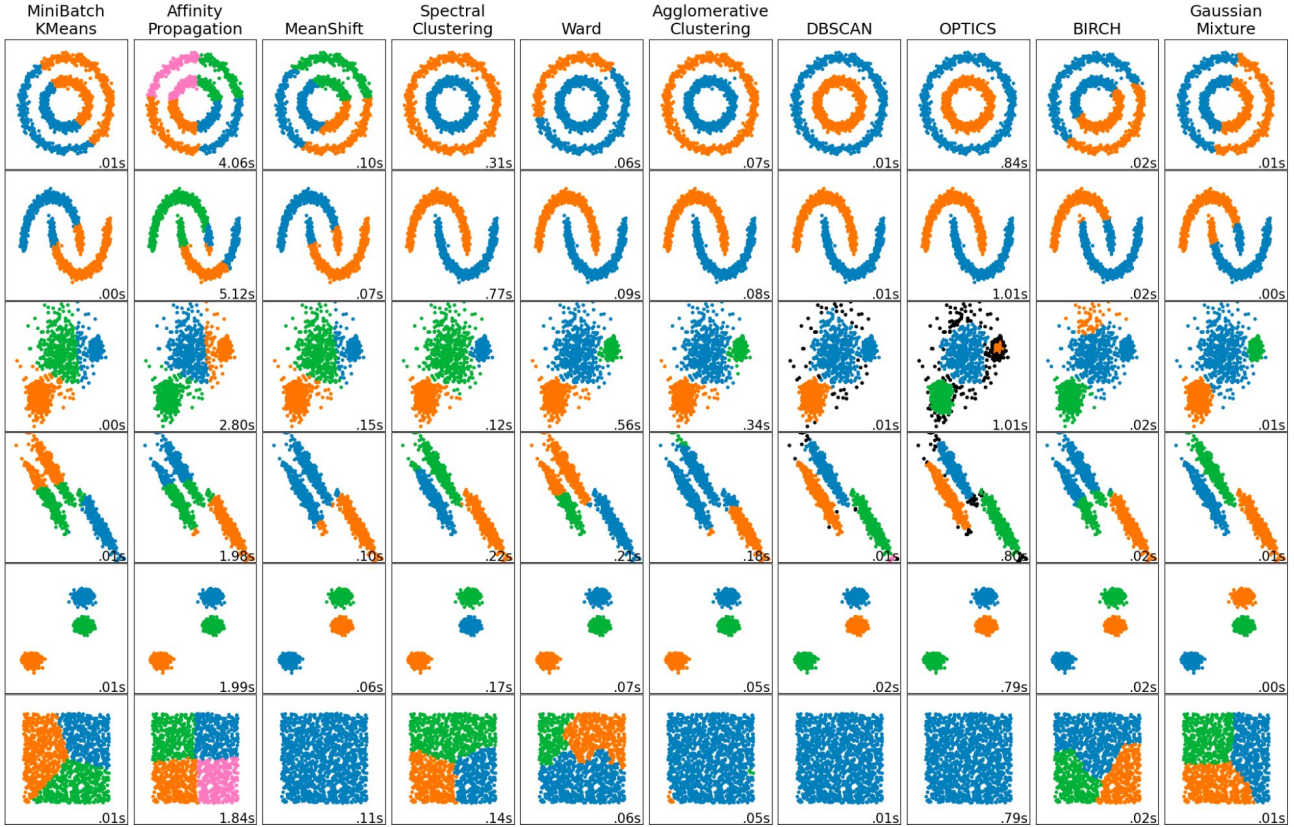
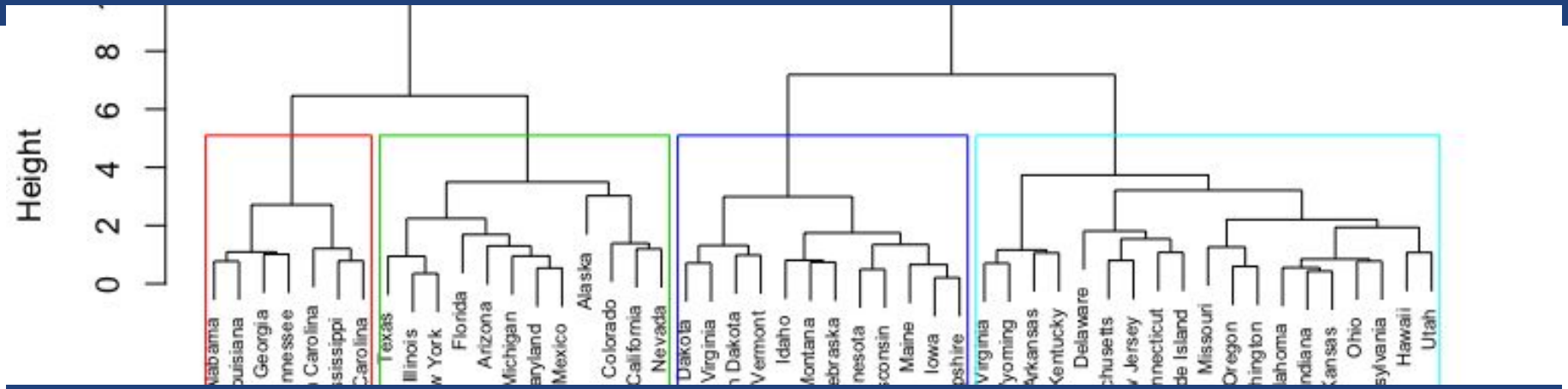
How does clustering work?

- **Partitional**
  - **Hard clustering:** K-means (McQueen ’67), K-medoids (Kaufman & Rausseeuw ’87)
  - **Soft Clustering:** Expectation Maximization (Dempster,Laird,Rubin ’77)
- **Hierarchical**
  - **Agglomerative (bottom-up)**
  - **Divisive (top-down)**
- **also:**
  - **Density based**
  - **Grid based**
  - **Model based**

Overview of clustering methods



https://scikit-learn.org/stable/modules/clustering.html



Hierarchical clustering

Hierarchical clustering

removes the issue of deciding K (number of clusters)

it calculates distance between clusters and single points: linkage

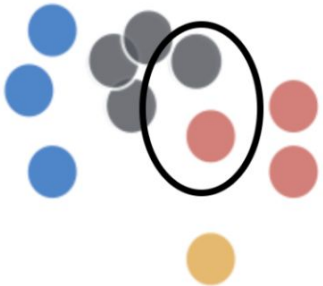
Divisive clustering (top down)

it is

non-deterministic

(like k-mean)

it is greedy -
just as k-means
two nearby points
may end up in
separate clusters



it is high complexity for

exhaustive search

$O(2^N)$

But can be reduced (~k-means)

$O(2Nk)$ or $O(N^2)$

Divisive clustering: the algorithm

1) Start with the whole dataset
2) Calculate clustering criterion for all subgroups, e.g. intracluster variance or sum of squared errors.
3) Partition the cluster into two least similar clusters
4) Take the cluster with biggest difference within cluster and split it (go to step 3 and repeat)

until

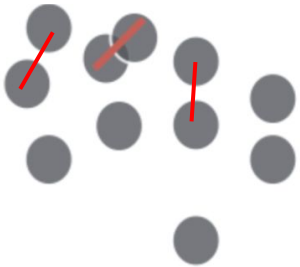
each data is in its own singleton cluster

Agglomerative clustering (bottom up)

1) Start with all data point as clusters on their own
2) Take the 2 nearest clusters and join them in one cluster

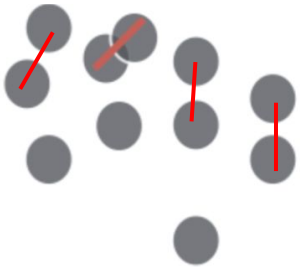
Agglomerative clustering (bottom up)

1) Start with all data point as clusters on their own
2) Take the 2 nearest clusters and join them in one cluster
3) Proceed until you have the desidered number of clusters



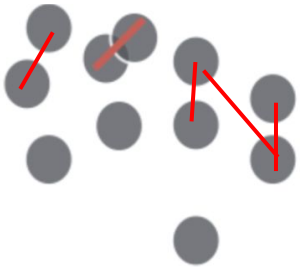
Agglomerative clustering (bottom up)

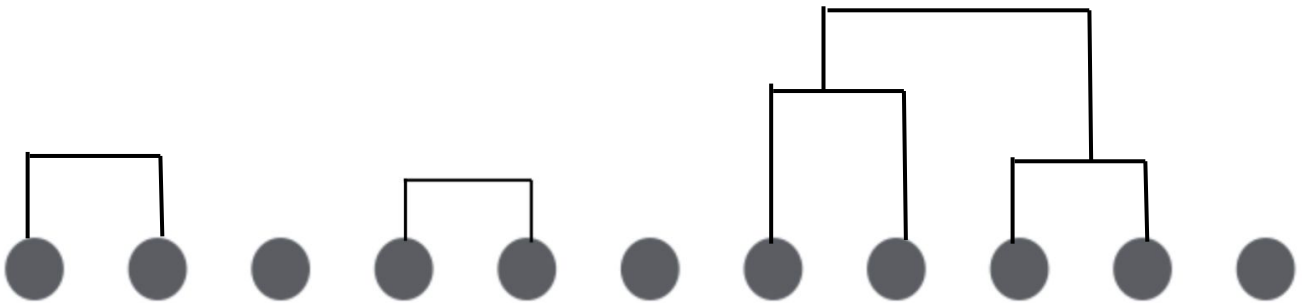
1) Start with all data point as clusters on their own
2) Take the 2 nearest clusters and join them in one cluster
3) Proceed until you have the desidered number of clusters



Agglomerative clustering (bottom up)

1) Start with all data point as clusters on their own
2) Take the 2 nearest clusters and join them in one cluster
3) Proceed until you have the desidered number of clusters





Agglomerative clustering (bottom up)

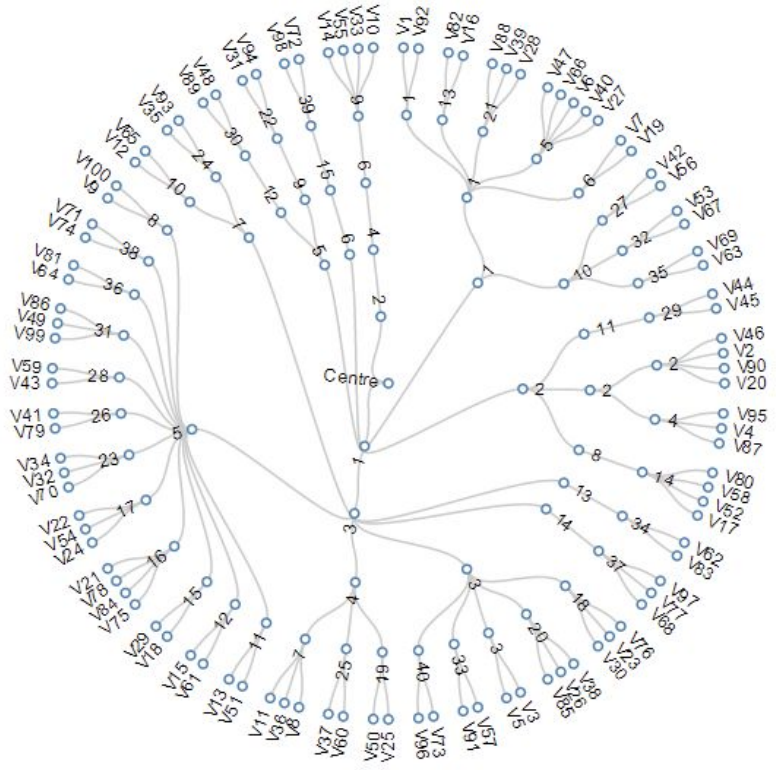
it's deterministic!

computationally intense
because every ***cluster pair***
distance has to be calculate

it is slow, though it can be optimized:

complexity

$$O(N^2 d + N^3)$$



Agglomerative clustering : the algorithm

compute the distance matrix

each data point is a singleton cluster

repeat

merge the 2 cluster with minimum distance

update the distance matrix

until

only a single (n) cluster(s) remains

Agglomerative clustering

Order: $O(N^2 d + N^3)$

PROs

It's deterministic

CONs

It's greedy (optimization is done step by step and agglomeration decisions cannot be undone)

It's computationally expensive

Agglomerative clustering: hyperparameters

class `sklearn.cluster.` **`AgglomerativeClustering`** (*n_clusters=2, affinity='euclidean', memory=None, connectivity=None, compute_full_tree='auto', linkage='ward', pooling_func='deprecated', distance_threshold=None*)
¶ [source]

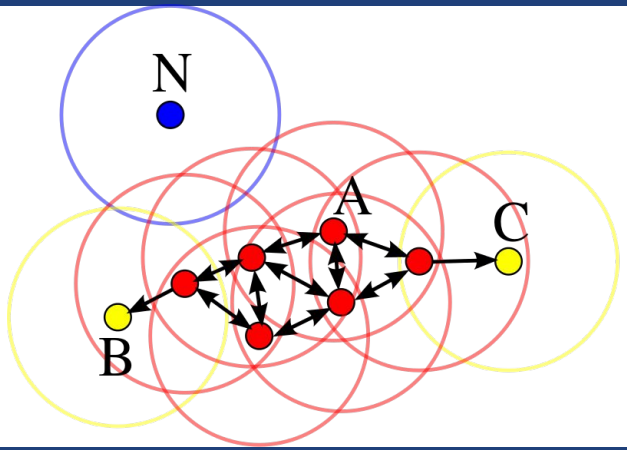
n_clusters : number of clusters

distance_threshold: The linkage distance threshold above which, clusters will not be merged

affinity : the distance/similarity definition

linkage : the scheme to measure distance to a cluster

random_state : for reproducibility



Density based (DBSCAN)

Density-based spatial clustering of applications with noise (DBSCAN)

Defines cluster membership based on local density: based on Nearest Neighbors algorithm.

- A point p is a core point if at least minPts points are within distance ε (including p).
- A point q is directly reachable from p if point q is within distance ε from core point p. Reachable from p if there is a path p1, ..., pn with p1 = p and pn = q, where each pi+1 is directly reachable from pi.
- All points not reachable from any other point are outliers or noise points.

DBSCAN: the algorithm

- for each point P count neighbours within minPts: label=C
- for each point P ~= C measure distance d to all Cs
  if d<minD: label = DR
- for each point P not C and not DR
  if distance d to C or DR > minD: label = outlier
  if distance d to C or DR <= minD: find path to closet C and cluster

DBSCAN: the hyperparameters

class `sklearn.cluster.` **`DBSCAN`** (*eps=0.5, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*) [source]

- **ε :** minimum distance to join points (very sensitive to this!!)
- **min_sample :** minimum number of points in a cluster, otherwise they are labeled outliers. (also very sensitive to this!!)
- **metric :** the distance metric
- **p :** float, optional The power of the Minkowski metric

DBSCAN: the hyperparameters

Order: $O(N^2)$

PROs

- Deterministic
- Deals with noise and outliers
- Can be used with any definition of distance or similarity

CONs

- Not entirely deterministic.
- Only works in a constant density field