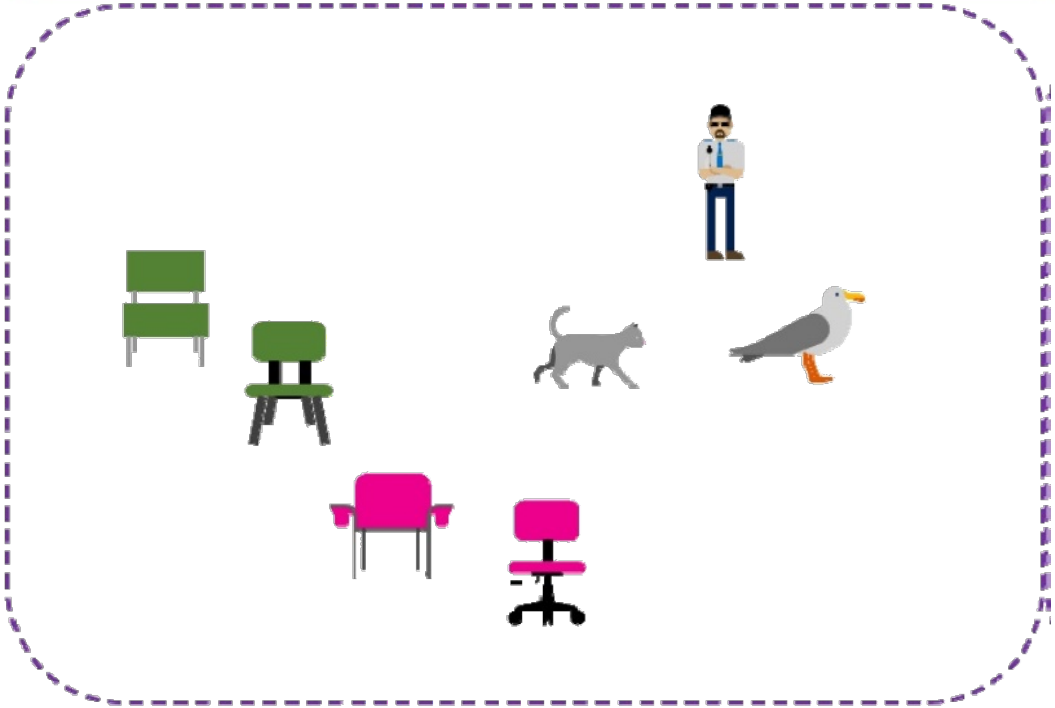
Details of ML part 3

Unsupervised learning

Clustering

Machine 1 machine B Machine 2

machine A Machine C machine 3




Machine 1 machine B Machine 2
machine A Machine C machine 3

(a)


Machine 1 machine A Machine 2
machine B Machine C machine 3

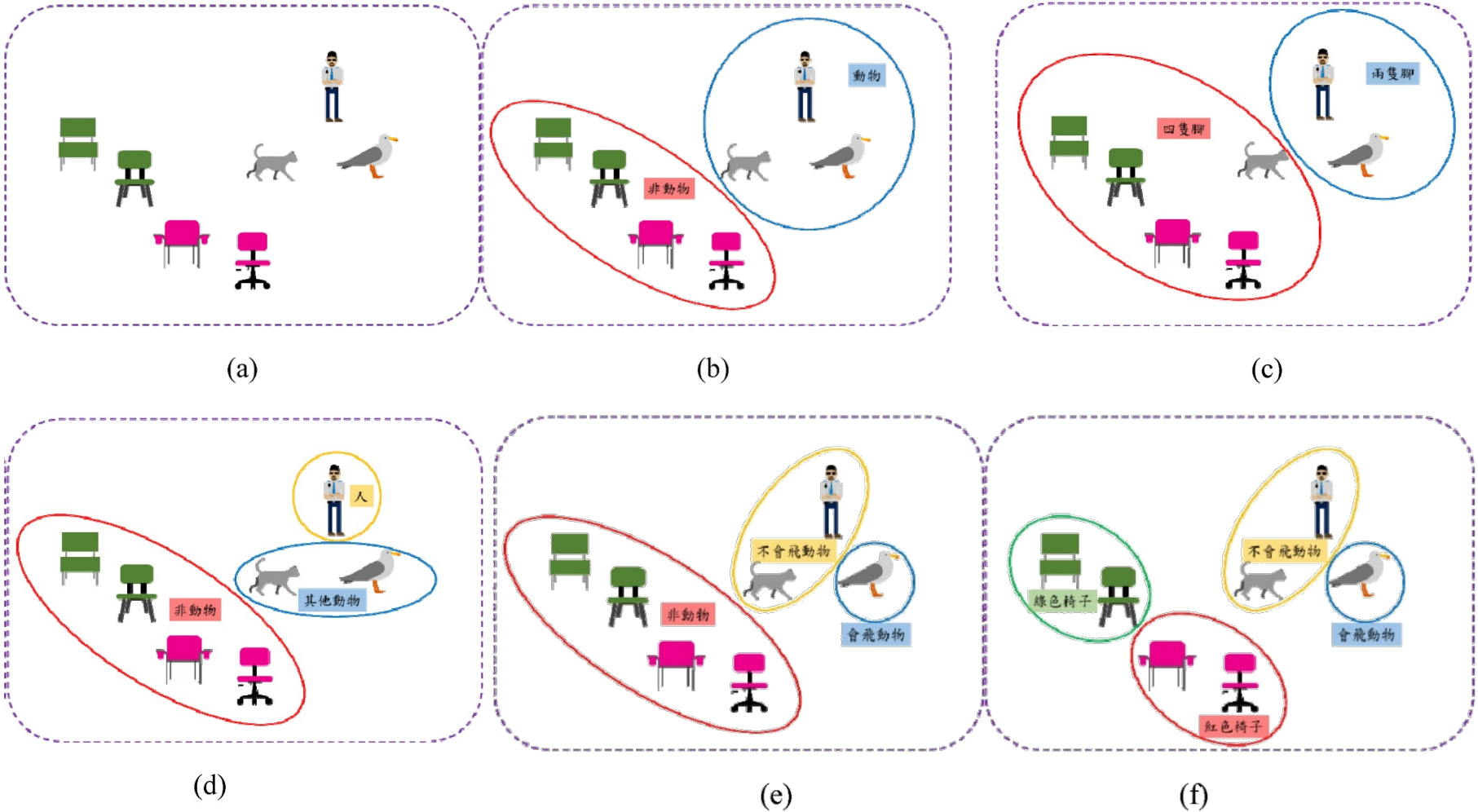
(b)


Machine 1 Machine C Machine 2
machine A machine B machine 3

(c)


machine A machine B Machine C
Machine 1 Machine 2 machine 3

(d)



(a) (b) (c)

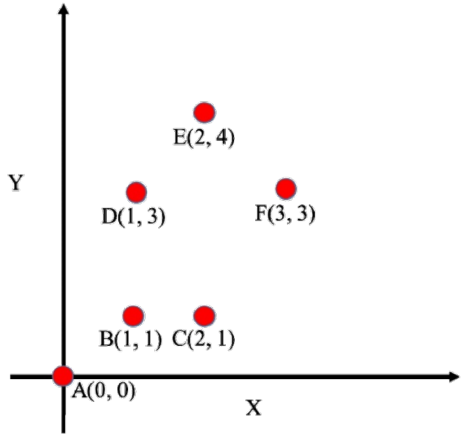
(d) (e) (f)

Clustering

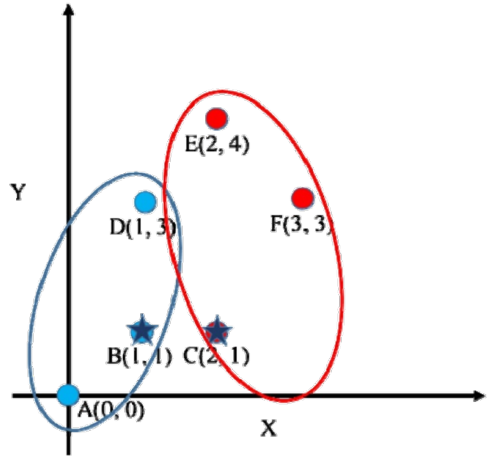
- A cluster is a subset of data which are similar.
- Clustering is the process of dividing a dataset into groups such that the members of each group are as similar (close) as possible to one another, and different groups are as dissimilar (far) as possible from one another.
- Generally, it is used as a process to find meaningful structure, generative features, and groupings inherent in a set of examples.
- Clustering can uncover previously undetected relationships in a dataset. There are many applications for cluster analysis. For example, in business, cluster analysis can be used to discover and characterize customer segments for marketing purposes and in biology, it can be used for classification of plants and animals given their features.

K-means algorithm

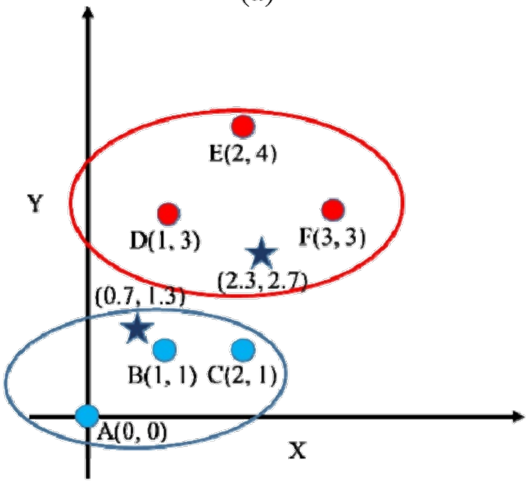
1. Specify number of clusters K
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters is not changing
   - Compute the sum of the squared distance between data points and all centroids
   - Assign each data point to the closest cluster (centroid)
   - Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster


E(2, 4)
Y
D(1, 3)
F(3, 3)
B(1, 1)
C(2, 1)
A(0, 0)
X

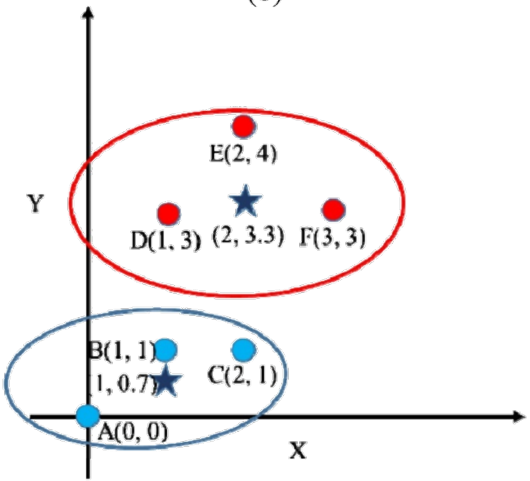
(a)


E(2, 4)
Y
D(1, 3)
F(3, 3)
B(1, 1)
C(2, 1)
A(0, 0)
X

(b)

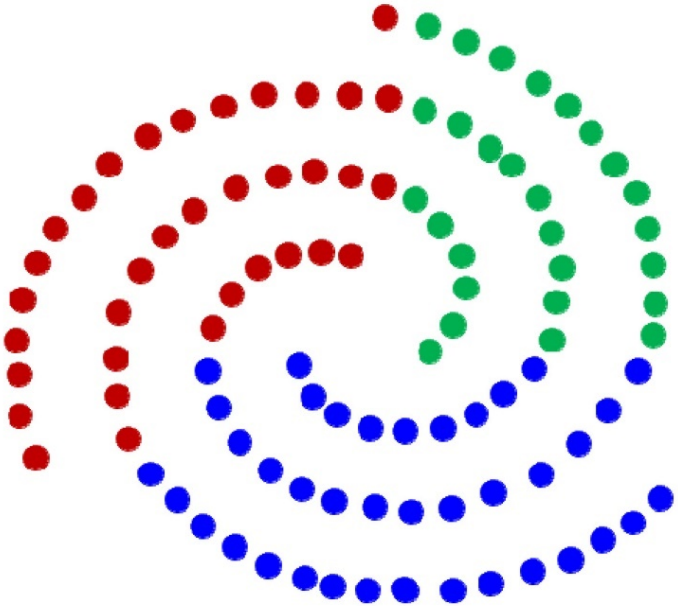

E(2, 4)
Y
D(1, 3)
F(3, 3)
(2.3, 2.7)
(0.7, 1.3)
B(1, 1)
C(2, 1)
A(0, 0)
X

(c)


E(2, 4)
Y
D(1, 3)
(2, 3.3)
F(3, 3)
B(1, 1)
1, 0.7)
C(2, 1)
A(0, 0)
X

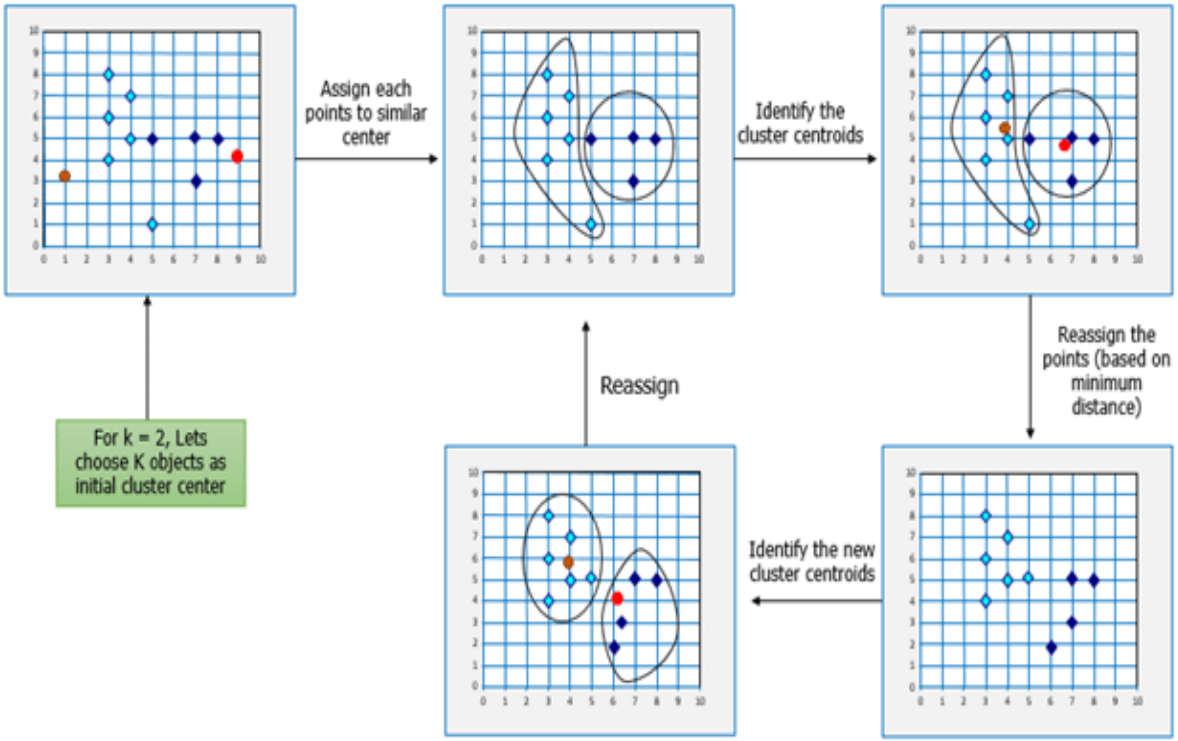
(d)

(a)

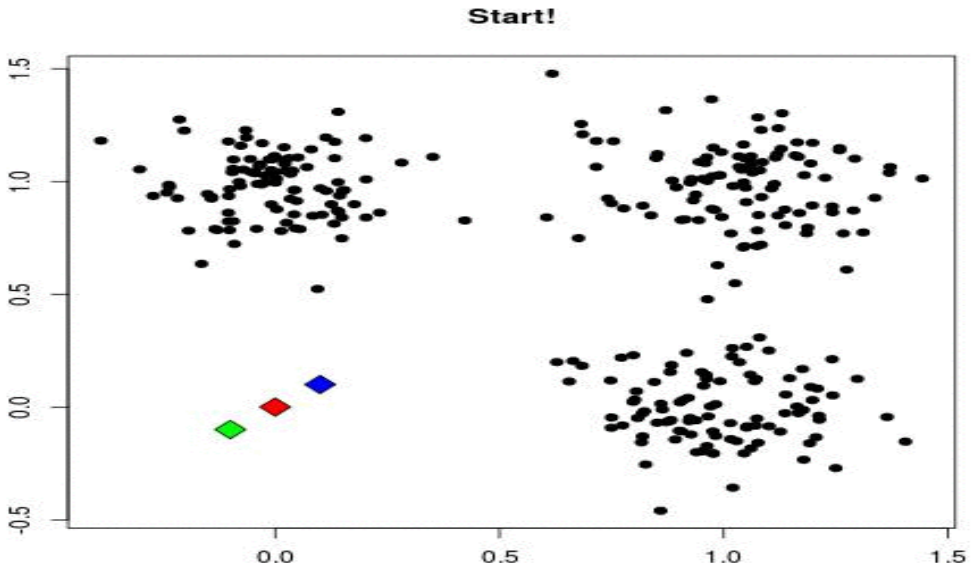


(b)

- **Another case, the step by step process:**



- **Advantage of K-MEANS:**
- K-means clustering algorithm has found to be very useful in grouping new data. Some practical applications which use k-means clustering are sensor measurements, activity monitoring in a manufacturing process, audio detection and image segmentation.

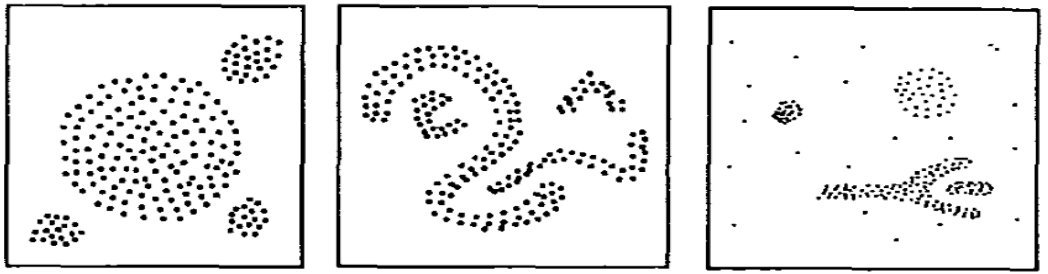


- **Disadvantage of K-MEANS:**
- K-Means forms spherical clusters only. This algorithm fails when data is not spherical (i.e. same variance in all directions).
- K-Means algorithm is sensitive towards outlier. Outliers can skew the clusters in K-Means in very large extent.
- K-Means algorithm requires one to specify the number of clusters and for which there is no global method to choose best value.

Density-based spatial clustering of applications with noise (DBSCAN) is a well-known data clustering algorithm that is commonly used in data mining and machine learning.

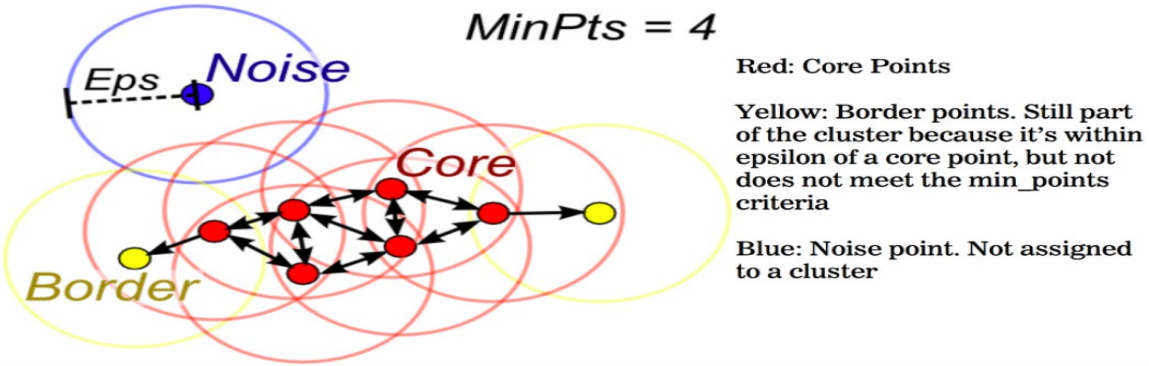
- Unlike to **K-means**, DBSCAN does not require the user to specify the number of clusters to be generated
- DBSCAN can find any shape of clusters. The cluster doesn't have to be circular.
- DBSCAN can identify outliers

The basic idea behind **density-based clustering** approach is derived from a human intuitive clustering method. by looking at the figure below, one can easily identify four clusters along with several points of noise, because of the differences in the density of points



• DBSCAN algorithm has two parameters:

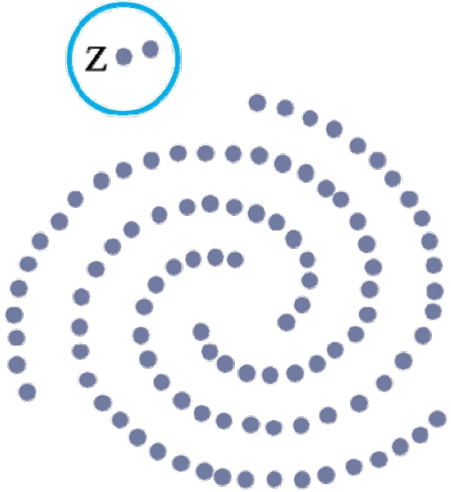
- ε: The radius of our neighborhoods around a data point *p*.
- *minPts*: The minimum number of data points we want in a neighborhood to define a cluster.
- Using these two parameters, DBSCAN categories the data points into three categories:
- *Core Points*: A data point *p* is a *core point* if **Nbhd**(*p*,ε) [ε-neighborhood of *p*] contains at least *minPts* ; |**Nbhd**(*p*,ε)| >= *minPts*.
- *Border Points: A data point *q* is a *border point* if **Nbhd**(*q*, ε) contains less than *minPts* data points, but *q* is *reachable* from some *core point p*.
- *Outlier*: A data point *o* is an *outlier* if it is neither a core point nor a border point. Essentially, this is the "other" class.



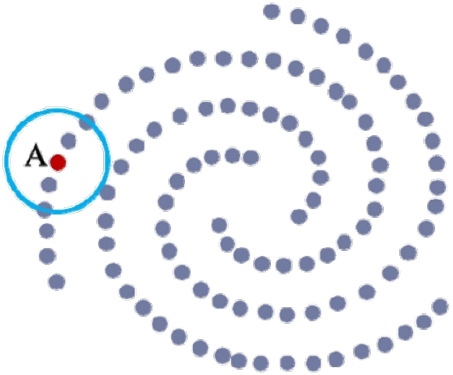
DBSCAN algorithm

1. Find the points in the ε (eps) neighborhood of every point, and identify the core points with more than the minimum number of points required to form a dense region (minPts) neighbors
2. Find the connected components of core points on the neighbor graph, ignoring all non-core points
3. Assign each non-core point to a nearby cluster if the cluster is an ε (eps) neighbor, otherwise assign it to noise

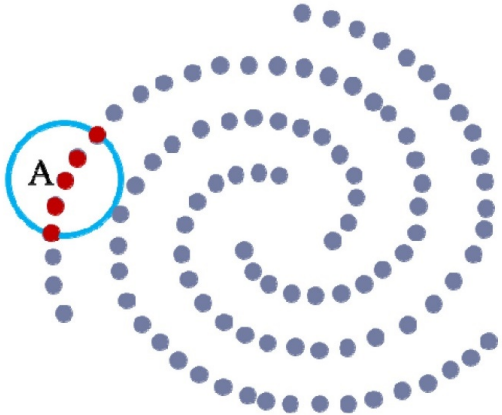
A naive implementation of this requires storing the neighborhoods in step 1, thus requiring substantial memory



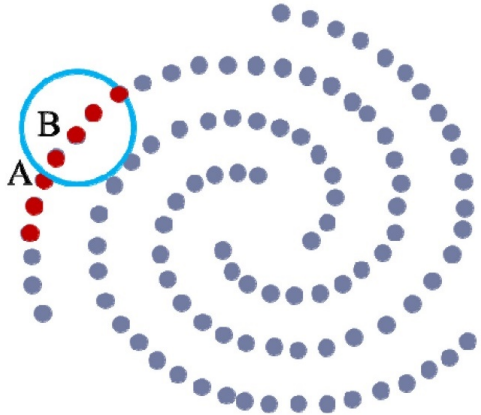
(a)



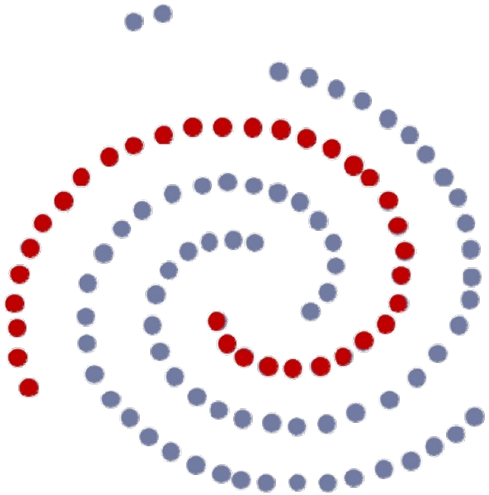
(b)



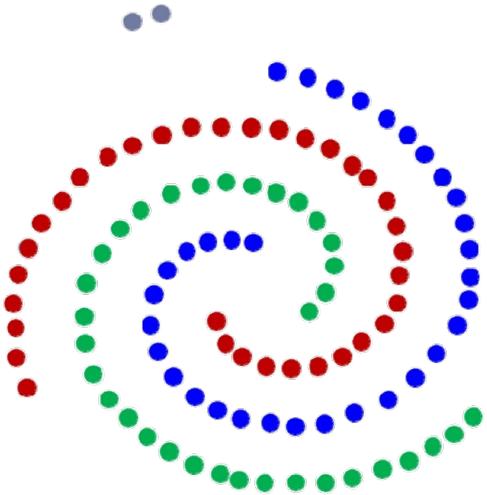
(c)



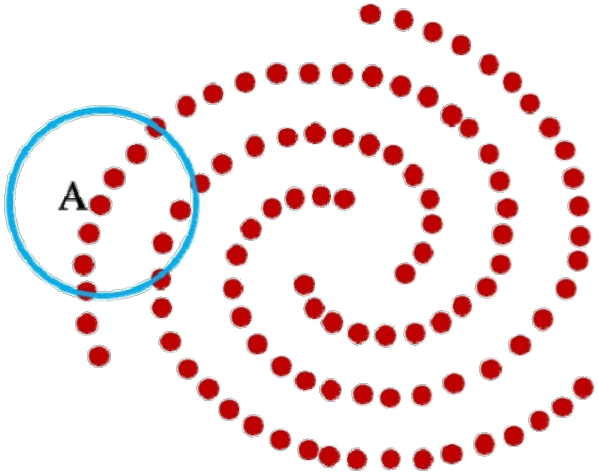
(d)



(e)



(f)



(g)

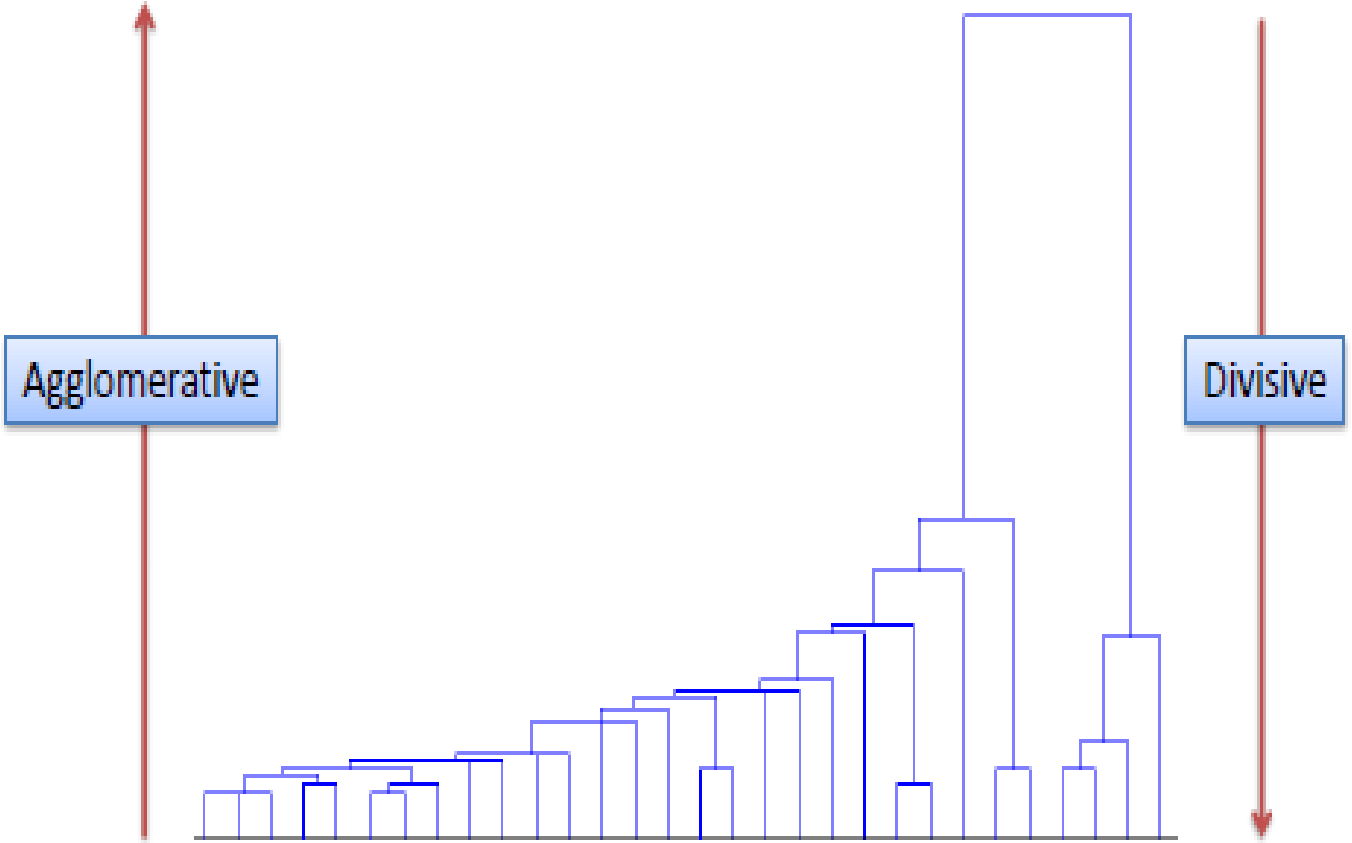
if set a larger ε

Hierarchical Clustering

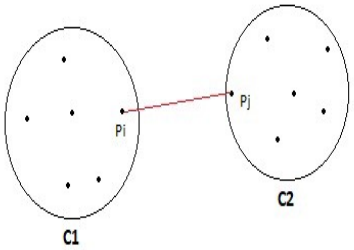
1. agglomerative clustering: bottom-up
   start: every node is a cluster
   repeat: merge similar nodes/clusters together (to a cluster)
   stop: all in one cluster
2. divisive clustering: top-down
   start: all in one cluster
   repeat: split a set to multiple clusters/nodes
   stop: every node is a cluster

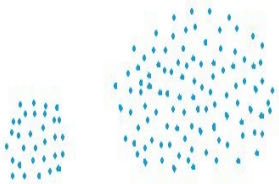
- **Another descriptions**
- Last but not the least are the hierarchical clustering algorithms. These algorithms have clusters sorted in an order based on the hierarchy in data similarity observations. Hierarchical clustering is categorised into two types, divisive(top-down) clustering and agglomerative (bottom-up) clustering.
- **Agglomerative Hierarchical clustering Technique:** In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.
- **Divisive Hierarchical clustering Technique:**Divisive Hierarchical clustering is exactly the opposite of the **Agglomerative Hierarchical clustering.** In Divisive Hierarchical clustering, we consider all the data points as a single cluster and in each iteration, we separate the data points from the cluster which are not similar. Each data point which is separated is considered as an individual cluster.
- Most of the hierarchical algorithms such as single linkage, complete linkage, average linkage, Ward's method, among others, follow the agglomerative approach.

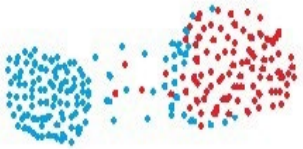
Hierarchical Clustering



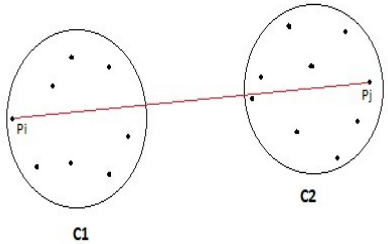
- Calculating the similarity between two clusters is important to merge or divide the clusters. There are certain approaches which are used to calculate the similarity between two clusters:
- **MIN:** Also known as single linkage algorithm can be defined as the similarity of two clusters C1 and C2 is equal to the **minimum** of the similarity between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.
- This approach can separate non-elliptical shapes as long as the gap between two clusters is not small.
- MIN approach cannot separate clusters properly if there is noise between clusters.

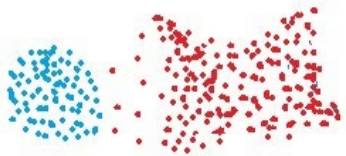


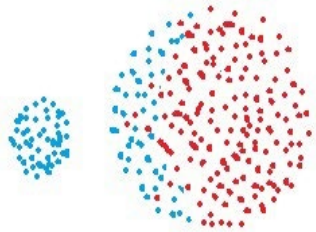




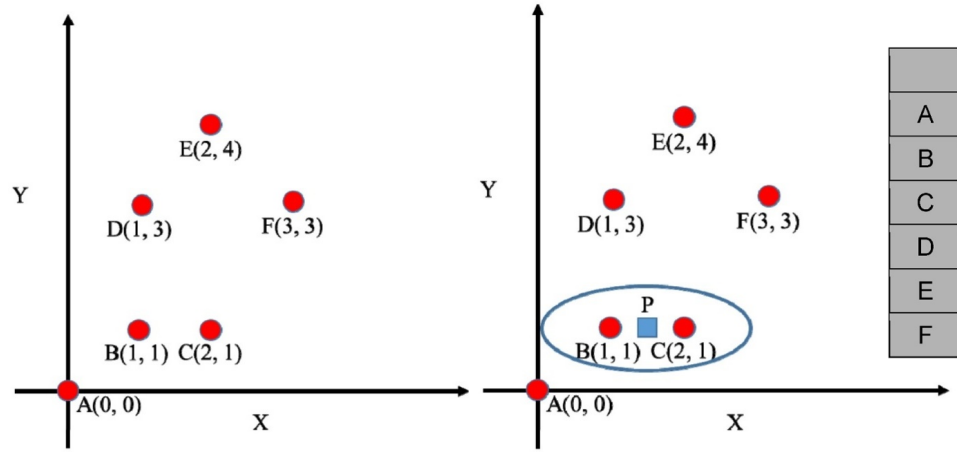
- **MAX:** Also known as the complete linkage algorithm, this is exactly opposite to the **MIN** approach. The similarity of two clusters C1 and C2 is equal to the **maximum** of the similarity between points Pi and Pj such that Pi belongs to C1 and Pj belongs to C2.
- MAX approach does well in separating clusters if there is noise between clusters but Max approach tends to break large clusters.







- **Group Average:** Take all the pairs of points and compute their similarities and calculate the average of the similarities.
- The group Average approach does well in separating clusters if there is noise between clusters but it is less popular technique in the real world.
- **Limitations of Hierarchical clustering Technique:**
- There is no mathematical objective for Hierarchical clustering.
- All the approaches to calculate the similarity between clusters has its own disadvantages.
- High space and time complexity for Hierarchical clustering. Hence this clustering algorithm cannot be used when we have huge data.



(a)

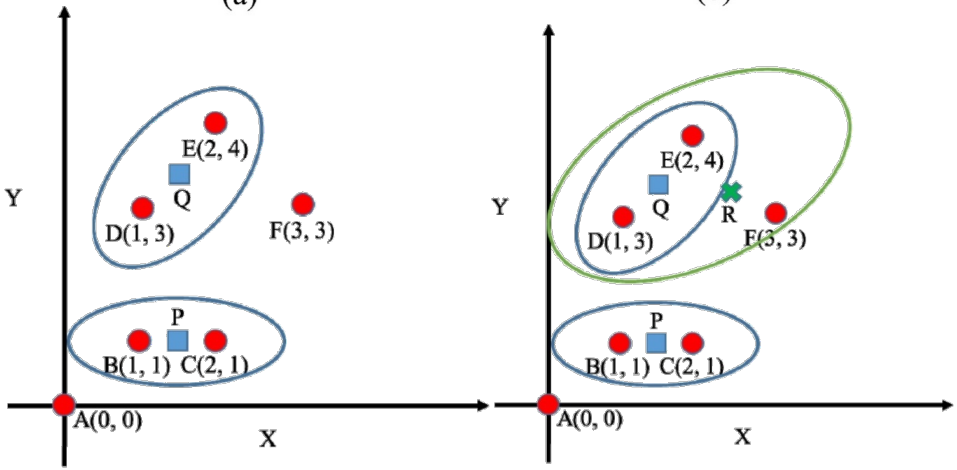
(b)

(a)

	A	B	C	D	E	F
A	0	1.4	2.2	3.2	4.5	42.
B		0	1	2	3.2	2.8
C			0	2.2	3	2.2
D				0	1.4	2
E					0	1.4
F						0

(b)

	A	D	E	F	P
A	0	3.2	4.5	4.2	1.8
D		0	1.4	2	2
E			0	1.4	3
F				0	2.5
P					0



(c)

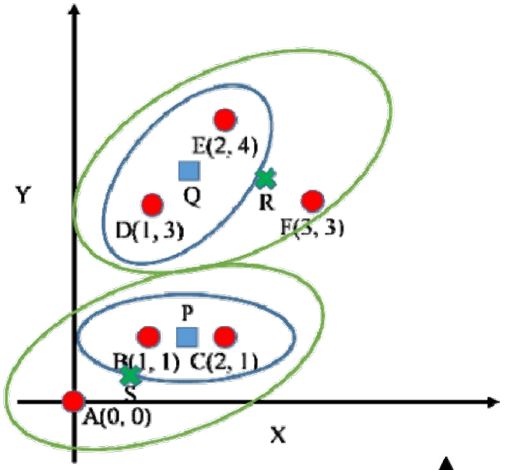
(d)

(c)

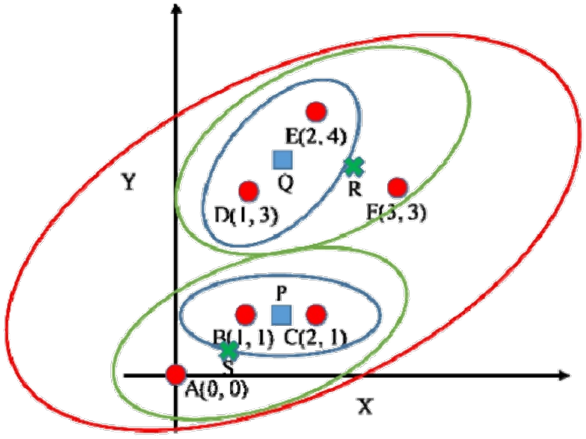
	A	F	P	Q
A	0	4.2	1.8	3.8
F		0	2.5	1.6
P			0	2.5
Q				0

(d)

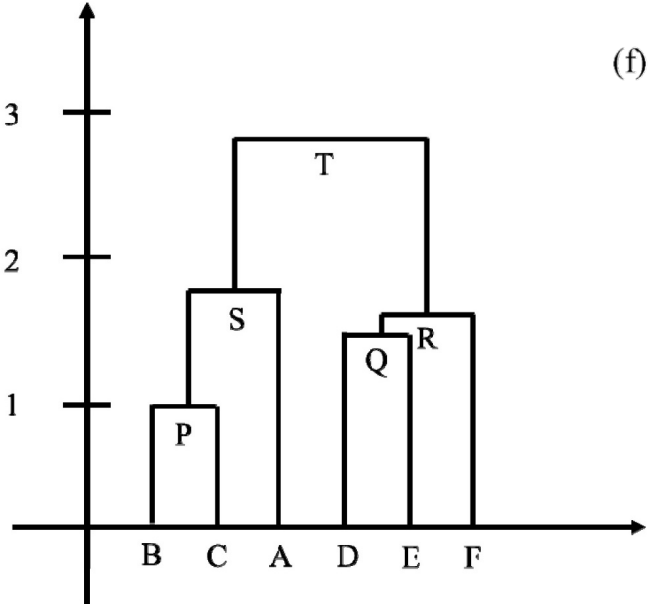
	A	P	R
A	0	1.8	3.9
P		0	2.3
Q			0


Y
E(2, 4)
Q
R
D(1, 3)
F(3, 3)
P
B(1, 1)
C(2, 1)
S
A(0, 0)
X

(e)


Y
E(2, 4)
Q
R
D(1, 3)
F(3, 3)
P
B(1, 1)
C(2, 1)
S
A(0, 0)
X

(f)


3
2
1
T
S
R
Q
P
B
C
A
D
E
F

Unsupervised learning: methodology in Python

- Select model, e.g., KMeans, DBSCAN, AgglomerativeClustering…
- Train model, i.e., select parameters
  - Data: input
    - training data
- Unsupervised learning model, e.g., clustering model
  - Data: input $\rightarrow$ result like clustering
- Production
  - Data: input $\rightarrow$ result like clustering