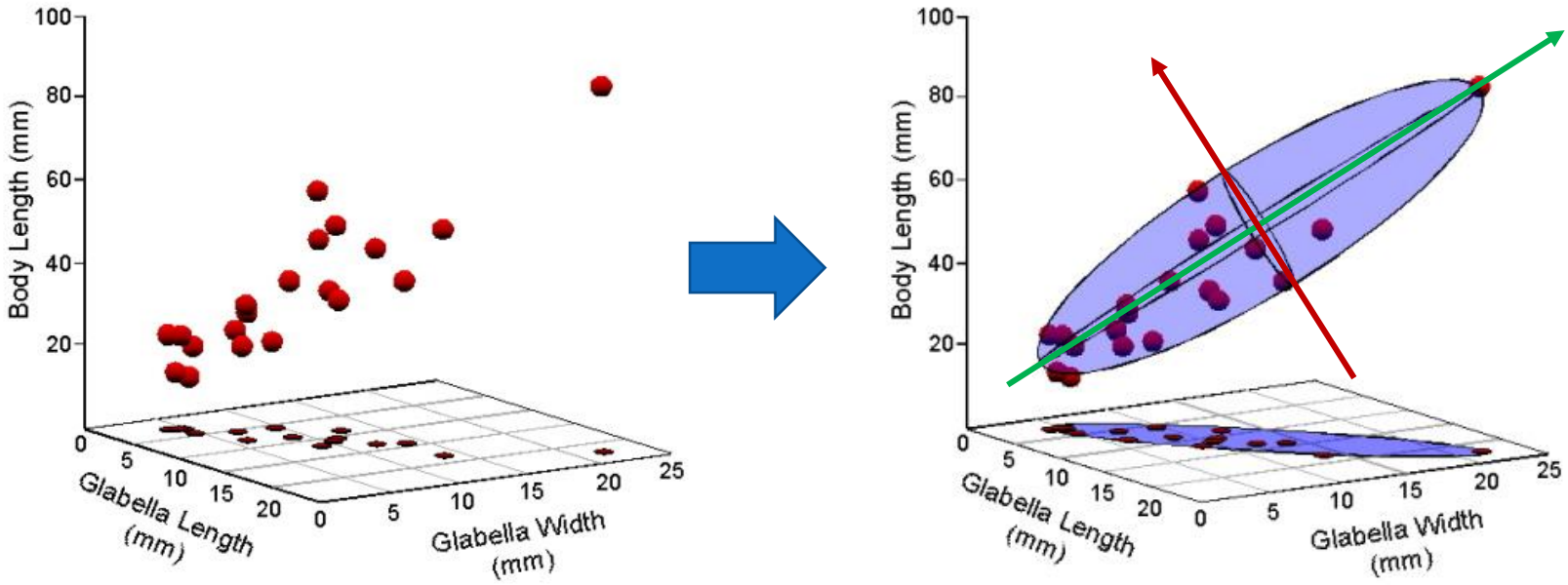
Some recaps on dimensionality reduction

Dimensionality reduction



- Collapse highly correlated / redundant features
- Preserve distance / neighbor between data points

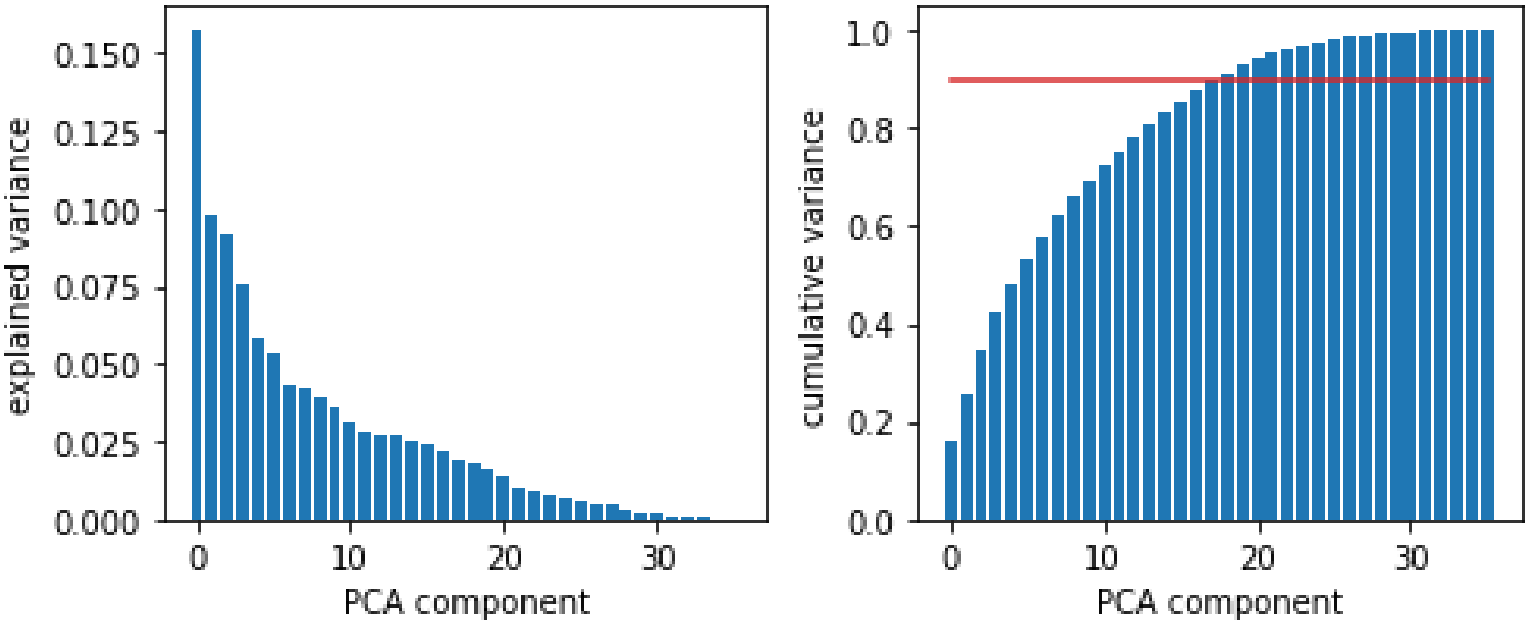
Principal component analysis (PCA)



Source: the paleontological association

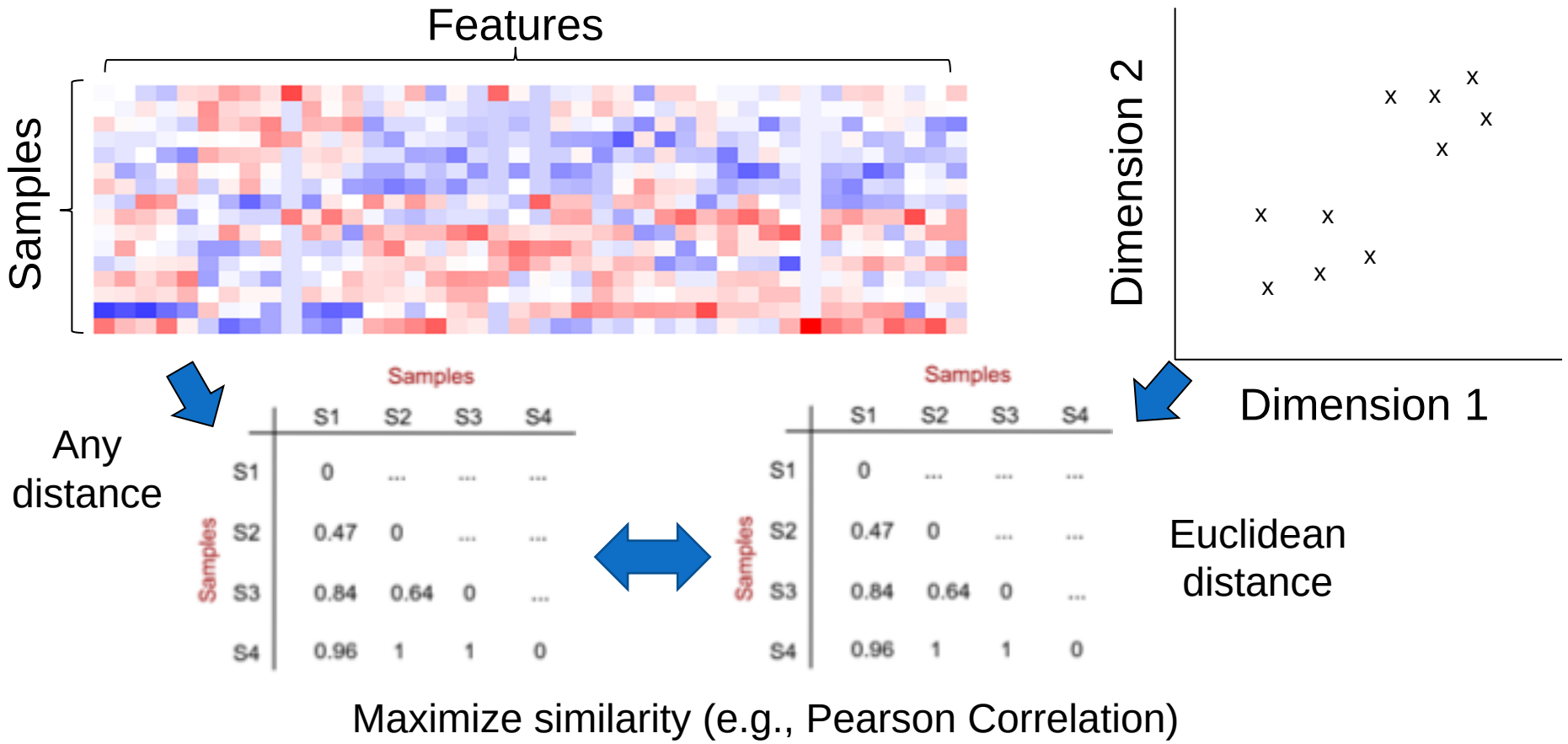
- Fit an *n*-dimensional ellipsoid to the data cloud
  - Essentially rotation of original axes
  - Preserve Euclidean distance between all data points
- Sort dimensions based on data variances

Explained variance



- Components that capture high variances are typically useful but not always
- Reduce dimension by keeping only the first *m* dimensions

Multidimensional scaling (MDS)



Maximize similarity (e.g., Pearson Correlation)

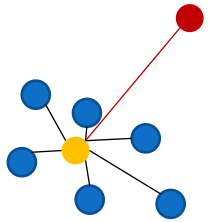
- MDS projects data points onto new dimensions while trying to preserve the similarity between two distance matrices

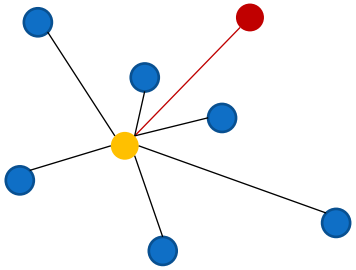
Limitation of PCA and MDS

- PCA does not affect distance / neighbor
- MDS preserves the whole pairwise distance matrix uniformly
  - Distance *d* between X_i and X_j means the same as distance *d* between X_n and X_m
  - What if different groups of data points were generated from different underlying processes / distributions?
- Pattern can be qualitative and relative

Stochastic neighbor embedding

Stochastic Neighbor Embedding (SNE)





score(o | o) = probability that o would pick o as neighbor under a normal distribution center at o

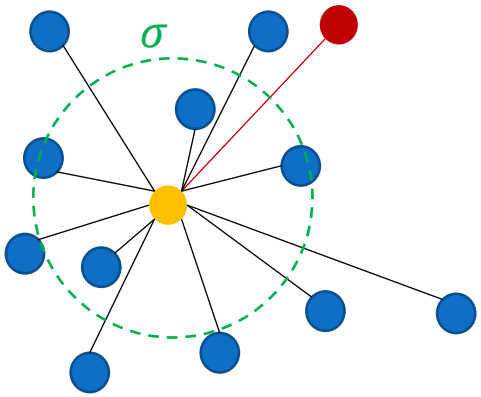
$$= \frac{e^{-\frac{(\text{dist}(o,o))^2}{2\sigma^2}}/\sigma}{\sum_{o} e^{-\frac{(\text{dist}(o,o))^2}{2\sigma^2}}/\sigma}$$

o = other data points

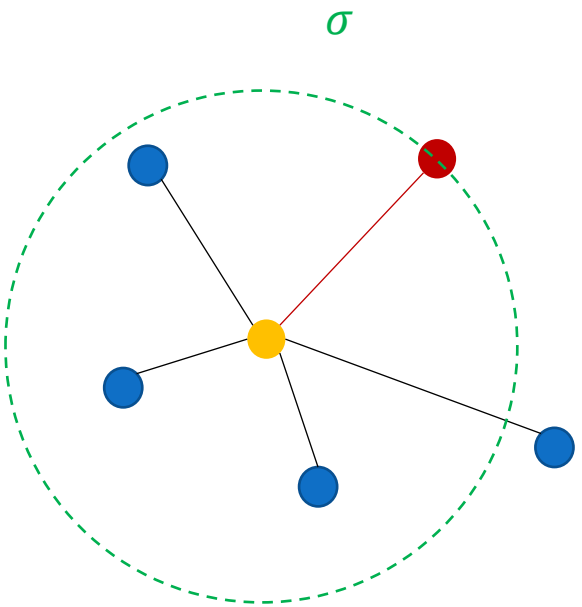
similarity(o, o) = [score(o | o) + score(o | o)] / 2 * # data points

- Turn distance into neighbor probability
- Normalization over other data points turn absolute distance into relative distance
- Parameter σ handles the difference in data density
  - Also called perplexity

Perplexity



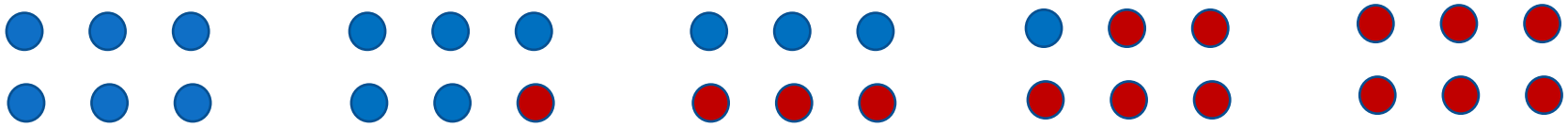
Many data points in proximity
Small σ



Few data points in proximity
Large σ

- Value of σ varies for each data point based on the density
- Perplexity can be interpreted as “the number of neighbors that each data point should be related to”
- Typical perplexity is set between 5 and 50

Entropy and information content

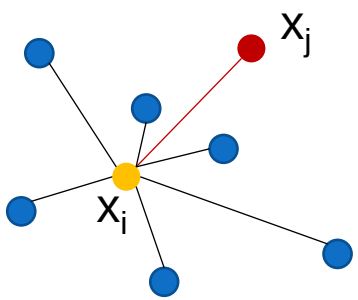


6/6, 0/6 | 5/6, 1/6 | 3/6, 3/6 | 1/6, 5/6 | 0/6, 6/6

- Which of these has the highest information content / entropy?

- Entropy of a distribution *P*: $\mathrm{H}(P) = -\sum_i p_i \log_2 p_i$
  - What are the values of H(P) in the examples above?
  - When is entropy maximized?
  - When is entropy minimized?

Perplexity and entropy

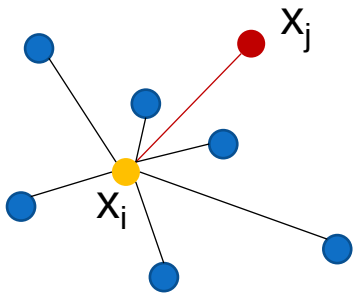


$$p_{j|i} = \frac{\exp\left(-\left\|x_i - x_j\right\|^2 / 2{\sigma_i}^2\right)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2{\sigma_i}^2)}$$

- For each *i*, $P_i = \{p_{j|i}\}$ is a distribution because $\sum_{k \neq i} p_{k|i} = 1$
- Perplexity of $x_i = 2^{H(P_i)} = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}$
- σ_i's are set so that the perplexity at every point is the same as the user input perplexity value

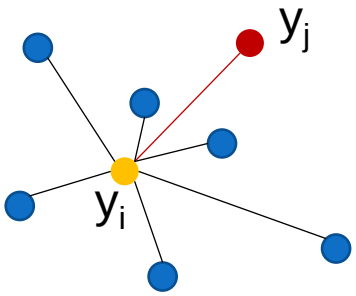
Embedding into low dimension

High-dimension



$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i{}^2\right)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i{}^2)}$$

Low-dimension



$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$$

- We want to make $\{q_{j|i}\} \approx \{p_{j|i}\}$
- Kullback-Leibler (KL) divergence $D_{KL}(P \,||\, Q)$ is a measure of how much distribution *P* differ from distribution *Q*
  - $D_{KL}(P \,||\, Q) = -\sum_i \sum_j p_{j|i} \log_2 \frac{p_{j|i}}{q_{j|i}}$
  - What happen if $q_{j|i} = p_{j|i}$?
- Challenge: Does KL preserve all distances uniformly?

Gradient descent

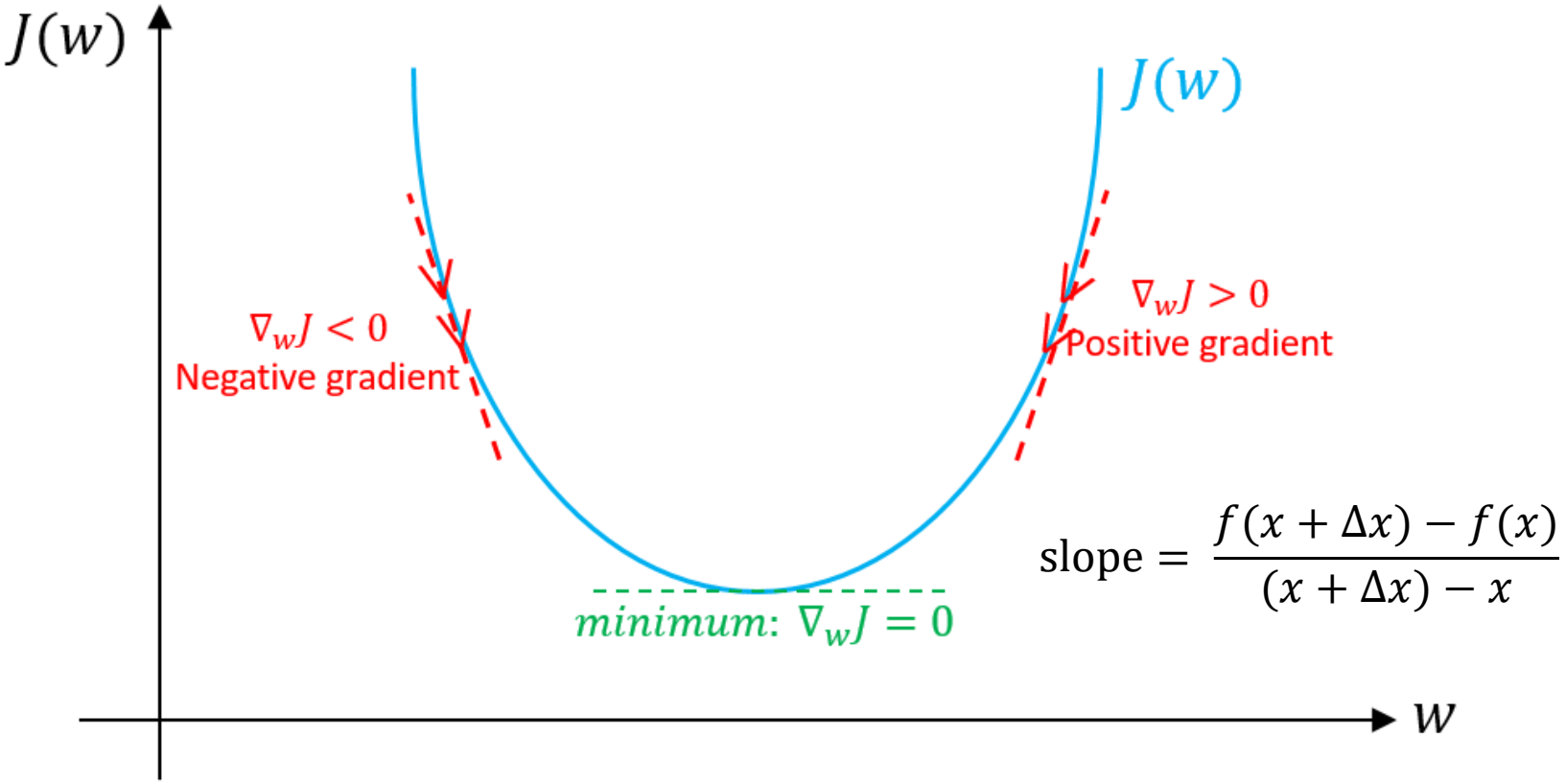


Image from towarddatascience.com

- Gradient ∇ is a generalization of slope in multi-dimension
- Gradient at optimal point = 0
- Gradient points toward the direction of increase in f(x)

Gradient descent in multi-dimension

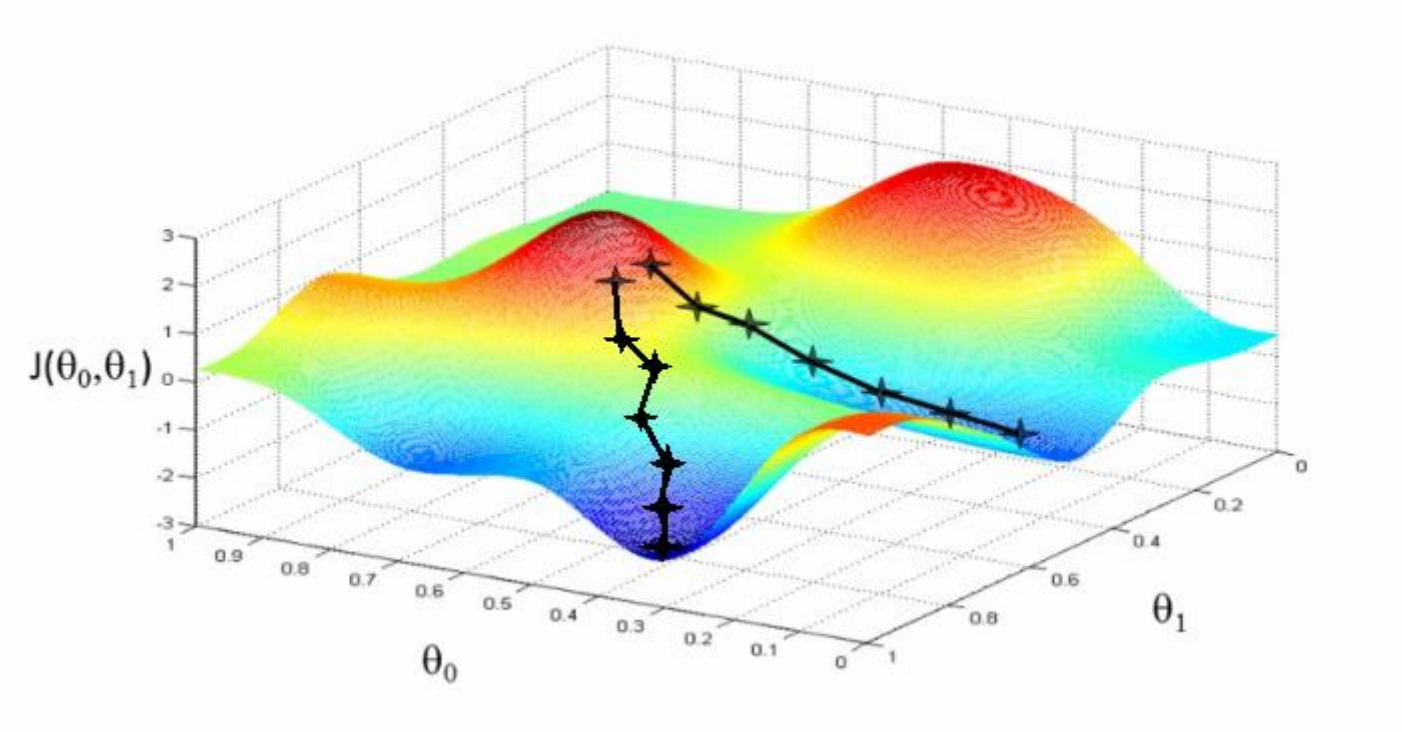


Image from shashank-ojha.github.io

- https://emiliendupont.github.io/2018/01/24/optimization-visualization/

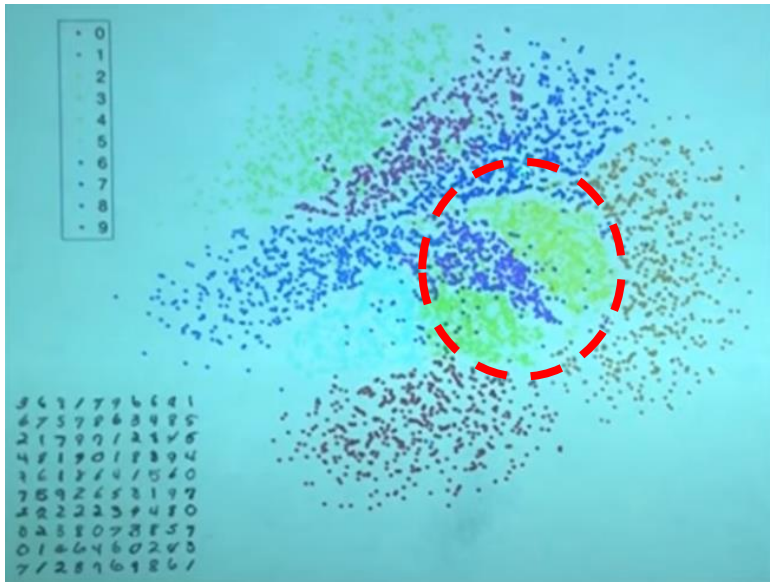
Gradient descent for SNE

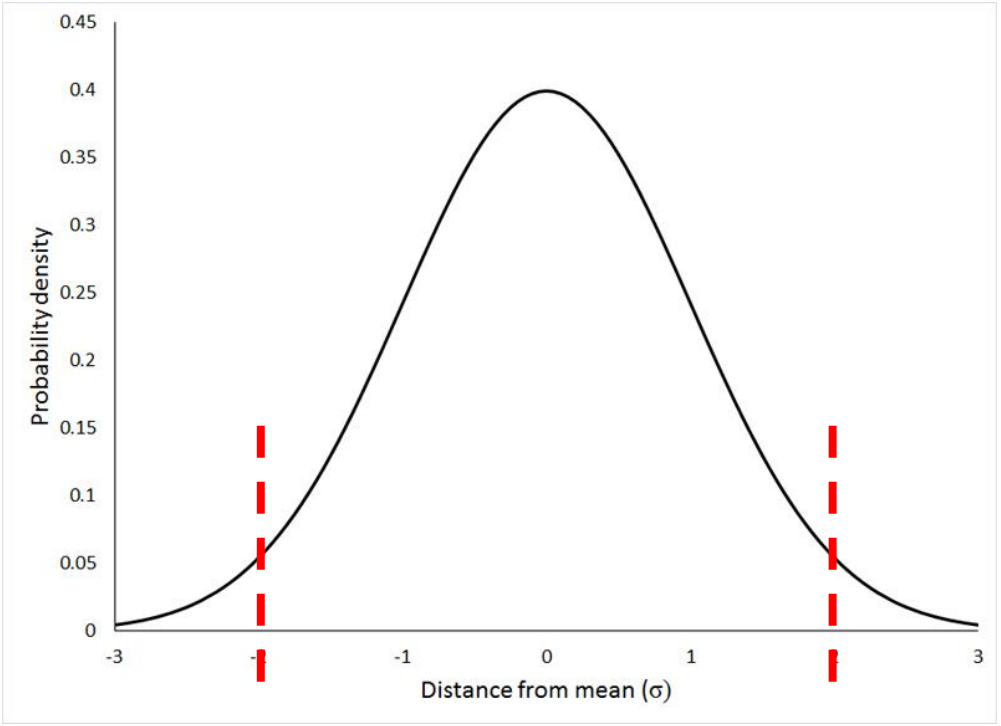
- $D_{KL}(P \,||\, Q) = -\sum_i \sum_j p_{j|i} \log_2 \frac{p_{j|i}}{q_{j|i}}$, $q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$

- When calculating partial derivative w.r.t y_i, other variables can be considered as constant
- y_i appears in $\log q_{j|i}$ and $\log q_{i|j}$ for j ≠ I

- The gradient is surprisingly simple!
  - $\frac{\delta D_{KL}}{\delta y_i} = 2 \sum_{j \neq i} \left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)\left(y_i - y_j\right)$
  - http://pages.di.unipi.it/errica/assets/files/sne_tsne.pdf
  - This tells us the direction to move y_i to reduce the divergence
  - Depends on relative positions with other points y_j

Problem with SNE





- Gaussian distribution is very narrow at the tails
  - $q_{j|i}$ reduces to zero very quickly
- SNE tends to place data points close together
  - A lot of overlap between clusters of data points
  - This is called “overcrowding” problem

From SNE to t-SNE

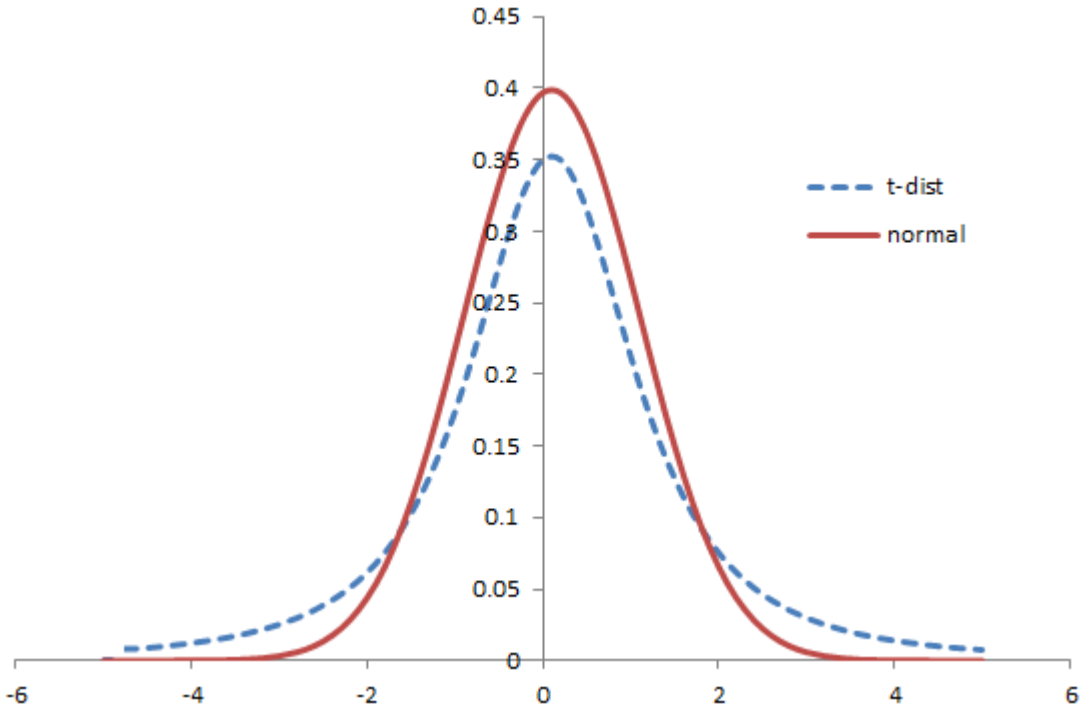
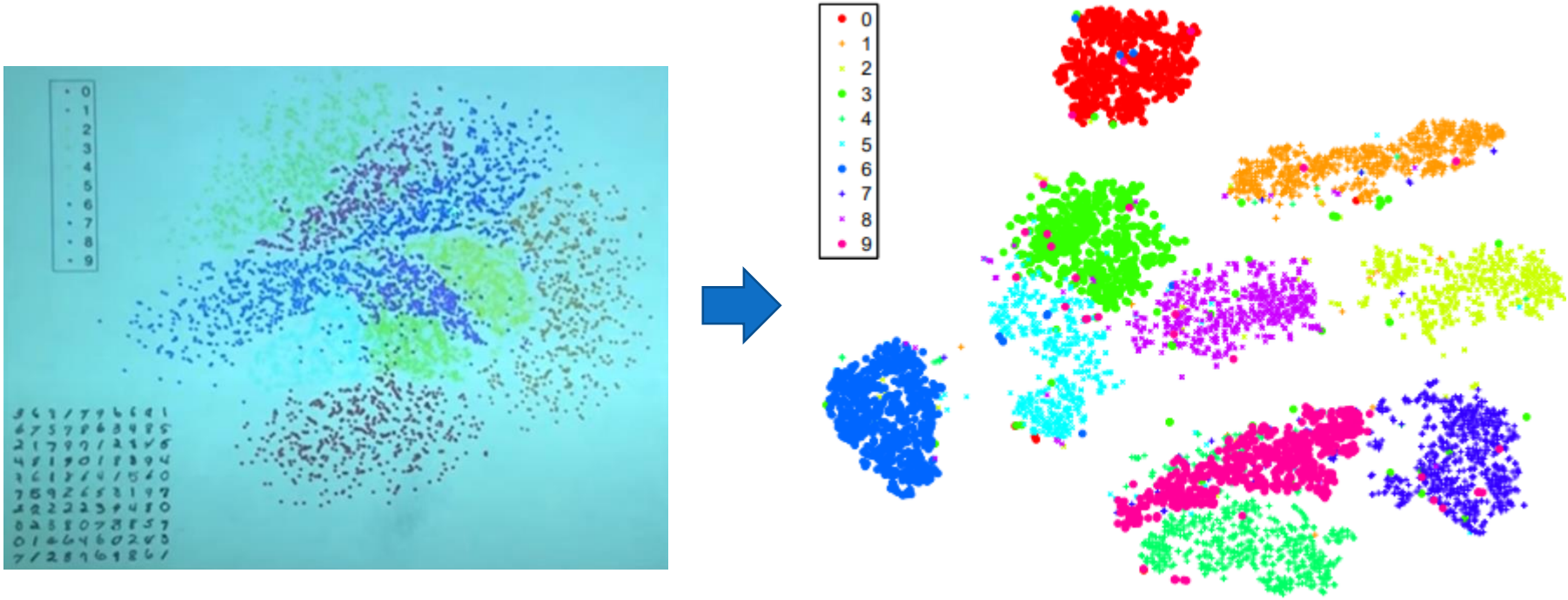


Image from riskprep.com

- Student's t distribution with one degree of freedom
  - Density = $\frac{\left(1+x^2\right)^{-1}}{\sqrt{\pi}\cdot\Gamma(0.5)}$ has a fatter tail compared to Gaussian distribution
  - Update $q_{j|i} = \frac{\left(1+\|y_i-y_j\|^2\right)^{-1}}{\sum_{k\neq i}(1+\|y_i-y_k\|^2)^{-1}}$

t-distribution spreads out the data points



Maaten, L. and Hinton, G. J of Machine Learning Research 9:2579-2605 (2008)

t-SNE in Python

sklearn.manifold.TSNE

class `sklearn.manifold.` **`TSNE`**(*n_components=2, *, perplexity=30.0, early_exaggeration=12.0, learning_rate=200.0, n_iter=1000, n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', init='random', verbose=0, random_state=None, method='barnes_hut', angle=0.5, n_jobs=None, square_distances='legacy'*) [source]

t-distributed Stochastic Neighbor Embedding.

t-SNE [1] is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.

It is highly recommended to use another dimensionality reduction method (e.g. PCA for dense data or TruncatedSVD for sparse data) to reduce the number of dimensions to a reasonable amount (e.g. 50) if the number of features is very high. This will suppress some noise and speed up the computation of pairwise distances between samples. For more tips see Laurens van der Maaten's FAQ [2].

Read more in the User Guide.

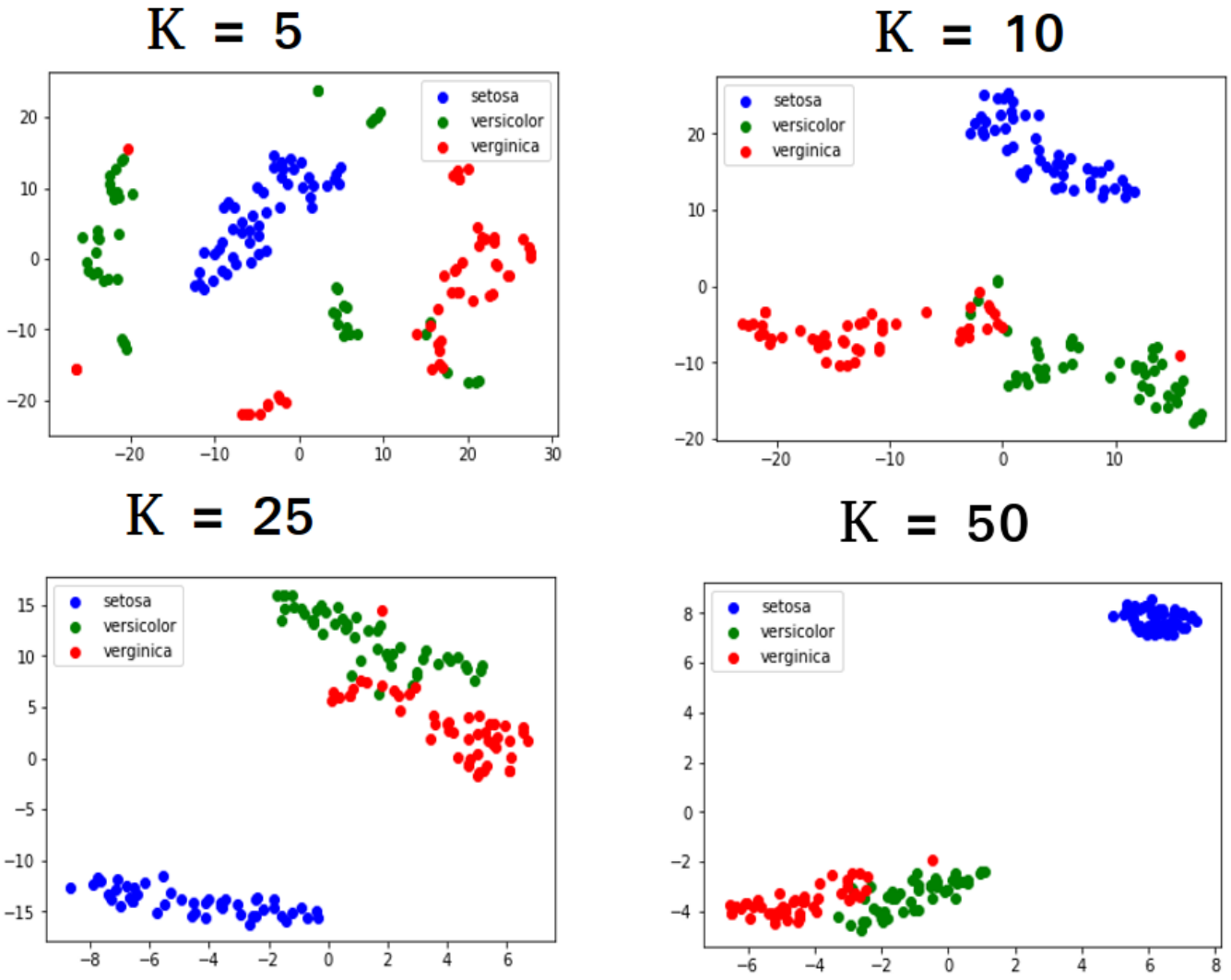
Parameters:

n_components : *int, default=2*
Dimension of the embedded space.

perplexity : *float, default=30.0*
The perplexity is related to the number of nearest neighbors that is used in other manifold learning algorithms. Larger datasets usually require a larger perplexity. Consider selecting a value between 5 and 50. Different values can result in significantly different results.

early_exaggeration : *float, default=12.0*
Controls how tight natural clusters in the original space are in the embedded space and how much space will be between them. For larger values, the space between natural clusters will be larger in the embedded space. Again, the choice of this parameter is not very critical. If the cost function increases during initial optimization, the early exaggeration factor or the learning rate might be too high.

Effect of perplexity (K) on *t*-SNE result



Source: blog.paperspace.com/dimension-reduction-with-t-sne/

Pros and cons of t-SNE

- Able to capture more qualitative “neighbor” relationship
  - Turn absolute distance into relative / probabilistic distance
- Address the issue of varying data density
  - Fixed perplexity via σ_i
- Excel at grouping nearby data points into clusters
  - KL divergence prioritize points with high $p_{j|i}$ (nearby points)
- Long-range relationship tends to be lost
- Quite slow, especially on large datasets
- Cannot transform new data points onto a previous map

Uniform manifold approximation and projection

Topology and manifold

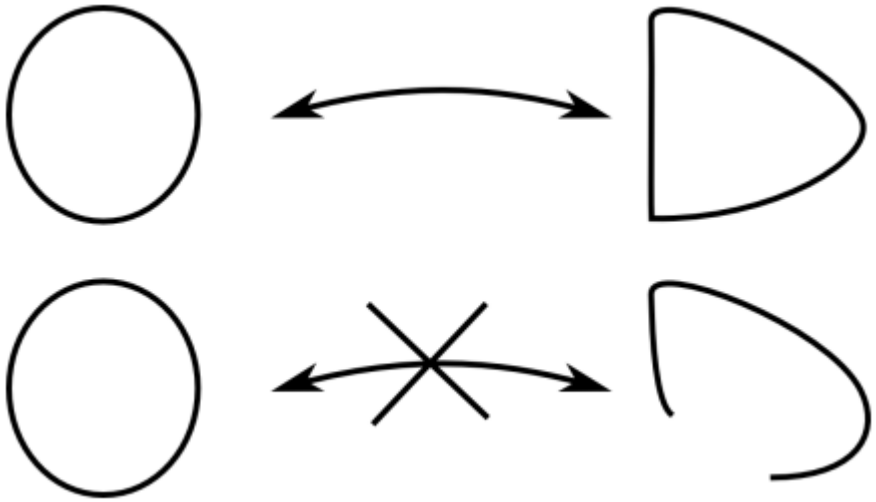


Image from Renzo's Math 490 note

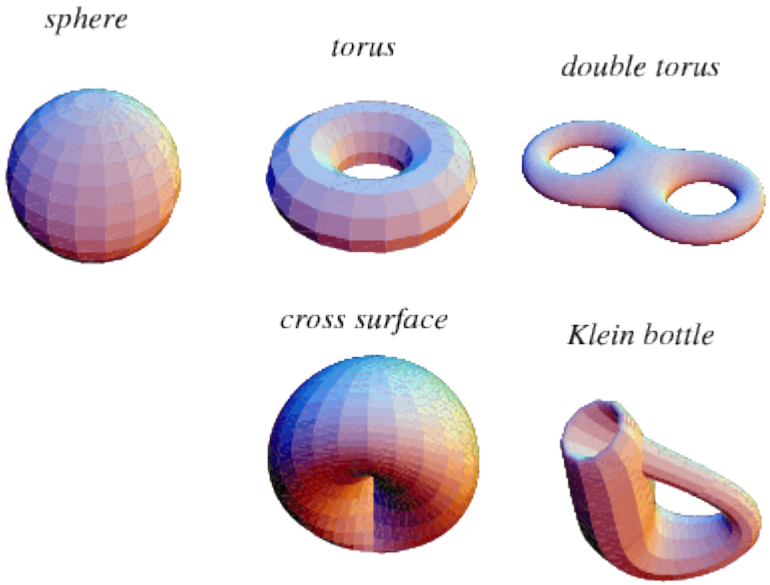
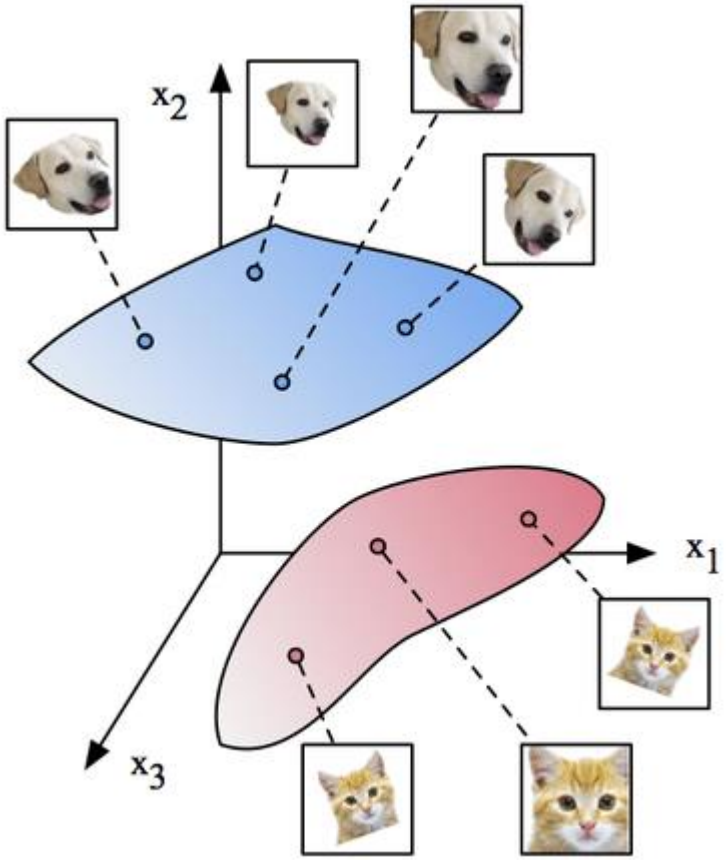
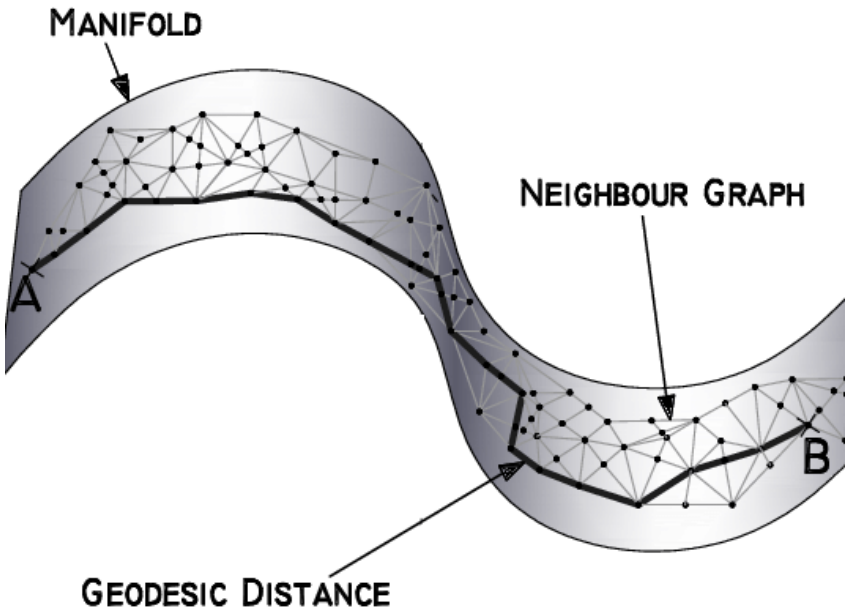


Image from bjlkeng.github.io

- Topology is the study of spaces
- Manifold is a topological space that locally resembles R^n
  - R^1 is a line, R^2 is a sheet, R^3 is a solid cube
- Challenge: Is a Y-shape pattern a manifold?

The manifold hypothesis





Karam, Z.N. and Campbell, W. “Graph embedding for speaker recognition”

Chung, S. et al. “Classification and Geometry of General Perceptual Manifolds”

- “Real-world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space”

Nerve theorem

Theorem 1 (Nerve theorem). *Let $\mathcal{U} = \{U_i\}_{i \in I}$ be a cover of a topological space X. If, for all $\sigma \subset I$ $\bigcap_{i \in \sigma} U_i$ is either contractible or empty, then $\mathcal{N}(\mathcal{U})$ is homtopically equivalent to X.*

UMAP presentation by Dr. McInnes

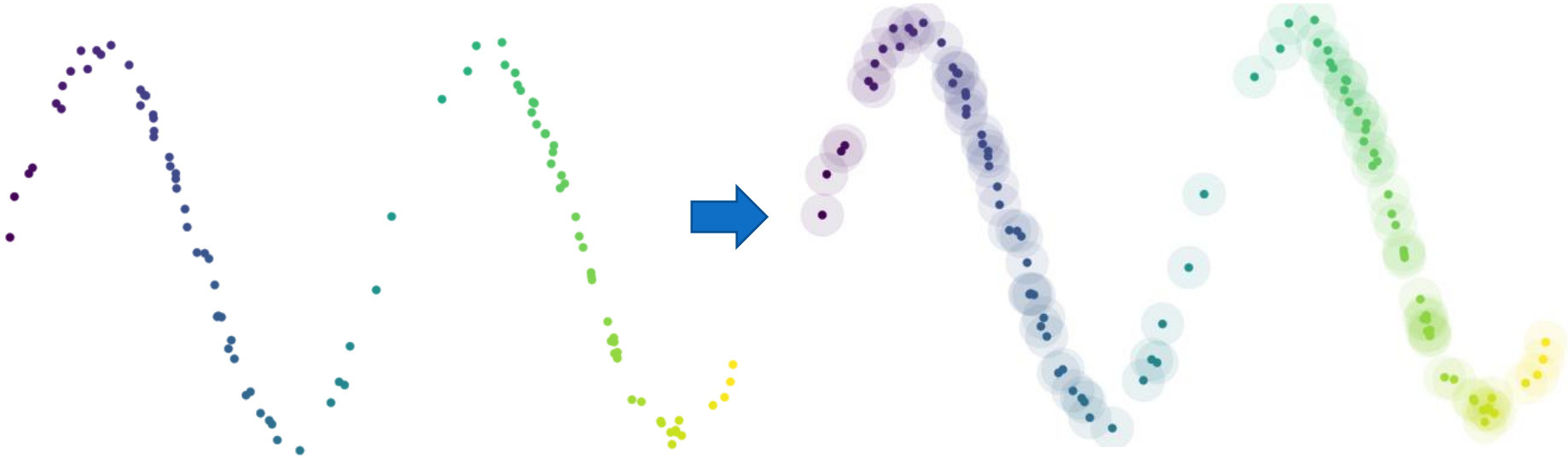


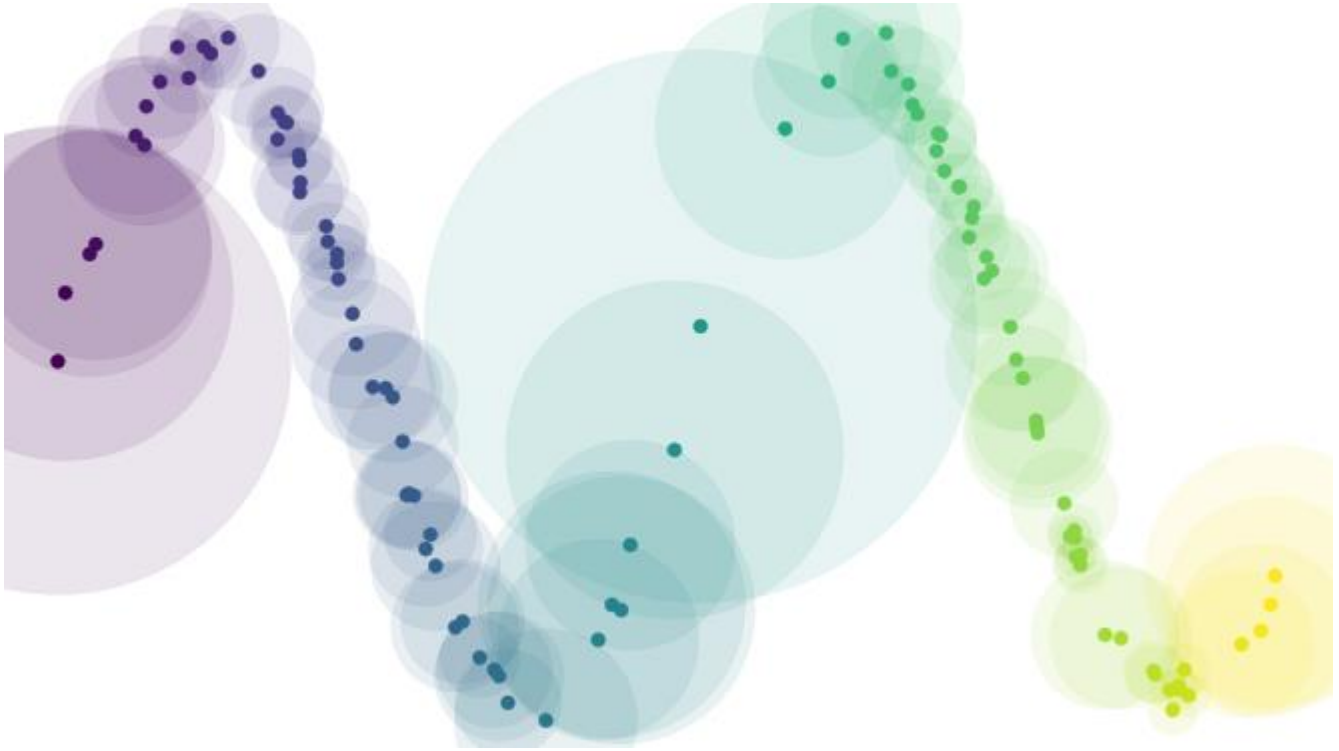
Image from documentation of umap-learn library

- A cover of a space is a collection of open disks that cover all points in that space

Implication of Nerve theorem

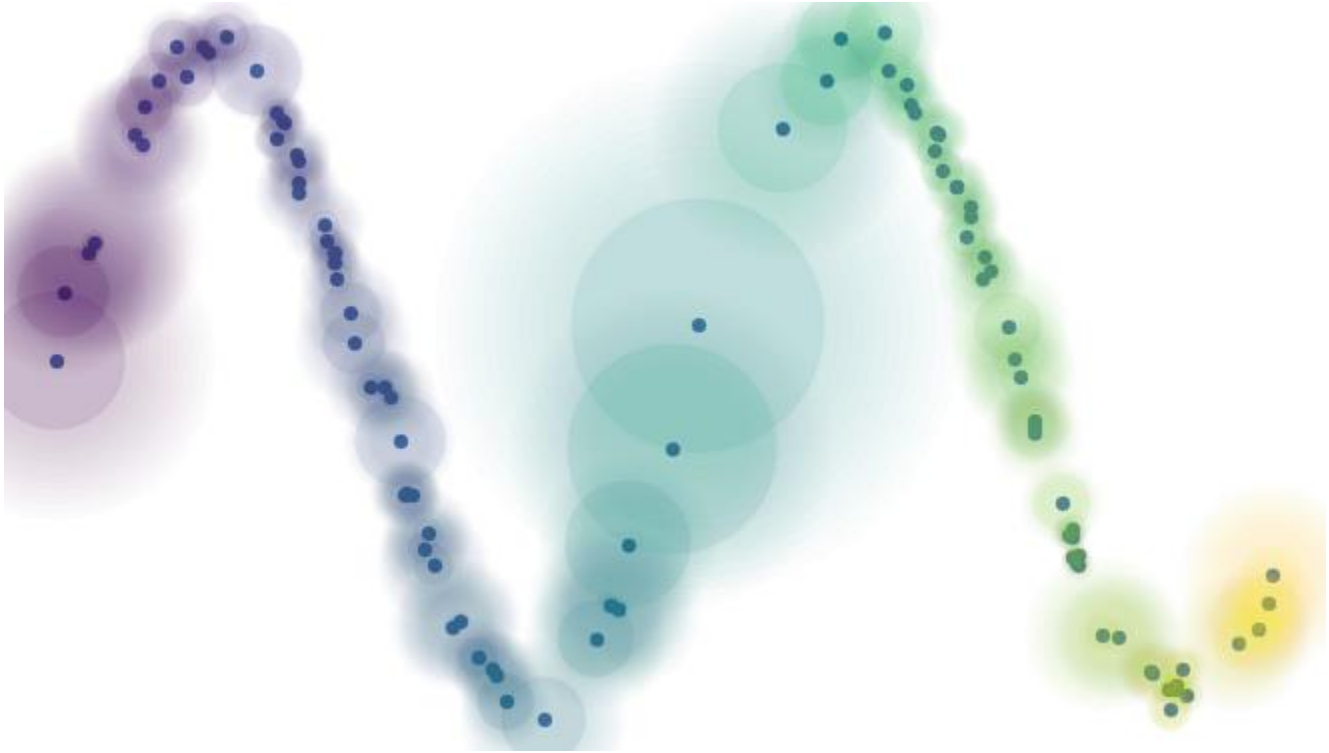
- The topological information of the network between data points induced by the cover is equivalent to the manifold that underlies this data
- There are some disconnected data points
  - Because we used the same disk radius everywhere!

Assume that data were uniformly drawn



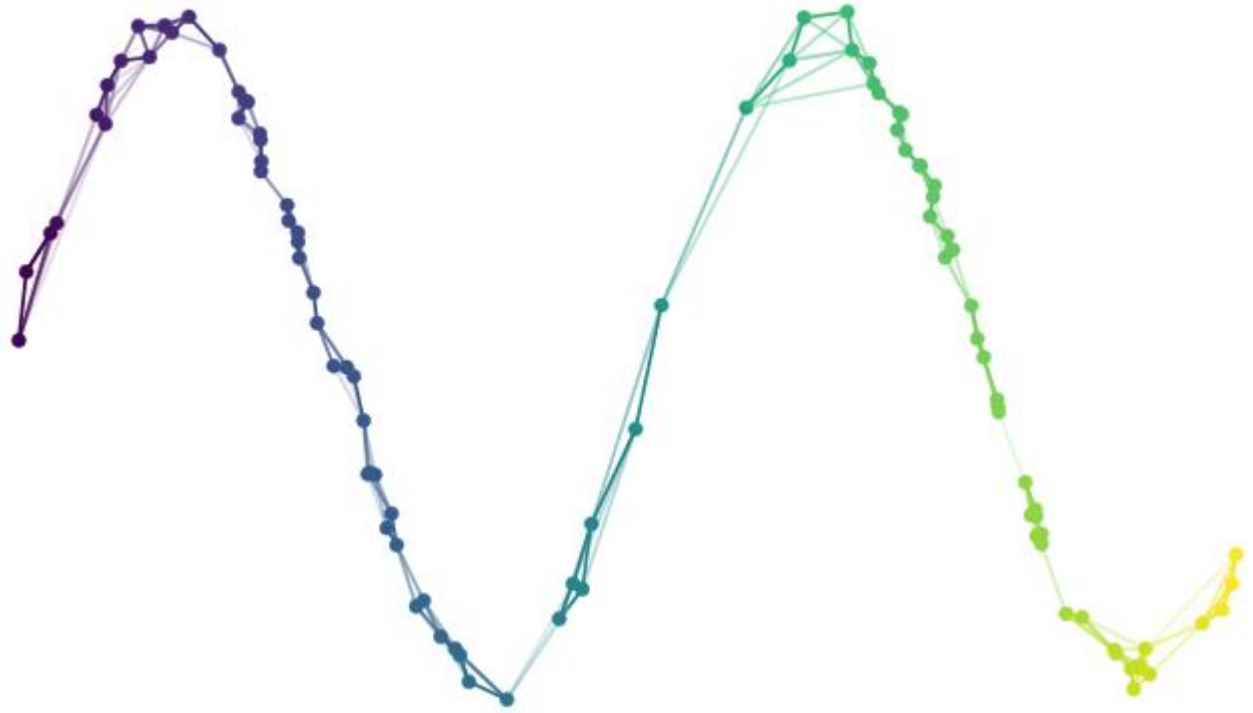
- Distance to the k-nearest neighbors should be the same for every data points
  - Dense area = contracted manifold = scale down the distance
  - Sparse area = stretched manifold = scale up the distance
- Has similar effect as σ (perplexity) in *t*-SNE

Adding fuzziness



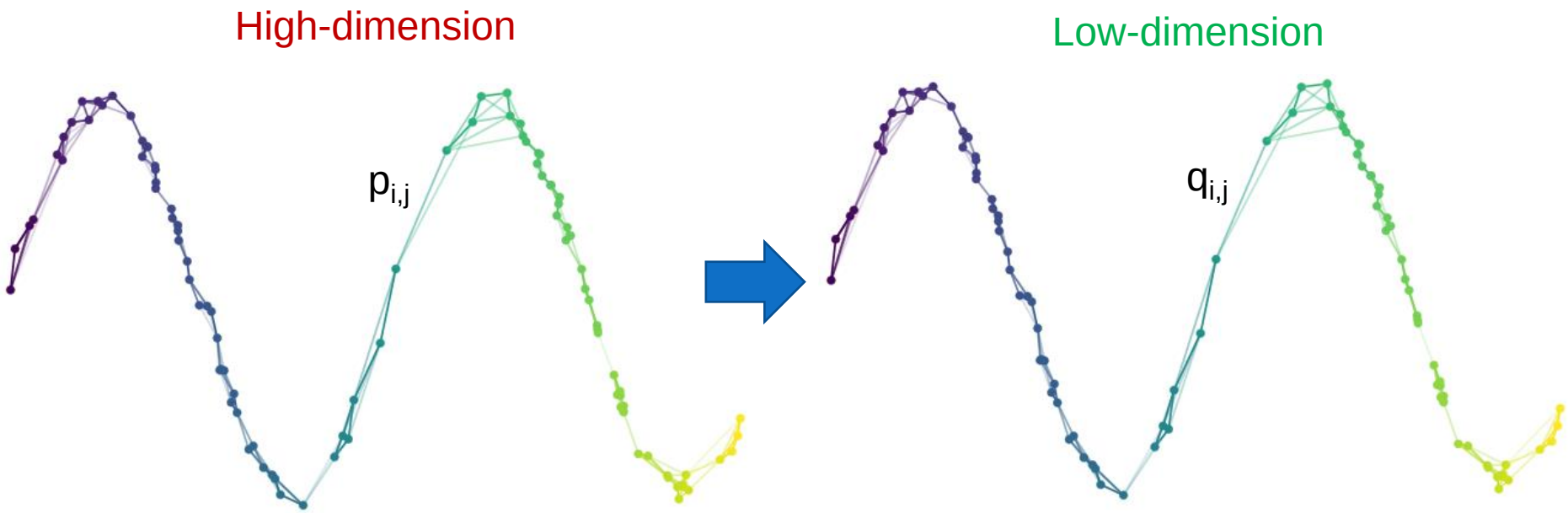
- High confidence up to the first neighbor (locally connected)
  - Probability that x_i is connected to its nearest neighbor is 1.0
- Long-range relationships are more fuzzy
  - Relationship between x_i and x_j may be supported by multiple disks
  - Probability that x_i is connected to x_j is aggregated over all disks

The final probabilistic network



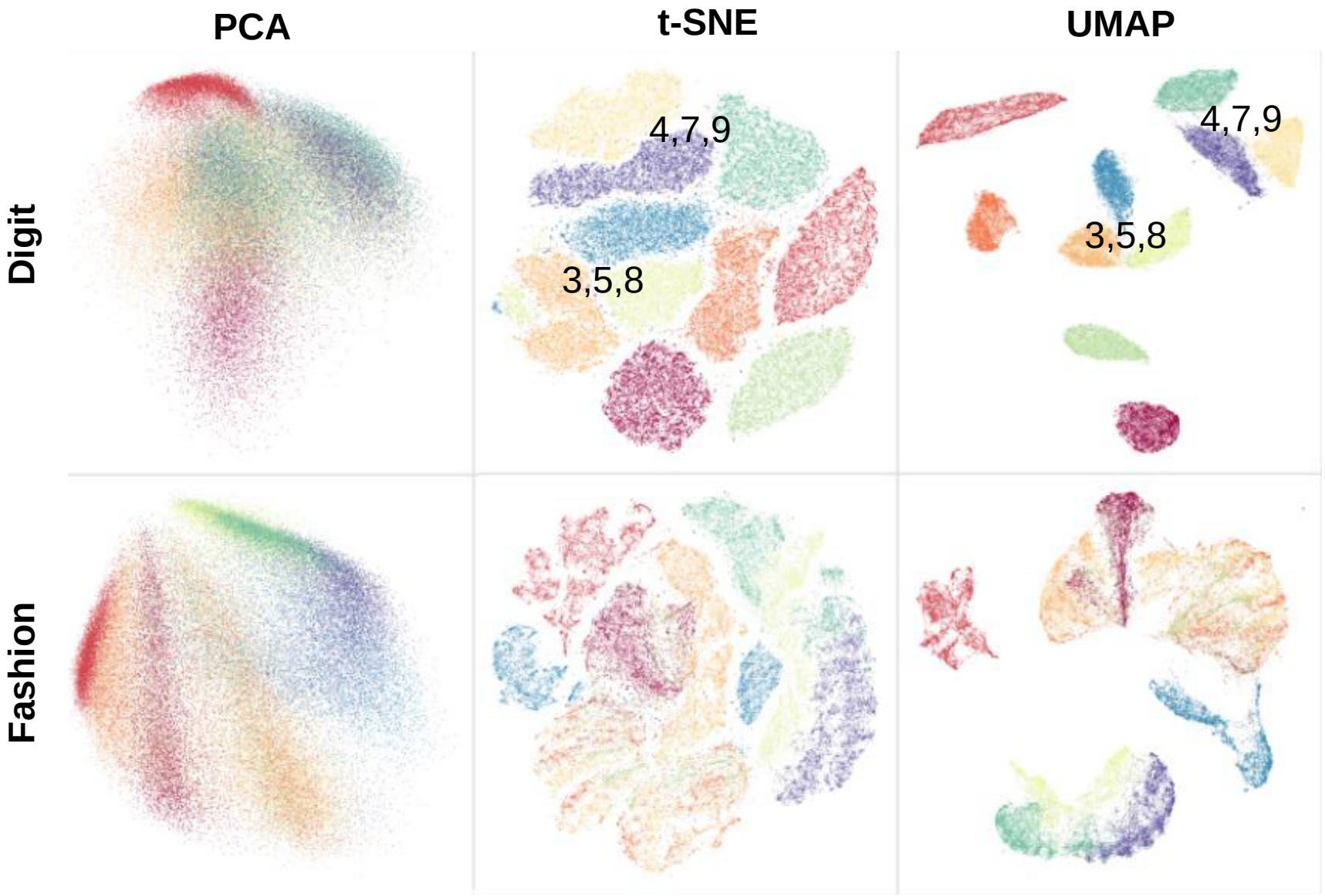
- Nodes are data points
- Edges connected data points with intersecting disks
- Edge weights are probability scores

Embedding into low dimension



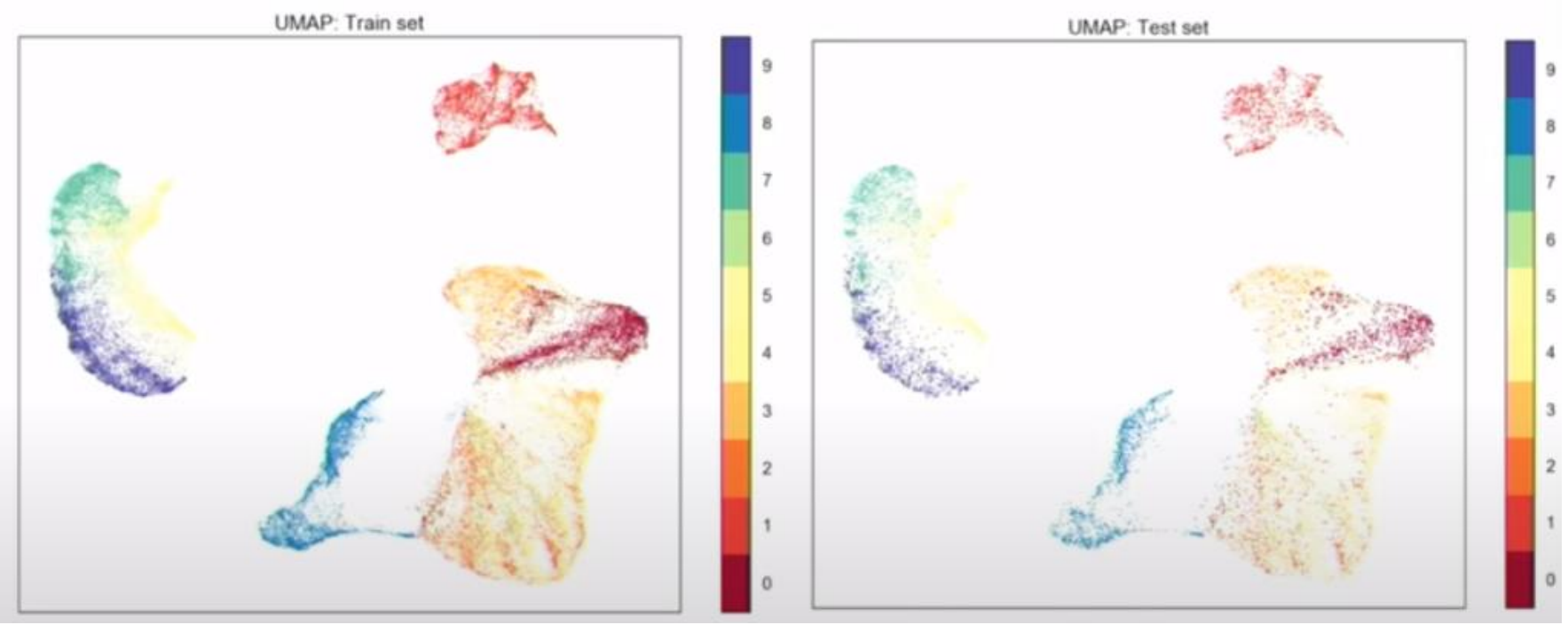
- We want to make $\{q_{j|i}\} \approx \{p_{j|i}\}$
- Instead of KL divergence, we use cross-entropy here
  - $\text{Crossentropy}\ (P \,||\, Q) = \sum_i \sum_j p_{i,j} \log_2 \frac{p_{i,j}}{q_{i,j}} + \sum_i \sum_j (1 - p_{i,j}) \log_2 \frac{(1-p_{i,j})}{(1-q_{i,j})}$
  - How does this differ from KL divergence?

UMAP also preserve long-range information



McInnes, L., Healy, J. and Melville, J. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”

UMAP can transform unseen data points



UMAP presentation by Dr. McInnes

- UMAP can theoretically be used as feature processing before feeding into machine learning model
- But expected to work well only on large datasets

UMAP in Python

class **umap.umap_.UMAP**(*n_neighbors=15, n_components=2, metric='euclidean', metric_kwds=None, output_metric='euclidean', output_metric_kwds=None, n_epochs=None, learning_rate=1.0, init='spectral', min_dist=0.1, spread=1.0, low_memory=True, n_jobs=-1, set_op_mix_ratio=1.0, local_connectivity=1.0, repulsion_strength=1.0, negative_sample_rate=5, transform_queue_size=4.0, a=None, b=None, random_state=None, angular_rp_forest=False, target_n_neighbors=-1, target_metric='categorical', target_metric_kwds=None, target_weight=0.5, transform_seed=42, transform_mode='embedding', force_approximation_algorithm=False, verbose=False, unique=False, densmap=False, dens_lambda=2.0, dens_frac=0.3, dens_var_shift=0.1, output_dens=False, disconnection_distance=None*) [source]

Uniform Manifold Approximation and Projection

Finds a low dimensional embedding of the data that approximates an underlying manifold.

n_neighbors: float (optional, default 15)

The size of local neighborhood (in terms of number of neighboring sample points) used for manifold approximation. Larger values result in more global views of the manifold, while smaller values result in more local data being preserved. In general values should be in the range 2 to 100.

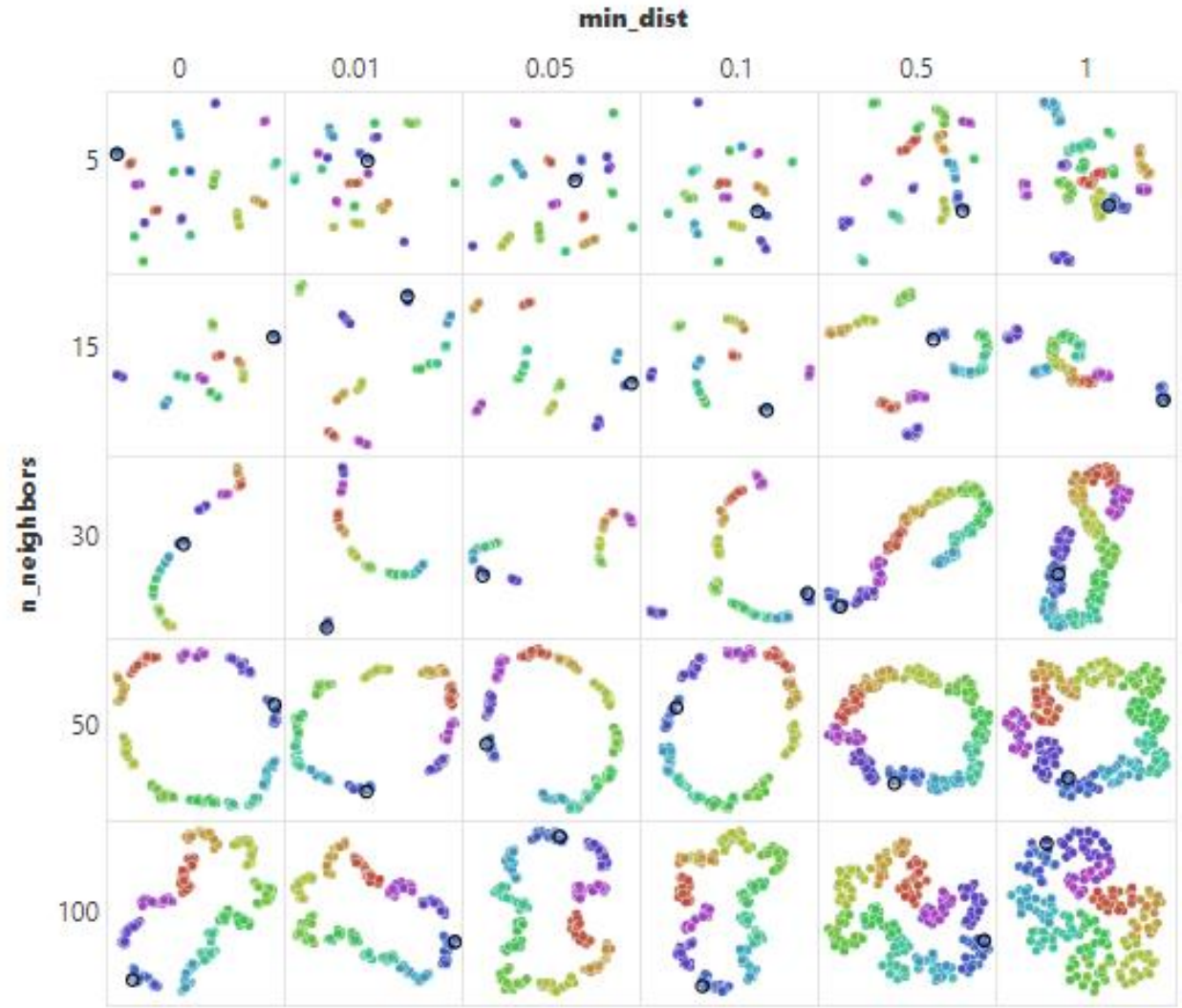
n_components: int (optional, default 2)

The dimension of the space to embed into. This defaults to 2 to provide easy visualization, but can reasonably be set to any integer value in the range 2 to 100.

metric: string or function (optional, default 'euclidean')

The metric to use to compute distances in high dimensional space. If a string is passed it must match a valid predefined metric. If a general metric is required a function that takes two 1d arrays and returns a float can be provided. For performance purposes it is required that this be a numba jit'd function. Valid string metrics include:

Effect of *n_neighbors* and *min_dist*



Source: https://pair-code.github.io/understanding-umap/

Any question?