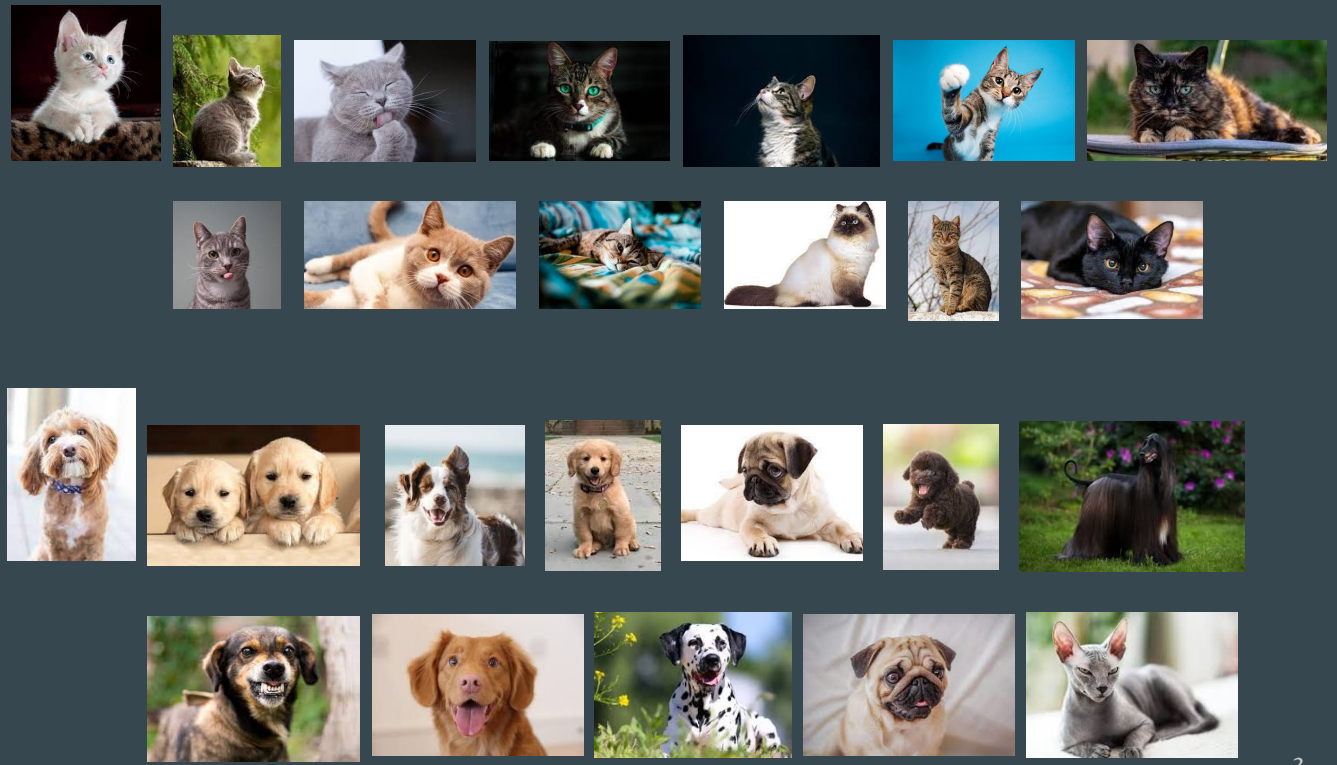
# Machine Learning

Unsupervised learning

# Machine Learning problems

# Machine learning

ML



# Supervised learning

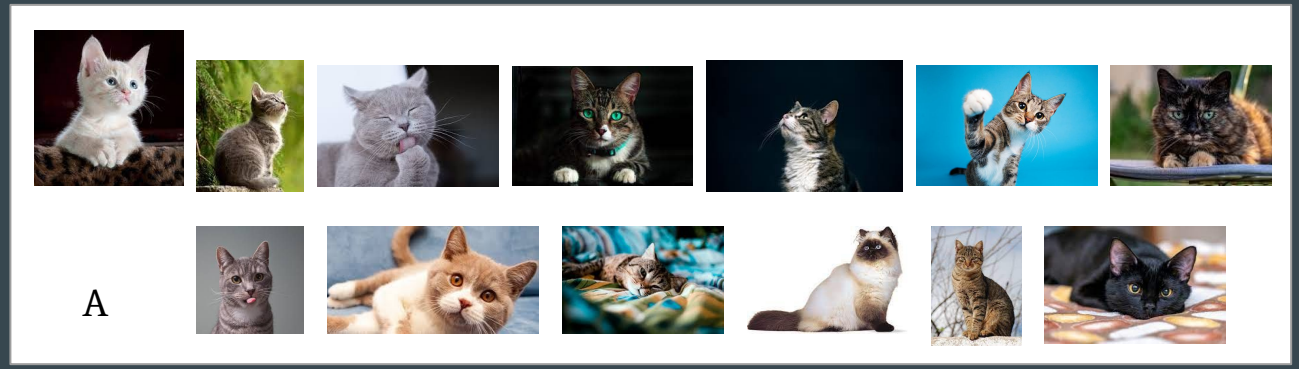
ML

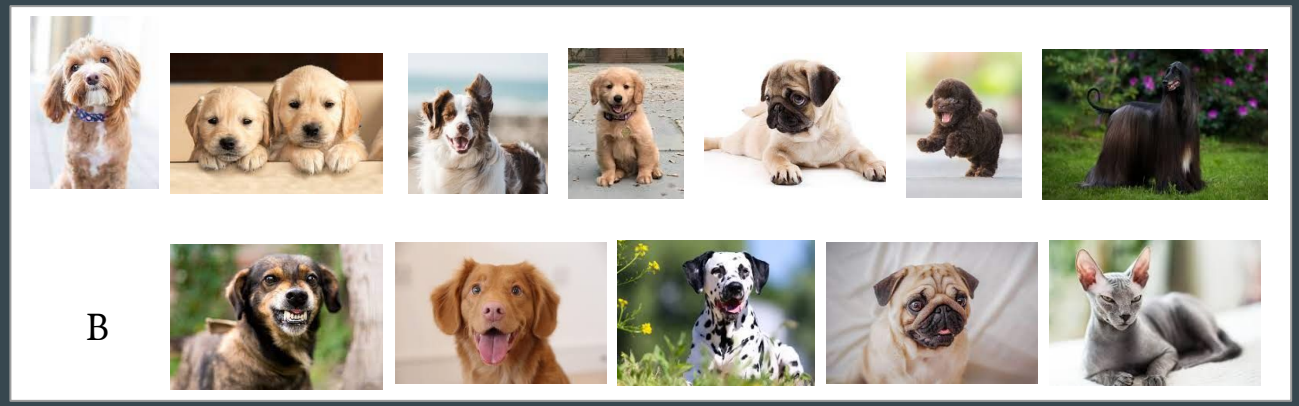
Cats

Dogs

# Unsupervised learning

ML



# Unsupervised learning

Unsupervised clustering is used to group similar data points into clusters without any predefined labels or categories. Unsupervised clustering is suitable for solving problems where the goal is to discover patterns and relationships in the data.

- There is no clear relationship between the input data and the output labels.
- There is a large amount of unlabeled data available.
- The goal is to discover patterns and relationships in the data.
- The data can be grouped into natural clusters based on similarity.

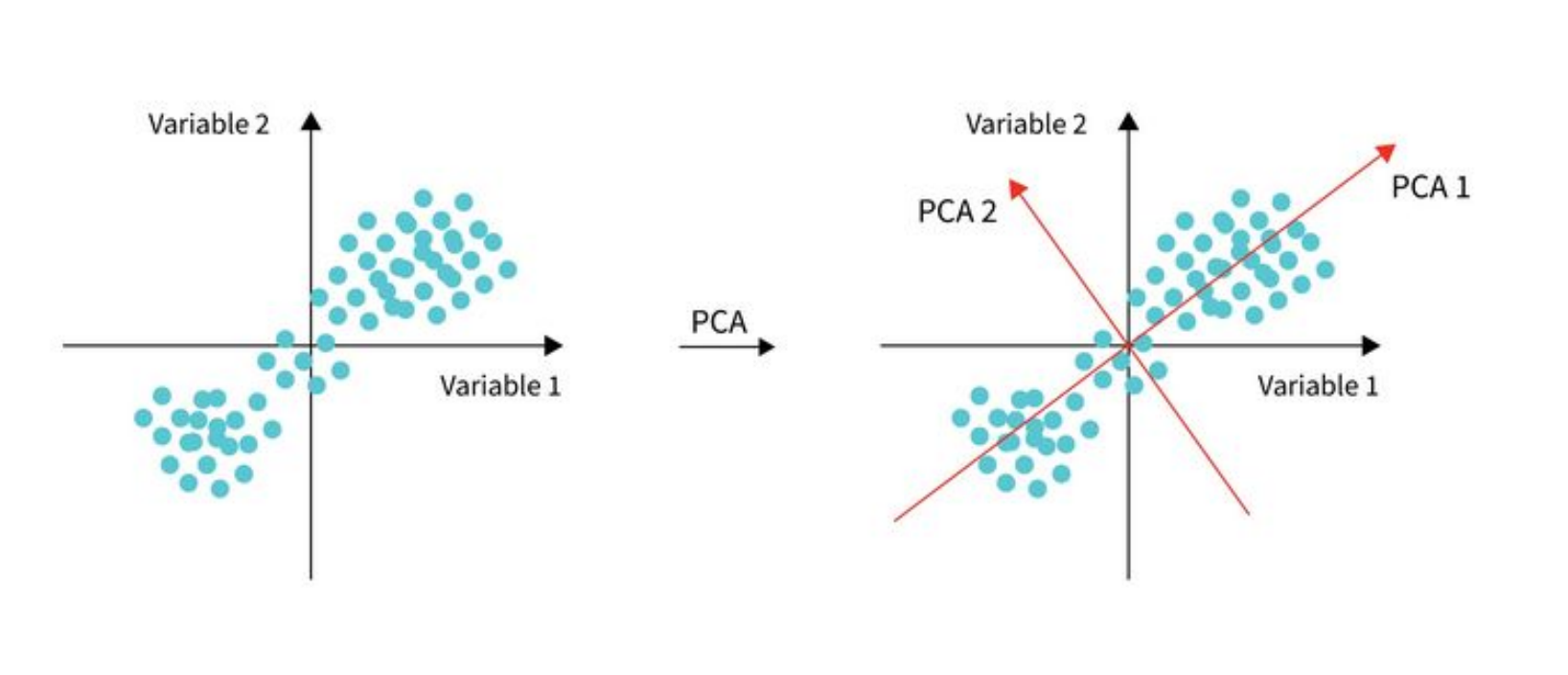
Some examples of applications that use unsupervised clustering include image segmentation, clustering, dimensionality reduction.

# Dimensionality reduction

# Principal Component Analysis (PCA)

PCA is a linear dimensionality reduction method used to identify the underlying patterns in a dataset by finding the principal components that explain the most variation in the data. Each principal component is a linear combination of the original variables, and the first principal component explains the most variance in the data, followed by the second principal component, and so on. The key idea of PCA is to preserve the maximum amount of variance in the data by generating linear projections of the data into a lower-dimensional space. It is often used in data preprocessing and visualization.

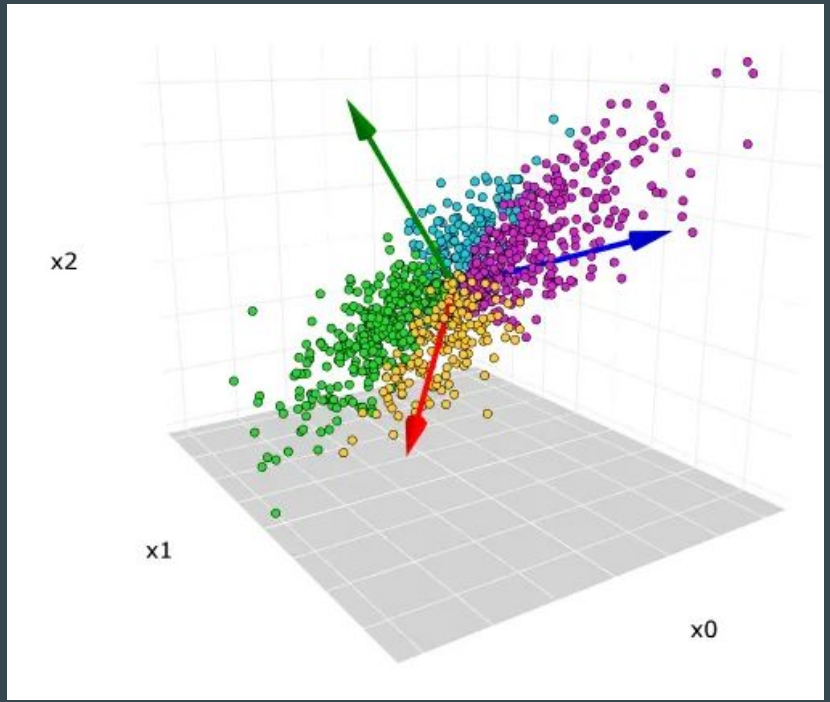
# Principal Component Analysis (PCA)




https://www.scaler.com/topics/nlp/what-is-pca/

# Principal Component Analysis (PCA)

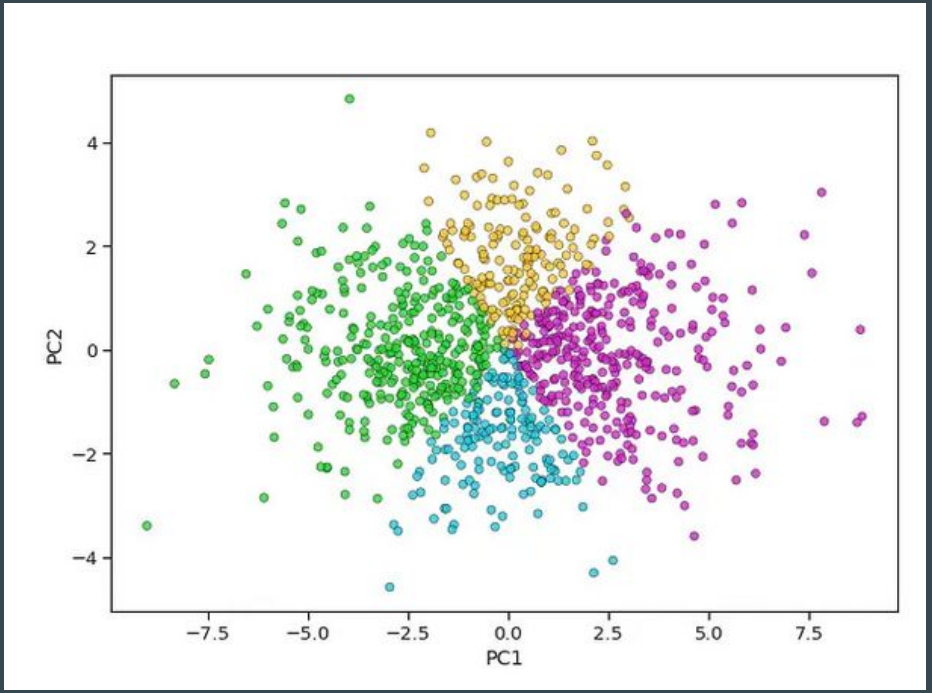
Principal component analysis (PCA) is a technique that transforms high-dimensions data into lower-dimensions while retaining as much information as possible.




Casey Cheng 2022

# Principal Component Analysis (PCA)

Principal component analysis (PCA) is a technique that transforms high-dimensions data into lower-dimensions while retaining as much information as possible.



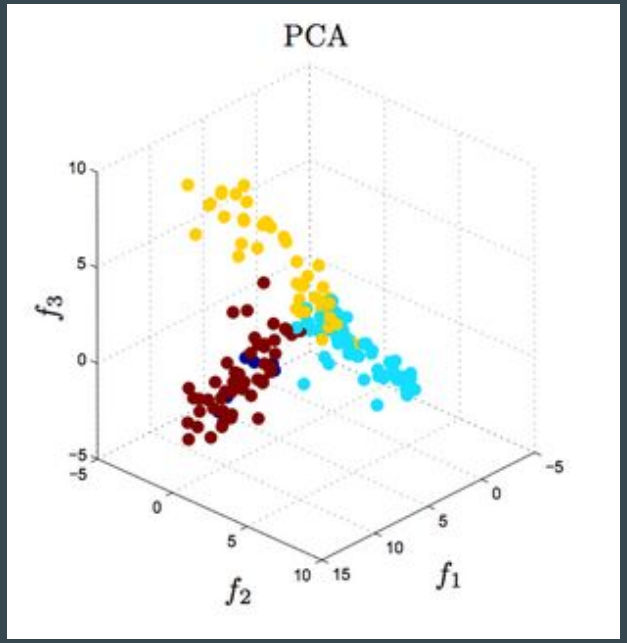

Casey Cheng 2022

# Principal Component Analysis (PCA)

The principal components are calculated in such a way that they are orthogonal to each other, which means they are perpendicular and independent of each other. This property ensures that the principal components do not duplicate each other and capture unique information in the data. PCA is commonly used in data science to reduce the dimensionality of a dataset by selecting the most important principal components and discarding the rest. This approach helps in visualizing the data, identifying outliers, and detecting underlying patterns that are not apparent in the original data.

# Principal Component Analysis (PCA)

500 genes were measured for a large number of samples. The factors f1, f2, f3 obtained by traditional PCA each use all 500 genes.

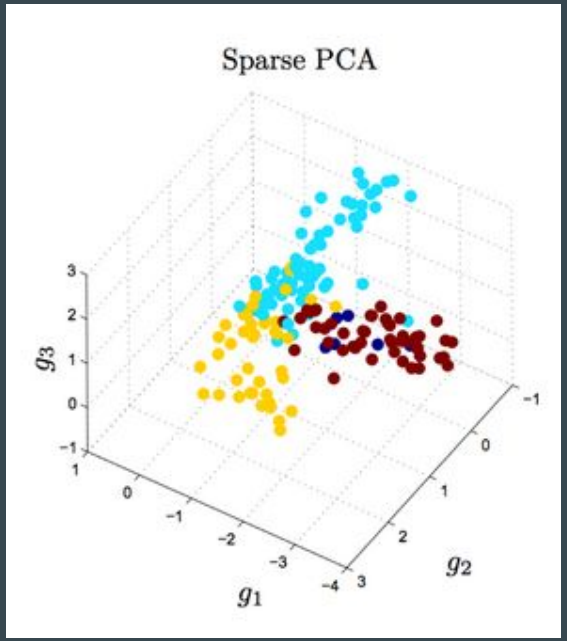



Alex Williams 2016

# Principal Component Analysis (PCA)

500 genes were measured for a large number of samples. The factors f1, f2, f3 obtained by traditional PCA each use all 500 genes.

The sparse factors g1, g2, g3 and on the right together involve only 14 genes, which can be useful for developing parsimonious hypotheses.



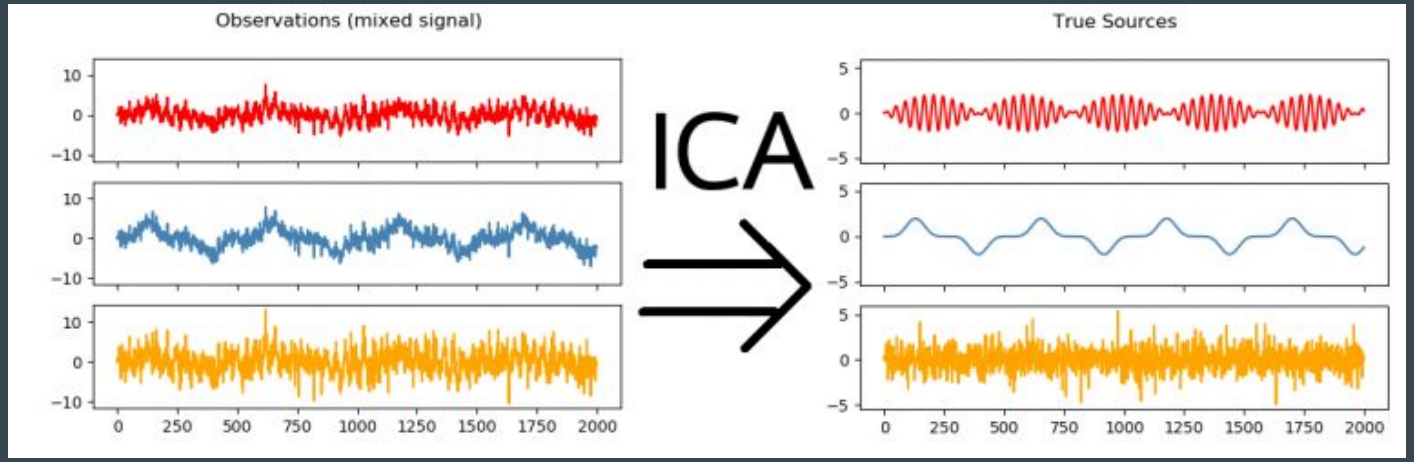

Alex Williams 2016

# Independent Component Analysis (ICA)

Independent Component Analysis (ICA) is a method that separates a multivariate signal into independent, non-Gaussian components. ICA assumes that the observed data is a linear combination of the independent components with unknown mixing coefficients. Unlike PCA, which identifies the principal components that explain the most variance in the data, ICA focuses on identifying the sources that are statistically independent of each other.

# Independent Component Analysis (ICA)

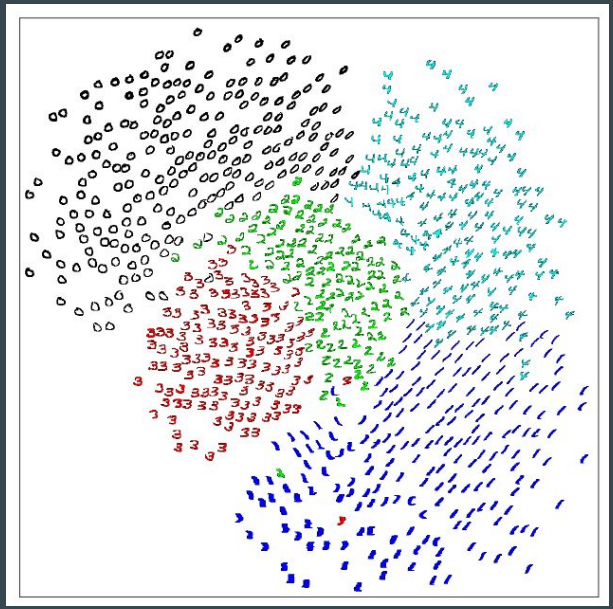
Independent component analysis (ICA) is a widely used data exploration technique in neuroscience, it is part of most EEG/MEG processing pipelines. It aims at decomposing signals into a mixture of independent sources.




https://team.inria.fr

# t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a nonlinear dimensionality reduction method that aims to preserve the pairwise similarity between data points in a lower-dimensional space. It is often used in data visualization and clustering. It is based on Stochastic Neighbor Embedding



Stochastic Neighbor Embedding

G

eoffrey Hinton and Sam Roweis 2002

# Stochastic Neighbor Embedding (SNE)

Given a set of N high-dimensional objects $x_1, \ldots, x_N$ t-SNE first computes probabilities $p_{i|j}$ that are proportional to the similarity of objects $x_i$ and $x_j$ as follows:

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

The similarity of datapoint $x_j$ to datapoint $x_i$ is the conditional probability p(j|i) that $x_i$ would pick $x_j$ as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at $x_i$

# Stochastic Neighbor Embedding (SNE)

Since from the probability we need to translate to distance measure, we need to make this probability symmetric. For a sample size N, we can assign 1/N to each sample, and then $p_{ij}$ is

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

From the definition of $p_{i|j}$ it follows that $p_{i|j} = p_{j|i}$ and therefore $p_{i|j} = Np_{ij}$

The bandwidth of the gaussian distribution is adapted to the density of the data.

# t-Stochastic Neighbor Embedding (tSNE)

The Gaussian kernel uses the Euclidean distance $\|x_i - x_j\|$ and as such for higher dimensional data $p_{ij}$ asymptotically converges towards a constant value. Therefore it is not possible to discriminate points of higher dimensions (curse of dimensionality).

For this reason, t-SNE adjusts the distances with a power transform that depends on the dimensionality of the data.

# t-SNE

An additional term is introduced

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

So that the points in the embedded space will be computed by minimizing the KL divergence:

$$\mathrm{KL}\left(P \parallel Q\right) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

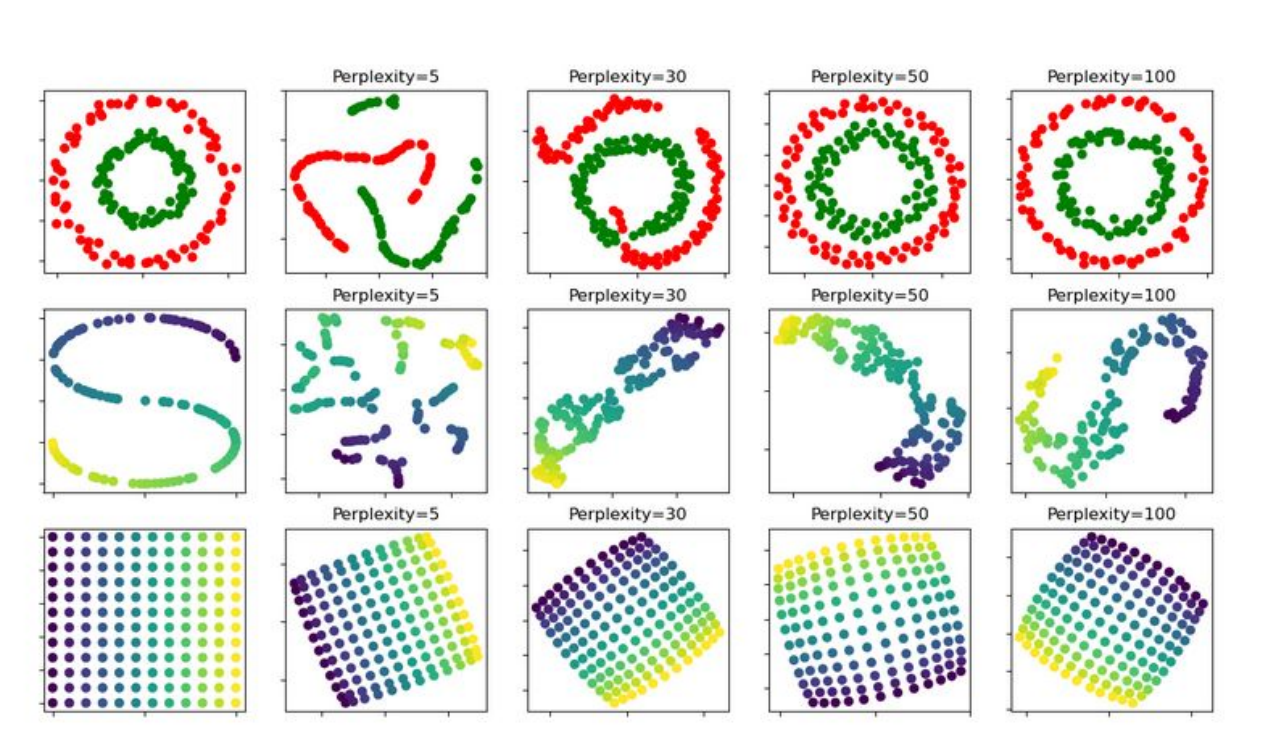
# t-SNE

Main parameter in t-SNE is perplexity. This parameter is used to balance the focus between local and global structure. It determines the number of nearest neighbors used to compute the conditional probability distribution of the data points in the high-dimensional space. Increasing the perplexity will result in a more global view of the data, while decreasing it will focus on the local structure. A typical value is between 5 and 100.

# t-SNE

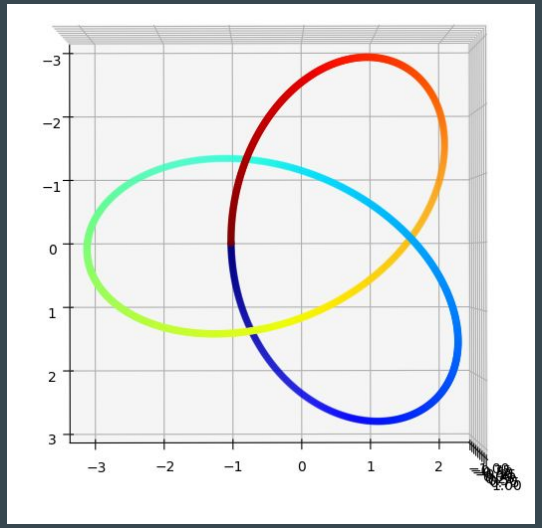
Example of parameter tuning for different example datasets.

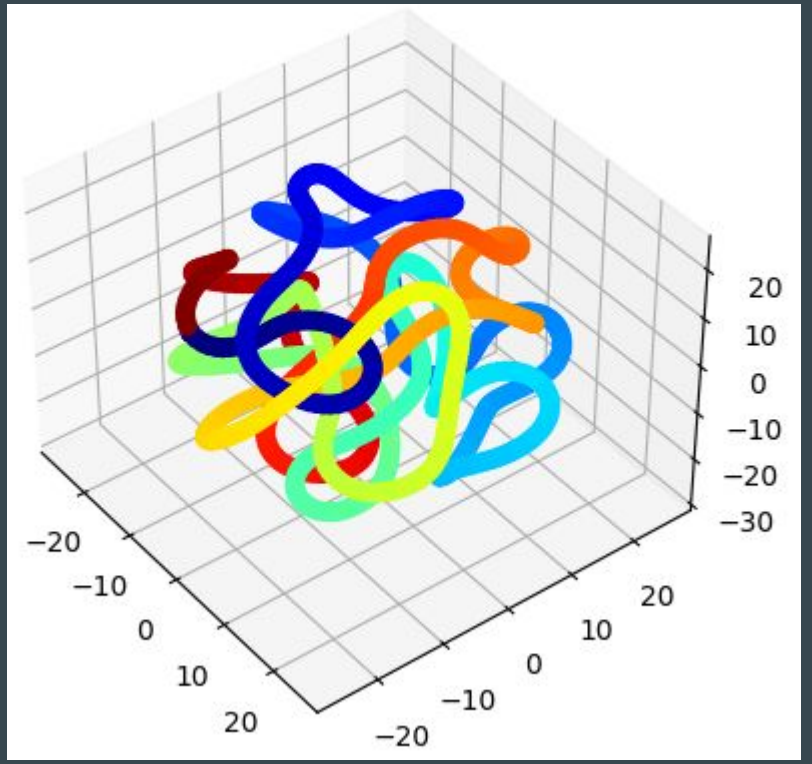
sklearn



# t-SNE

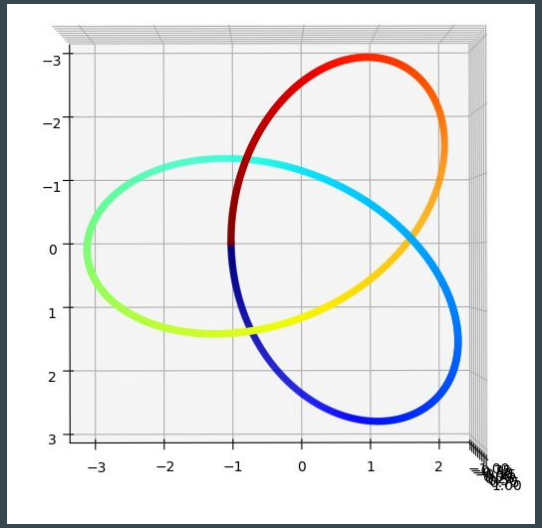
Perplexity = 3

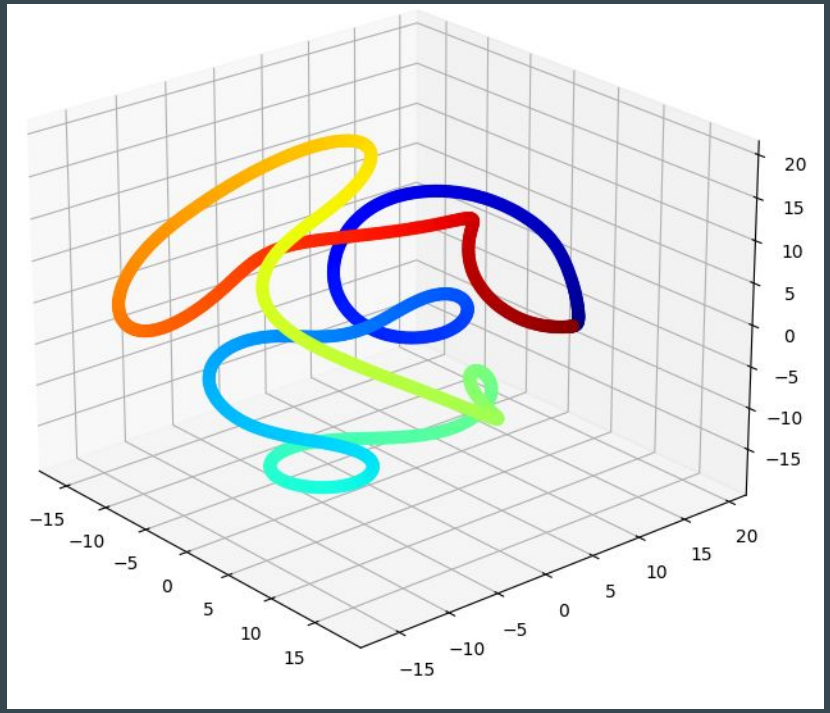


# t-SNE

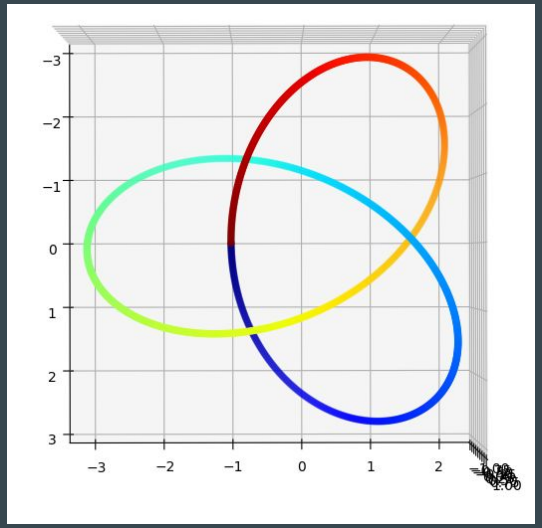
Perplexity = 13

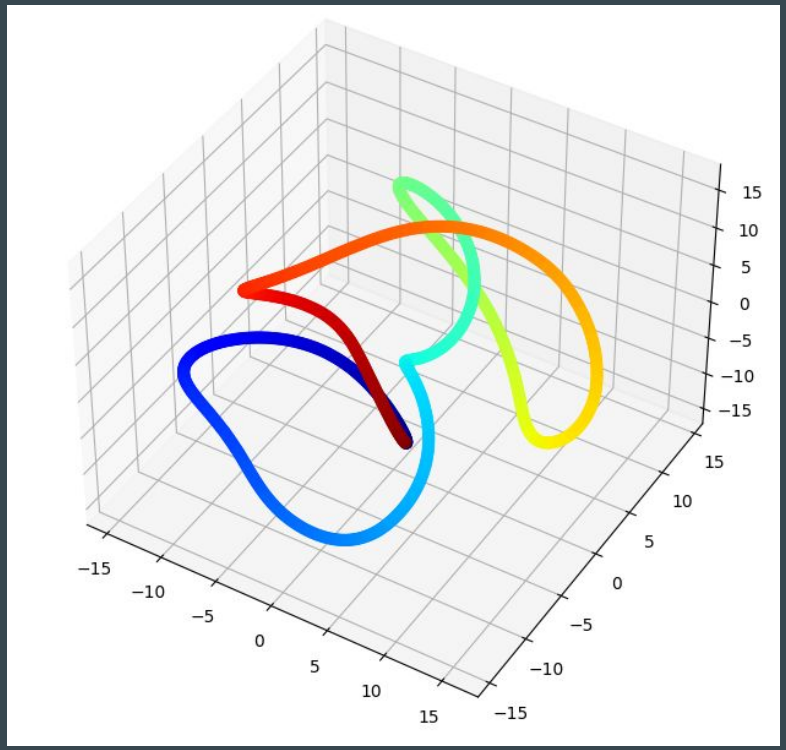


# t-SNE

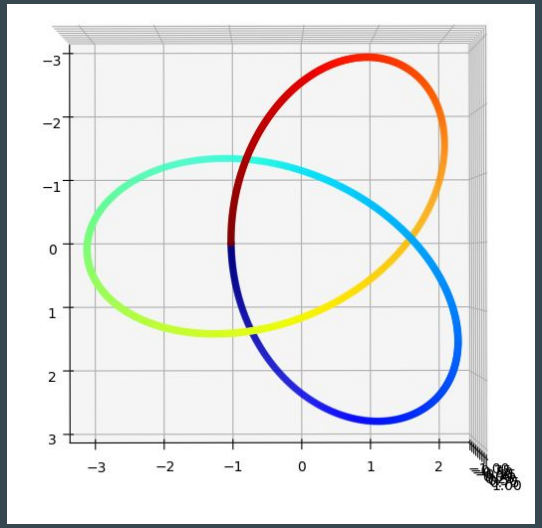
Perplexity = 20

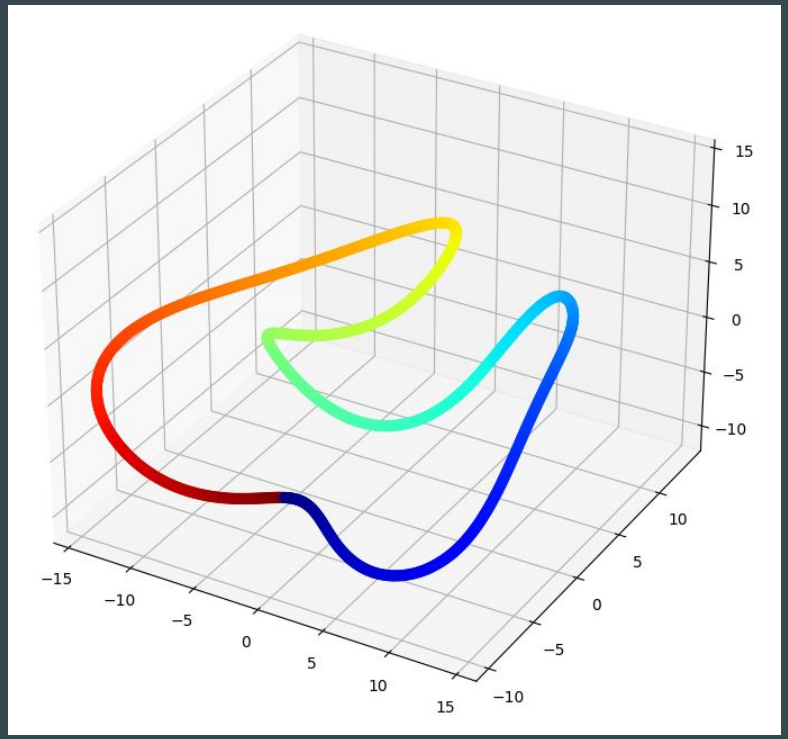


# t-SNE

Perplexity = 30



# UMAP (Uniform Manifold Approximation and Projection)

UMAP is an algorithm for dimension reduction based on manifold learning techniques and ideas from topological data analysis. It provides a very general framework for approaching manifold learning and dimension reduction, but can also provide specific concrete realizations. UMAP is used for visualizing and analyzing high-dimensional data.

UMAP, at its core, works very similarly to t-SNE. Both use graph layout algorithms to arrange data in low-dimensional space.

# UMAP (Uniform Manifold Approximation and Projection)

UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible. More specifically, it constructs a high-dimensional graph of the data points using a nearest neighbor approach, where each point is connected to its k nearest neighbors. It then applies a series of optimization steps to find a low-dimensional representation of the data that preserves the topology of the high-dimensional graph.

# UMAP (Uniform Manifold Approximation and Projection)

One of the key features of UMAP is that it is designed to be highly scalable and can handle very large datasets. It uses a stochastic gradient descent approach to optimize the embedding, which makes it computationally efficient.

UMAP is also highly flexible and can be used for a variety of tasks, including data exploration, visualization, and clustering. It has been shown to be particularly effective at preserving global structure in the data, making it a useful tool for tasks such as visualizing complex datasets or identifying clusters of similar data points.

# UMAP

The main parameters in UMAP are the number of neighbors and the minimum distance.

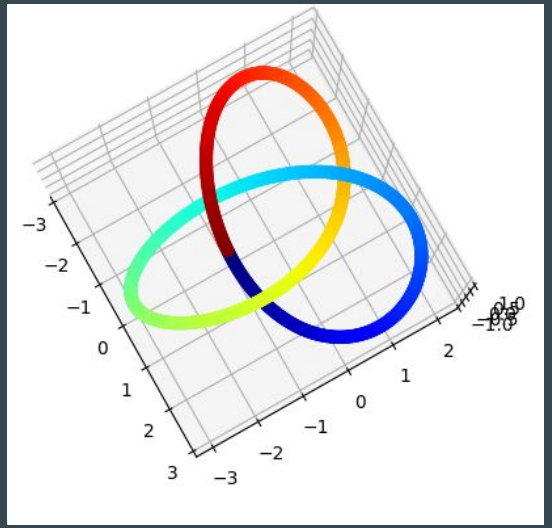
Number of neighbors: controls the parameter k in the k-nearest neighbors that are connected to each point in the higher dimensional representation.

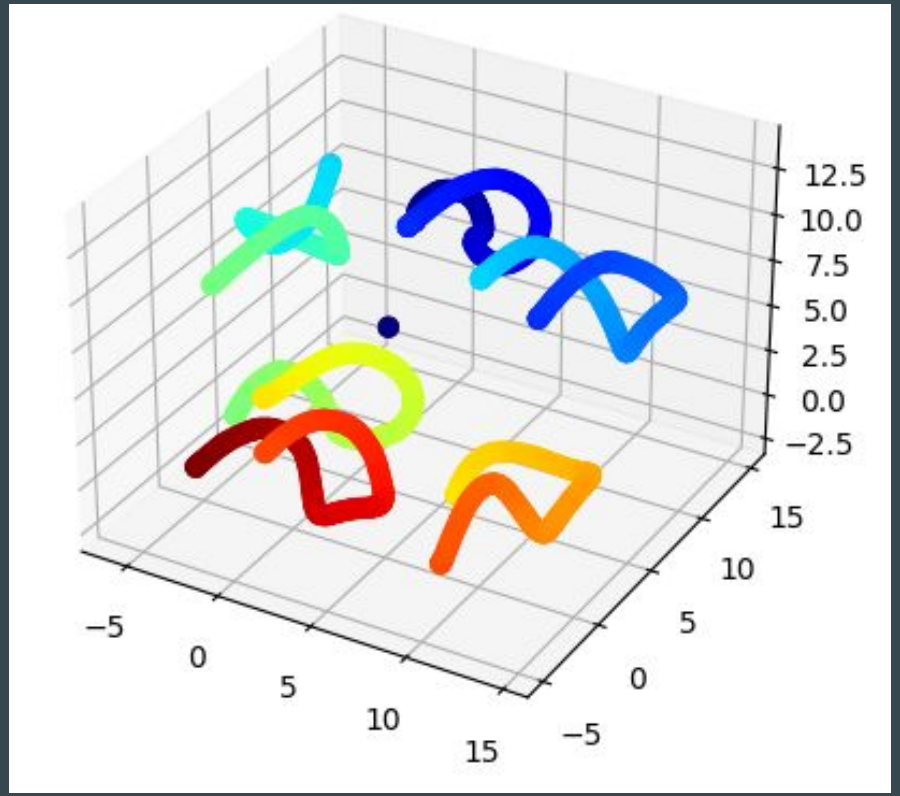
Minimum distance: controls the minimum distance between embedded points in the low-dimensional space. Specifically, it controls the tightness of the clusters in the resulting embedding. For small values, the clusters in the low-dimensional space are more tightly packed (more pronounced clusters). For large values, the clusters are more spread out, resulting in more gradual transitions between clusters.

# UMAP

Min_dist = 0.0

N_neigh = 2

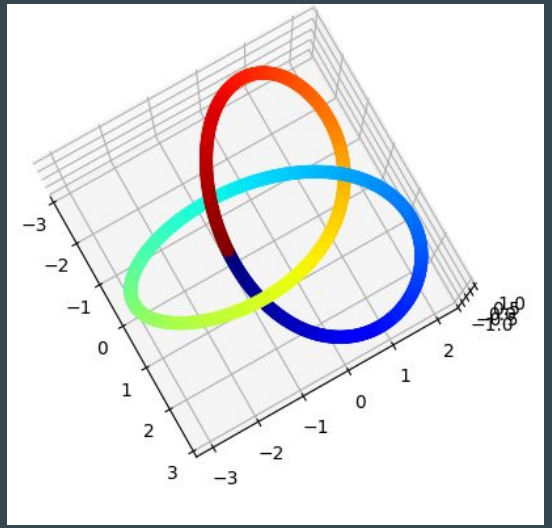


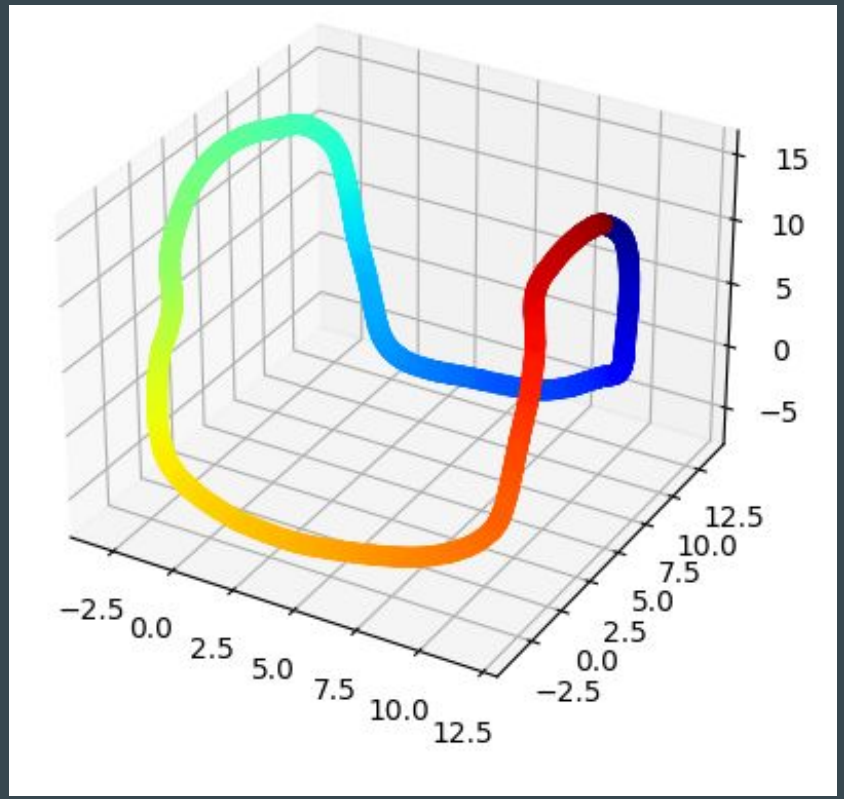



# UMAP

Min_dist = 0.0

N_neigh = 10

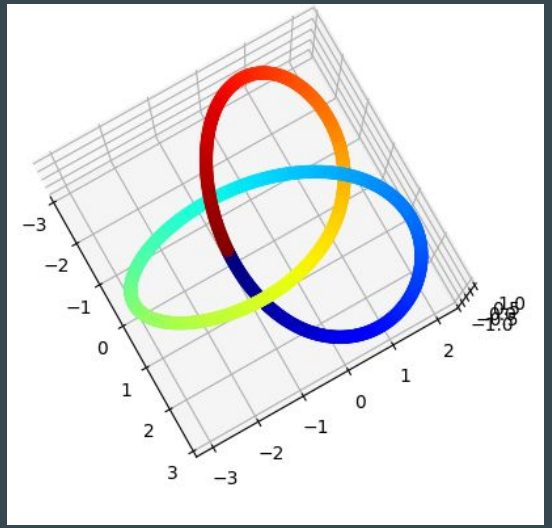


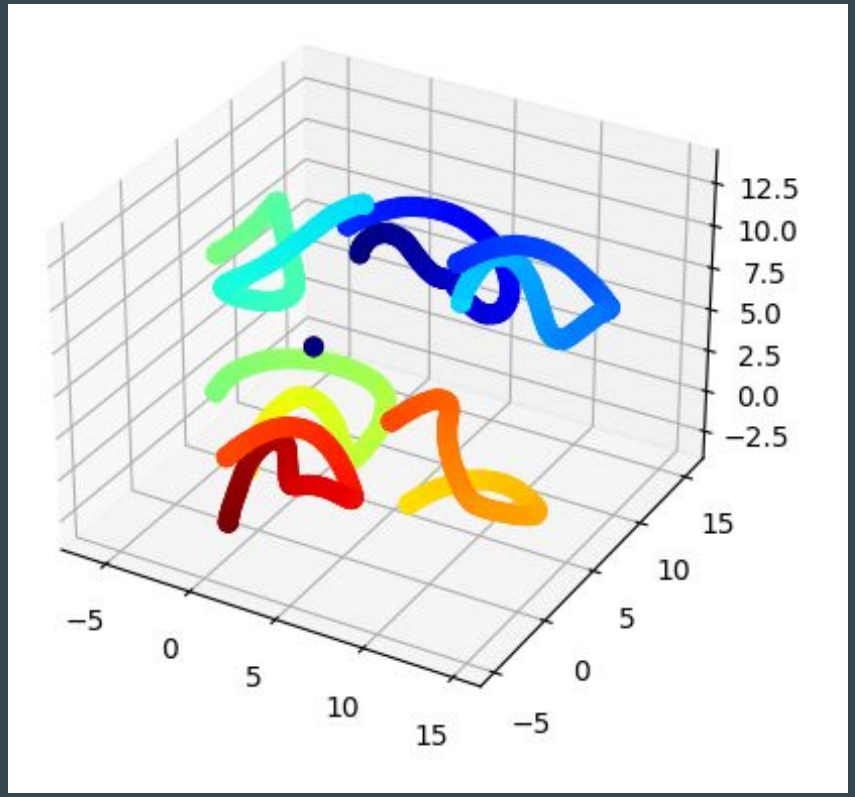



# UMAP

Min_dist = 0.01

N_neigh = 2

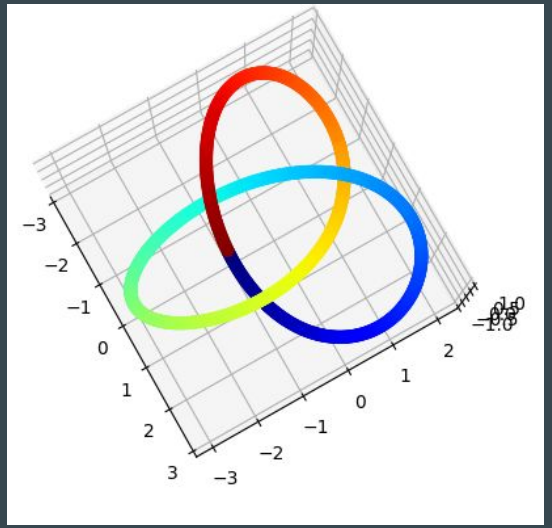


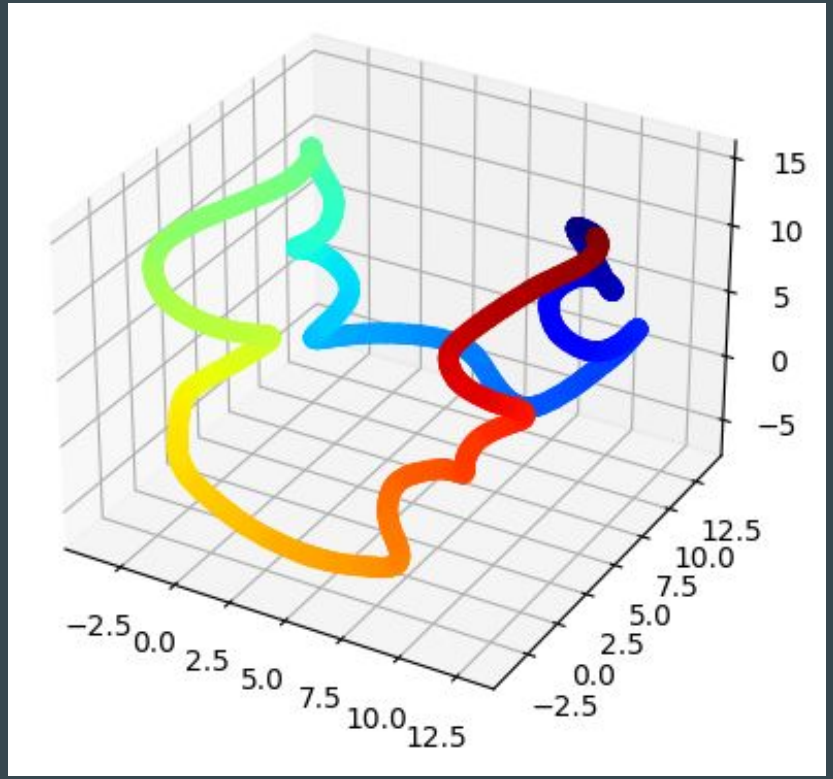



# UMAP

Min_dist = 0.01

N_neigh = 5

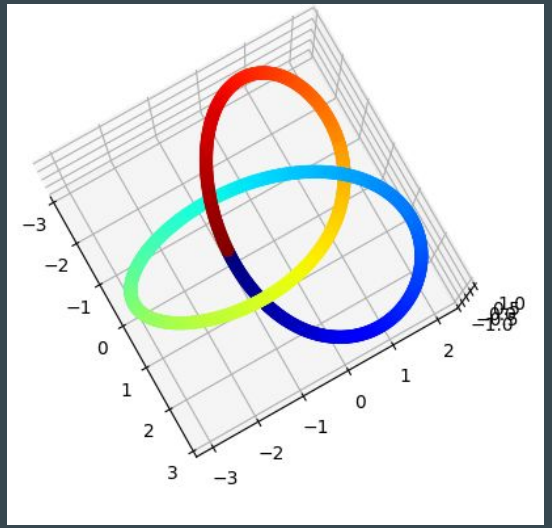


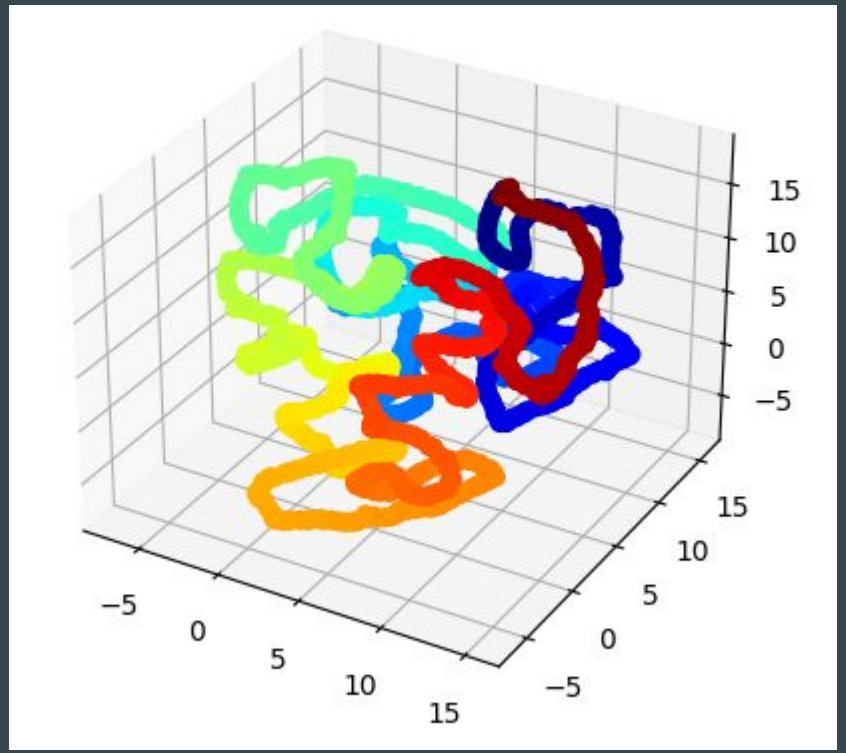



# UMAP

Min_dist = 1.0

N_neigh = 5

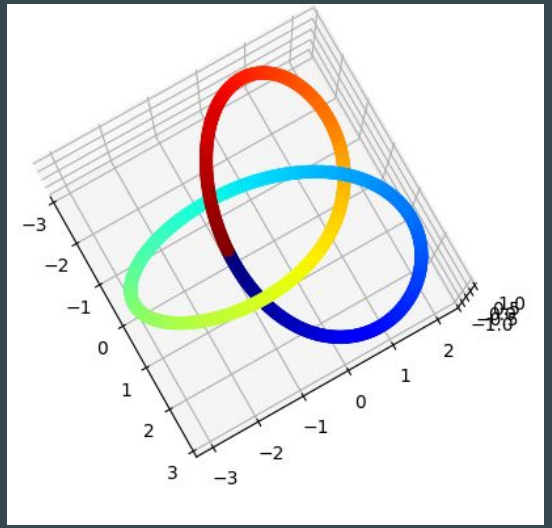


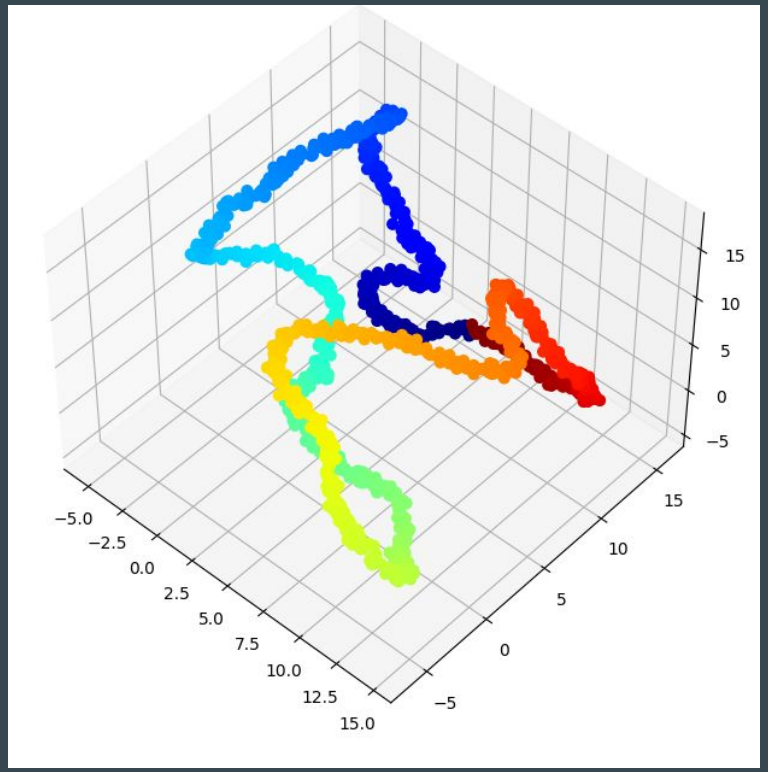



# UMAP

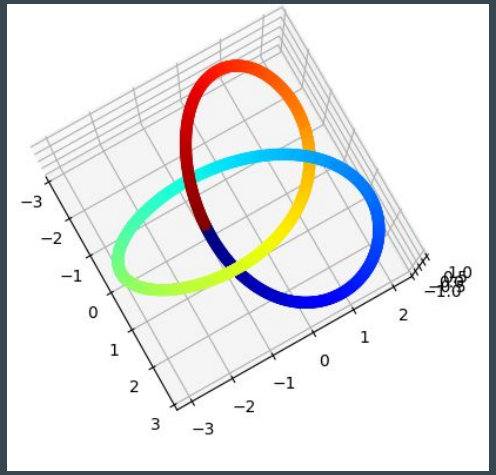
Min_dist = 1.0

N_neigh = 50



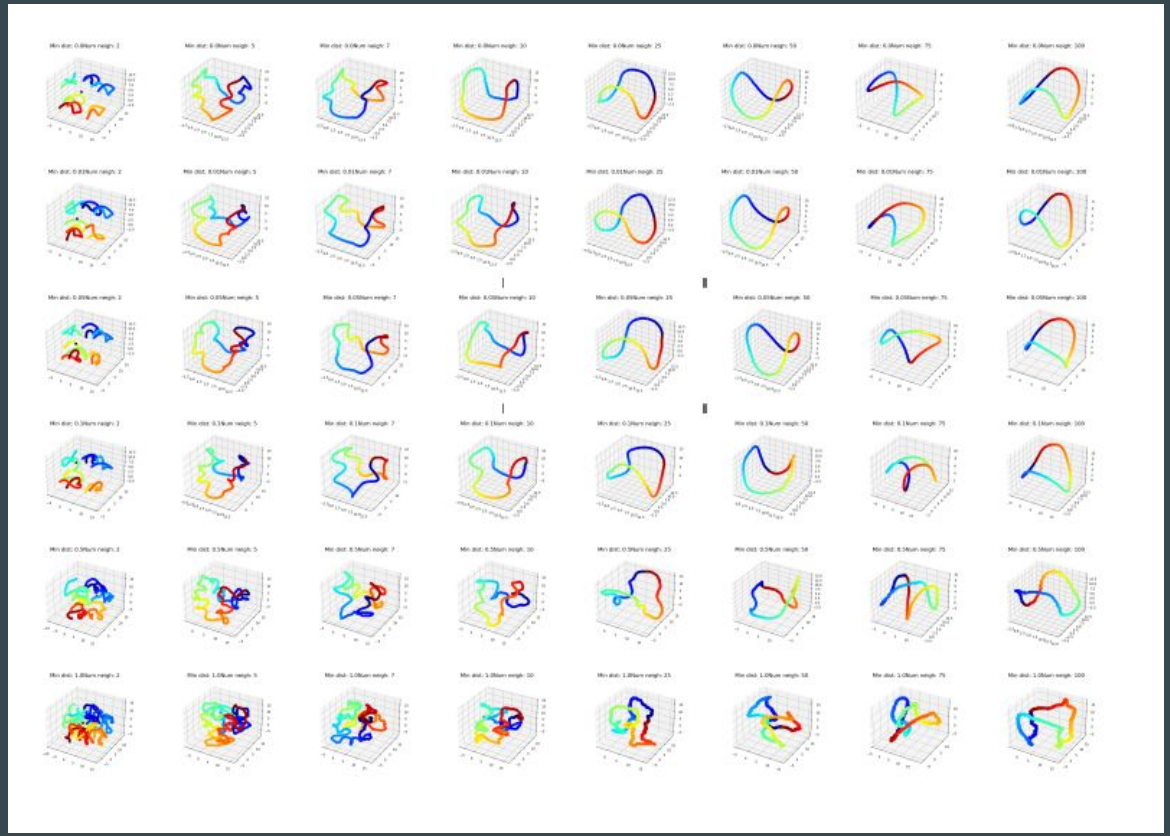



# UMAP



Num: 2 5 7 10 25 50 75 100

Min dist: 0 0.01 0.05 0.1 0.5 0.1



# Unsupervised classification problems

# K-means

K-means clustering is an unsupervised machine learning algorithm used for grouping data into K clusters based on their similarity. The algorithm works by iteratively assigning data points to the nearest cluster centroid and recalculating the centroid of each cluster based on the new assignments. This process continues until the centroids no longer change or a maximum number of iterations is reached.

# K-means

The K-means algorithm aims to select the centroids of the groups, with the objective of minimising the within-cluster sum-of-squares criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

# K-means

1) The first step chooses the initial centroids, with the most basic method being to choose samples from the dataset .
2) After initialization, K-means consists of looping between the two other steps. The first step assigns each sample to its nearest centroid.
3) The second step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid.

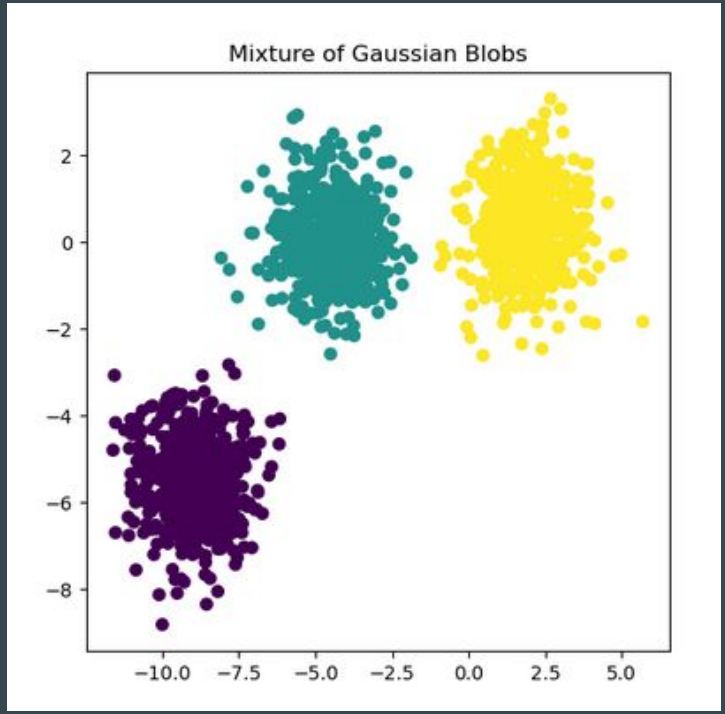
The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until the centroids do not move significantly.

# K-means

Inertia is a measure of how coherent clusters are internally

- inertia makes the assumption that clusters are convex and isotropic
- It responds poorly to elongated clusters, or manifolds with irregular shapes

sklearn

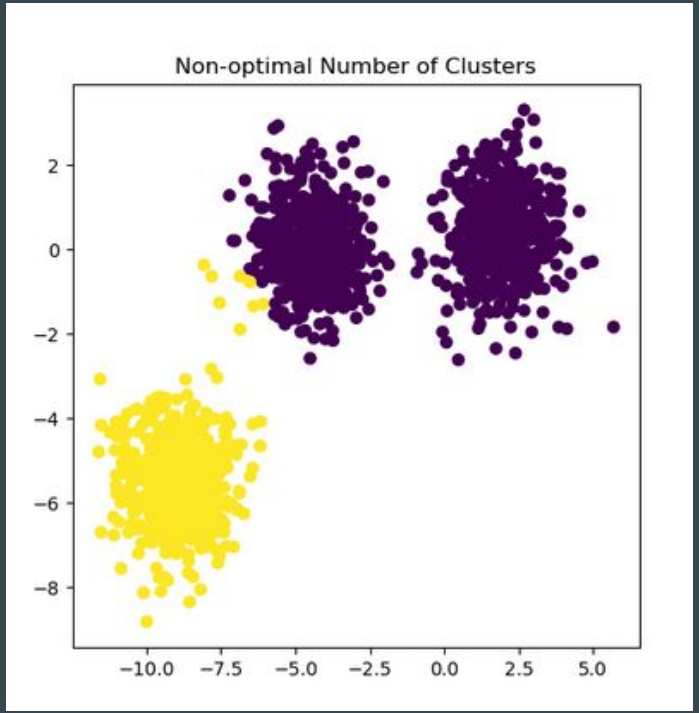


# K-means

Inertia is a measure of how coherent clusters are internally

- inertia makes the assumption that clusters are convex and isotropic
- It responds poorly to elongated clusters, or manifolds with irregular shapes

sklearn

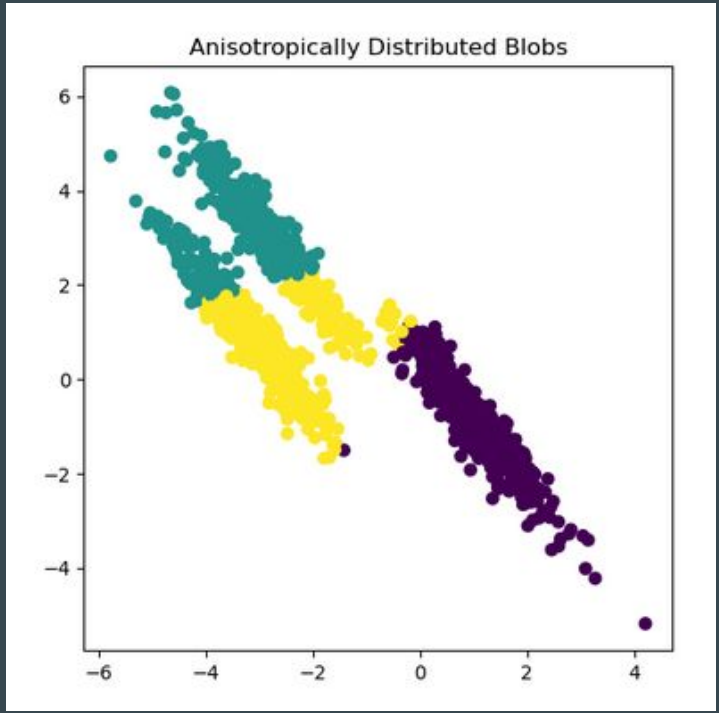


# K-means

Inertia can be recognized as a measure of how coherent clusters are internally

- inertia makes the assumption that clusters are convex and isotropic
- It responds poorly to elongated clusters, or manifolds with irregular shapes

sklearn



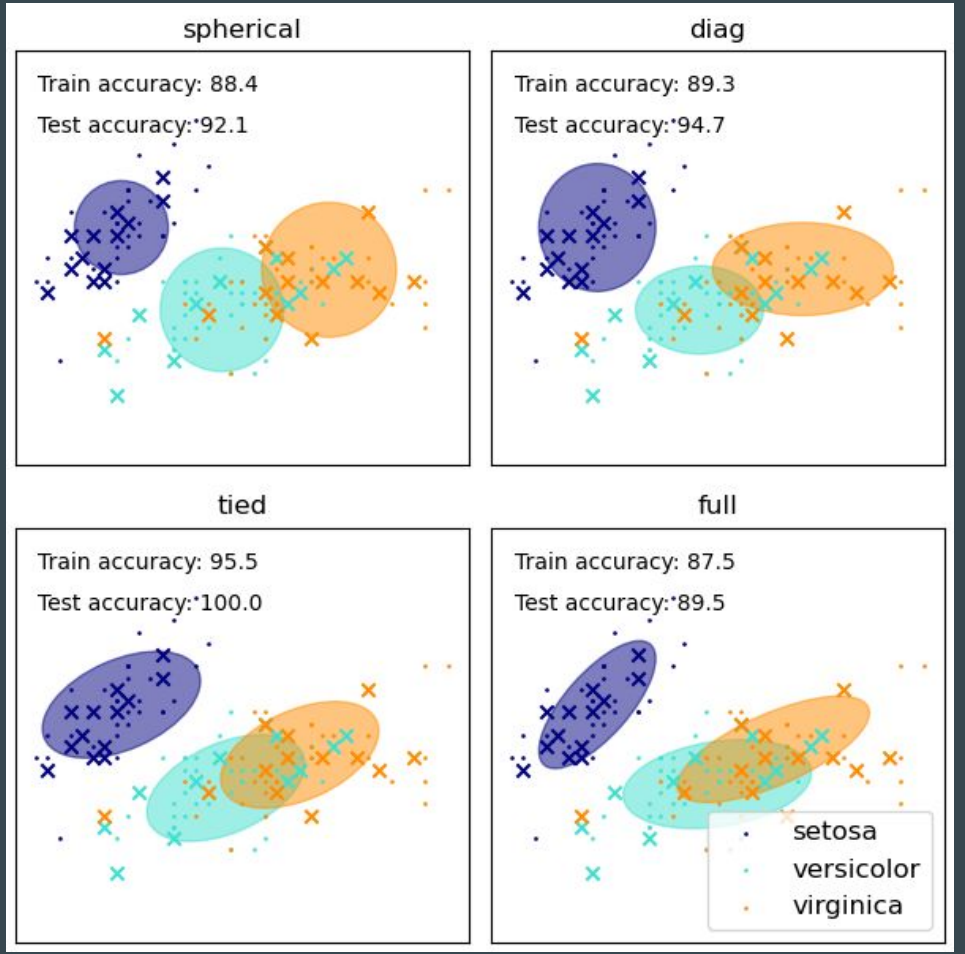
# Gaussian Mixture Model (GMM)

A Gaussian mixture model (GMM) is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering that incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. GMM estimates the parameters of the distributions using the Expectation-Maximization (EM) algorithm and assigns each data point to the most likely cluster.

# GMM

Although GMM are often used for clustering, we can compare the obtained clusters with the actual classes from the dataset. By initializing the means of the Gaussians with the means of the classes from the training set to make this comparison valid.

sklearn

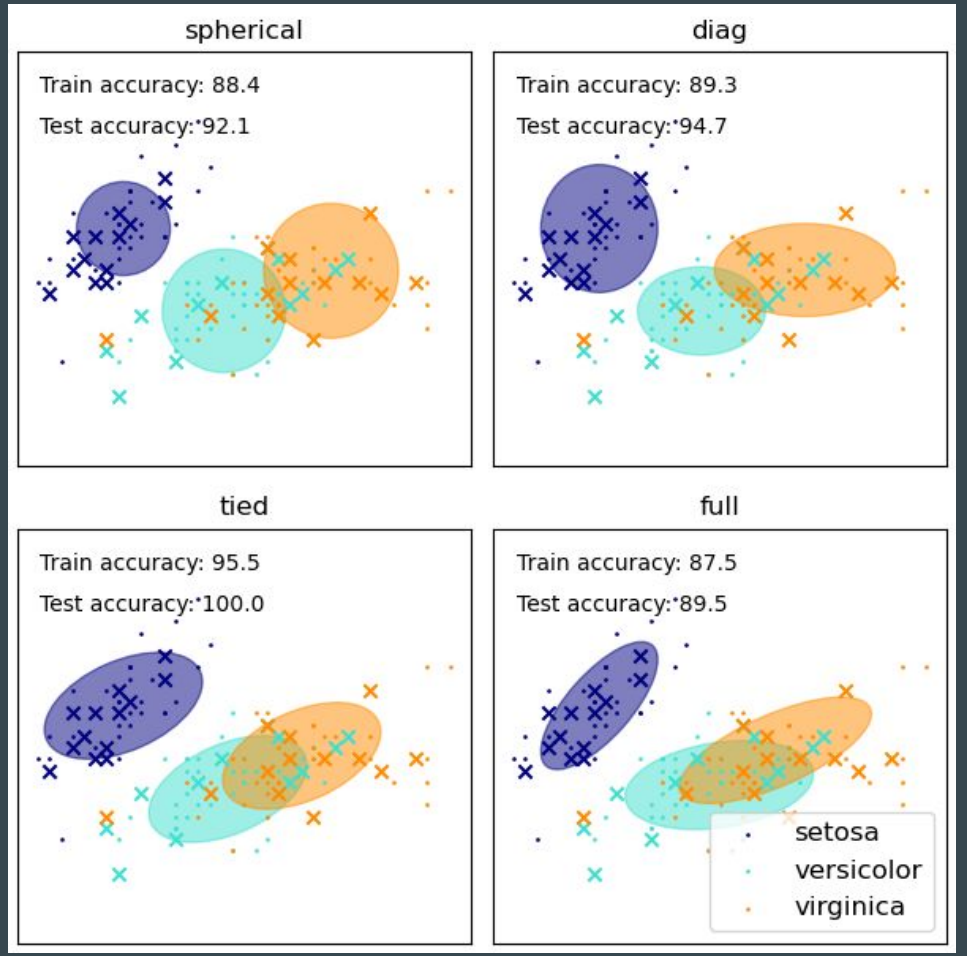


# GMM

The predicted labels on both training and test data can be plotted using a variety of GMM covariance types on the iris dataset.

Train data is shown as dots, while test data is shown as crosses. The iris dataset is four-dimensional. Only the first two dimensions are shown here, and thus some points are better separated in other dimensions.

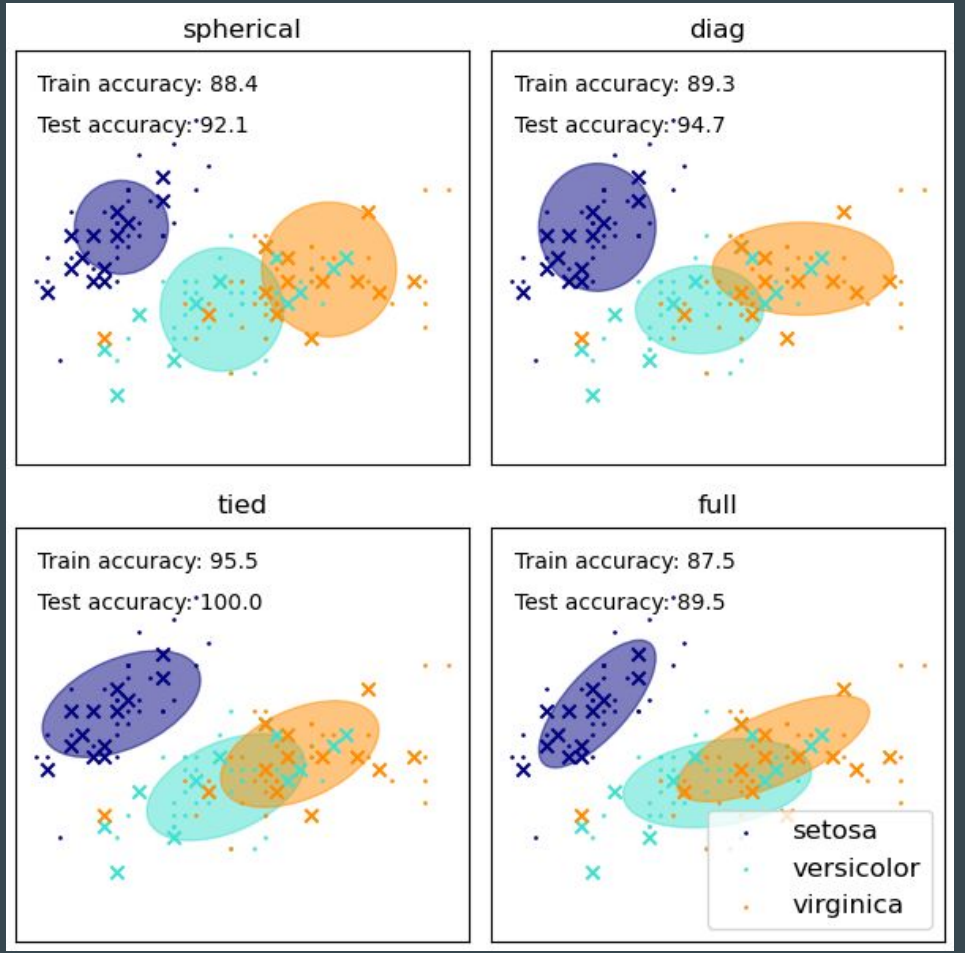
sklearn



# GMM

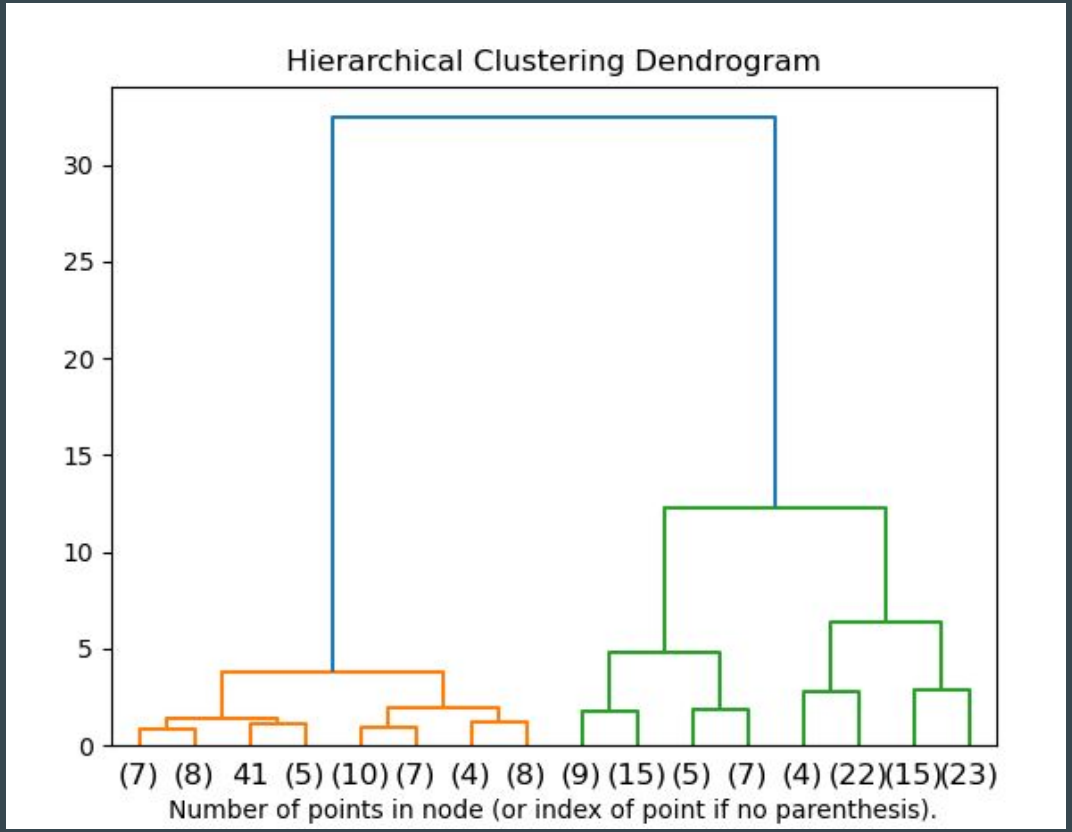
We compare GMMs with spherical, diagonal, full, and tied covariance matrices in increasing order of performance. Although one would expect full covariance to perform best in general, it is prone to overfitting on small datasets and does not generalize well to held out test data.

sklearn



# Hierarchical clustering

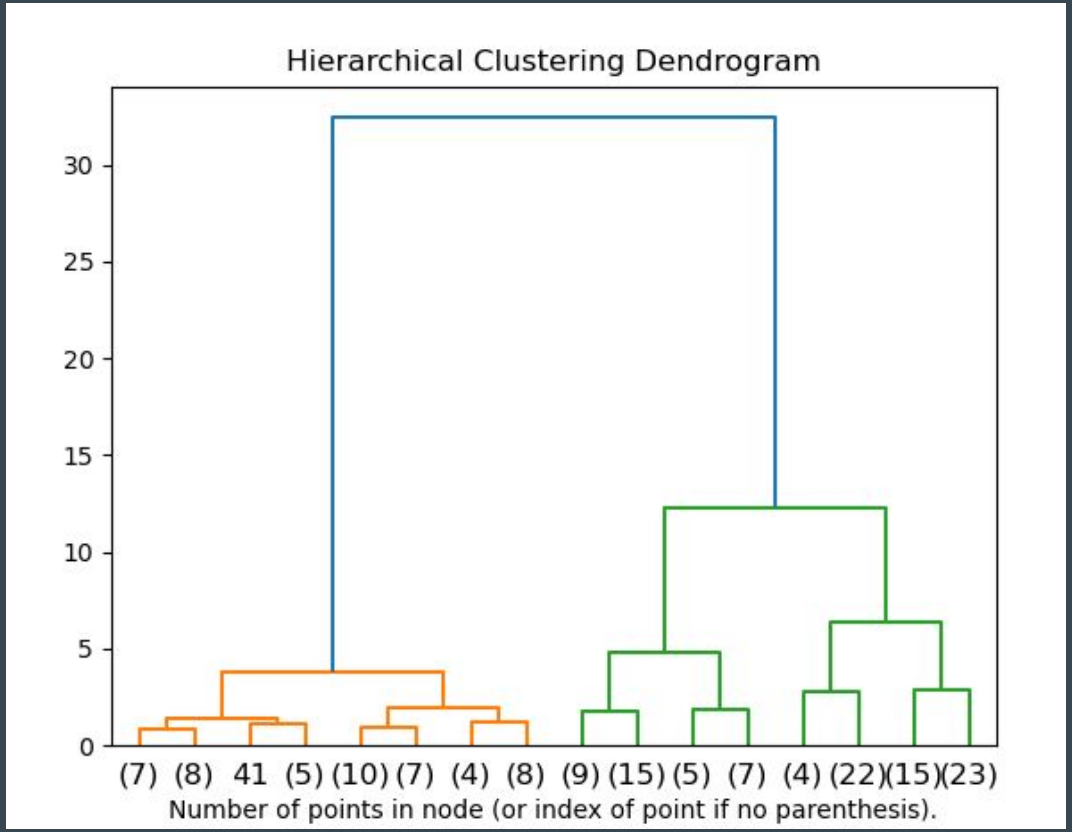
Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.




sklearn

# Hierarchical clustering

There are two main types of hierarchical clustering: agglomerative and divisive. Agglomerative clustering starts with each item as its own cluster and then recursively merges clusters based on their similarity until a single, all-encompassing cluster is formed. Divisive clustering, on the other hand, starts with all items in a single cluster and recursively splits them into smaller, more specialized clusters.




sklearn

# Agglomerative clustering

The Agglomerative Clustering algorithm performs a hierarchical clustering using a bottom up approach: each observation starts in its own cluster, and clusters are successively merged together. The linkage criteria determines the metric used for the merge strategy.

# Agglomerative clustering (Single)

Single linkage minimizes the distance between the closest observations of pairs of clusters. The distance between two clusters is defined as the shortest distance between any two items in the clusters.

# Agglomerative clustering (Maximum)

Maximum or complete linkage minimizes the maximum distance between observations of pairs of clusters.  The distance between two clusters is defined as the maximum distance between any two items in the clusters.

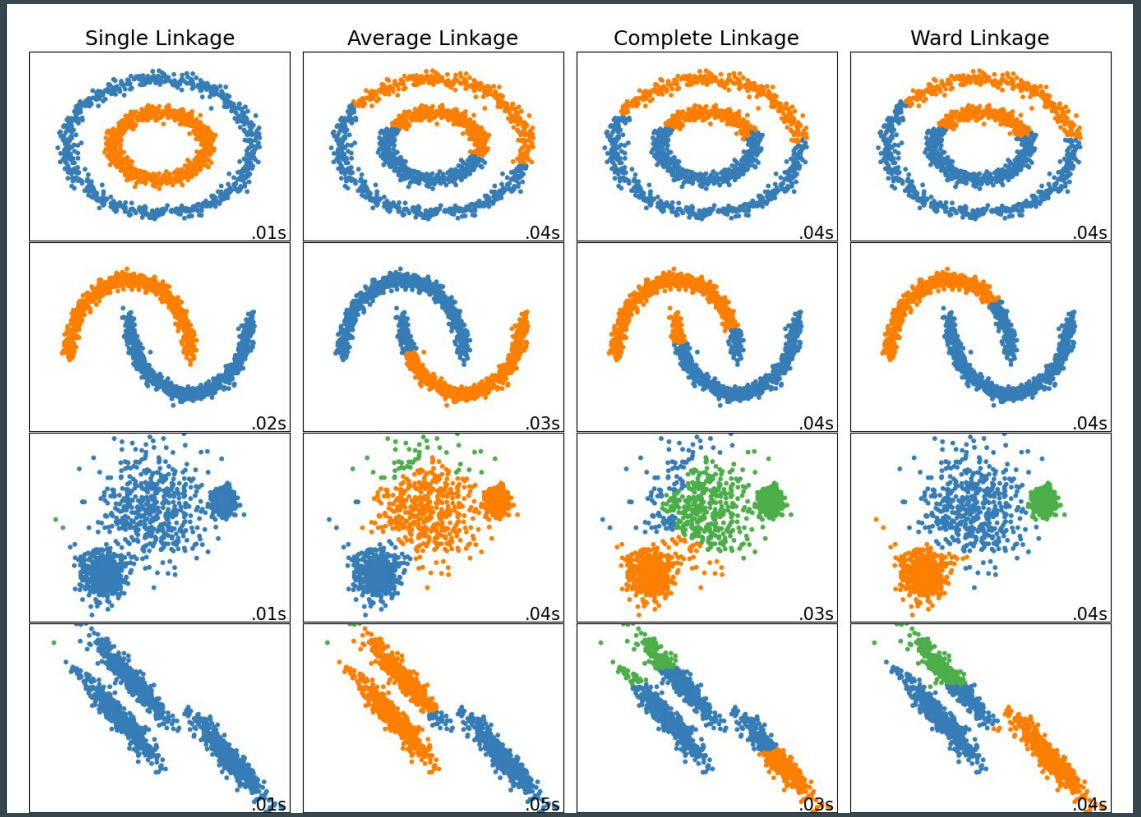
# Agglomerative clustering (Average)

Average linkage minimizes the average of the distances between all observations of pairs of clusters. The distance between two clusters is defined as the average distance between all pairs of items in the clusters.

# Agglomerative clustering (Ward)

Ward minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.  The distance between two clusters is defined as the increase in variance that results from merging the clusters.

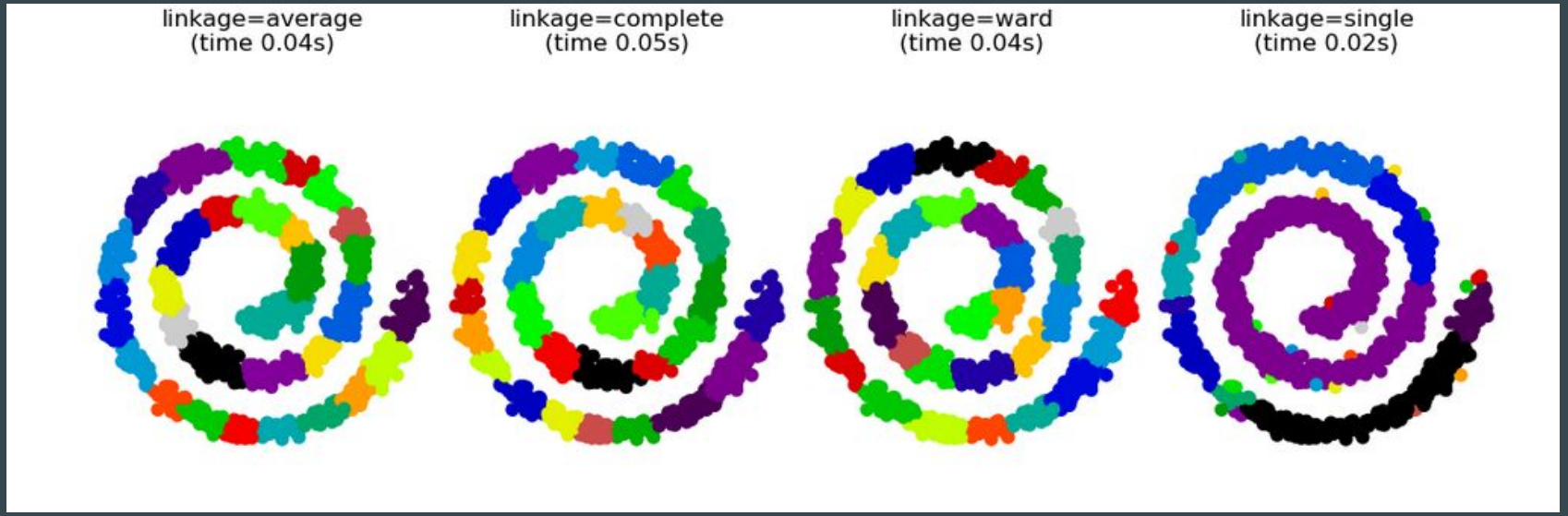
# Agglomerative clustering




sklearn

# Agglomerative clustering

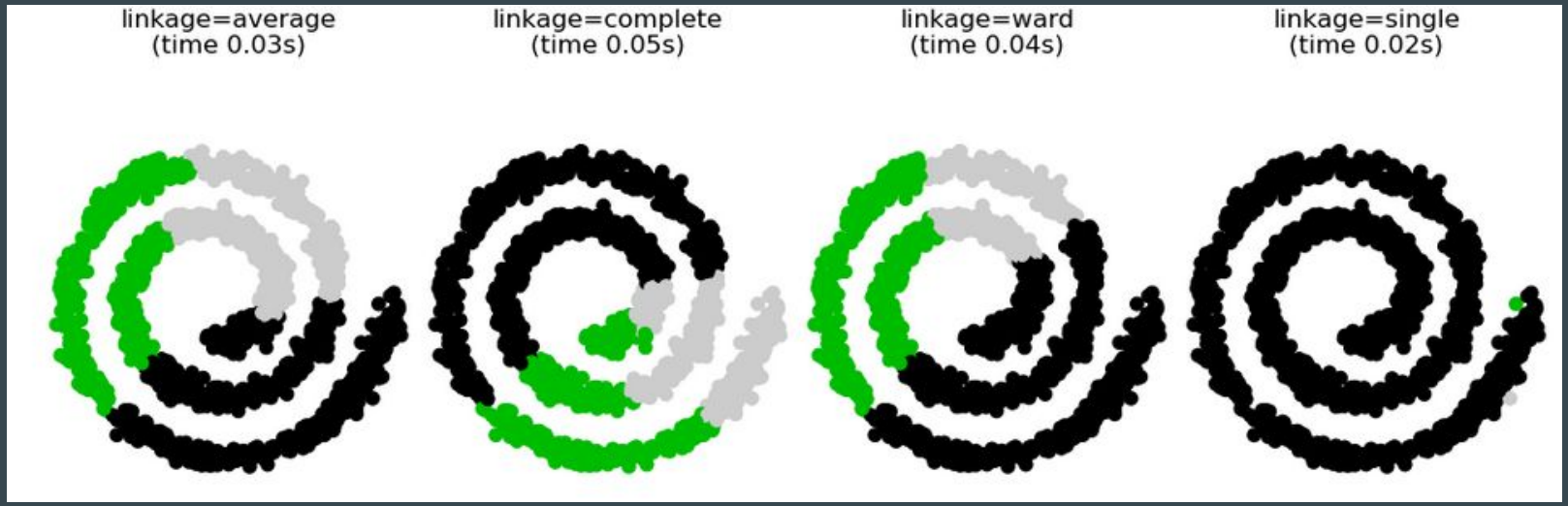
Clusters: 30




sklearn

# Agglomerative clustering

Clusters: 3




sklearn

# Agglomerative clustering (connectivity constraint)

An interesting aspect of Agglomerative Clustering is that connectivity constraints can be added to this algorithm (only adjacent clusters can be merged together), through a connectivity matrix that defines for each sample the neighboring samples following a given structure of the data.

# Agglomerative clustering (connectivity constraint)

There are two advantages of imposing a connectivity.

First, clustering with a connectivity matrix is much faster.

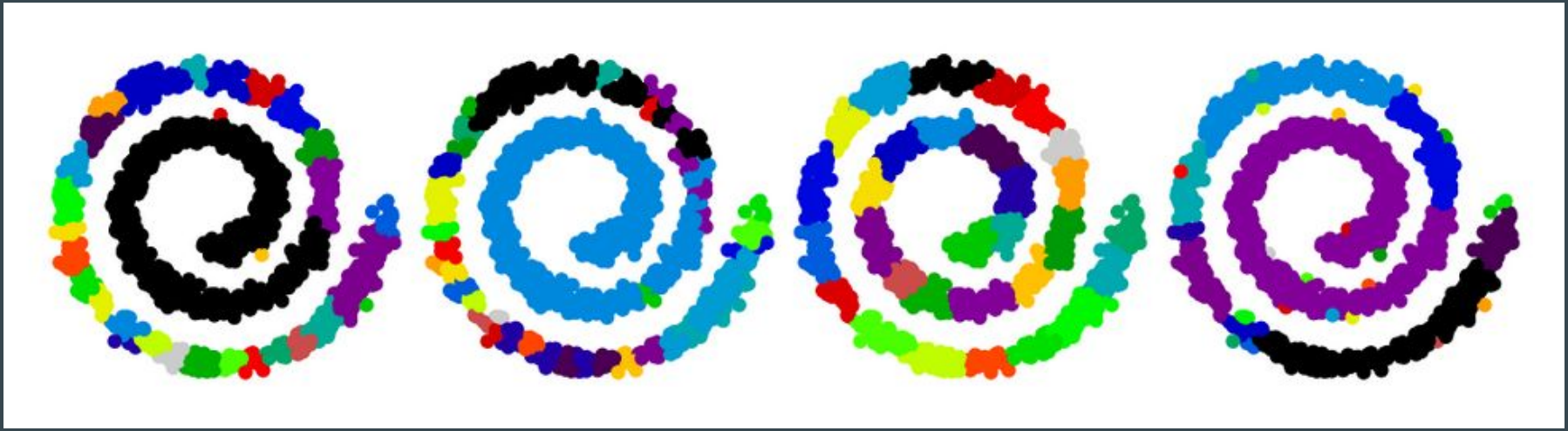
Second, when using a connectivity matrix, single, average and complete linkage are unstable and tend to create a few clusters that grow very quickly. Indeed, average and complete linkage fight this percolation behavior by considering all the distances between two clusters when merging them ( while single linkage exaggerates the behaviour by considering only the shortest distance between clusters). The connectivity graph breaks this mechanism for average and complete linkage, making them resemble the more brittle single linkage.

# Agglomerative clustering (connectivity constraint)

For instance, in the swiss-roll example above, the connectivity constraints forbid the merging of points that are not adjacent on the swiss roll, and thus avoid forming clusters that extend across overlapping folds of the roll.

# Agglomerative clustering (connectivity constraint)

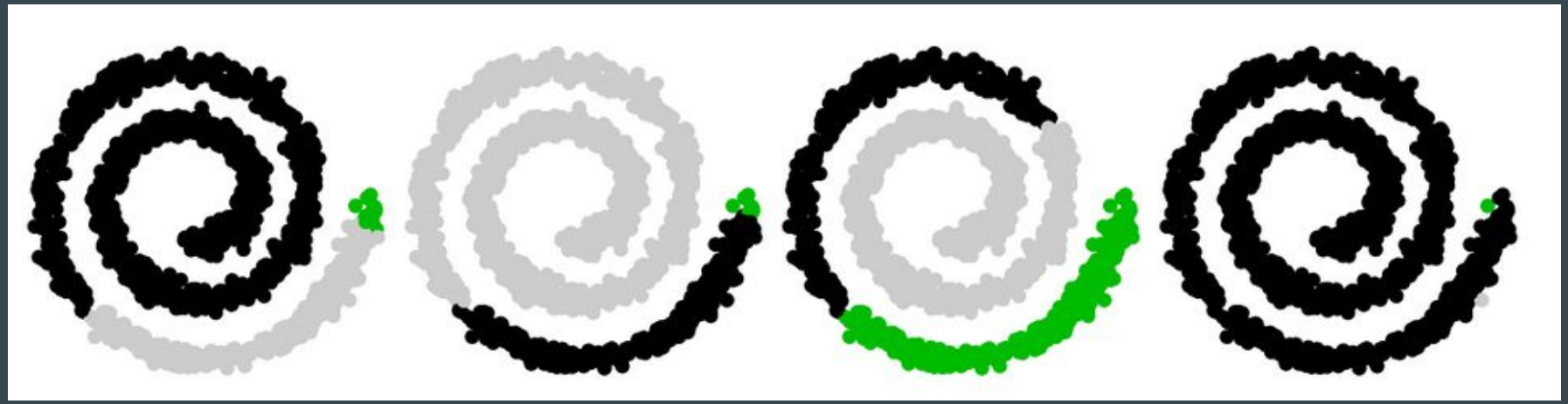
Clusters: 30



sklearn

# Agglomerative clustering (connectivity constraint)

Clusters: 3



sklearn

# DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. Due to this rather generic view, clusters found by DBSCAN can be any shape, as opposed to k-means which assumes that clusters are convex shaped. The central component to the DBSCAN is the concept of core samples, which are samples that are in areas of high density.

# DBSCAN

A cluster is therefore a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples).

There are two parameters (eps, min_samples) to the algorithm which define formally what we mean when we say dense. Higher min_samples or lower eps indicate higher density necessary to form a cluster.

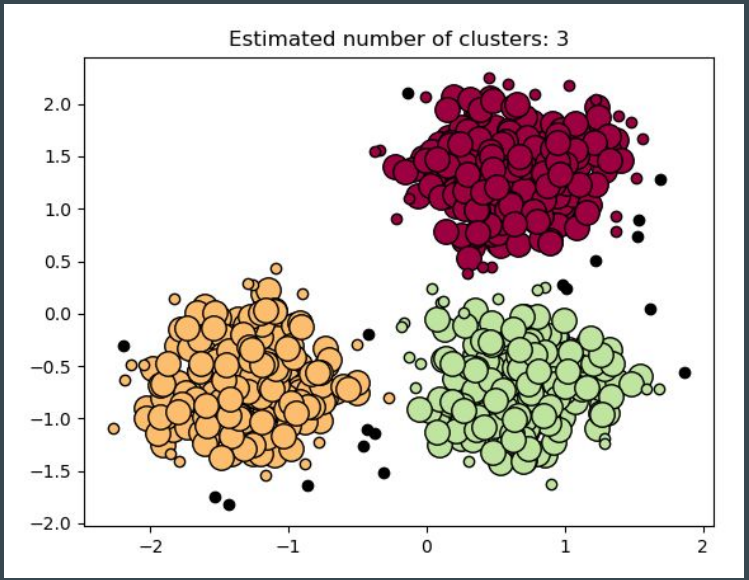
# DBSCAN

Epsilon (ε): This parameter determines the radius of the neighborhood around each point. Points within this radius are considered to be part of the same cluster.

Min samples: This parameter specifies the minimum number of points that must be within the ε-neighborhood of a core point. Core points are the central points in a cluster, and they form the backbone of the clustering process.

# DBSCAN

Color indicates cluster membership, with large circles indicating core samples found by the algorithm. Smaller circles are non-core samples that are still part of a cluster. Moreover, the outliers are indicated by black points.




sklearn

# BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

BIRCH is a hierarchical clustering algorithm and it is designed to efficiently cluster large datasets by constructing a compact representation of the data in memory, and then performing clustering on this representation.

# BIRCH

The BIRCH algorithm has three main components:

Clustering Feature (CF): BIRCH uses a clustering feature that compresses the data into a compact summary. The CF Subclusters hold the necessary information for clustering, which prevents the need to hold the entire input data in memory. The CF consists of a set of sub-clusters, each with a center, a radius, and a count of the number of points in the sub-cluster. The clustering feature is updated as new data points are added to the dataset. =

# BIRCH

CF Tree: BIRCH organizes the clustering features into a tree structure called a CF tree. The CF tree is a hierarchical structure that enables BIRCH to efficiently cluster the data in a top-down manner. Each node in the tree corresponds to a sub-cluster, and the tree is constructed by recursively merging sub-clusters until the entire dataset is represented by a single root node.

Clustering: Once the CF tree is constructed, BIRCH performs clustering by traversing the tree in a bottom-up manner. At each node, BIRCH checks whether the sub-cluster represented by the node is dense enough to be considered a cluster. If so, the sub-cluster is assigned a cluster label, and the process continues up the tree.

# BIRCH

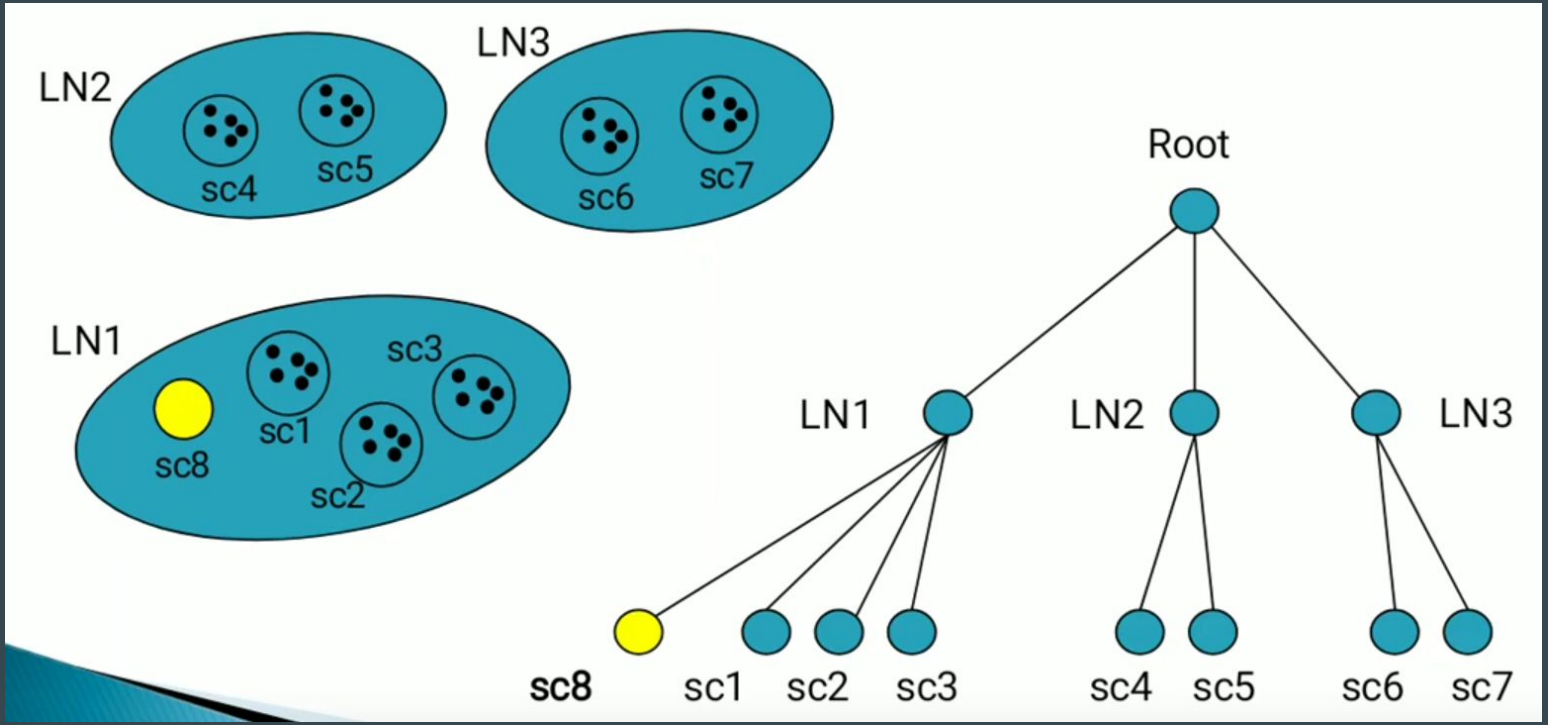
The BIRCH algorithm has two parameters, the threshold and the branching factor. The branching factor limits the number of subclusters in a node and the threshold limits the distance between the entering sample and the existing subclusters.

# BIRCH

Each new sample is inserted into the root of the CF Tree. BIRCH finds the leaf node in the CF tree that is closest to the new sample and updates the summary in that node to include the new point. If the radius of the subcluster obtained by merging the two nearest subclusters is greater than the threshold, BIRCH splits the node into two new sub-nodes, and updates the summary statistics accordingly.

Once the CF tree is constructed, BIRCH performs clustering by traversing the tree in a bottom-up manner. At each node, BIRCH checks whether the sub-cluster represented by the node is dense enough to be considered a cluster. If so, the sub-cluster is assigned a cluster label, and the process continues up the tree.

# BIRCH

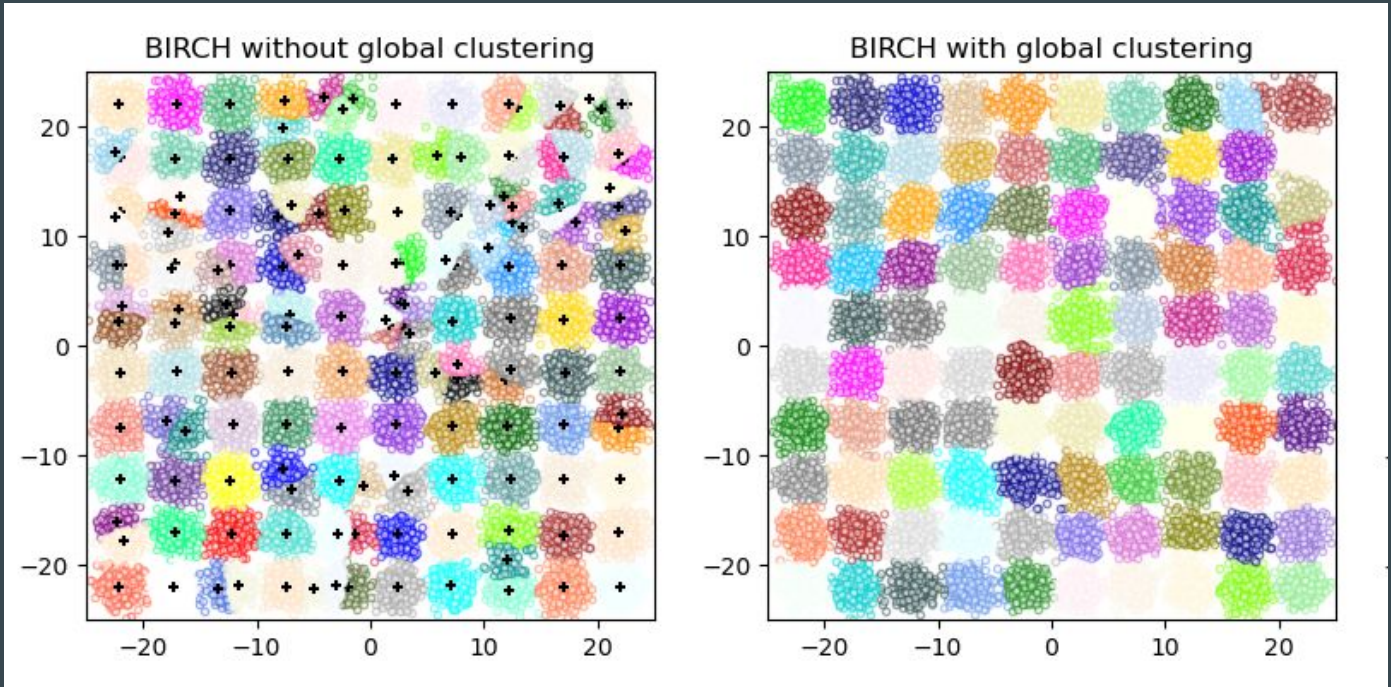

LN2
sc4
sc5
LN3
sc6
sc7
LN1
sc8
sc1
sc3
sc2
Root
LN1
LN2
LN3
sc8
sc1
sc2
sc3
sc4
sc5
sc6
sc7

# BIRCH

This algorithm can be viewed as an instance of data reduction method, since it reduces the input data to a set of subclusters which are obtained directly from the leaves of the CFT.

If the number of clusters is selected, a global clustering step labels these subclusters into global clusters (labels) and the samples are mapped to the global label of the nearest subcluster. If n_clusters is set to None, the subclusters from the leaves are directly used.

# BIRCH




sklearn

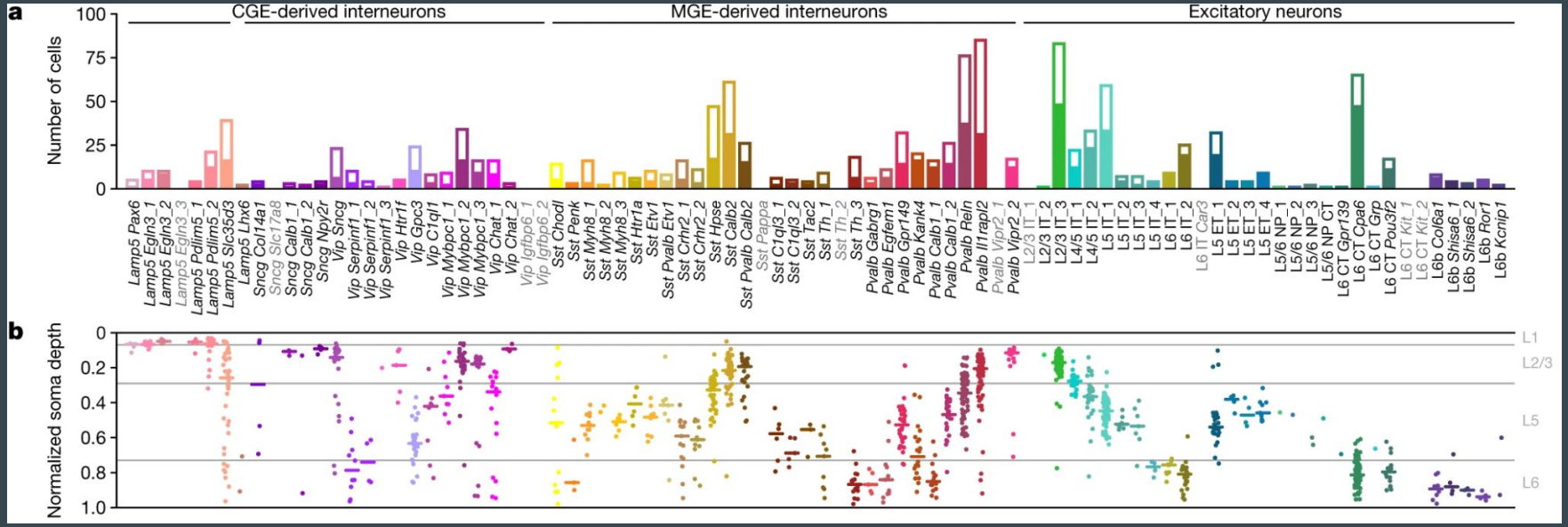
# Phenotypic variation of transcriptomic cell types in mouse motor cortex

Scala, Kobak, Bernabucci et al.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex

Cortical neurons exhibit extreme diversity in gene expression as well as in morphological and electrophysiological properties. Most existing neural taxonomies are based on either transcriptomic or morpho-electric criteria, as it has been technically challenging to study both aspects of neuronal diversity in the same set of cells. Here we used Patch-seq to combine patch-clamp recording, biocytin staining, and single-cell RNA sequencing of more than 1,300 neurons in adult mouse primary motor cortex, providing a morpho-electric annotation of almost all transcriptomically defined neural cell types. We found that, although broad families of transcriptomic types (those expressing Vip, Pvalb, Sst and so on) had distinct and essentially non-overlapping morpho-electric phenotypes, individual transcriptomic types within the same family were not well separated in the morpho-electric space. Instead, there was a continuum of variability in morphology and electrophysiology, with neighbouring transcriptomic cell types showing similar morpho-electric features, often without clear boundaries between them. Our results suggest that neuronal types in the neocortex do not always form discrete entities. Instead, neurons form a hierarchy that consists of distinct non-overlapping branches at the level of families, but can form continuous and correlated transcriptomic and morpho-electrical landscapes within families.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex

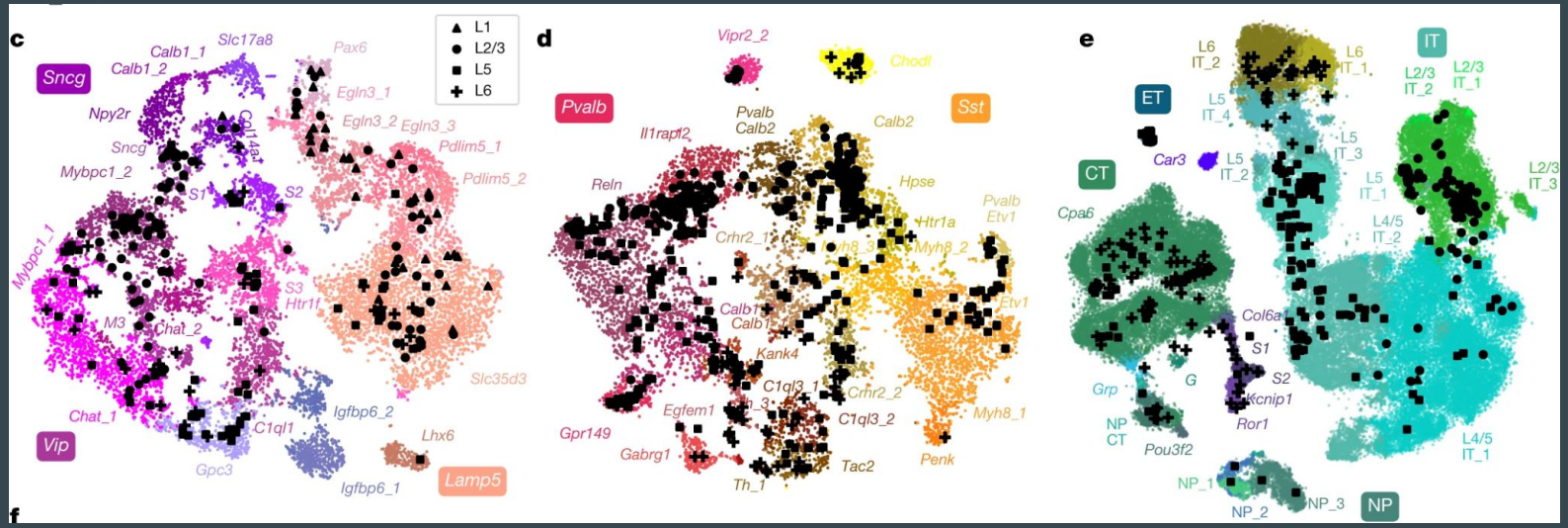



(a) Number of Patch-seq cells assigned to each of the neural transcriptomic types (t-types). (b) Normalized soma depths of all neurons of each t-type.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex

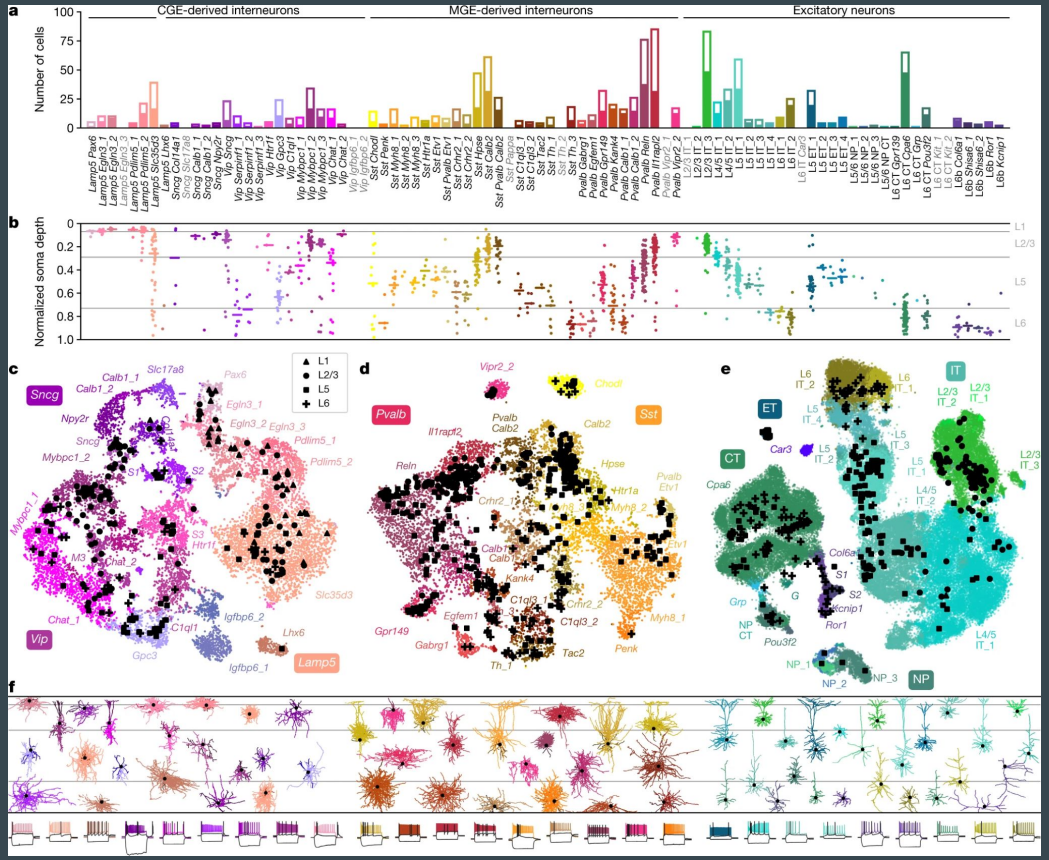
Example morphologies coloured by t-type. For interneurons, dendrites are shown in darker colours. For excitatory neurons, only dendrites are shown. Black dots mark soma locations. Voltage traces are shown below for some exemplary cells: the hyperpolarization trace obtained with the smallest current stimulation, the first depolarization trace that elicited at least one action potential, and the depolarization trace showing maximal firing rate. Stimulation length, 600 ms.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex




(c) t-SNE representation of CGE-derived interneurons (n = 15K, perplexity: 30). (d) MGE-derived interneurons (n = 12K, perplexity, 30). (e) Excitatory neurons (n = 93K; perplexity: 100).

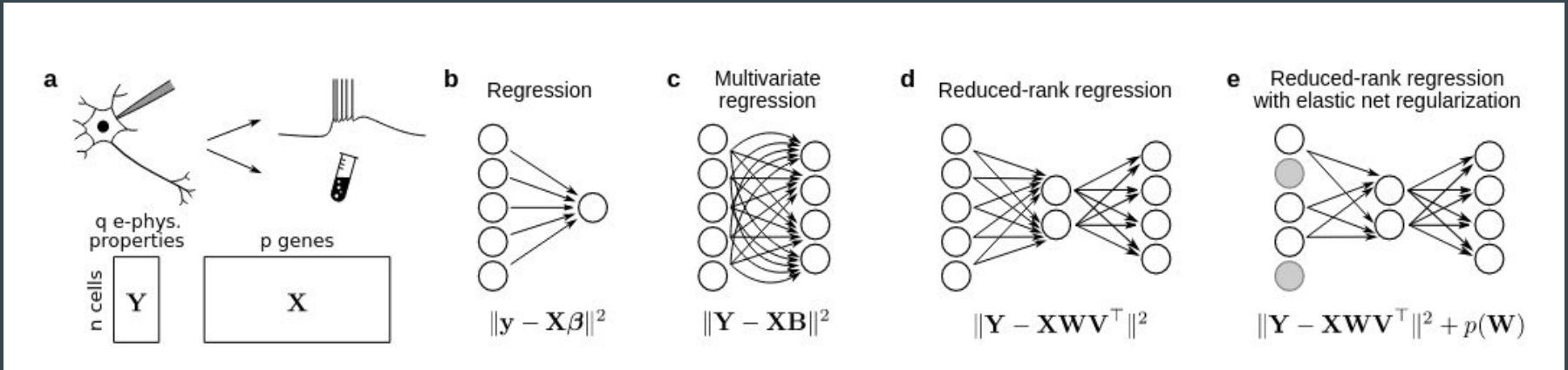
# Phenotypic variation of transcriptomic cell types in mouse



# Sparse reduced-rank regression for exploratory visualisation of paired multivariate data

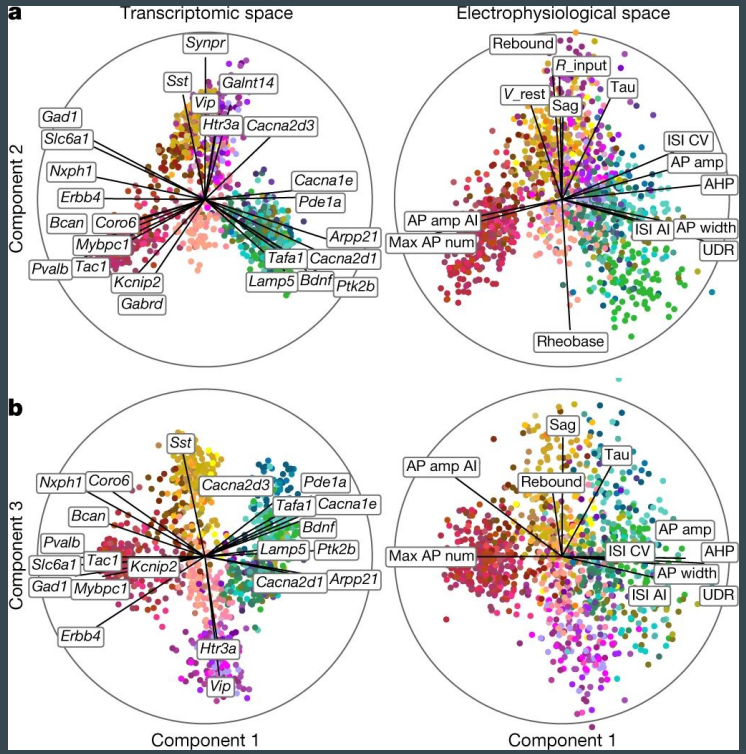
Kobak et al. 2021

# Sparse reduced-rank regression



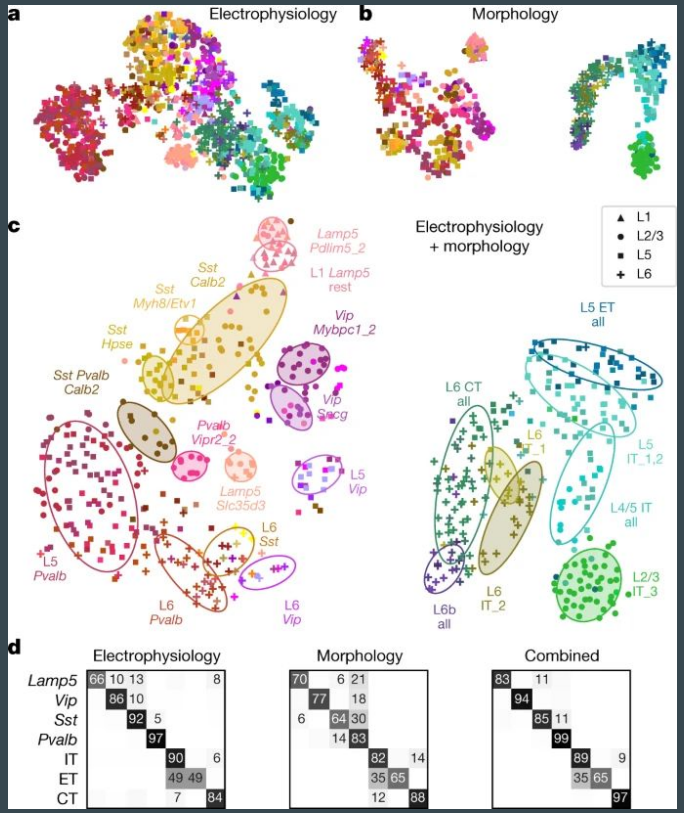

(a) Schematic illustration of a Patch- seq experiment: electrophysiological activity is recorded by patch clamping, followed by RNA extraction and sequencing. Below: data matrices after computational characterisation of electrophysiological properties (Y) and estimation of gene counts (X). (b–e). Schematic illustrations and loss functions for several regression methods. (b) Simple regression. (c) Multivariate regression. (d) Reduced- rank regression. (e) Regularised reduced-rank regression. Grey circles denote predictors that are left out of the sparse model

# Phenotypic variation of transcriptomic cell types in mouse motor cortex



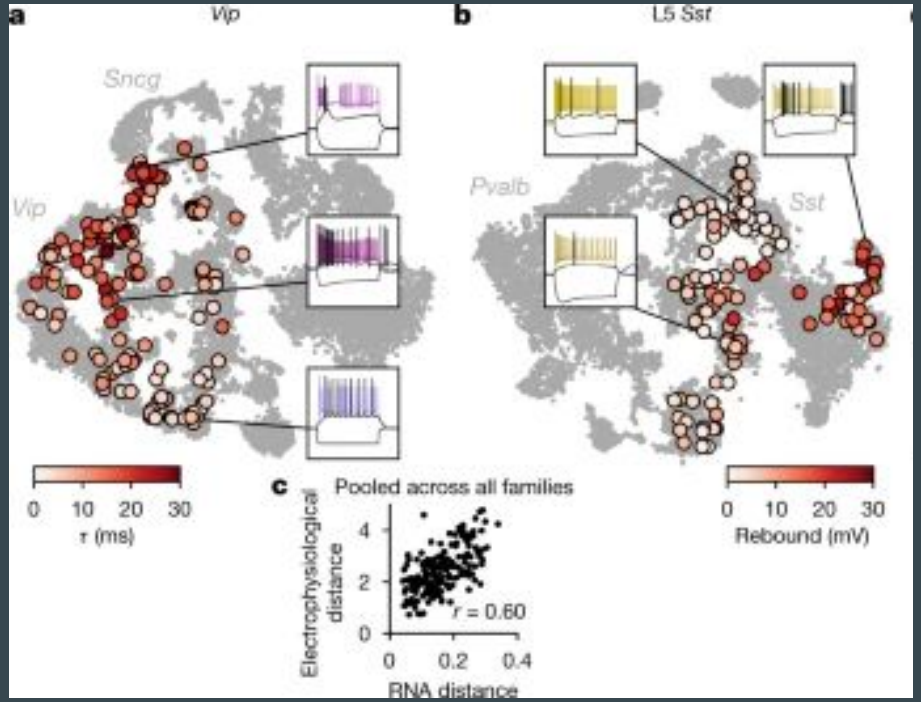

A sparse reduced-rank regression (RRR) model to predict combined electrophysiological features from gene expression. Transcriptomic data are linearly projected to a low-dimensional space that allows reconstruction of electrophysiological data; components 1 and 2 (a) and 1 and 3 (b) of rank-5 model are shown. n = 1,219. Colour corresponds to t-type. The model selected 25 genes (left).

# Phenotypic variation of transcriptomic cell types in mouse motor cortex



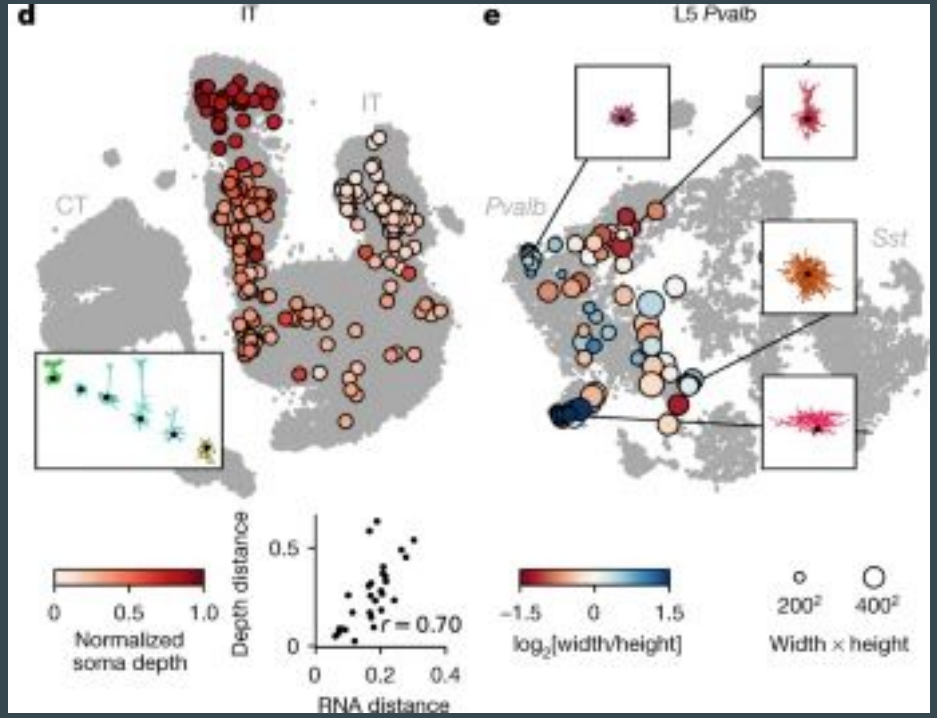

t-SNE embedding (perplexity=30) using (a) electrophysiological features, (b) using morphometric features and z-profiles and (c) using electrophysiological and morphological features. Ellipses show 80% coverage for the most prominent t-types (shaded) and for some groups of related t-types and some layer-restricted families (unshaded). We chose these groups to reduce the overlap between ellipses. (d) Confusion matrices for classifying cells into seven transcriptomic families using kNN classifier (k = 10) and three feature sets. Each row shows what fraction of cells from a given family is classified in each of the seven families. The values in each row sum to 100% but only values above 5% are shown.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex



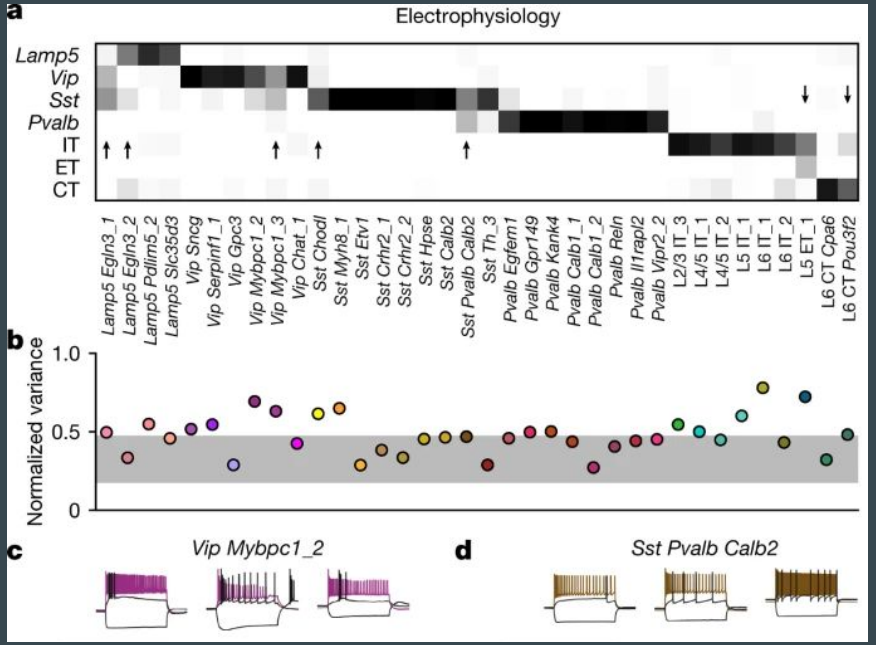

Vip neurons mapped to the reference t-SNE embedding from Fig. 1c, coloured by membrane time constant (τ). Correlation between transcriptomic distances and electrophysiological distances across all 200 pairs of t-types from the same family (for 50 t-types with at least 5 cells), pooled across all families. Transcriptomic distance was computed based on correlation between average log-expression across most variable genes. Electrophysiological distance is Euclidean distance between the average feature vectors.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex




IT neurons mapped to the reference t-SNE embedding from Fig. 1e, coloured by normalized soma depth. Inset, examples of IT neurons at different depths, coloured by t-type. Scatter plot used eight t-types with at least five cells and shows correlation between transcriptomic distances and cortical depth distances. Cortical depth distance is Euclidean distance between the average normalized soma depths. e, Pvalb neurons from layer 5 mapped to the reference t-SNE embedding from Fig. 1d, coloured by axonal width/height log-ratio. Insets, example morphologies.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex




a, Confusion matrix for classifying cells from each t-type into seven transcriptomic families using electrophysiological features. Only t-types with at least ten cells are shown. Arrows mark t-types that are classified into wrong families more than 25% of the time. We used a kNN-based classifier with k = 10. b, Normalized total variance of features in each t-type. c, Three exemplary traces from Vip Mybpc1_2 cells (all with confidence ≥ 95%) and t-SNE overlay coloured by rebound. d, Three exemplary traces from Sst Pvalb Calb2 cells (confidence ≥ 95%) and t-SNE overlay coloured by maximum firing rate.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex

We used Patch-seq to provide the missing link between transcriptomic and morpho-electric descriptions of neurons in adult mouse motor cortex.

Morpho-electric properties within transcriptomic families are highly variable and this variation is structured across the transcriptomic landscape, such that the morpho-electric distance between t-types within a family is correlated with their transcriptomic distance.

# Phenotypic variation of transcriptomic cell types in mouse motor cortex

We therefore suggest that the ‘tree of cortical cell types’ may look more like a banana tree with a few large leaves, rather than an olive tree with many small ones: neurons follow a hierarchy consisting of distinct, non-overlapping branches at the level of families (large leaves), but with a spectrum of cells forming continuous and correlated transcriptomic and morpho-electrical landscapes within each leaf.

Thus, the goal to assemble an exhaustive inventory of neural cell types might be unattainable if the types, unlike the chemical elements in the periodic table, are not discrete entities.