Clustering
K-means & density-based scan

Johannes Müller

K-means clustering

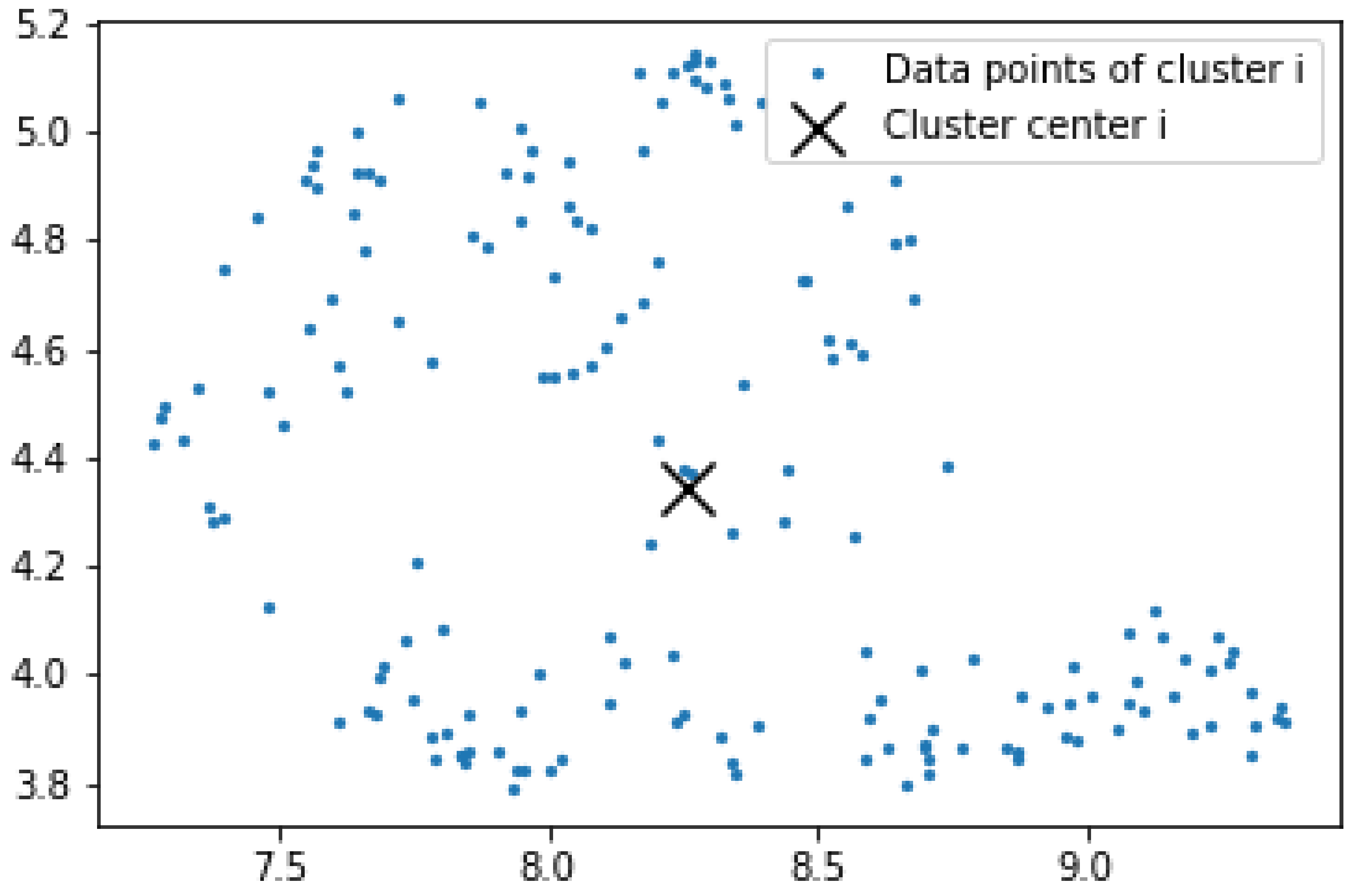
Strategy: Group data points into n groups so that variance within group is minimal

$$\sum_{i=1}^{k} \underbrace{\sum_{x_j \in S_i} \left\| x_j - \mu_i \right\|^2}_{\text{Variance within cluster i}}$$

μ_i: Center of cluster i

S_i: Cluster i

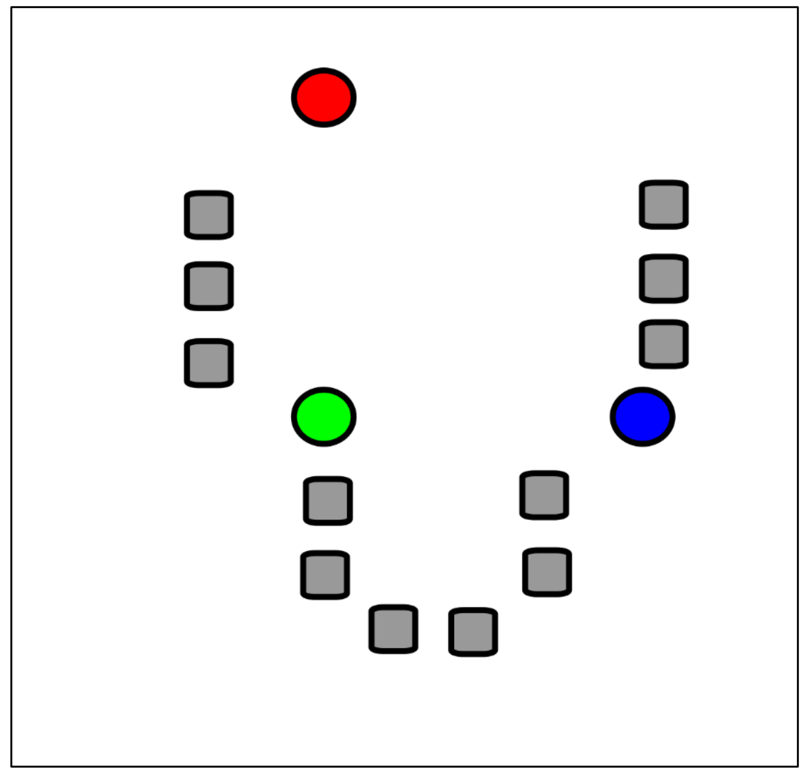
x_j: Datapoint j



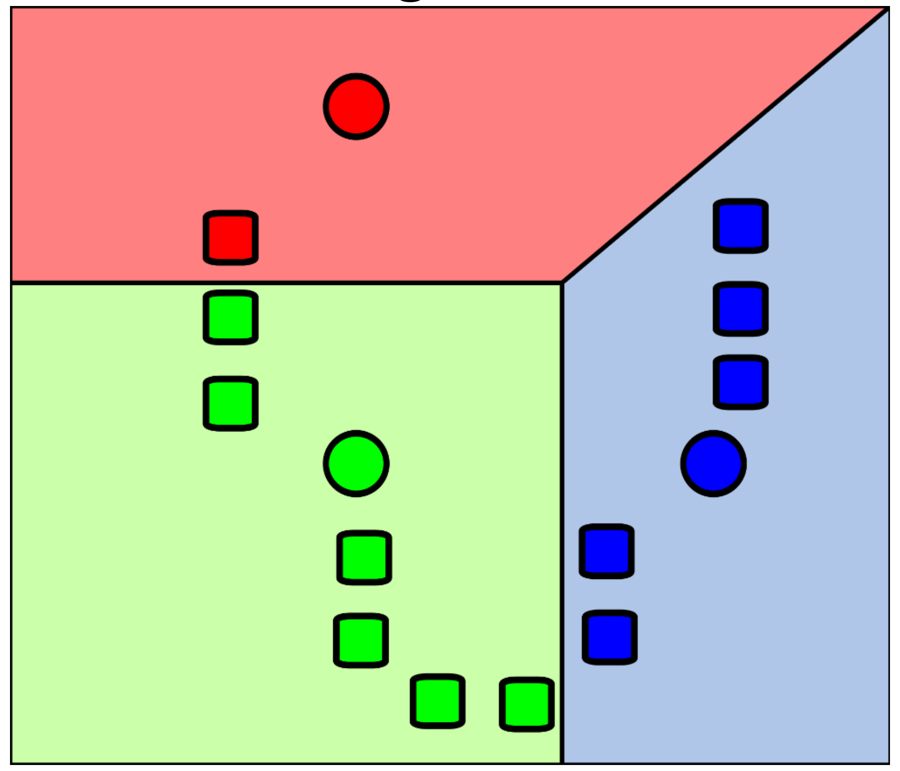
K-means clustering

Strategy: Group data points into n groups so that variance within group is minimal

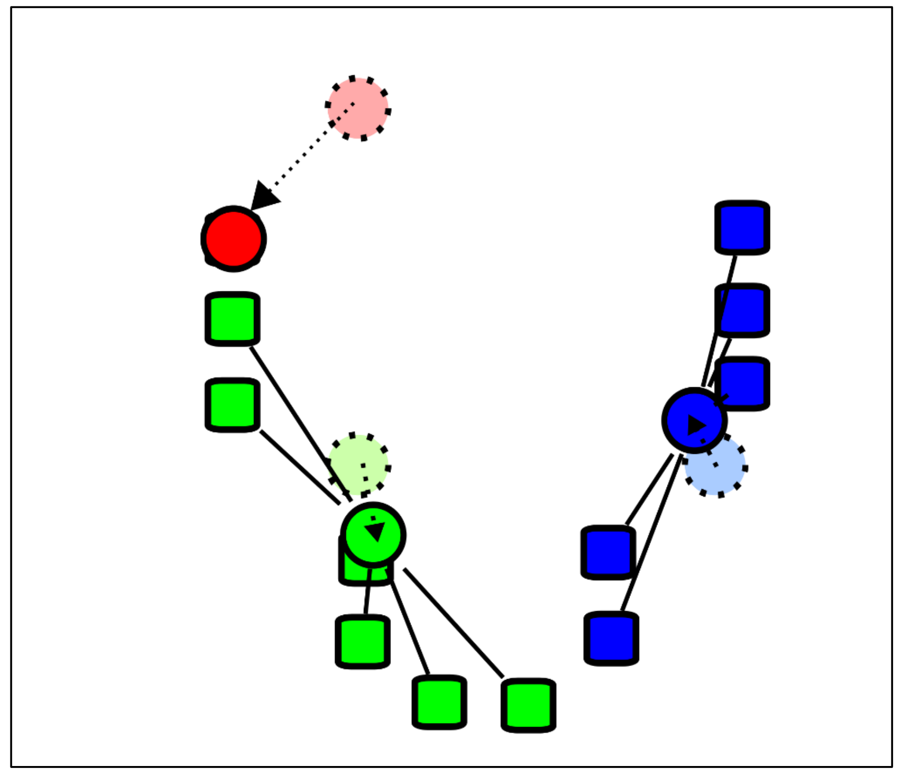
Step1: Random initialization of cluster centers

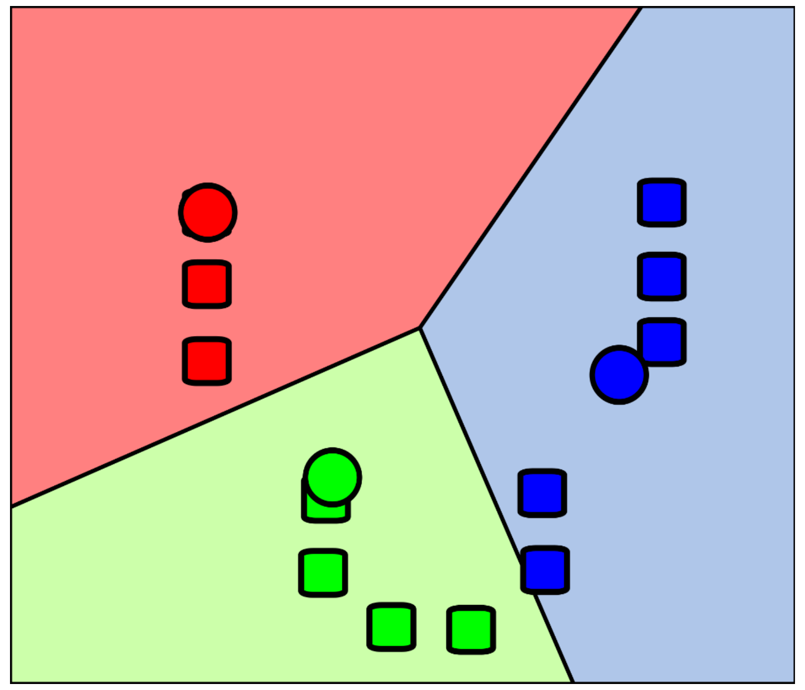


Step2: Tessellation of space into cluster regions



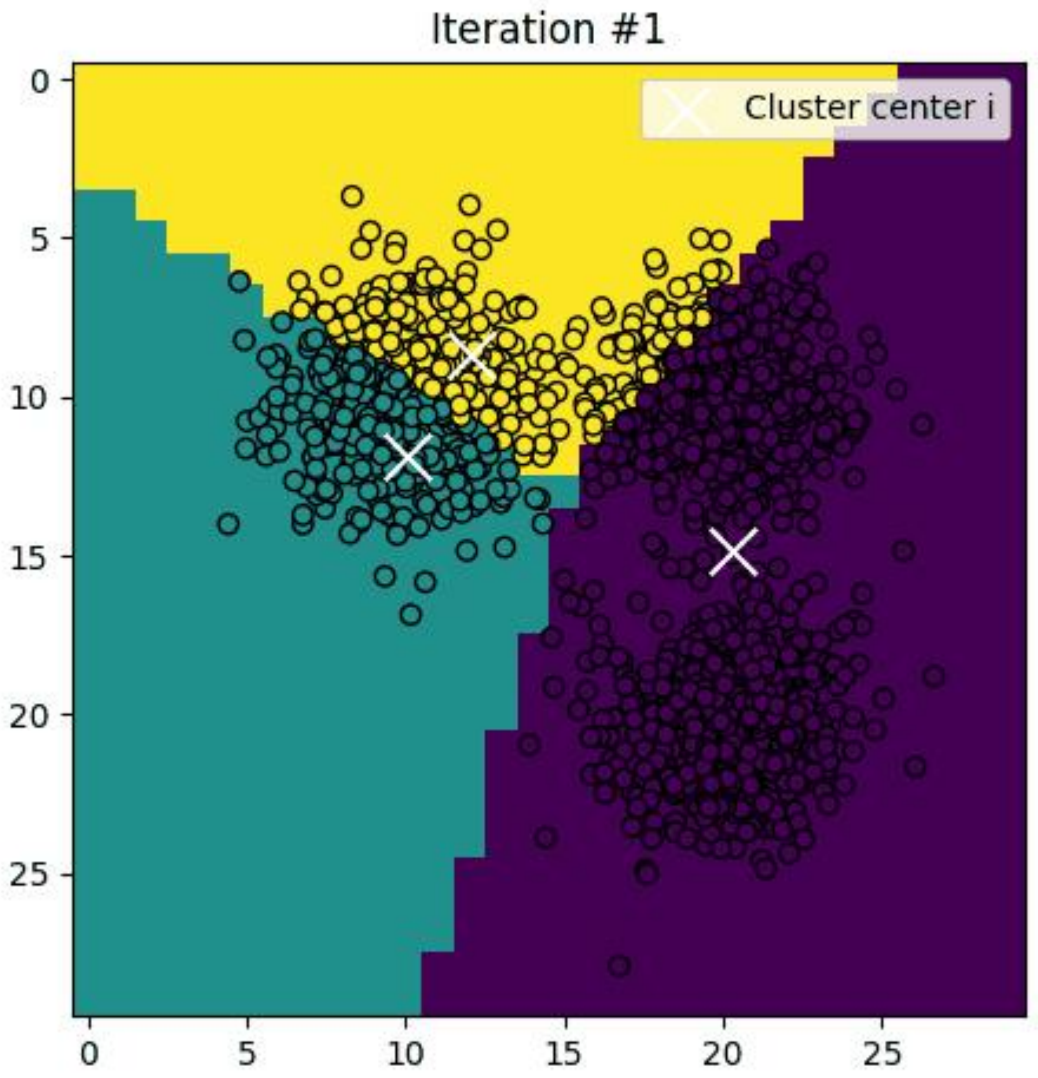
Step3: Replace cluster center with centrois





Step4: Repeat 2&3 until convergence

→ Fast convergence



In Python:

Import

```python
from sklearn import cluster
```

Create

```python
clusterer = cluster.KMeans(n_clusters=3)
clusterer.fit(X)
```

Predict

```python
predicted_class = clusterer.predict(X)
```

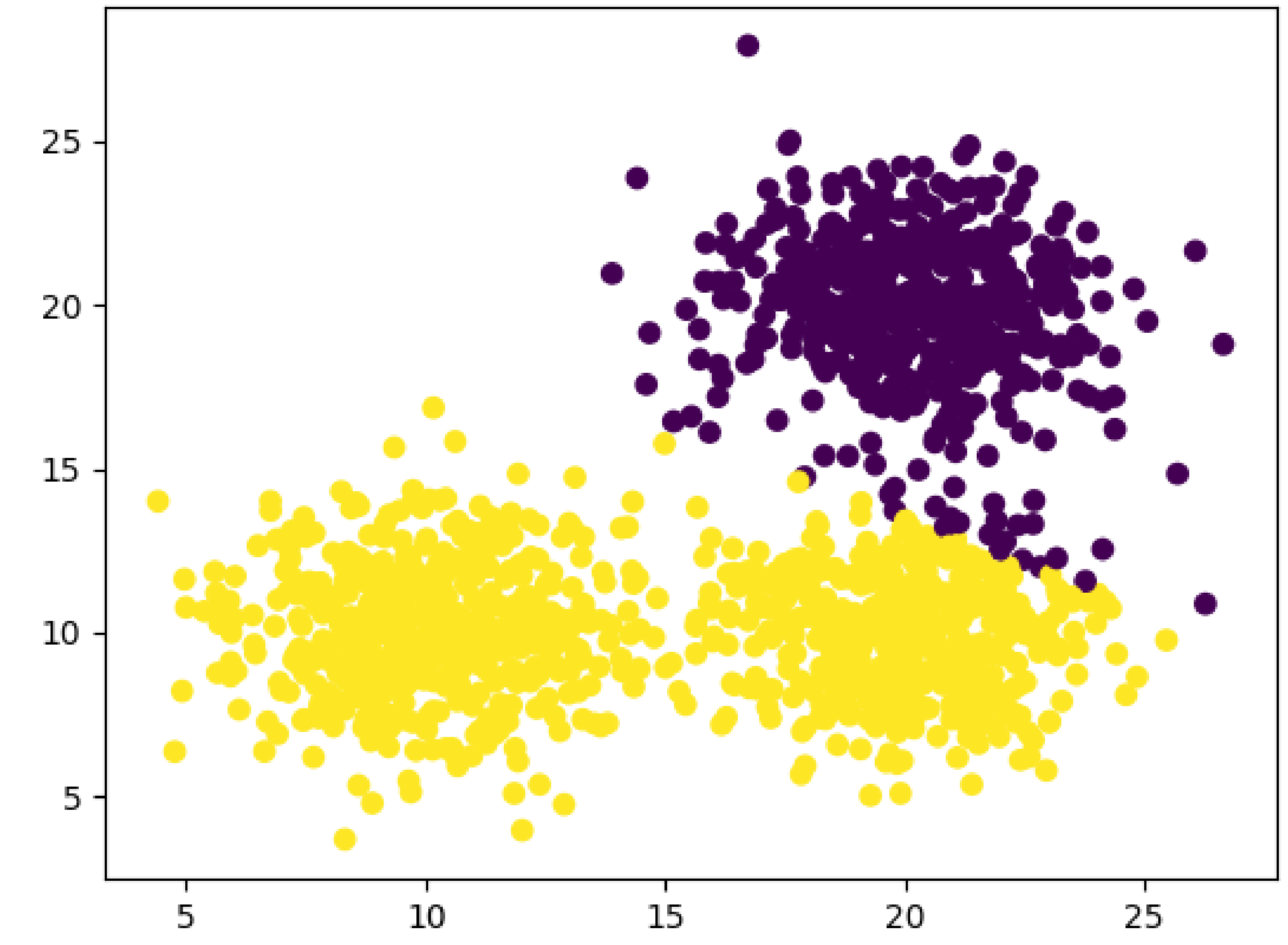
Source: https://de.wikipedia.org/wiki/K-Means-Algorithmus

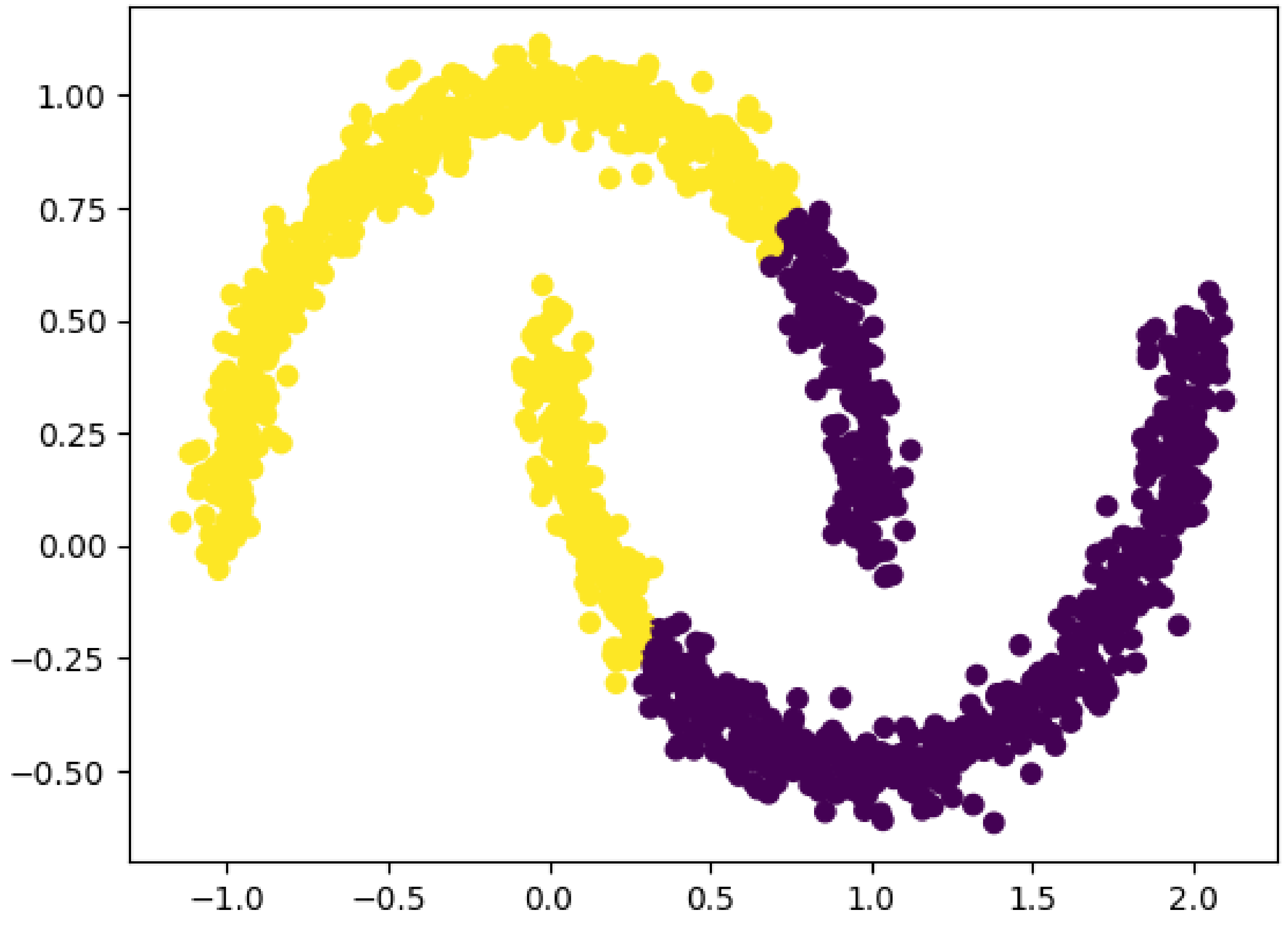
K-means clustering

Strength and weaknesses

- Number of clusters needs to be known
- Clusters can not capture more complex topologies
- Very fast
- Based on Euclidian metrics → every new point can be assigned to a cluster

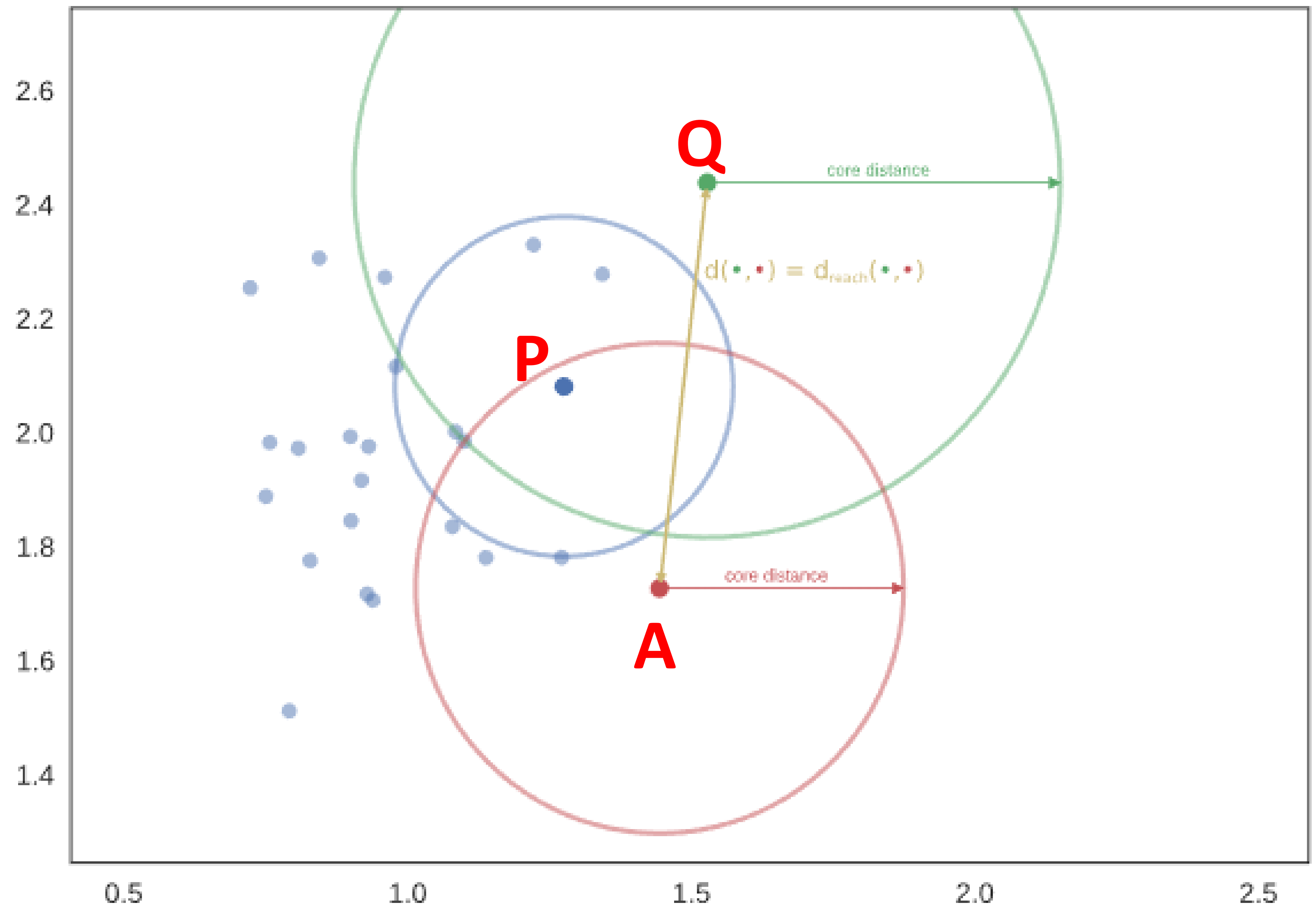
```
clusterer = cluster.KMeans(n_clusters=2)
clusterer.fit(X)
```





Hierarchical Density-Based Spatial Clustering of Applications with Noise

Strategy: Build neighborhood graph and identify strongly connected groups



Core distance: Distance to n-th nearest neighbor

Distance metric: Mutual reachability

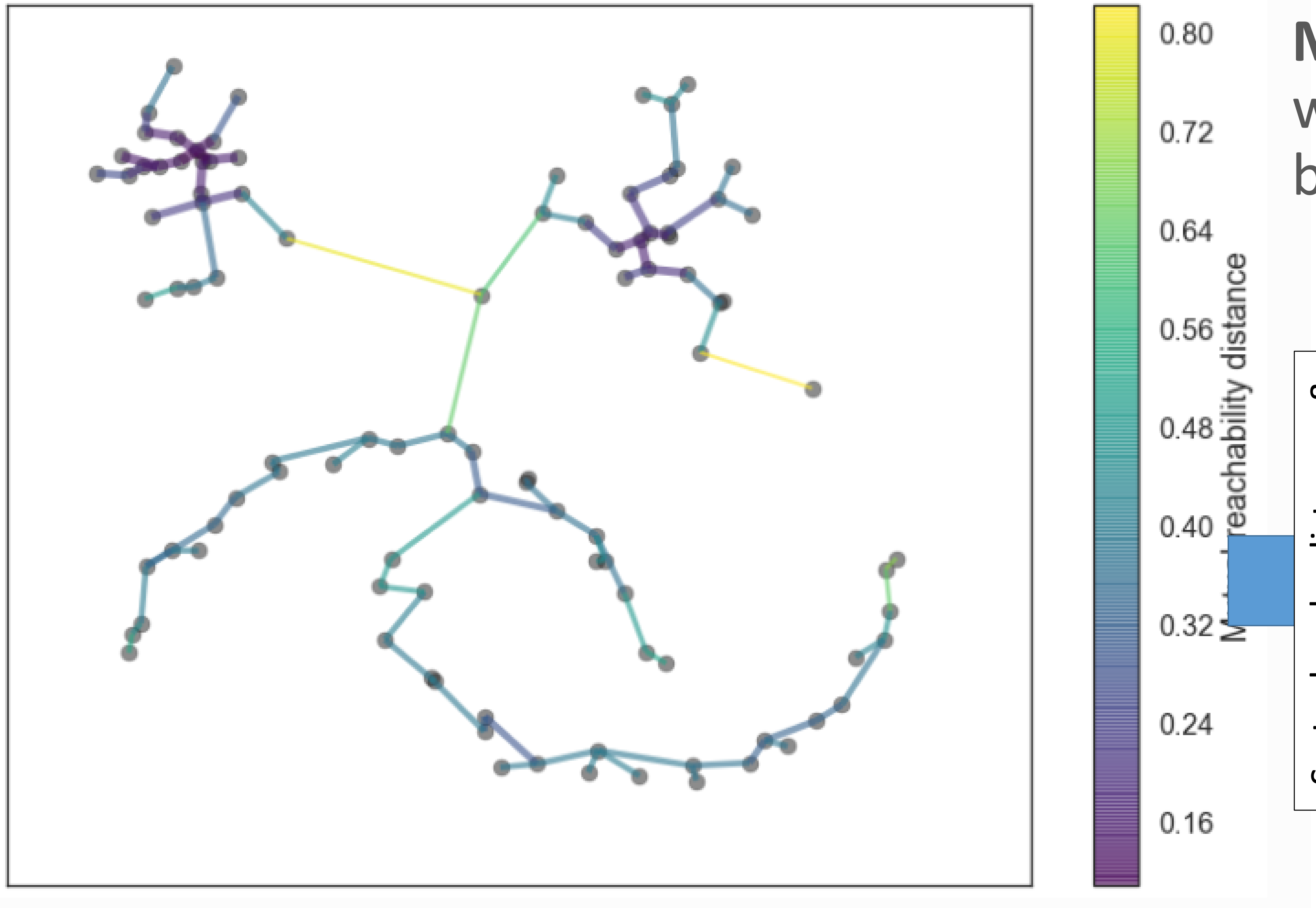
Core distance of Q > d(P, Q) ➔ d_{new}(P,Q) = core distance

Core distance of Q < d(A, Q) ➔ d_{new}(A,Q) = d(A,Q)

→ Isolated points are pushed further away from clusters

"To find clusters we want to find the islands of higher density amid a sea of sparser noise […] For practical purposes that means making 'sea' points more distant from each other and from the 'land'."

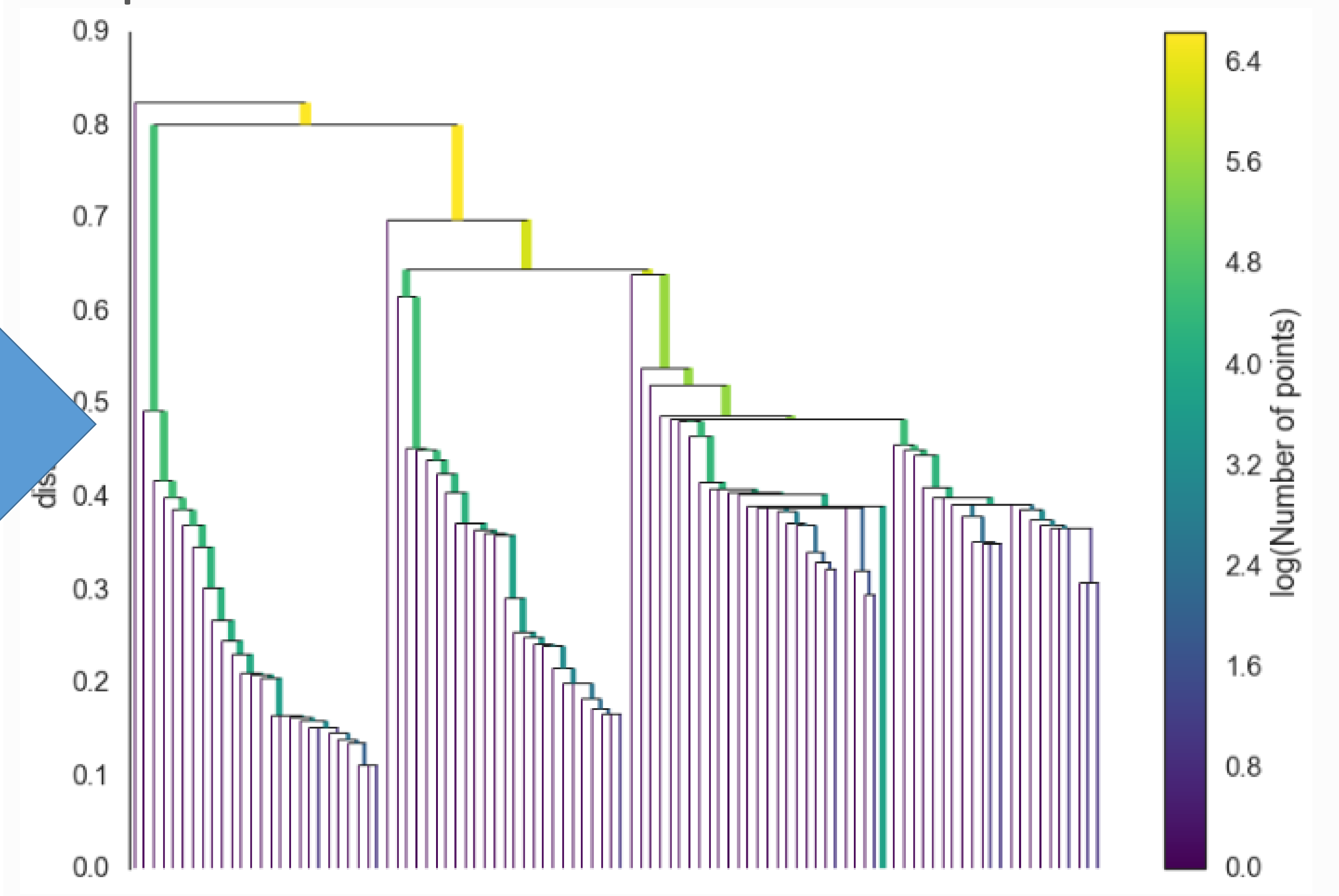
Strategy: Build neighborhood graph and identify strongly connected groups



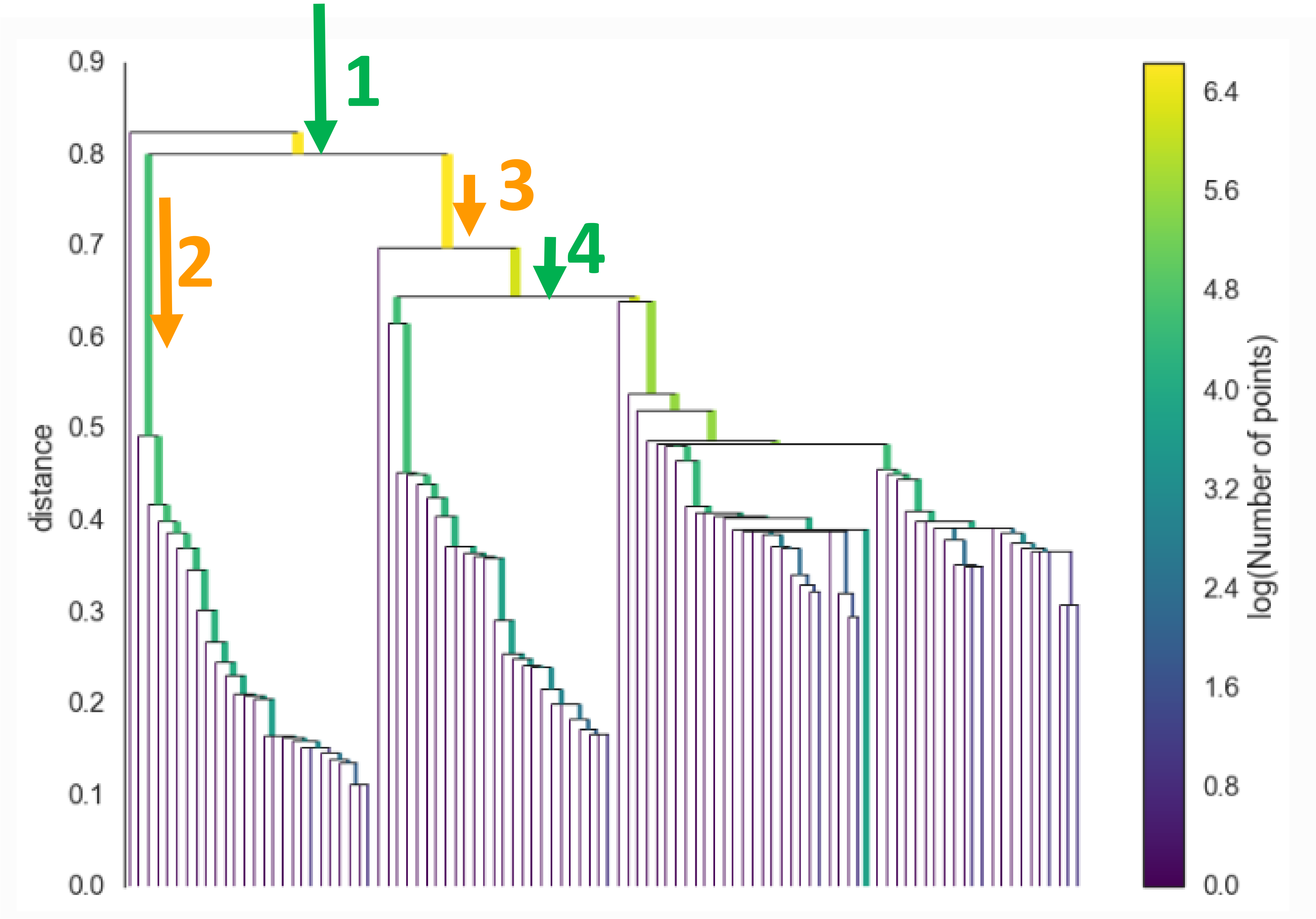
Source: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

Sort edges by distances & Merge close points

Minimum spanning tree: Neighborhood graph with minimal total distances (=mutual reachability) between points



Strategy: Build neighborhood graph and identify strongly connected groups



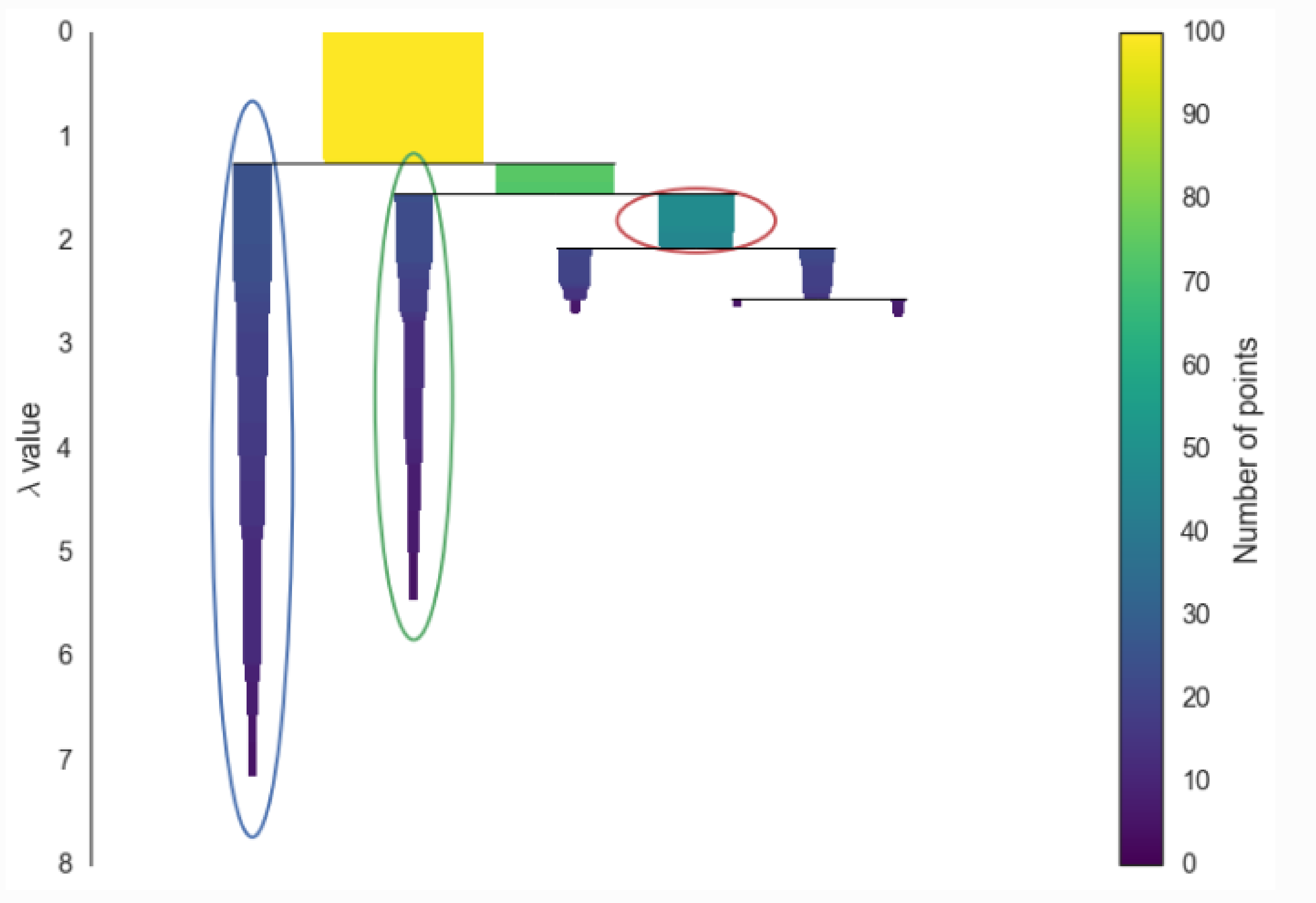
Source: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

Condense the tree: Traverse graph from top to bottom and decide whether a new cluster is formed at every crossroads

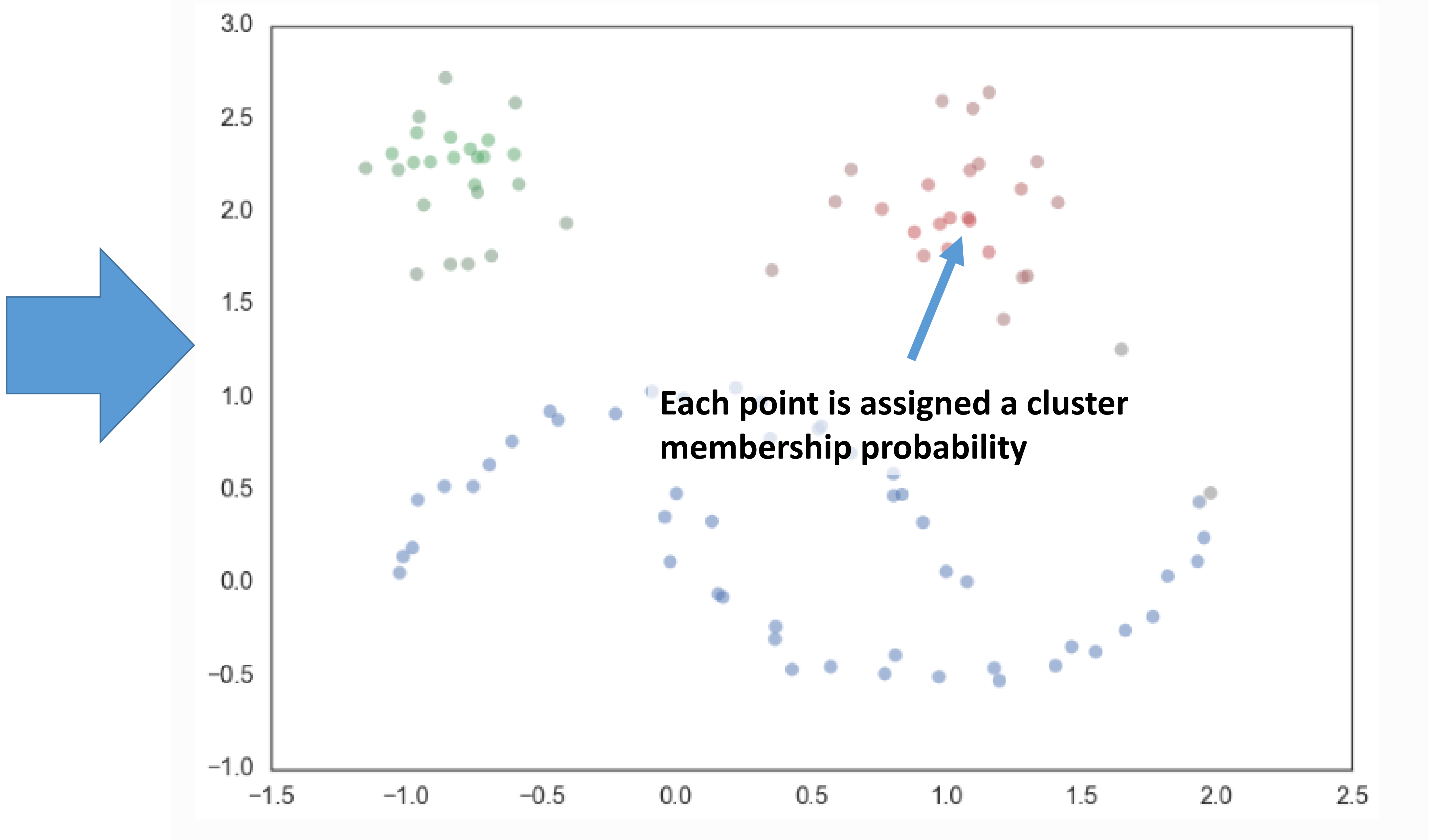
1. If points are split into clusters here – are both clusters larger than min_size? Yes
2. No – this part of the tree remains a single cluster
3. No – this part of the tree remains a single cluster
4. Yes – remaining points are split into new clusters here

Strategy: Build neighborhood graph and identify strongly connected groups

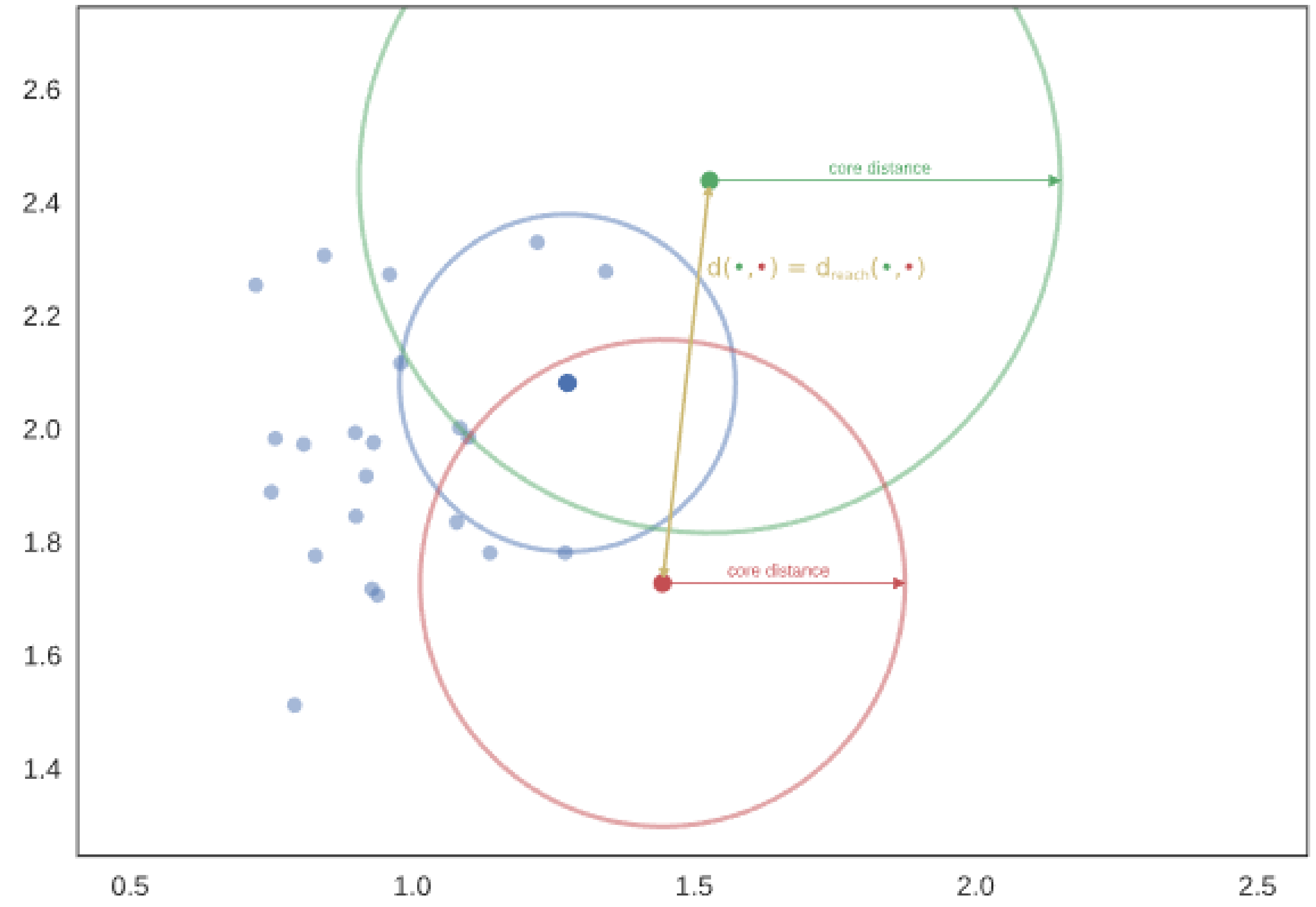
Extracting the clusters with ‘largest total ink area’ leads to the final selection of clusters

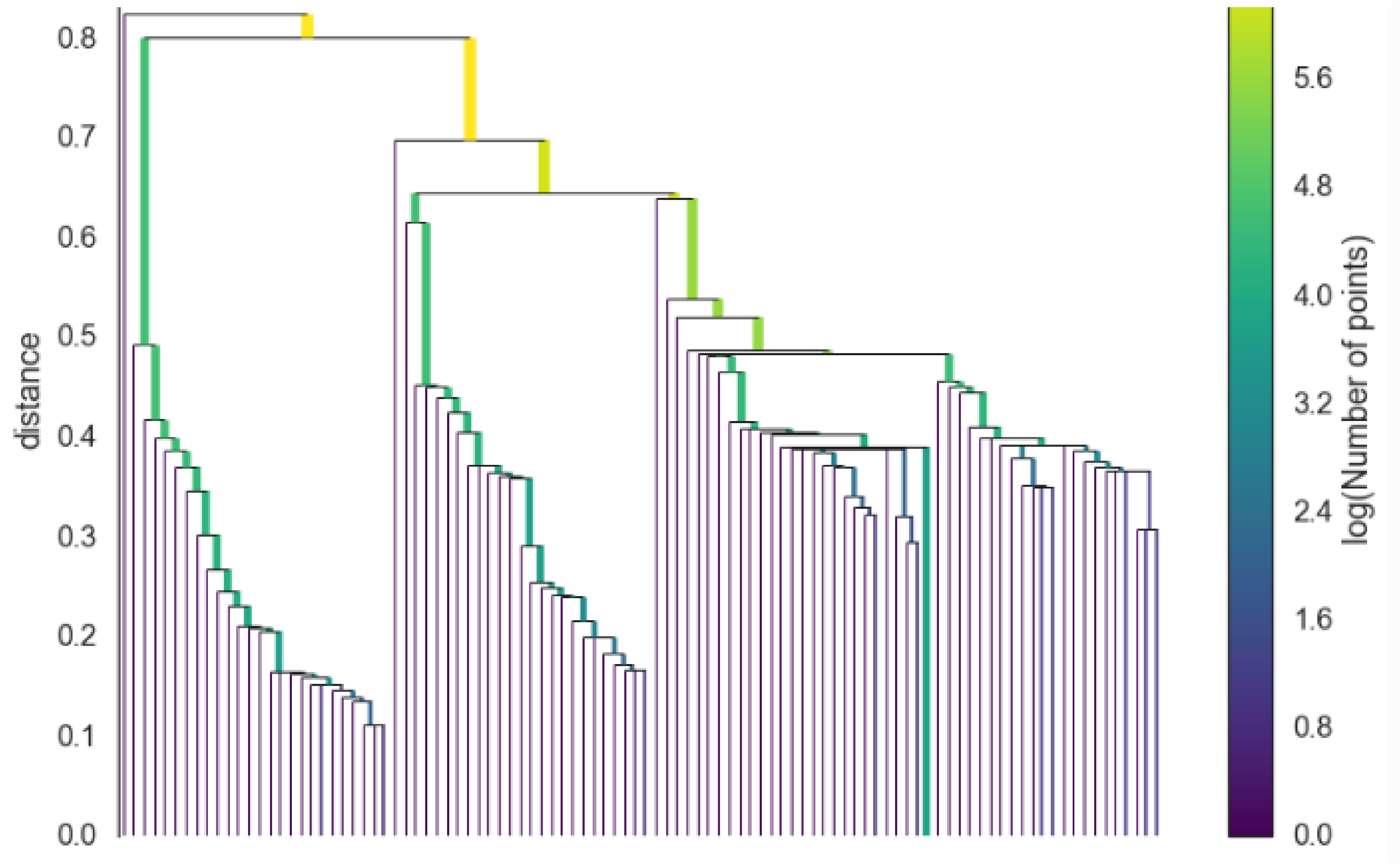


Source: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html



Variants of linkage-clustering



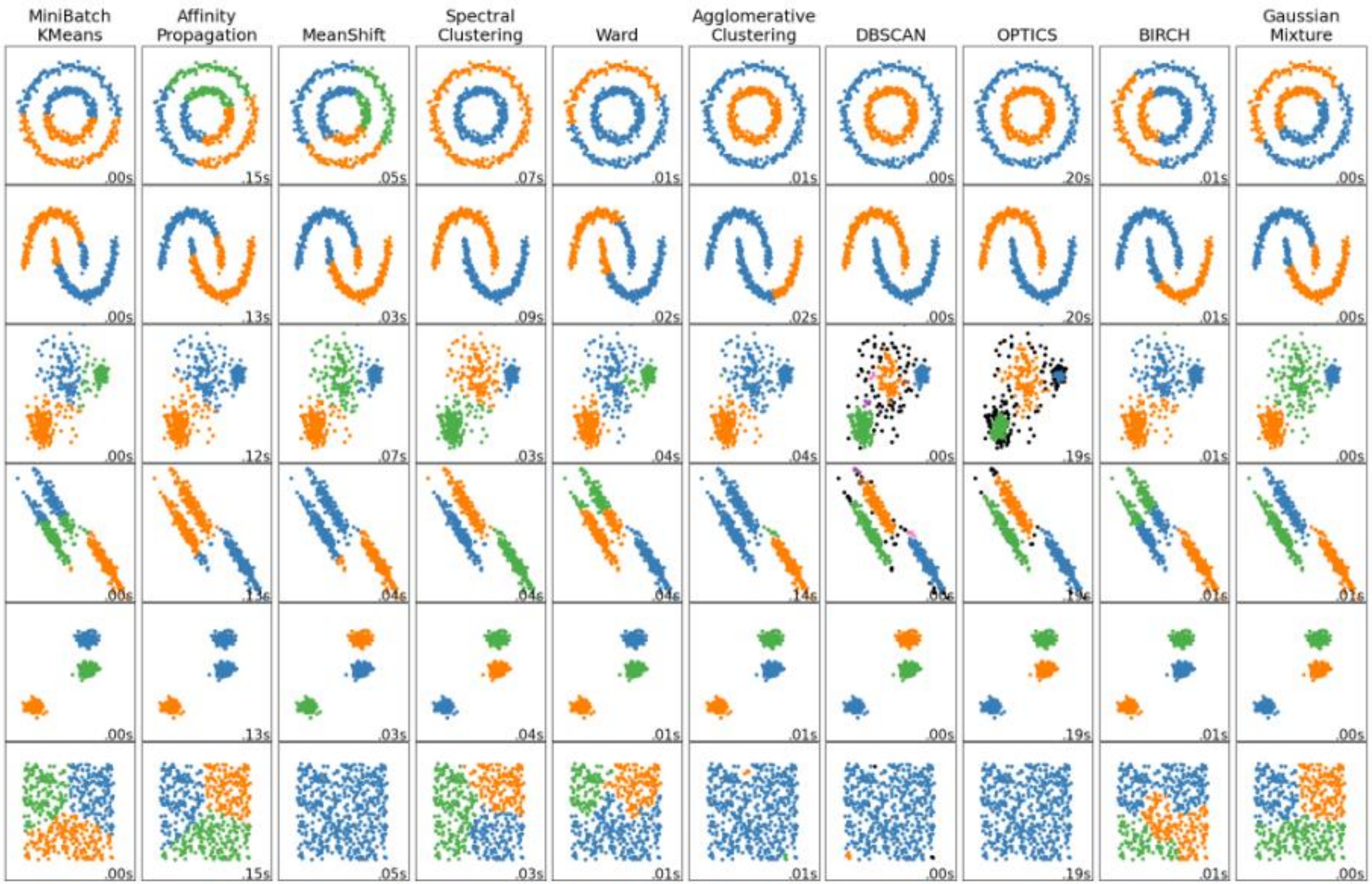


There are multiple ways to reconstruct the neighborhood graph and the clusters in the hierarchy schematic:

- Setting a maximum distance between two points to be considered neighbors → DBSCAN
  https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

- Aggregate points into clusters bottom-up → Agglomerative clustering
  https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

Other methods



A comparison of the clustering algorithms in scikit-learn

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