3011979 Practical Python for Data Sciences and Machine Learning

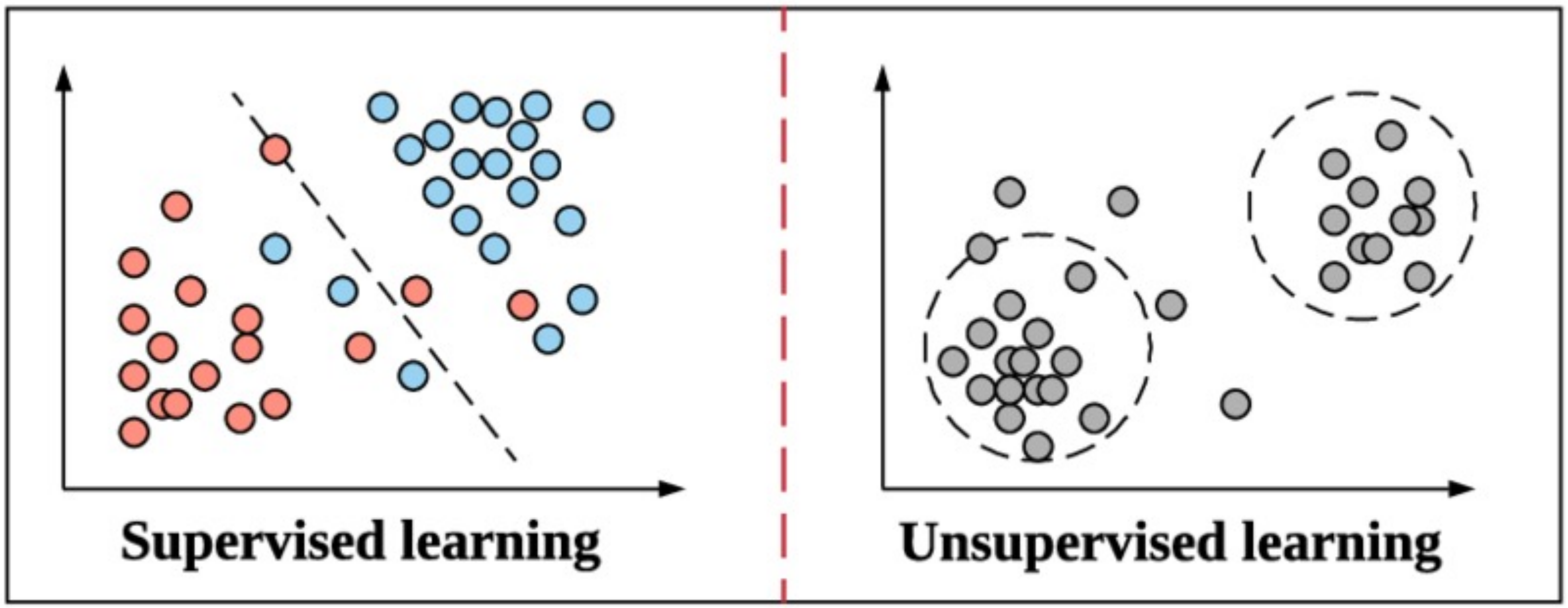
L5: Dimensionality Reduction

Feb 11th, 2022

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

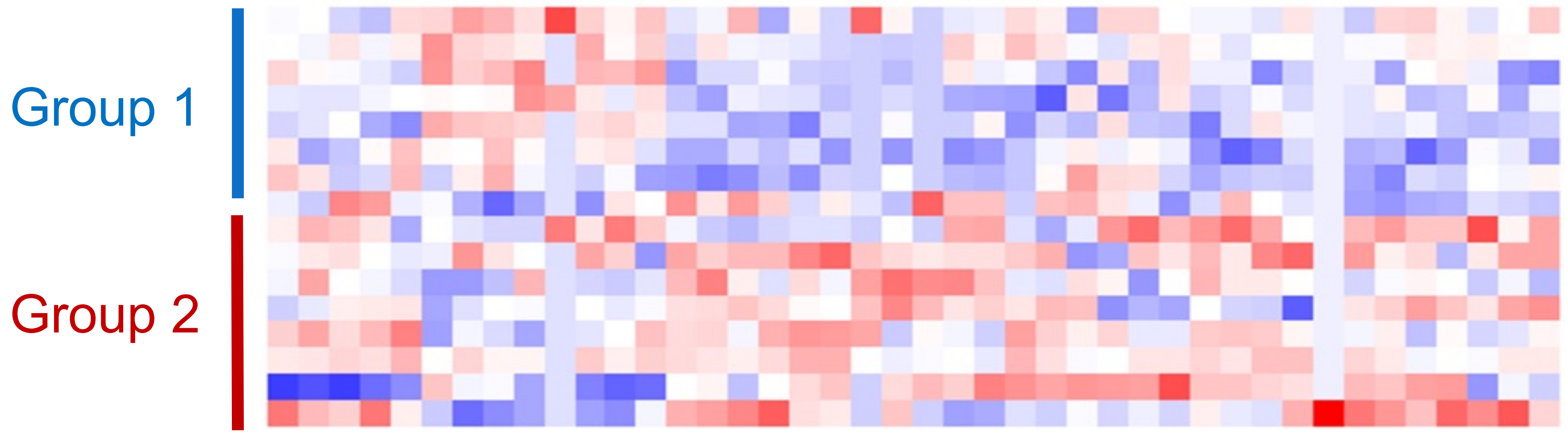
Patterns behind data

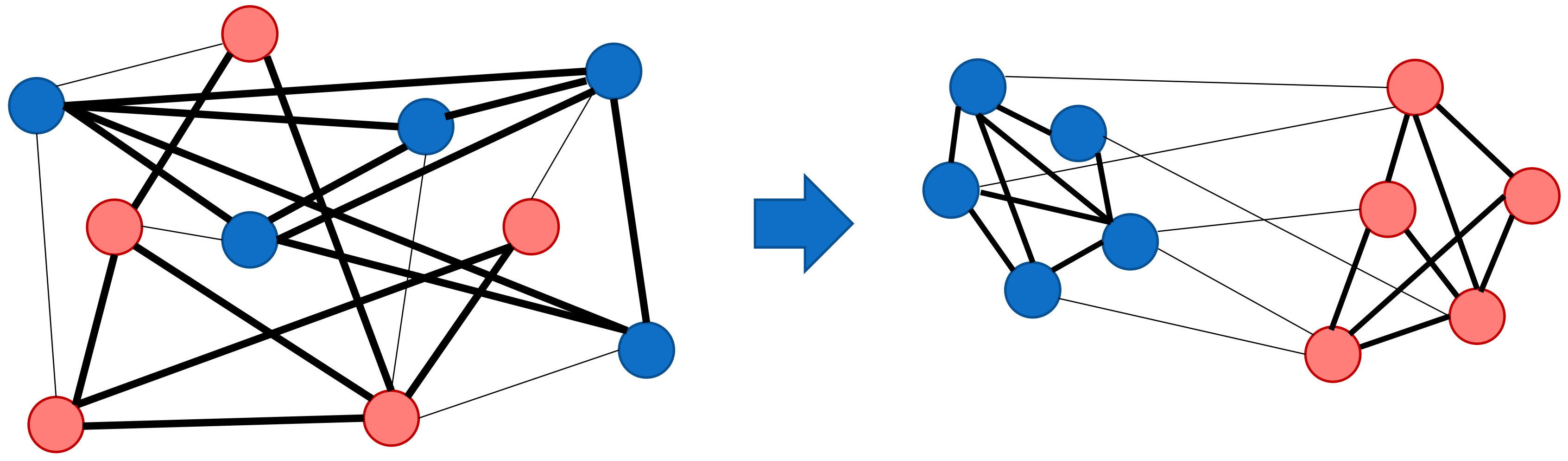
Unsupervised learning



Qian, B. et al. “Orchestrating the Development Lifecycle of Machine Learning-Based IoT Applications: A Taxonomy and Survey”

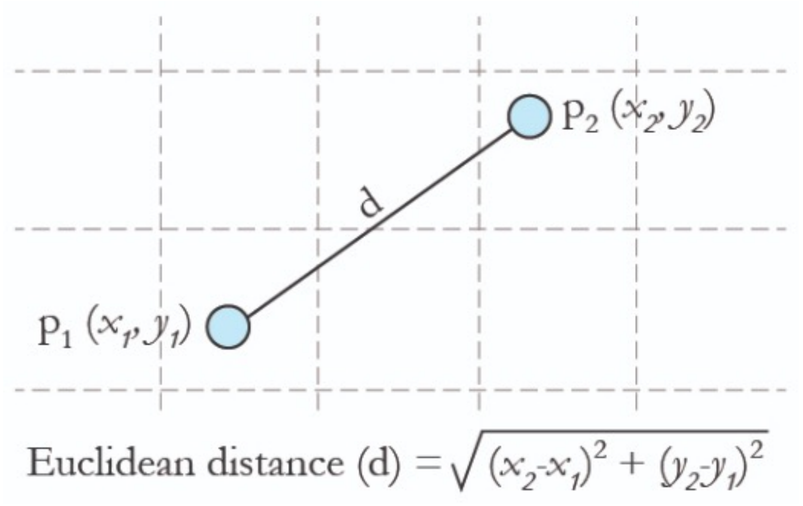
Patterns are defined by distances



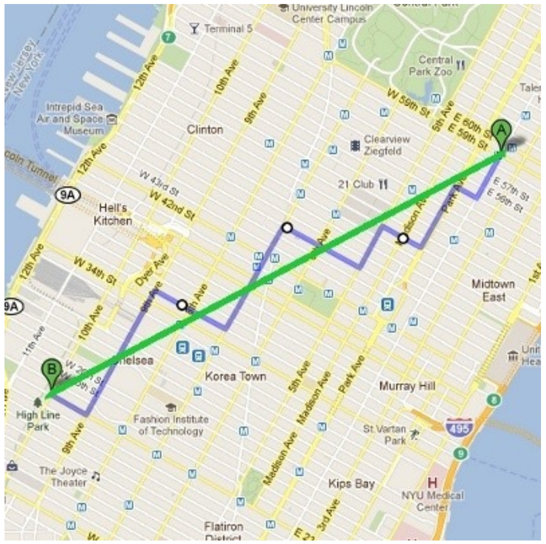


Thick edges = small distances = high similarities

Choices of distance measurement



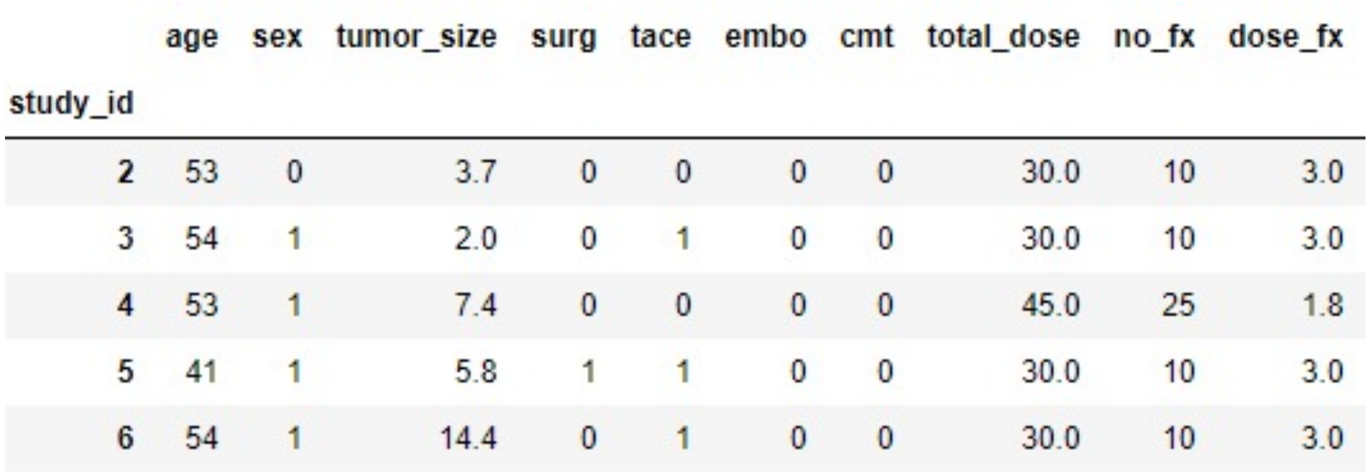
www.tutorialexample.com



www.quora.com

- Euclidean distance = $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
- Manhattan distance = $|x_1 - x_2| + |y_1 - y_2|$
- Pearson and Spearman correlation coefficients
- Cosine similarity = $\frac{\overrightarrow{u_1} \cdot \overrightarrow{u_2}}{|u_1||u_2|} = \frac{x_1 x_2 + y_1 y_2}{\sqrt{{x_1}^2 + {y_1}^2}\sqrt{{x_2}^2 + {y_2}^2}}$

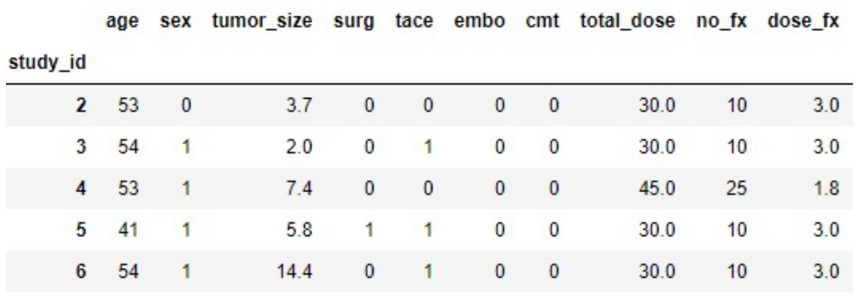
Features can have different scales and units



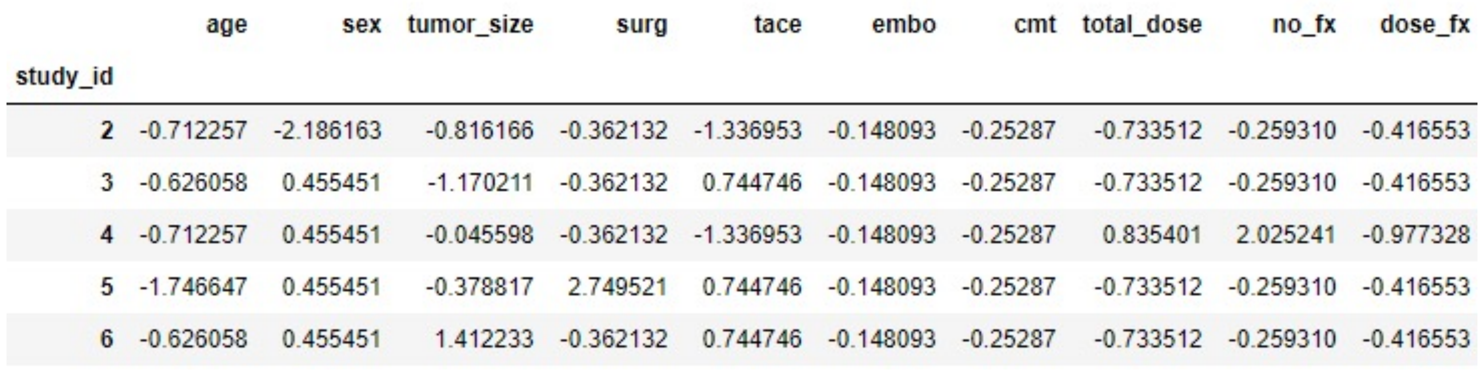
	age	sex	tumor_size	surg	tace	embo	cmt	total_dose	no_fx	dose_fx
study_id										
2	53	0	3.7	0	0	0	0	30.0	10	3.0
3	54	1	2.0	0	1	0	0	30.0	10	3.0
4	53	1	7.4	0	0	0	0	45.0	25	1.8
5	41	1	5.8	1	1	0	0	30.0	10	3.0
6	54	1	14.4	0	1	0	0	30.0	10	3.0

- $d_{\mathrm{Euclidean}}(\mathrm{p}_1, \mathrm{p}_4) = \sqrt{(53-41)^2 + (0-1)^2 + \cdots + (3-3)^2}$
- What would correlation between these data look like?

Data standardization



study_id	age	sex	tumor_size	surg	tace	embo	cmt	total_dose	no_fx	dose_fx
2	53	0	3.7	0	0	0	0	30.0	10	3.0
3	54	1	2.0	0	1	0	0	30.0	10	3.0
4	53	1	7.4	0	0	0	0	45.0	25	1.8
5	41	1	5.8	1	1	0	0	30.0	10	3.0
6	54	1	14.4	0	1	0	0	30.0	10	3.0

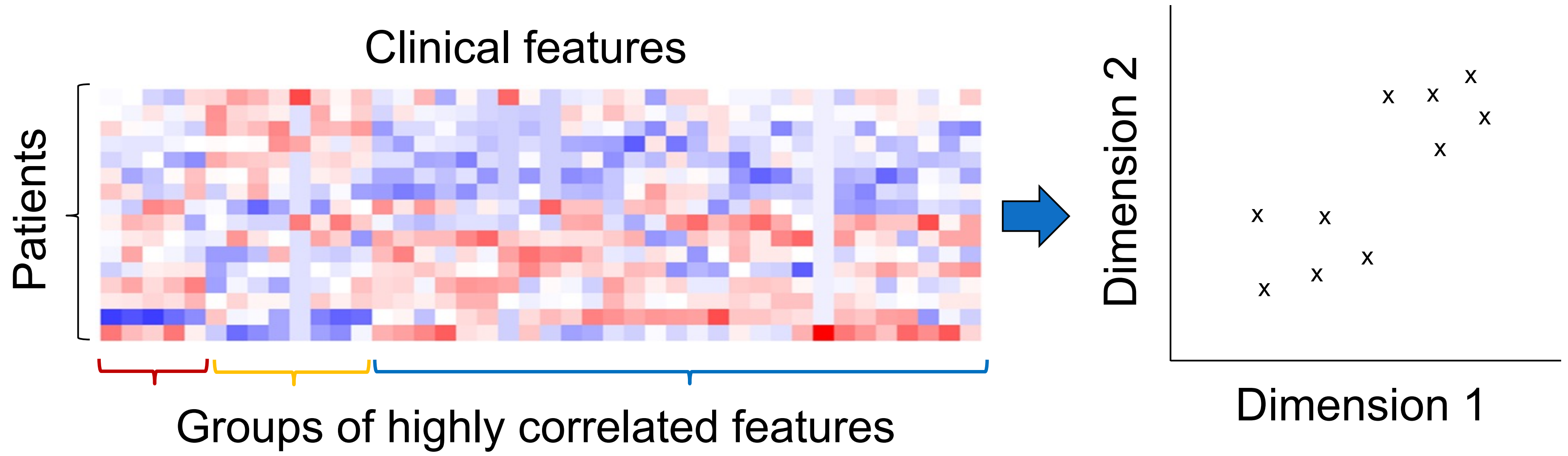


study_id	age	sex	tumor_size	surg	tace	embo	cmt	total_dose	no_fx	dose_fx
2	-0.712257	-2.186163	-0.816166	-0.362132	-1.336953	-0.148093	-0.25287	-0.733512	-0.259310	-0.416553
3	-0.626058	0.455451	-1.170211	-0.362132	0.744746	-0.148093	-0.25287	-0.733512	-0.259310	-0.416553
4	-0.712257	0.455451	-0.045598	-0.362132	-1.336953	-0.148093	-0.25287	0.835401	2.025241	-0.977328
5	-1.746647	0.455451	-0.378817	2.749521	0.744746	-0.148093	-0.25287	-0.733512	-0.259310	-0.416553
6	-0.626058	0.455451	1.412233	-0.362132	0.744746	-0.148093	-0.25287	-0.733512	-0.259310	-0.416553

- $x_{\text{standardized}} = \dfrac{x - \text{mean}}{\text{s.d.}}$ $\qquad \text{s.d.} = \sqrt{\dfrac{\sum (x_i - \text{mean})^2}{N}}$

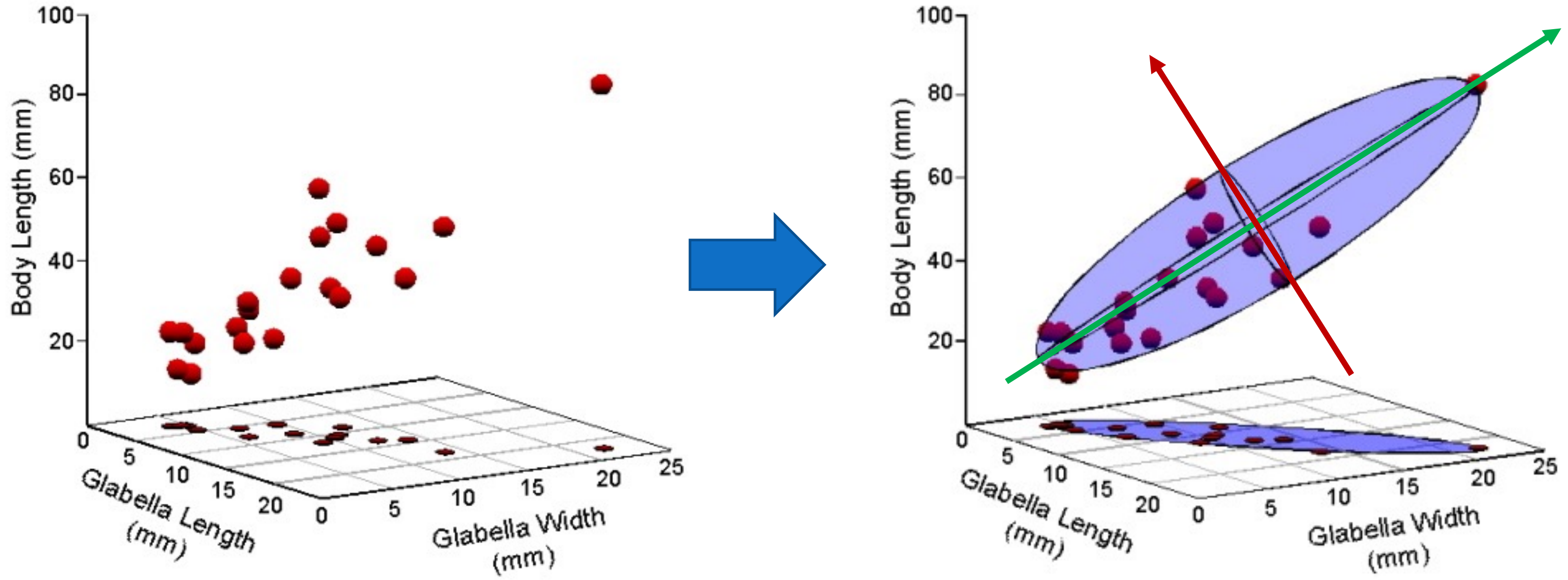
Dimensionality reduction

Dimensionality reduction



- Highly correlated / redundant features should be collapsed
- The major pattern (two groups of patients) can be sufficiently represented with just a few features
- Visualization on 2D or 3D for inspection by human

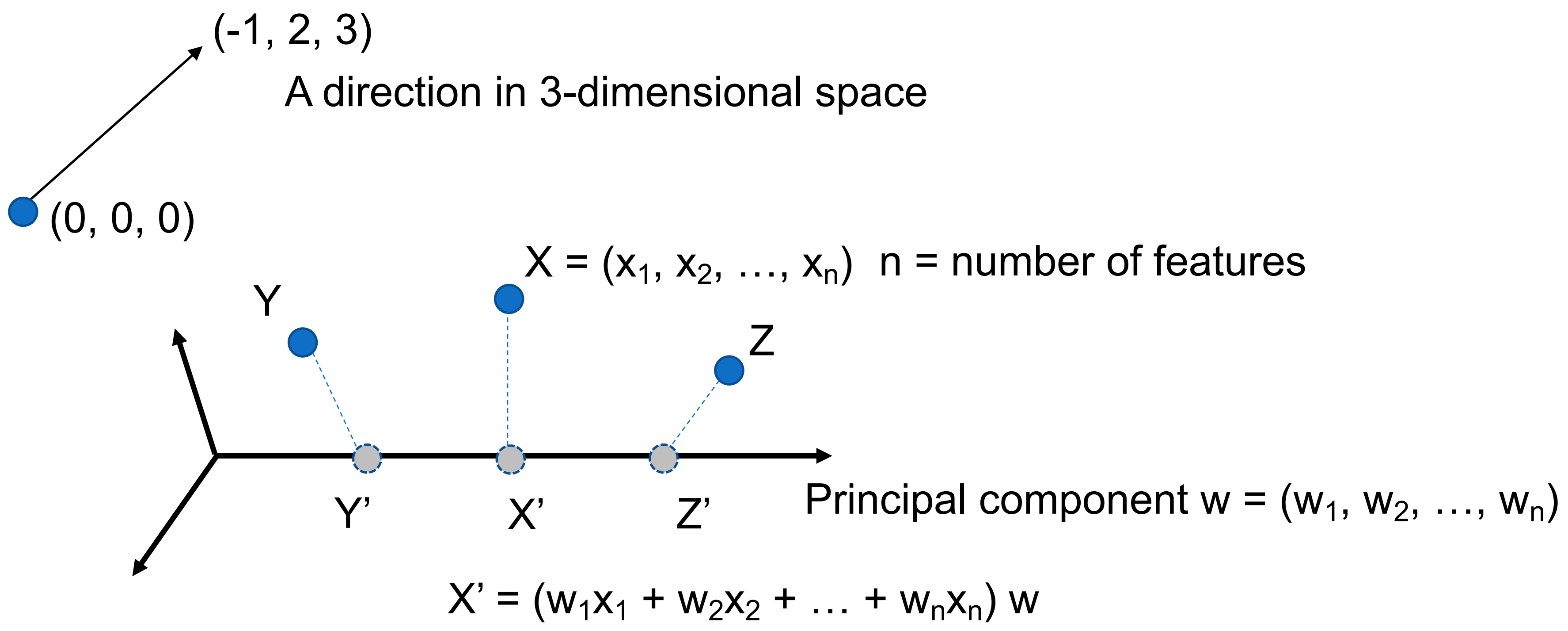
Principal component analysis (PCA)



Source: the paleontological association

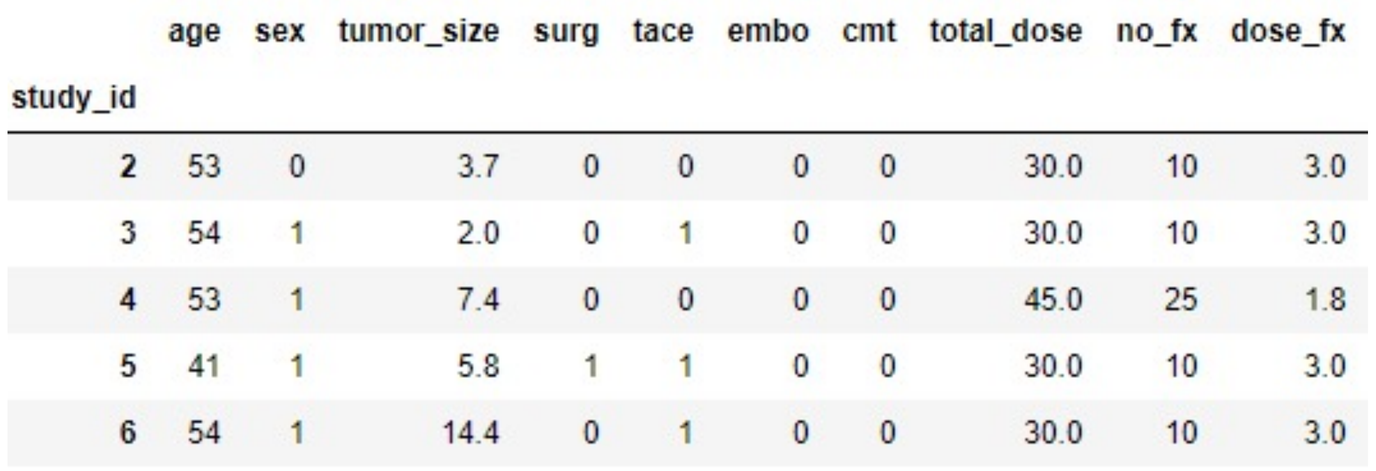
- Fit an *n*-dimensional ellipsoid to the data cloud
  - Essentially rotation of original axes
  - Preserve Euclidean distance between all data points
- Sort dimensions based on data variances

Mapping data onto a principal component


(-1, 2, 3)
A direction in 3-dimensional space
(0, 0, 0)
$X = (x_1, x_2, \ldots, x_n)$ n = number of features
Y
Z
Y'
X'
Z'
Principal component $w = (w_1, w_2, \ldots, w_n)$
$X' = (w_1x_1 + w_2x_2 + \ldots + w_nx_n)\, w$

- Original data belong to an *n*-dimensional space
- A principal component is a direction in *n*-dimensional space and can be characterized by a unit vector $(w_1, \ldots, w_n)$
- X, Y, Z can be projected onto X', Y', Z' on this principal component and the variance can be calculated

PCA favors features with high variances



study_id	age	sex	tumor_size	surg	tace	embo	cmt	total_dose	no_fx	dose_fx
2	53	0	3.7	0	0	0	0	30.0	10	3.0
3	54	1	2.0	0	1	0	0	30.0	10	3.0
4	53	1	7.4	0	0	0	0	45.0	25	1.8
5	41	1	5.8	1	1	0	0	30.0	10	3.0
6	54	1	14.4	0	1	0	0	30.0	10	3.0

- Which feature most strongly influence principal component?
  - **Hint**: The first principal component is the direction that maximize the variance of the data points
  - **Hint**: Variance scales linearly with magnitude of data value

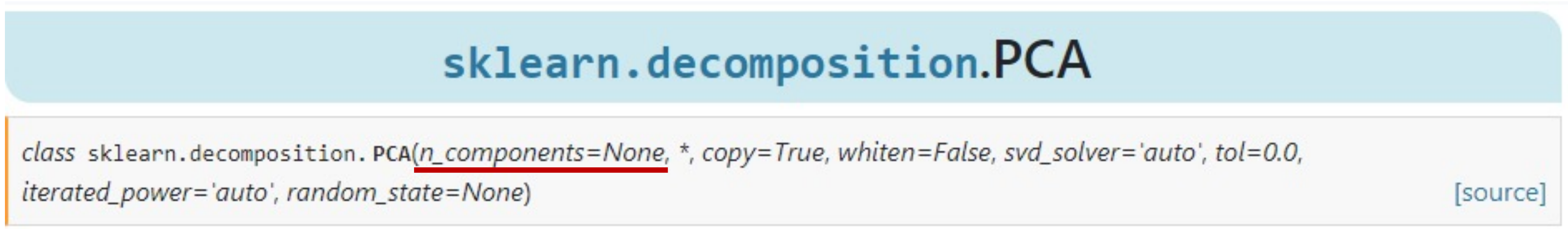
Key behaviors of PCA

- The projection $X' = (w \cdot X)w$ is a linear combination of the original features
  - We can inspect *w* to interpret feature-level contribution
  - *w* = (–10, 1, 0.2, 3) means that the first feature is quite important
  - Cannot capture nonlinear relationship between features

- PCA = rotation of the original axes
  - PCA preserve Euclidean distances between data points
  - Sometimes, Euclidean distance is inappropriate

- Highly correlated features tend to be grouped together in the same principal component
  - PCA can reduce redundancy
  - PCA is a good initial step for other complex algorithms, including t-SNE and UMAP which we will learn later

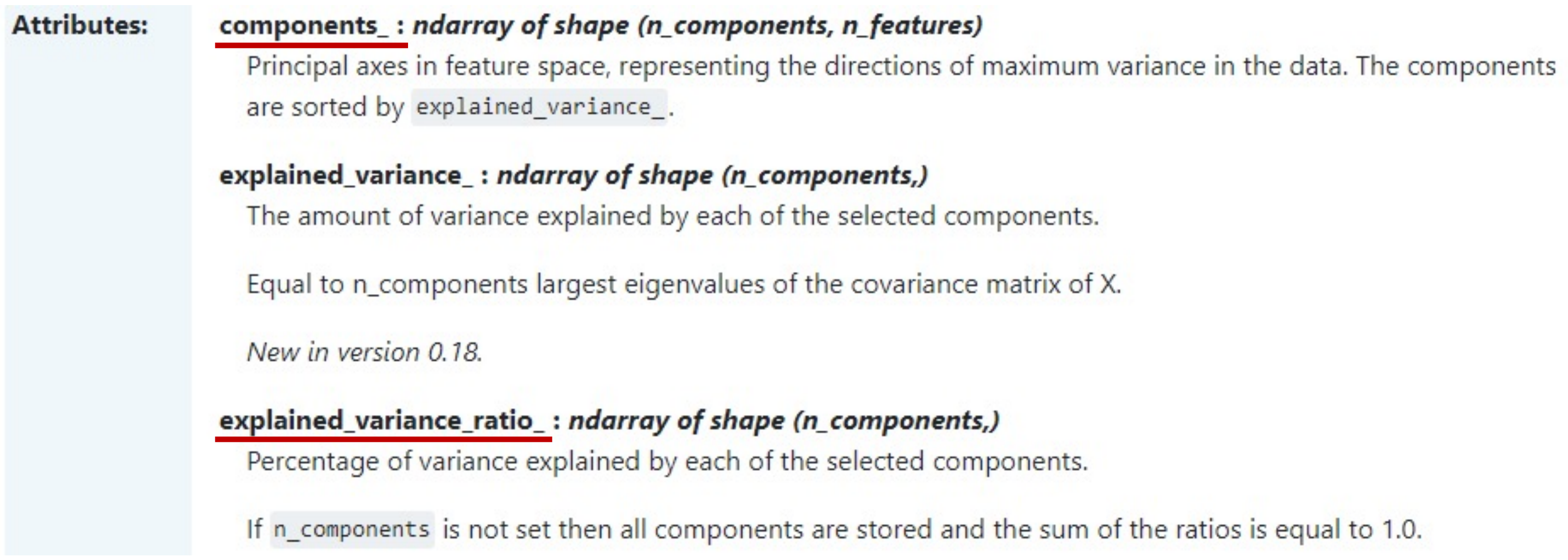
PCA in Python



sklearn.decomposition.PCA

class `sklearn.decomposition.PCA`(*n_components=None*, ***, *copy=True*, *whiten=False*, *svd_solver='auto'*, *tol=0.0*, *iterated_power='auto'*, *random_state=None*) [source]

Returned values that you can use



Attributes:

components_ : ***ndarray of shape (n_components, n_features)***

Principal axes in feature space, representing the directions of maximum variance in the data. The components are sorted by `explained_variance_`.

explained_variance_ : ***ndarray of shape (n_components,)***

The amount of variance explained by each of the selected components.

Equal to n_components largest eigenvalues of the covariance matrix of X.

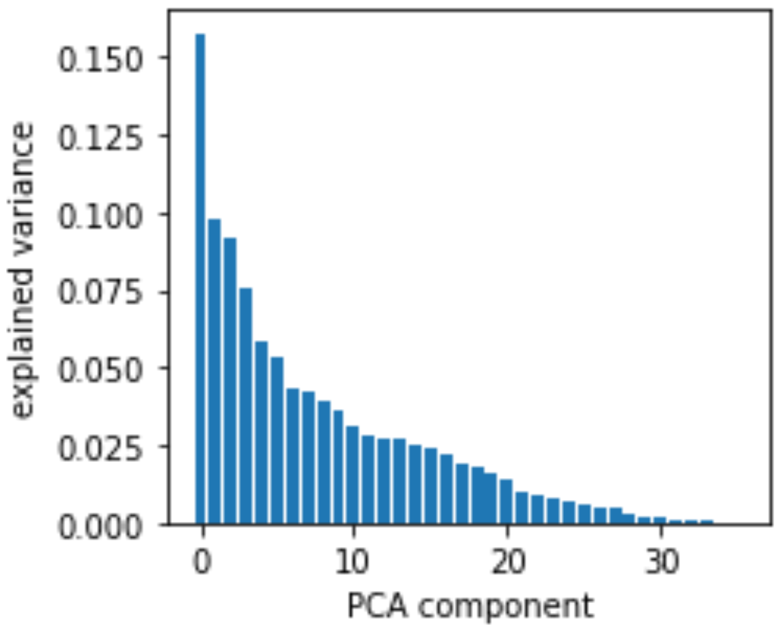
New in version 0.18.

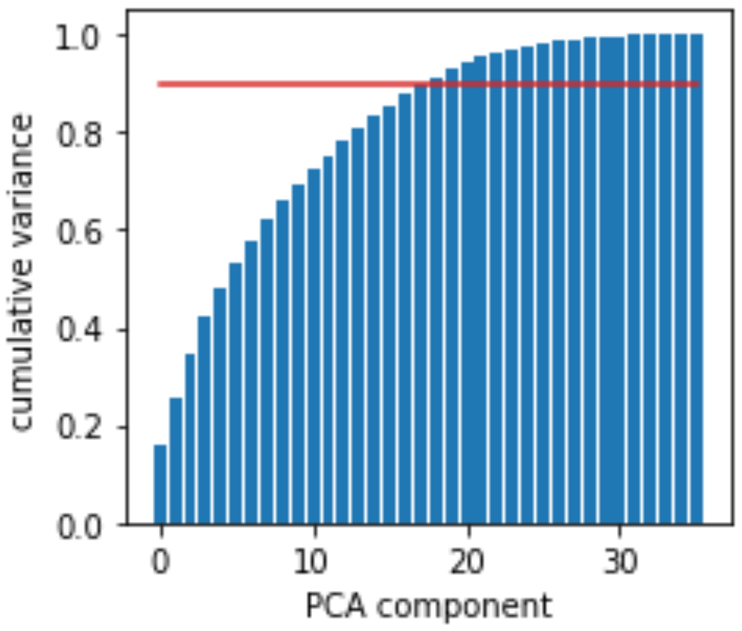
explained_variance_ratio_ : ***ndarray of shape (n_components,)***

Percentage of variance explained by each of the selected components.

If `n_components` is not set then all components are stored and the sum of the ratios is equal to 1.0.

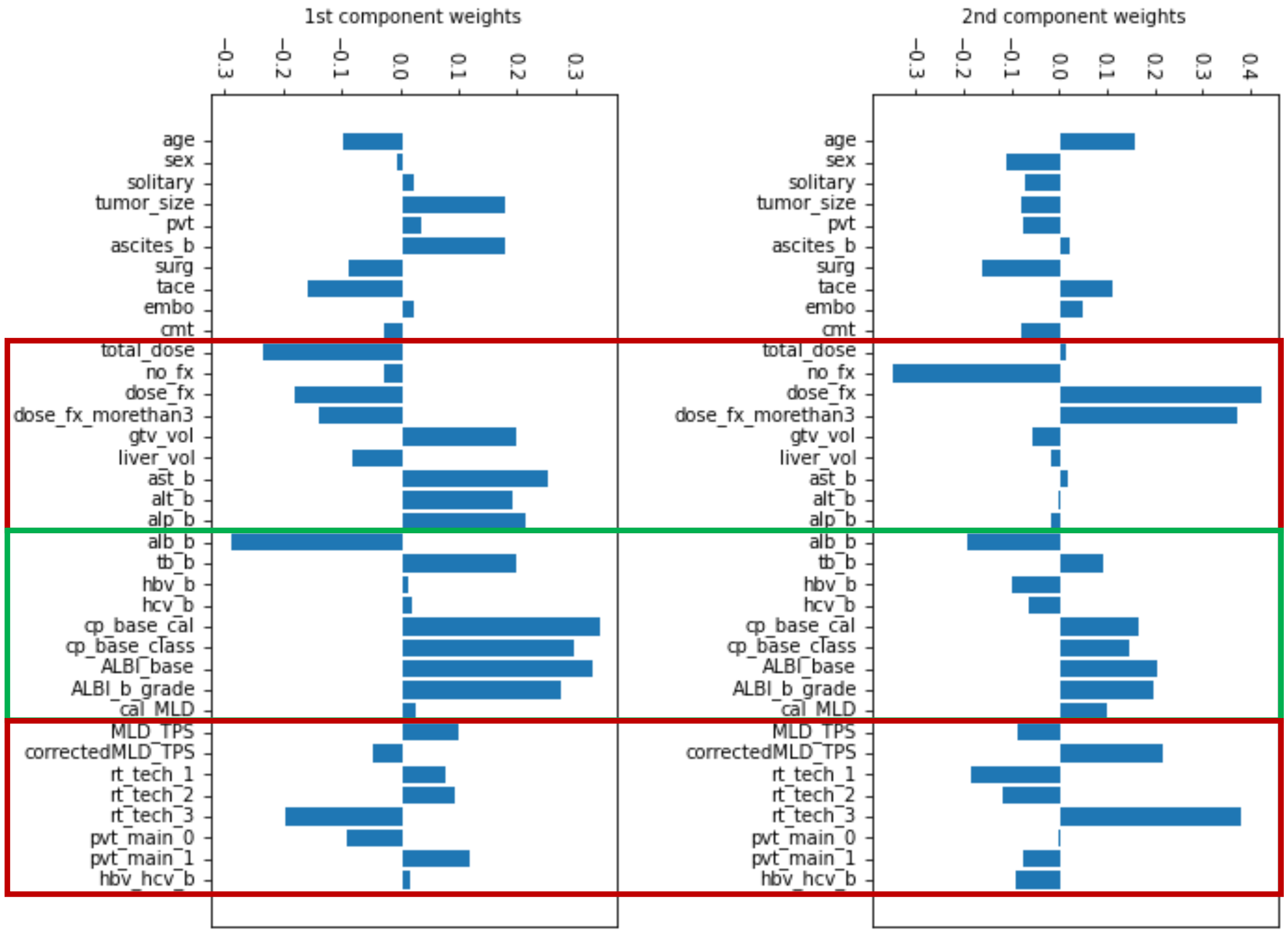
Explained variance ratio



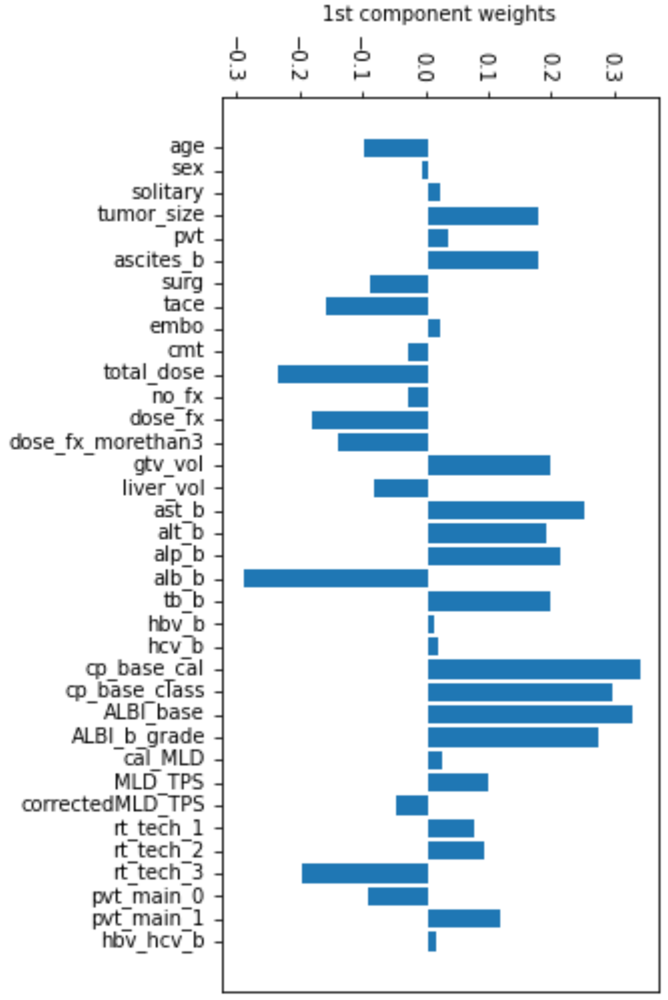


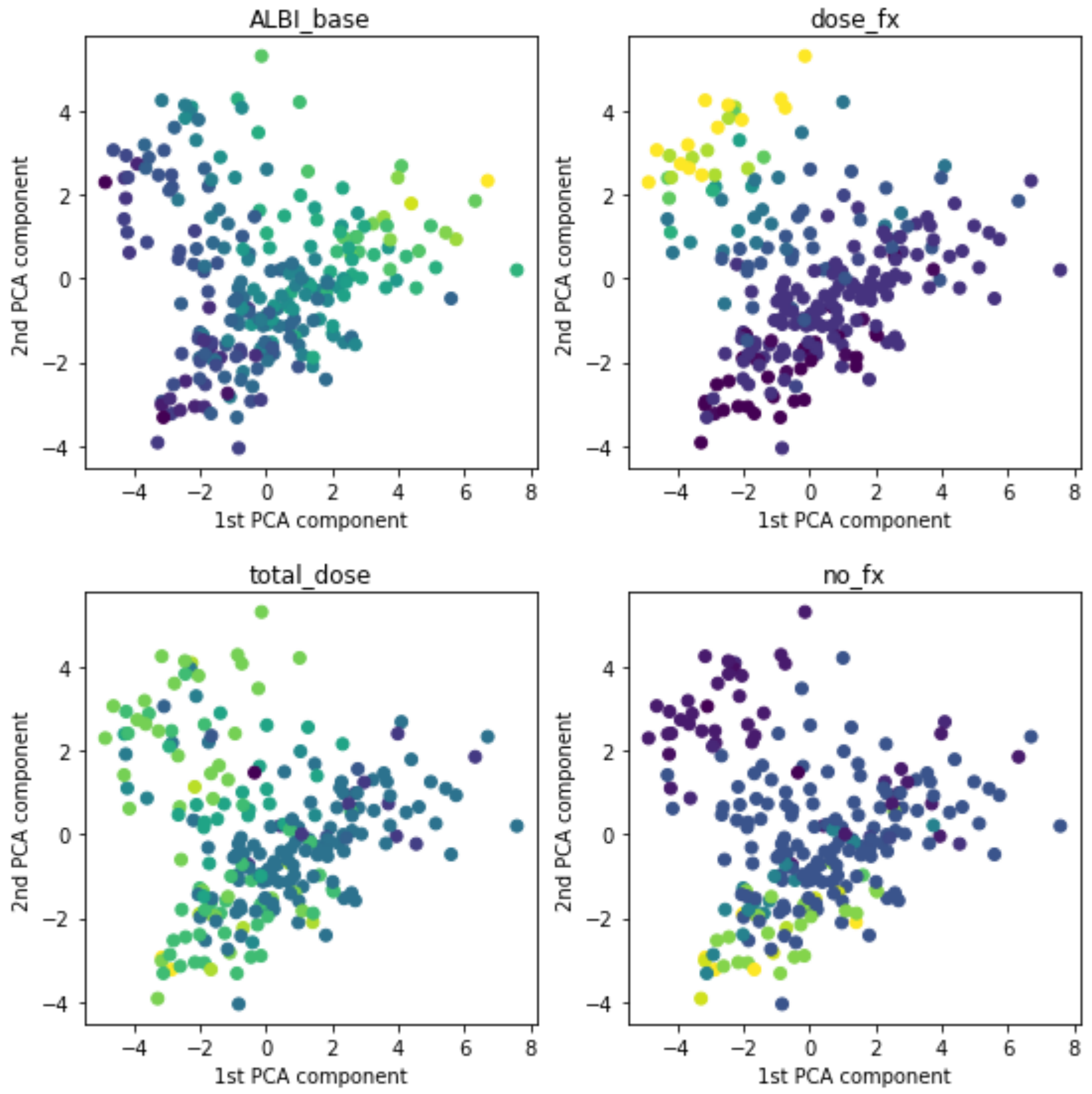
- Components that capture high variances are typically useful for explaining the data, but not always
- Cumulative variance suggests the true dimension of data
  - Use first *k* PCA components as input for downstream analyses

Principal component’s loadings, *w*

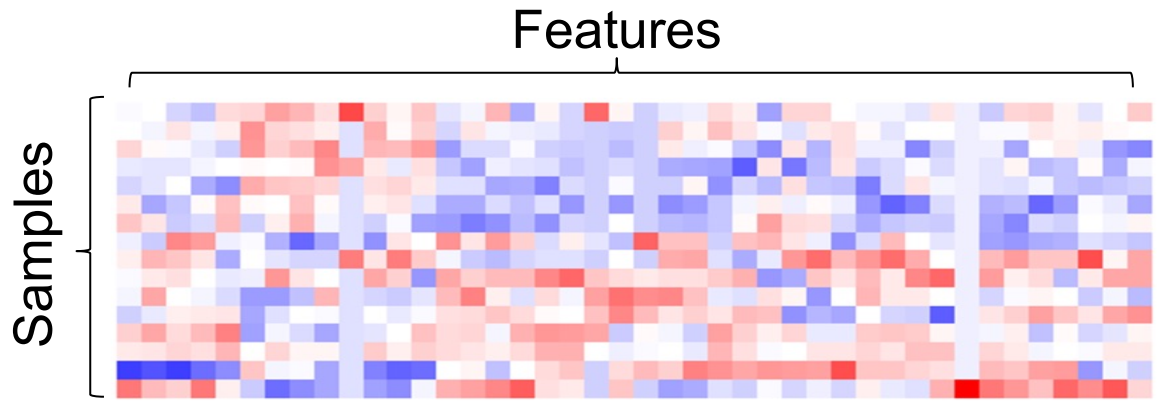


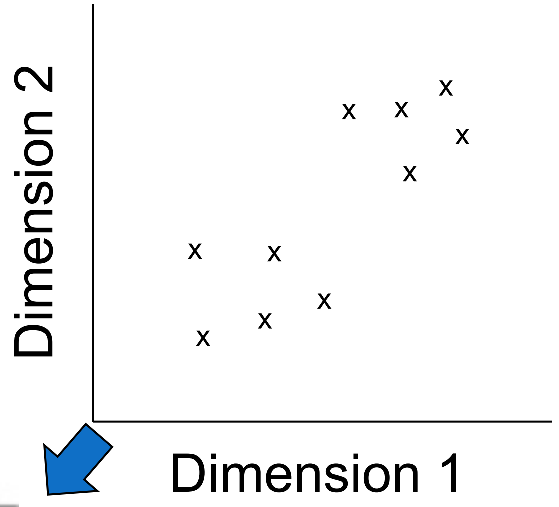
PCA-transformed data



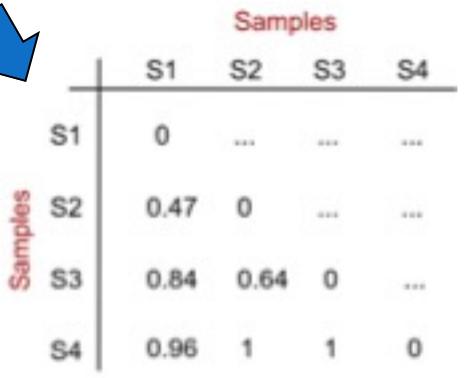


Multidimensional scaling (MDS)

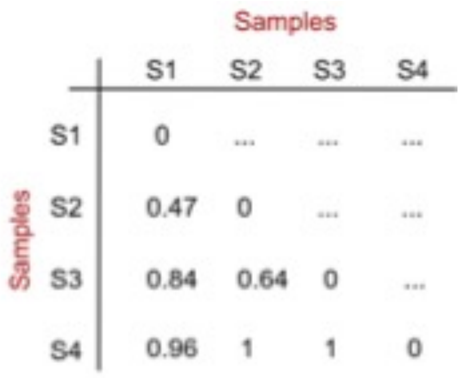




Any distance



Samples	S1	S2	S3	S4
S1	0	...	...	...
S2	0.47	0	...	...
S3	0.84	0.64	0	...
S4	0.96	1	1	0



Samples	S1	S2	S3	S4
S1	0	...	...	...
S2	0.47	0	...	...
S3	0.84	0.64	0	...
S4	0.96	1	1	0

Euclidean distance

Maximize similarity

- MDS projects data points onto new dimensions while trying to preserve the similarity between two distance matrices
  - Maximize Pearson or Spearman correlation

Principal Coordinate Analysis (PCoA)

- Also called Classical MDS

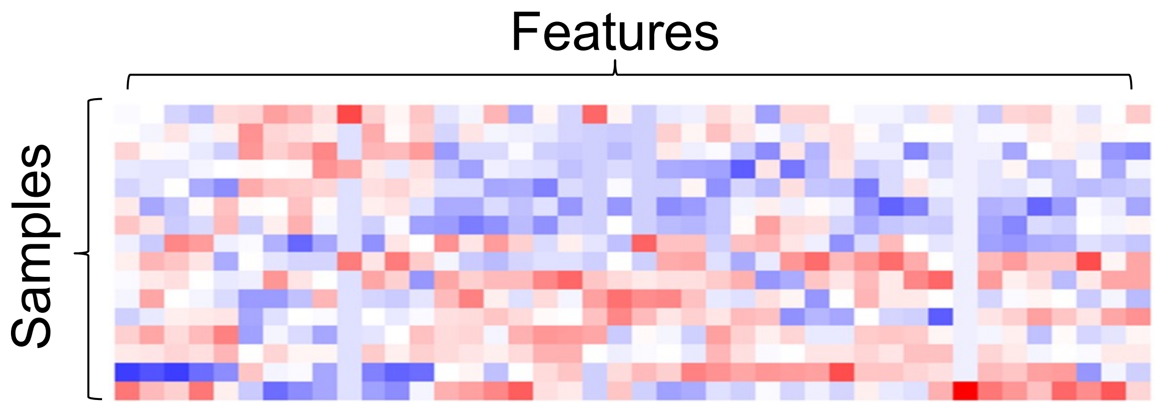
- Users provide distance matrix $d(X_i, X_j)$

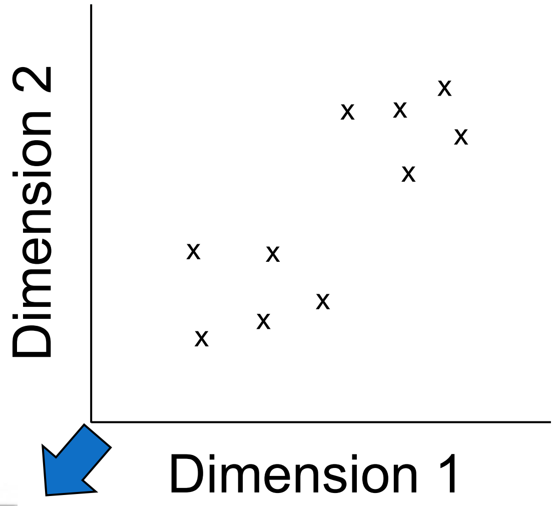
- Let Y_i be a projection of X_i onto a new *k*-dimensional space
  - This induces Euclidean distances $d_{\text{Euclidean}}(Y_i, Y_j)$

- Maximize similarity (*strain*) between $d(X_i, X_j)$ and $d_{\text{Euclidean}}(Y_i, Y_j)$ as a function of Y_i's

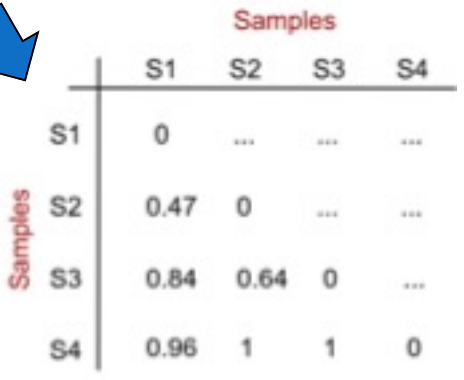
- In practice, finding Y_i's is related to finding the eigenvectors and eigenvalues of some matrix related to $d(X_i, X_j)$

Non-classical MDS

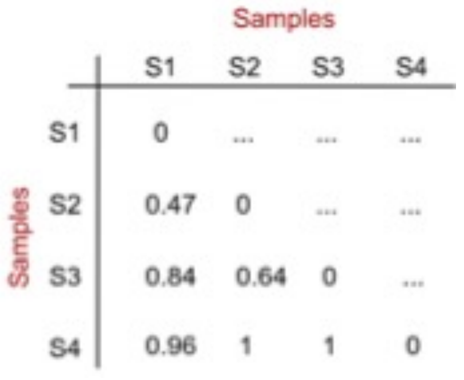




Any distance



	S1	S2	S3	S4
S1	0	...	...	...
S2	0.47	0	...	...
S3	0.84	0.64	0	...
S4	0.96	1	1	0



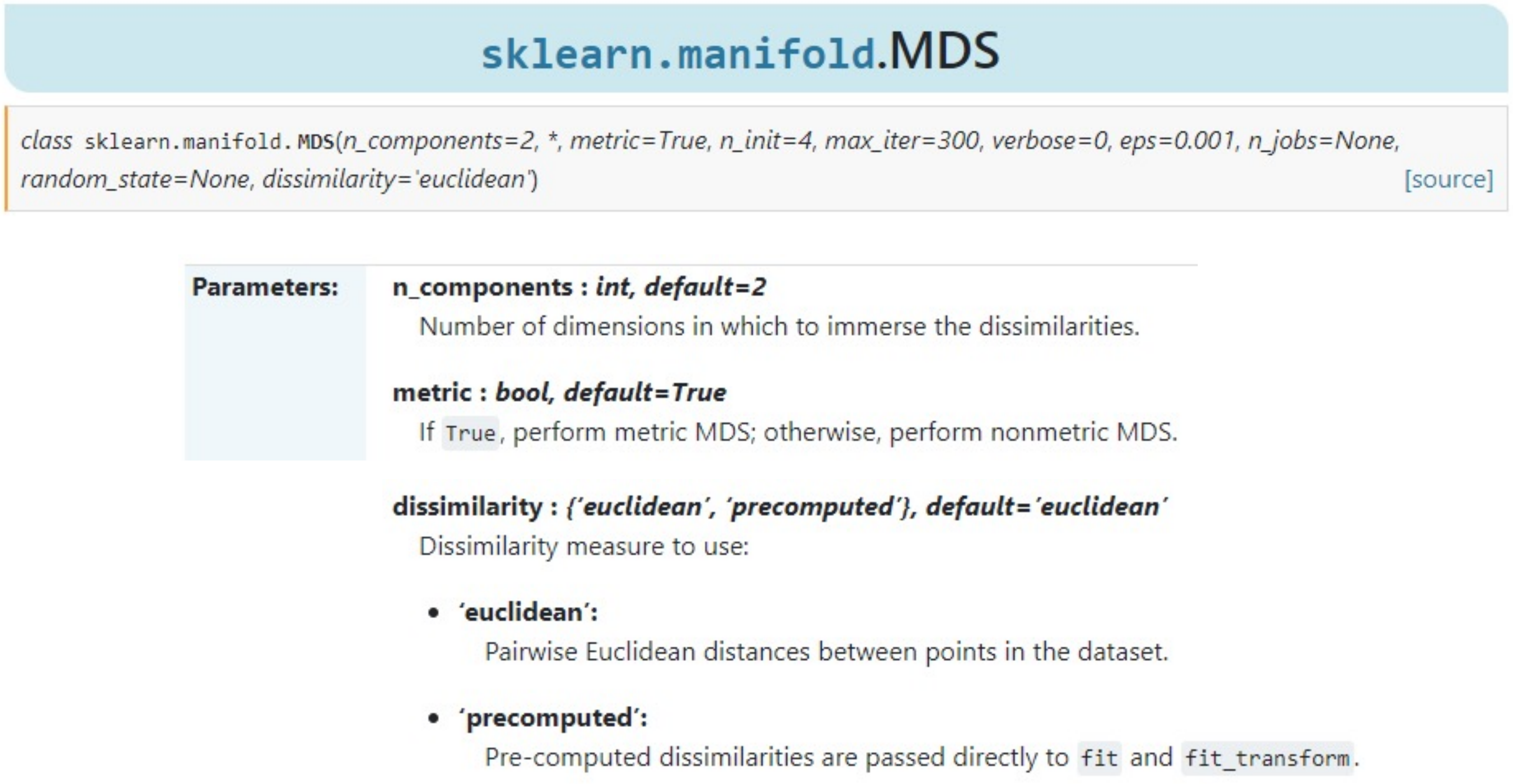
	S1	S2	S3	S4
S1	0	...	...	...
S2	0.47	0	...	...
S3	0.84	0.64	0	...
S4	0.96	1	1	0

Non-Euclidean distance

Maximize Spearman correlation

- Cannot be explicitly solved
- Use heuristic search or gradient descent instead

MDS in Python



sklearn.manifold.MDS

class sklearn.manifold.MDS(*n_components=2*, ***, *metric=True*, *n_init=4*, *max_iter=300*, *verbose=0*, *eps=0.001*, *n_jobs=None*, *random_state=None*, *dissimilarity='euclidean'*) [source]

Parameters:

n_components : *int, default=2*
Number of dimensions in which to immerse the dissimilarities.

metric : *bool, default=True*
If `True`, perform metric MDS; otherwise, perform nonmetric MDS.

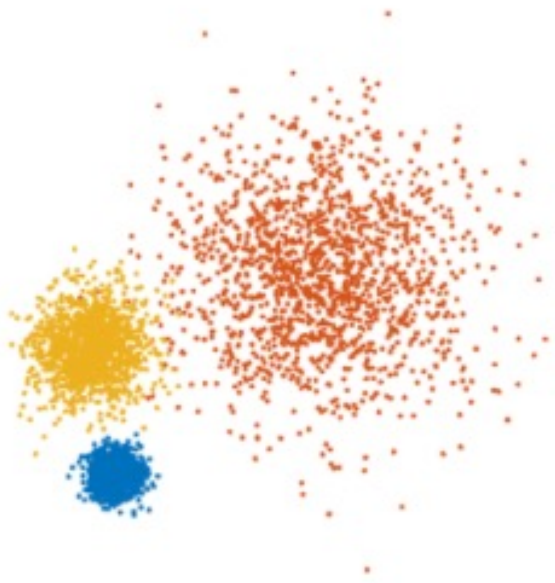
dissimilarity : *{'euclidean', 'precomputed'}, default='euclidean'*
Dissimilarity measure to use:

- **'euclidean':**
  Pairwise Euclidean distances between points in the dataset.
- **'precomputed':**
  Pre-computed dissimilarities are passed directly to `fit` and `fit_transform`.

- Default (metric = True) is Principal Coordinate Analysis

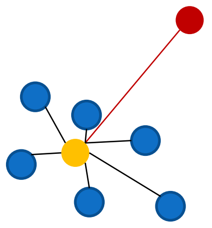
Advanced dimensionality reduction

Limitation of PCA and MDS

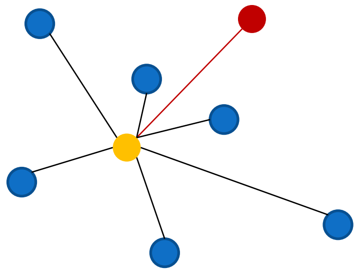


- PCA does not affect Euclidean distance
- MDS preserves the whole pairwise distance matrix uniformly
  - Distance of *d* between two data points X_i and X_j has the same importance as distance of *d* between X_n and X_m
  - What if different groups of data points were generated from different underlying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

Stochastic Neighbor Embedding (SNE)



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



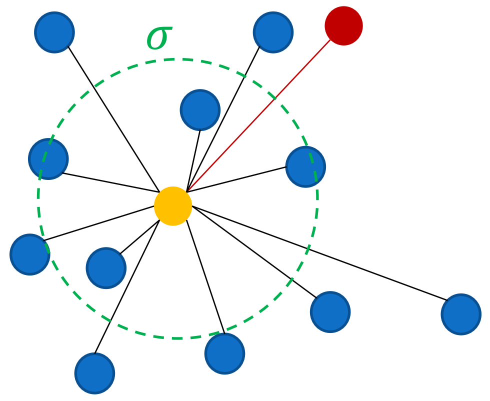
$$= \frac{e^{-\frac{(\mathrm{dist}(o,o))^2}{2\sigma^2}} / \sigma}{\sum e^{-\frac{(\mathrm{dist}(o,o))^2}{2\sigma^2}} / \sigma}$$

o = other data points

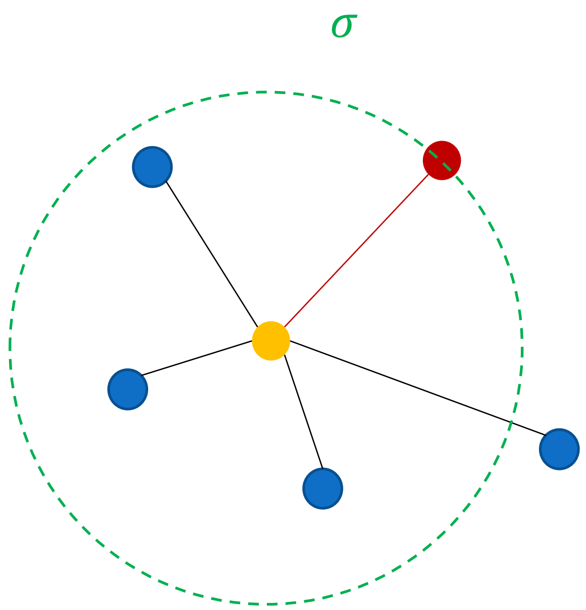
similarity(o, o) = [score(o | o) + score(o | o)] / 2 * # data points

- Turn distance into probability of being neighbor
- Normalization over other data points turn absolute distance into relative distance
- Parameter σ handles the difference in data density
  - Called perplexity here

Perplexity



Many data points in proximity
Small σ

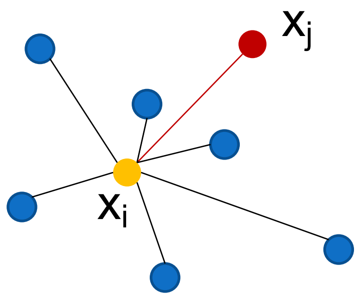


Few data points in proximity
Large σ

- 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

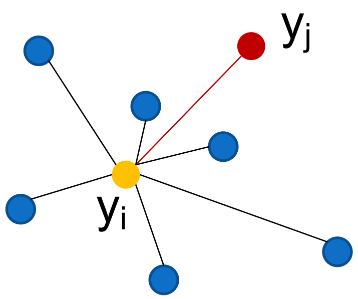
Embedding into low dimension

High-dimension



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

Low-dimension



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

- 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*
  - $D_{KL}(P \,||\, Q) = -\sum_i \sum_j p_{j|i} \log_2 \frac{p_{j|i}}{q_{j|i}}$
- KL gives more attention to data pairs with high $p_{j|i}$

Gradient descent

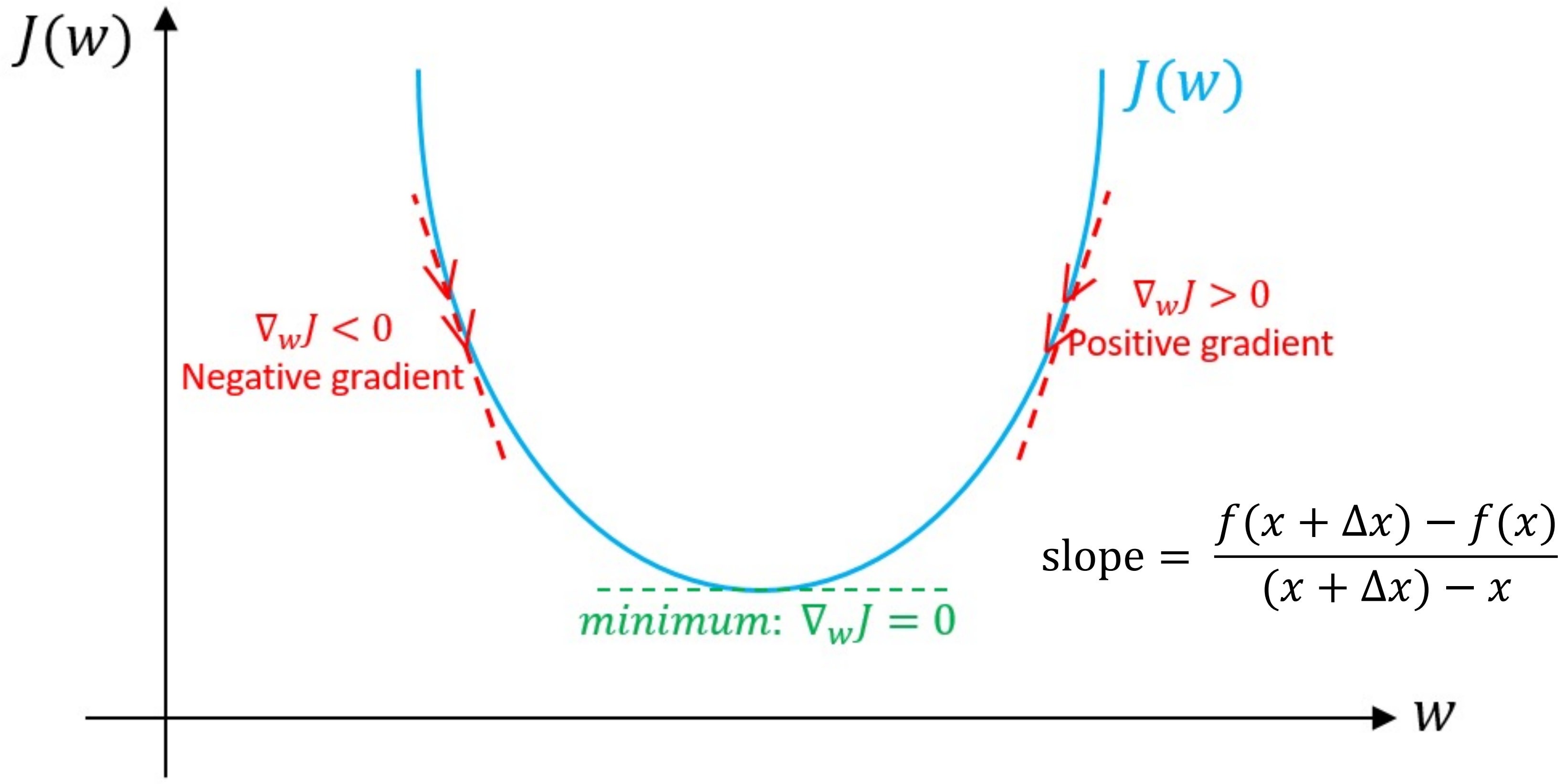

$J(w)$
$J(w)$
$\nabla_w J < 0$
Negative gradient
$\nabla_w J > 0$
Positive gradient
minimum: $\nabla_w J = 0$
$\text{slope} = \frac{f(x+\Delta x) - f(x)}{(x+\Delta x) - x}$
w

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

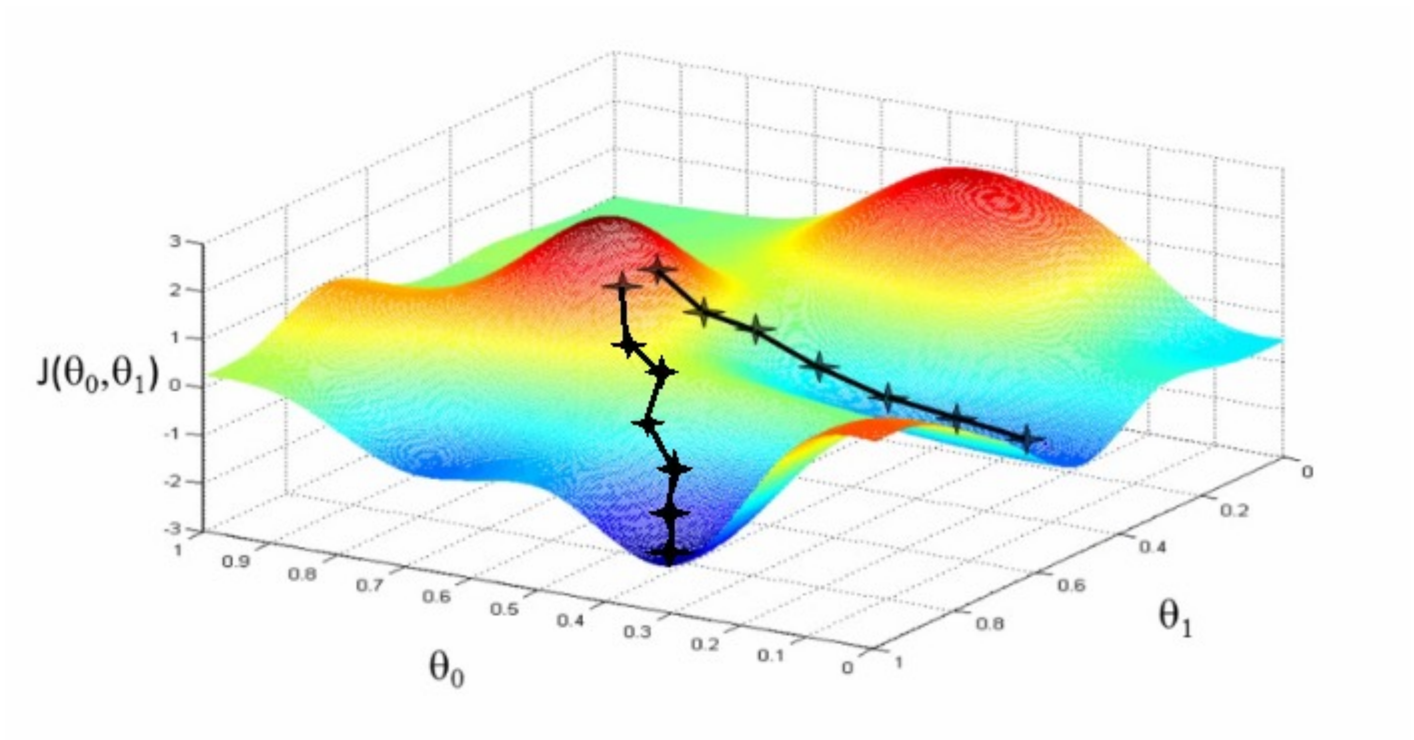


Image from shashank-ojha.github.io

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

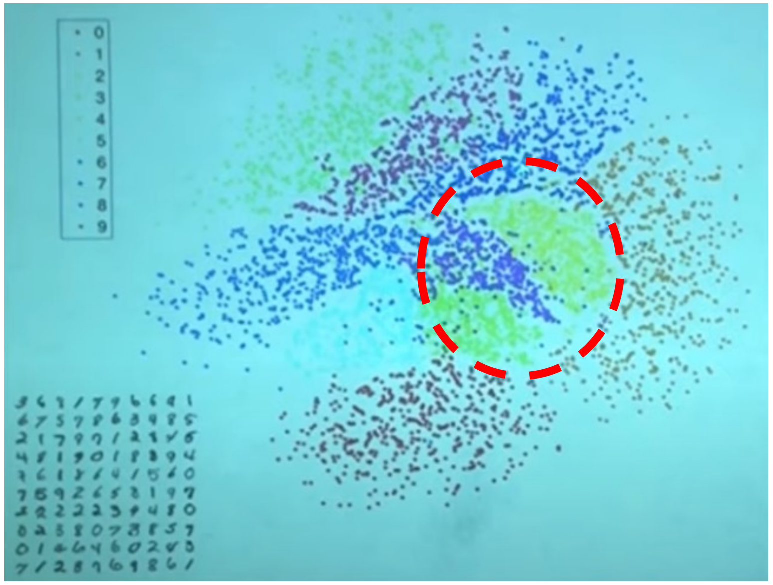
Gradient descent for SNE

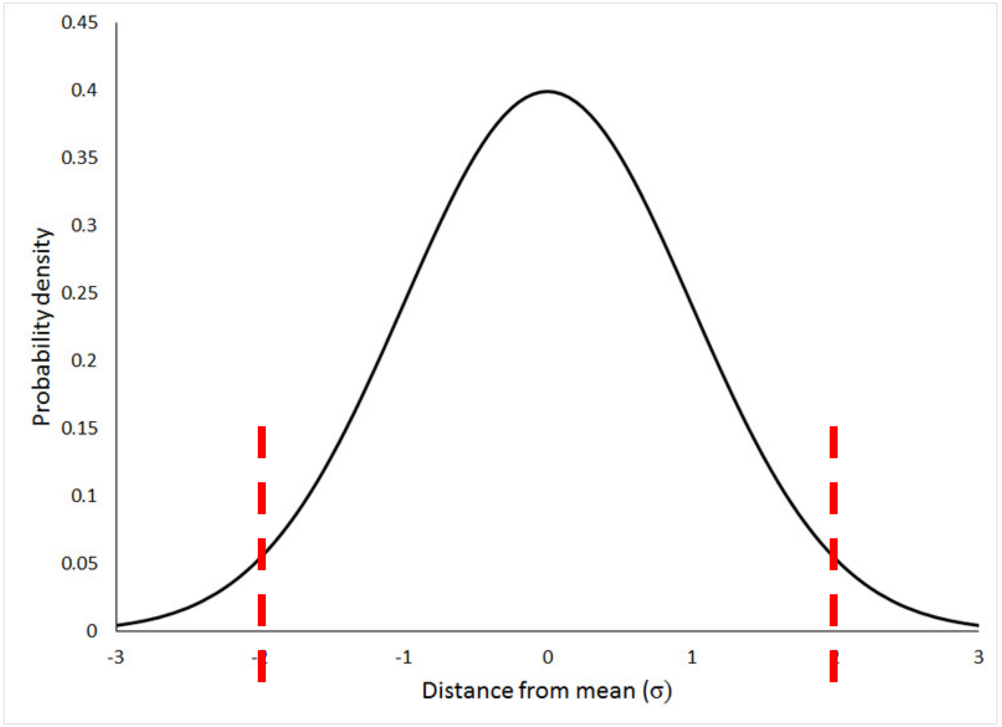
- $D_{KL}(P\,||\,Q) = -\sum_i \sum_j p_{j|i} \log_2 \frac{p_{j|i}}{q_{j|i}}, \;\; q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$

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

- The gradient is surprisingly simple!
  - $\frac{\delta D_{KL}}{\delta y_i} = 2 \sum_{j \neq i} \left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)\left(y_i - y_j\right)$
  - This tells us the direction to move y_i to reduce the divergence
  - Depends on relative positions of y_i with other points y_j

Limitation of the original SNE





- Gaussian distribution is very narrow at the tails
  - $q_{j|i}$ diminishes 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

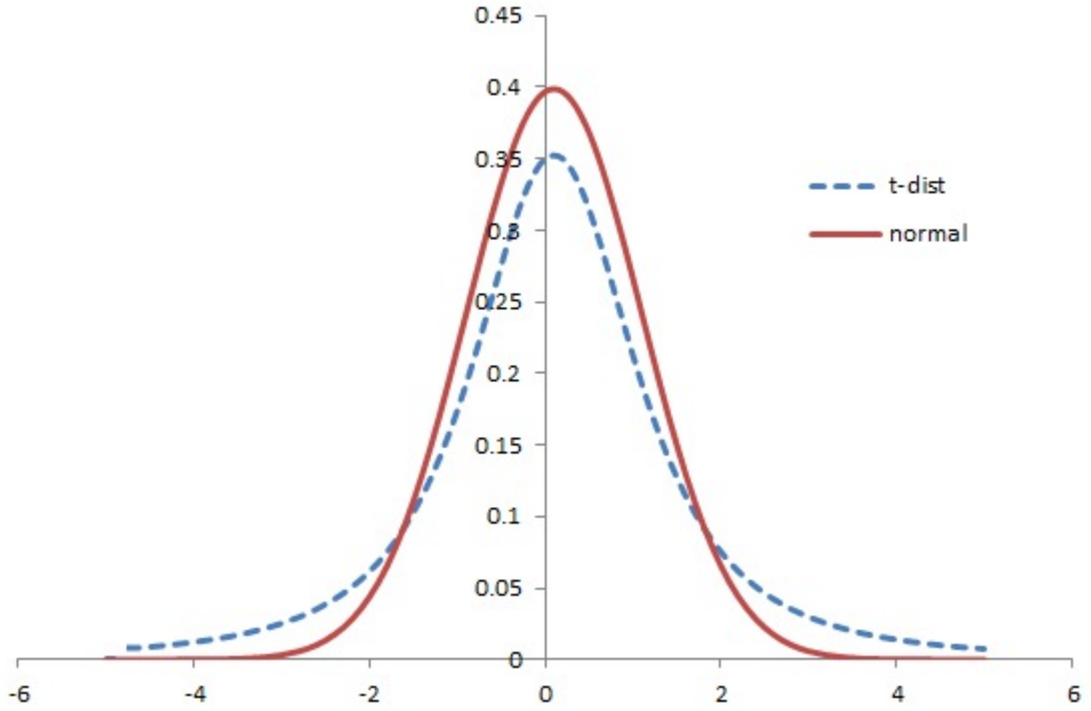
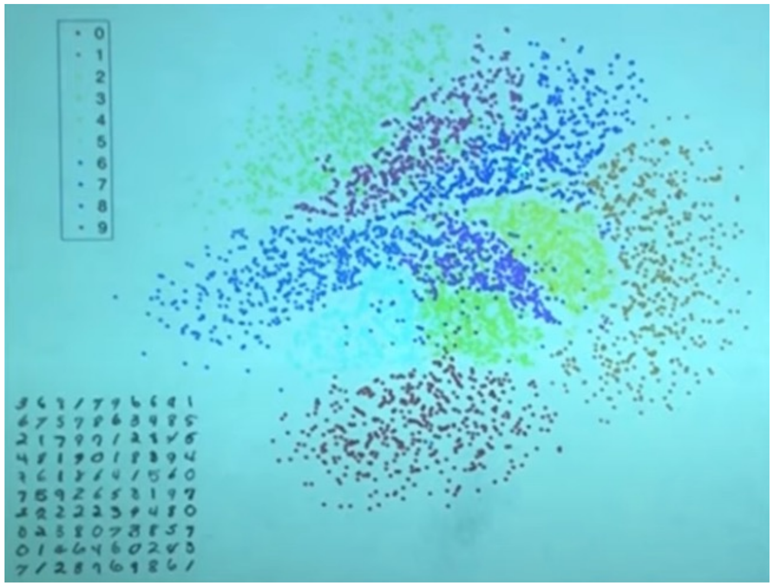
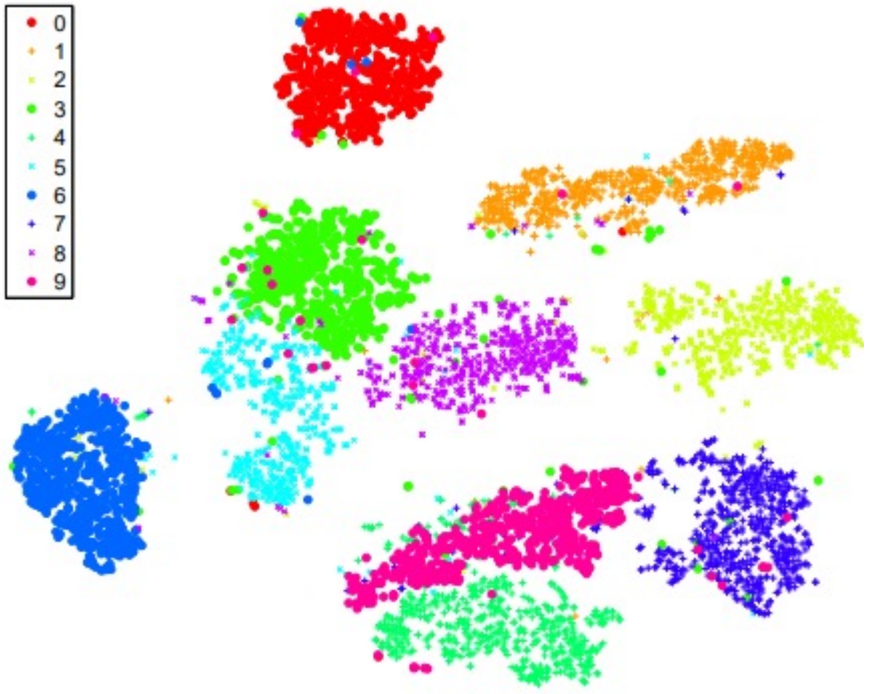


Image from riskprep.com

- 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 $q_{j|i} = \frac{\left(1+\|y_i-y_j\|^2\right)^{-1}}{\sum_{k\neq i}(1+\|y_i-y_k\|^2)^{-1}}$

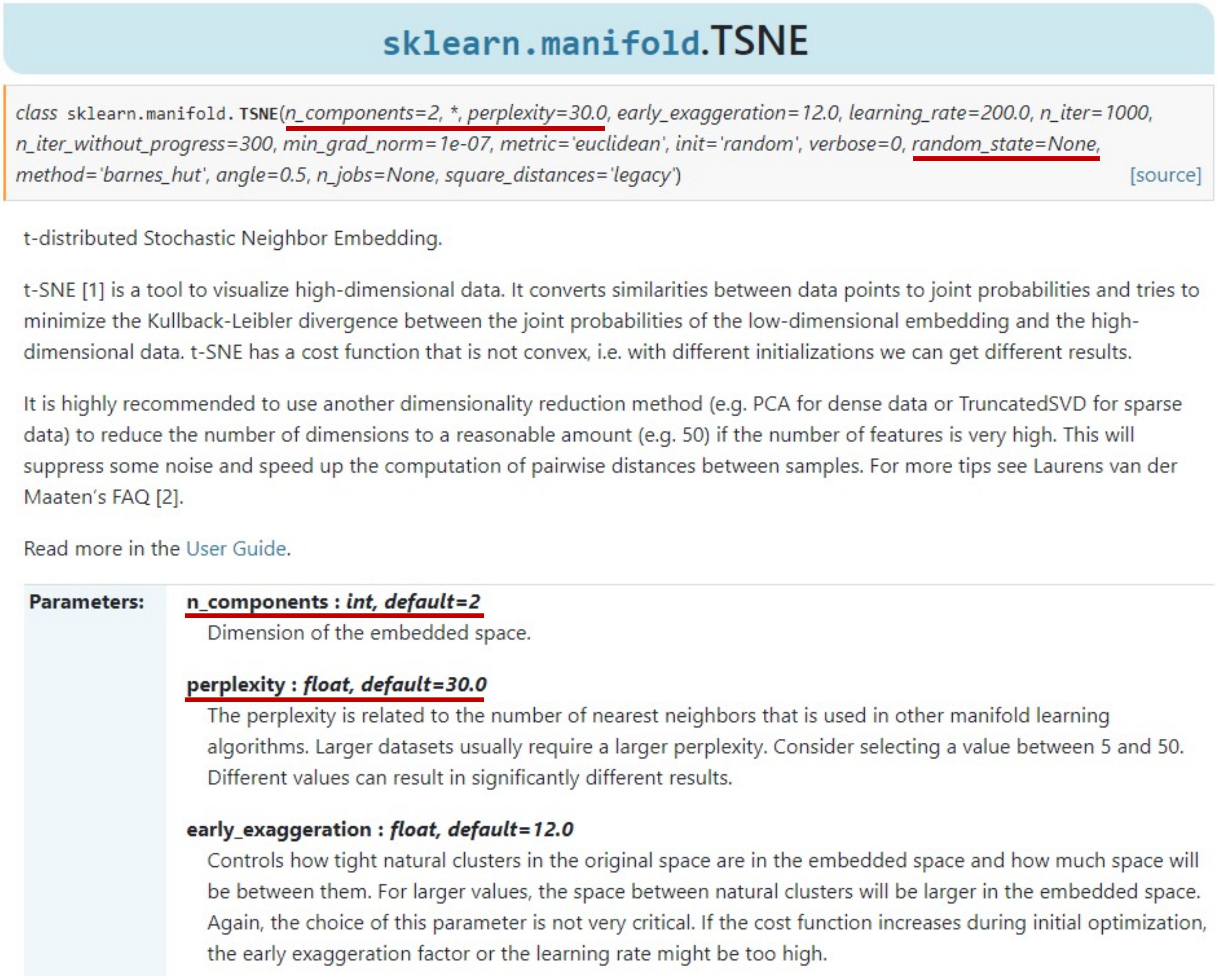
t-distribution spreads out the data points





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:

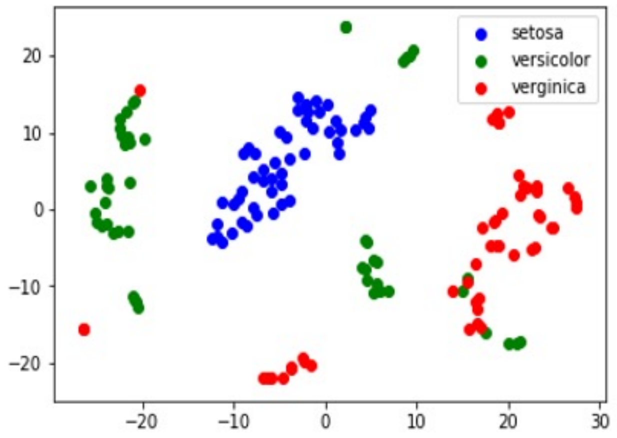
n_components : *int, default=2*
Dimension of the embedded space.

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.

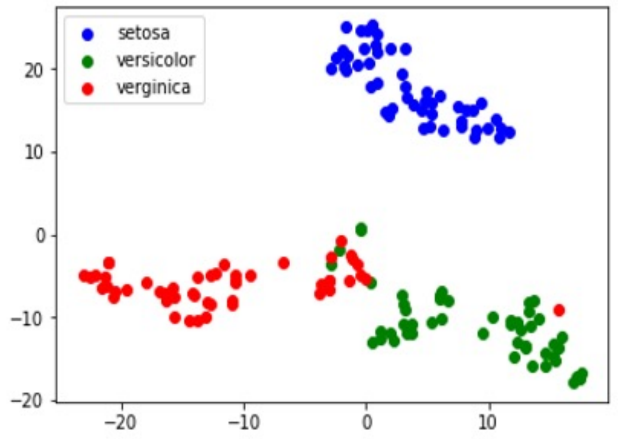
early_exaggeration : *float, default=12.0*
Controls how tight natural clusters in the original space are in the embedded space and how much space will be between them. For larger values, the space between natural clusters will be larger in the embedded space. Again, the choice of this parameter is not 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

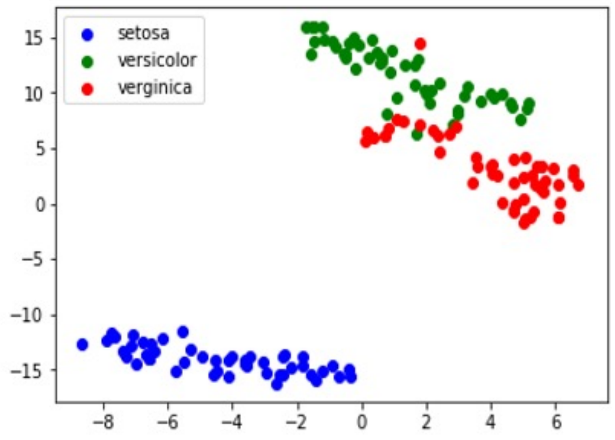
K = 5



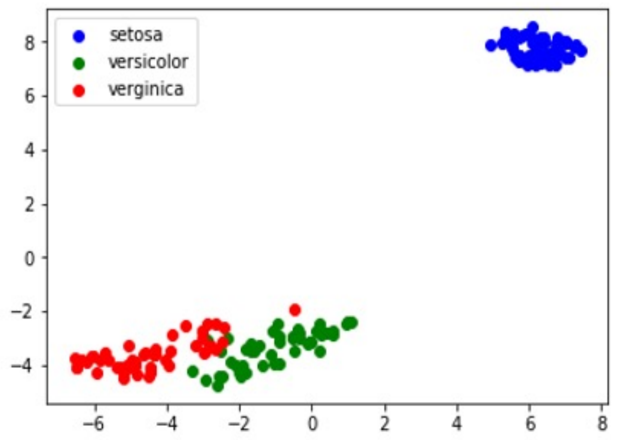
K = 10



K = 25



K = 50



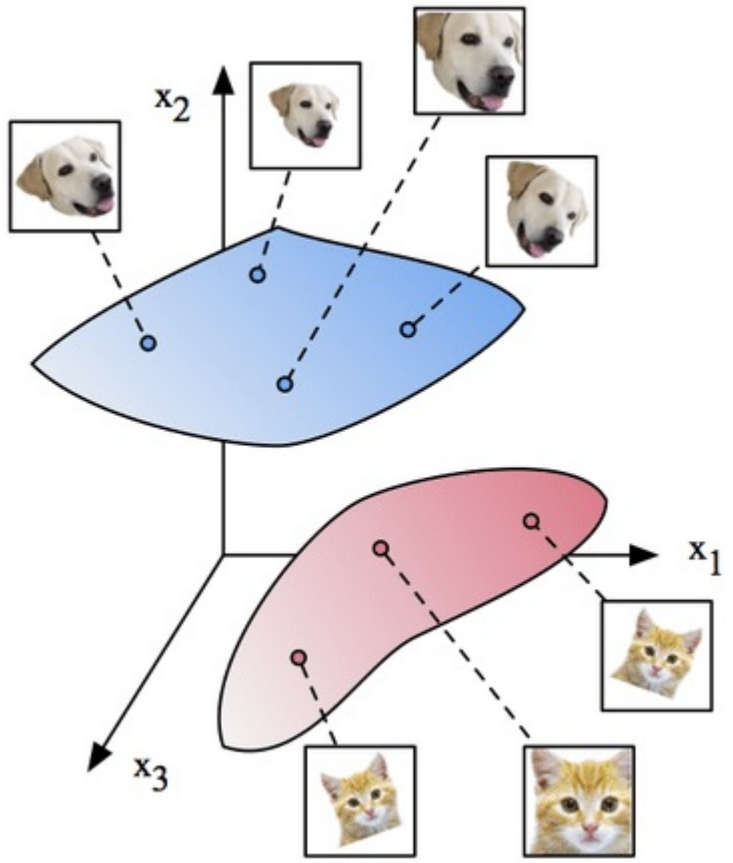
Source: blog.paperspace.com/dimension-reduction-with-t-sne/

Pros and cons of t-SNE

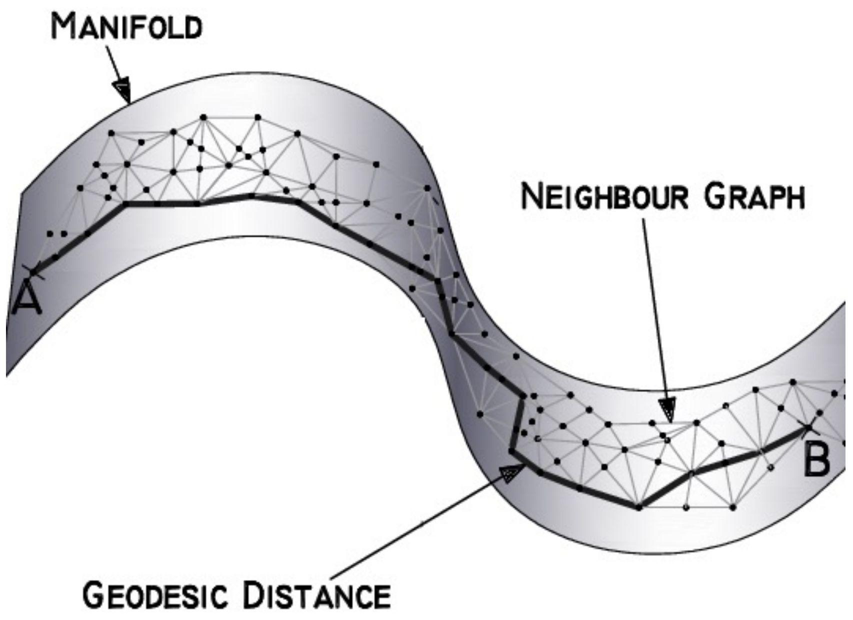
- Able to capture qualitative "neighbor" relationship
  - Turn absolute distance into relative / probabilistic distance
- Address the issue of varying data density
  - Adjust density via perplexity σ_i
- Excel at grouping nearby data points into clusters
  - KL divergence prioritize points with high $p_{j|i}$ (nearby points)

- Long-range relationship tends to be lost
- Quite slow, especially on large datasets
- Cannot transform new data points onto a previous map

The manifold hypothesis



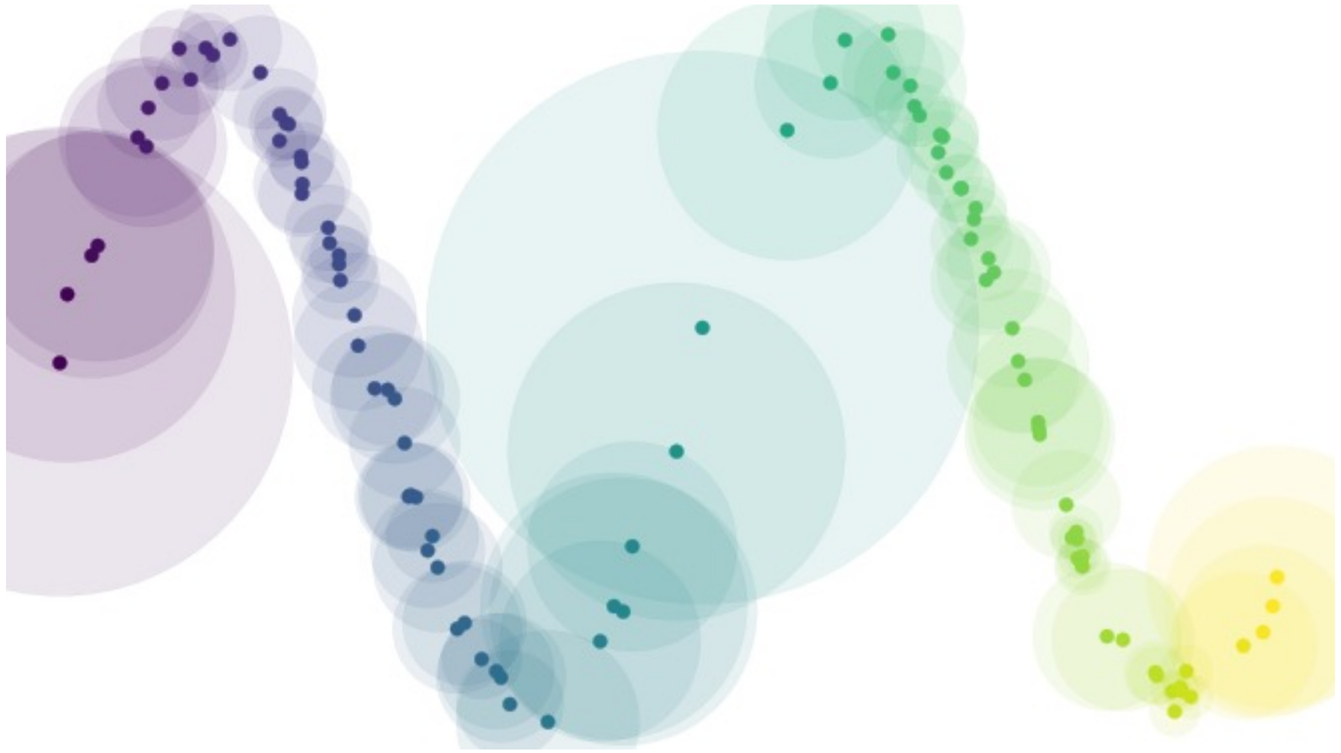
Chung, S. et al. “Classification and Geometry of General Perceptual Manifolds”



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

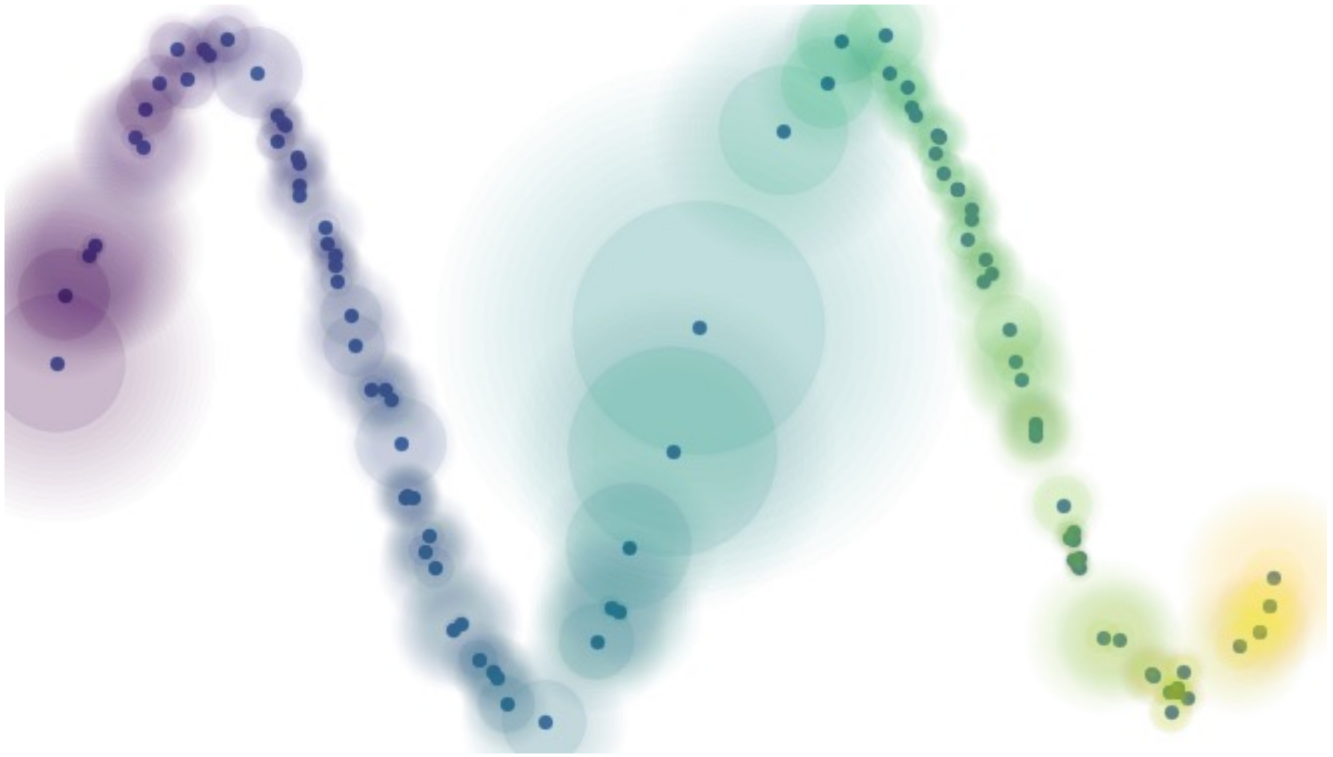
- “Real-world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space”

Assume that data were uniformly drawn



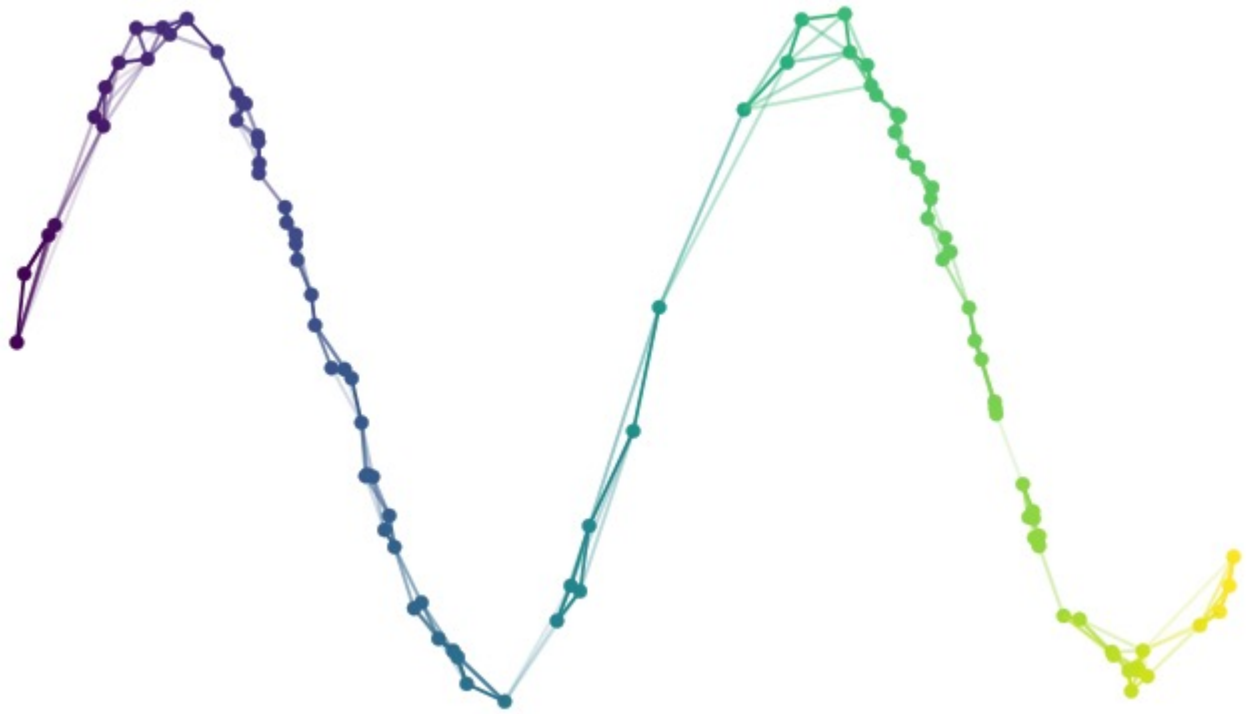
- Distance to the *k*-nearest neighbors should be the same for every data points
  - Dense area = contracted manifold = scale down the distance
  - Sparse area = stretched manifold = scale up the distance
- Has similar effect as σ (perplexity) in *t*-SNE

Adding fuzziness



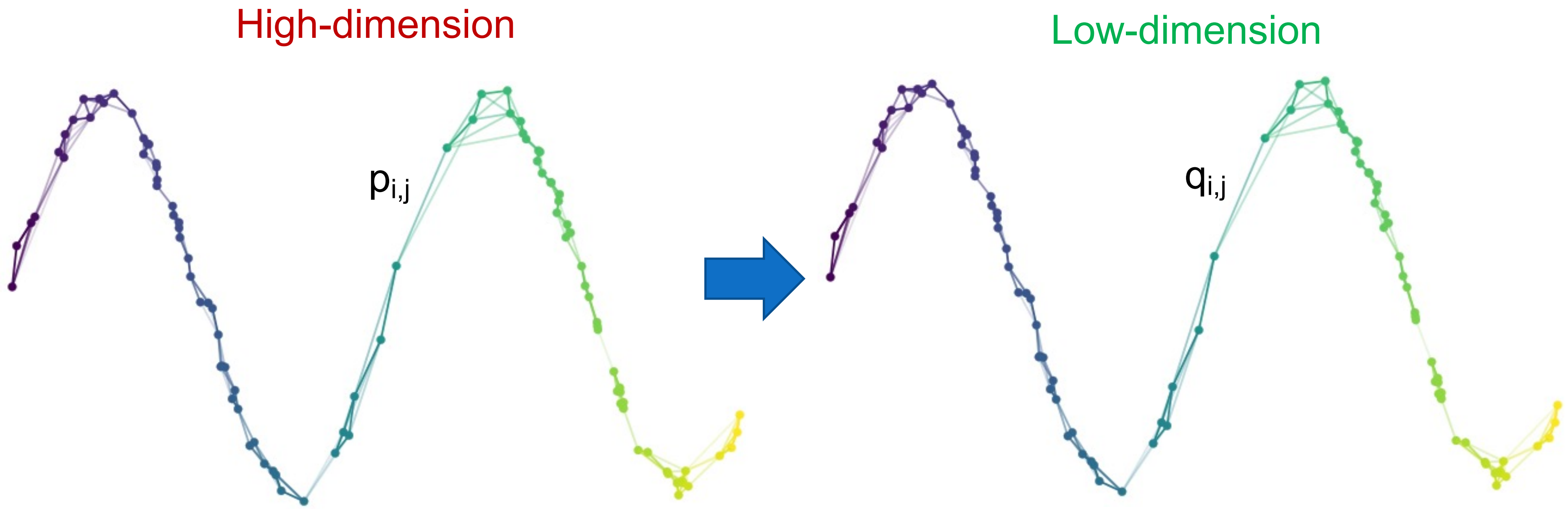
- High confidence up to the first neighbor (locally connected)
  - Probability that x_i is connected to its nearest neighbor is 1.0
- Long-range relationships are more fuzzy
  - Relationship between x_i and x_j may be supported by multiple disks
  - Probability that x_i is connected to x_j is aggregated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

The final probabilistic network



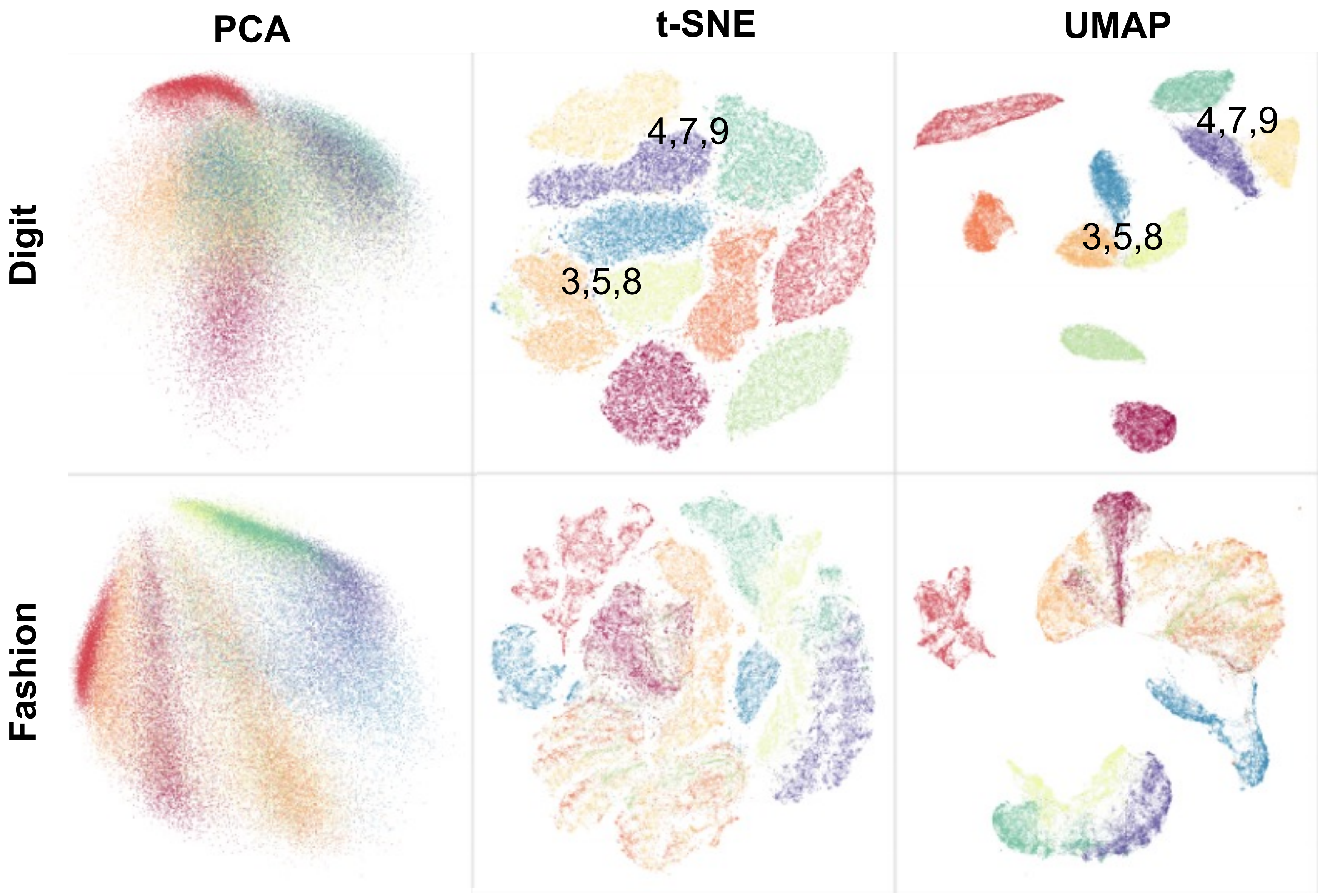
- Nodes are data points
- Edges connected data points with intersecting disks
- Edge weights are probability scores

Embedding into low dimension



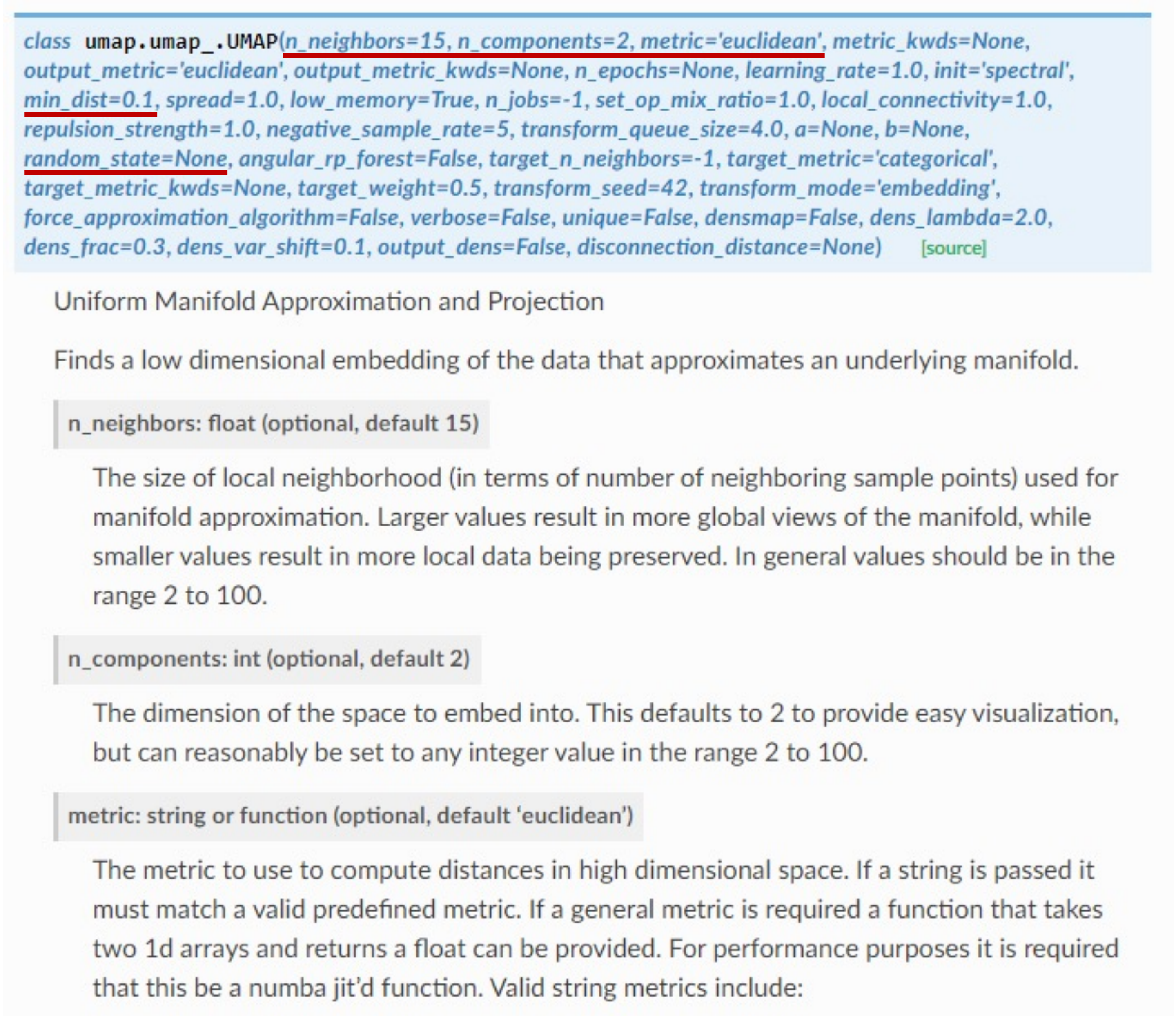
- We want to make $\{q_{i,j}\} \approx \{p_{i,j}\}$
- Instead of KL divergence, we use cross-entropy here
  - $\text{Crossentropy}\ (P \,||\, Q) = \sum_i \sum_j p_{i,j} \log_2 \frac{p_{i,j}}{q_{i,j}} + \sum_i \sum_j (1 - p_{i,j}) \log_2 \frac{(1-p_{i,j})}{(1-q_{i,j})}$
  - Cares about both $p_{i,j}$ and $1 - p_{i,j}$

UMAP also preserve long-range information



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

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.

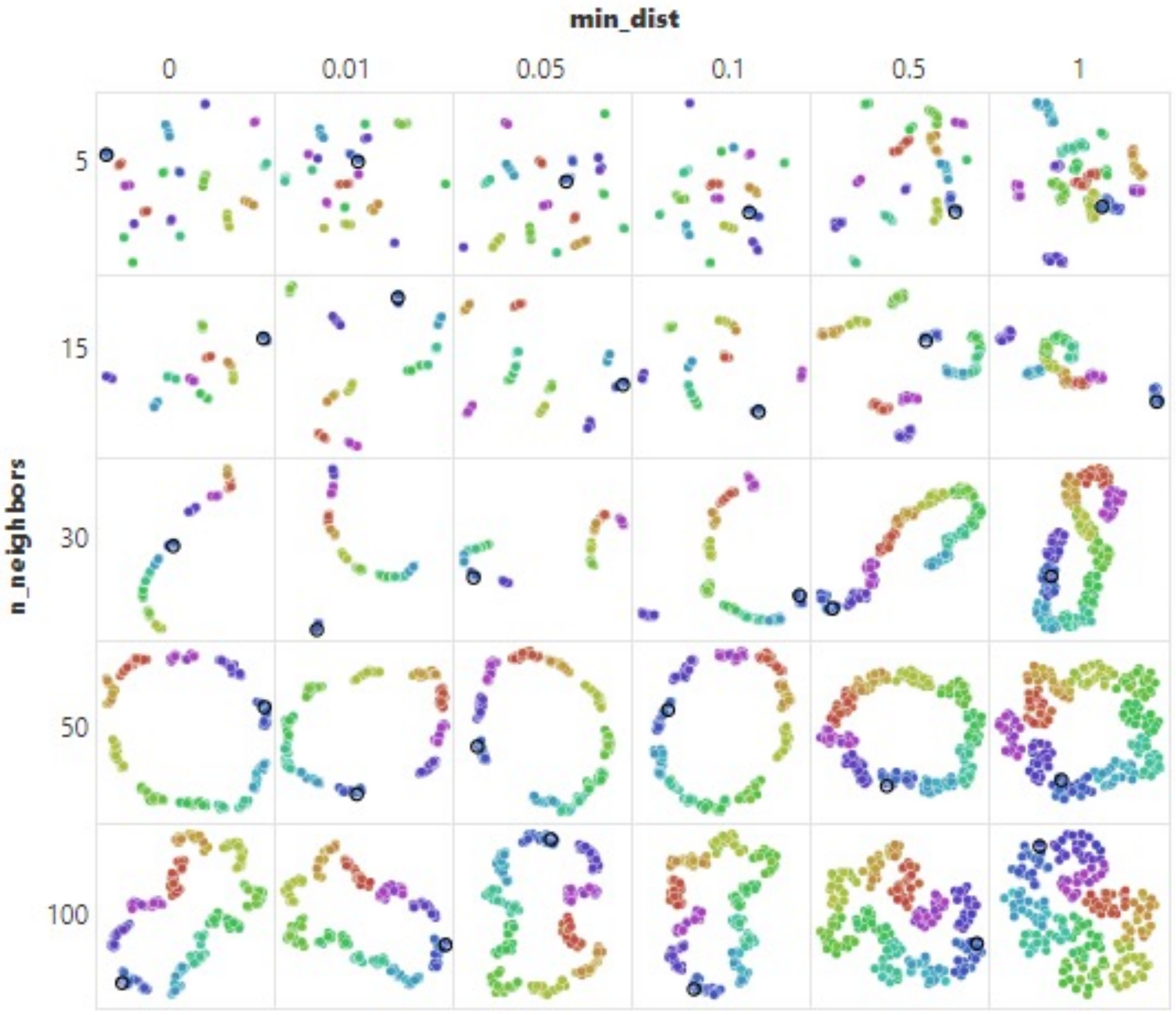
n_components: int (optional, default 2)

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:

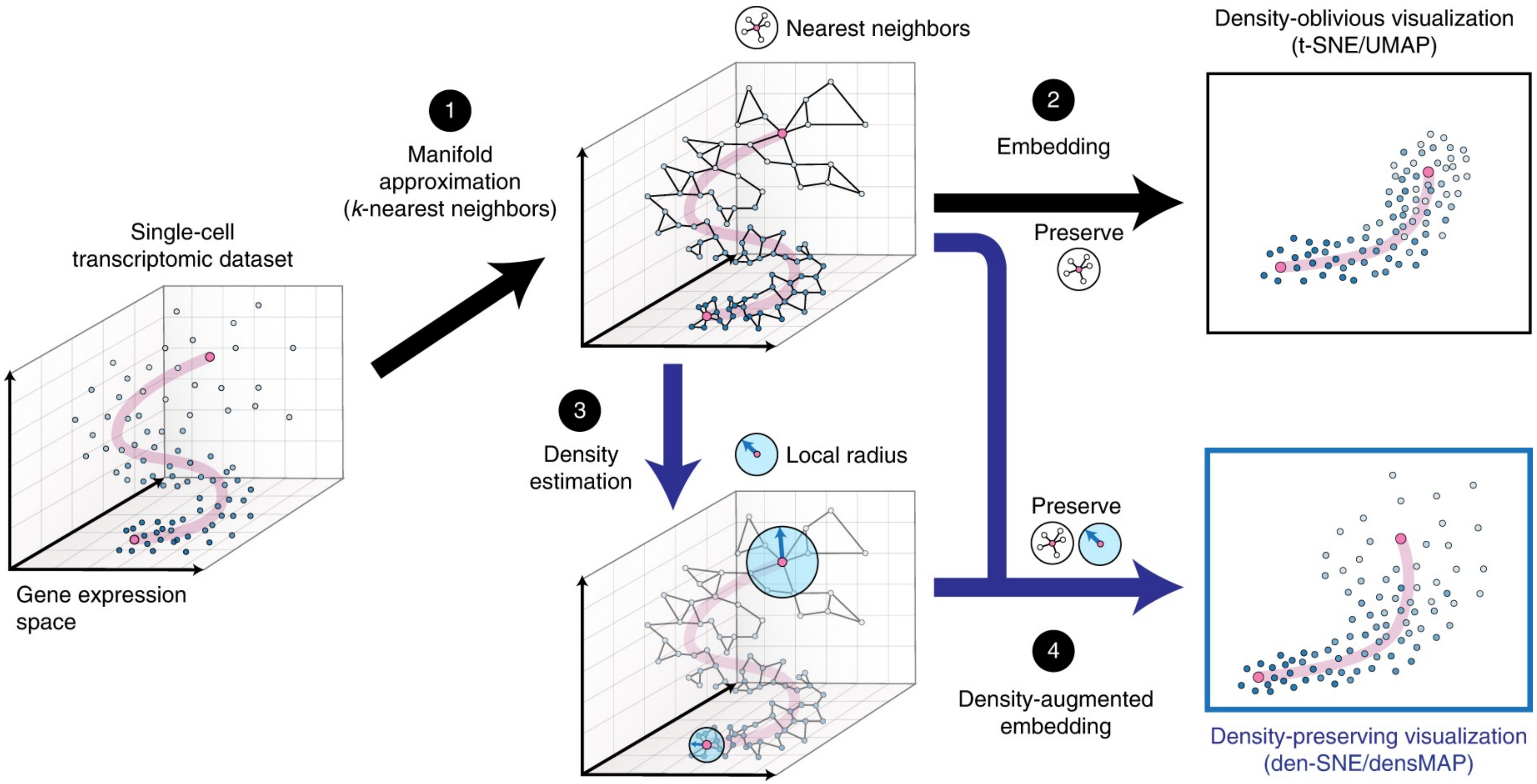
Effect of *n_neighbors* and *min_dist*



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

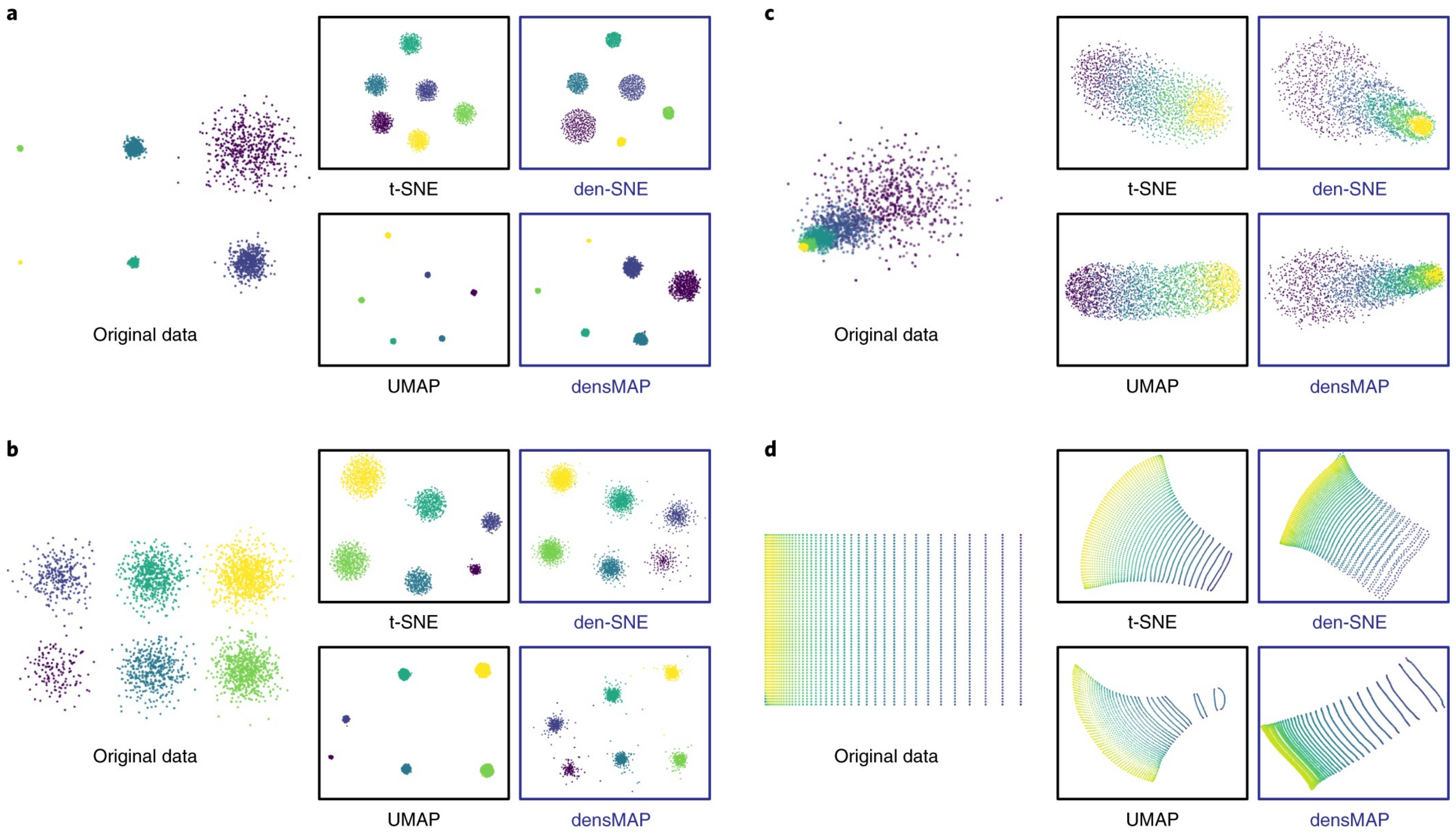
Density-preserving technique

Variance can also be informative

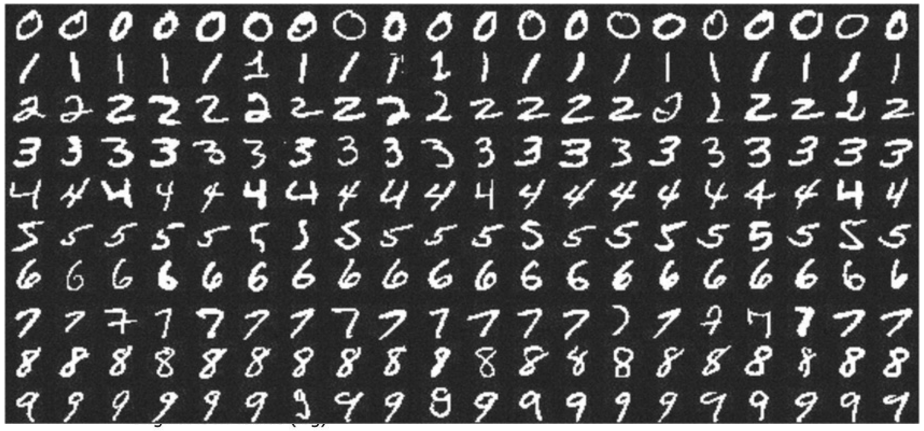


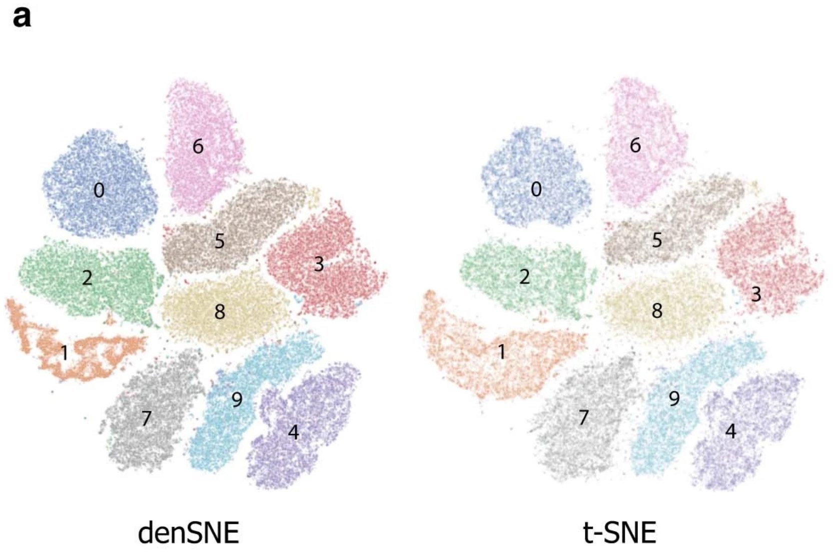
Source: Narayan, A. et al. Nature Biotechnology 39:765-774 (2021)

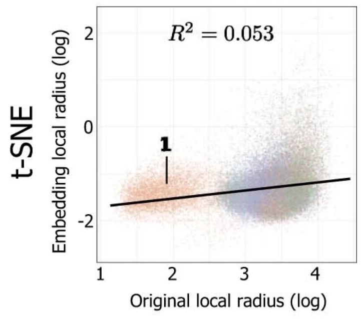
Theoretical examples

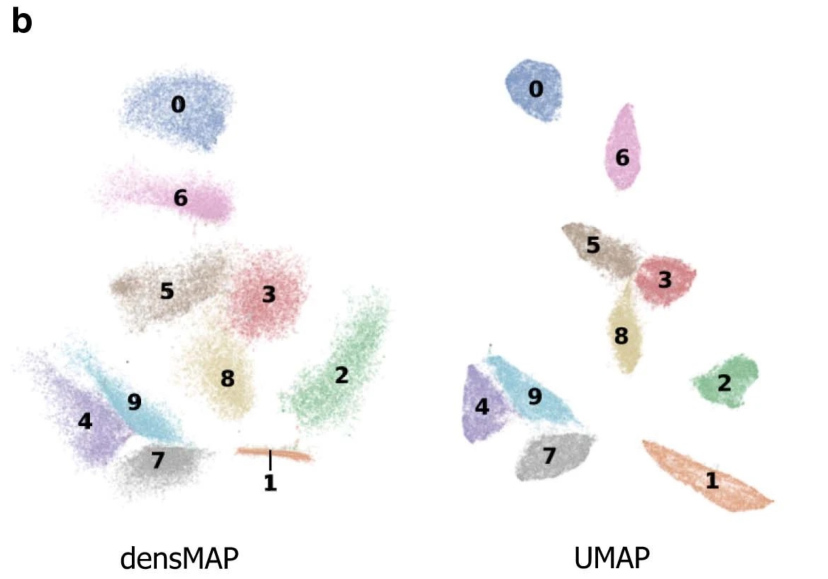


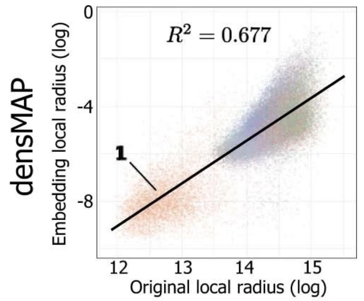
Digit MNIST

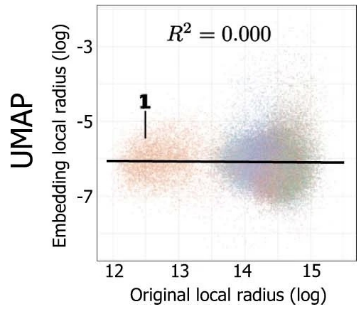



a
0
6
5
3
2
8
1
7
9
4
denSNE
t-SNE


t-SNE
$R^2 = 0.053$
Embedding local radius (log)
Original local radius (log)
1


b
0
6
5
3
8
2
4
9
7
1
densMAP
UMAP


densMAP
$R^2 = 0.677$
Embedding local radius (log)
Original local radius (log)
1


UMAP
$R^2 = 0.000$
Embedding local radius (log)
Original local radius (log)
1

Any question?