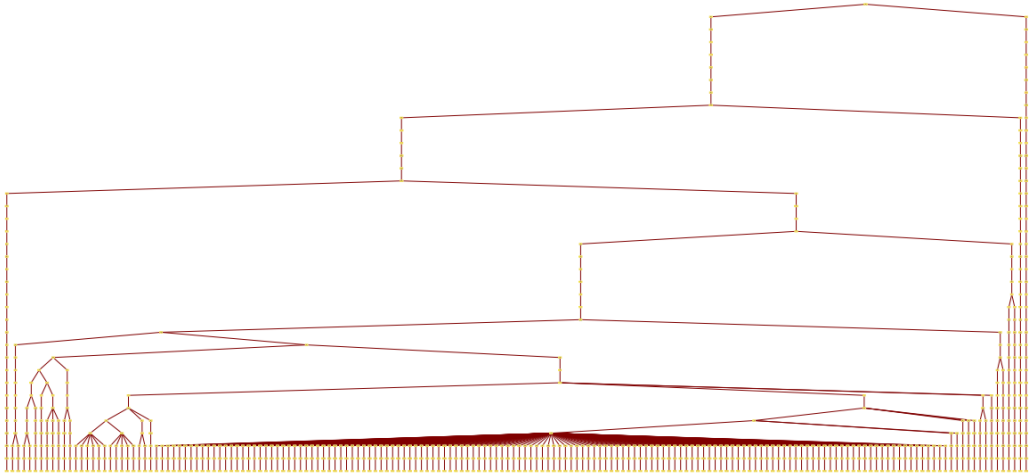
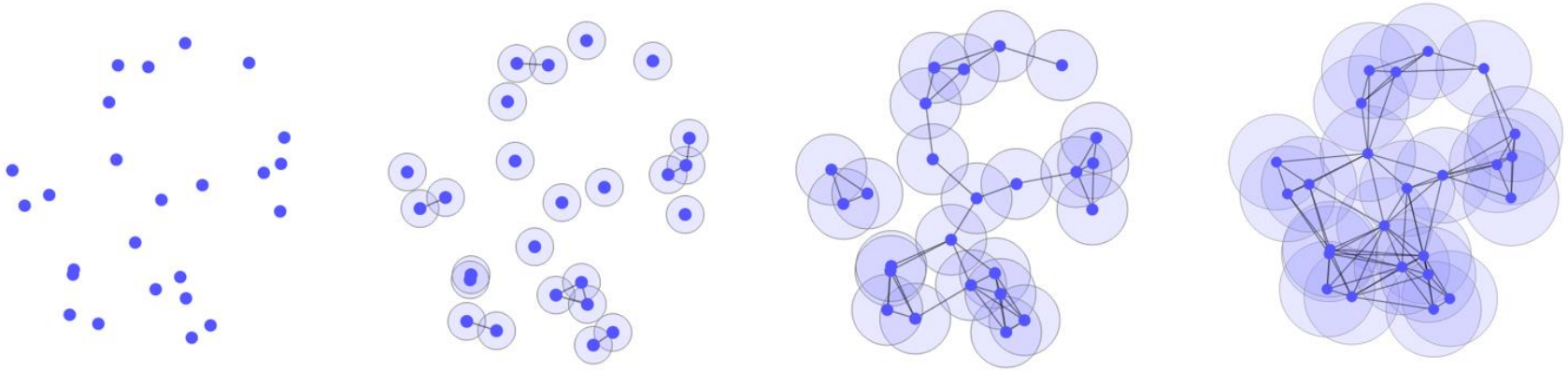


Hierarchical Clustering



Mustafa Hajij

Hierarchical Clustering

Hierarchical clustering is a family of clustering algorithms that build a tree clusters. It is usually done be done by merging or splitting the clusters successively.

Hierarchical Clustering

Hierarchical clustering is a family of clustering algorithms that build a tree clusters. It is usually done be done by merging or splitting the clusters successively.

Hierarchical clustering is usually represented by a tree called the dendrogram that represents the clusterings at all levels.

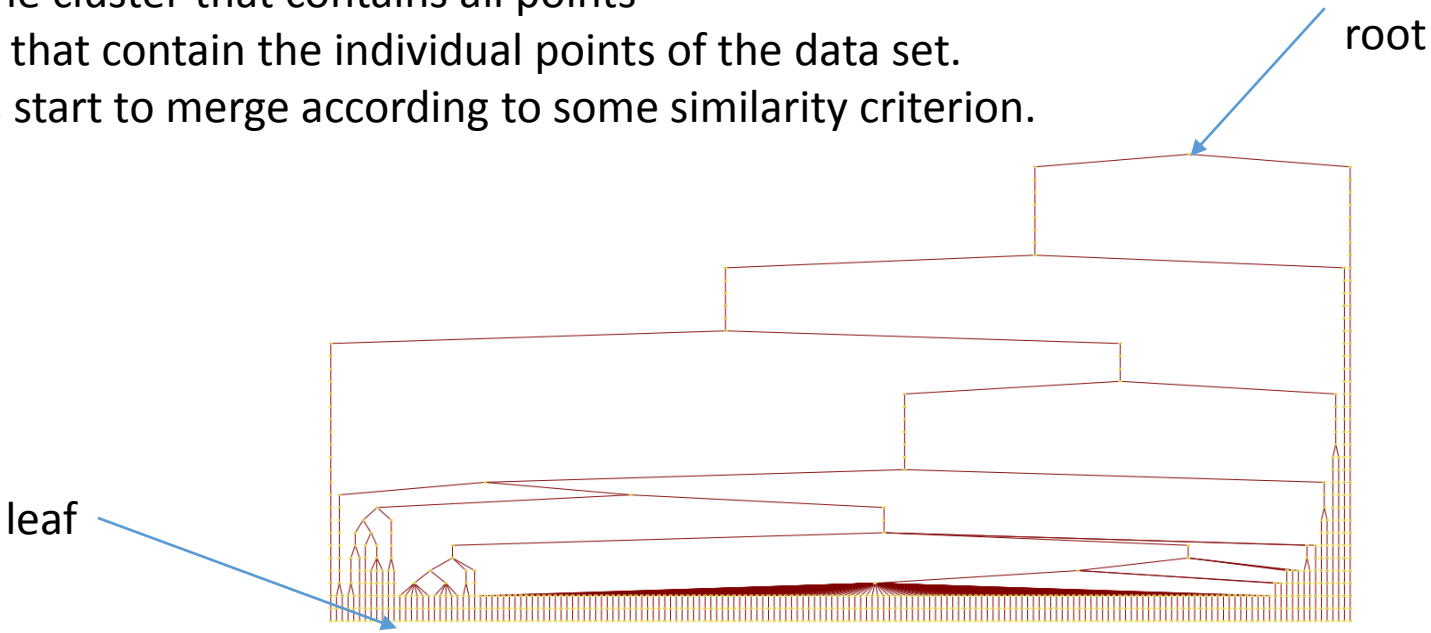
Each node in the tree represents a cluster

- In particular the root of the tree represents the cluster that contains all points
- The leaves of the tree represents the clusters that contain the individual points of the data set.
- As we go from the leaves to the root, clusters start to merge according to some similarity criterion.

There are two types of Hierarchical clustering

Agglomerative : bottom up (Merge).

Divisive : Top down (Split).



General Steps

General Steps in a standard hierarchical agglomerative clustering algorithm:

1. Compute the distance matrix, or the dissimilarity, between all points in the input data. Choosing the metric will impact the results largely.

General Steps

General Steps in a standard hierarchical agglomerative clustering algorithm:

1. Compute the distance matrix, or the dissimilarity, between all points in the input data. Choosing the metric will impact the results largely.
2. Initialize every point in the dataset to be its own cluster

General Steps

General Steps in a standard hierarchical agglomerative clustering algorithm:

1. Compute the distance matrix, or the dissimilarity, between all points in the input data. Choosing the metric will impact the results largely.
2. Initialize every point in the dataset to be its own cluster

3. Compute the distance between all clusters

General Steps

General Steps in a standard hierarchical agglomerative clustering algorithm:

1. Compute the distance matrix, or the dissimilarity, between all points in the input data. Choosing the metric will impact the results largely.
2. Initialize every point in the dataset to be its own cluster

3. Compute the distance between all clusters
4. Combine the closest two clusters: the two clusters $c_i and\ c_j\ with \min_{i,j} D(c_i, c_j)$

General Steps

General Steps in a standard hierarchical agglomerative clustering algorithm:

1. Compute the distance matrix, or the dissimilarity, between all points in the input data. Choosing the metric will impact the results largely.
2. Initialize every point in the dataset to be its own cluster
3. Compute the distance between all clusters
4. Combine the closest two clusters: the two clusters $c_i and\ c_j\ with \min_{i,j} D(c_i, c_j)$
5. Remove the clusters $c_i and\ c_j$ and add the cluster $c_i + c_j$.

General Steps

General Steps in a standard hierarchical agglomerative clustering algorithm:

1. Compute the distance matrix, or the dissimilarity, between all points in the input data. Choosing the metric will impact the results largely.
2. Initialize every point in the dataset to be its own cluster

3. Compute the distance between all clusters
4. Combine the closest two clusters: the two clusters $c_i and\ c_j\ with \min_{i,j} D(c_i, c_j)$
5. Remove the clusters $c_i and\ c_j$ and add the cluster $c_i + c_j$.
6. Go back to 3 and repeat until we have a single cluster.

General Steps

General Steps in a standard hierarchical agglomerative clustering algorithm:

1. Compute the distance matrix, or the dissimilarity, between all points in the input data. Choosing the metric will impact the results largely.
2. Initialize every point in the dataset to be its own cluster

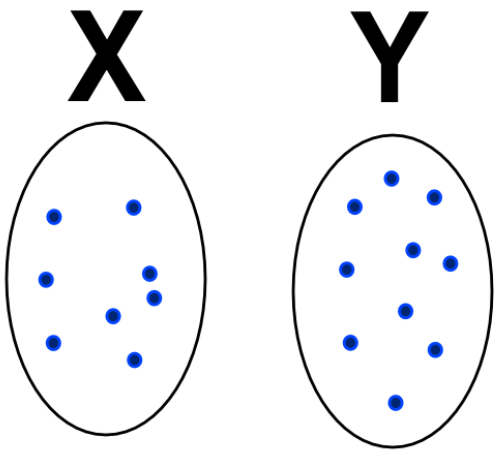
3. Compute the distance between all clusters
4. Combine the closest two clusters: the two clusters $c_i and\ c_j\ with \min_{i,j} D(c_i, c_j)$
5. Remove the clusters $c_i and\ c_j$ and add the cluster $c_i + c_j$.
6. Go back to 3 and repeat until we have a single cluster.

Question : how do we measure the distance between two clusters ?

This is important because step 3 we need to measure the distance between clusters rather than points.

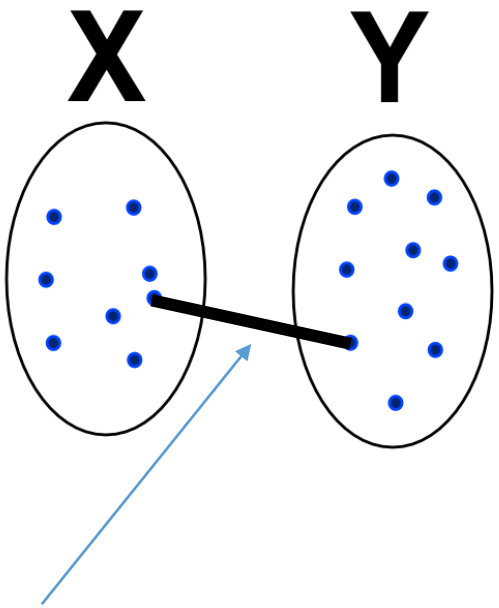
Distance between clusters

Given two clusters X and Y. How do we measure the distance between them ?



Distance between clusters

Given two clusters X and Y. How do we measure the distance between them ?

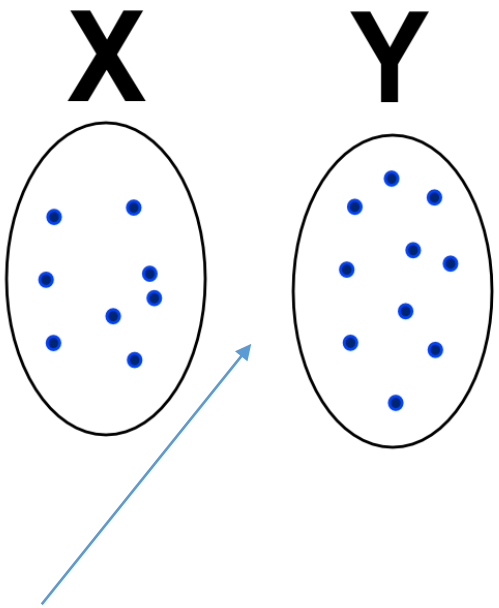


$$D(X,Y) := \min_{x \text{ in } X, y \text{ in } Y} d(x,y)$$

One way is to measure the minimal distance between all points of X and Y.
This distance induce single linkage clustering.

Distance between clusters

Given two clusters X and Y. How do we measure the distance between them ?

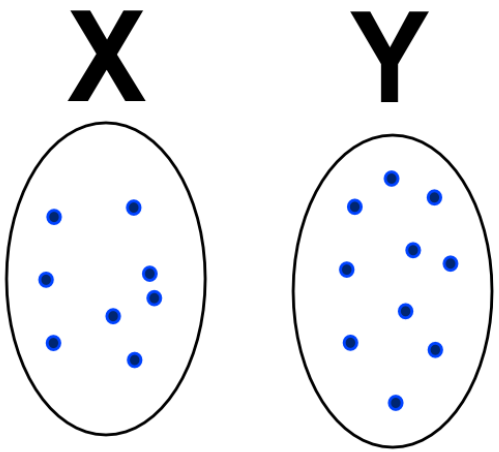


$$D(X,Y) := \frac{1}{|X||Y|} \sum_{x\ in\ X, y\ inY}^{n} d(x,y)$$

We could also consider the mean distance between the points of the clusters

Distance between clusters

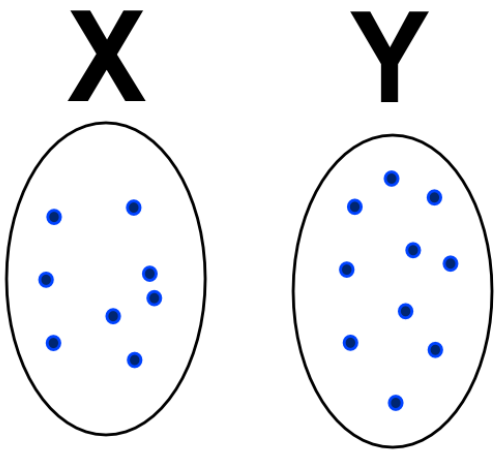
Given two clusters X and Y. How do we measure the distance between them ?



There are other measures as well : The minimal energy criterion , the distance between the centroids of the clusters

Distance between clusters

Given two clusters X and Y. How do we measure the distance between them ?



For efficient calculations we usually require the following condition:
Knowing the distance D(A,C), D(B,C)
Implies we can calculate in constant time the distance D(A+B,C)

There are other measures as well : The minimal energy criterion , the distance between the centroids of the clusters

Single-linkage clustering algorithm

The input of the algorithm is a distance matrix D contains all distances d(i,j) between the points. At the beginning each point is its own cluster.

Single-linkage clustering algorithm

1-Find the most similar pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.

Single-linkage clustering algorithm

1-Find the most similar pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.

2- Merge clusters (r) and (s) into a single cluster to form the next clustering m.

Single-linkage clustering algorithm

1-Find the most similar pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.

2- Merge clusters (r) and (s) into a single cluster to form the next clustering m.

3-Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster.

Single-linkage clustering algorithm

1-Find the most similar pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.

2- Merge clusters (r) and (s) into a single cluster to form the next clustering m.

3-Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster.

4- The distance between the new cluster, denoted (r,s) and old cluster (k) is defined as d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)].

Single-linkage clustering algorithm

1-Find the most similar pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.

2- Merge clusters (r) and (s) into a single cluster to form the next clustering m.

3-Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster.

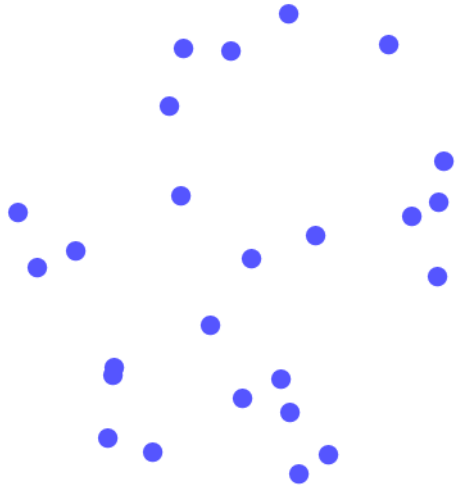
4- The distance between the new cluster, denoted (r,s) and old cluster (k) is defined as d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)].

5- If all objects are in one cluster, stop. Else, go to step 1.

Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \dots, p_n\}$ in R^d with a distance function d defined one them.

Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.

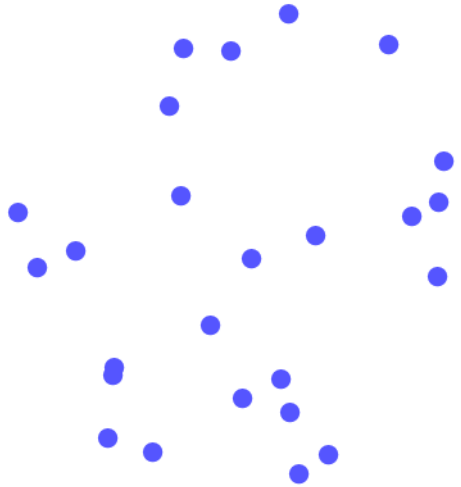


Every point is a connected component

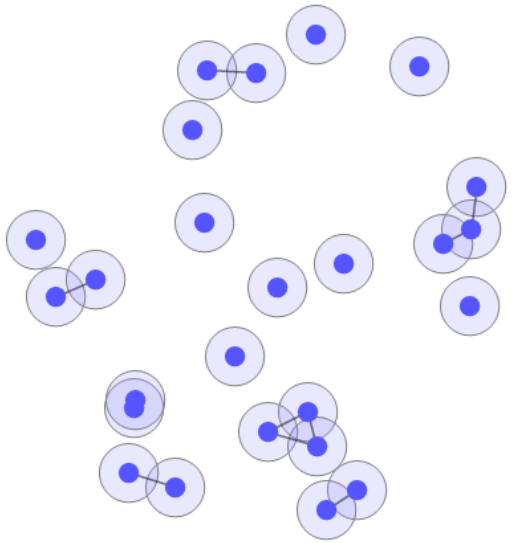
Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \dots, p_n\}$ in R^d with a distance function d defined one them.

Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.



Every point is a connected component

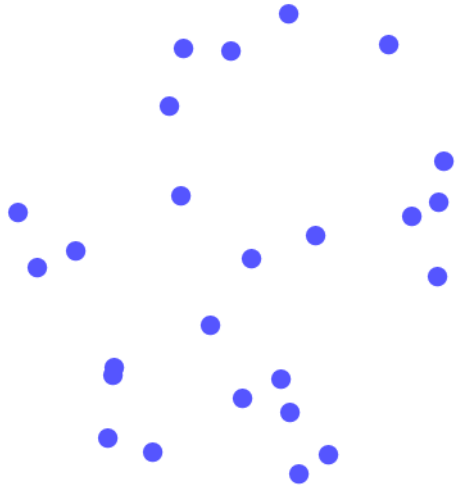


When ε is a little larger we start some clusters starts to get form

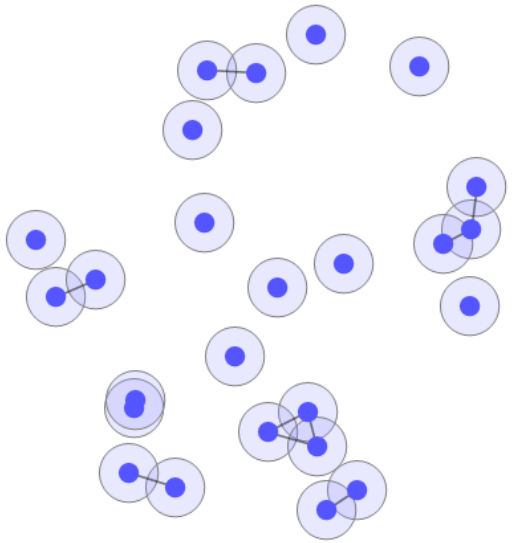
Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \dots, p_n\}$ in R^d with a distance function d defined one them.

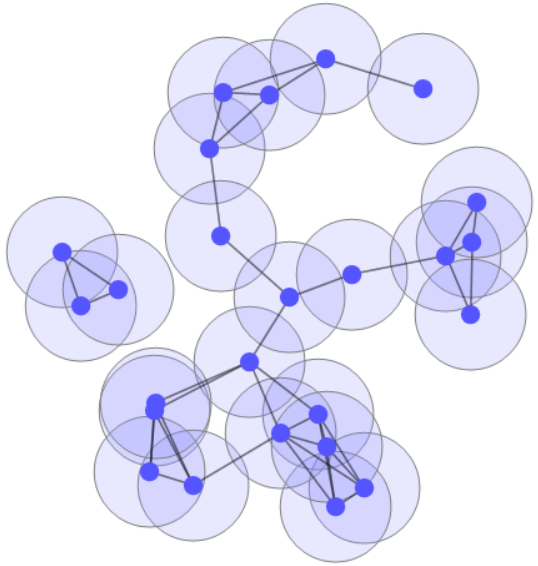
Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.



Every point is a connected component



When ε is a little larger we start some clusters starts to get form

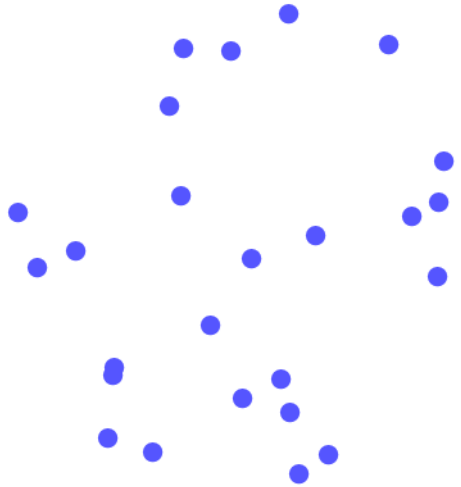


When ε is even larger we have few clusters
As the clusters get larger and larger

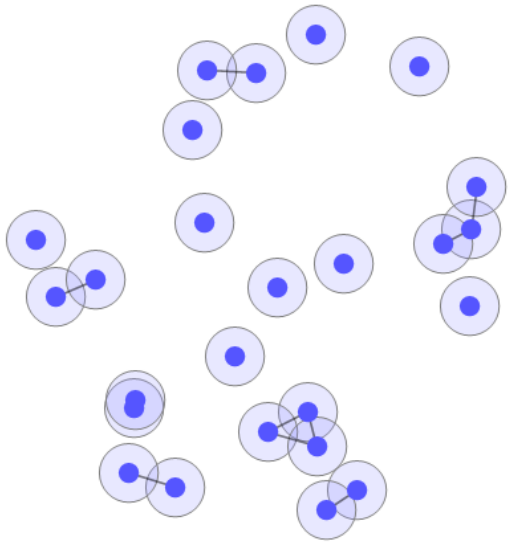
Single Linkage Hierarchical Clustering and the ε- Neighborhood Graph

Suppose that we are given a set of points $X = \{p_1, p_2, \dots, p_n\}$ in R^d with a distance function d defined one them.

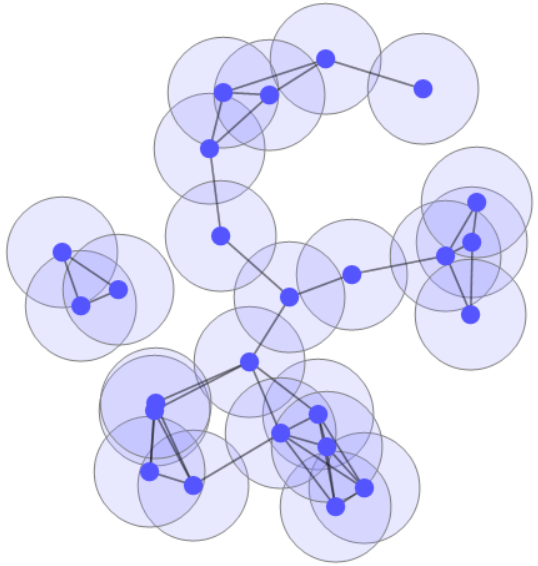
Consider the connected components of the ε-neighborhood graph as we continuously increase ε from zero to infinity.



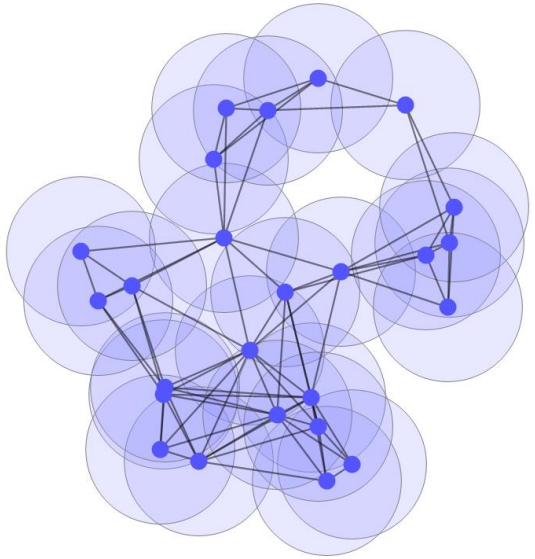
Every point is a connected component



When ε is a little larger we start some clusters starts to get form



When ε is even larger we have few clusters
As the clusters get larger and larger



At some point all points become a par to of a single cluster

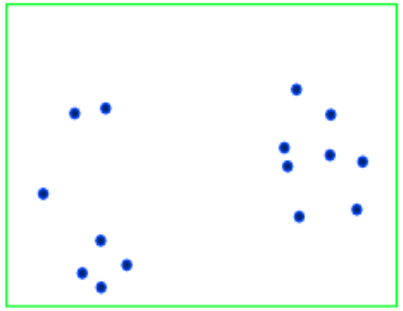
Recall : Kruskal's Algorithm

Let $G = (V, E, w)$ be a connected weighted graph.

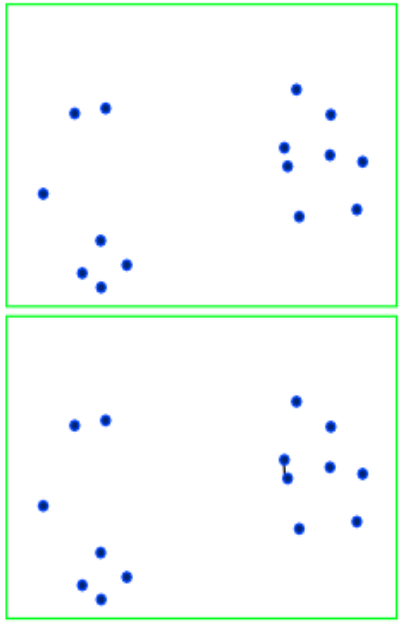
Informally, the algorithm can be given by the following three steps :

1. Set V_T $to\ be\ V, Set\ E_T = \{\}$. Let $S = E$
2. While S is not empty and T is not a spanning tree
    1. Select an edge e from S with the minimum weight and delete e from S.
    2. If e connects two separate trees of T then add e to E_T

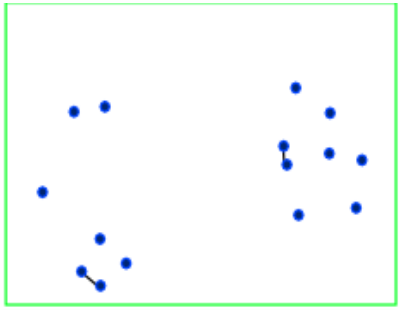
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm

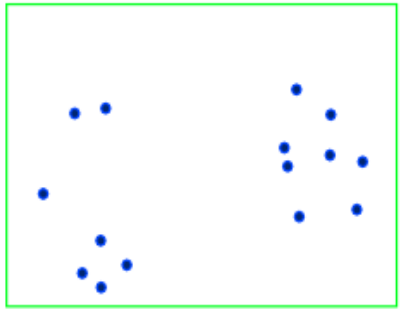


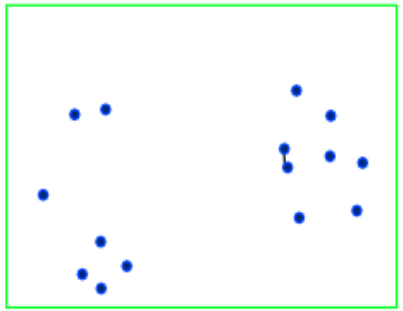
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



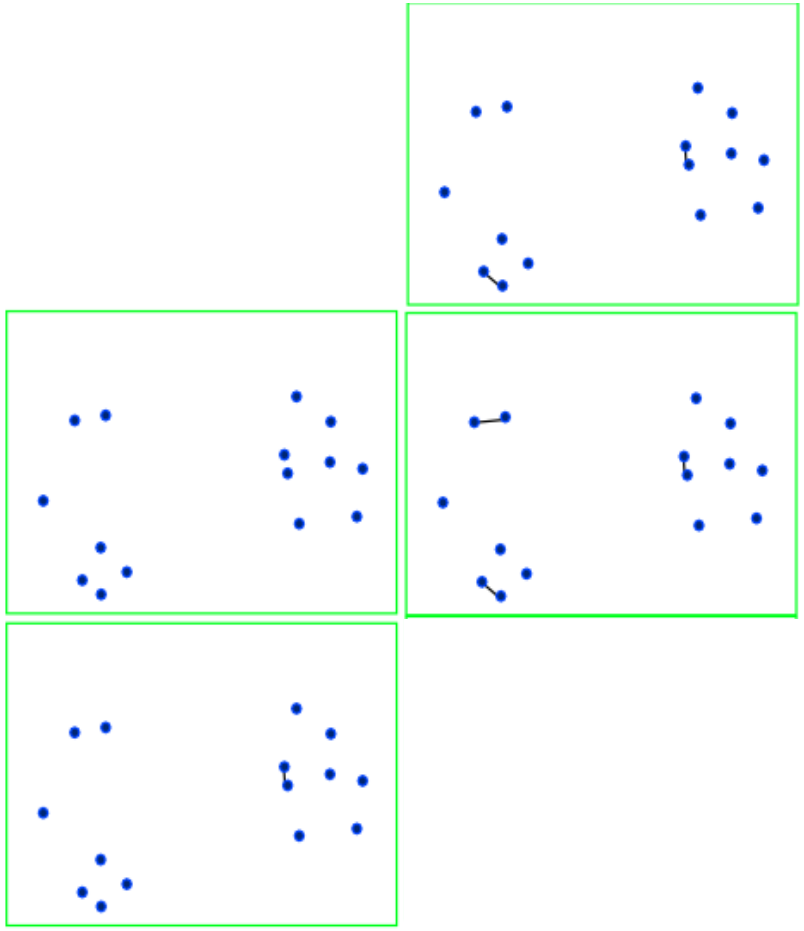
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



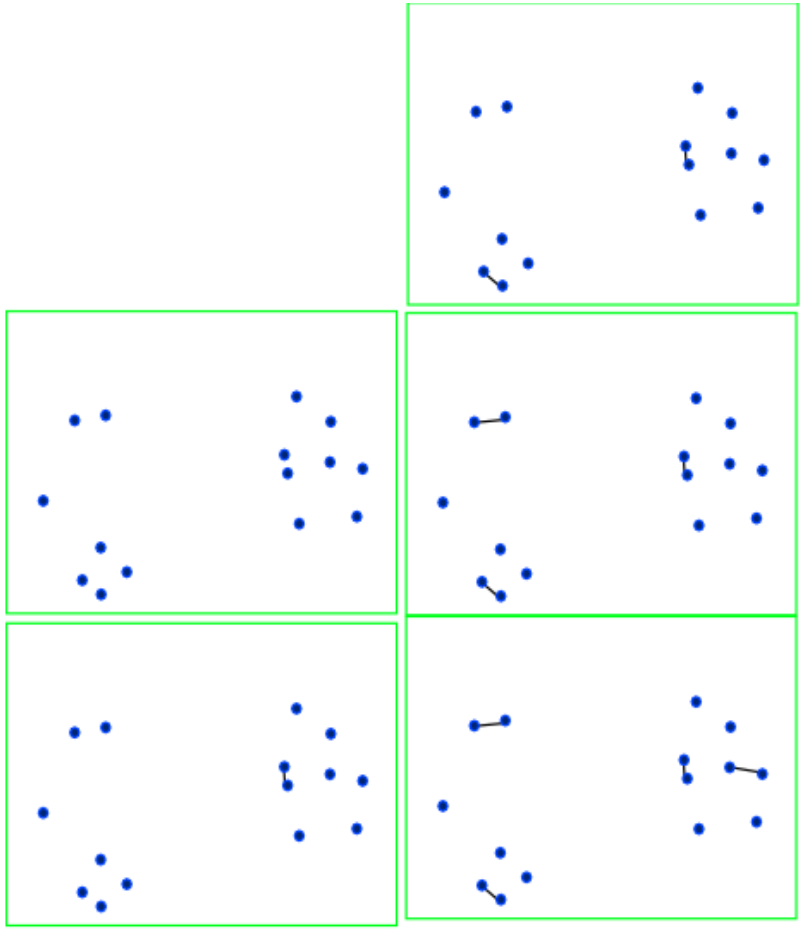




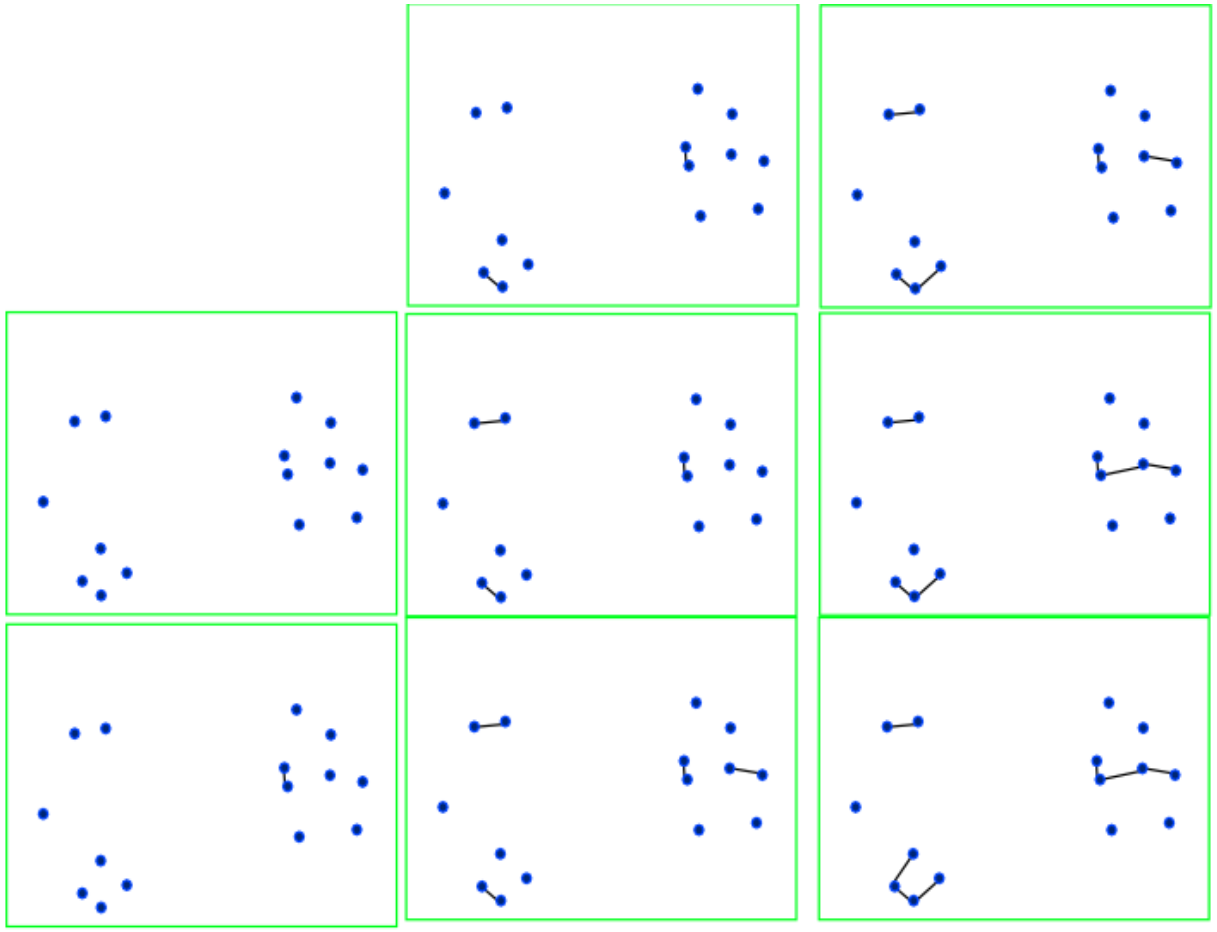
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



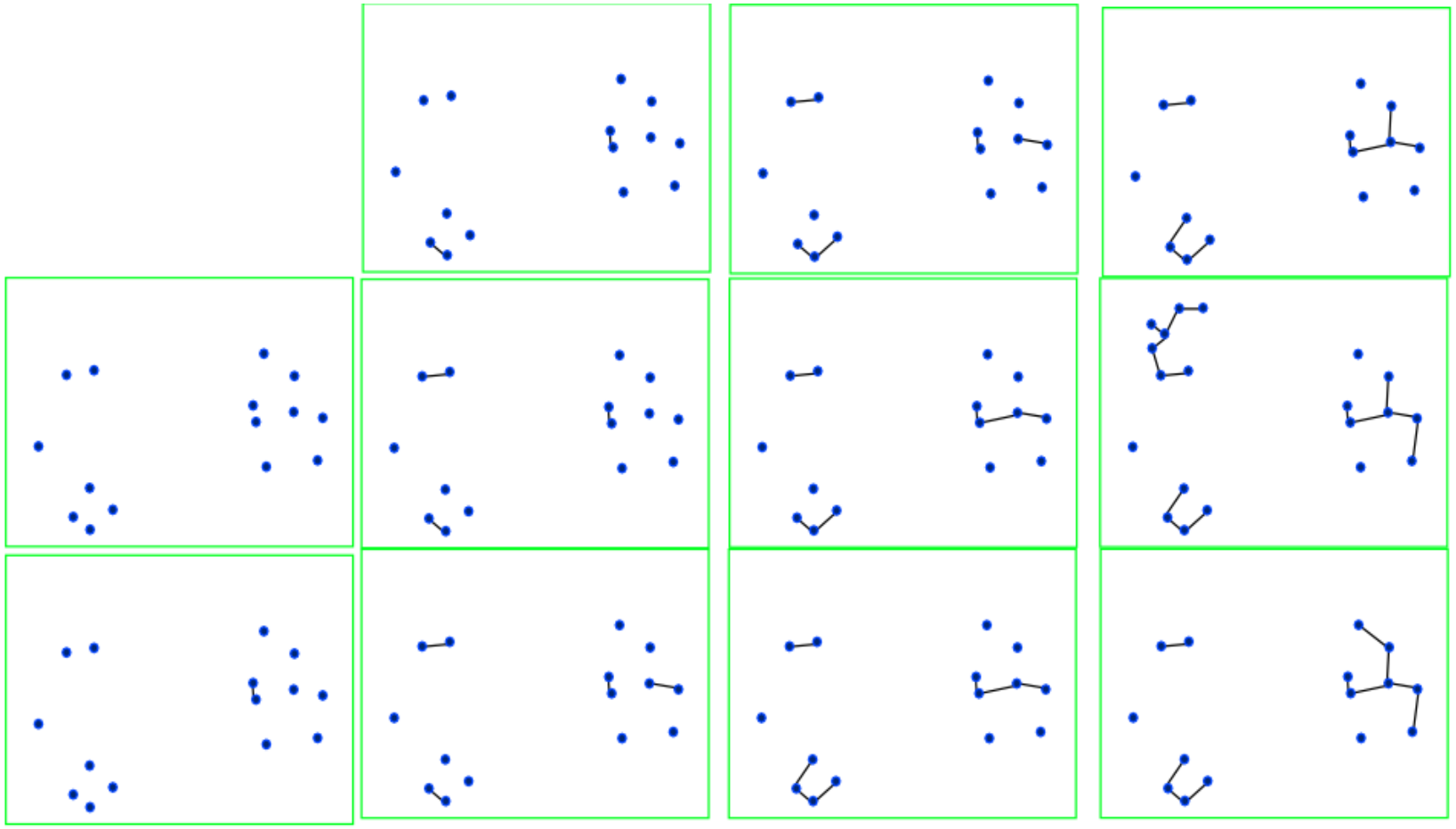
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



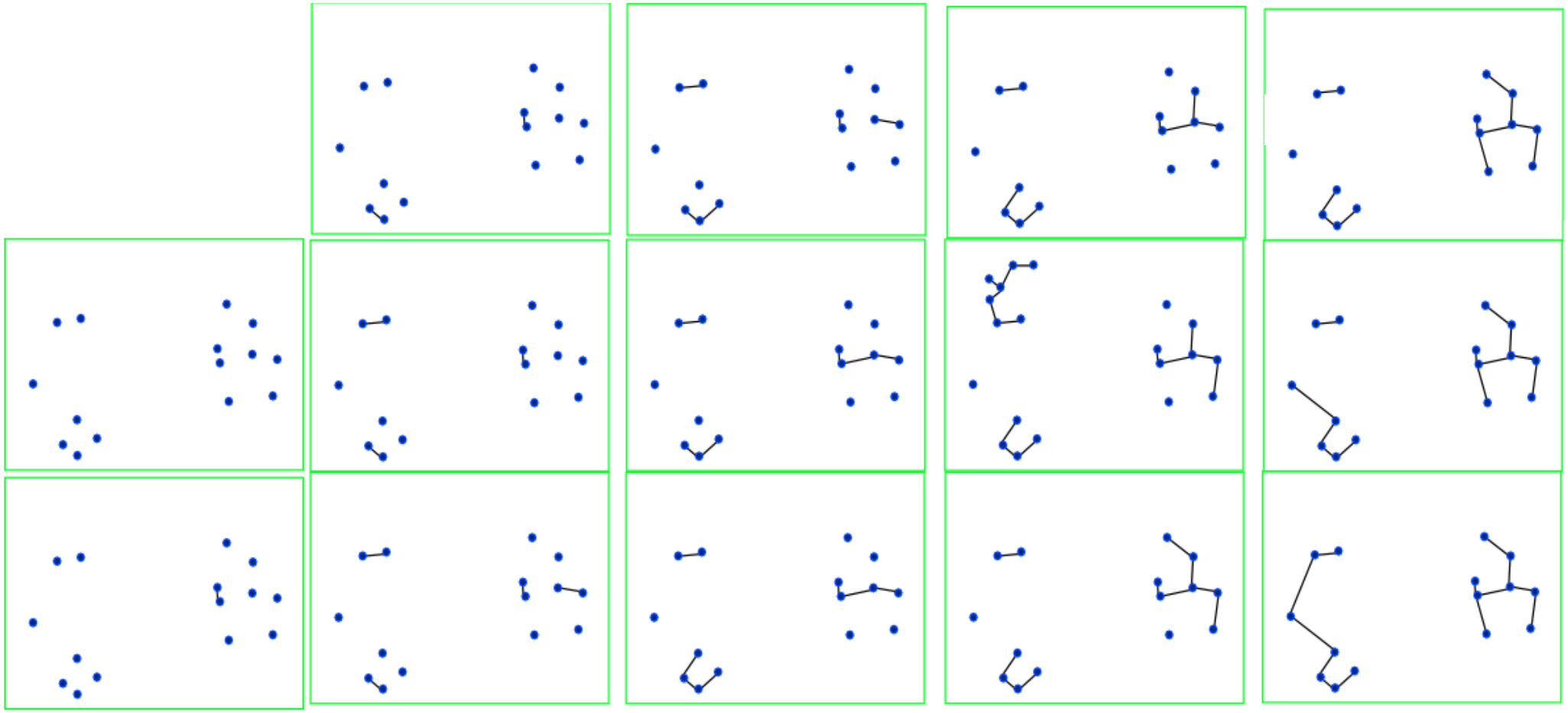
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



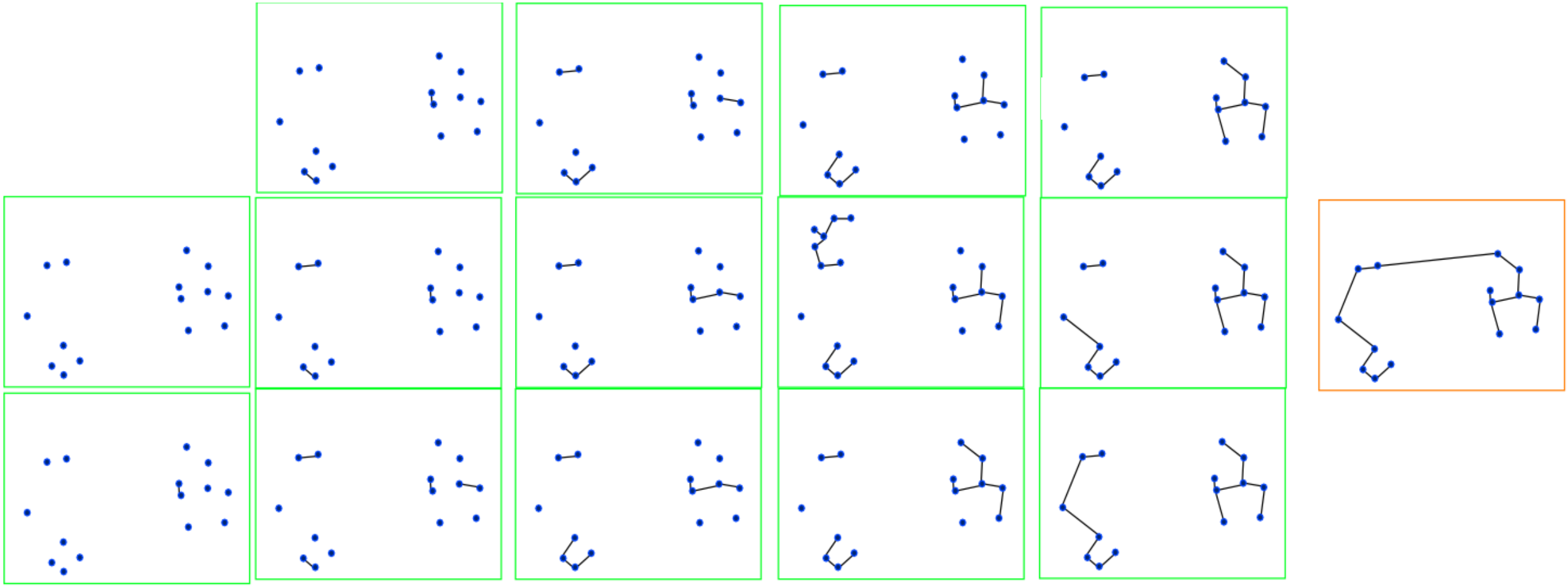
Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



Single Linkage Hierarchical Clustering and the and Kruskal's algorithm



Lance–Williams algorithms

1. Compute the distance matrix : $D = \{D_{ij}: \text{distance between i and j for i, j between 1 and n}\}$

2. Iterate n times :
   1. Find i and j with $\min_{i,j} D(c_i, c_j)$
   2. Add the cluster $i + j$ and delete the clusters i and j
   3. For each remaining cluster k
      1. $D_{k,i+j} = min\{D_{k,j}, D_{k,i}\}$

Single Linkage H-Clustering

Lance–Williams algorithms

1. Compute the distance matrix : $D = \{D_{ij}: \text{distance between i and j for i, j between 1 and n}\}$

2. Iterate n times :
   1. Find i and j with $\min_{i,j} D(c_i, c_j)$
   2. Add the cluster $i + j$ and delete the clusters i and j
   3. For each remaining cluster k
      1. $D_{k,i+j} = min\{D_{k,j}, D_{k,i}\}$

Single Linkage H-Clustering

This can be replaced by a more general condition.

Lance–Williams algorithms

1. Compute the distance matrix : $D = \{D_{ij}: \text{distance between i and j for i, j between 1 and n}\}$

2. Iterate n times :
   1. Find i and j with $\min_{i,j} D(c_i, c_j)$
   2. Add the cluster $i + j$ and delete the clusters i and j
   3. For each remaining cluster k
      1. $D_{k,i+j} = a_i D_{k,i} + a_j D_{k,j} + \beta D_{i,j} + \alpha |D_{k,j} - D_{k,i}|$

Every algorithm has a special
a_i, a_j, β and α

The neat thing about the algorithm : all needs to be changed is how to *update* the new distance

Lance–Williams algorithms

1. Compute the distance matrix : $D = \{D_{ij}: \text{distance between i and j for i, j between 1 and n}\}$

2. Iterate n times :
   1. Find i and j with $\min_{i,j} D(c_i, c_j)$
   2. Add the cluster $i + j$ and delete the clusters i and j
   3. For each remaining cluster k
      1. $D_{k,i+j} = a_i D_{k,i} + a_j D_{k,j} + \beta D_{i,j} + \alpha |D_{k,j} - D_{k,i}|$

Every algorithm has a special
a_i, a_j, β and α

For complete linkage, choose a_i= ? a_j =? α=? and ?

Lance–Williams algorithms

1. Compute the distance matrix : $D = \{D_{ij}: \text{distance between i and j for i, j between 1 and n}\}$

2. Iterate n times :
   1. Find i and j with $\min_{i,j} D(c_i, c_j)$
   2. Add the cluster $i + j$ and delete the clusters i and j
   3. For each remaining cluster k
      1. $D_{k,i+j} = a_i D_{k,i} + a_j D_{k,j} + \beta D_{i,j} + \alpha |D_{k,j} - D_{k,i}|$

Ward's method : $a_l = \frac{n_l + n_k}{n_i + n_j + n_k}$, $\beta = \frac{-n_k}{n_i + n_j + n_k}$, $\alpha = 0$

Lance–Williams algorithms

What is the complexity here ?

1. Compute the distance matrix : $D = \{D_{ij}: \text{distance between i and j for i, j between 1 and n}\}$ ← n^2 steps

2. Iterate n times : ← n steps
   1. Find i and j with $\min_{i,j} D(c_i, c_j)$ ← n^2 steps
   2. Add the cluster $i + j$ and delete the clusters i and j
   3. For each remaining cluster k ← n steps at most
      1. $D_{k,i+j} = a_i D_{k,i} + a_j D_{k,j} + \beta D_{i,j} + \alpha |D_{k,j} - D_{k,i}|$ ← Constant time!

This is just the naïve implementation, there are better algorithms that perform with $O(n^2)$

Divisive Clustering

- Begin with the entire dataset and consider it as a single cluster
- At each iteration, we select an existing cluster and split it into two clusters using any clustering algorithm we have seen so far (k-means for instance).

In Sklearn

Scikit learn supports Agglomerative Clustering . Many features discussed in this lecture are also supported.