More on Classical clustering algorithms

Unsupervised Machine Learning

Alex Rodriguez.
alejandro.rodriguezgarcia@units.it
Universita degli Studi di Trieste
Building C5, office 3.18

REVIEW OF THE PREVIOUS LECTURE

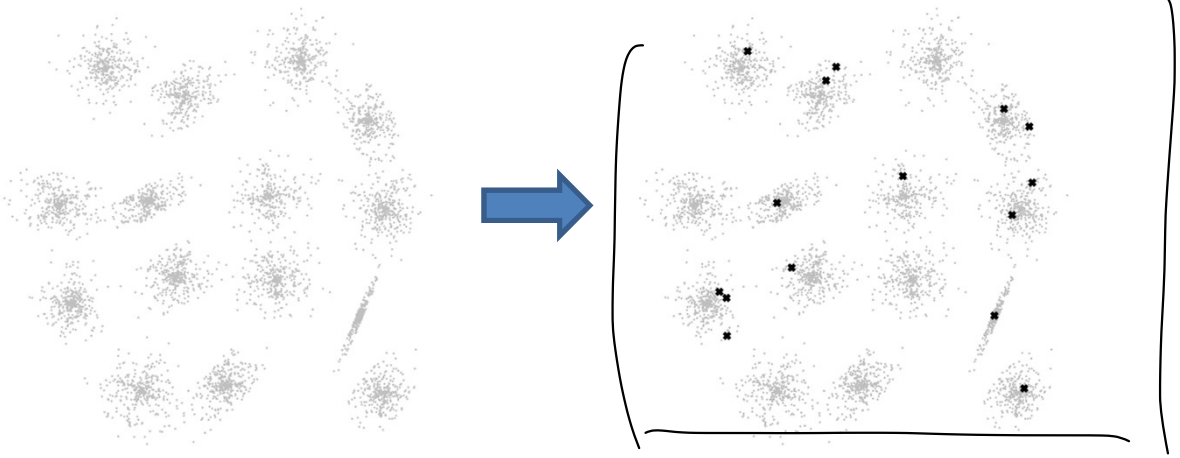
- We lack a mathematical definition of cluster.
- Real data sets are complex, often mixing types of data or with different number of features by data point.
- Often we need to perform feature selection (supervised) for simplifying the analysis. Variable ranking and subset selection.
- Working with distances (instead of features) is (the best/a) way to transform complex data sets in tractable ones.

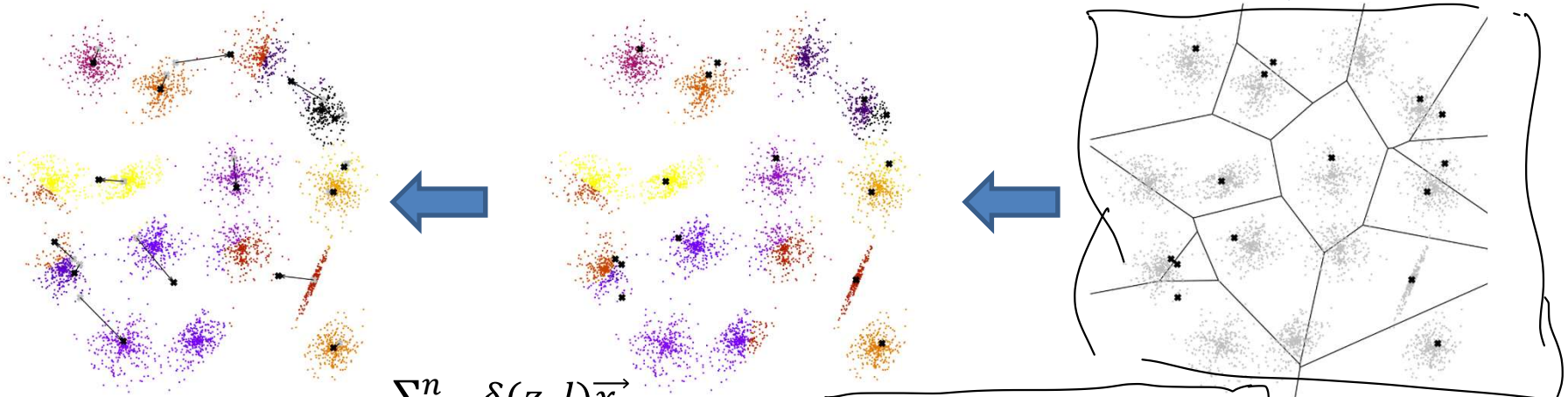
- Clustering is able to provide groups of data points similar each other (not to tell you the common characteristics of these groups).
- We can classify clustering algorithms according with their outcome:
  - **Flat clustering:** A single partition.
  - **Fuzzy clustering:** A single partition but with degrees of membership.
  - **Hierarchical clustering:** Tree like partition.

- K-means is a flat clustering algorithm.
- It aims to minimize the distance between the elements belonging to the same cluster while maximizing the distance between cluster centers.
- The global optimization of loss/objective function is NP hard. So we use local optimization methods

$$O(z) = \sum_{l=1}^{k} \sum_{i=1}^{n} \delta(z_i, l) \|\vec{x_i} - \vec{c_l}\|^2$$

Kmeans algo :-



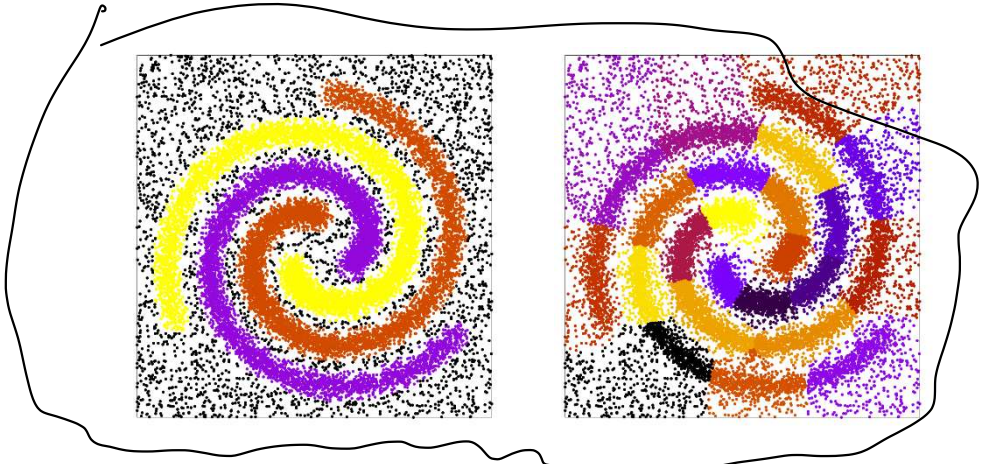


$$\vec{c_l} = \frac{\sum_{i=1}^{n} \delta(z_i, l)\vec{x_i}}{\sum_{i=1}^{n} \delta(z_i, l)}$$

$$z_i = argmin_l(d(\vec{c_l}, \vec{x_i}))$$

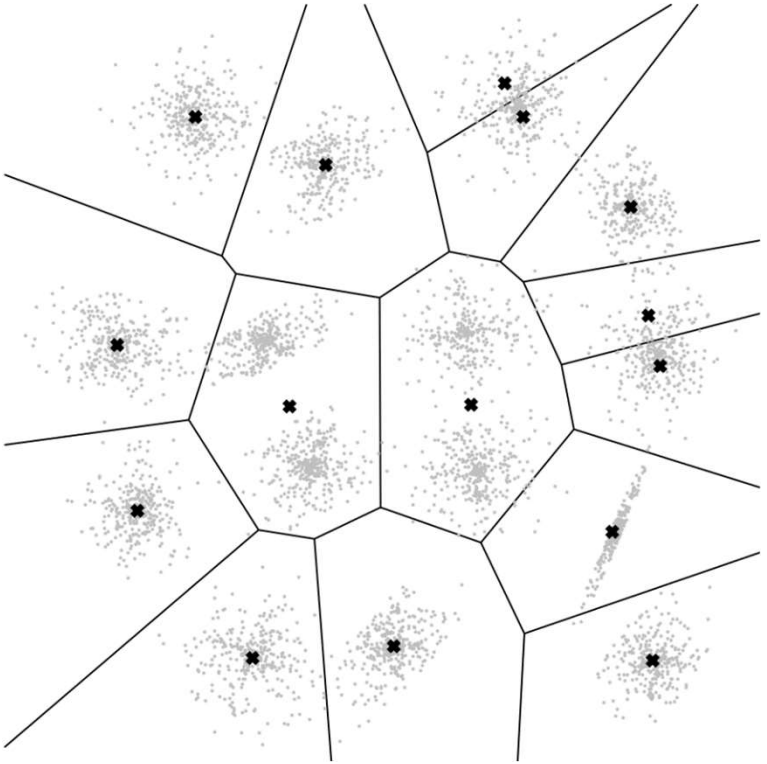
Kmeans performs a Voronoi partition of the space

Why *k*-means?



Expected result K-means result

- **K-means provides an optimal Voronoi partition of the data.**
- **Many other methods are somehow related to k-means.**



Kernel k-means

Observation: The k-means algorithm only needs to compute the distances from the centers of the clusters to the data points.

→ to prevent the problem that kmeans allows separation only for linear partitions of the data

Kernel k-means

Observation: The k-means algorithm only needs to compute the distances from the centers of the clusters to the data points.

Let's use the kernel trick! (introduce nonlinear features)

- Project the data set into a feature space by means of a nonlinear mapping $\overrightarrow{x_i} \rightarrow \overrightarrow{\phi_i}$.
- Distances in the feature space can be easily computed with the kernel trick:

Eq. of dist → $d_{ij}^2 = \|\overrightarrow{\phi_i} - \overrightarrow{\phi_j}\|^2 = K^{ii} + K^{jj} - 2K^{ij}$

- The loss function and the centroids are analogous to the ones in standard k-means, but in the feature space.

$$O(z) = \sum_{l=1}^{k} \sum_{i=1}^{n} \delta(z_i, l) \|\overrightarrow{\phi_i} - \overrightarrow{v_l}\|^2$$

$$\overrightarrow{v_l} = \frac{\sum_{i=1}^{n} \delta(z_i, l) \overrightarrow{\phi_i}}{\sum_{i=1}^{n} \delta(z_i, l)}$$

$d_{ij} = (\vec{\phi}_i - \vec{\phi}_j)(\vec{\phi}_i - \vec{\phi}_j) = \vec{\phi}_i \cdot \vec{\phi}_i + \vec{\phi}_j \cdot \vec{\phi}_j - 2\,\vec{\phi}_i \vec{\phi}_j$

K_{ii} K_{jj} K_{ij}

Kernel k-means

- We can compute the Gramm matrix as in the case of kernel PCA.

$$G^{ij} = -\frac{1}{2}\left(K^{ij} - \frac{1}{n}\sum_{i=1}^{n} K^{ij} - \frac{1}{n}\sum_{j=1}^{n} K^{ij} + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{i=1}^{n} K^{ij} \right)$$

- How to compute the distance from each element to a cluster center?
  - We cannot compute directly $\overrightarrow{v_l} = \frac{\sum_{i=1}^{n} \delta(z_i, l)\overrightarrow{\phi_i}}{\sum_{i=1}^{n} \delta(z_i, l)}$ → (centroids) (we don't know how our vectors are projected in feature space)
  - But we can compute the value of $\left\|\overrightarrow{\phi_i} - \overrightarrow{v_l}\right\|^2$ (next page)

$$d_{ik}^2 = \|\vec{\phi}_i - \vec{v}_k\|^2 = \vec{\phi}_i \vec{\phi}_i - 2\vec{\phi}_i \vec{v}_k + \vec{v}_k \vec{v}_k$$

$$\vec{v}_k = \frac{\sum \delta(z_i, k)\, \vec{\phi}_i}{\sum \delta(z_i, k)} \qquad [v_i \to \text{centroid pas } \vec{v}]$$

$$d_{ik}^2 = G^{ii} - 2\frac{\phi_i \sum \delta(z_j, k)\phi_j}{\sum \delta(z_j, k)} + \frac{\sum_j^n \delta(z_i, k)\phi_j}{\sum_{k=1}^n \delta(z_k, k)\phi_k}$$

$$= G^{ii} - 2\frac{\sum \delta(z_i, k)\, G^{ij}}{\sum \delta(z_i, k)} + \frac{\sum_j \sum_k \delta(z_j, k)\delta(z_k, k) G^{jk}}{\left[\sum_j \delta(z_j, k)\right]^2}$$

Kernel k-means

$$d_{il}{}^2 = \left\|\overrightarrow{\phi_i} - \overrightarrow{v_l}\right\|^2 = \left(\overrightarrow{\phi_i} - \overrightarrow{v_l}\right) \cdot \left(\overrightarrow{\phi_i} - \overrightarrow{v_l}\right) = \overrightarrow{\phi_i} \cdot \overrightarrow{\phi_i} - 2\overrightarrow{\phi_i} \cdot \overrightarrow{v_l} + \overrightarrow{v_l} \cdot \overrightarrow{v_l}$$
$$= G^{ii} - 2\frac{\sum_{j=1}^{n} \delta\left(z_{i,} l\right) G^{ij}}{\sum_{j=1}^{n} \delta\left(z_{j,} l\right)} + \frac{\sum_{j=1}^{n} \sum_{k=1}^{n} \delta(z_k, l) \delta\left(z_{j,} l\right) G^{kj}}{\left(\sum_{j=1}^{n} \delta\left(z_{j,} l\right)\right)^2}$$

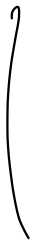
The algorithm:

1. Compute the Gramm matrix.
2. Randomly pick k centers.
3. Iterate the equation $z_i = argmin_l\left(d_{il}{}^2\right)$ until convergence

compute distances from each data point to centers, so we don't need to recompute the centroids :: we work with distances & not in feature space.

Now that we have dist. from each element to centroid, we modify k means.

Fuzzy c-means: A fuzzy clustering algorithm



Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, *10*(2), 191-203.

Fuzzy c-means

- It can be considered a version of k-means with a soft assignation criteria.

$$\overrightarrow{Cl}(i) = (u_1, u_2, \dots u_l \dots, u_k)$$

must be normalized = 1, ∴ it's like a probability,

$$\left\{ \sum_{l=1}^{k} u_l = 1 \right.$$

$\in$ (0, 0, ..., 1, 0, ..0)

- Therefore, the objective function and the optimization algorithm must be adapted to fulfill these conditions.

→ "degree of membership"

Fuzzy c-means algorithm

- Objective function:

$$O(\mathbb{U}) = \sum_{l=1}^{k} \sum_{i=1}^{n} (u_{il})^m \|\overrightarrow{x_i} - \overrightarrow{c_l}\|^2$$

- $\mathbb{U}$ is a $n \times k$ matrix with the membership of the n elements in the k clusters.
- m is the fuzzification parameter. Usually m=2.
- $\overrightarrow{c_l}$ is the vector with the coordinates of the l cluster center. As in the case of k-means, it is the average value of the cluster coordinates. $\overrightarrow{c_l} = \frac{\sum_{i=1}^{n} (u_{il})^m \overrightarrow{x_i}}{\sum_{i=1}^{n} (u_{il})^m}$

Fuzzy c-means algorithm

- Random initialization of $\mathbb{U}$
- Iterative optimization:
  - Given $\mathbb{U}$ compute the centers:

$$\overrightarrow{c_l} = \frac{\sum_{i=1}^{n} (u_{il})^m \overrightarrow{x_i}}{\sum_{i=1}^{n} (u_{il})^m}$$

  - Given the centers, update $\mathbb{U}$:

$$u_{ij} = \frac{1}{\sum_{l=1}^{k} \left(\frac{\|\overrightarrow{x_i} - \overrightarrow{c_j}\|}{\|\overrightarrow{x_i} - \overrightarrow{c_l}\|} \right)^{2/m-1}}$$

- Repeat until the change in $\mathbb{U}$ is smaller than a given threshold

Fuzzy c-means algorithm

- Similar problems to k-means:
  - Initialization
  - Number of clusters
  - Sensitive to outliers
  - Spherical clusters

people don't use it so much, ∴ it is NOT a real probabilistic method.

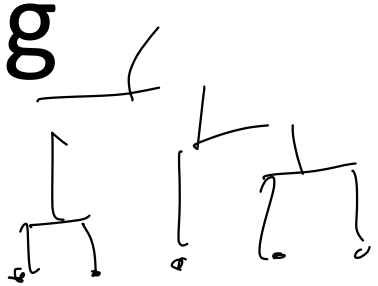
(only works with spherical clusters !!

assignation to a cluster depends on distances like in k-means)

Hierarchical clustering algorithms

Based on slides by Evimaria Terzi
(http://www.cs.bu.edu/~evimaria/)

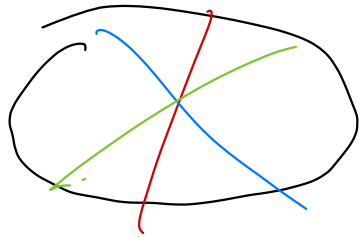
Hierarchical Clustering



- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or **k** clusters) are left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are **k** clusters)

- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

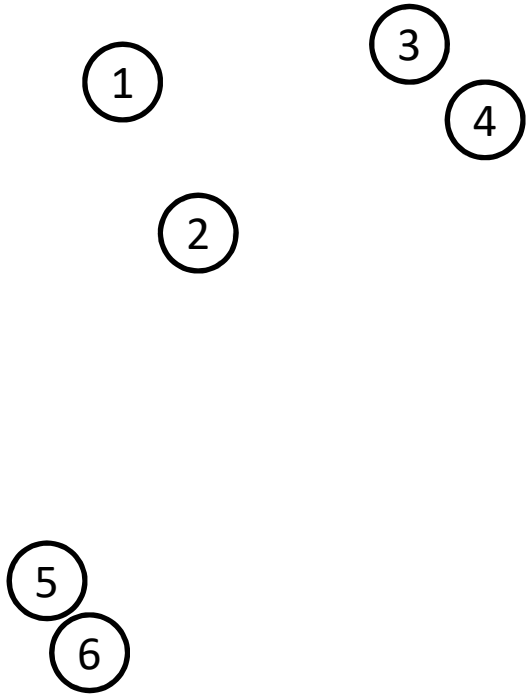
Agglomerative clustering algorithm



- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. **Until** only a single cluster remains

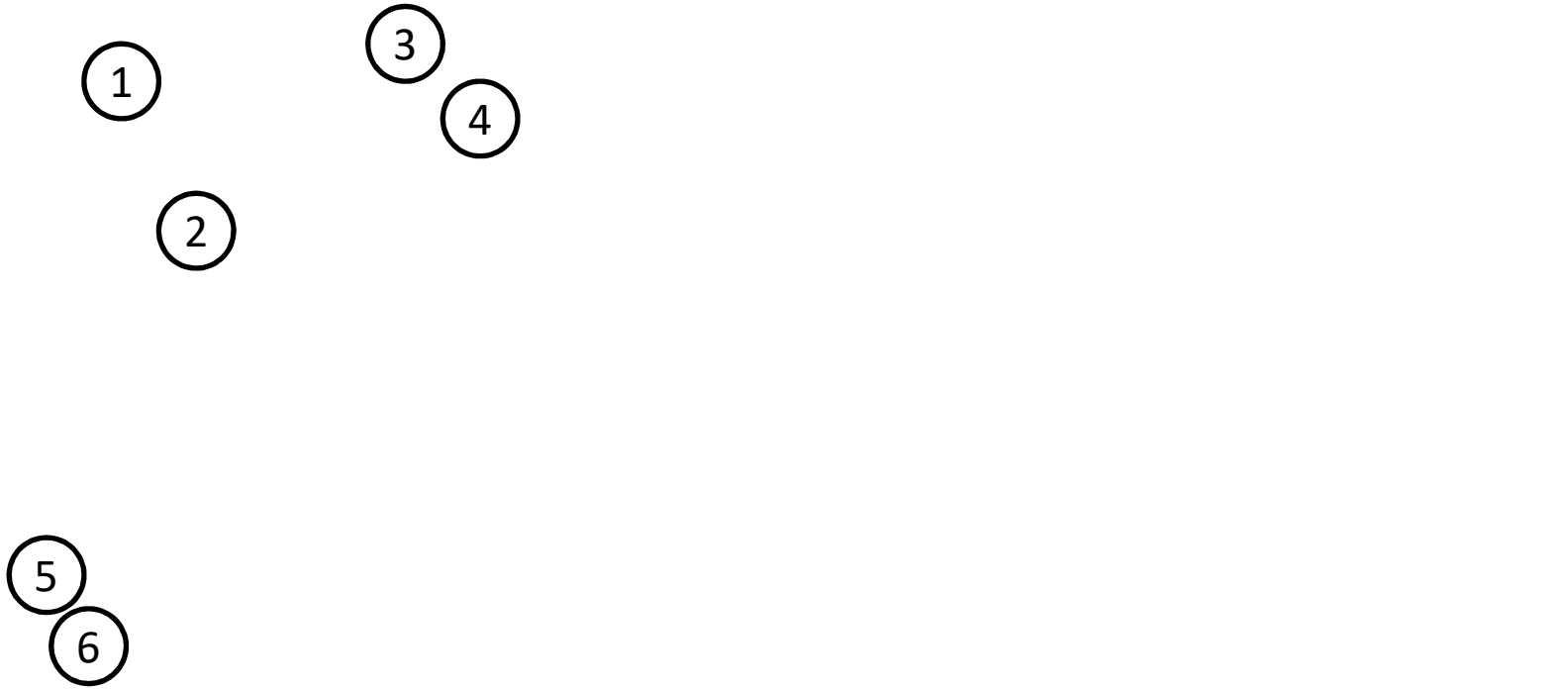
→ they are preferred : agglomerative is more computationally efficient than divisive.

Hierarchical clustering at glance



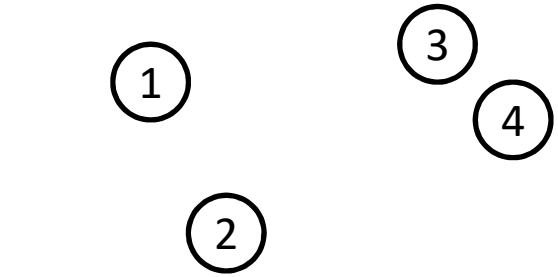
At each step: Pick the minimum distance and merge these elements

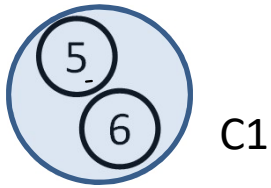
Hierarchical clustering at glance



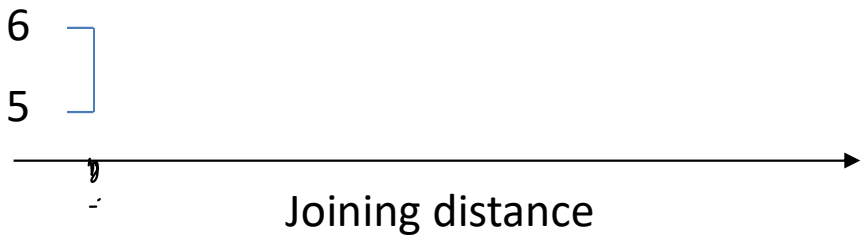
At each step: Pick the minimum distance and merge these elements

Hierarchical clustering at glance



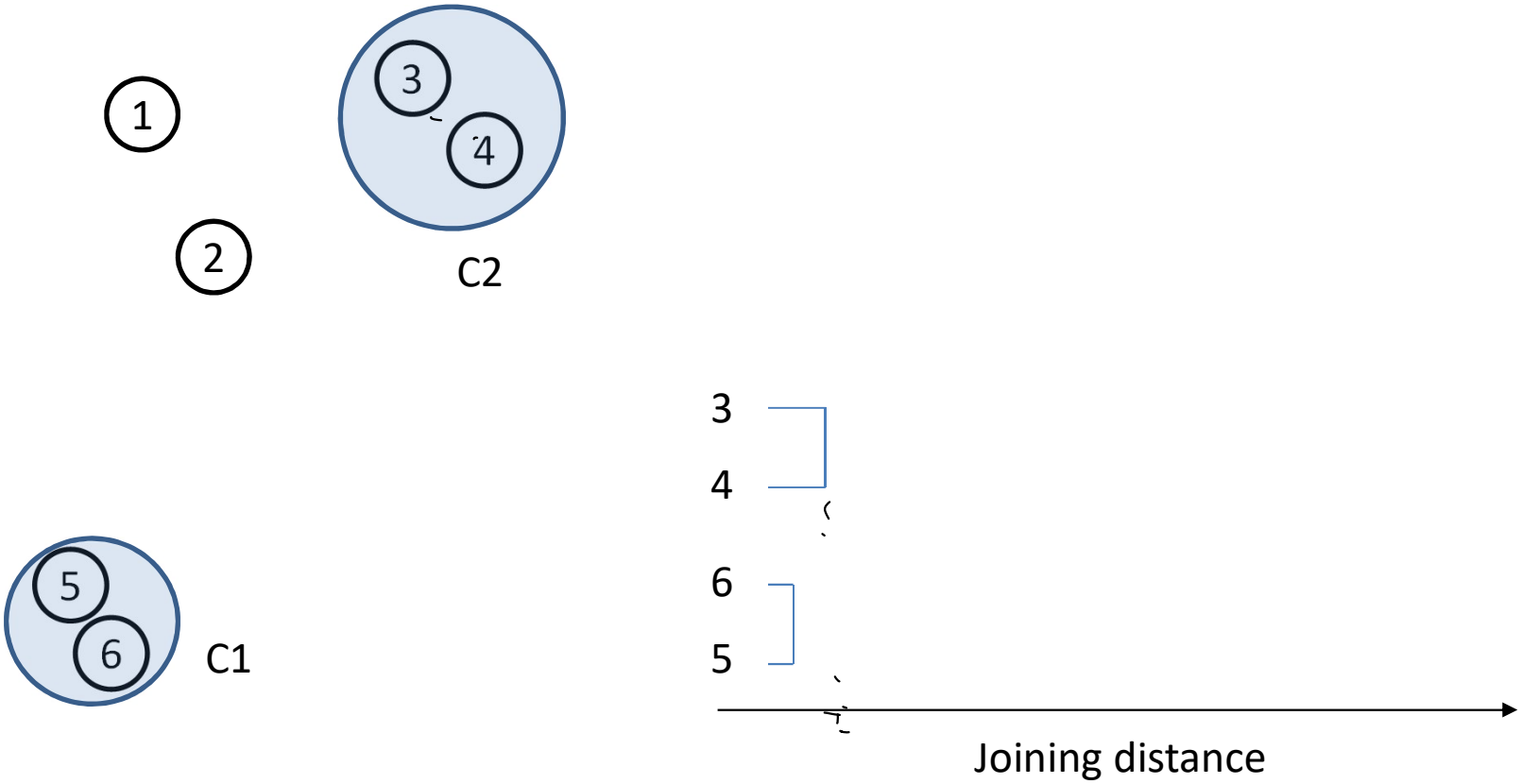


C1



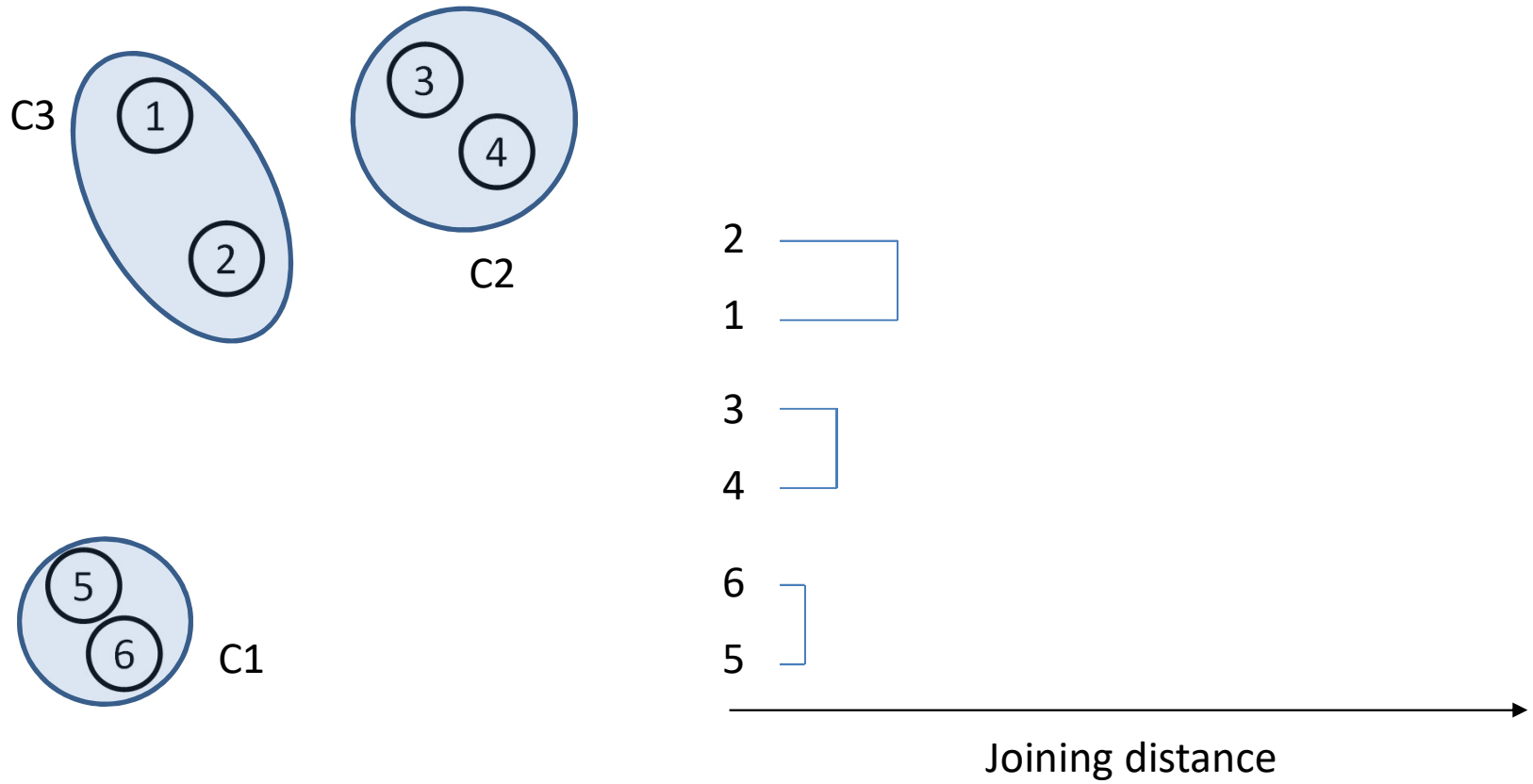
At each step: Pick the minimum distance and merge these elements

Hierarchical clustering at glance



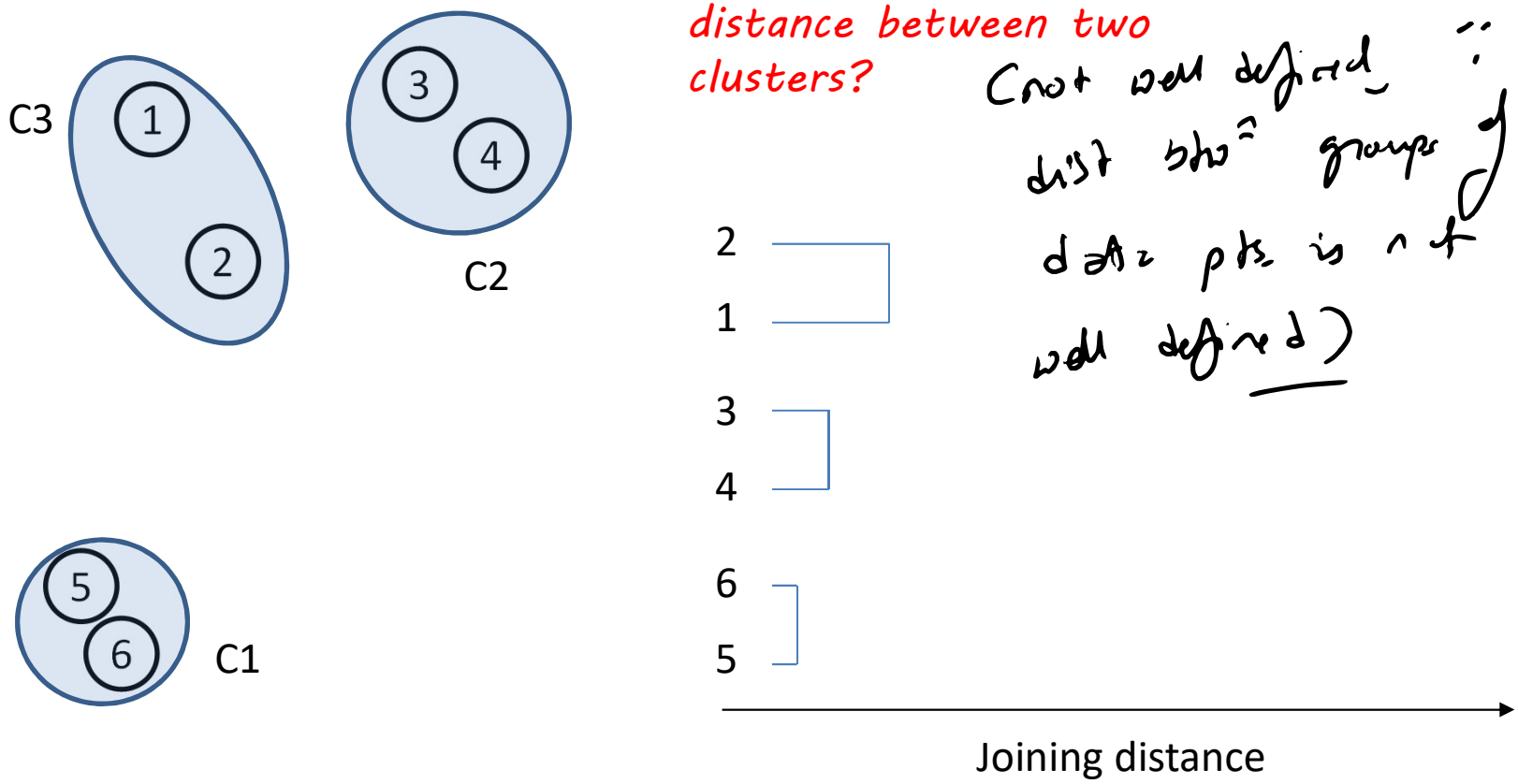
At each step: Pick the minimum distance and merge these elements

Hierarchical clustering at glance



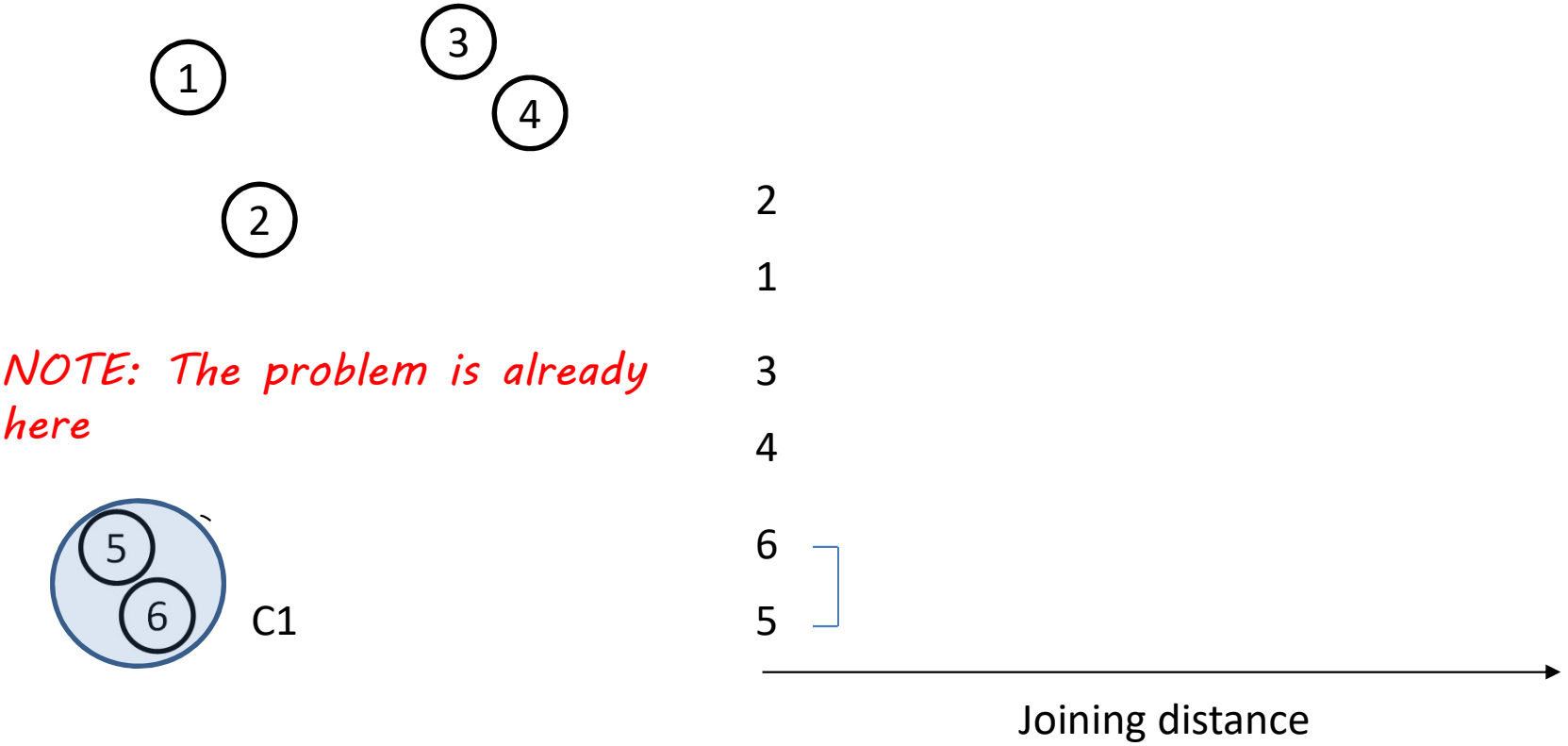
At each step: Pick the minimum distance and merge these elements

Hierarchical clustering at glance



At each step: Pick the minimum distance and merge these elements

Hierarchical clustering at glance



At each step: Pick the minimum distance and merge these elements

Agglomerative clustering algorithm

- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the distance between two clusters
  - Different definitions of the distance between clusters lead to different algorithms

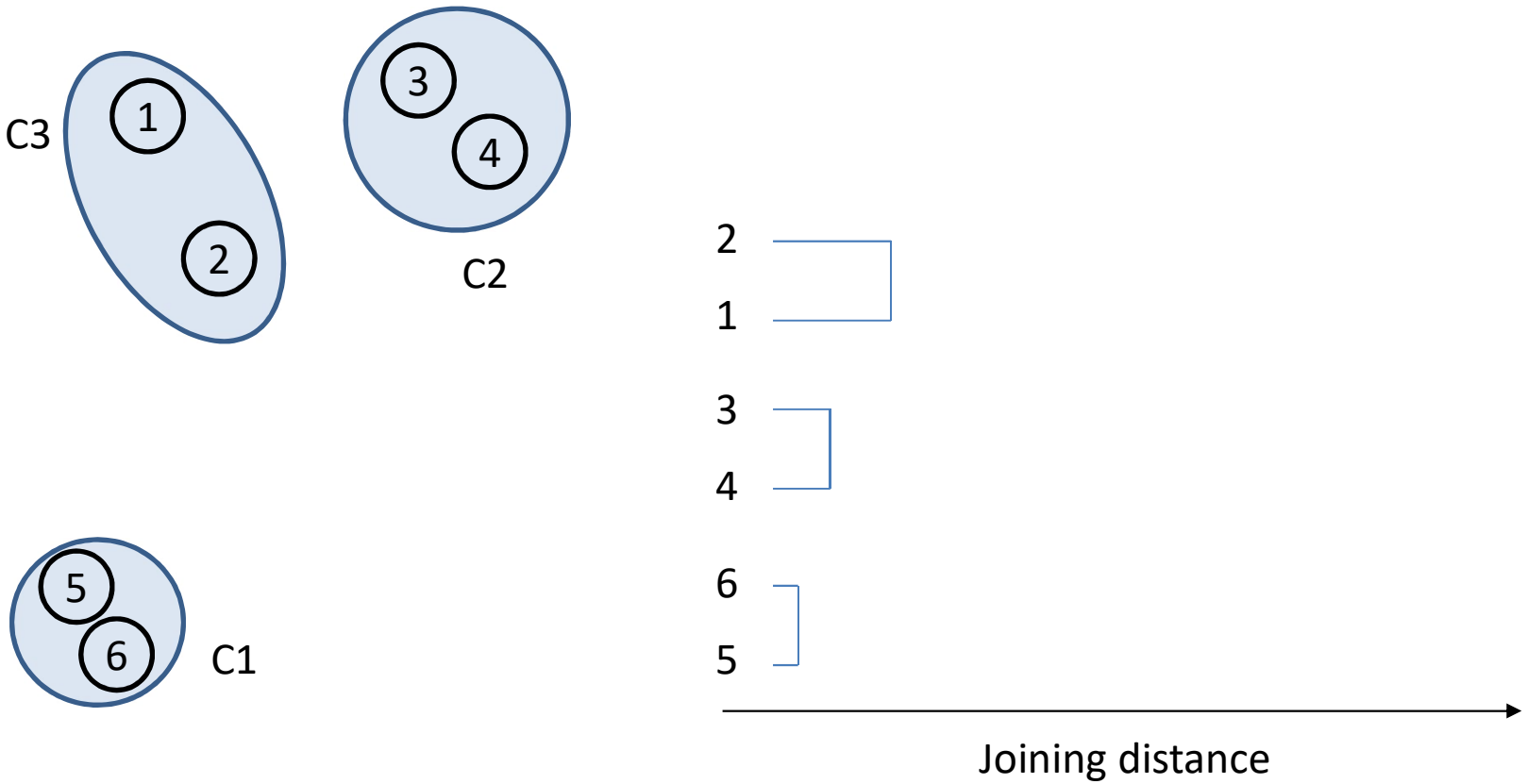
Single-link distance

- **Single-link distance** between clusters C_i and C_j is the ***minimum distance*** between any object in C_i and any object in C_j

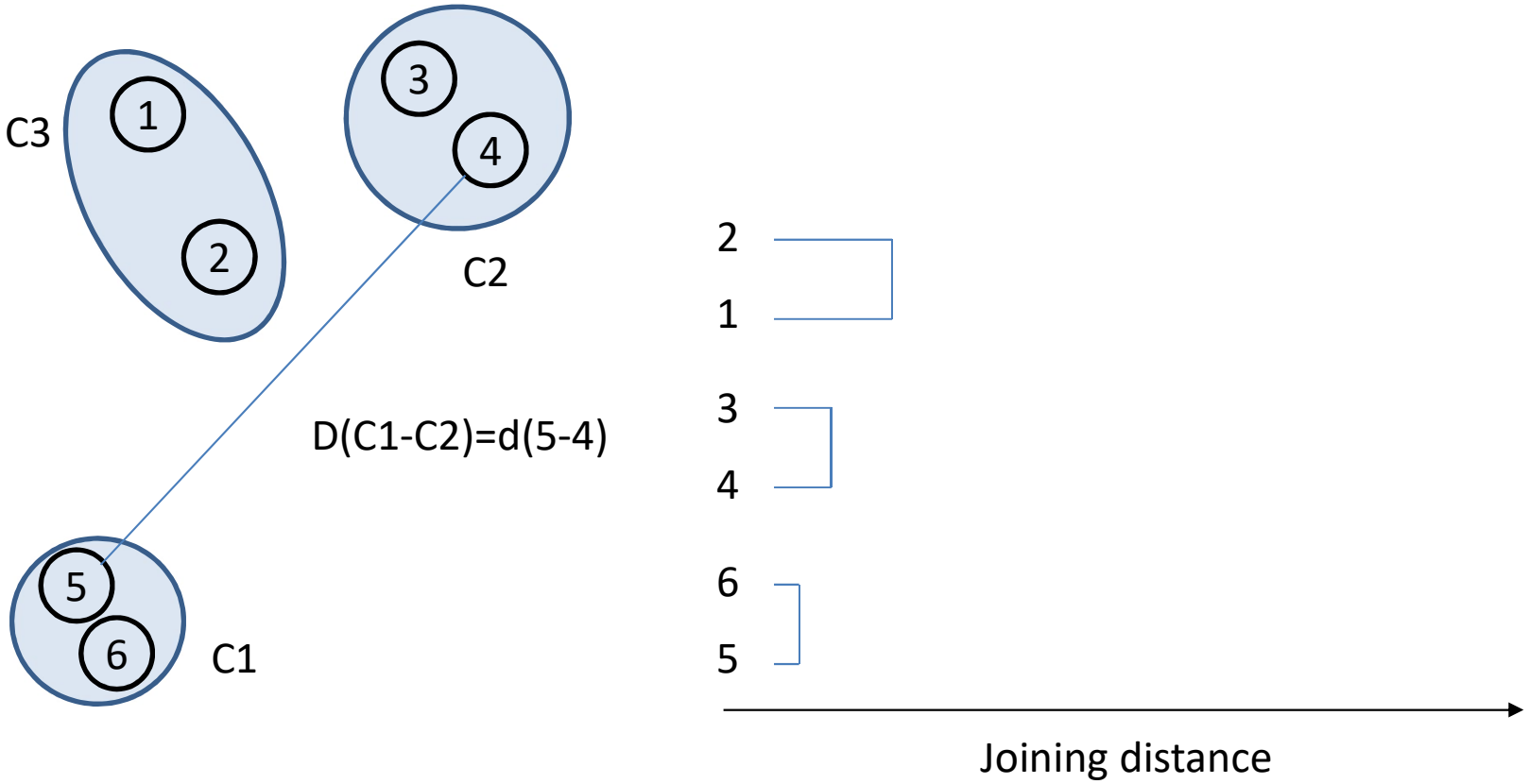
- The distance is **defined by the two most similar (i.e nearest) objects**

$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x, y) \middle| x \in C_i, y \in C_j \right\}$$

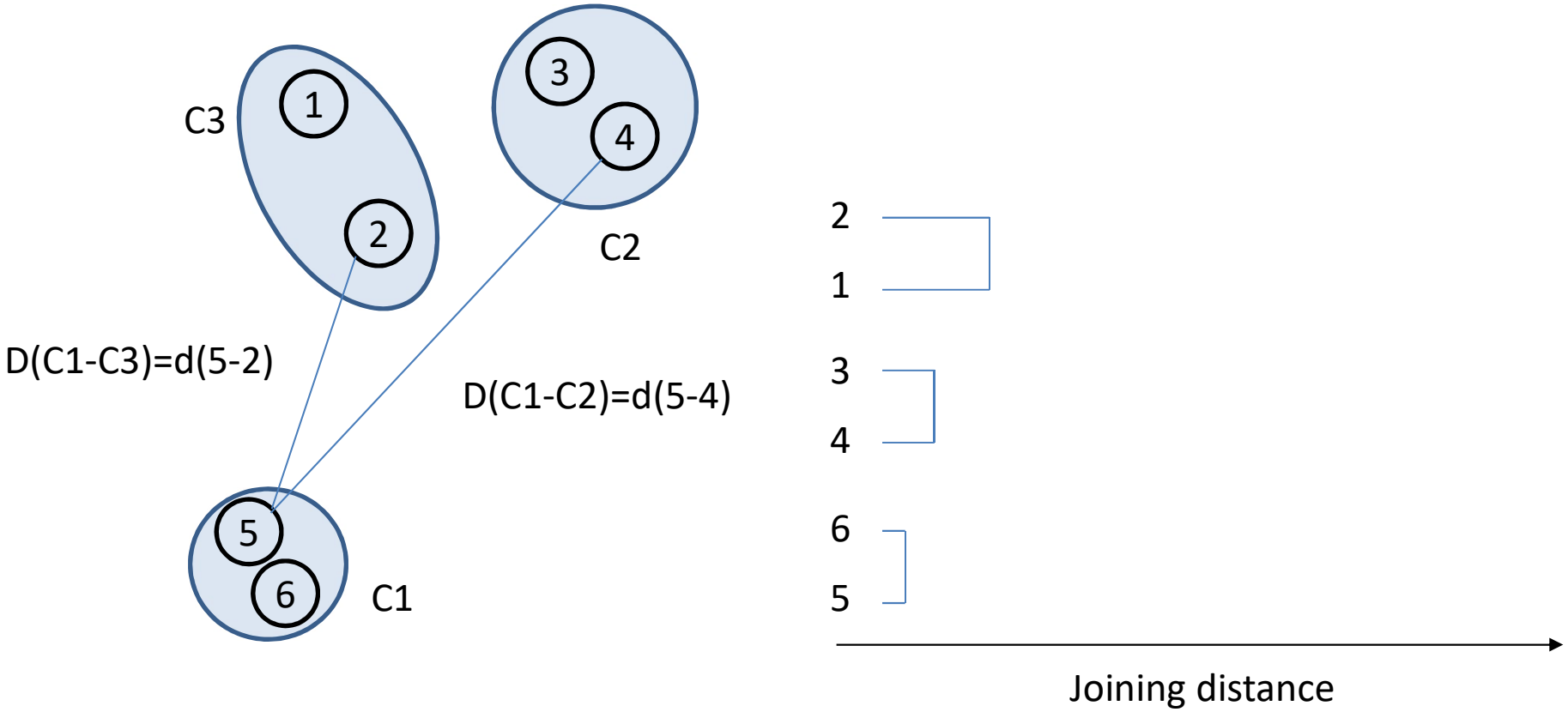
$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



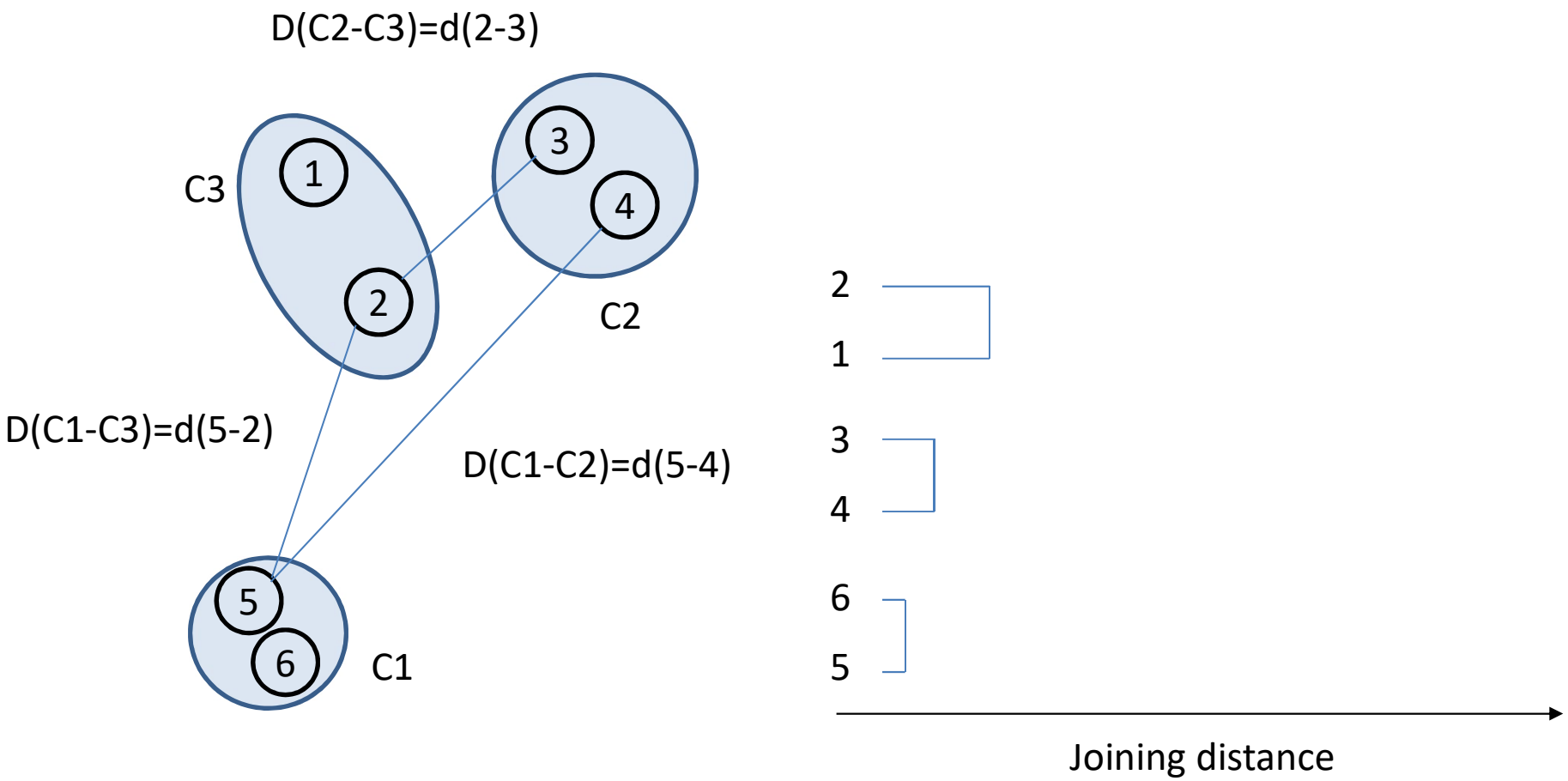
$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



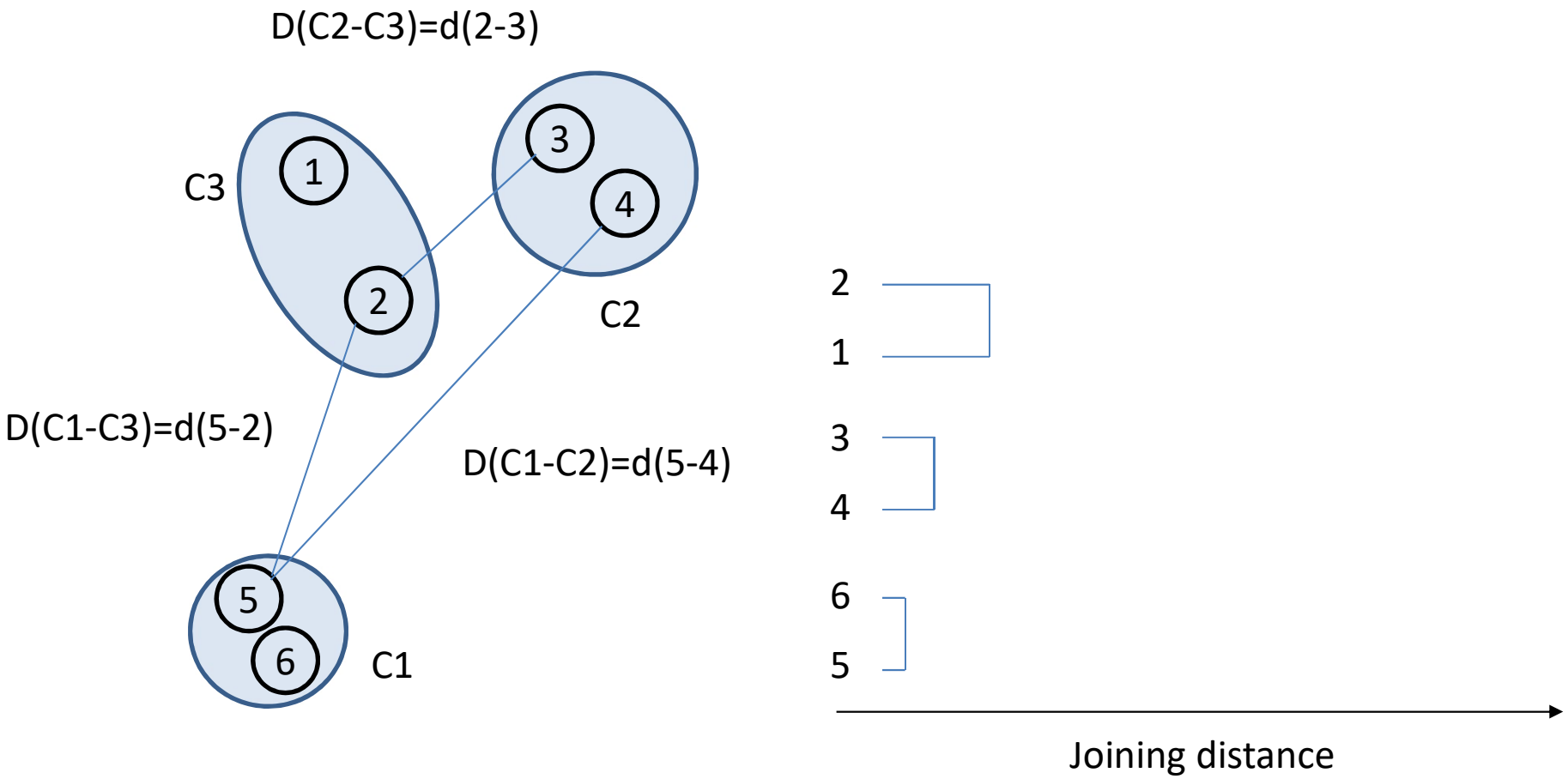
$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$

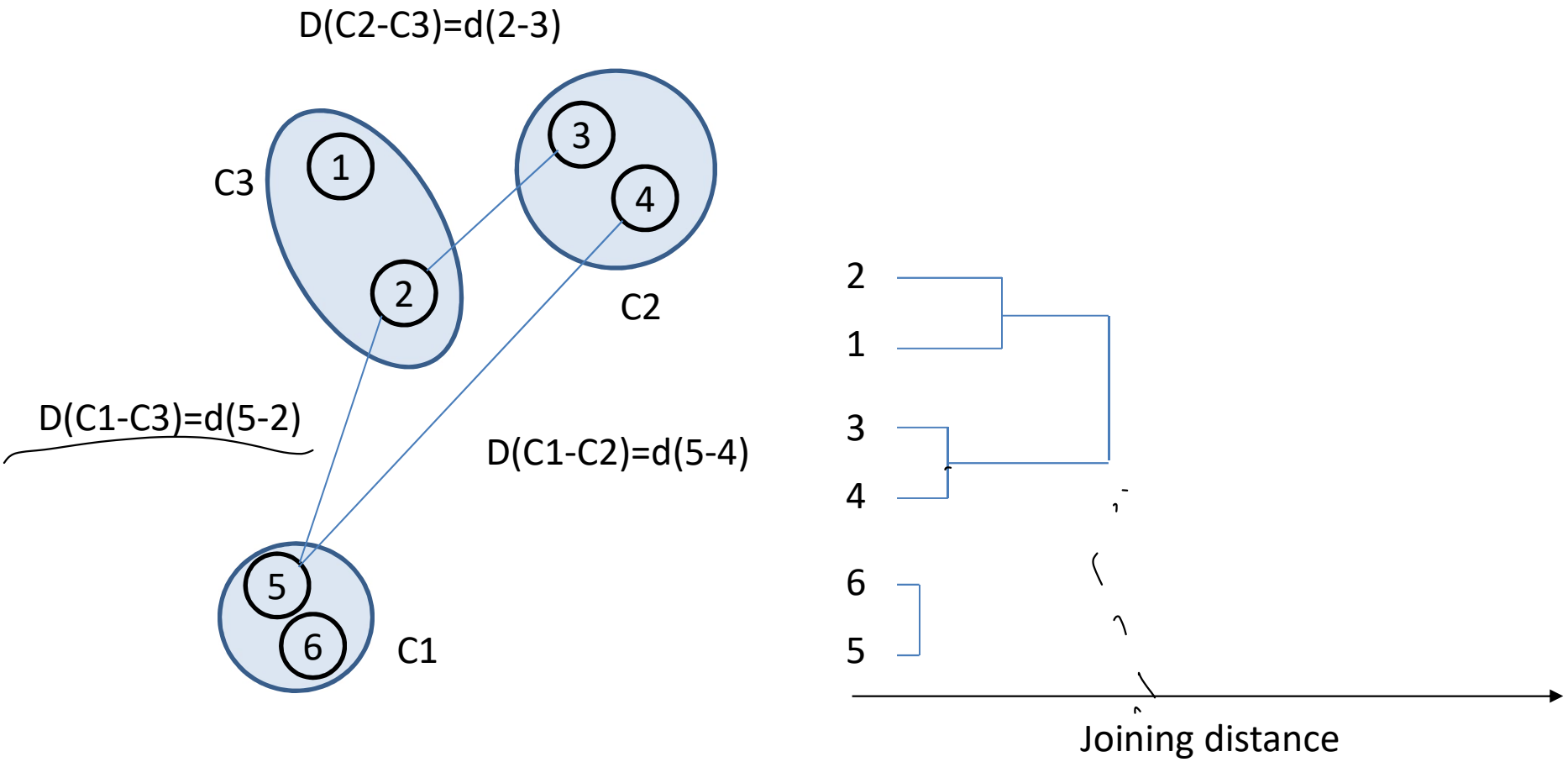


$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



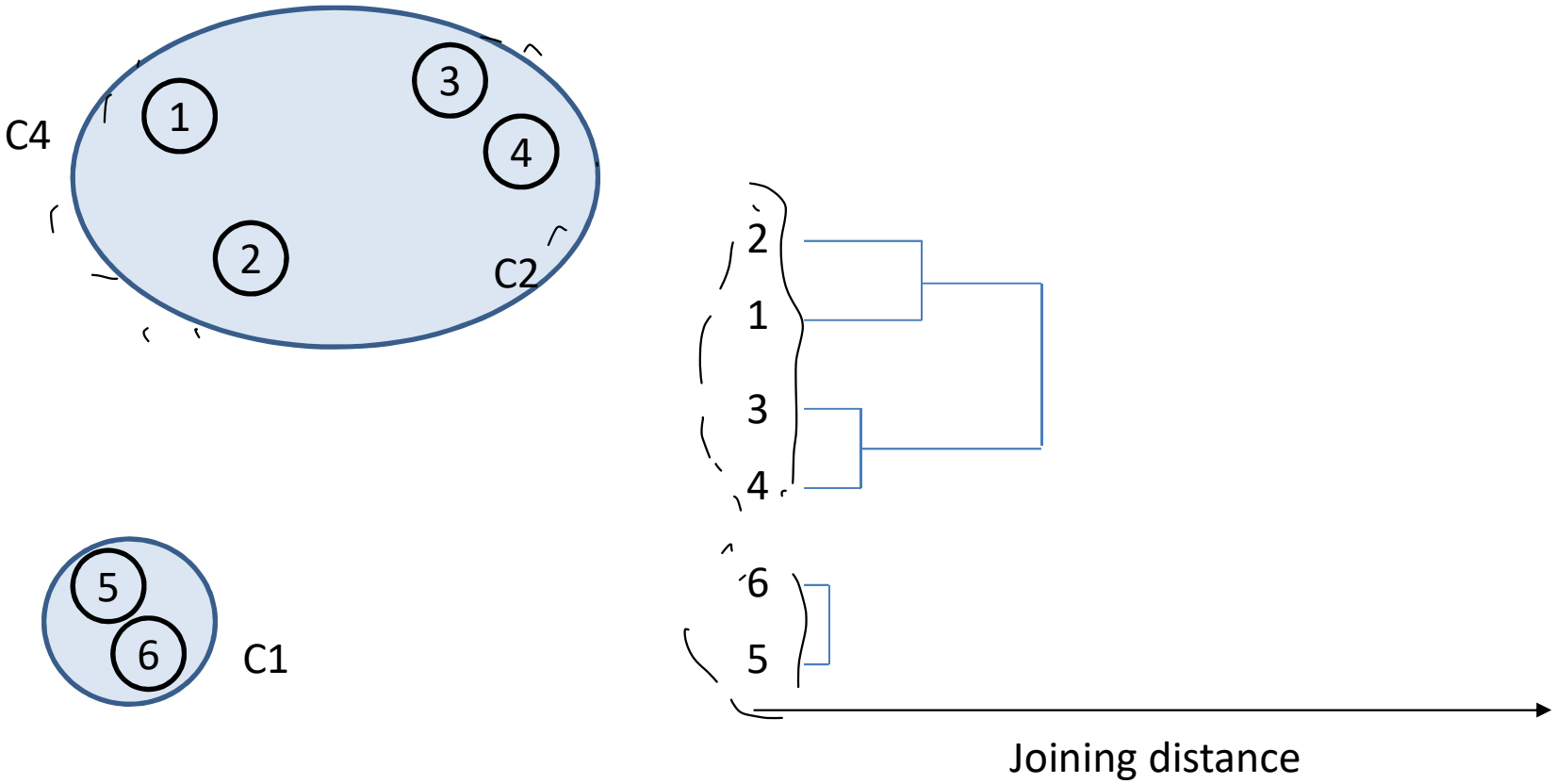
At each step: Pick the minimum distance and merge these elements

$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



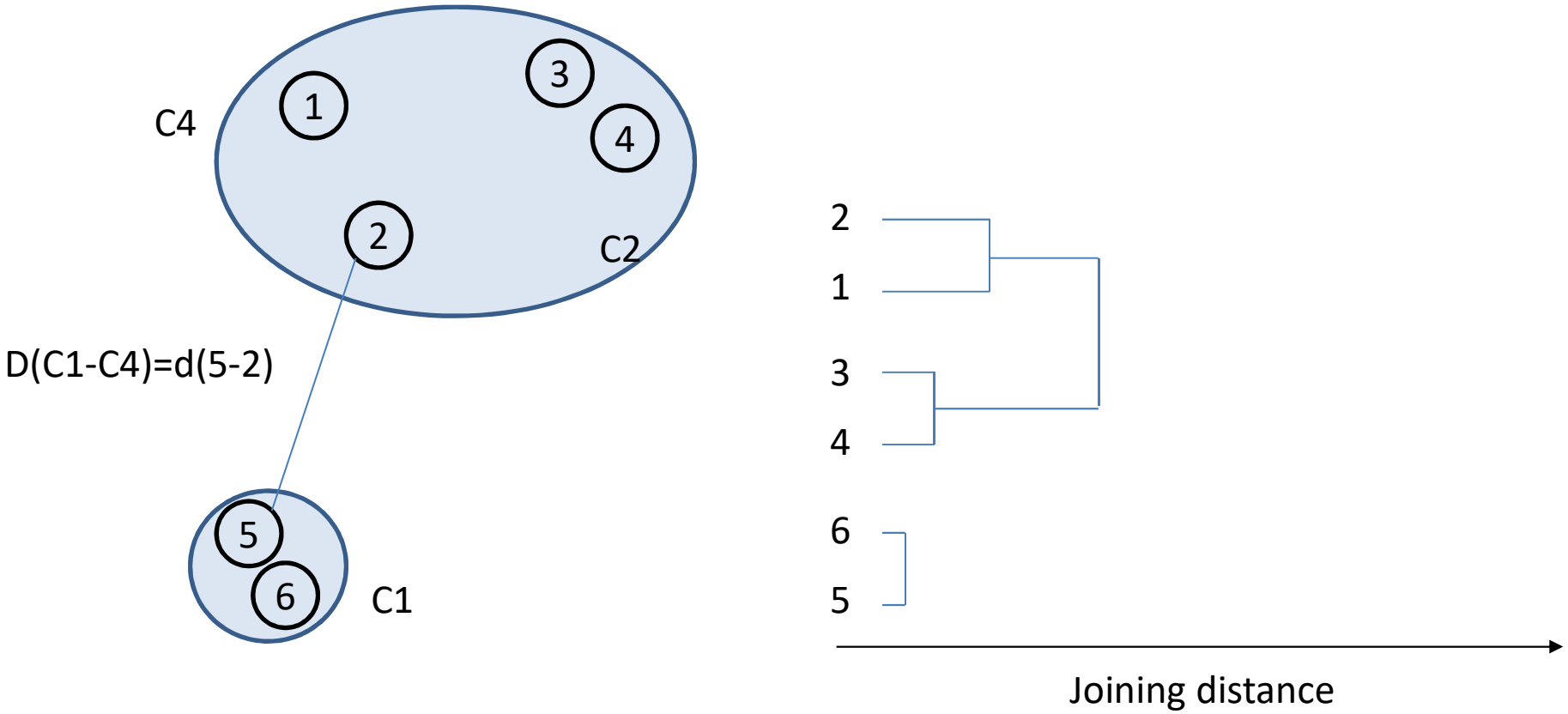
At each step: Pick the minimum distance and merge these elements

$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



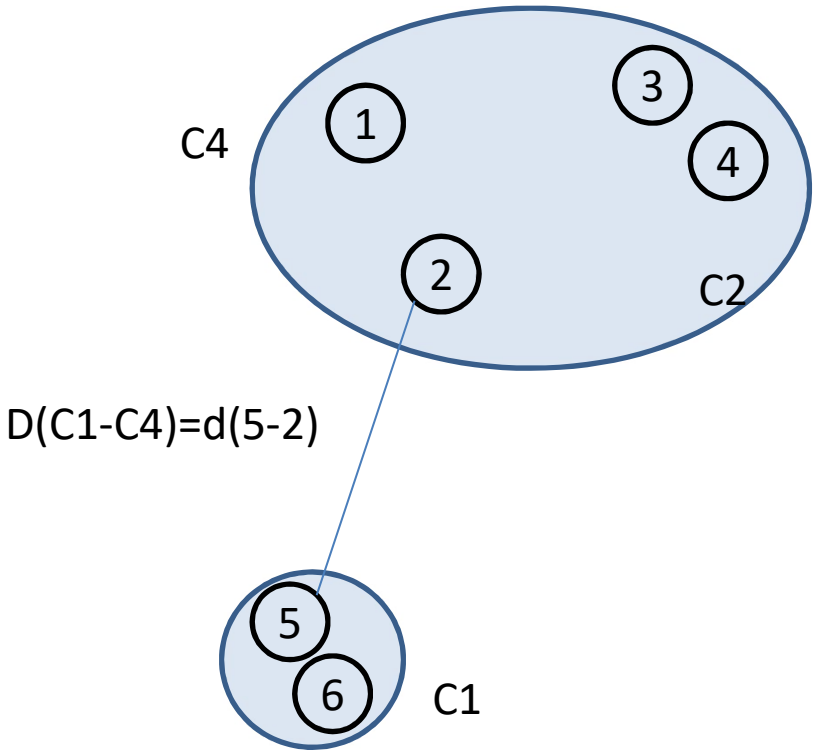
At each step: Pick the minimum distance and merge these elements

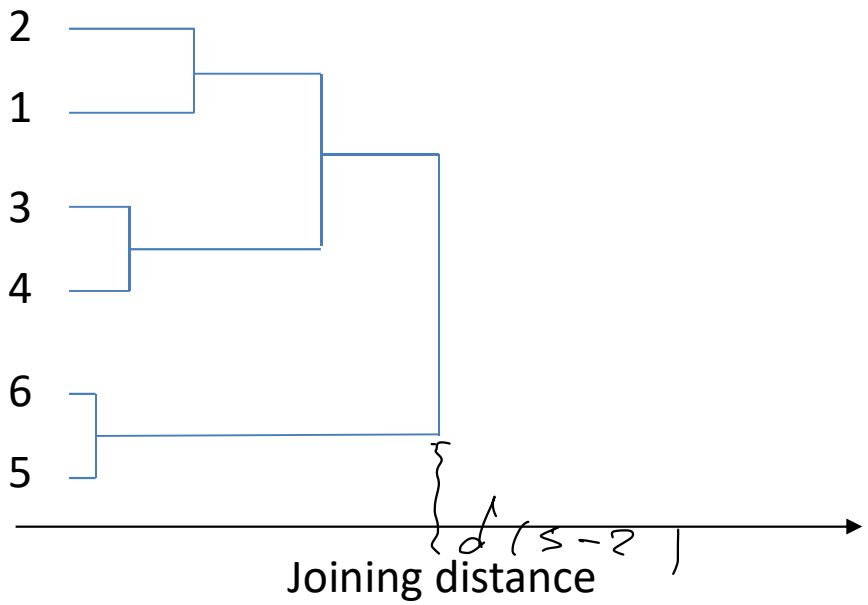
$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



At each step: Pick the minimum distance and merge these elements

$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$





At each step: Pick the minimum distance and merge these elements

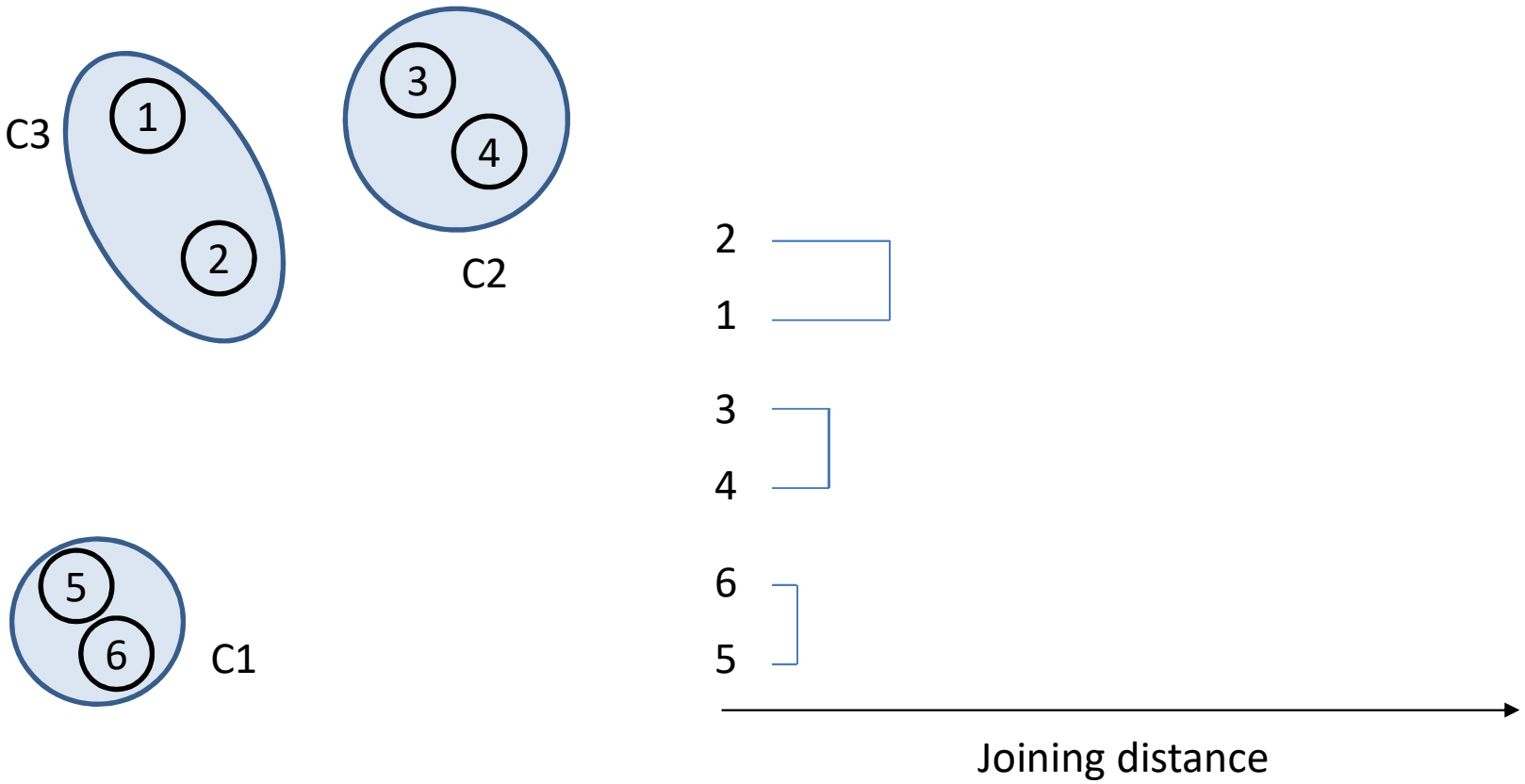
Complete-link distance

- **Complete-link distance** between clusters C_i and C_j is the ***maximum distance*** between any object in C_i and any object in C_j

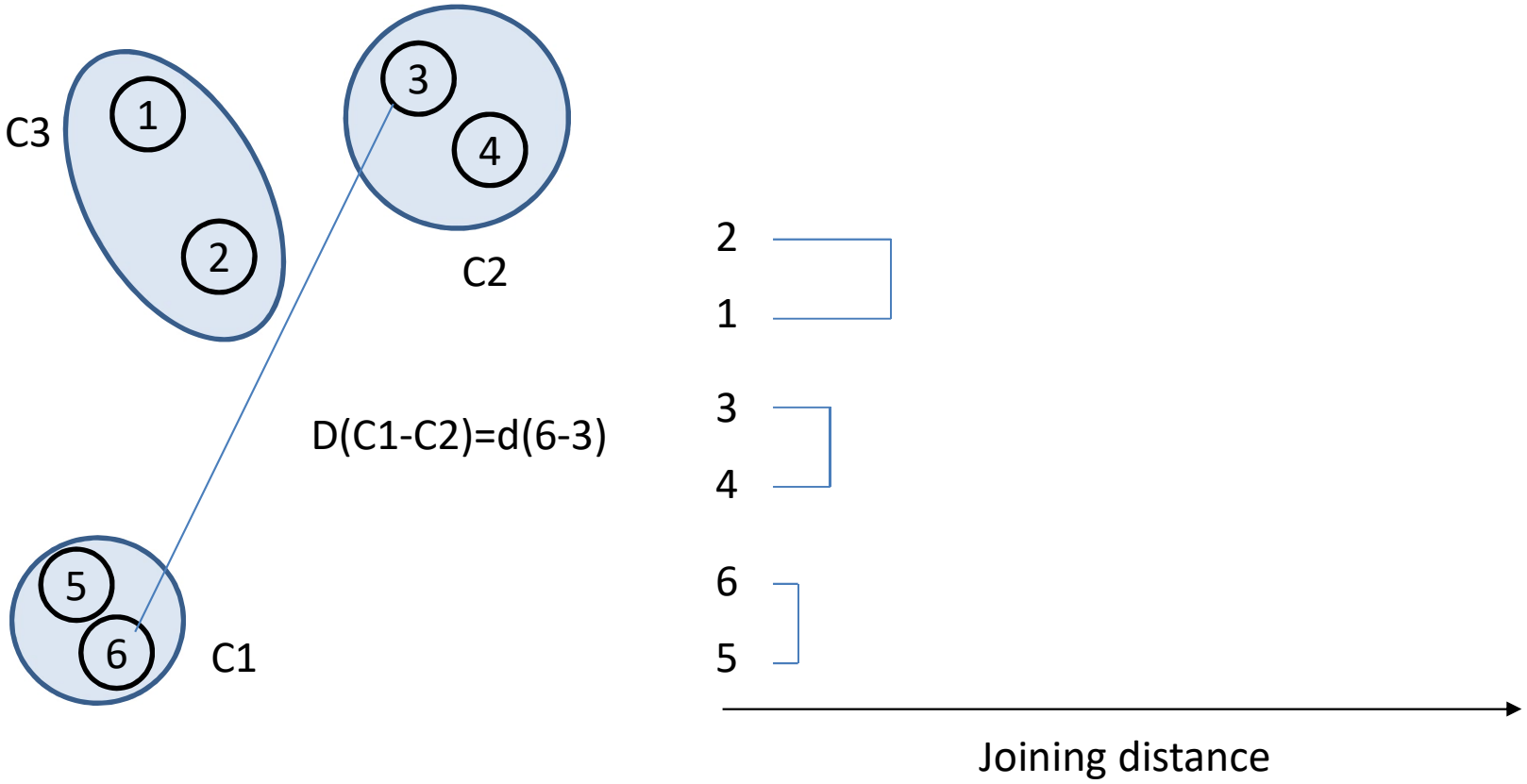
- The distance is **defined by the two most dissimilar (i.e. furthest) objects**

$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x, y) \middle| x \in C_i, y \in C_j \right\}$$

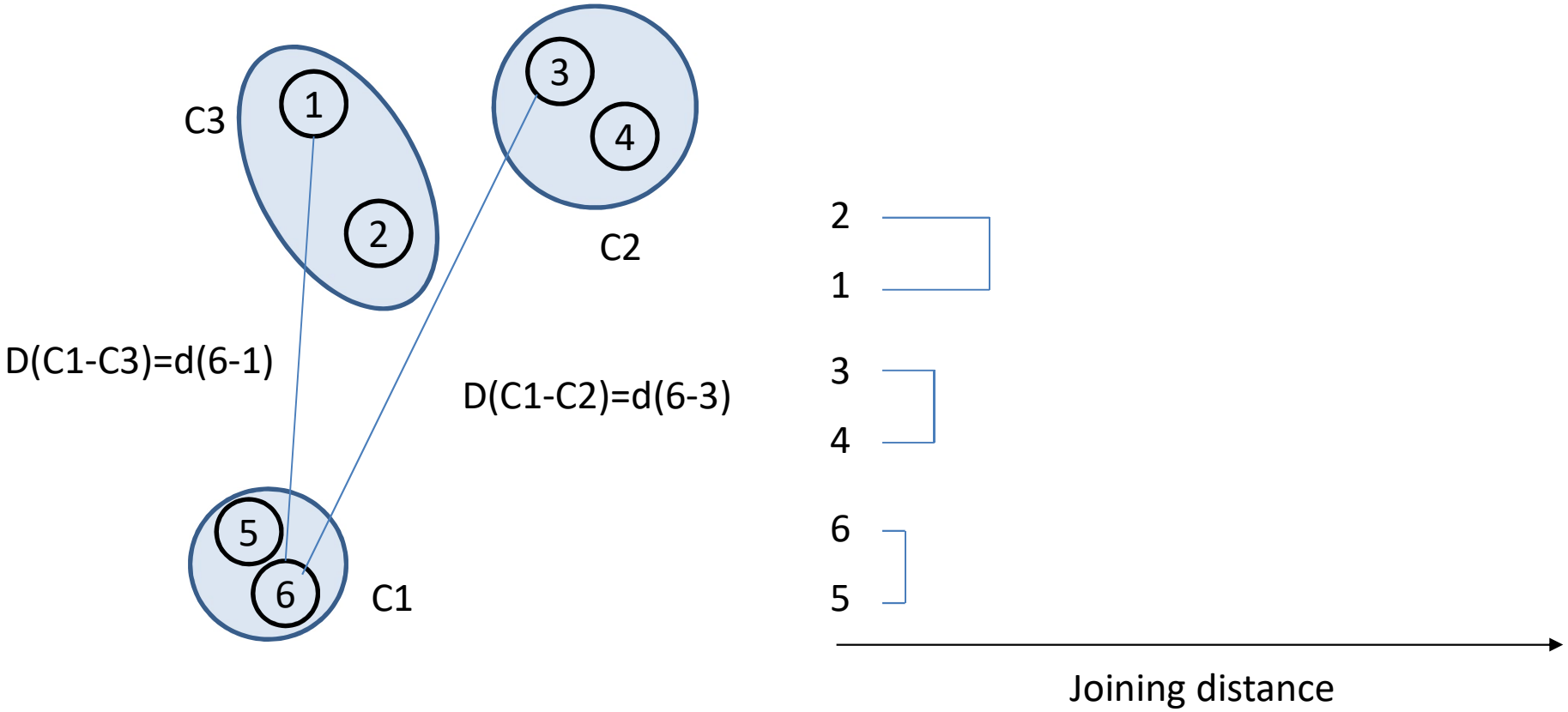
$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



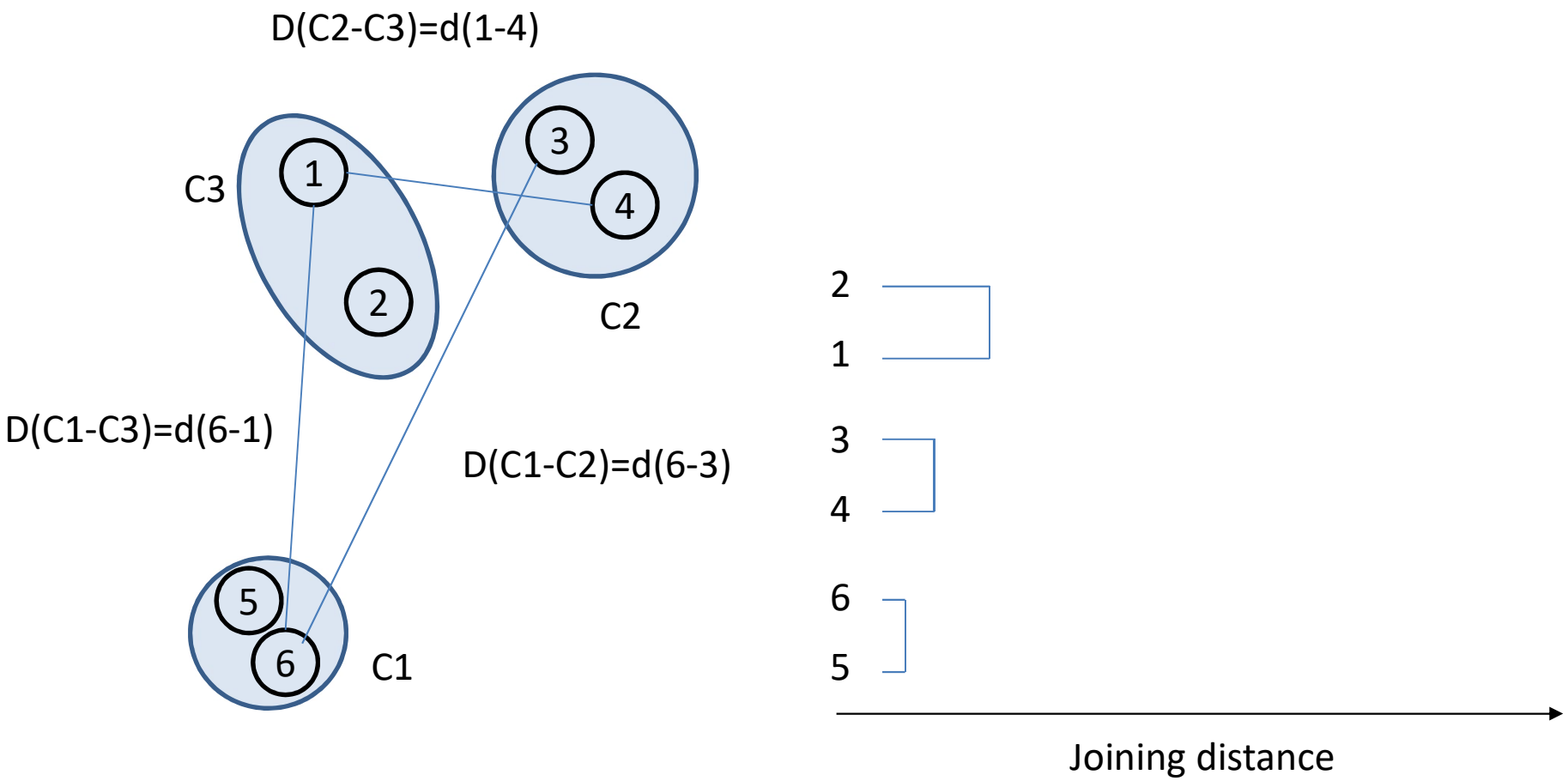
$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



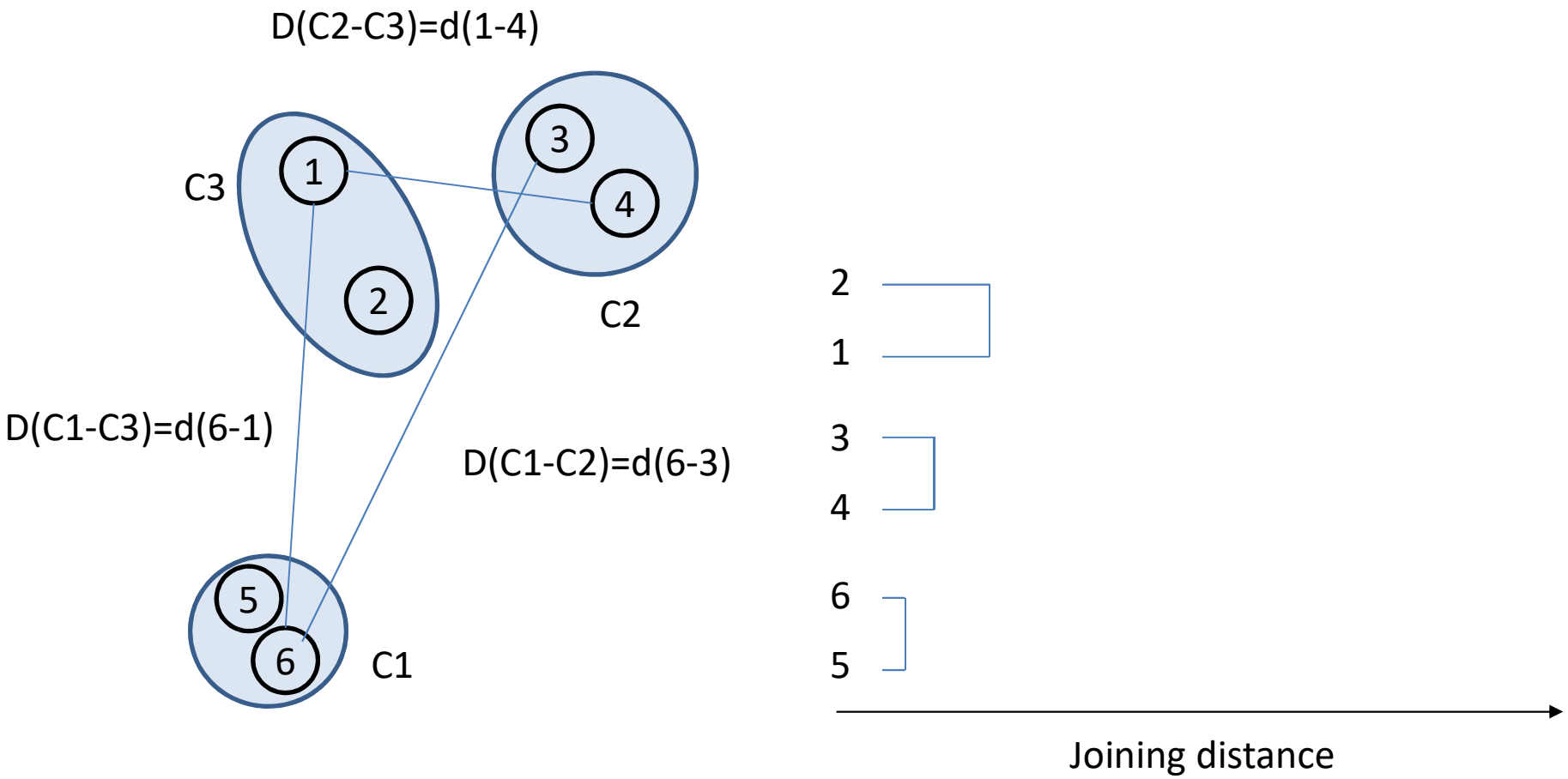
$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$

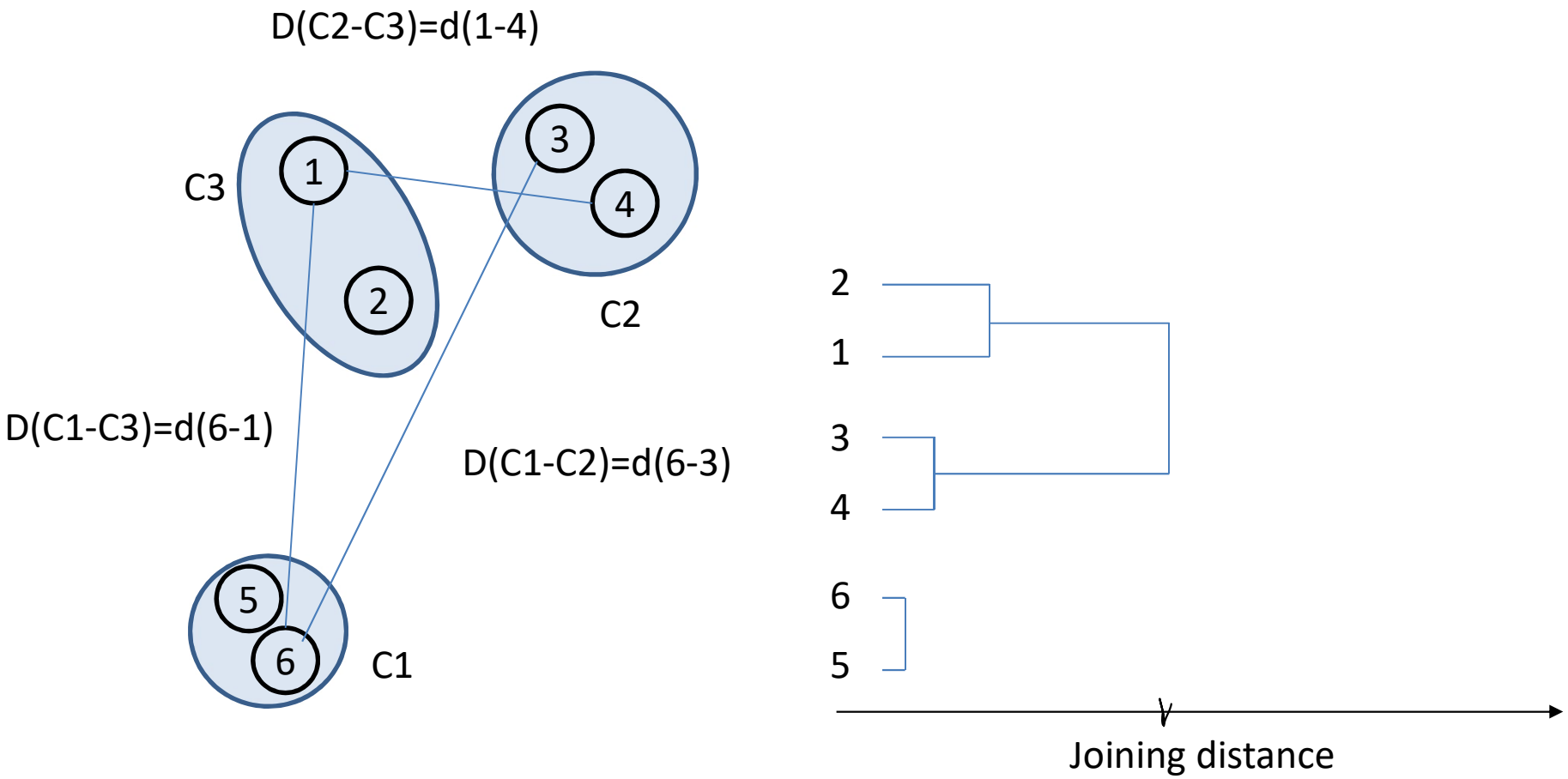


$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



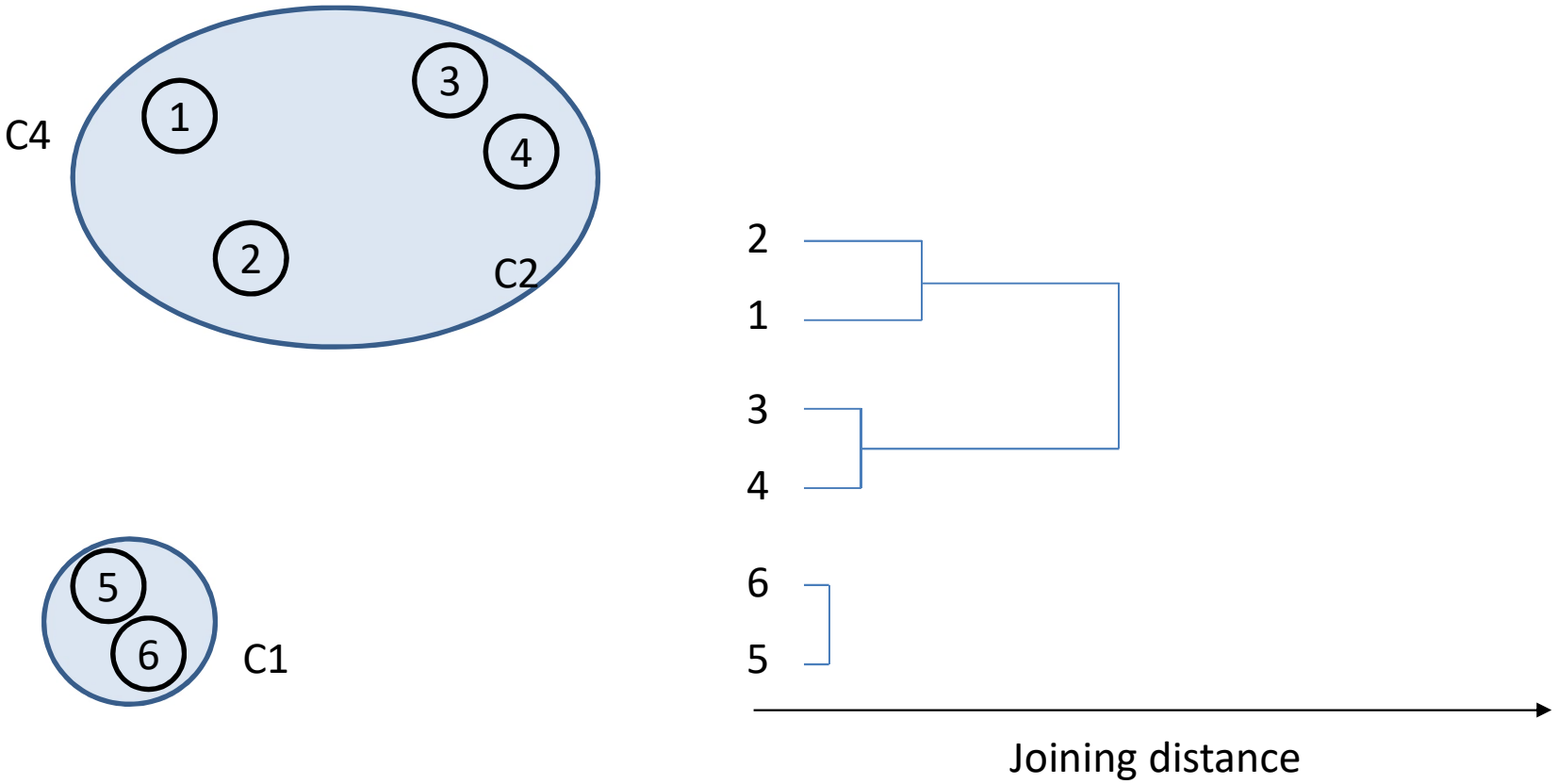
At each step: Pick the minimum distance and merge these elements

$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



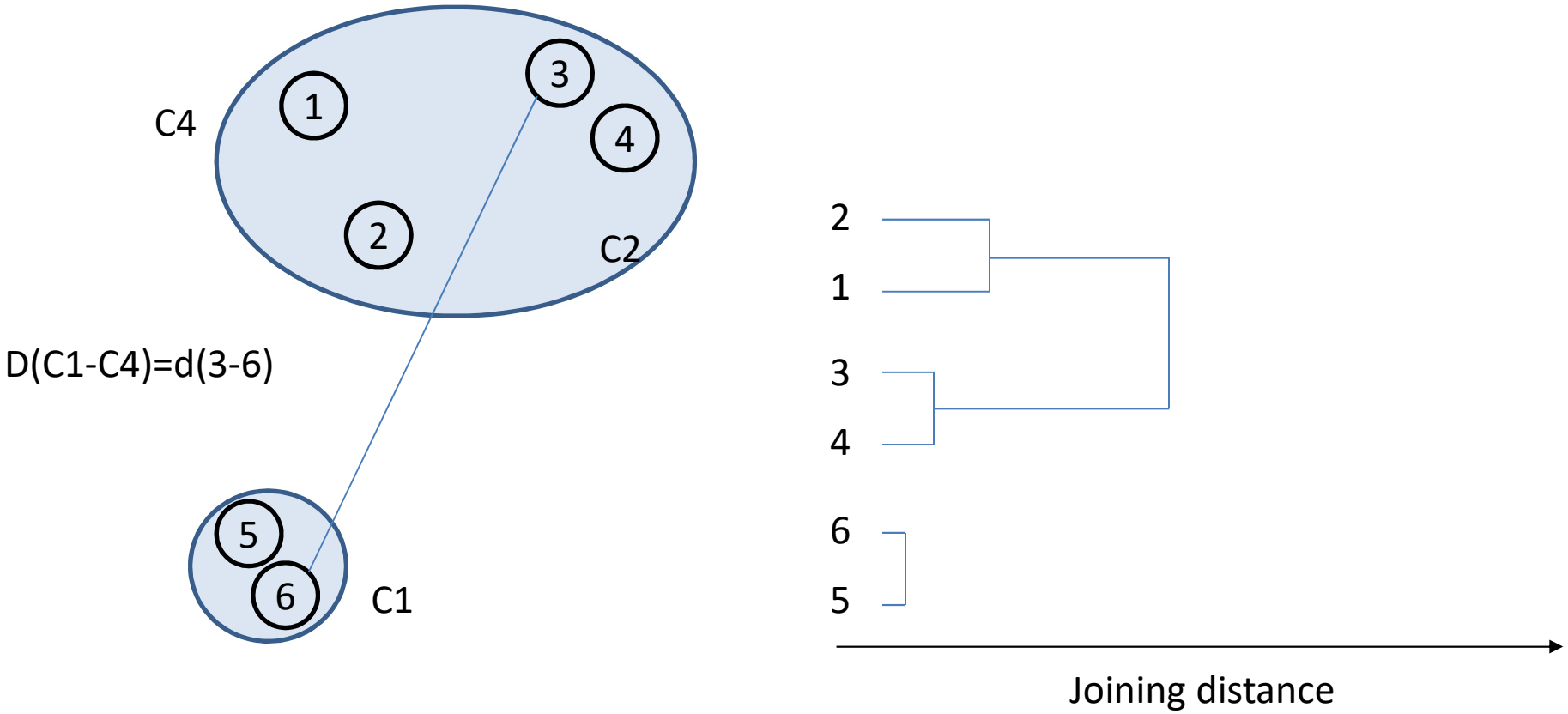
At each step: Pick the minimum distance and merge these elements

$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



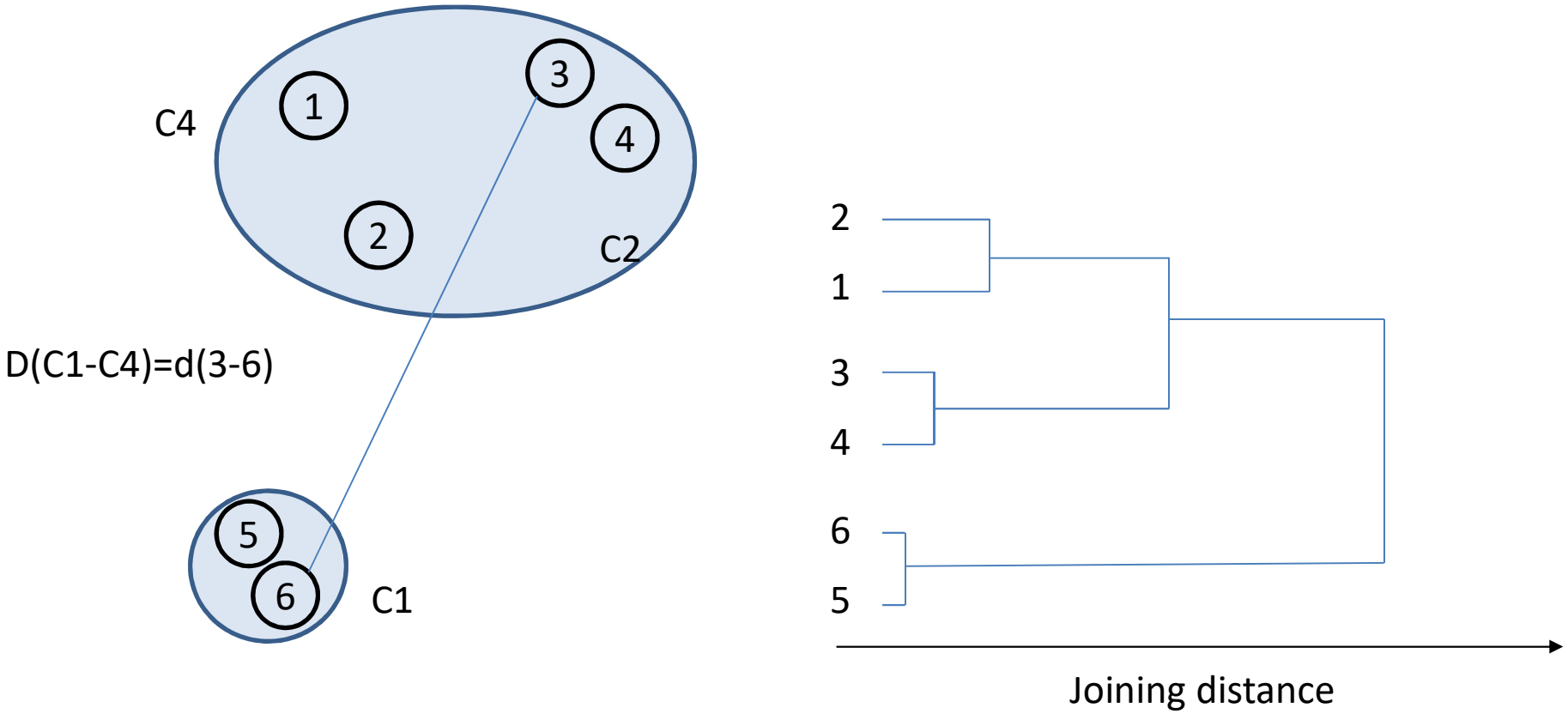
At each step: Pick the minimum distance and merge these elements

$$D_{cl}(C_i, C_j) = \max_{x,y} \left\{ d(x,y) \middle| x \in C_i, y \in C_j \right\}$$



At each step: Pick the minimum distance and merge these elements

$$D_{cl}(C_i, C_j) = \max_{x,y} \{ d(x,y) \mid x \in C_i, y \in C_j \}$$



At each step: Pick the minimum distance and merge these elements

Group average distance

- **Group average distance** between clusters C_i and C_j is the ***average distance*** between any object in C_i and any object in C_j

$$D_{avg}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

Distance between two clusters

- **Centroid distance** between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{centroids}(C_i, C_j) = d(r_i, r_j)$$

Distance between two clusters

- **Ward's distance** between clusters C_i and C_j is the ***difference*** between the ***total within cluster sum of squares for the two clusters separately***, and the ***within cluster sum of squares resulting from merging the two clusters*** in cluster C_{ij}

$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

- r_i: centroid of C_i
- r_j: centroid of C_j
- r_{ij}: centroid of C_{ij}

Updating Ward's distance

- After merging C_i with C_j, the distance between the new cluster $C_{i\cup j}$ and another cluster C_k is computed with the formula:

(kind of a weighted average)

$$D\left(C_{i\cup j}, C_k\right) =$$

$$= \frac{n_i + n_k}{n_i + n_j + n_k} D(C_i, C_k) + \frac{n_j + n_k}{n_i + n_j + n_k} D\left(C_j, C_k\right)$$

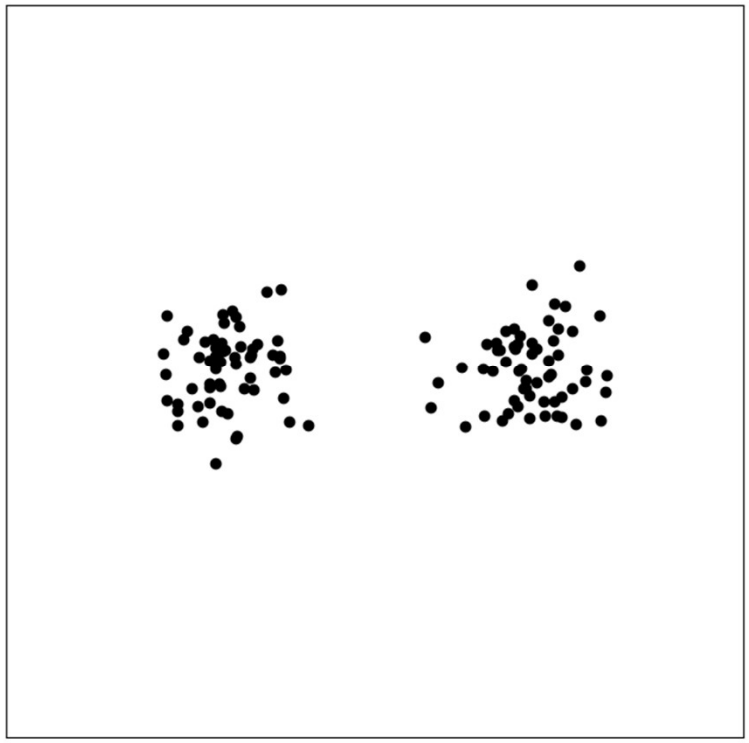
$$- \frac{n_k}{n_i + n_j + n_k} D\left(C_i, C_j\right)$$

Ward's distance for clusters

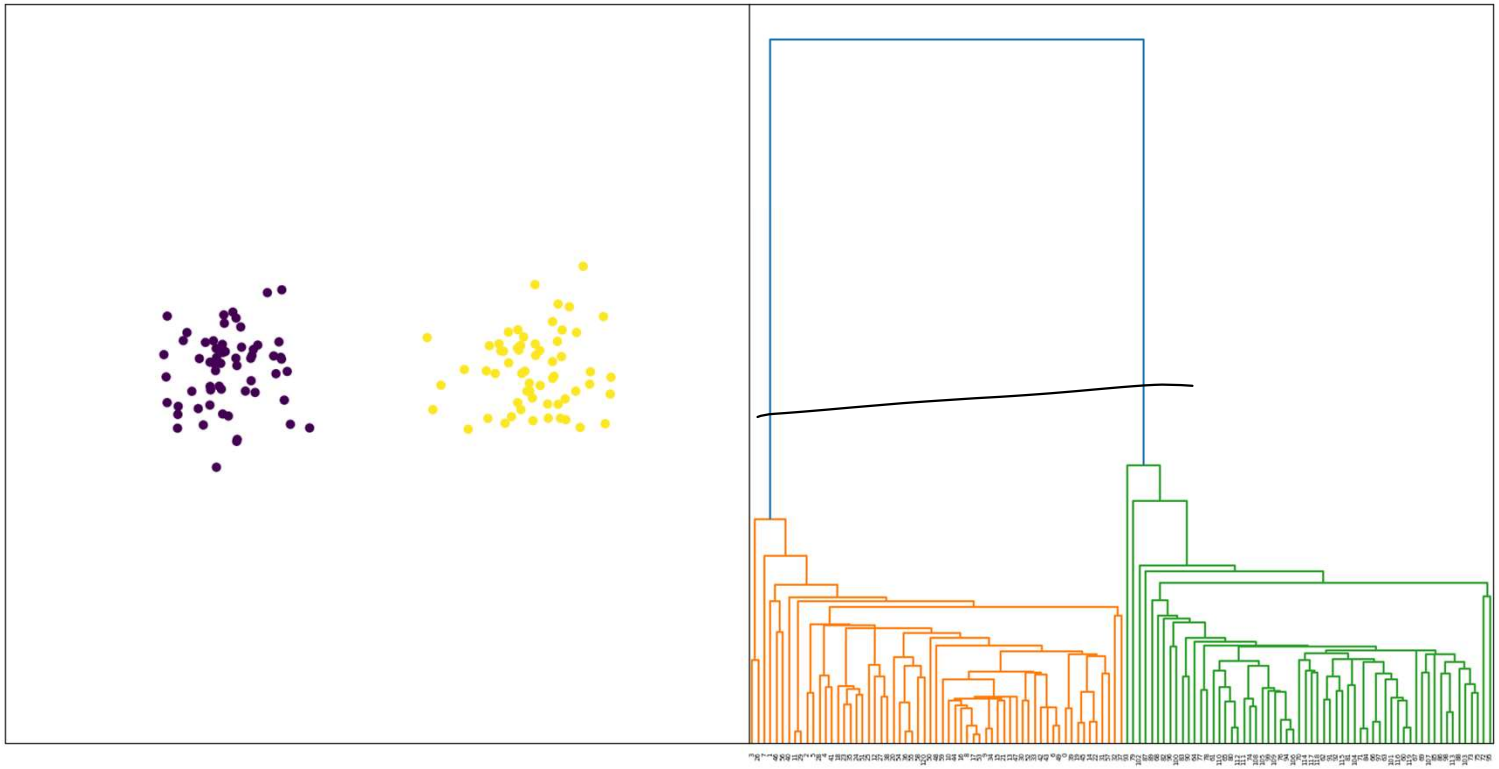
- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means

Mix of two equally sized gaussian distributions

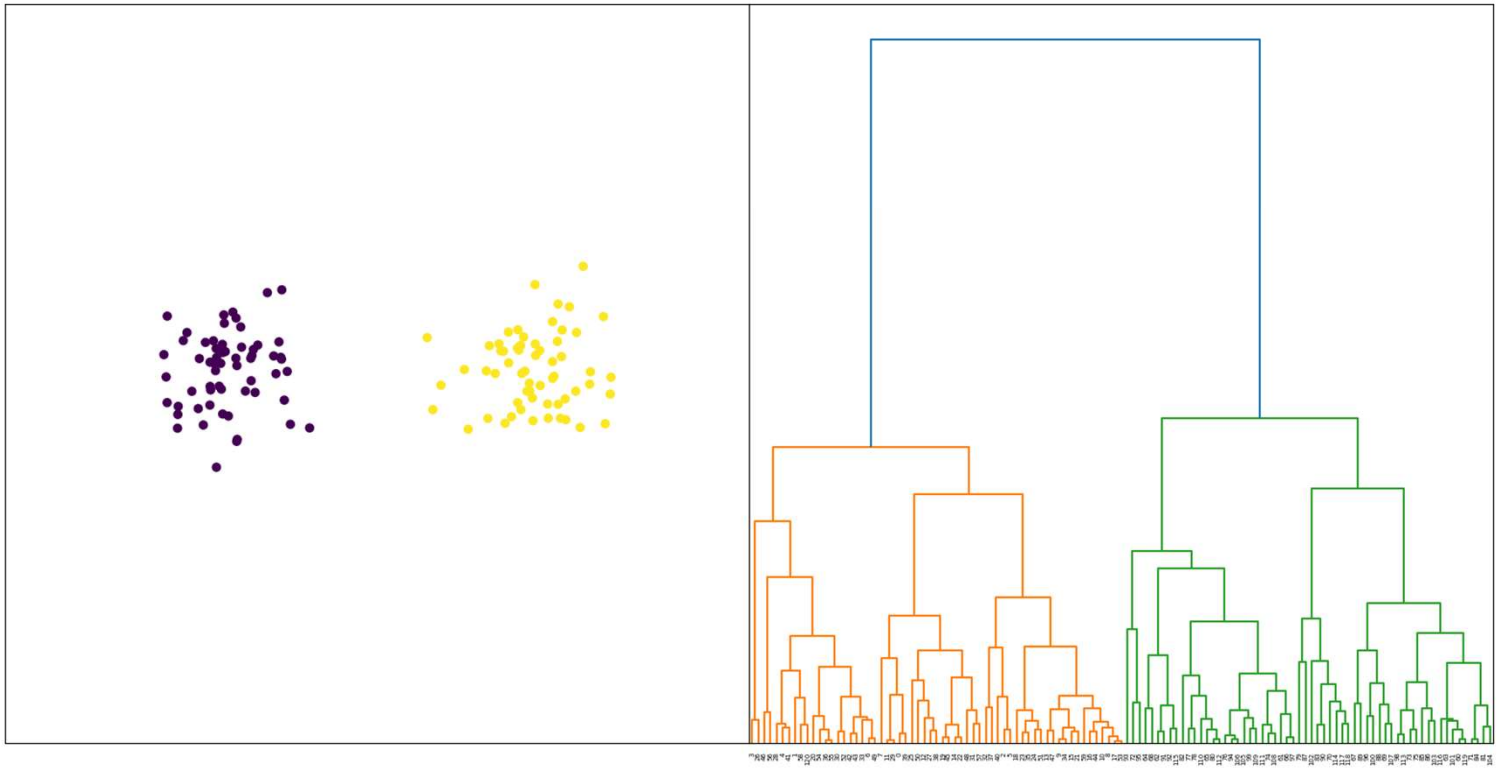
Let's see how these methods work for diff clusters :-



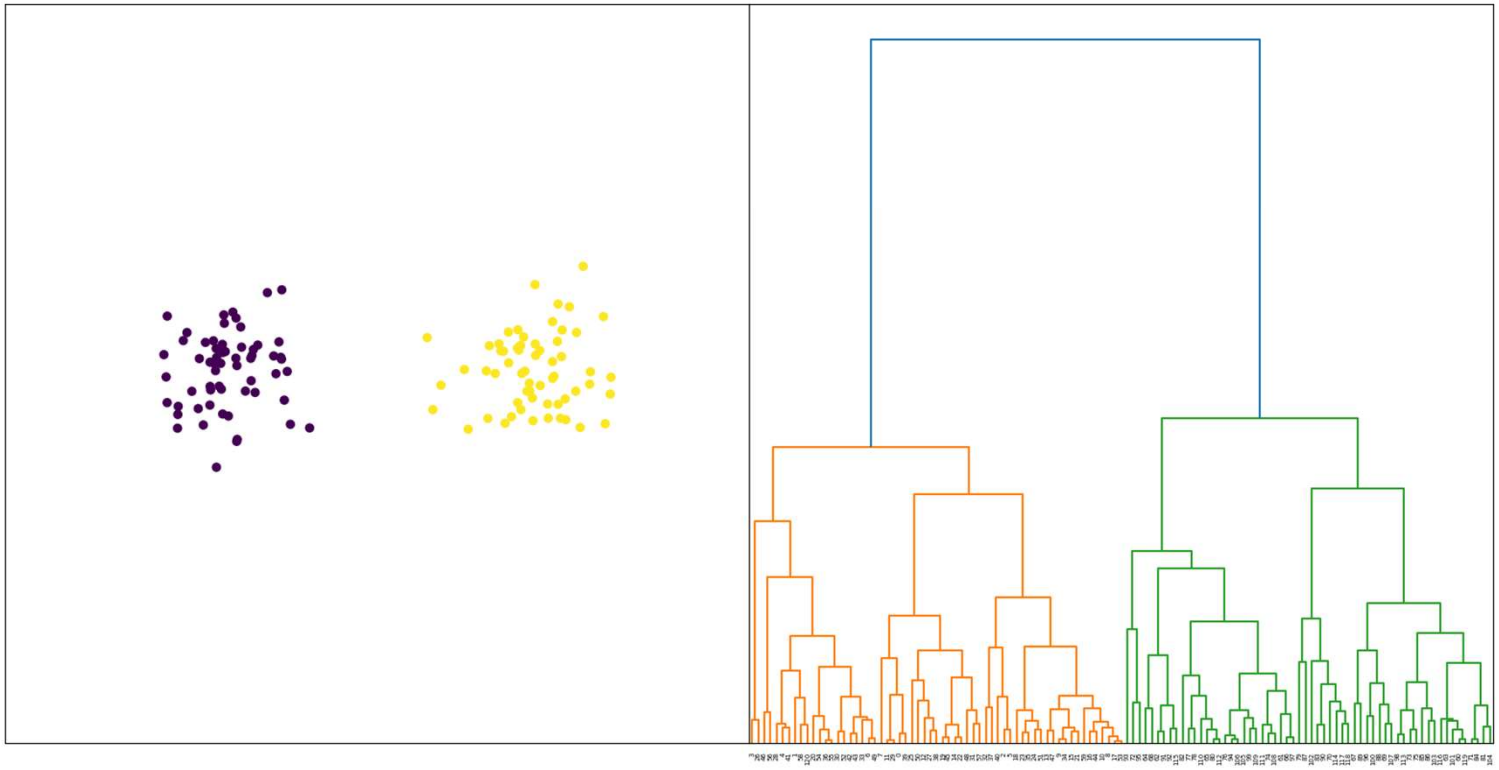
Single Linkage



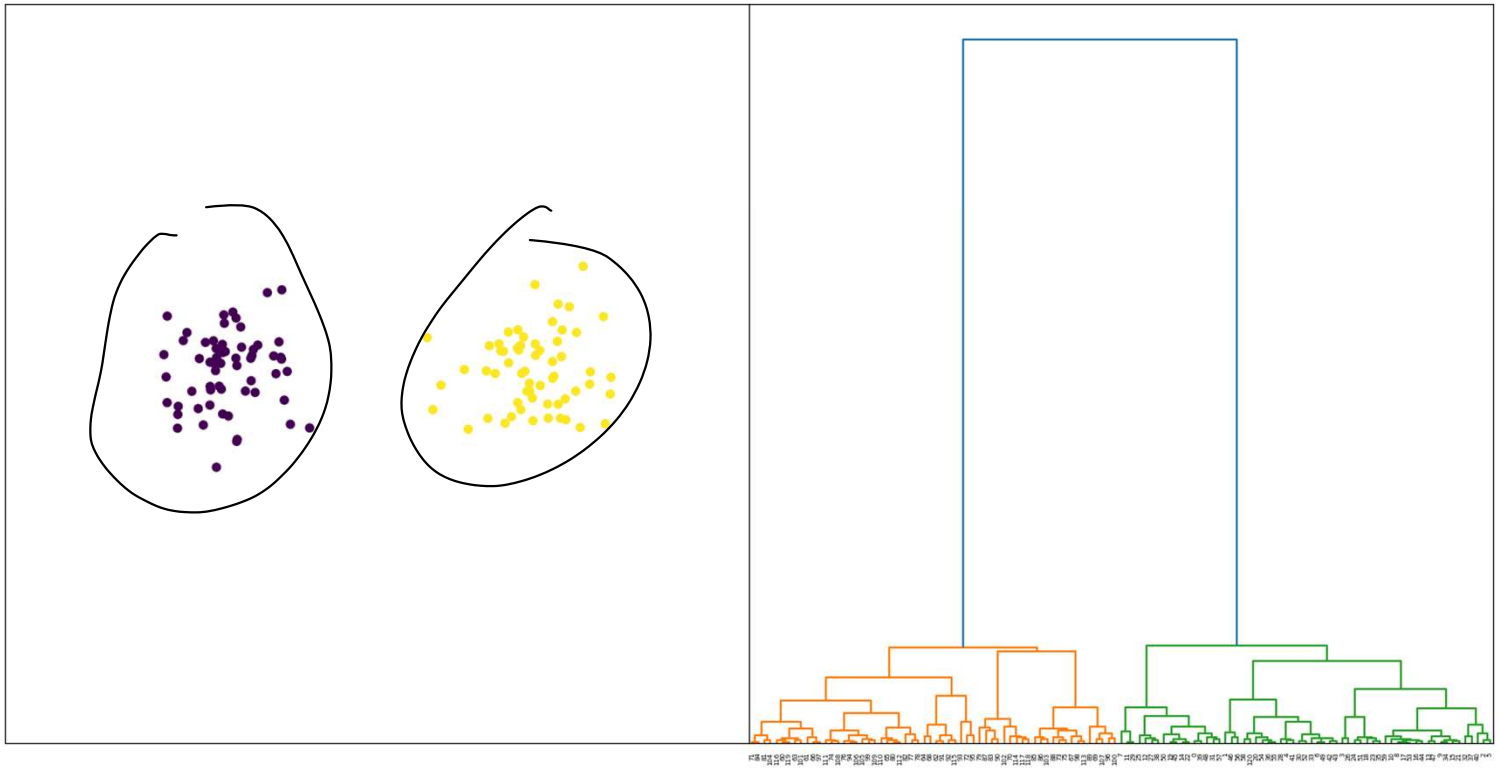
Complete Linkage



Average Linkage

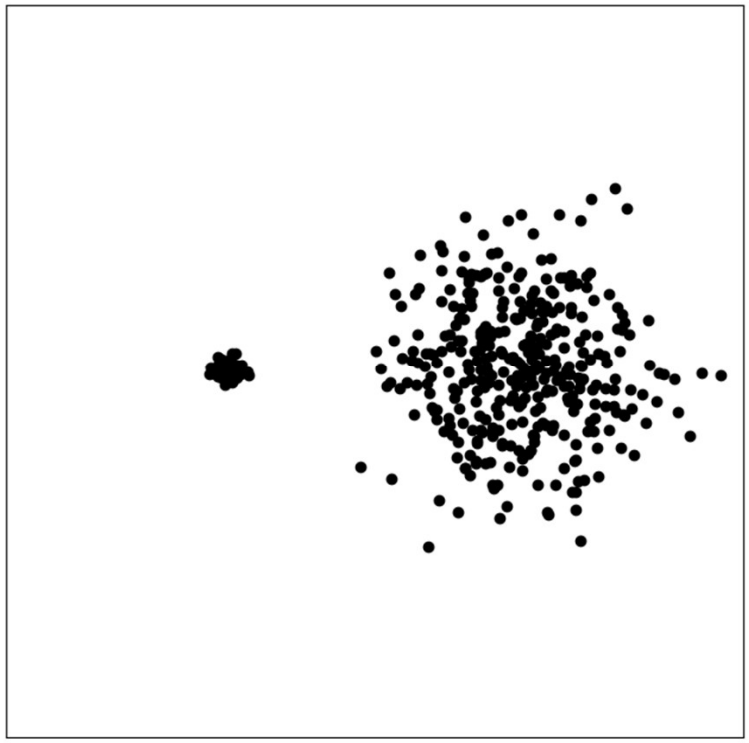


Ward's Linkage

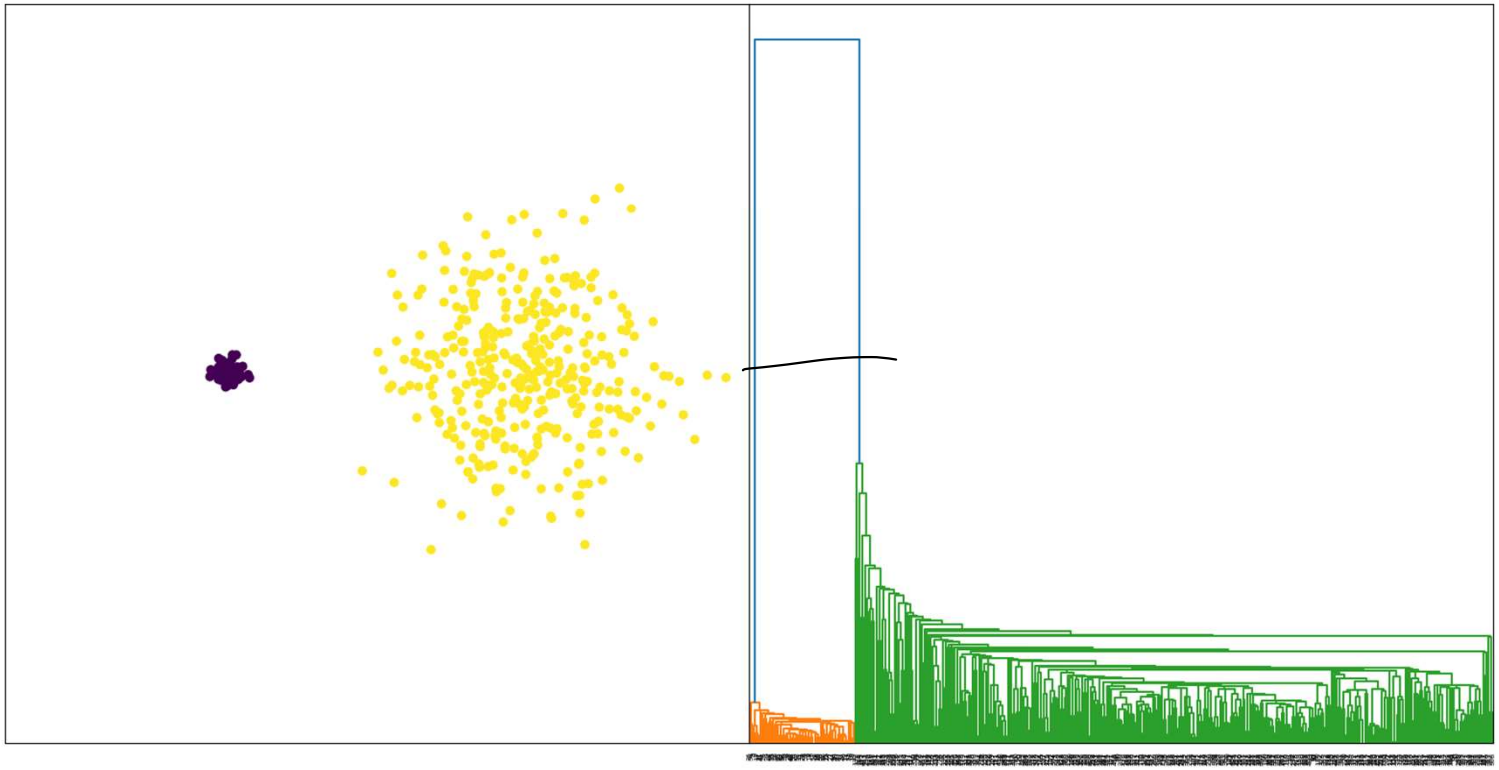


all work well for simple case

Mix of two different sized gaussian distributions

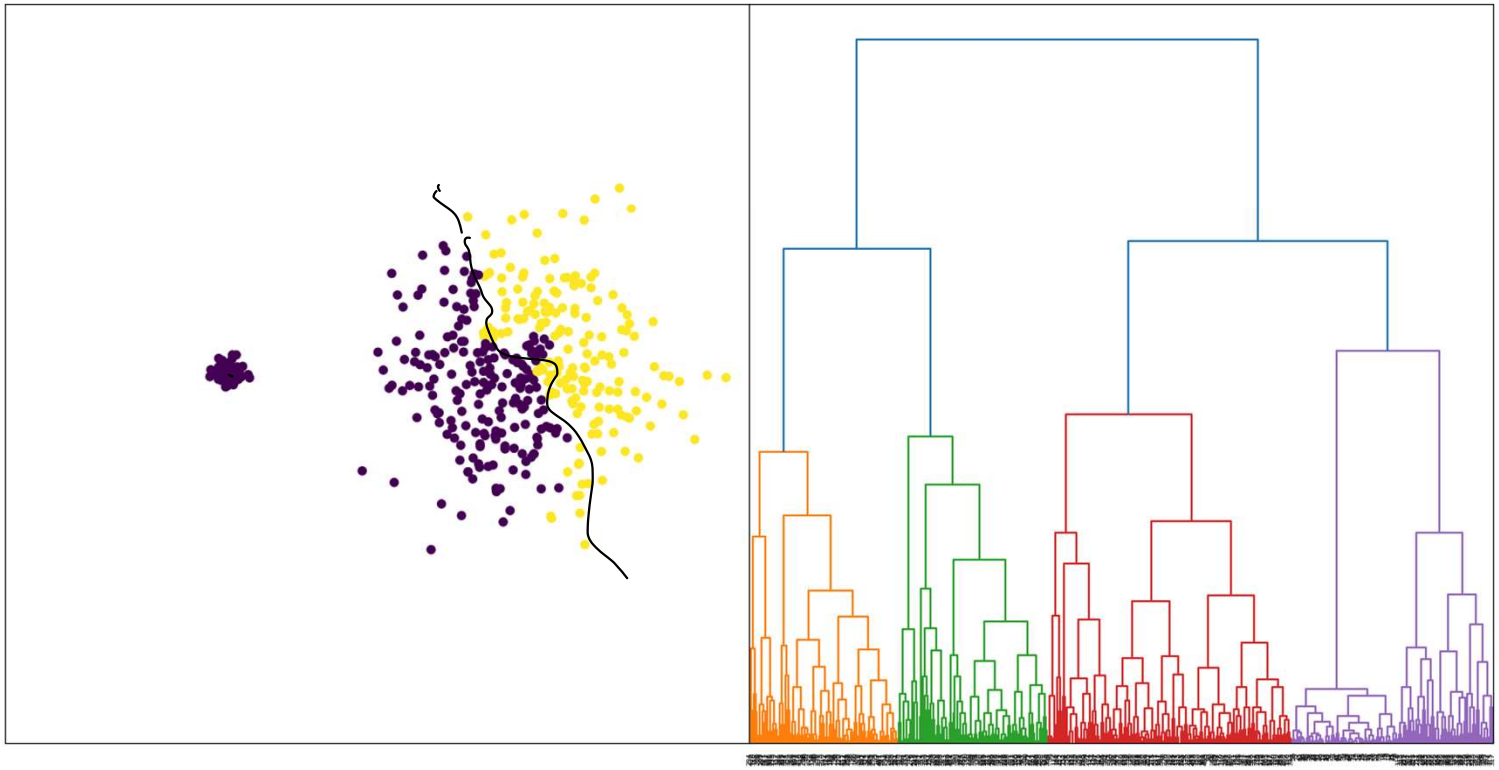


Single Linkage



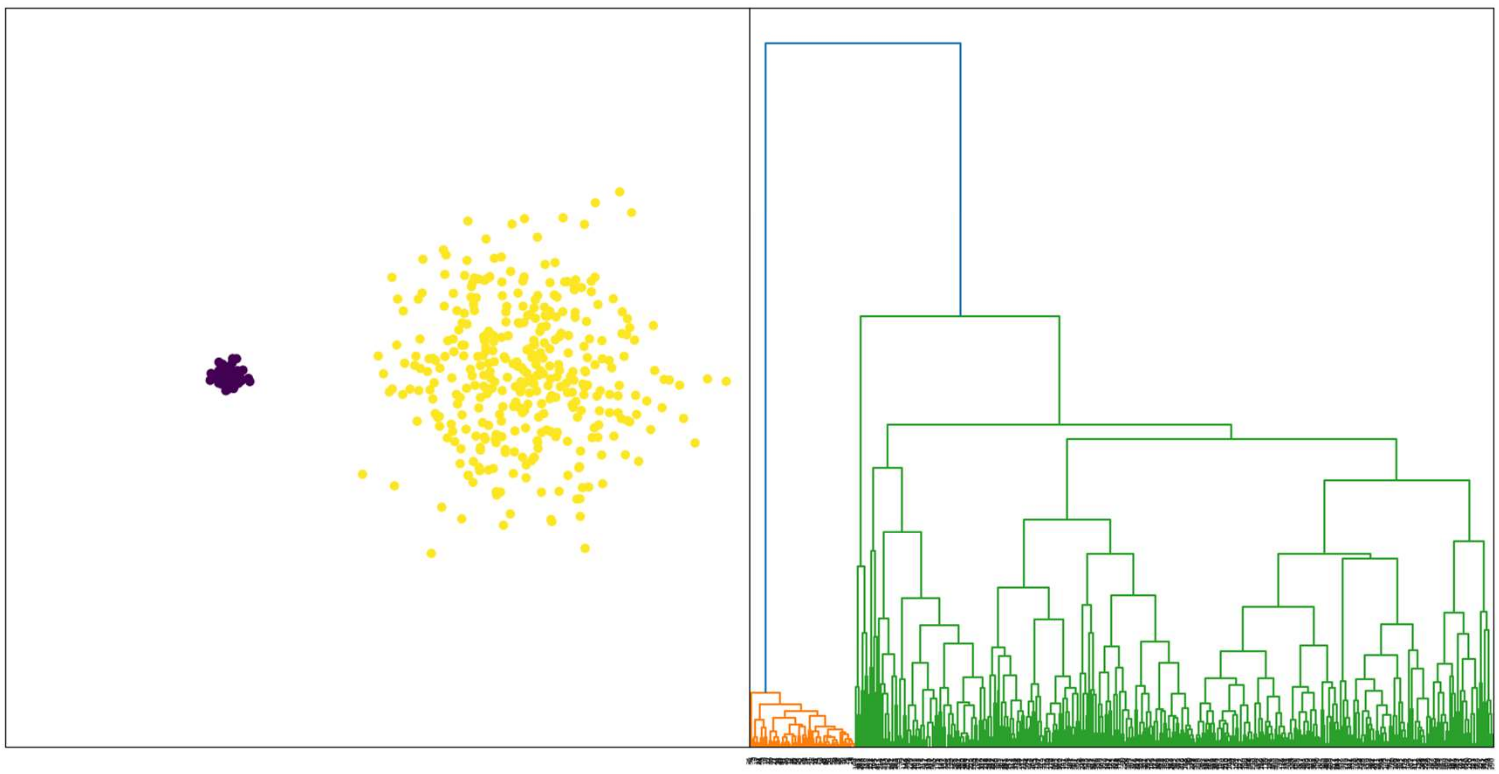
works fine

Complete Linkage



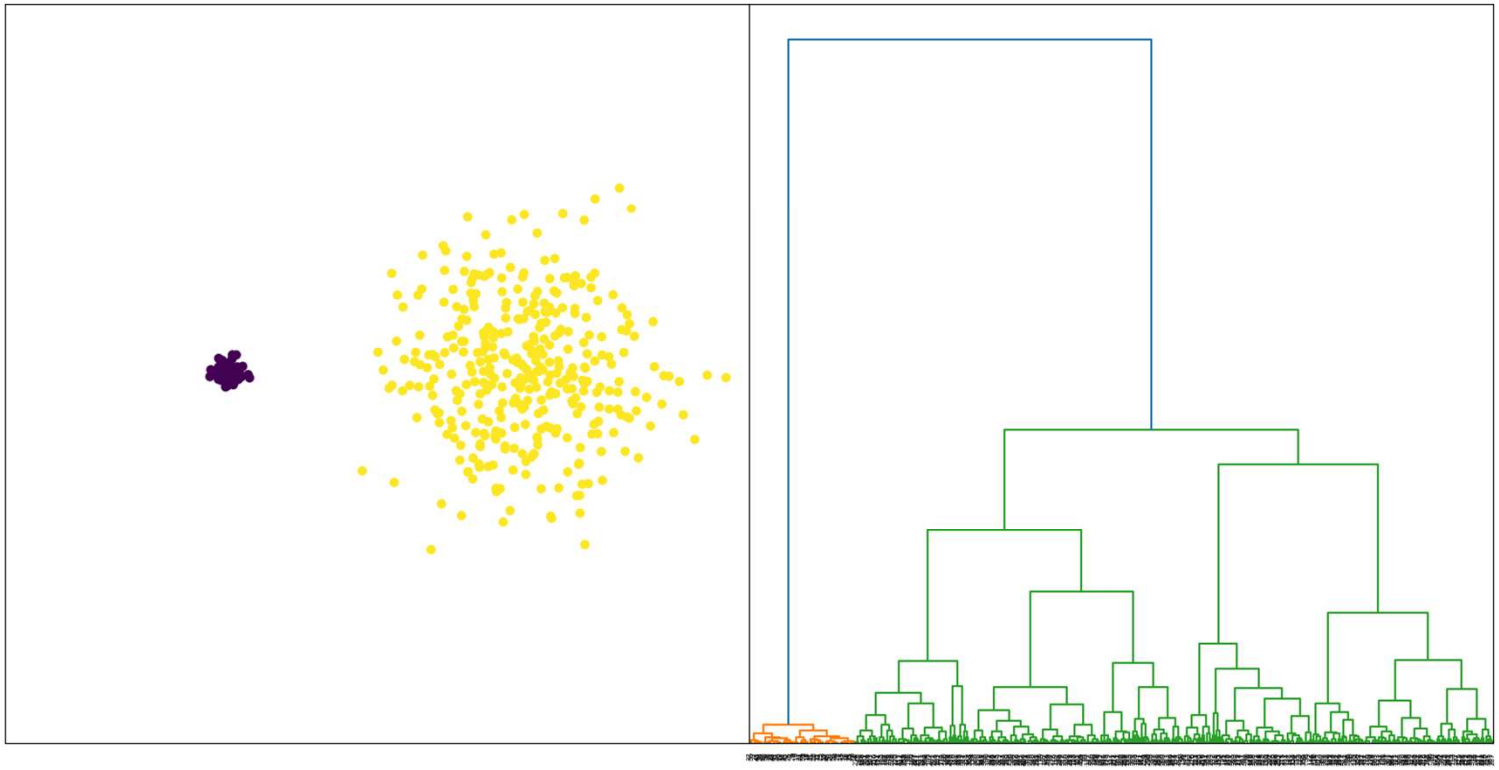
doesn't work well : complete linkage tries to generate clusters of same size.

Average Linkage



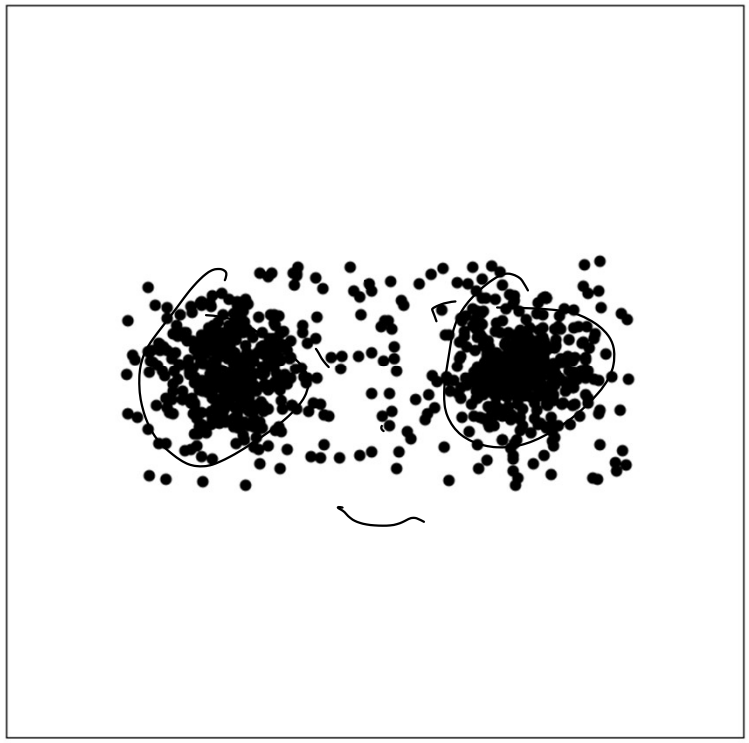
works well

Ward's Linkage



works well

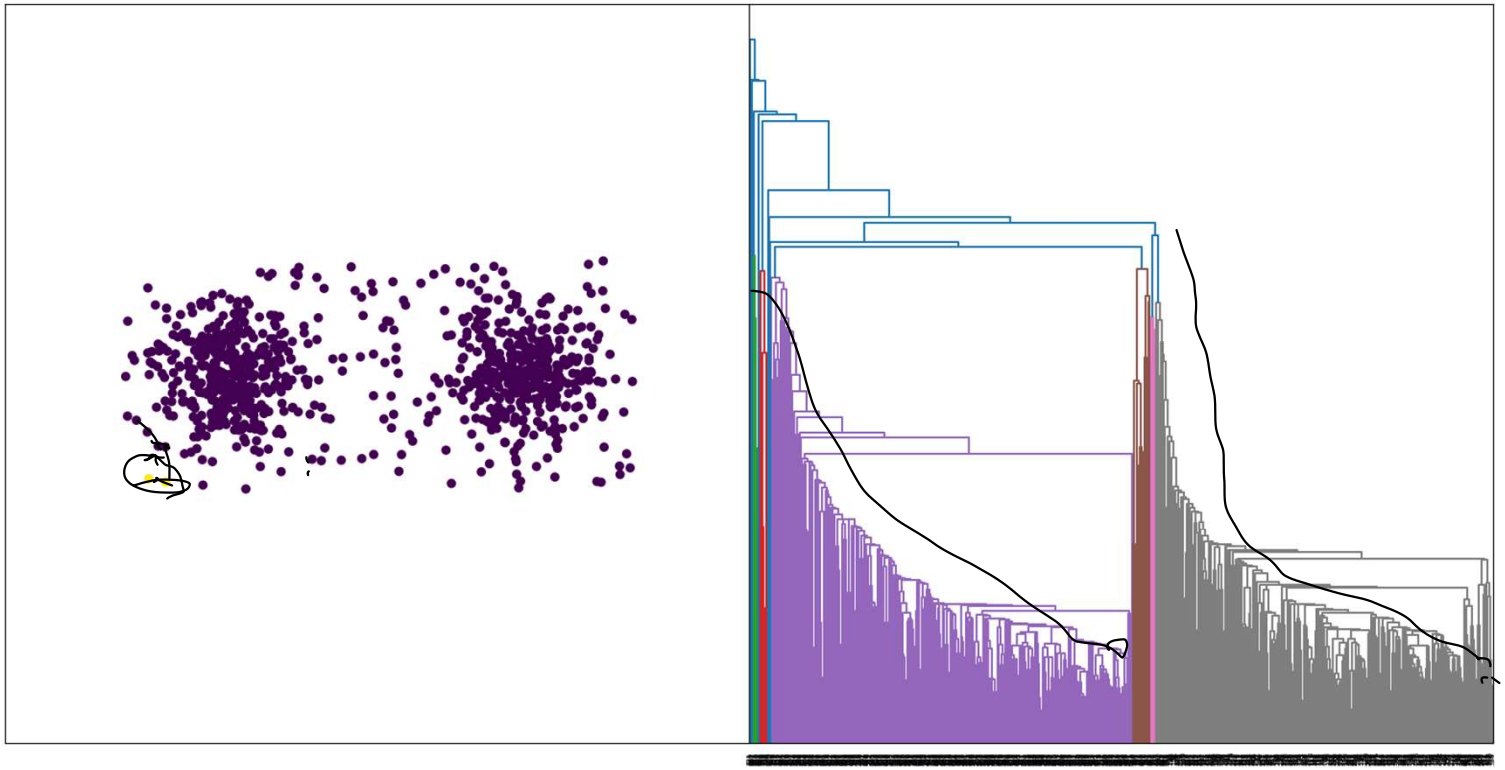
Mix of two equally sized gaussian distributions *with noise*



Single Linkage

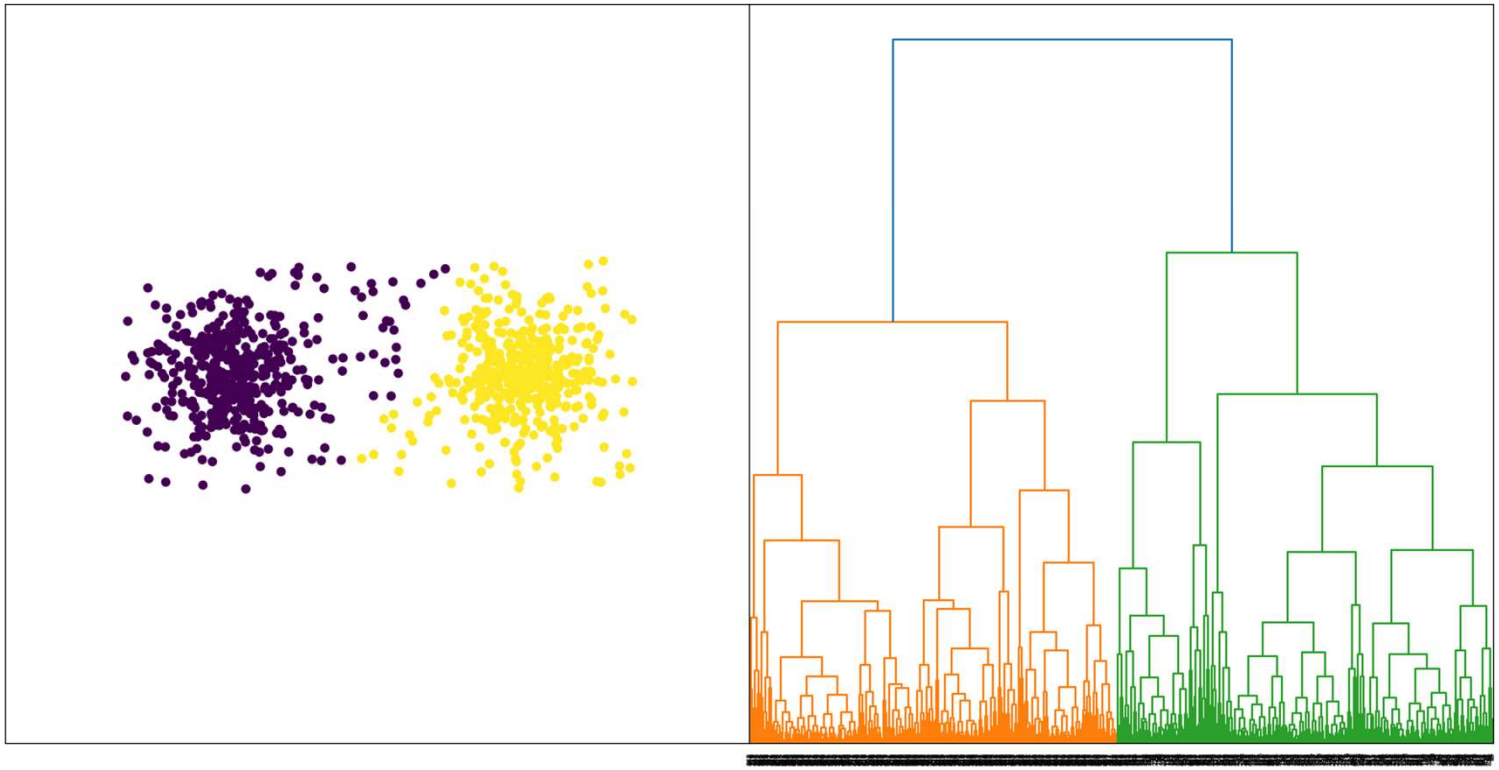
(very sensitive to noise)

; nearest link can be dominated by noise.

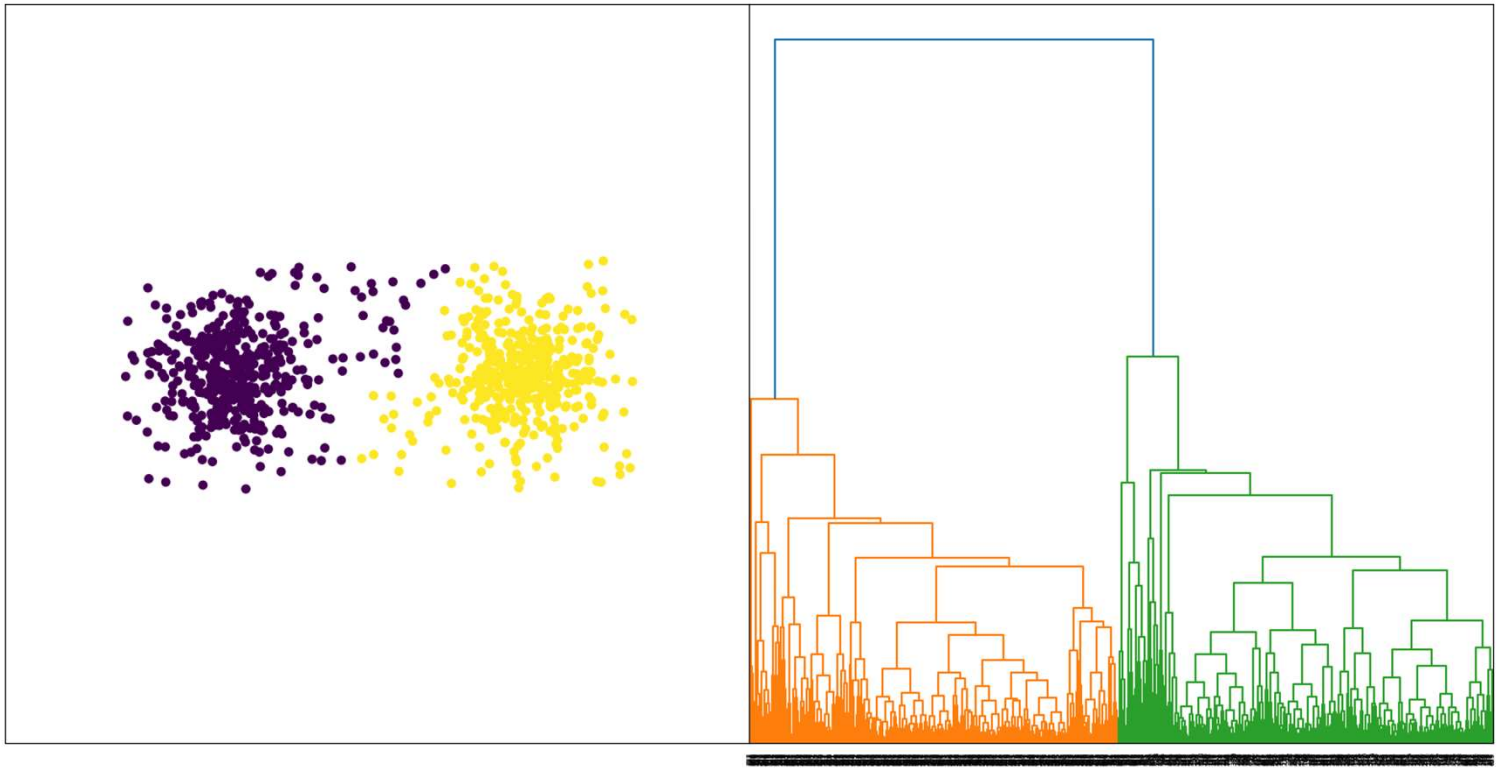


weird results

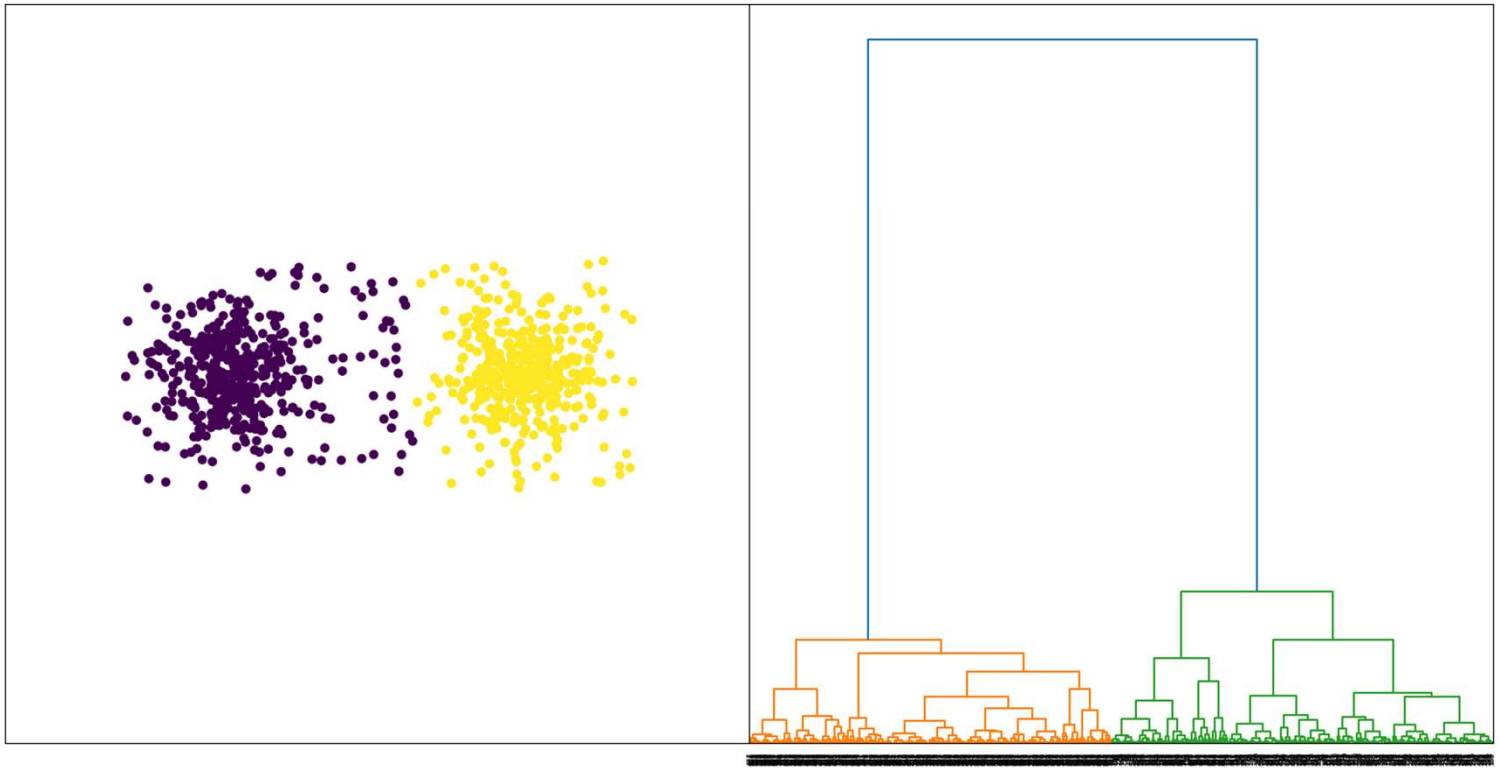
Complete Linkage



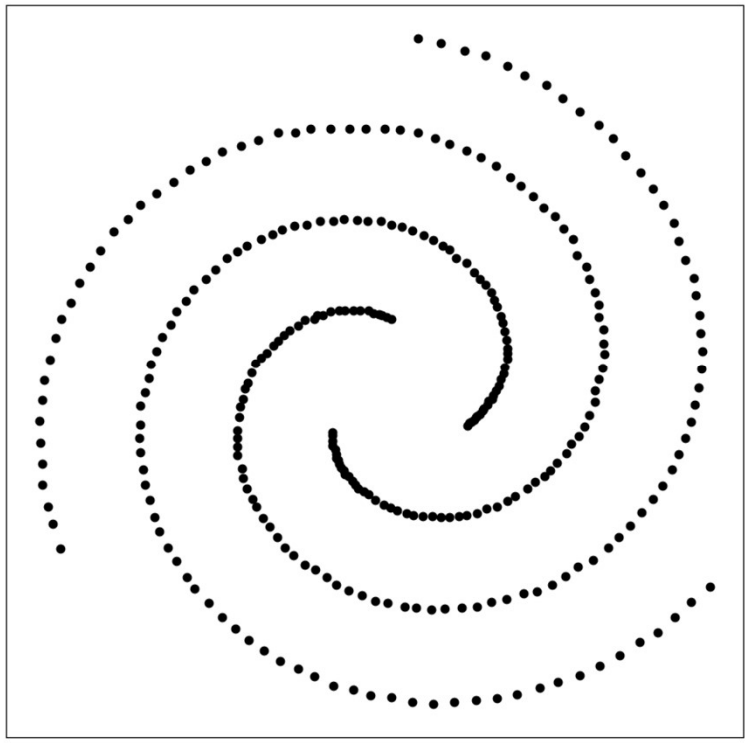
Average Linkage



Ward's Linkage

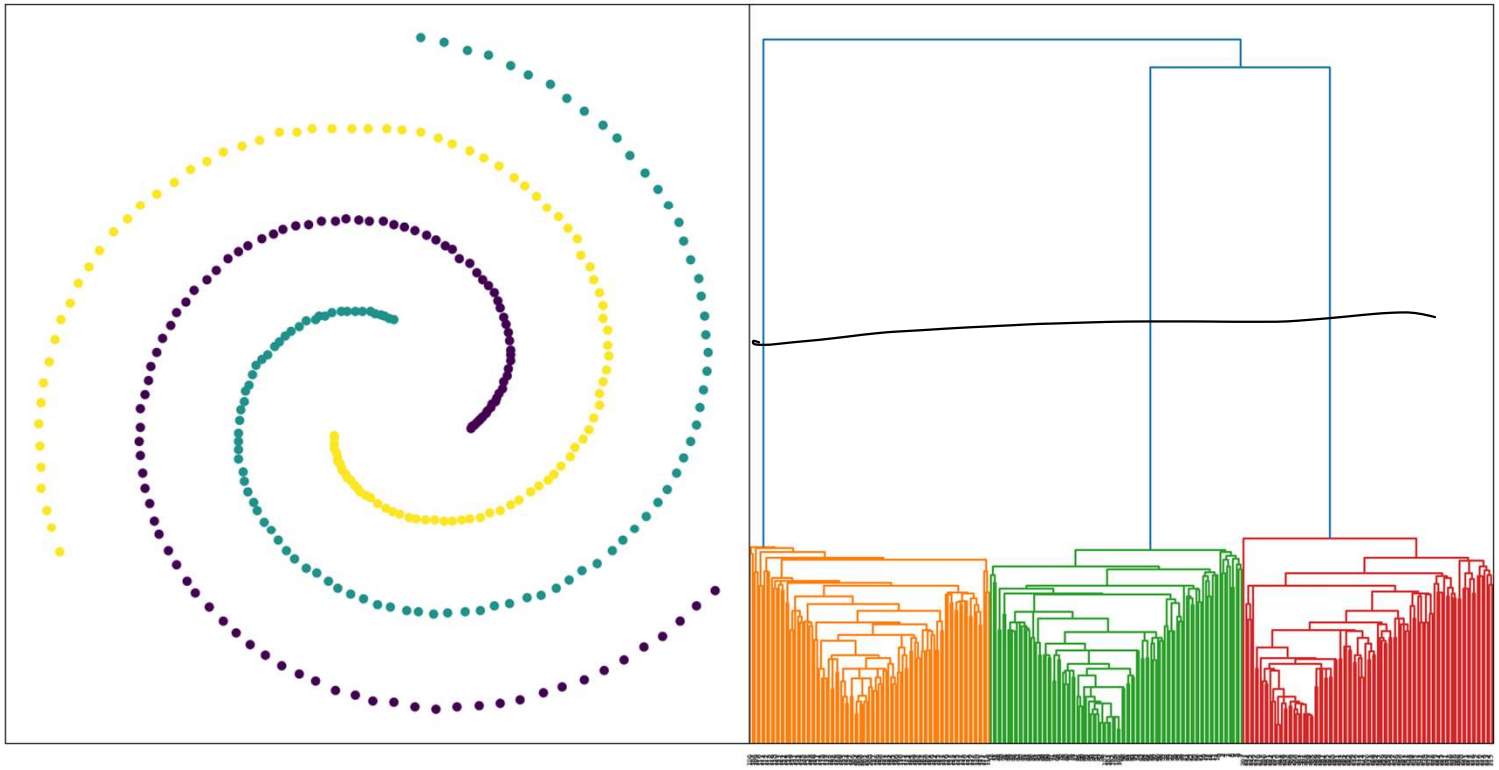


Weird-shaped clusters



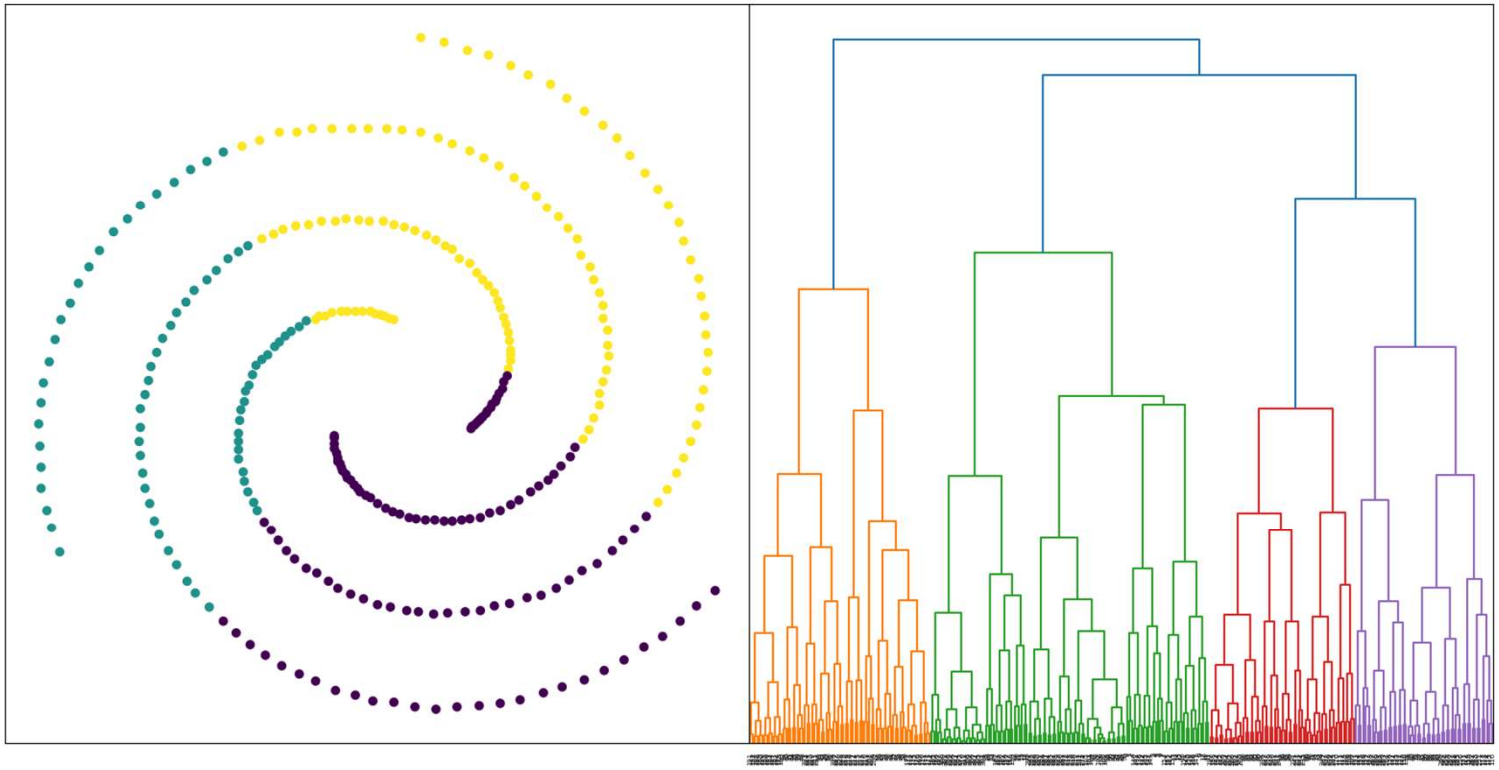
Single Linkage

works well

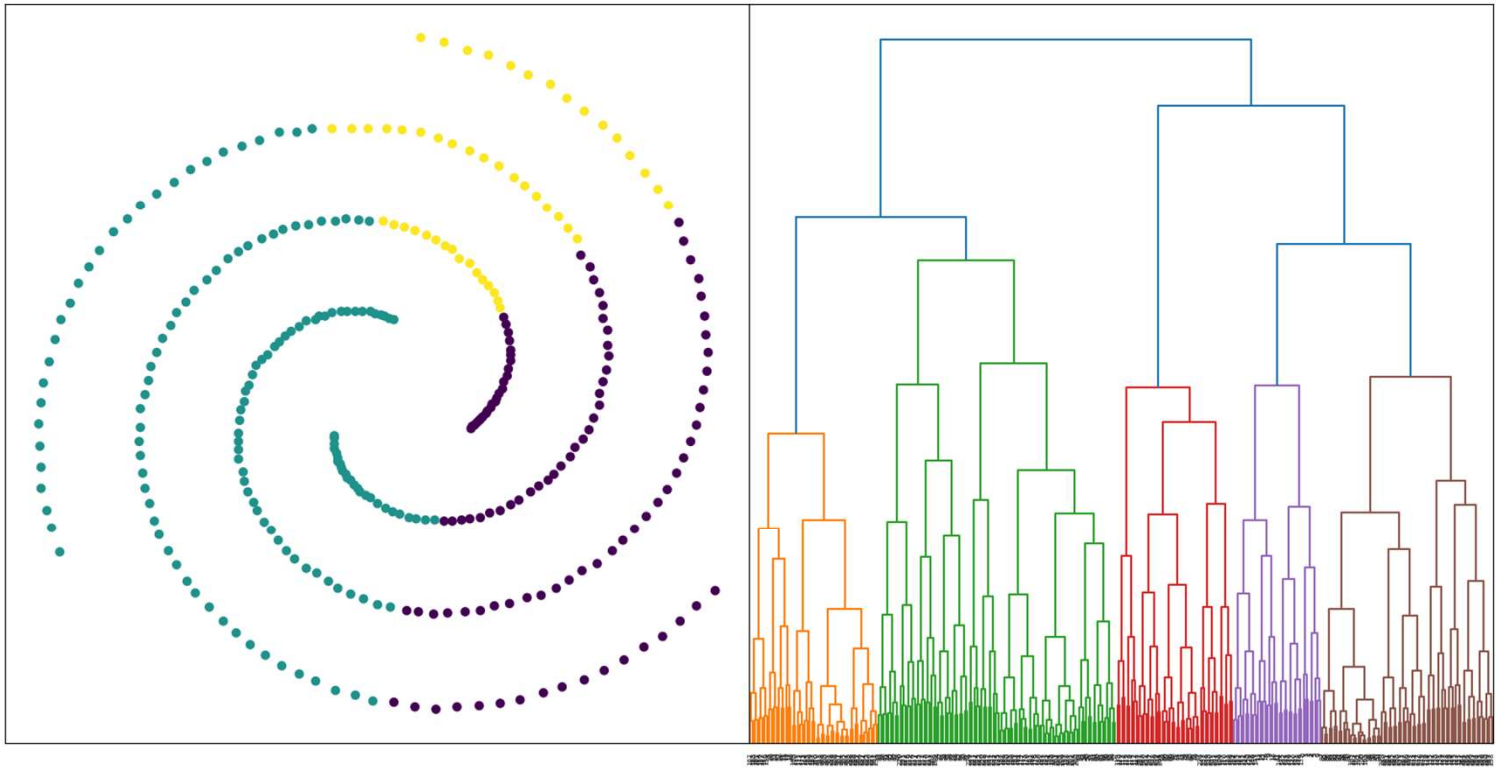


Complete Linkage

WRONG

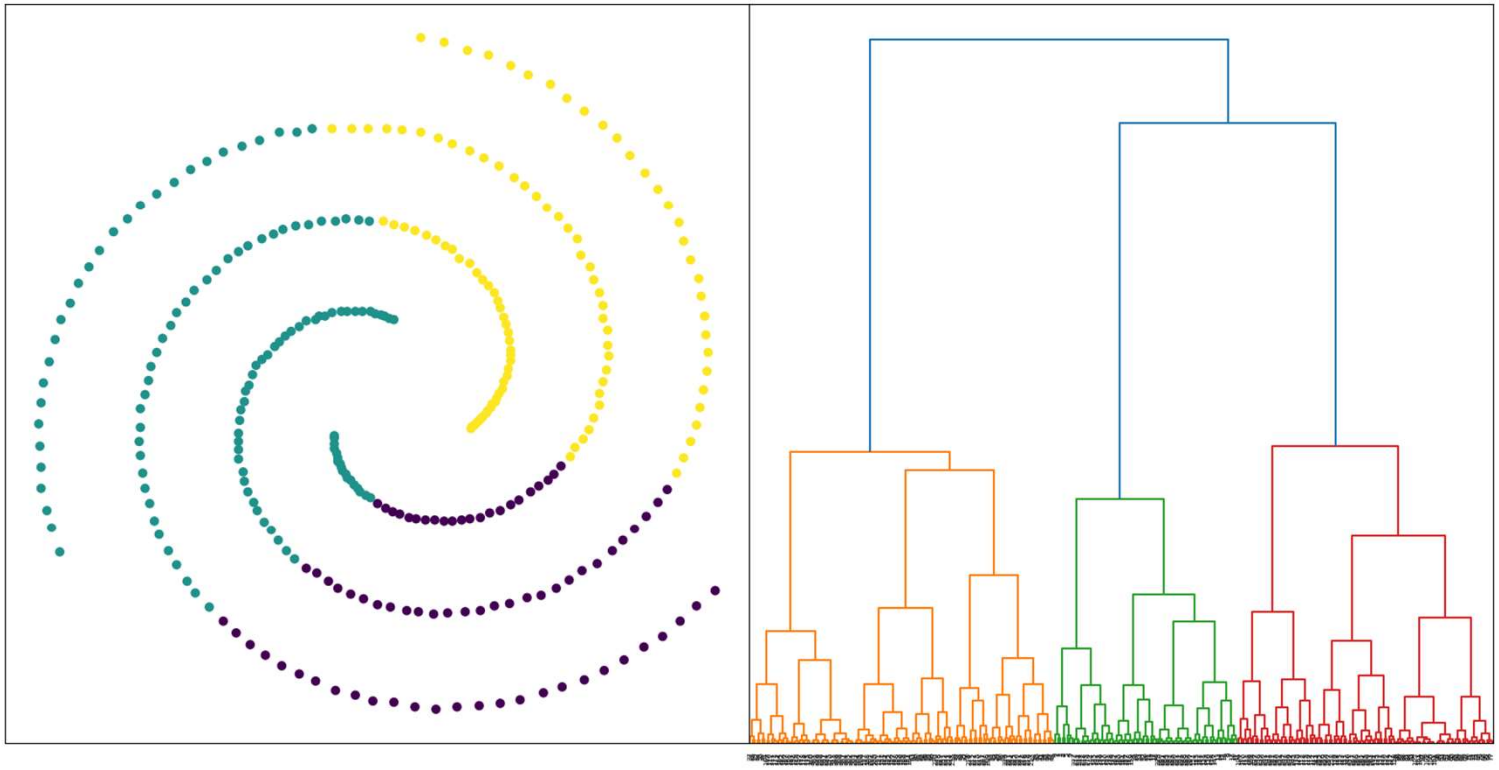


Average Linkage



Ward's Linkage

WRONG



How to choose the method?

- If there's information about similar data, use it!
- Test different methods (Usually ward's and single linkage) and check consistency.
- If your data is noisy, distrust single linkage.

Ward's 4 Dog. assume globular clusters so they fail for simple cases & single linkage works well for these weird cases

Hierarchical clustering: summary (I)

- Agglomerative hierarchical clustering is the most widely employed hierarchical clustering variant.
- At each step, merge the two clusters at lower distance.
- Start considering each data point as a single cluster.
- The variants (Single Linkage, Complete Linkage, Average Linkage, Ward's method…) between hierarchical methods are determined by the way in which the distances among data points are transformed in distances among clusters.
- Different methods are well suited for different situations