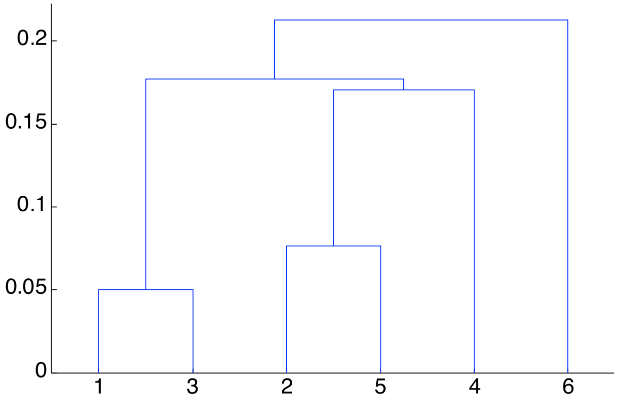
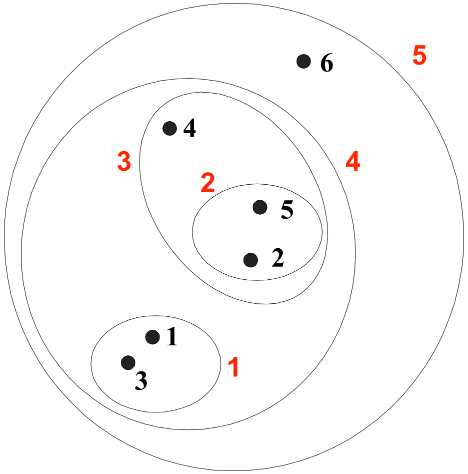
Hierarchical Clustering

Hierarchical Clustering

- Produces a set of **nested clusters** organized as a hierarchical tree
- Can be visualized as a **dendrogram**
  - A tree-like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- No assumptions on the number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level

- Hierarchical clusterings may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs) etc

Hierarchical Clustering: Problem definition

- Given a set of points $X = \{x_1, x_2, \ldots, x_n\}$ find a sequence of **nested partitions** $P_1, P_2, \ldots, P_n$ of X, consisting of $1, 2, \ldots, n$ clusters respectively such that $\Sigma_{i=1\ldots n}\mathrm{Cost}(P_i)$ is **minimized.**

- Different definitions of $\mathrm{Cost}(P_i)$ lead to different hierarchical clustering algorithms
  - $\mathrm{Cost}(P_i)$ can be formalized as the cost of any partition-based clustering

Hierarchical Clustering Algorithms

- 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) 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

Complexity of hierarchical clustering

- Distance matrix is used for deciding which clusters to merge/split
- At least quadratic in the number of data points
- Not usable for large datasets

Agglomerative clustering algorithm

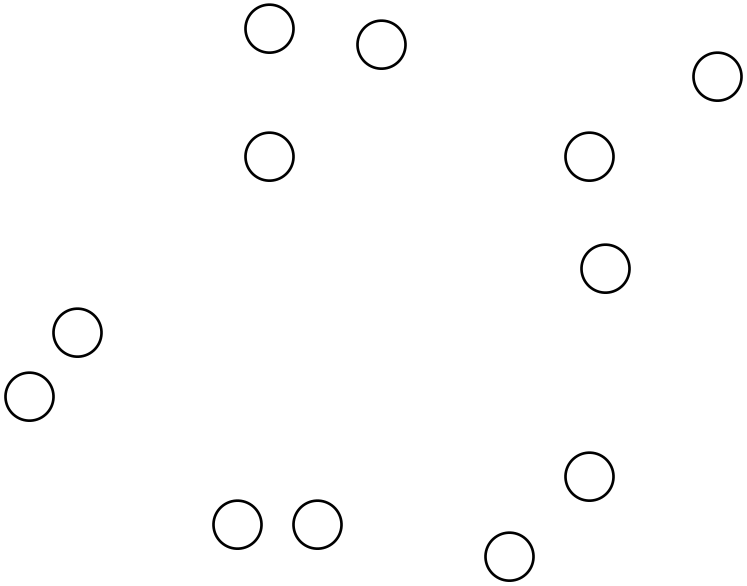
- Most popular hierarchical clustering technique
- Basic algorithm

```
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

Input/ Initial setting

- Start with clusters of individual points and a distance/proximity matrix

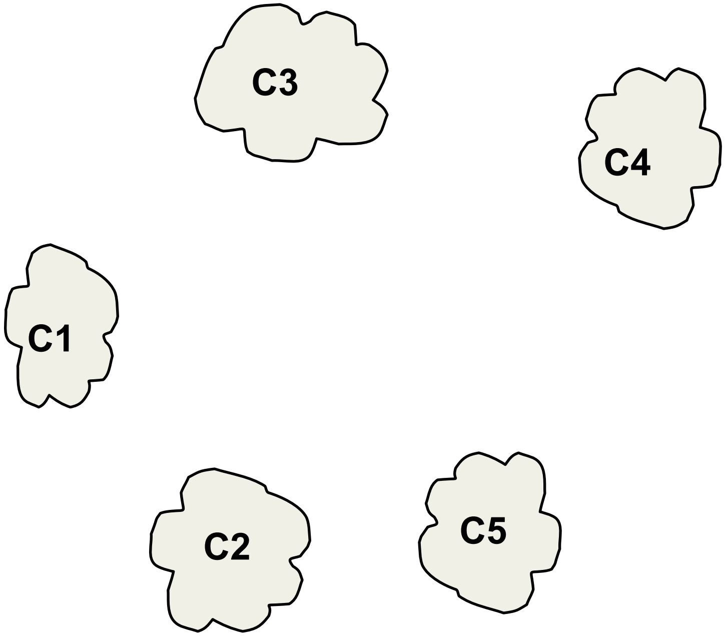


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
. . .						

Distance/Proximity Matrix

Intermediate State

- After some merging steps, we have some clusters

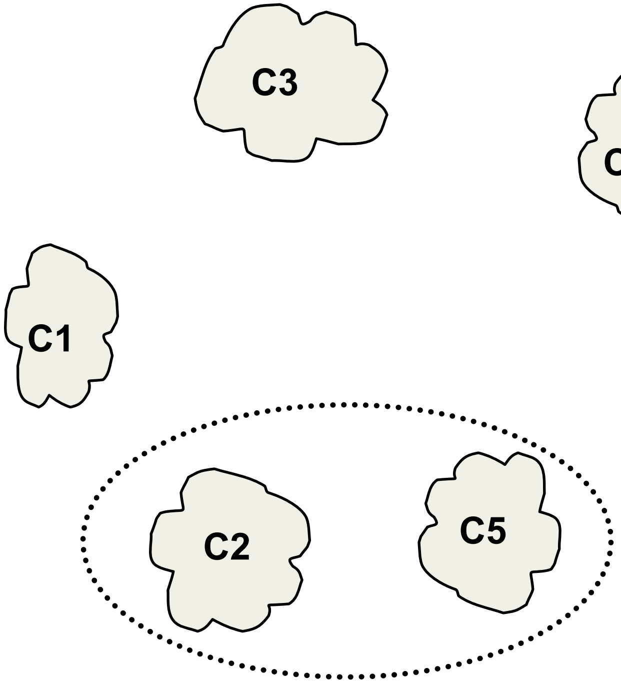


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix

Intermediate State

- Merge the two closest clusters (C2 and C5) and update the distance matrix.

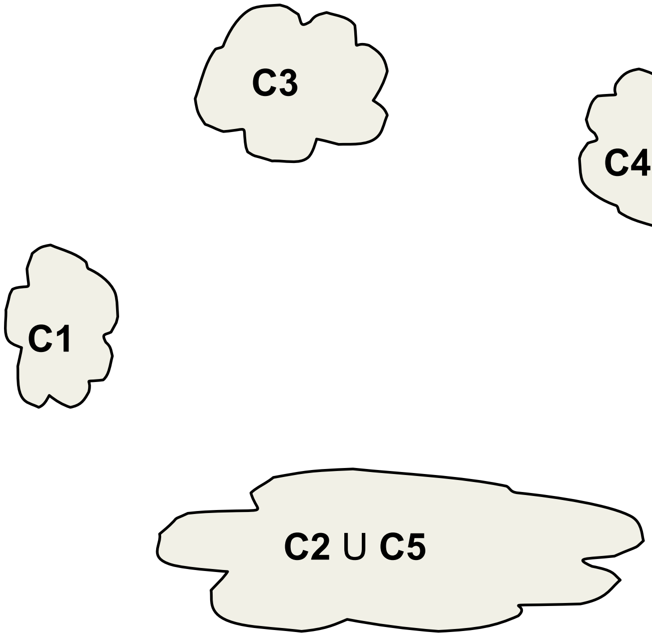


	C1	C2	C3	C4	C5
C1		▨			▨
C2	▨	▨	▨	▨	▨
C3		▨			▨
C4		▨			▨
C5	▨	▨	▨	▨	▨

Distance/Proximity Matrix

After Merging

- “How do we update the distance matrix?”



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Distance between two clusters

- Each cluster is a set of points
- How do we define distance between two sets of points
  - Lots of alternatives
  - Not an easy task

Distance between two clusters

- **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

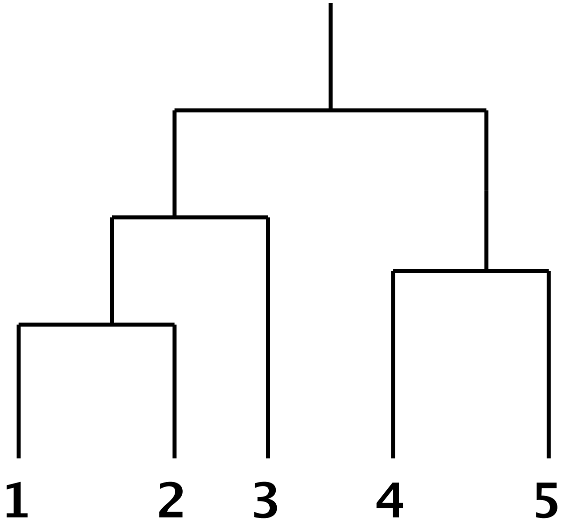
- The distance is **defined by the two most similar objects**

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

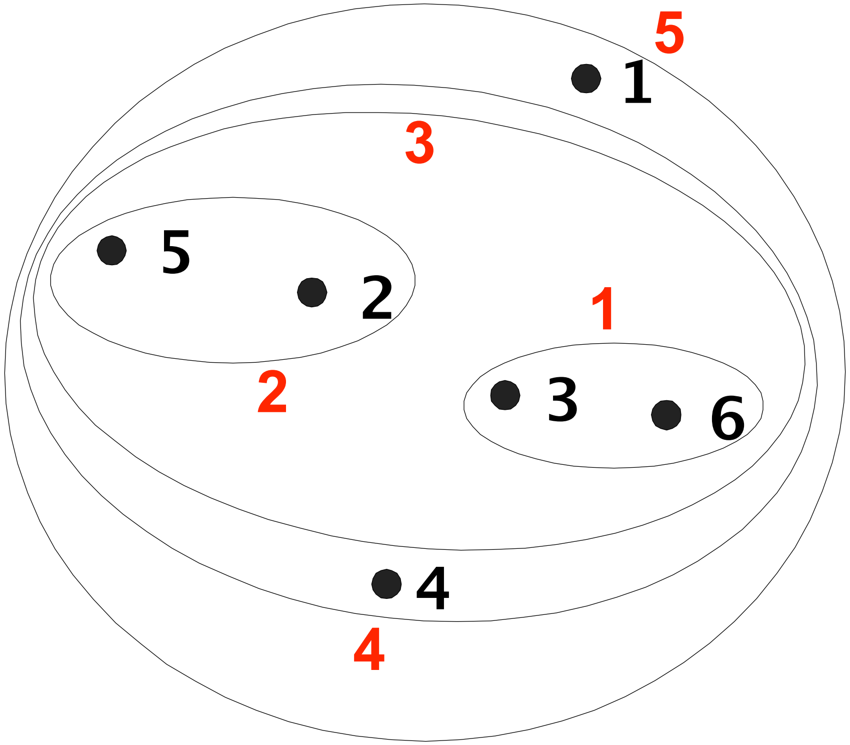
Single-link clustering: example

- Determined by one pair of points, i.e., by one link in the proximity graph.

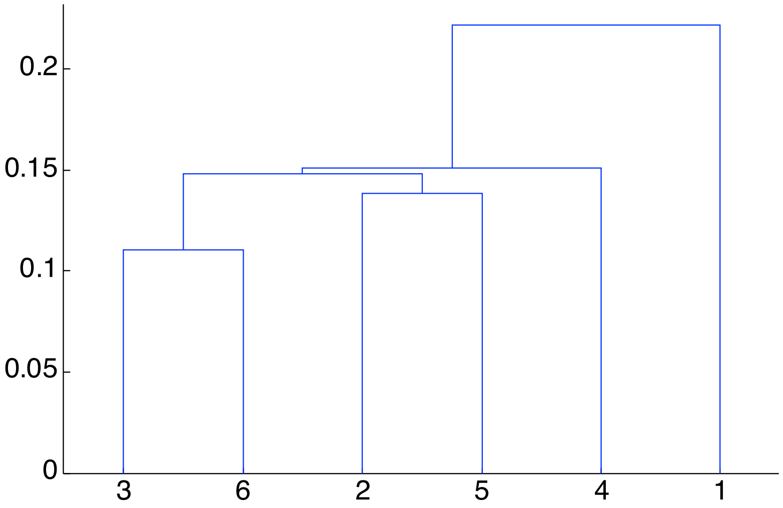
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Single-link clustering: example

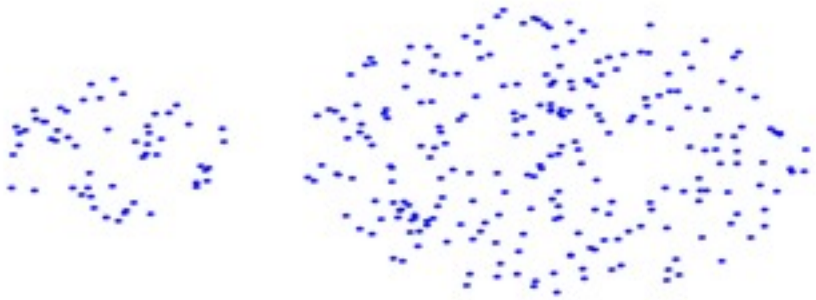


Nested Clusters

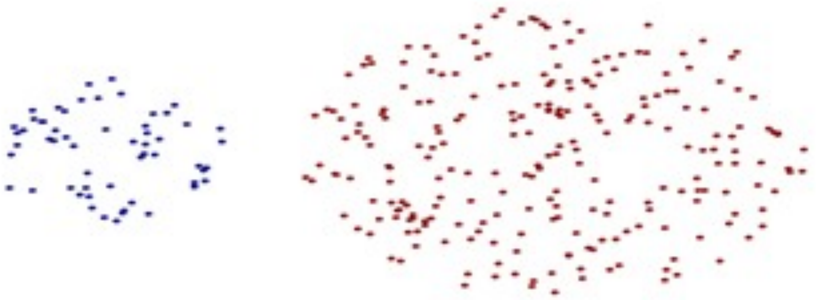


Dendrogram

Strengths of single-link clustering



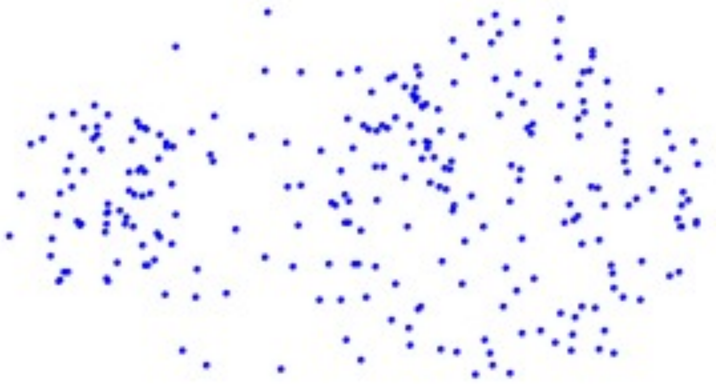
Original Points



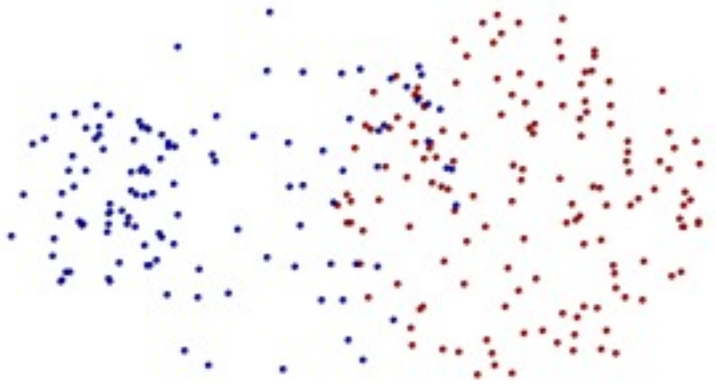
Two Clusters

- **Can handle non-elliptical shapes**

Limitations of single-link clustering



Original Points



Two Clusters

- **Sensitive to noise and outliers**
- **It produces long, elongated clusters**

Distance between two clusters

- **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

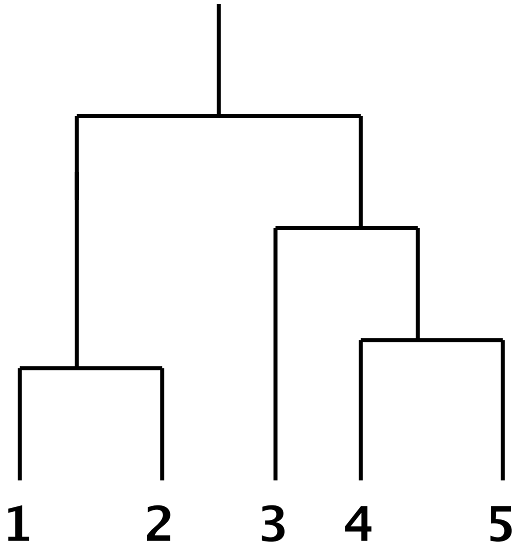
- The distance is **defined by the two most dissimilar objects**

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

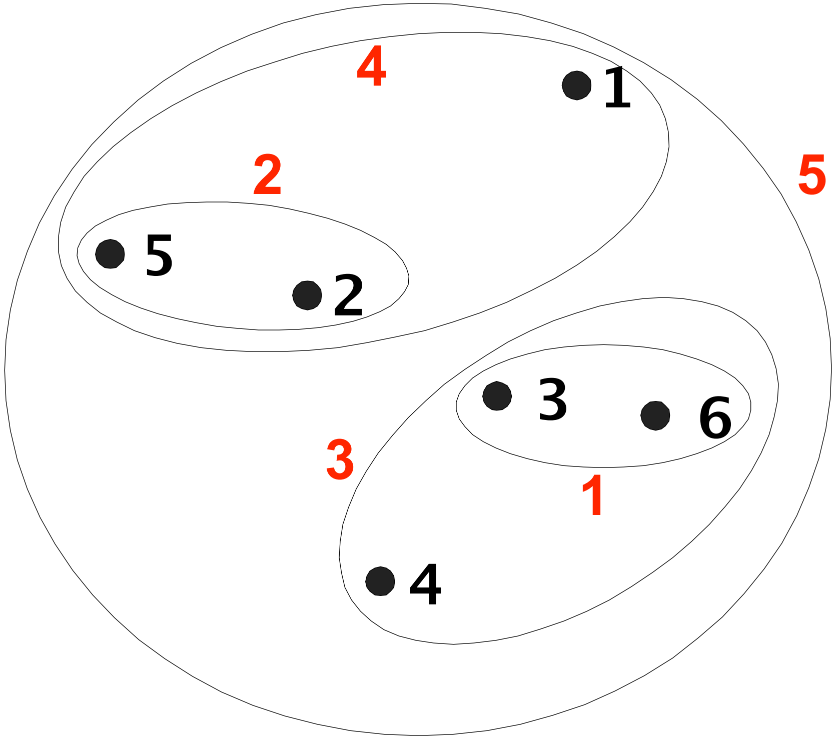
Complete-link clustering: example

- Distance between clusters is determined by the two most distant points in the different clusters

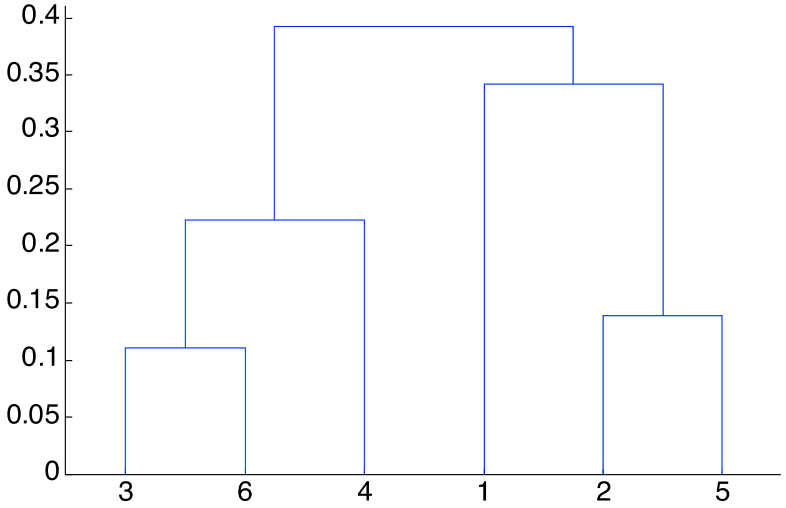
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Complete-link clustering: example

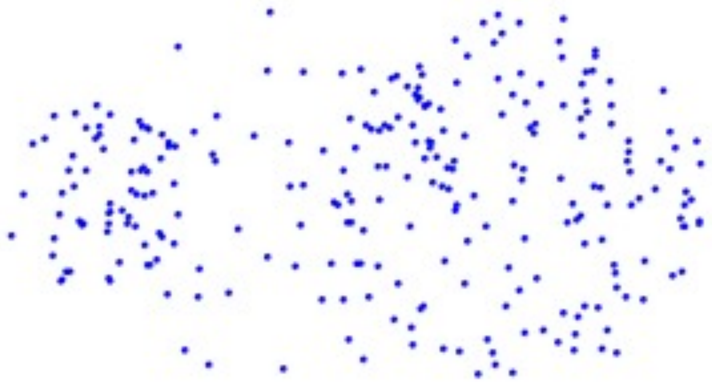


Nested Clusters

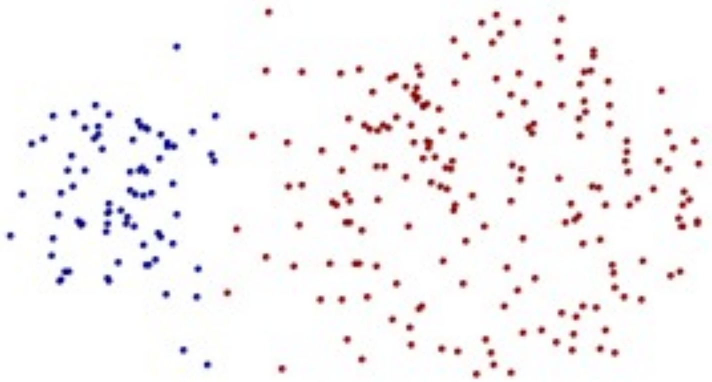


Dendrogram

Strengths of complete-link clustering



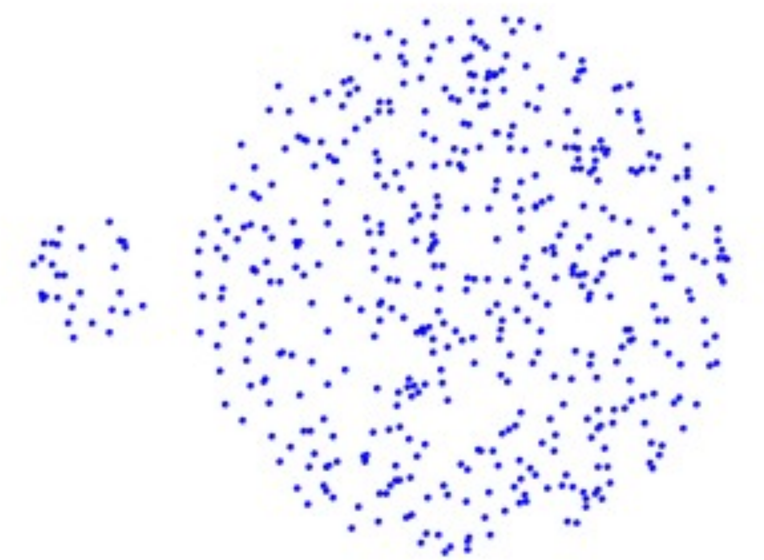
Original Points



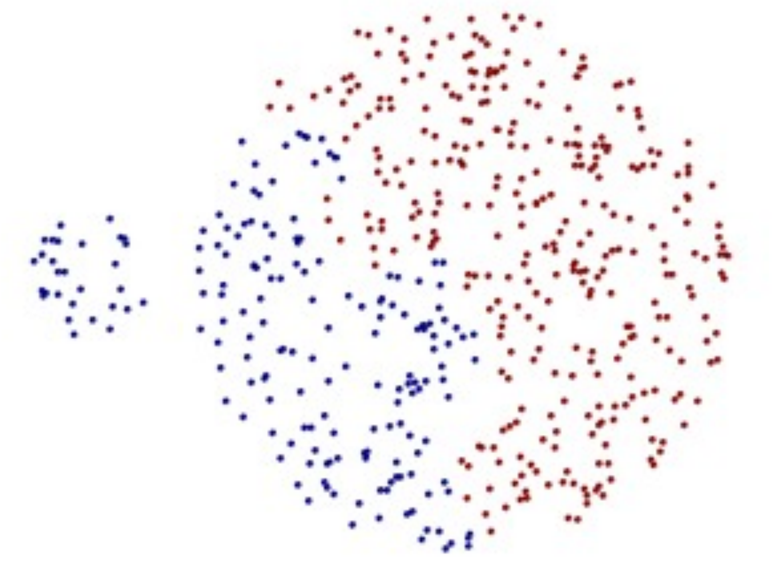
Two Clusters

- More balanced clusters (with equal diameter)
- Less susceptible to noise

Limitations of complete-link clustering



Original Points



Two Clusters

- **Tends to break large clusters**
- **All clusters tend to have the same diameter – small clusters are merged with larger ones**

Distance between two clusters

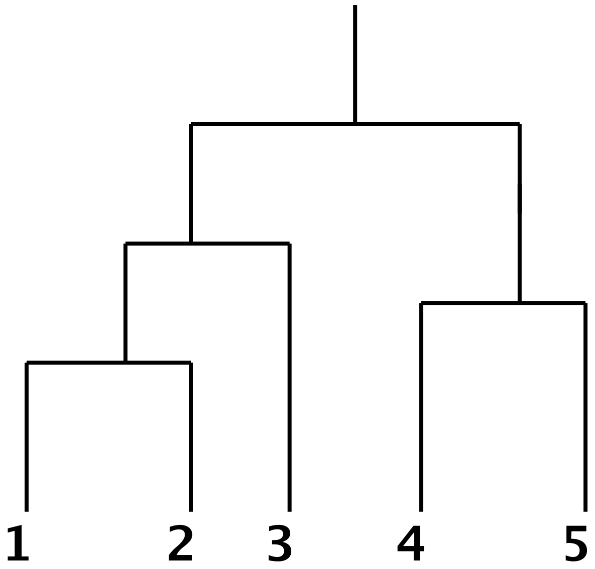
- **Group average distance** between clusters C_i and C_j is the **average distance** between any object in C_i and any object in C_j

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

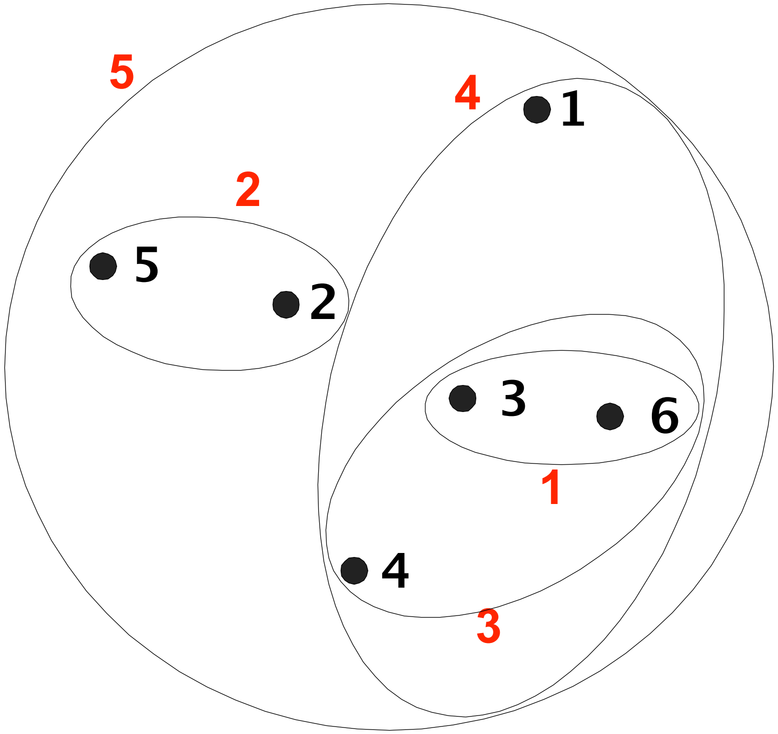
Average-link clustering: example

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

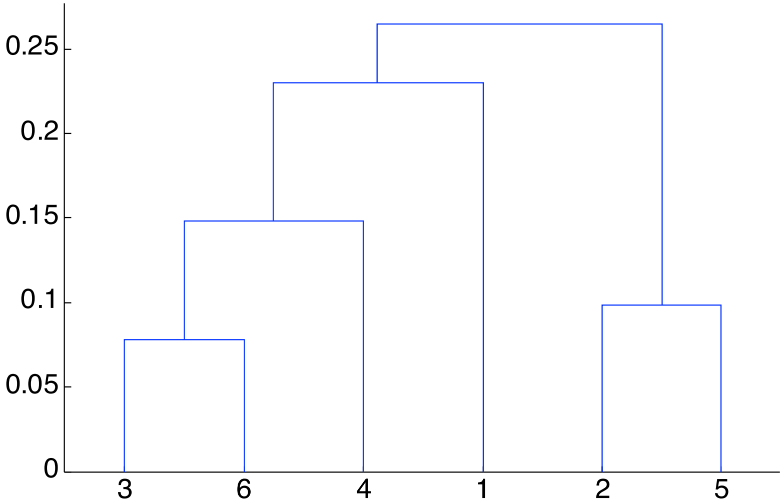
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Average-link clustering: example



Nested Clusters



Dendrogram

Average-link clustering: discussion

- Compromise between Single and Complete Link

- Strengths
  - Less susceptible to noise and outliers

- Limitations
  - Biased towards globular clusters

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

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

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}

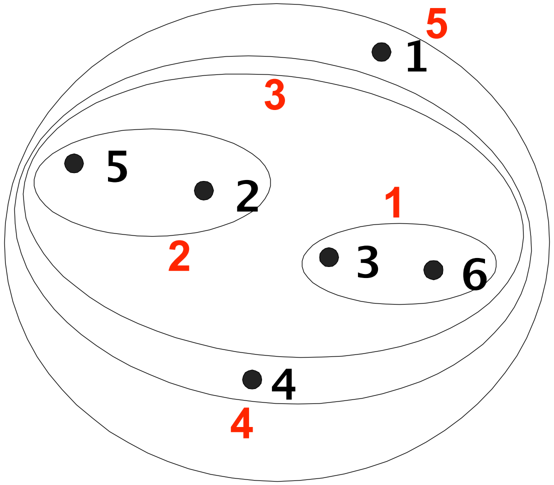
$$D_W(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

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

Ward’s distance for clusters

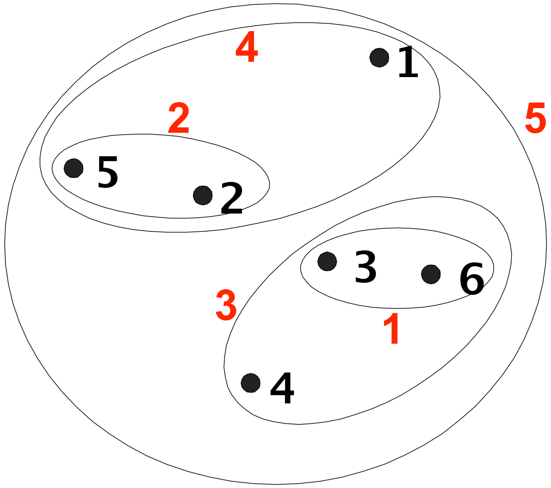
- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
  - Can be used to initialize k-means

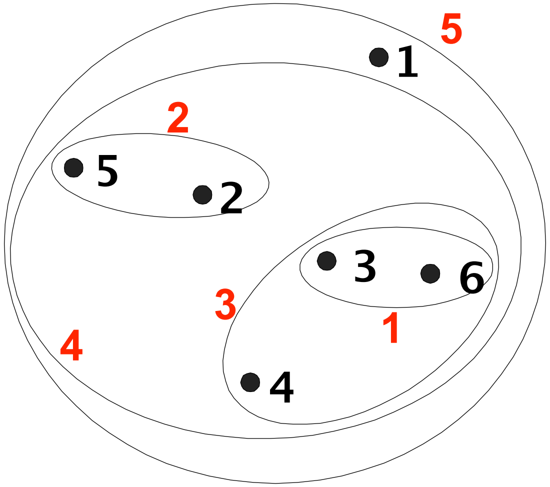
Hierarchical Clustering: Comparison



MIN

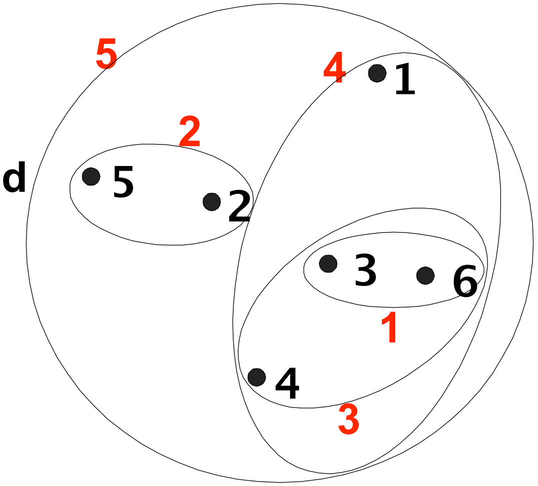
MAX





Group Average

Ward's Method



Hierarchical Clustering: Time and Space requirements

- For a dataset **X** consisting of **n** points
- **O(n^2) space**; it requires storing the distance matrix
- **O(n^3) time** in most of the cases
  - There are **n** steps and at each step the size **n^2** distance matrix must be updated and searched
  - Complexity can be reduced to **O(n^2 log(n))** time for some approaches by using appropriate data structures

Divisive hierarchical clustering

- Start with a single cluster composed of all data points
- Split this into components
- Continue recursively
- Computationally intensive, less widely used than agglomerative methods