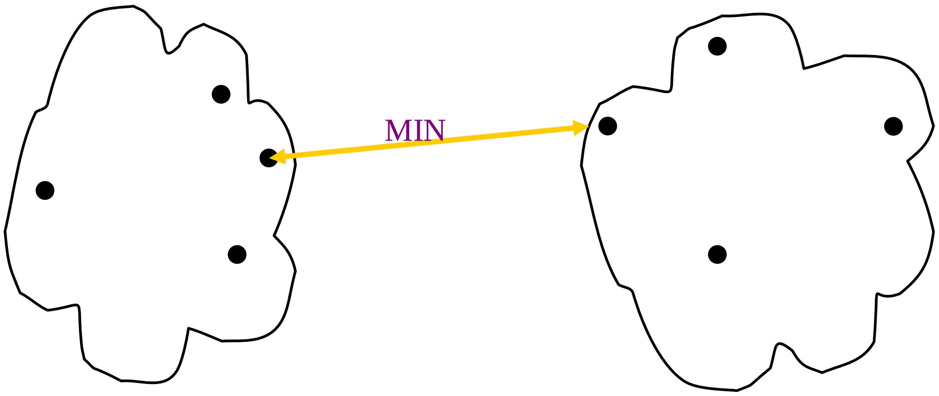
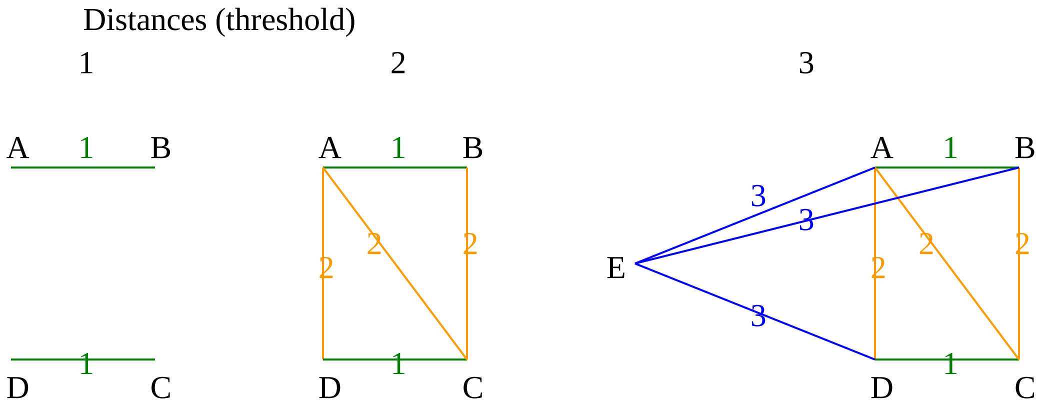
Single Link Clustering

Single link algorithm is an example of agglomerative hierarchical clustering method. We recall that is a bottom-up strategy: compare each point with each point. Each object is placed in a separate cluster, and at each step we merge the closest pair of clusters, until certain termination conditions are satisfied. This requires defining a notion of cluster proximity.

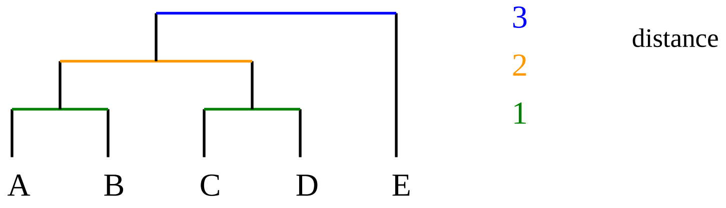
For the single link, the proximity of two clusters is defined as the minimum of the distance between any two points in the two clusters.



Using graph terminology, if we start with all points, each one a separate cluster on its own (called a singleton cluster), and then add links between all points one at a time – shortest links first, then these single links combine the points into clusters. (i.e. the points with the shortest distance between each other are combined into a cluster first, then the next shortest distance are combined, and so on)



Dendogram – shows the same information as in the graph above, however distance threshold is vertical, and points are at the bottom (horizontal). The height at which two clusters are merged in the dendogram reflects the distance of the two clusters.



Example

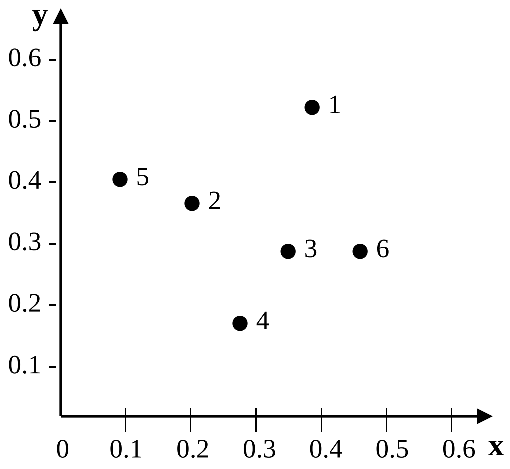
Problem: Assume that the database D is given by the table below. Follow single link technique to find clusters in D. Use Euclidean distance measure.

D

	x	**y**
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

Solution:

Step 1. Plot the objects in *n*-dimensional space (where *n* is the number of attributes). In our case we have 2 attributes – x and y, so we plot the objects p1, p2, … p6 in 2-dimensional space:



Step 2. Calculate the distance from each object (point) to all other points, using Euclidean distance measure, and place the numbers in a distance matrix.

We recall from the previous lecture, the formula for Euclidean distance between two points i and j is:

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j1}|^2 + \ldots + |x_{ip} - x_{jp}|^2}$$

where x_{i1} is the value of attribute 1 for i and x_{j1} is the value of attribute 1 for j, and so on, as many attributes we have … shown up to p - x_{ip} in the formula.

In our case, we only have 2 attributes. So, the Euclidean distance between our points p1 and p2, which have attributes x and y would be calculated as follows:

$$d(p1, p2) = \sqrt{|x_{p1} - x_{p1}|^2 + |y_{p1} - y_{p2}|^2}$$

$$= \sqrt{|0.40 - 0.22|^2 + |0.53 - 0.38|^2}$$

$$= \sqrt{|0.18|^2 + |0.15|^2}$$

$$= \sqrt{0.0324 + 0.0225}$$

$$= \sqrt{0.0549}$$

$$= 0.2343$$

*Note: Euclidean distance calculator can be found here:
http://people.revoledu.com/kardi/tutorial/Similarity/EuclideanDistance.html
Square root calculator can be found here:
http://www.csgnetwork.com/squarerootsquarecalc.html

Analogically, we calculate the distance to the remaining points, and we will receive the following values:

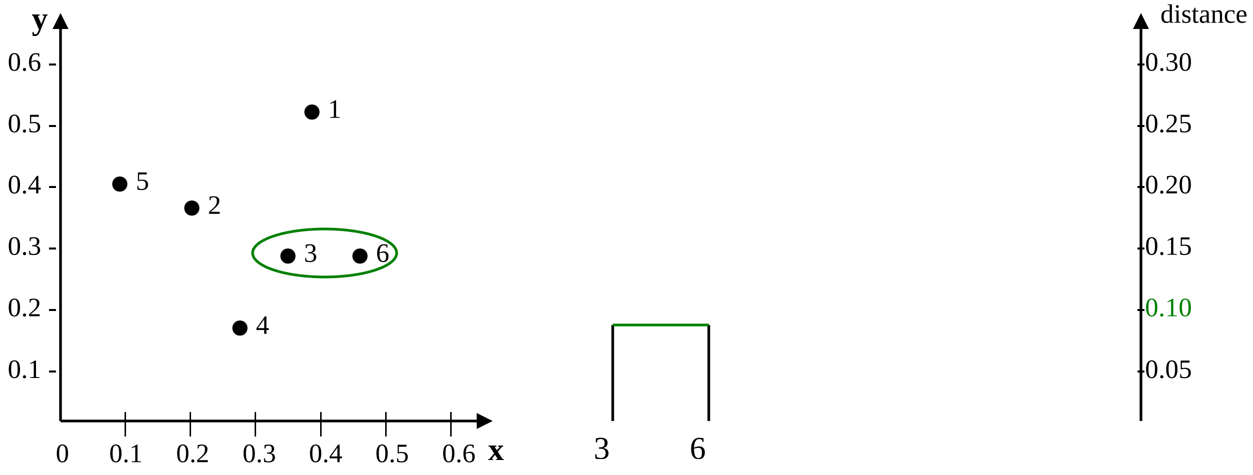
Distance matrix

p1	0					
p2	0.24	0				
p3	0.22	0.15	0			
p4	0.37	0.20	0.15	0		
p5	0.34	0.14	0.28	0.29	0	
p6	0.23	0.25	0.11	0.22	0.39	0
	p1	p2	p3	p4	p5	p6

Step 3 Identify the two clusters with the shortest distance in the matrix, and merge them together. Re-compute the distance matrix, as those two clusters are now in a single cluster, (no longer exist by themselves).

By looking at the distance matrix above, we see that p3 and p6 have the smallest distance from all - 0.11 So, we merge those two in a single cluster, and re-compute the distance matrix.

space dendogram



Distance matrix

p1	0				
p2	0.24	0			
(p3, p6)	**0.22**	**0.15**	0		
p4	0.37	0.20	**0.15**	0	
p5	0.34	0.14	**0.28**	0.29	0
	p1	p2	**(p3, p6)**	p4	p5

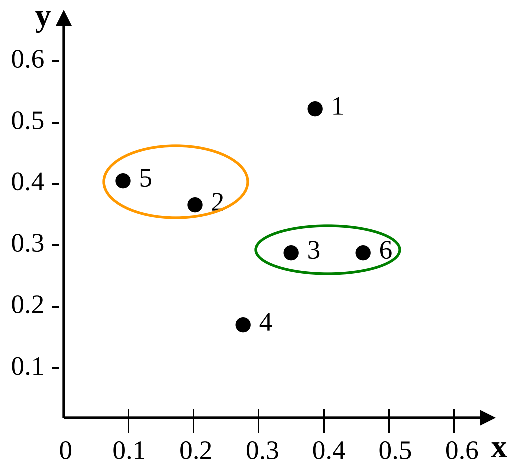
Since, we have merged (p3, p6) together in a cluster, we now have one entry for (p3, p6) in the table, and no longer have p3 or p6 separately. Therefore, we need to re-compute the distance from each point to our new cluster - (p3, p6). We recall that, with the single link method the proximity of two clusters is defined as the minimum of the distance between any two points in the two clusters. Therefore, the distance between let's say (p3, p6) and p1 would be calculated as follows:

dist((p3, p6), p1) = MIN (*dist*(p3, p1) , *dist*(p6, p1))
= MIN (0.22 , 0.23) //from original matrix
= 0.22

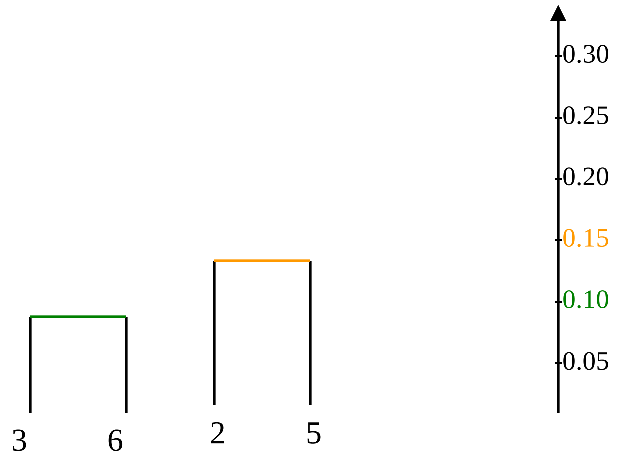
Step 4 Repeat Step 3 until all clusters are merged.

a. So, looking at the last distance matrix above, we see that p2 and p5 have the smallest distance from all - 0.14 So, we merge those two in a single cluster, and re-compute the distance matrix.

space



dendogram



Distance matrix

	p1	**(p2, p5)**	(p3, p6)	p4
p1	0			
(p2, p5)	**0.24**	0		
(p3, p6)	0.22	**0.15**	0	
p4	0.37	**0.20**	0.15	0

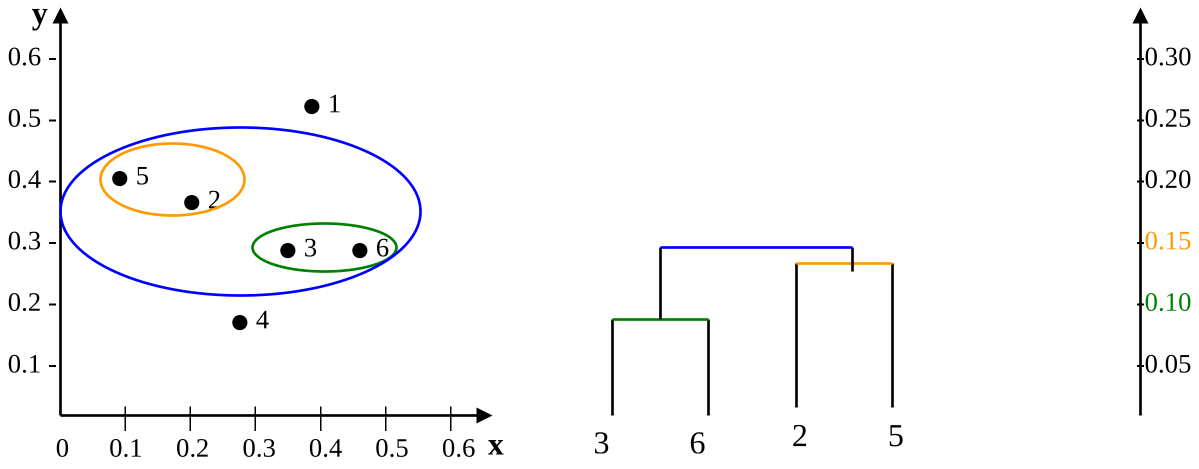
Since, we have merged (p2, p5) together in a cluster, we now have one entry for (p2, p5) in the table, and no longer have p2 or p5 separately. Therefore, we need to re-compute the distance from all other points / clusters to our new cluster - (p2, p5). The distance between (p3, p6) and (p2, p5) would be calculated as follows:

dist((p3, p6), (p2, p5)) = MIN (*dist*(p3, p2) , *dist*(p6, p2), *dist*(p3, p5), *dist*(p6, p5))
= MIN (0.15 , 0.25, 0.28, 0.39) //from original matrix
= 0.15

b. Since we have more clusters to merge, we continue to repeat Step 3.
So, looking at the last distance matrix above, we see that (p2, p5) and (p3, p6) have the smallest distance from all - 0.15 . We also notice that p4 and (p3, p6) have the same distance - 0.15 . In that case, we can pick either one. We choose (p2, p5) and (p3, p6). So, we merge those two in a single cluster, and re-compute the distance matrix.

space

dendogram

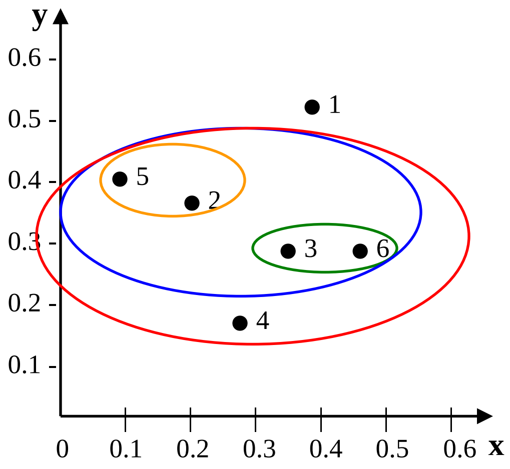


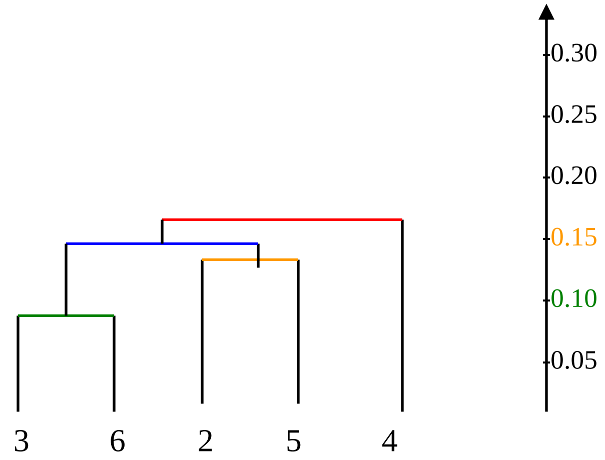
Distance matrix

p1	0		
(p2, p5, p3, p6)	**0.22**	0	
p4	0.37	**0.15**	0
	p1	**(p2, p5, p3, p6)**	p4

c. Since we have more clusters to merge, we continue to repeat Step 3. So, looking at the last distance matrix above, we see that (p2, p5, p3, p6) and p4 have the smallest distance from all - 0.15 . So, we merge those two in a single cluster, and re-compute the distance matrix.

space dendogram



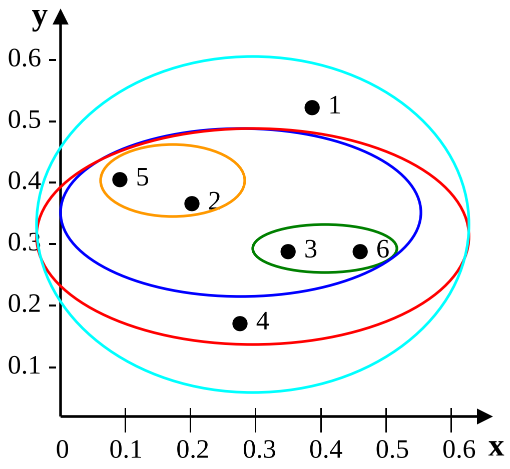


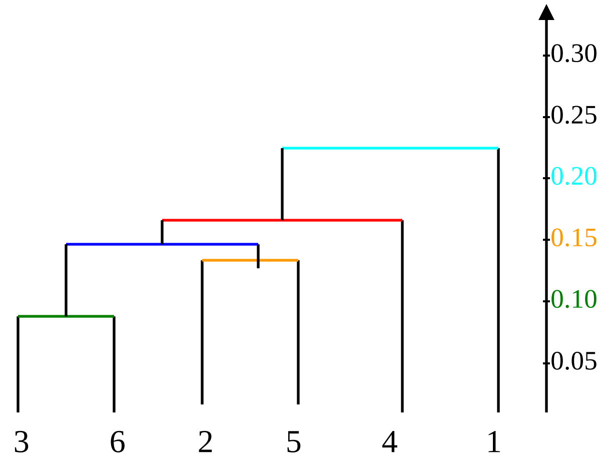
Distance matrix

	p1	(p2, p5, p3, p6, p4)
p1	0	
(p2, p5, p3, p6, p4)	0.22	0

d. Since we have more clusters to merge, we continue to repeat Step 3. So, looking at the last distance matrix above, we see that (p2, p5, p3, p6, p4) and p1 have the smallest distance - 0.22 (the only one left). So, we merge those two in a single cluster. There is no need to re-compute the distance matrix, as there are no more clusters to merge.

space dendogram





Stopping condition - when we explained the single link technique earlier in this lecture, we indicated that “each object is placed in a separate cluster, and at each step we merge the closest pair of clusters, *until certain termination conditions are satisfied*”.

In the example above, we have merged all points into a single cluster at the end. Of course, this is not the goal of the user. The user would like the data partitioned into several clusters for unsupervised learning purposes. Therefore, the algorithm has to stop clustering at some point – either the user will specify the number of clusters he/she would like to have, or the algorithm has to make a decision on its own.

In either case, it is a good strategy to let the algorithm run until the end (until all points are merged into 1 cluster), and record the distances at which each merge was made as it goes. (we recall those are shown in the dendogram). We will notice that the distances increase, as we merge more points into each cluster. We can use that to monitor, if there is a large *jump* in the dendogram i.e. the distance increases dramatically all of a sudden. Then, we know the new cluster is very far away from the one we are about to merge it with, so that is a good indication that the two probably should not be merged – objects have large dissimilarities. The algorithm will use this strategy to make a decision on its own about the number of clusters that should stay separate. The same can be used when deciding which clusters should stay separate and the user provided the number of clusters he/she would like to have.