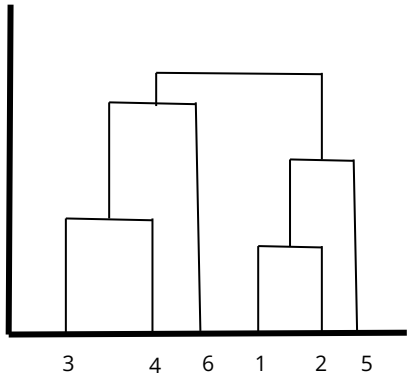
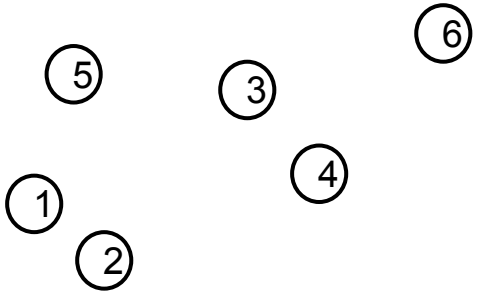
Hierarchical Clustering

Boston University CS 506 - Lance Galletti

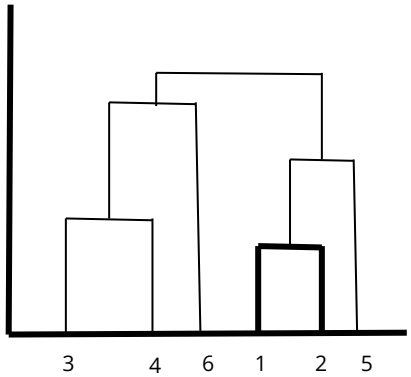
Hierarchical Clustering

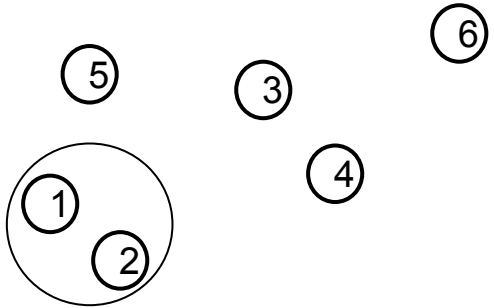




Hierarchical Clustering

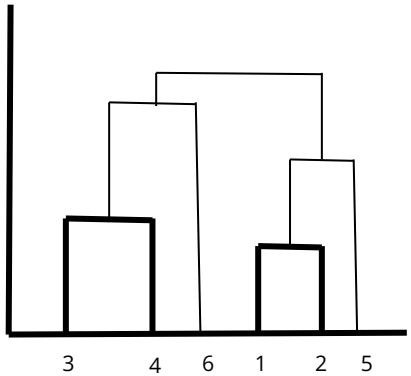
At every step, we record which clusters were merged in order to produce a dendrogram:

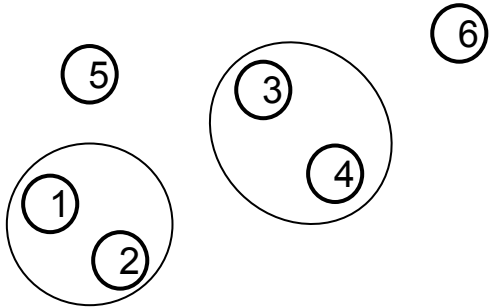




Hierarchical Clustering

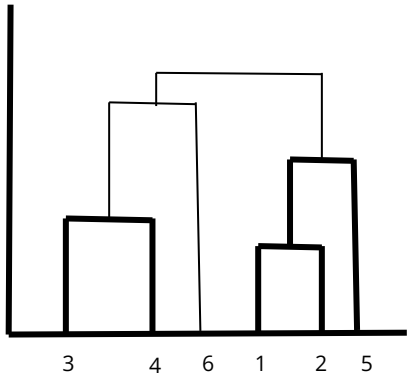
At every step, we record which clusters were merged in order to produce a dendrogram:

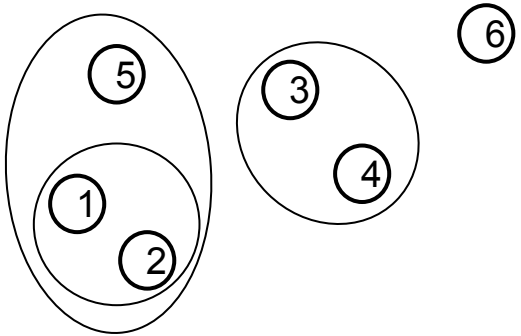




Hierarchical Clustering

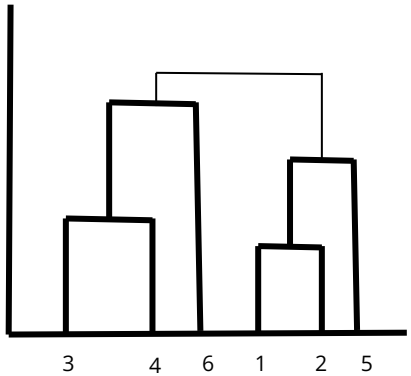
At every step, we record which clusters were merged in order to produce a dendrogram:

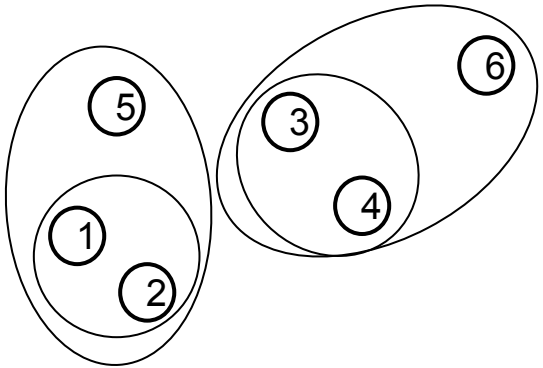




Hierarchical Clustering

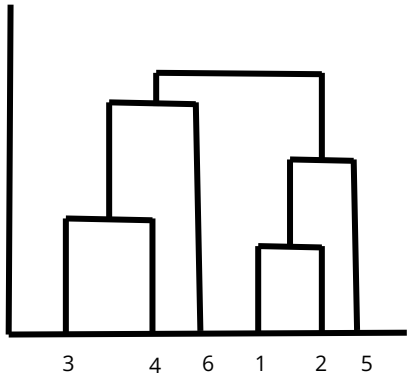
At every step, we record which clusters were merged in order to produce a dendrogram:

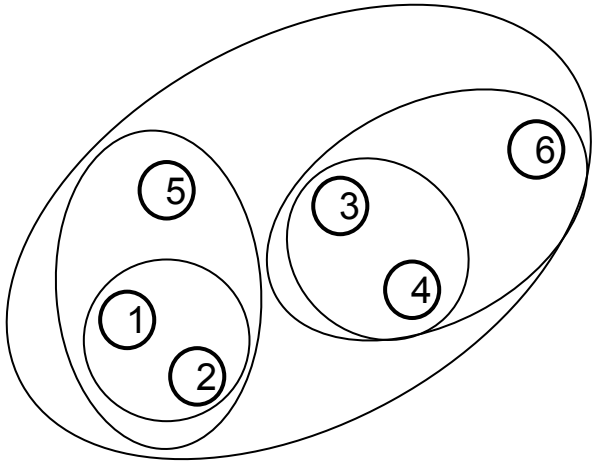




Hierarchical Clustering

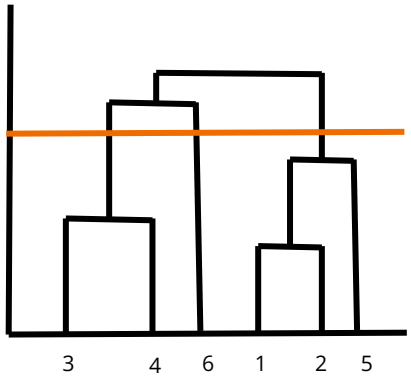
At every step, we record which clusters were merged in order to produce a dendrogram:

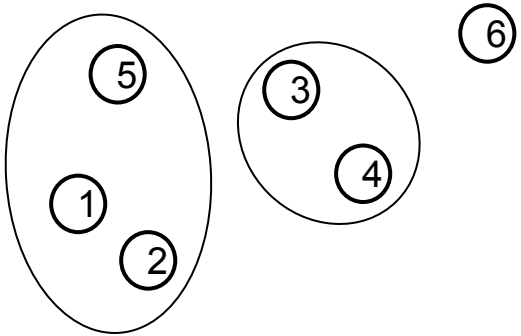




Hierarchical Clustering

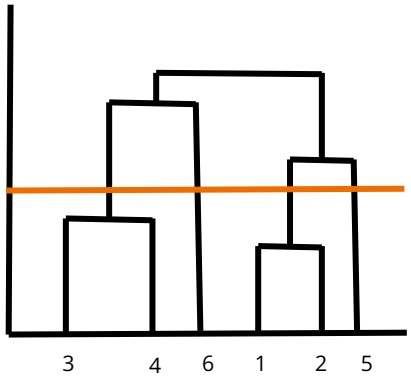
We can “cut” the dendrogram at any threshold to produce any number of clusters

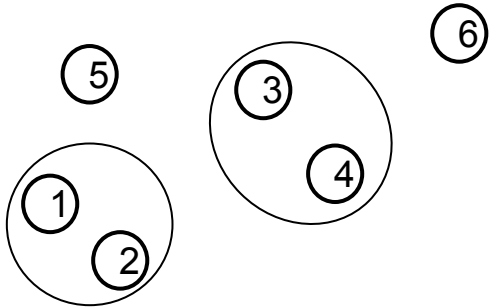




Hierarchical Clustering

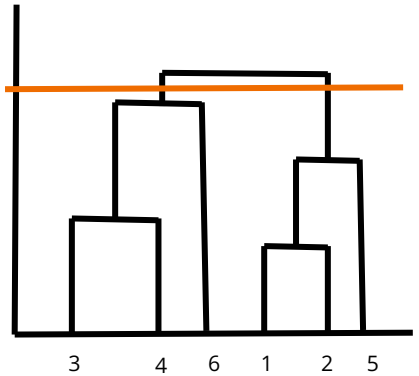
We can “cut” the dendrogram at any threshold to produce any number of clusters

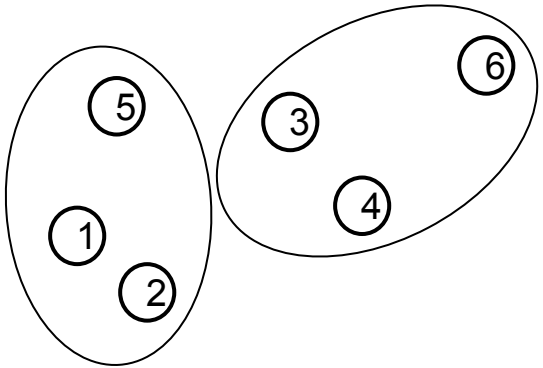




Hierarchical Clustering

We can “cut” the dendrogram at any threshold to produce any number of clusters





Example

https://www.catalogueoflife.org/

Hierarchical Clustering

Two types of hierarchical clustering:

Agglomerative:

1. Start with every point in its own cluster
2. At each step, merge the two closest clusters
3. Stop when every point is in the same cluster

Divisive:

1. Start with every point in the same cluster
2. At each step, split until every point is in its own cluster

Agglomerative Clustering Algorithm

1. Let each point in the dataset be in its own cluster
2. Compute the distance between all pairs of clusters
3. Merge the two closest clusters
4. Repeat 3 & 4 until all points are in the same cluster

Hierarchical Clustering

Can we implement this? Are we missing anything?

How would you define distance between clusters?

you could take average distance between all points from cluster A to cluster B

you could take the distance between the centroids of both clusters

Hierarchical Clustering - Distance Functions

Let's first define:

Distance between points: $\mathbf{d(p_1, p_2)}$

Distance between clusters: $\mathbf{D(C_1, C_2)}$

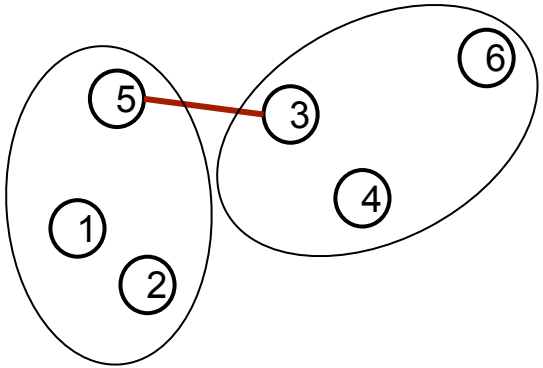
d is distance between points
D is distance between clusters

Single-Link Distance

Is the **minimum** of all pairwise distances between a point from one cluster and a point from the other cluster.

$$D_{SL}(C_1, C_2) = \min \{d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\}$$

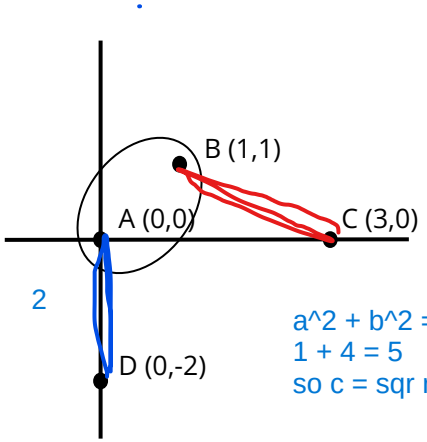
AKA distance between the closest two points of each cluster



Depends on choice of **d**

Q: Single-Link Distance

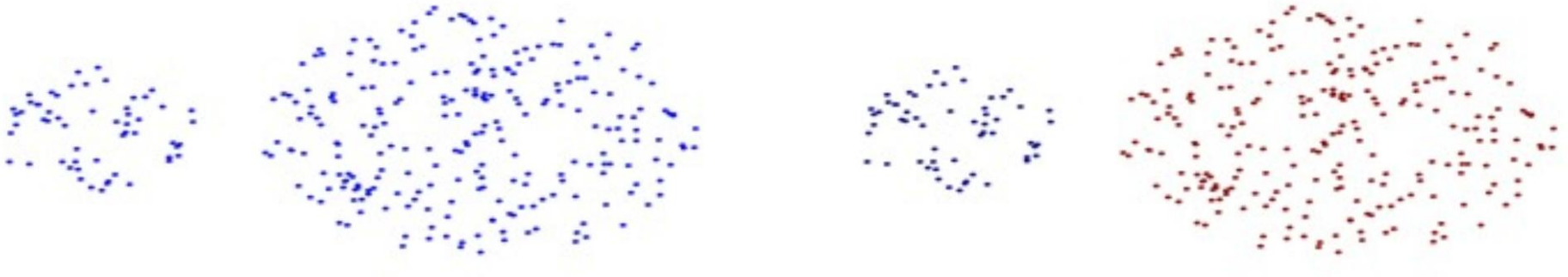
Is C or D closer to {A, B} ?



a^2 + b^2 = c^2
1 + 4 = 5
so c = sqr root of 5

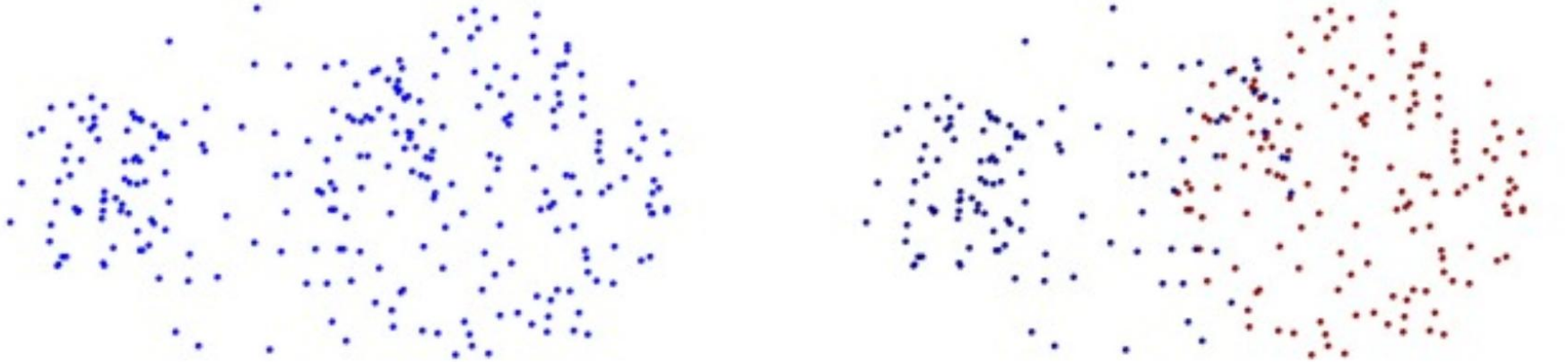
Since 2^2 = 4 < 5
we would merge D into cluster {A,B}

Single-Link Distance



Can handle clusters of different sizes

Single-Link Distance



But... Sensitive to noise points
Tends to create elongated clusters

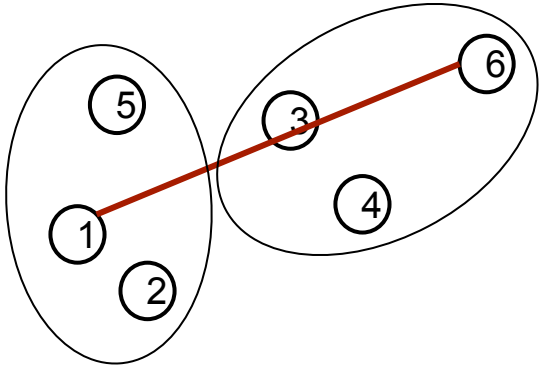
this points will be interfering with this min distance

Complete-Link Distance

Is the **maximum** of all pairwise distances between a point from one cluster and a point from the other cluster.

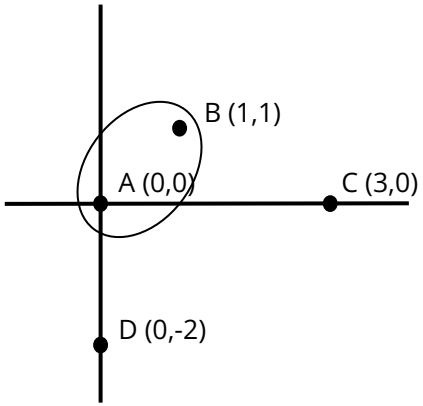
$$D_{CL}(C_1, C_2) = \max \{ d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2 \}$$

min of the max



Q: Complete-Link Distance

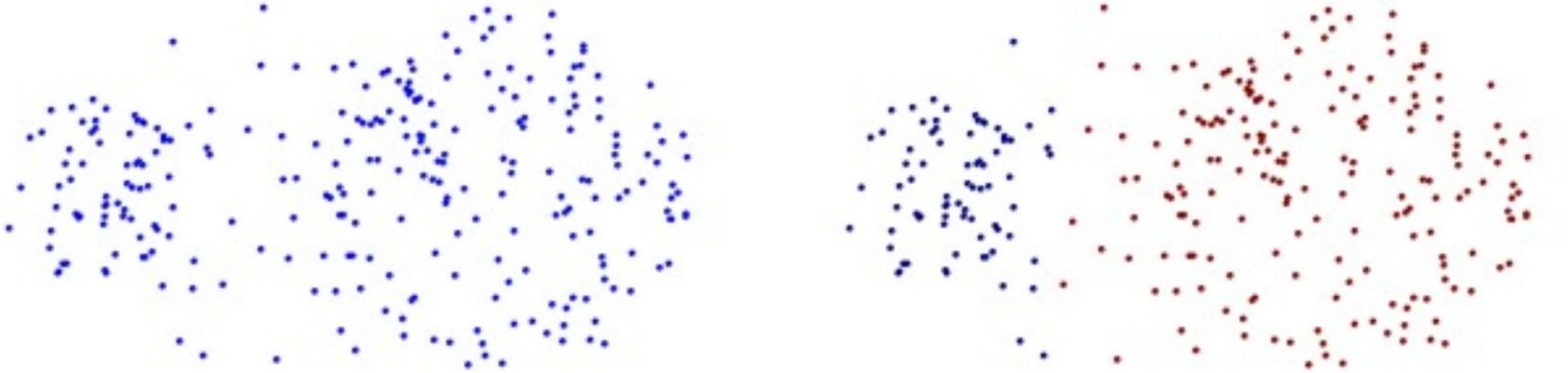
Is C or D closer to {A, B} ?



distance between C and {A,B} is 3
distance between D and {A,B} is sqr root of 10

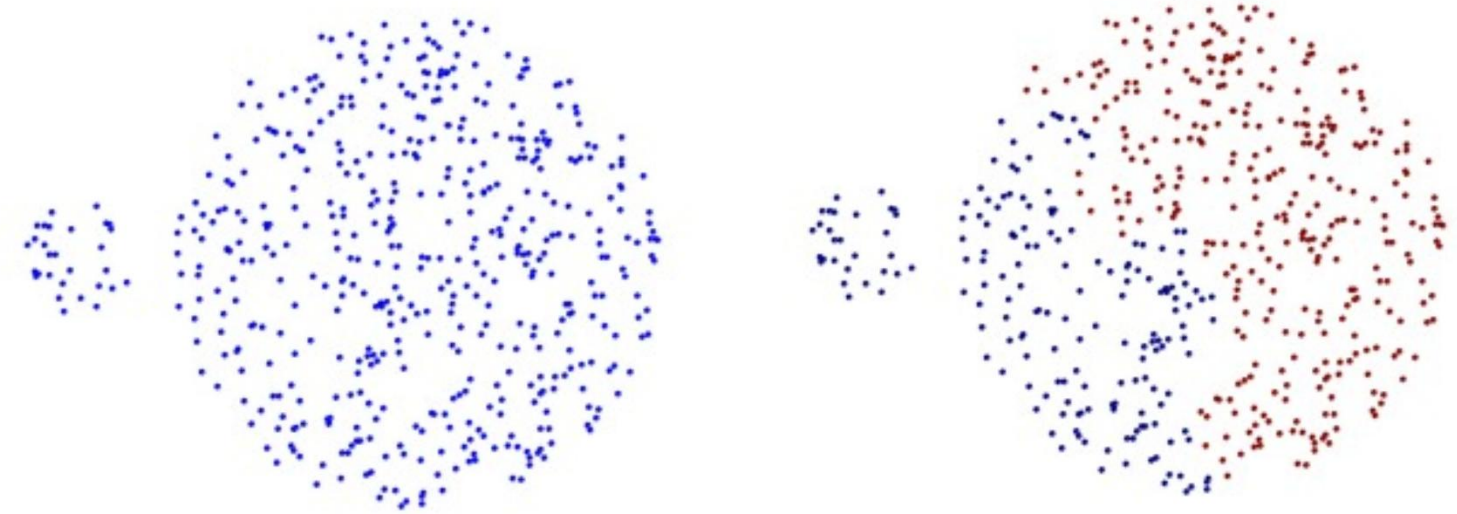
Since 3 < sqr root 10,
C gets absorbed into the cluster first

Complete-Link Distance



Less susceptible to noise
Creates more balanced (equal diameter) clusters

Complete-Link Distance



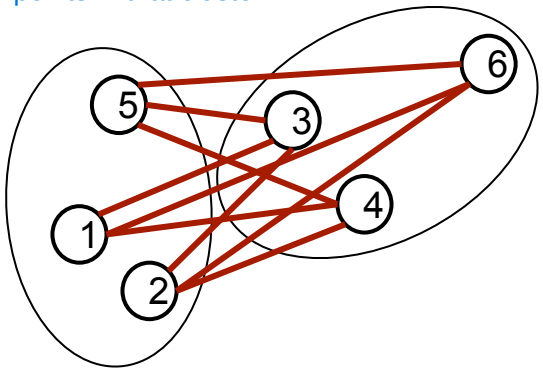
But... Tends to split up large clusters.
All clusters tend to have the same diameter

Average-Link Distance

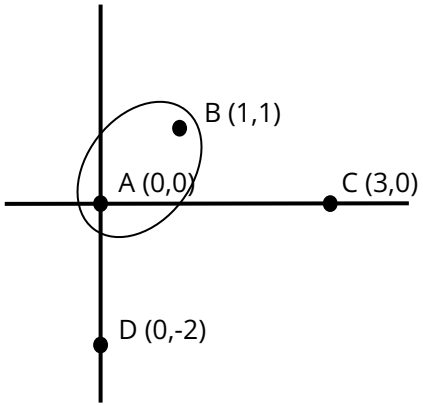
Is the **average** of all pairwise distances between a point from one cluster and a point from the other cluster.

$$D_{AL}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p_1 \in C_1, p_2 \in C_2} d(p_1, p_2)$$

| C_1 | is the cardinality of the set
AKA number of points in that cluster



Q: Average-Link Distance



Is C or D closer to {A, B} ?

Average-Link Distance

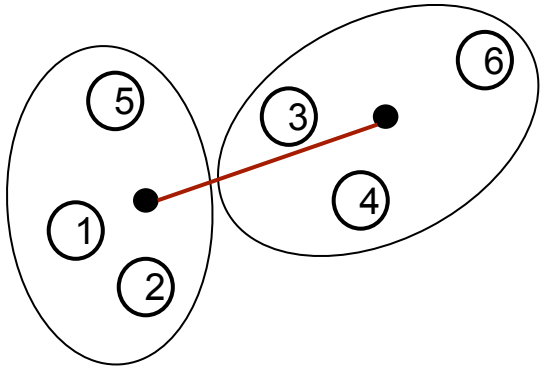
Less susceptible to noise and outliers.

But… Tends to be biased toward globular clusters

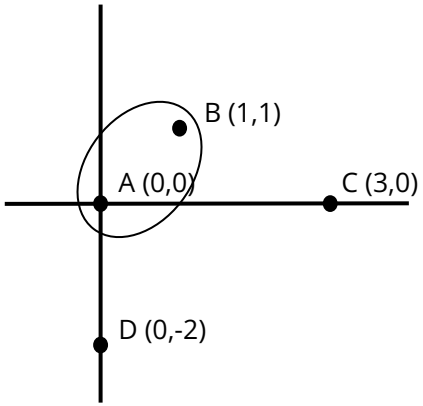
Centroid Distance

The distance between the centroids of clusters.

$$D_C(C_1, C_2) = d(\mu_1, \mu_2)$$



Q: Centroid Distance



Is C or D closer to {A, B} ?

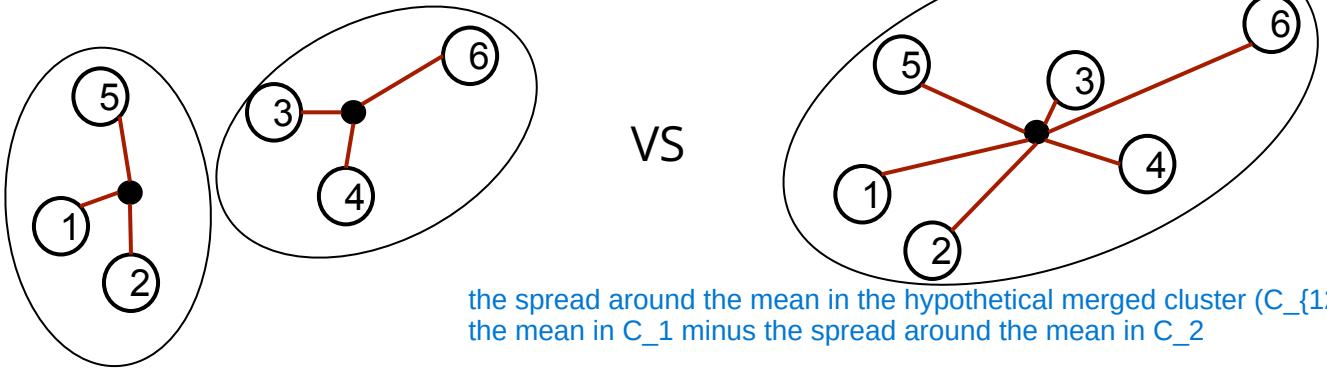
Ward's Distance

compares spread between hypothetical merged cluster vs unmerged original clusters

Is the difference between the spread / variance of points in the merged cluster and the unmerged clusters.

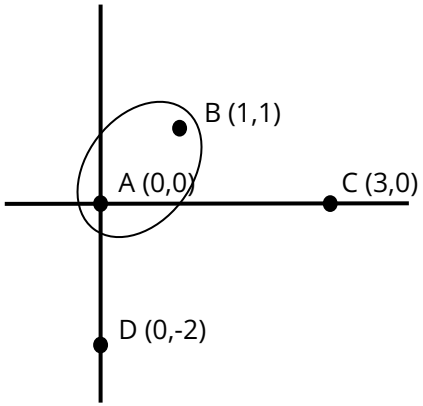
u is mean of that new cluster

$$D_{WD}(C_1, C_2) = \sum_{p \in C_{12}} d(p, \mu_{12}) - \sum_{p_1 \in C_1} d(p_1, \mu_1) - \sum_{p_2 \in C_2} d(p_2, \mu_2)$$



the spread around the mean in the hypothetical merged cluster (C_{12) minus the spread around the mean in C_1 minus the spread around the mean in C_2

Q: Ward's Distance



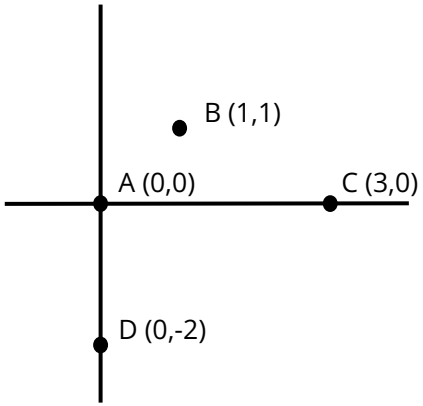
Is C or D closer to {A, B} ?

Agglomerative Clustering Algorithm

1. Let each point in the dataset be in its own cluster
2. Compute the distance between all pairs of clusters
3. Merge the two closest clusters
4. Repeat 3 & 4 until all points are in the same cluster

Example

d = Euclidean
D = Single-Link

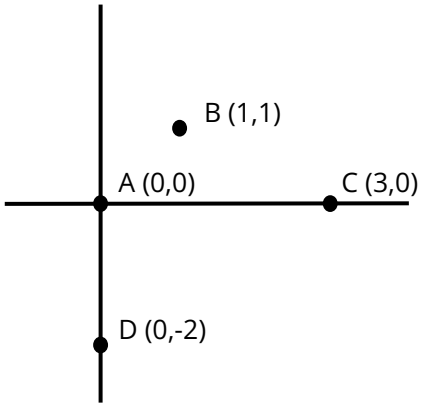


Distance Matrix

	A	B	C	D
A				
B				
C				
D				

Example

d = Euclidean
D = Single-Link

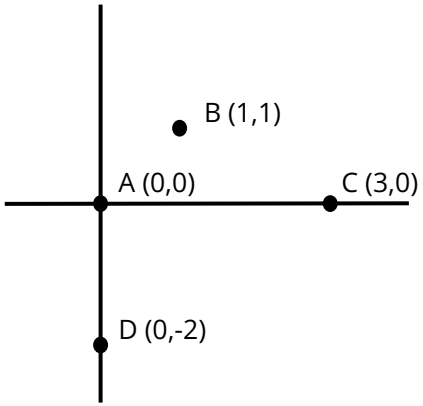


Distance Matrix

	A	B	C	D
A	0			
B		0		
C			0	
D				0

Example

d = Euclidean
D = Single-Link



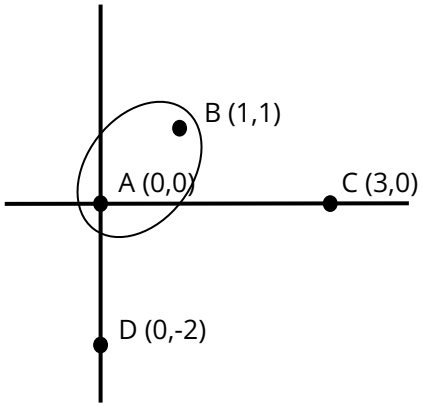
Filled out distance matrix using euclidean distance function

Distance Matrix

	A	B	C	D
A	0	√2	3	2
B	√2	0	√5	√10
C	3	√5	0	√13
D	2	√10	√13	0

Example

d = Euclidean
D = Single-Link



We merge {A,B} because they are the smallest pair of distances on the table

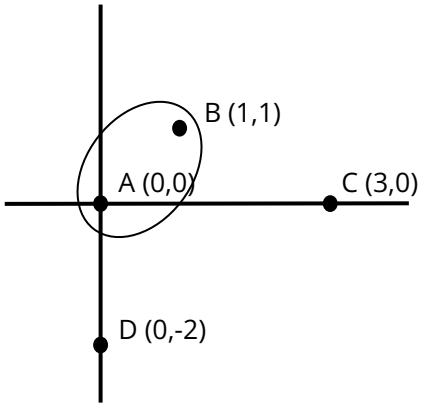
Distance Matrix

	A	B	C	D
A	0	√2	3	2
B	√2	0	√5	√10
C	3	√5	0	√13
D	2	√10	√13	0

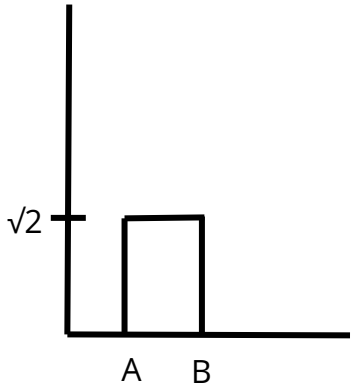
Example

Add to dendrogram

d = Euclidean
D = Single-Link

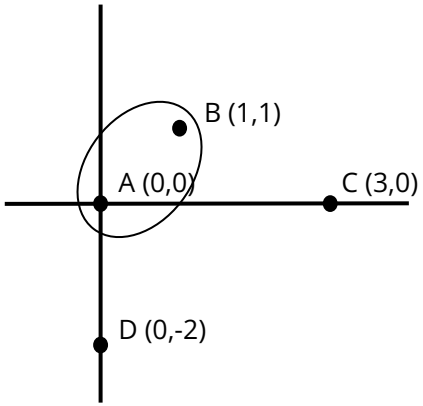


Dendrogram



Example

d = Euclidean
D = Single-Link



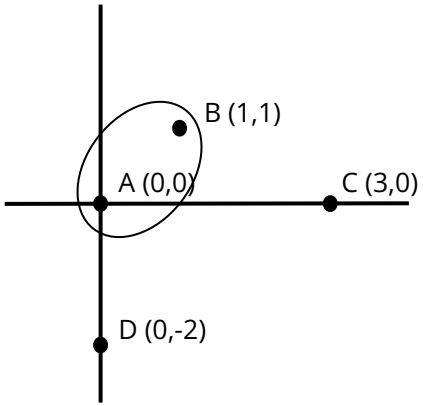
We need to remove the things from the matrix, make it smaller, refill values that don't have A OR B and recalculate the distance for cluter {A,B} and points C and D

Distance Matrix

	A & B	C	D
A & B	0		
C		0	√13
D		√13	0

Example

d = Euclidean
D = Single-Link

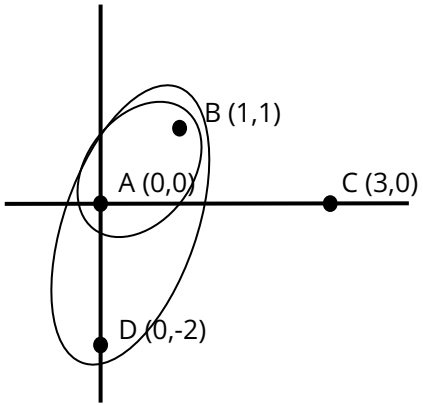


Distance Matrix

	A & B	C	D
A & B	0	√5	2
C	√5	0	√13
D	2	√13	0

Example

d = Euclidean
D = Single-Link



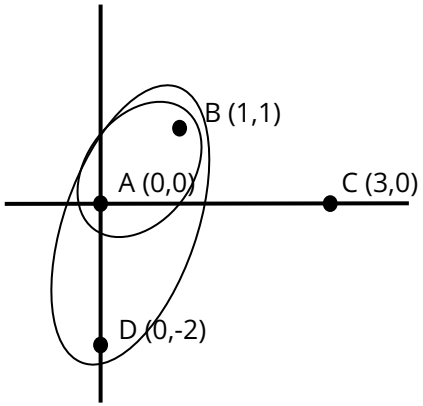
Distance Matrix

	A & B	C	D
A & B	0	√5	2
C	√5	0	√13
D	2	√13	0

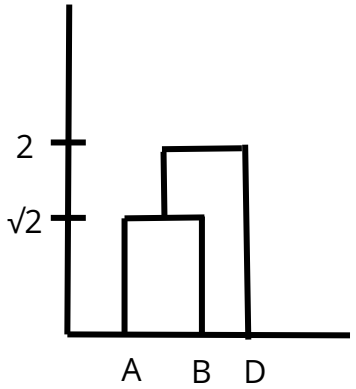
Example

d = Euclidean
D = Single-Link

record

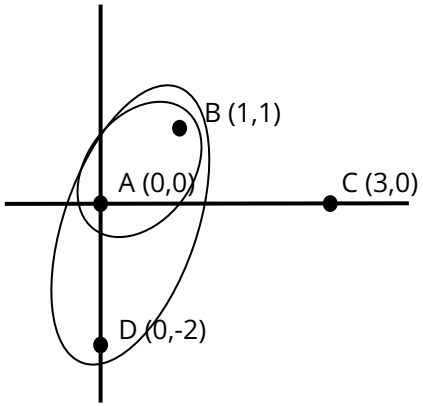


Dendrogram



Example

d = Euclidean
D = Single-Link



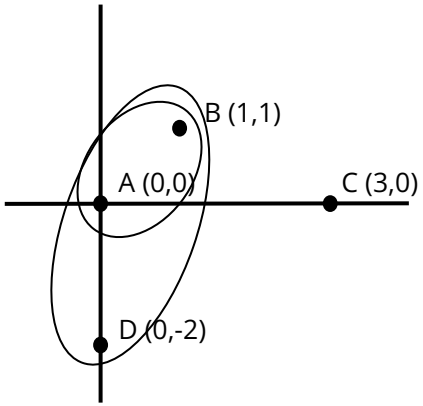
reinitalize the matrix

Distance Matrix

	A & B & D	C
A & B & D	0	
C		0

Example

d = Euclidean
D = Single-Link

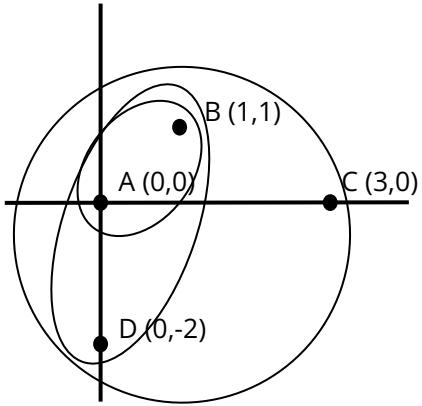


Distance Matrix

	A & B & D	C
A & B & D	0	√5
C	√5	0

Example

d = Euclidean
D = Single-Link



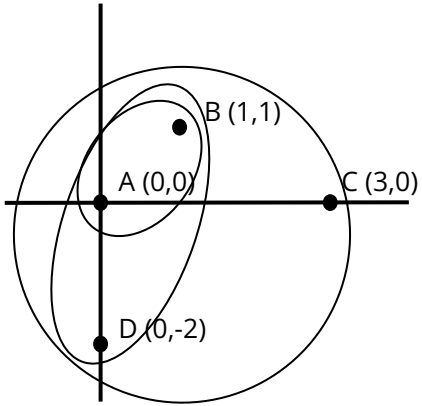
We must recalculate so we have the distance data for the dendrogram

Distance Matrix

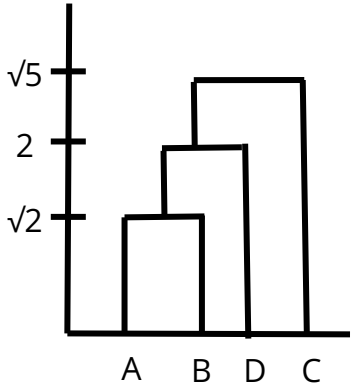
	A & B & D	C
A & B & D	0	√5
C	√5	0

Example

d = Euclidean
D = Single-Link



Dendrogram



Hierarchical Clustering

Finding the threshold with which to cut the dendrogram requires exploration and tuning. But in general hierarchical clustering is used to expose a hierarchy in the data (ex: finding/defining species via DNA similarity).

To capture the difference between clusterings you can use a cost function, or methods that we will discuss later when we look at clustering aggregation.