Lecture-25-27
Course: Applied Data Science

Clustering

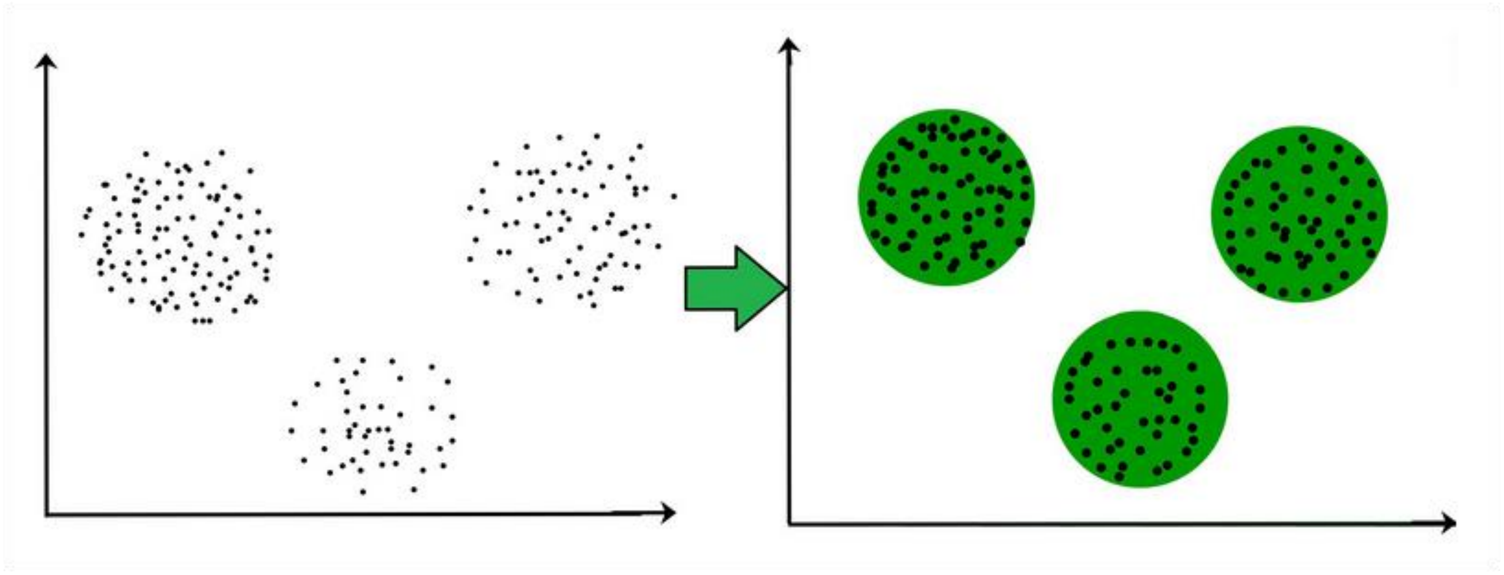
By
Dr. Sibarama Panigrahi
Senior Member, IEEE
Assistant Professor, Department of Computer Sc. & Engineering
National Institute of Technology, Rourkela, Odisha, 769008, India
Mobile No.: +91-7377302566
Email: panigrahis[at]nitrkl[dot]ac[dot]in
panigrahi[dot]sibarama[at]gmail[dot]com

Unsupervised Learning

- Supervised learning used labeled data pairs (x, y) to learn a function f : X→Y
  - But, what if we don't have labels?
- No labels = unsupervised learning
- Only some points are labeled = semi-supervised learning – Labels may be expensive to obtain, so we only get a few
- Clustering is the unsupervised grouping of data points. It can be used for knowledge discovery.

Clustering

- The task of grouping data points based on their similarity with each other is called Clustering or Cluster Analysis.
- Groups homogeneous data points from a heterogeneous dataset.
- 3 circular clusters forming on the basis of distance.



Types of Clustering

- **Hard Clustering** – data points completely belong to a cluster or not.

- **Soft clustering -** instead of assigning each data point into a separate cluster, a probability or likelihood of that point being that cluster is evaluated.

Data Points	Clusters
A	C1
B	C2
C	C2
D	C1

Data Points	Probability of C1	Probability of C2
A	0.91	0.09
B	0.3	0.7
C	0.17	0.83
D	1	0

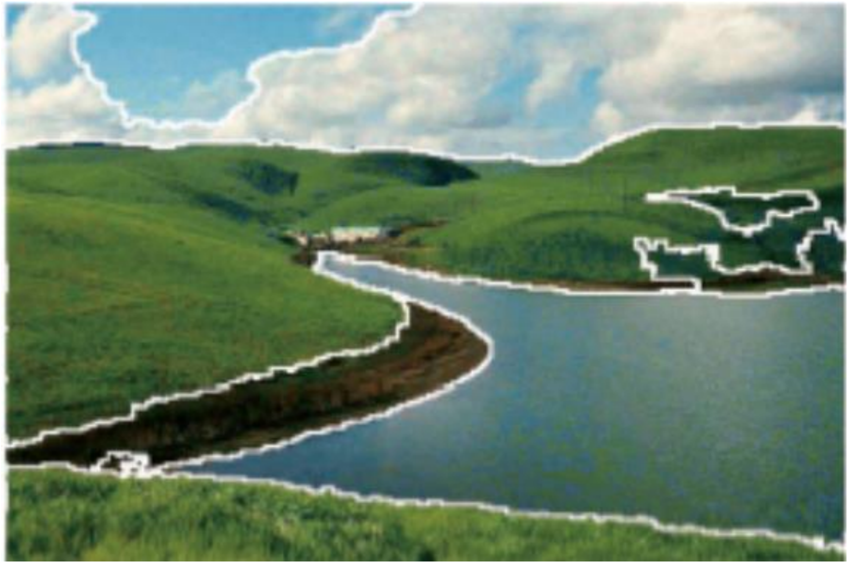
Types of Clustering Algorithms

- **Centroid-based Clustering** (Partitioning methods)
  - k-means
- **Density-based Clustering** (Model-based methods)
  - DBSCAN
- **Connectivity-based Clustering** (Hierarchical clustering)
  - Divisive and Agglomerative Clustering:
- **Distribution-based Clustering**
  - Gaussian Mixture Model.

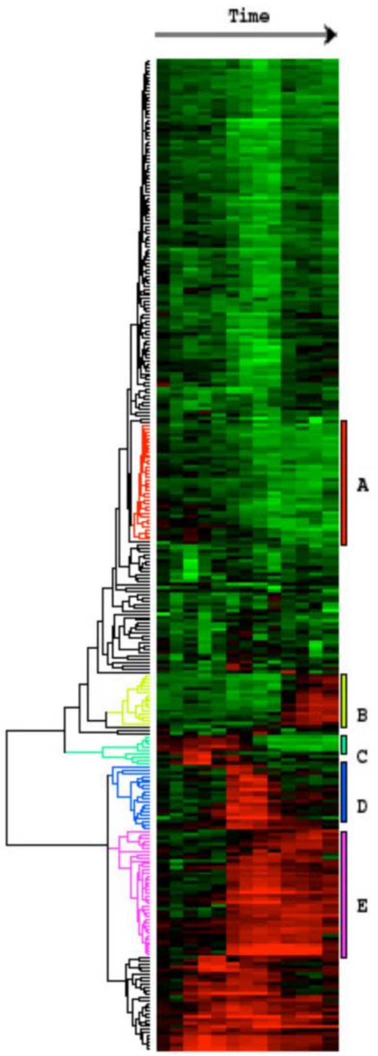
Clustering examples

Image segmentation

Goal: Break up the image into meaningful or perceptually similar regions

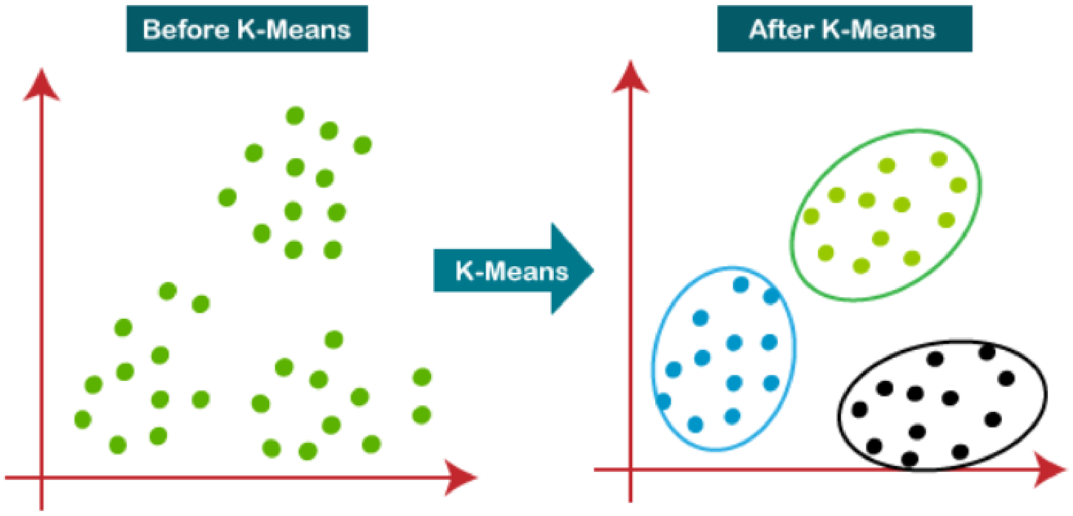


Clustering gene expression data

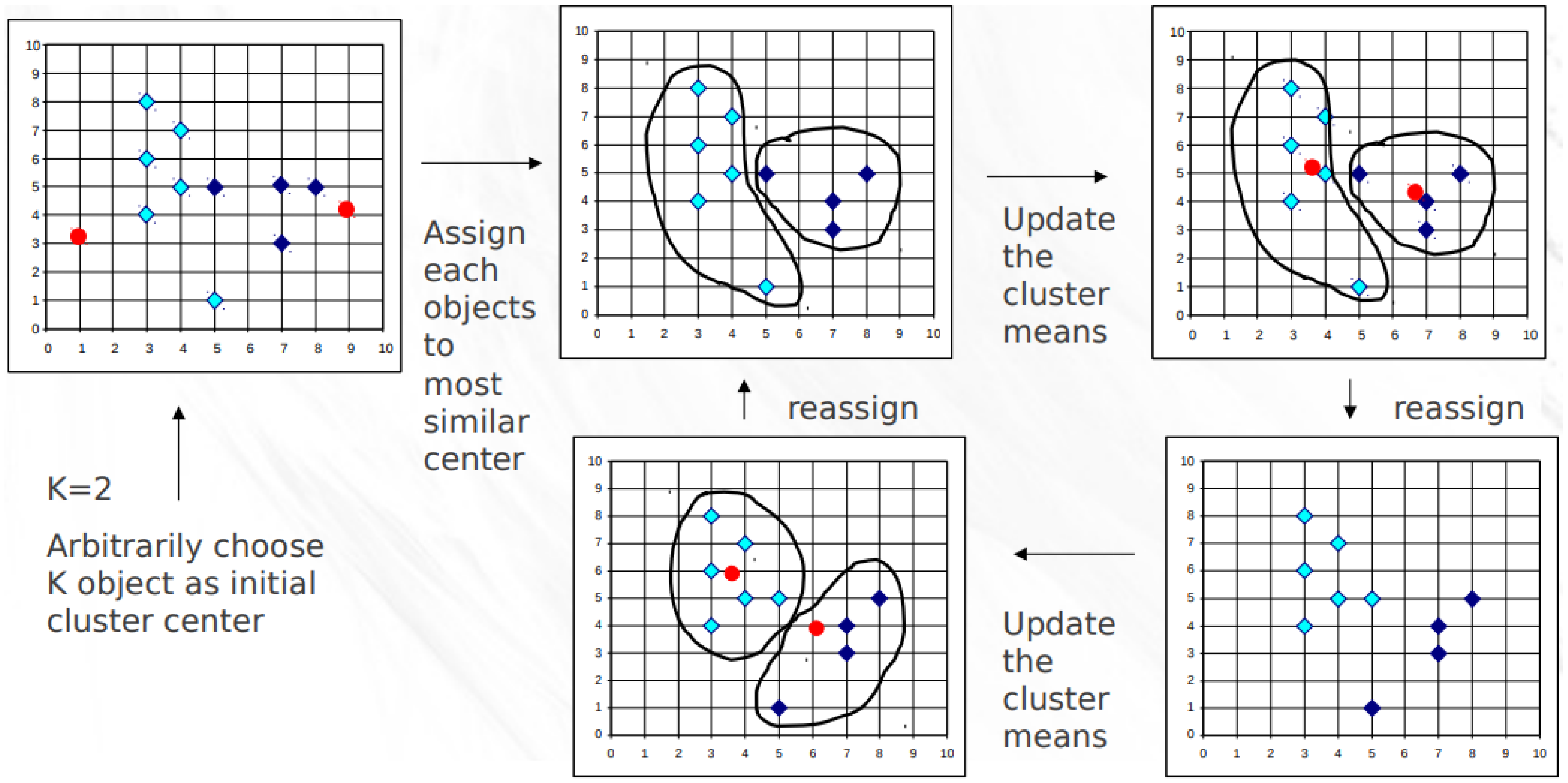


K-means clustering algorithm

- Performs 2 main tasks:
  - Determines the best value for K center points or centroids by an iterative process.
  - Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.
- Hence each cluster has datapoints with some commonalities, and it is away from other clusters



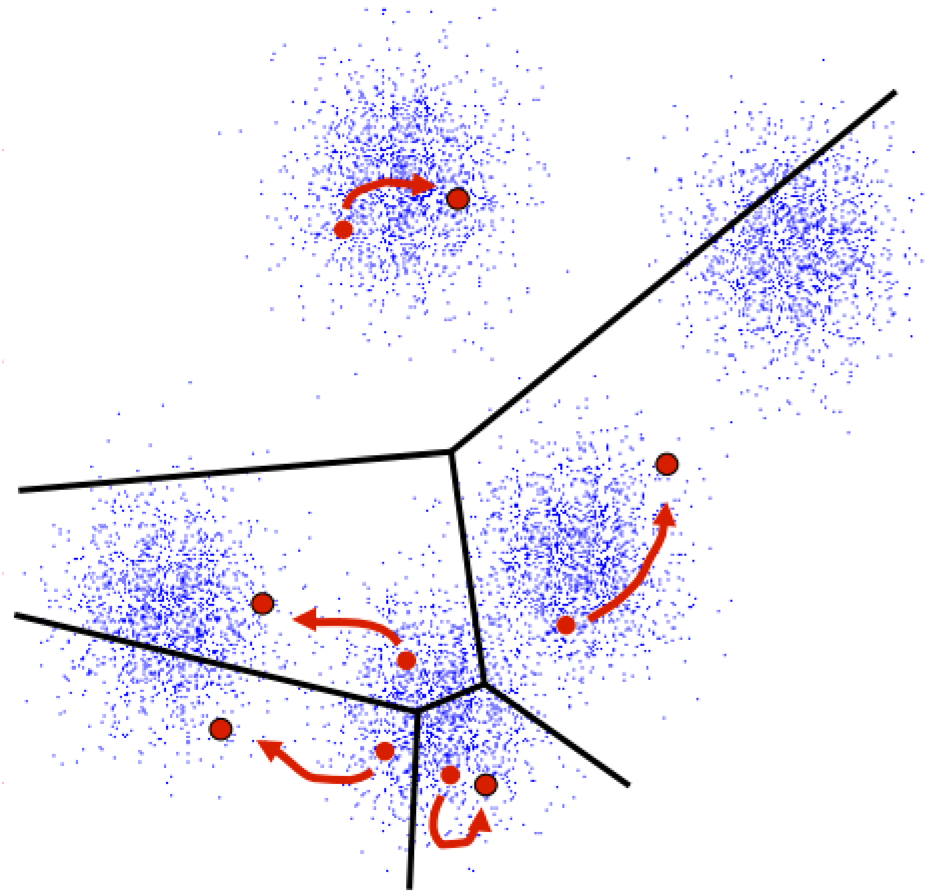
Procedure


K=2
Arbitrarily choose K object as initial cluster center
Assign each objects to most similar center
Update the cluster means
reassign
reassign
Update the cluster means

Procedure

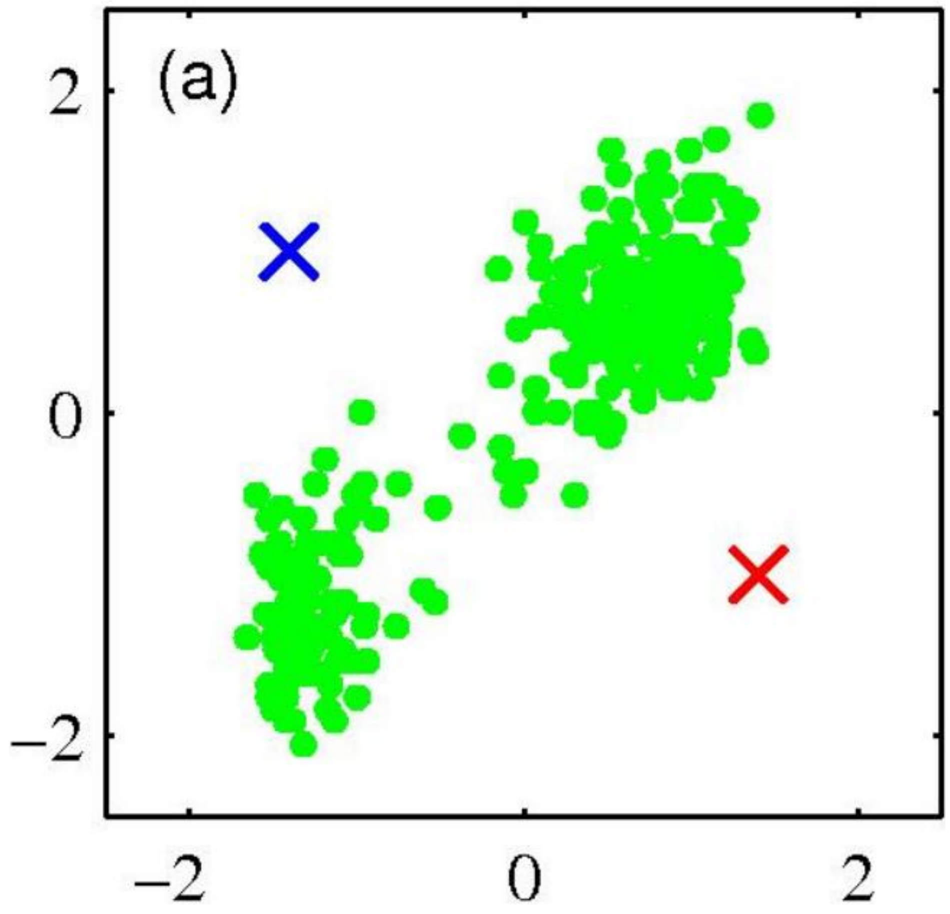
K-Means algorithm is explained in the below steps:

- **Initialize:** Pick *K* random points as cluster centers
- **Alternate:**
  1. Assign data points to closest cluster center
  2. Change the cluster center to the average of its assigned points
- **Stop when no points' assignments change**



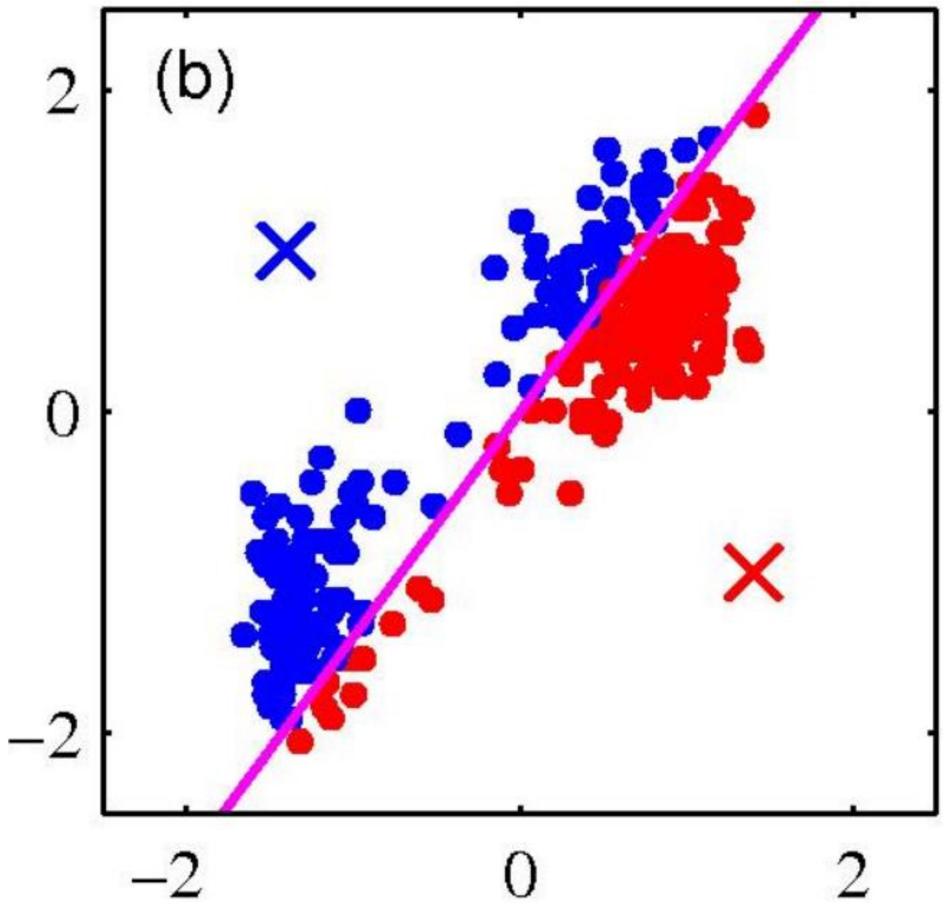
K-means clustering: Example

- Pick K random points as cluster centers (means) Shown here for K=2



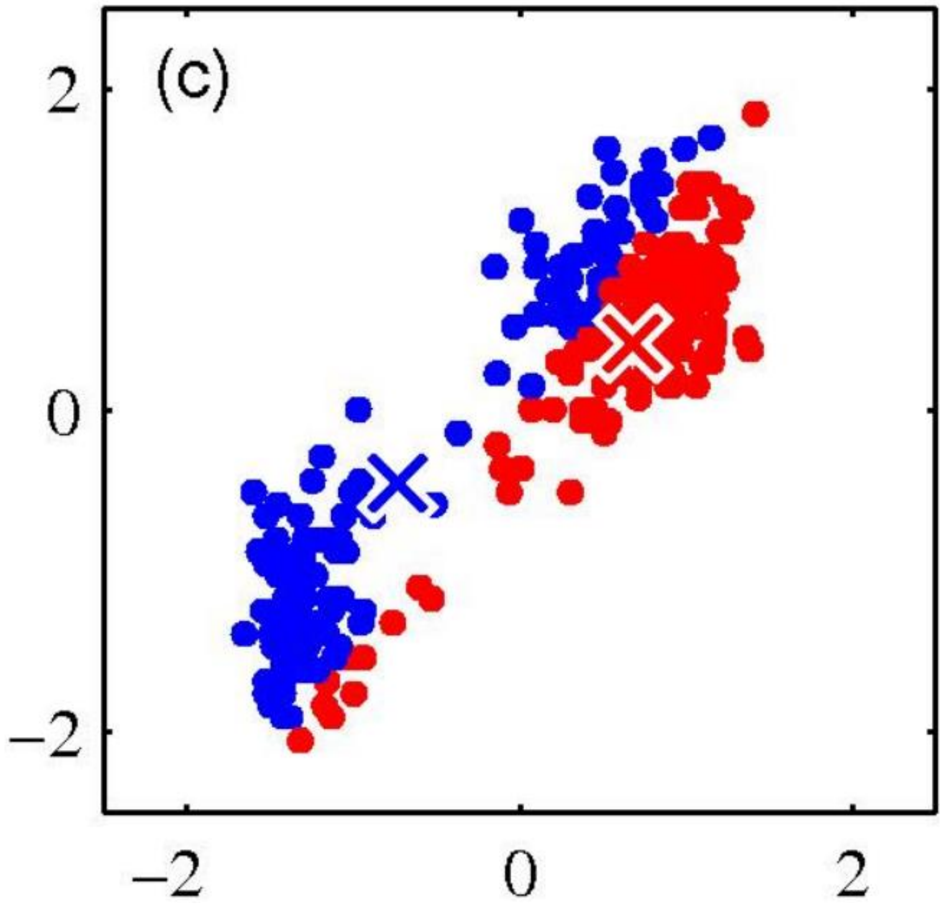
K-means clustering: Example

- Iterative Step 1
  - Assign data points to closest cluster center



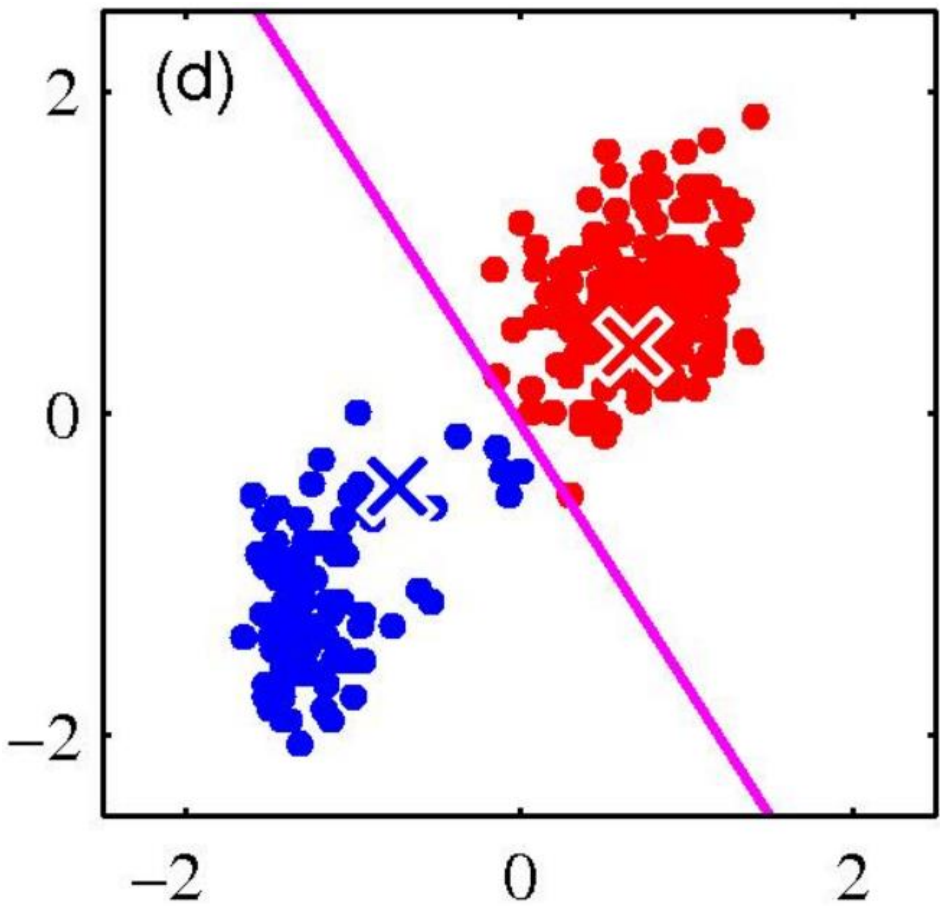
K-means clustering: Example

- Iterative Step 2
  - Change the cluster center to the average of the assigned points



K-means clustering: Example

- Repeat untill convergence



K-means objective function

- K-means finds a local optimum of the following objective function:

$$\arg\min_{\boldsymbol{\mathcal{S}}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{S}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2$$

where $\boldsymbol{\mathcal{S}} = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$ is a partitioning over

$X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ s.t. $X = \bigcup_{i=1}^{k} \mathcal{S}_i$

and $\boldsymbol{\mu}_i = \mathrm{mean}(\mathcal{S}_i)$

Properties of K-means algorithm

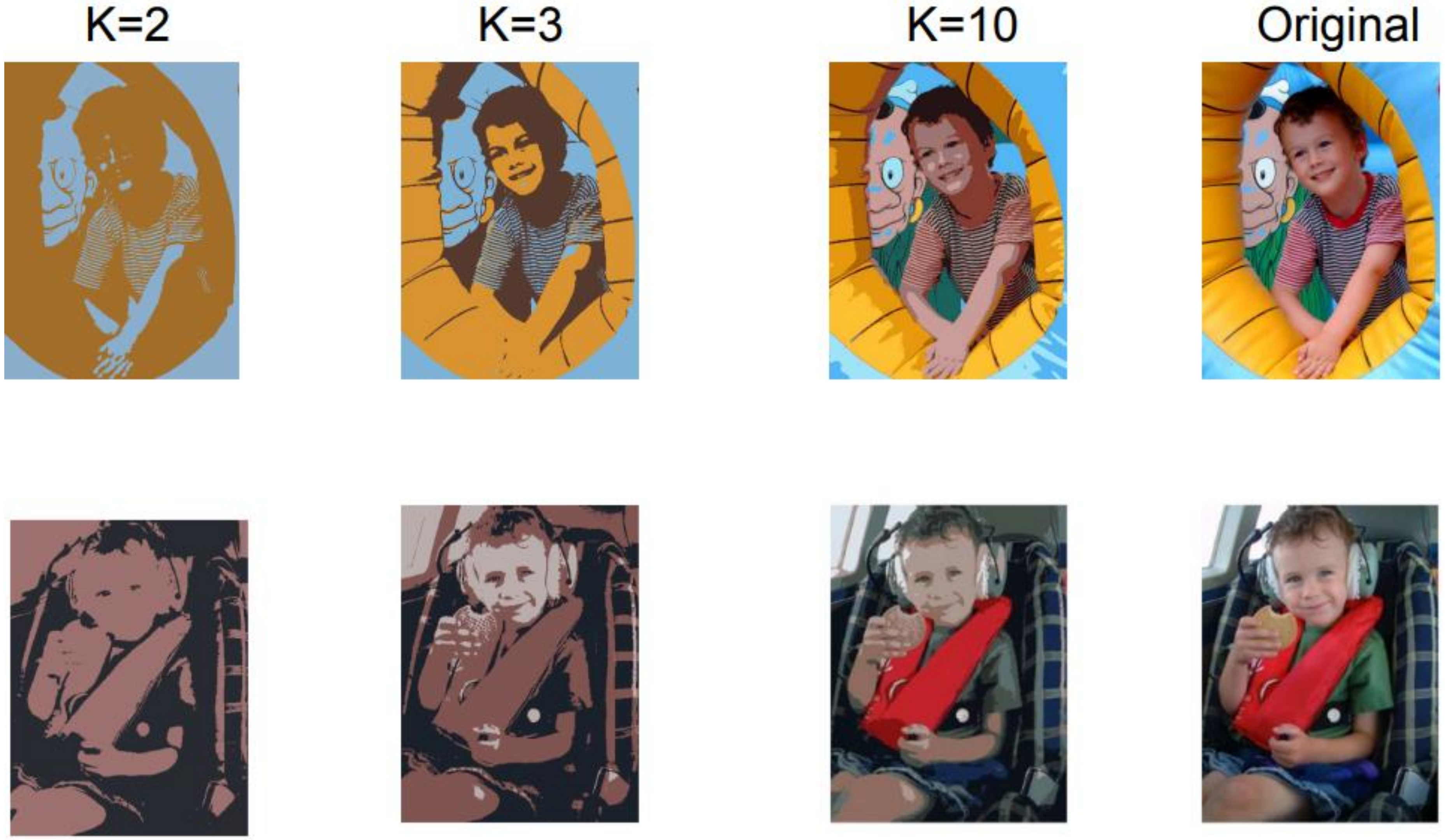
- Guaranteed to converge in a finite number of iterations

- Running time per iteration:
  - 1. Assign data points to closest cluster center **O(KN) time**
  - 2. Change the cluster center to the average of its assigned points **O(N)**

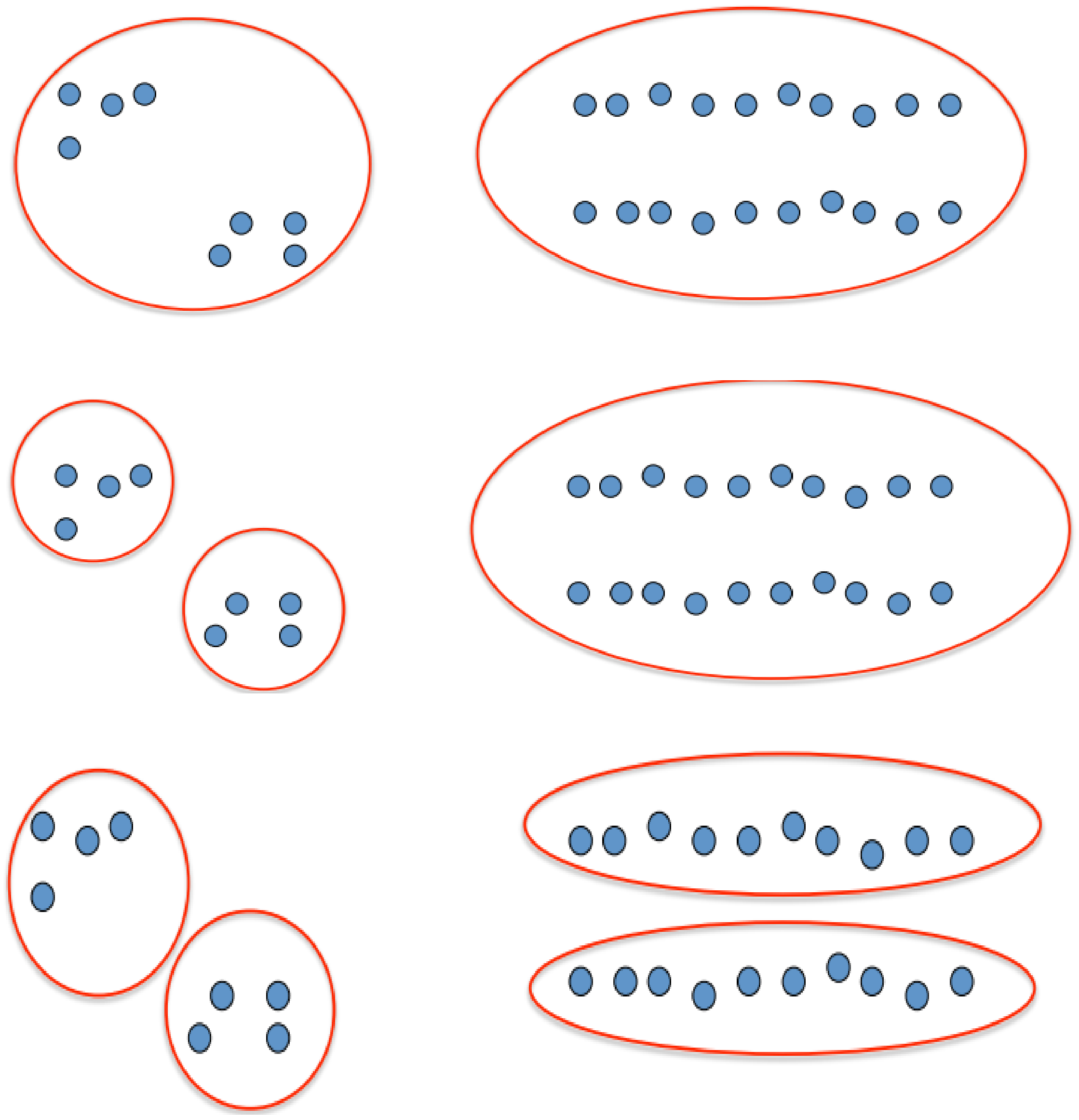
What properties should a distance measure have?

- Symmetric
  - D(A,B)=D(B,A)
  - Otherwise, we can say A looks like B but B does not look like A
- Positivity, and self-similarity
  - D(A,B)≥0, and D(A,B)=0 iff A=B
  - Otherwise there will different objects that we cannot tell apart
- Triangle inequality
  - D(A,B)+D(B,C) ≥ D(A,C)
  - Otherwise one can say “A is like B, B is like C, but A is not like C at all”

Example: K-Means for Segmentation



Impact of K value



K=2

K=3

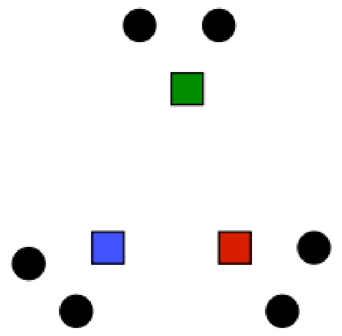
K=4

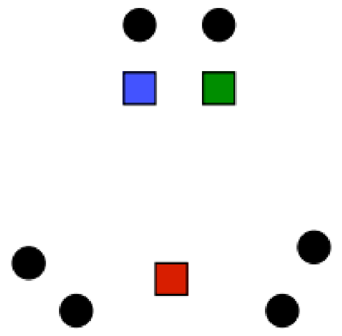
Impact of cluster Initialization

- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!

  - What can go wrong?

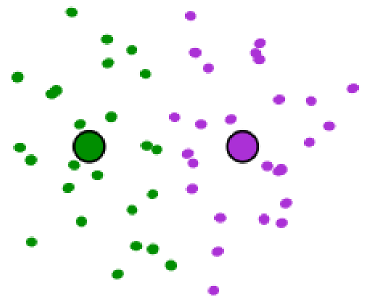
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



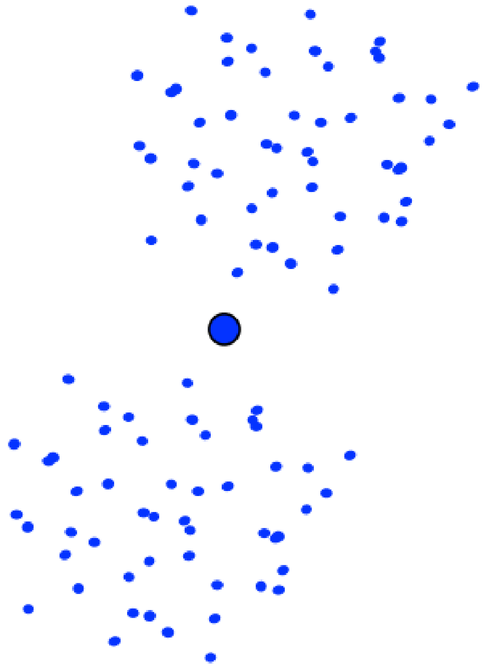


K-Means Getting Stuck

A local optimum:



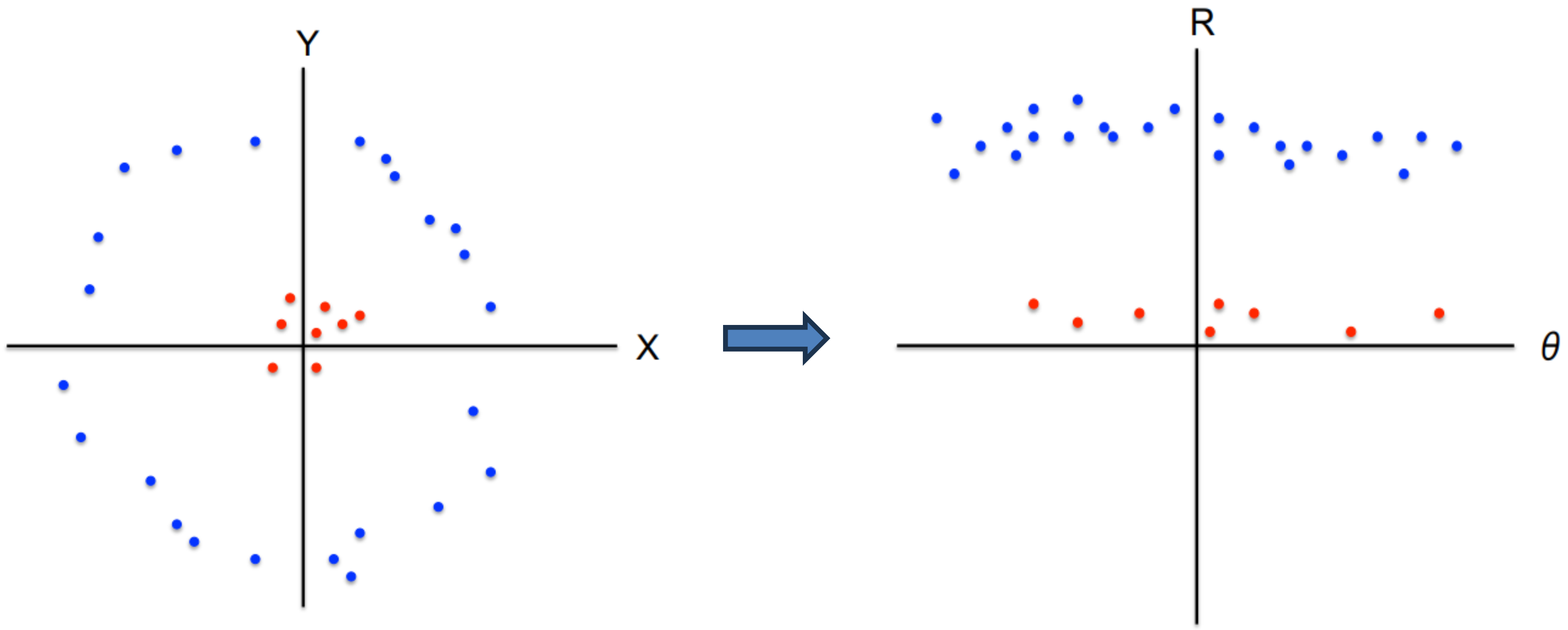
Would be better to have one cluster here



… and two clusters here

K-means not able to properly cluster

- Changing the features (distance function) can help



Numerical exercise

- For the given data, compute two clusters using K-means algorithm for clustering where initial cluster centers are (1.0, 1.0) and (5.0, 7.0). Execute for two iterations.

Record Number	A	B
R1	1.0	1.0
R2	1.5	2.0
R3	3.0	4.0
R4	5.0	7.0
R5	3.5	5.0
R6	4.5	5.0
R7	3.5	4.5

Solution:

- Initialization:
  - Number of clusters (K) = 2,
  - centroid for cluster1 (C1)= (1.0, 1.0) and
  - centroid for cluster2 (C2) = (5.0, 7.0).

- Use Euclidean distance to find closest point to centroids.
- Formula =

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Numerical exercise

Iteration1:

Record Number	**Close to C1**(1.0, 1.0)	**Close to C2**(5.0, 7.0)	**Assign to cluster**
R1(1.0,1.0)	dist(R1, C1)=0.0	dist(R1, C2)=7.21	Cluster1
R2(1.5,2.0)	dist(R2, C1)=1.12	dist(R2, C2)=6.12	Cluster1
R3(3.0,4.0)	dist(R3, C1)=3.61	dist(R3, C2)=3.61	Cluster1
R4(5.0,7.0)	dist(R4, C1)=7.21	dist(R4, C2)=0.0	Cluster2
R5(3.5,5.0)	dist(R5, C1)=4.12	dist(R5, C2)=2.5	Cluster2
R6(4.5,5.0)	dist(R6, C1)= 5.31	dist(R6, C2)=2.06	Cluster2
R7(3.5,4.5)	dist(R7,C1)=4.30	dist(R7, C2)=2.92	Cluster2

We obtain two clusters containing:

Cluster1 {R1, R2, R3} and

Cluster2 {R4, R5, R6, R7}.

Their new centroids are:

C1 = (1.0+1.5+3.0)/3, (1.0+2.0+4.0)/3

= 5.5/3, 7.0/3

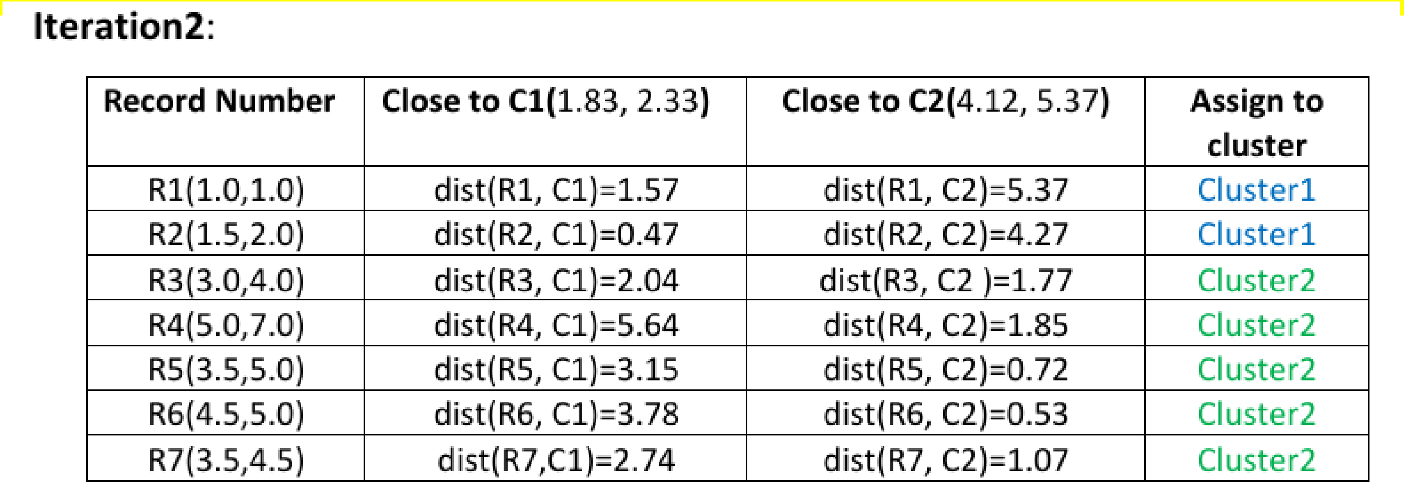
= 1.83, 2.33

C2 = (5.0+3.5+4.5+3.5)/4, (7+5+5+4.5)/4

= 16.5/4, 21.5/4

= 4.12, 5.37

Numerical exercise



Iteration2:

Record Number	**Close to C1**(1.83, 2.33)	**Close to C2**(4.12, 5.37)	**Assign to cluster**
R1(1.0,1.0)	dist(R1, C1)=1.57	dist(R1, C2)=5.37	Cluster1
R2(1.5,2.0)	dist(R2, C1)=0.47	dist(R2, C2)=4.27	Cluster1
R3(3.0,4.0)	dist(R3, C1)=2.04	dist(R3, C2)=1.77	Cluster2
R4(5.0,7.0)	dist(R4, C1)=5.64	dist(R4, C2)=1.85	Cluster2
R5(3.5,5.0)	dist(R5, C1)=3.15	dist(R5, C2)=0.72	Cluster2
R6(4.5,5.0)	dist(R6, C1)=3.78	dist(R6, C2)=0.53	Cluster2
R7(3.5,4.5)	dist(R7,C1)=2.74	dist(R7, C2)=1.07	Cluster2

Therefore, new clusters are:

Cluster1 {R1, R2} and

Cluster2 {R3, R4, R5, R6, R7}.

Their new centroids are:

C1 = (1.0+1.5)/2, (1.0+2.0)/2

= 2.50/2, 3.0/2

= 1.25, 1.5

C2 = (3.0+5.0+3.5+4.5+3.5)/5, (4+7+5+5+4.5)/5

= 19.5/5, 25.5/5

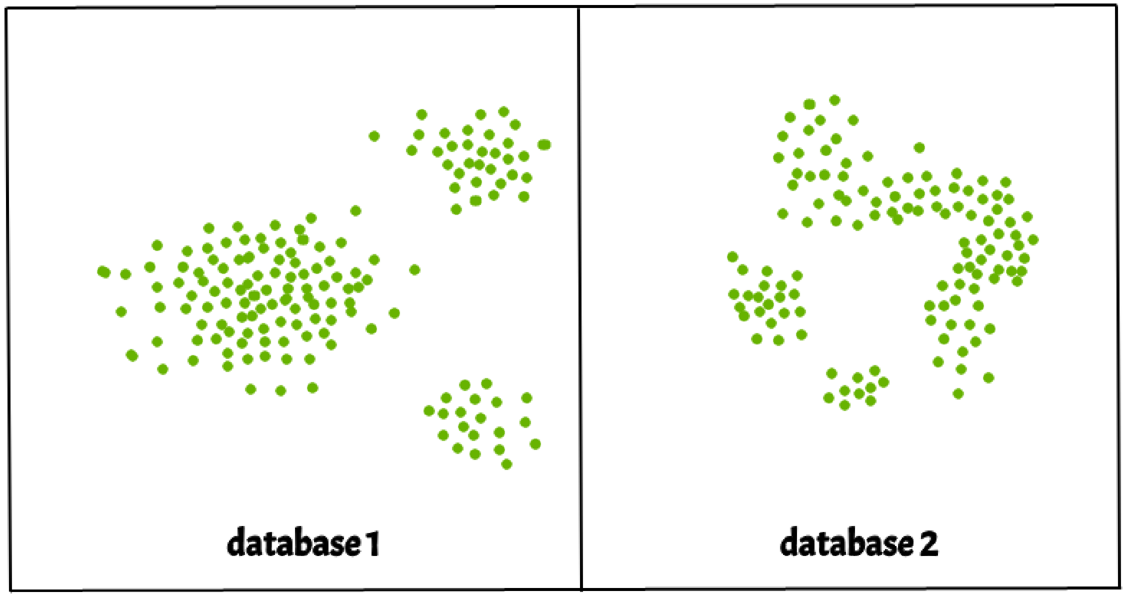
= 3.9, 5.1

DBSCAN: Ester, et al. (KDD'96)

- **DBSCAN**: Density-based spatial clustering of applications with noise
- It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.
- The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

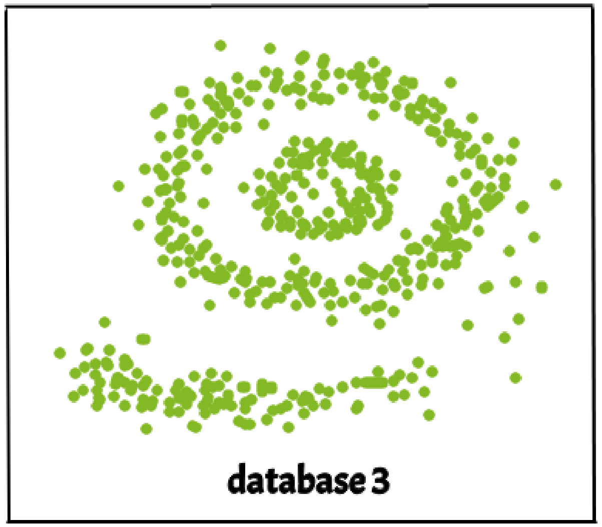
DBSCAN: Ester, et al. (KDD'96)

- **DBSCAN**: Density-based spatial clustering of applications with noise
- It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.
- The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.



Why DBSCAN?

- Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters.
- Real-life data may contain irregularities, like:
  - Clusters can be of arbitrary shape such as shown in below figure.
  - Data may contain noise.



The figure shows a data set containing non-convex shape clusters and outliers.

Parameters Required For DBSCAN Algorithm

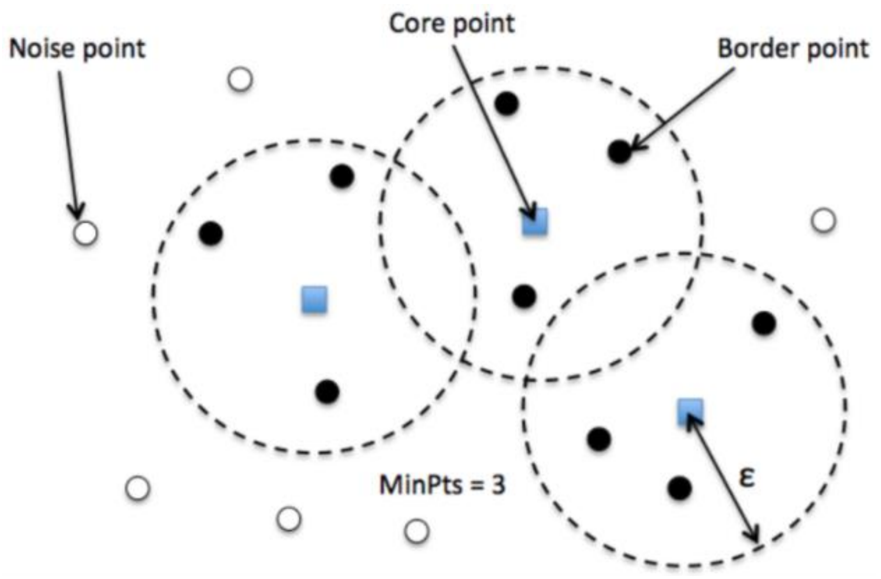
- **eps (ε):** If the distance between two points is lower or equal to 'eps' then they are considered neighbors.
  - If ε chosen too small then a large part of the data will be considered as an outlier.
  - If ε chosen very large then the clusters will merge majority of the data points.

- **Distance function**: The choice of distance function is critical and tightly linked to the choice of ε.

- **MinPts:** Minimum number of data points within ε radius.
  - The larger the dataset, the larger value of MinPts must be chosen.
  - The minimum value of MinPts must be chosen at least 3.
  - As a rule of thumb, a minimum minPts can be derived from the number of dimensions D in the data set, as minPts ≥ D + 1.

Types of data points

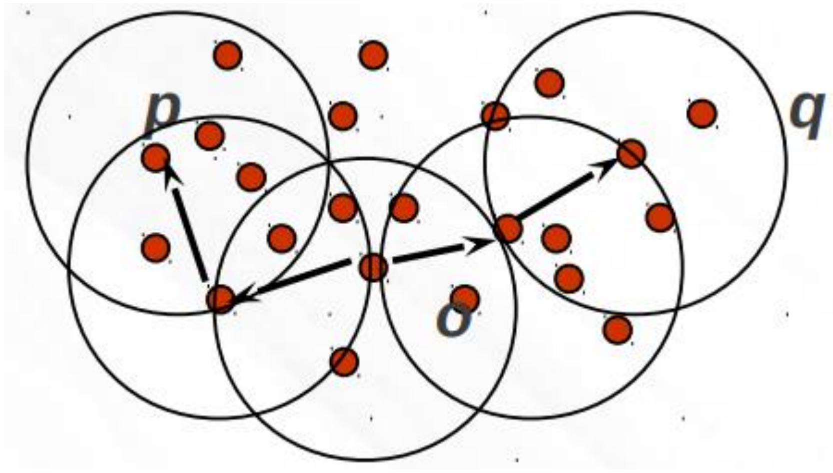
- **Core**: This is a point that has at least m points within distance n from itself.
- **Border**: This is a point that has at least one Core point at a distance n.
- **Noise**: This is a point that is neither a Core nor a Border. And it has less than m points within distance n from itself.



Density Reachability and Density Connectivity

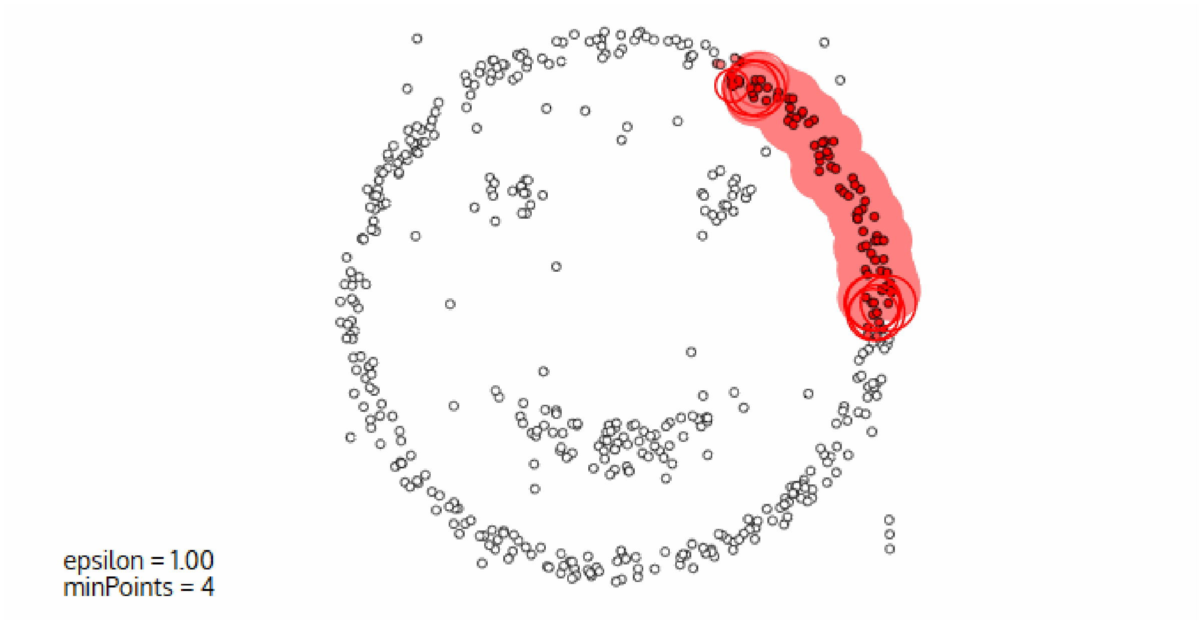
- **Reachability** in terms of density establishes a point to be reachable from another if it lies within a particular distance (eps) from it.

- **Connectivity**, on the other hand, involves a transitivity based chaining-approach to determine whether points are located in a particular cluster. For example, p and q points could be connected if p->r->s->t->q, where a->b means b is in the neighborhood of a.

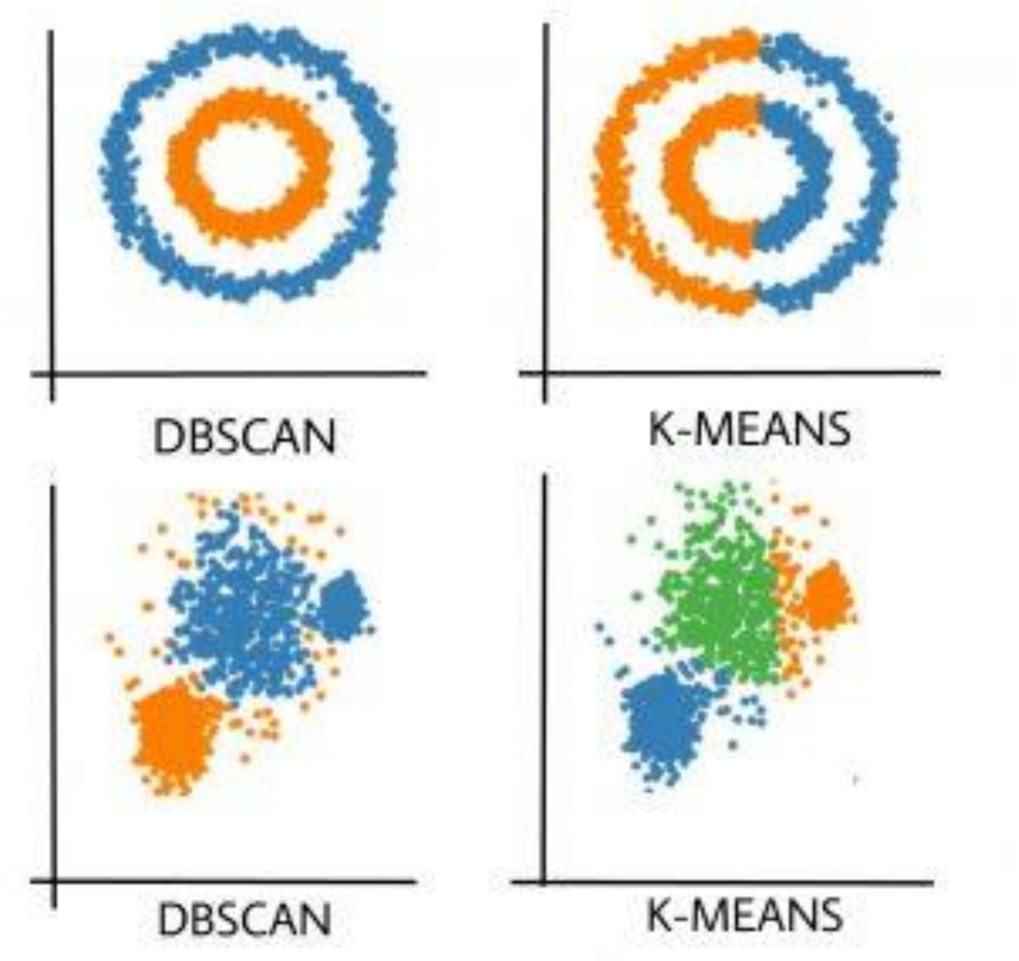


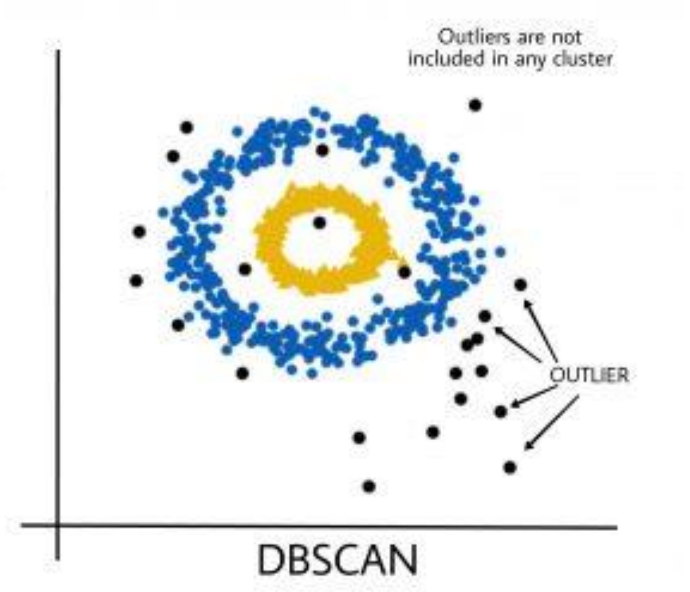
Algorithmic steps for DBSCAN clustering

1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new one.
3. Find recursively all its density-connected points and assign them to the same cluster as the core point.
4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

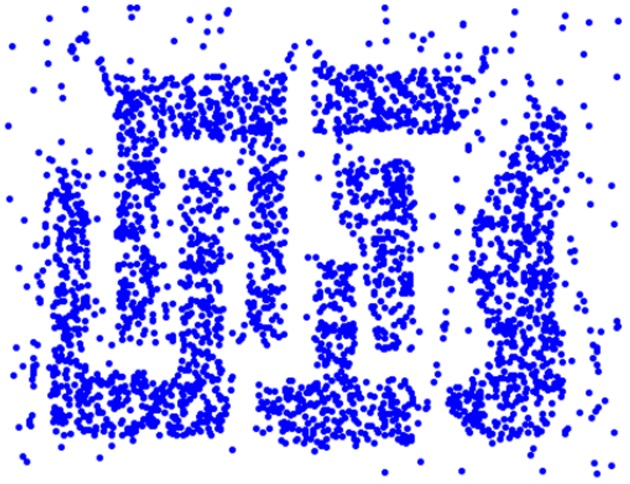


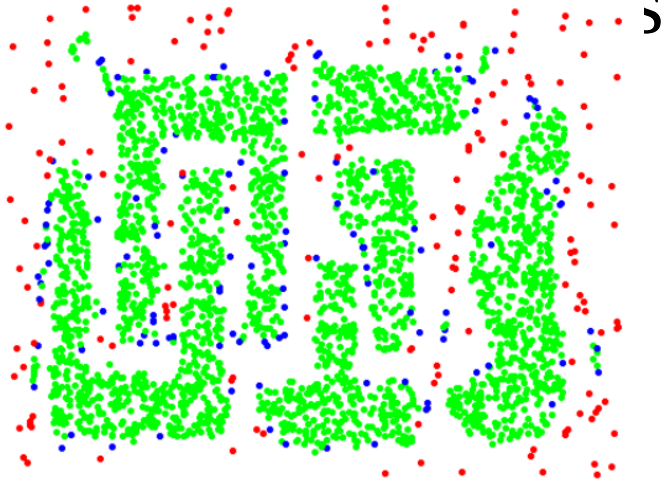
When Should We Use DBSCAN Over K-Means



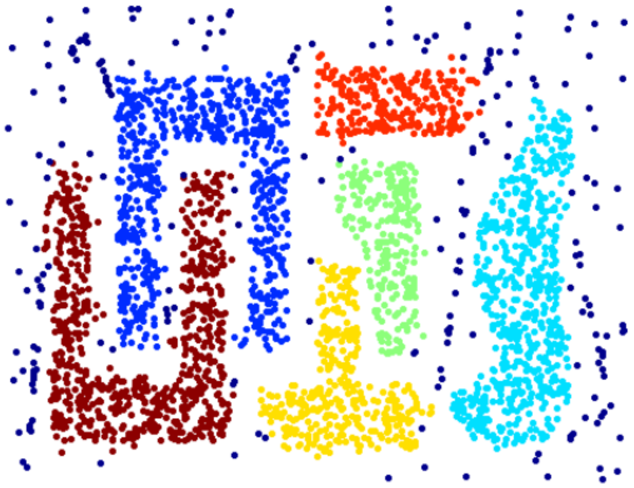


Impact of eps





Optimal eps



DBSCAN: Complexity

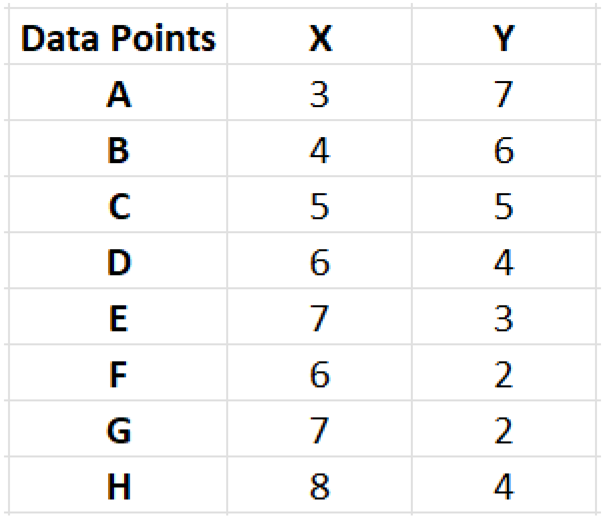
- **Time Complexity**: O(n^2)—for each point it has to be determined if it is a core point, can be reduced to O(n*log(n)) in lower dimensional spaces by using efficient data structures (n is the number of objects to be clustered);

- **Space Complexity**: O(n)

Numerical example

- **Question:** Given the points A(3, 7), B(4, 6), C(5, 5), D(6, 4), E(7, 3), F(6, 2), G(7, 2) and H(8, 4), Find the core points and outliers using DBSCAN. Take Eps = 2.5 and MinPts = 3.

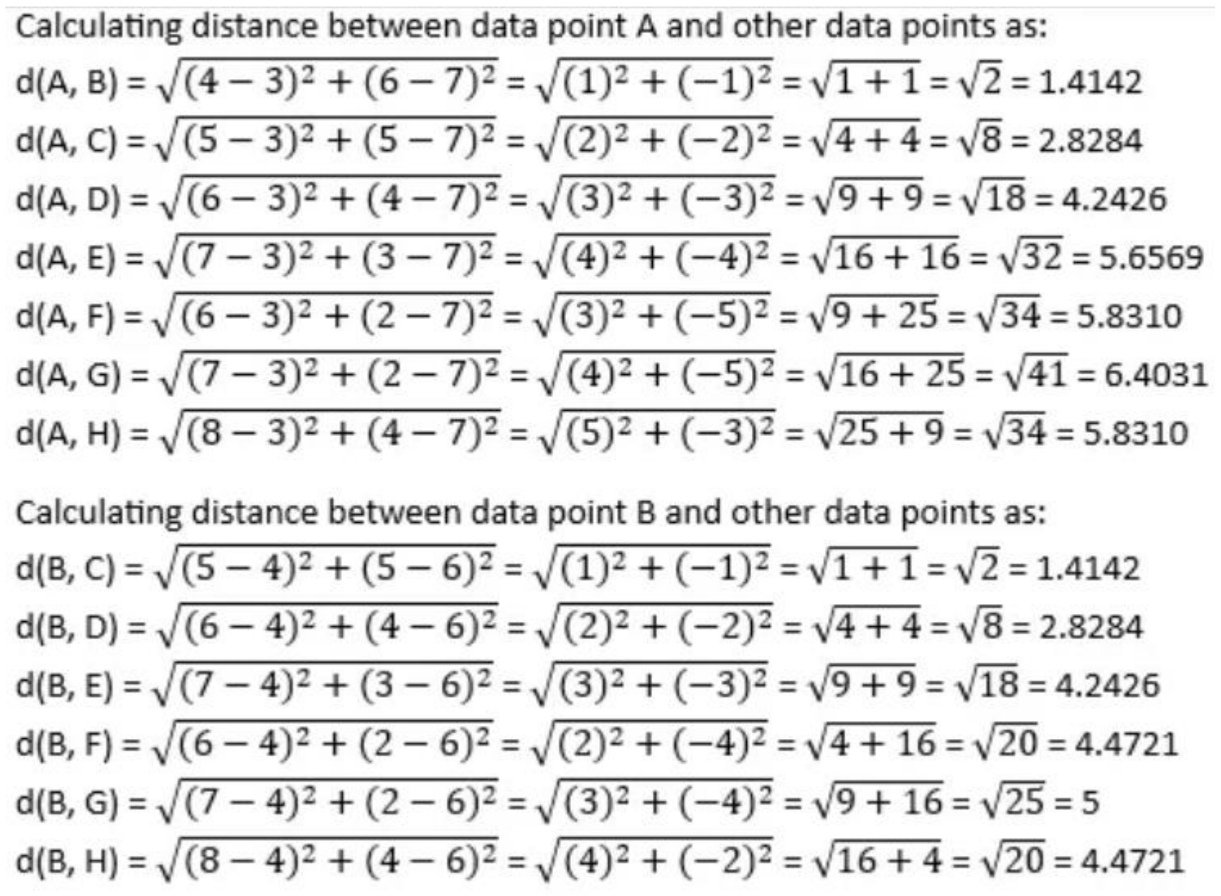
- **Solution:**
  Given, Epsilon(*Eps*) = 2.5
  Minimum Points(*MinPts*) = 3
  Let's represent the given data points in tabular form:



Data Points	X	Y
A	3	7
B	4	6
C	5	5
D	6	4
E	7	3
F	6	2
G	7	2
H	8	4

Numerical example

- Step 1: To find the core points, outliers and clusters by using DBSCAN we need to first calculate the distance among all pairs of given data point. Let us use Euclidean distance measure for distance calculation.



Calculating distance between data point A and other data points as:

$d(A, B) = \sqrt{(4-3)^2 + (6-7)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(A, C) = \sqrt{(5-3)^2 + (5-7)^2} = \sqrt{(2)^2 + (-2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

$d(A, D) = \sqrt{(6-3)^2 + (4-7)^2} = \sqrt{(3)^2 + (-3)^2} = \sqrt{9+9} = \sqrt{18} = 4.2426$

$d(A, E) = \sqrt{(7-3)^2 + (3-7)^2} = \sqrt{(4)^2 + (-4)^2} = \sqrt{16+16} = \sqrt{32} = 5.6569$

$d(A, F) = \sqrt{(6-3)^2 + (2-7)^2} = \sqrt{(3)^2 + (-5)^2} = \sqrt{9+25} = \sqrt{34} = 5.8310$

$d(A, G) = \sqrt{(7-3)^2 + (2-7)^2} = \sqrt{(4)^2 + (-5)^2} = \sqrt{16+25} = \sqrt{41} = 6.4031$

$d(A, H) = \sqrt{(8-3)^2 + (4-7)^2} = \sqrt{(5)^2 + (-3)^2} = \sqrt{25+9} = \sqrt{34} = 5.8310$

Calculating distance between data point B and other data points as:

$d(B, C) = \sqrt{(5-4)^2 + (5-6)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(B, D) = \sqrt{(6-4)^2 + (4-6)^2} = \sqrt{(2)^2 + (-2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

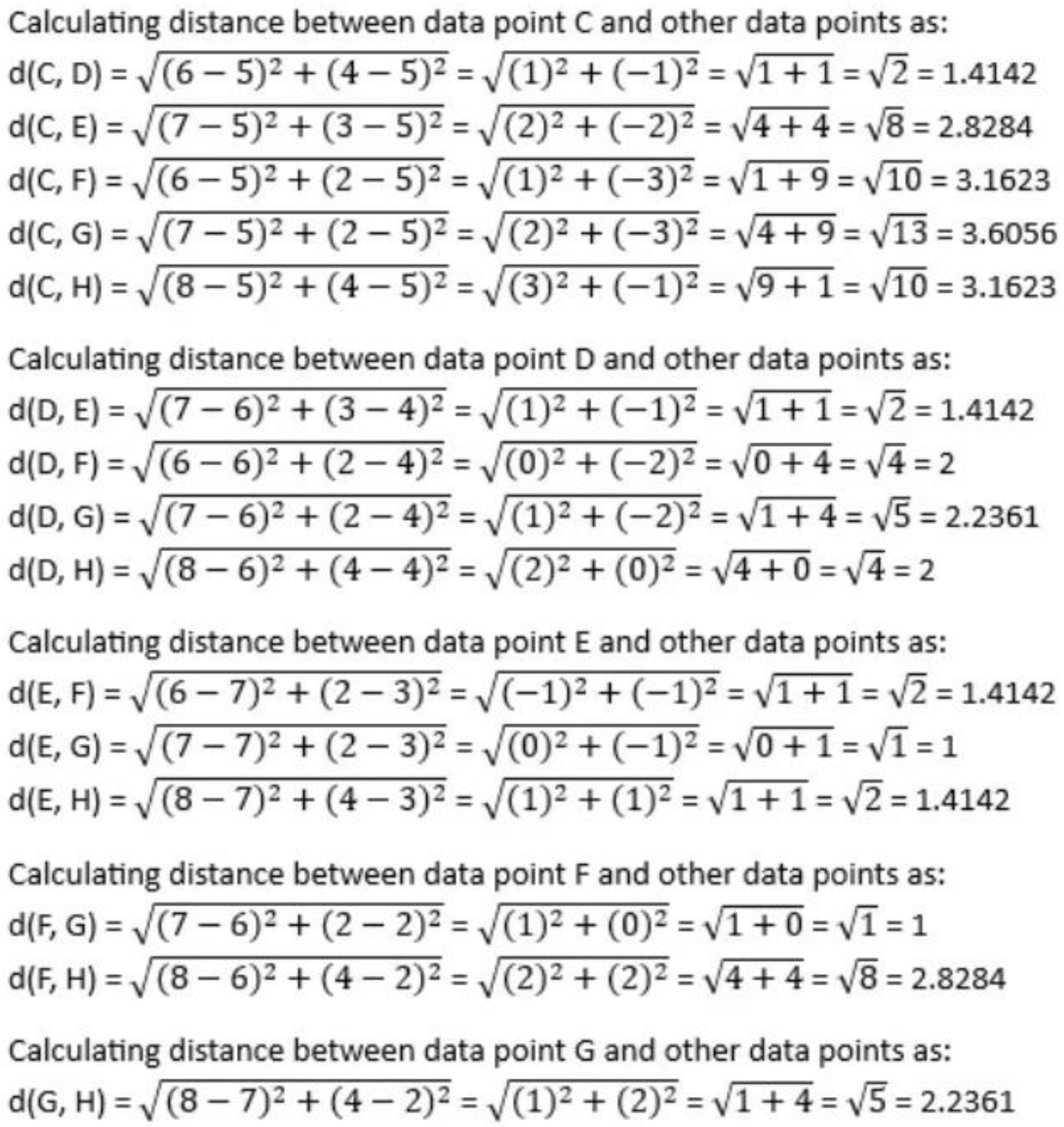
$d(B, E) = \sqrt{(7-4)^2 + (3-6)^2} = \sqrt{(3)^2 + (-3)^2} = \sqrt{9+9} = \sqrt{18} = 4.2426$

$d(B, F) = \sqrt{(6-4)^2 + (2-6)^2} = \sqrt{(2)^2 + (-4)^2} = \sqrt{4+16} = \sqrt{20} = 4.4721$

$d(B, G) = \sqrt{(7-4)^2 + (2-6)^2} = \sqrt{(3)^2 + (-4)^2} = \sqrt{9+16} = \sqrt{25} = 5$

$d(B, H) = \sqrt{(8-4)^2 + (4-6)^2} = \sqrt{(4)^2 + (-2)^2} = \sqrt{16+4} = \sqrt{20} = 4.4721$

Numerical example



Calculating distance between data point C and other data points as:

$d(C, D) = \sqrt{(6-5)^2 + (4-5)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(C, E) = \sqrt{(7-5)^2 + (3-5)^2} = \sqrt{(2)^2 + (-2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

$d(C, F) = \sqrt{(6-5)^2 + (2-5)^2} = \sqrt{(1)^2 + (-3)^2} = \sqrt{1+9} = \sqrt{10} = 3.1623$

$d(C, G) = \sqrt{(7-5)^2 + (2-5)^2} = \sqrt{(2)^2 + (-3)^2} = \sqrt{4+9} = \sqrt{13} = 3.6056$

$d(C, H) = \sqrt{(8-5)^2 + (4-5)^2} = \sqrt{(3)^2 + (-1)^2} = \sqrt{9+1} = \sqrt{10} = 3.1623$

Calculating distance between data point D and other data points as:

$d(D, E) = \sqrt{(7-6)^2 + (3-4)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(D, F) = \sqrt{(6-6)^2 + (2-4)^2} = \sqrt{(0)^2 + (-2)^2} = \sqrt{0+4} = \sqrt{4} = 2$

$d(D, G) = \sqrt{(7-6)^2 + (2-4)^2} = \sqrt{(1)^2 + (-2)^2} = \sqrt{1+4} = \sqrt{5} = 2.2361$

$d(D, H) = \sqrt{(8-6)^2 + (4-4)^2} = \sqrt{(2)^2 + (0)^2} = \sqrt{4+0} = \sqrt{4} = 2$

Calculating distance between data point E and other data points as:

$d(E, F) = \sqrt{(6-7)^2 + (2-3)^2} = \sqrt{(-1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(E, G) = \sqrt{(7-7)^2 + (2-3)^2} = \sqrt{(0)^2 + (-1)^2} = \sqrt{0+1} = \sqrt{1} = 1$

$d(E, H) = \sqrt{(8-7)^2 + (4-3)^2} = \sqrt{(1)^2 + (1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

Calculating distance between data point F and other data points as:

$d(F, G) = \sqrt{(7-6)^2 + (2-2)^2} = \sqrt{(1)^2 + (0)^2} = \sqrt{1+0} = \sqrt{1} = 1$

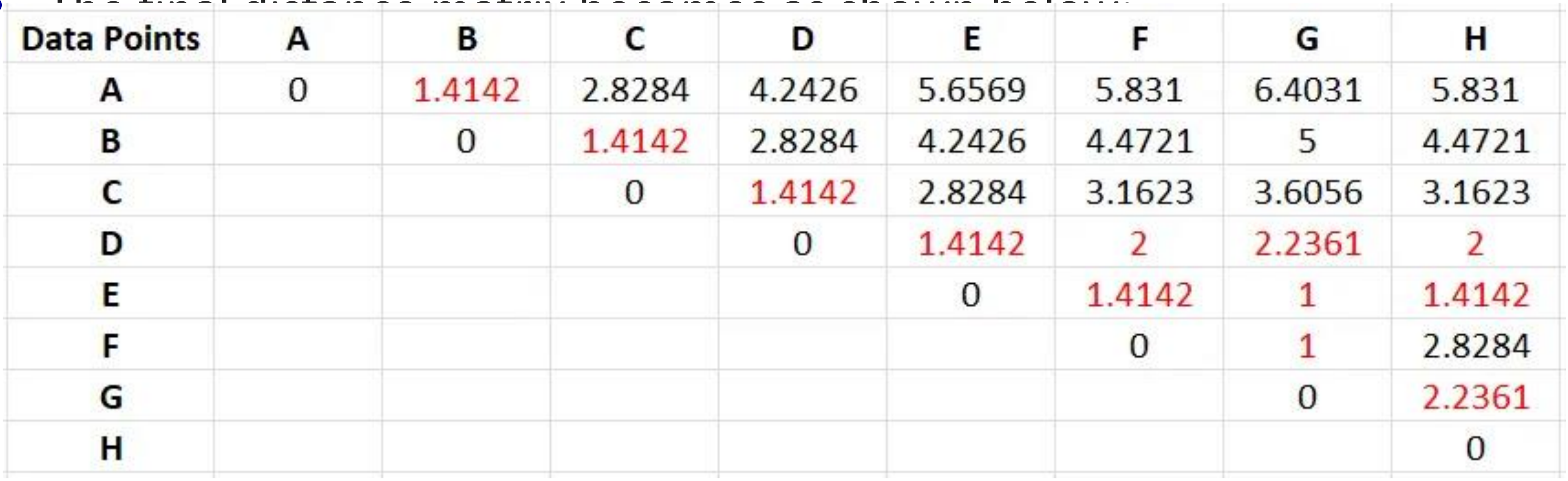
$d(F, H) = \sqrt{(8-6)^2 + (4-2)^2} = \sqrt{(2)^2 + (2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

Calculating distance between data point G and other data points as:

$d(G, H) = \sqrt{(8-7)^2 + (4-2)^2} = \sqrt{(1)^2 + (2)^2} = \sqrt{1+4} = \sqrt{5} = 2.2361$

Numerical example

- The final distance matrix becomes as shown below:



Data Points	A	B	C	D	E	F	G	H
A	0	1.4142	2.8284	4.2426	5.6569	5.831	6.4031	5.831
B		0	1.4142	2.8284	4.2426	4.4721	5	4.4721
C			0	1.4142	2.8284	3.1623	3.6056	3.1623
D				0	1.4142	2	2.2361	2
E					0	1.4142	1	1.4142
F						0	1	2.8284
G							0	2.2361
H								0

N(A) = {B}

N(B) = {A, C}
N(C) = {B, D}
N(D) = {C, E, F, G, H}
N(E) = {D, F, G, H}; N(F) = {D, E, G};
N(G) = {D, E, F, H};
N(H) = {D, E, G};

Data points D, E, F, G and H are the ***core data points***.

Data points A, B and C are not core points. However, they belong to the neighborhood of other core points, so they are **border points** and hence there exist ***no outliers*** in the given set of data points.

For Your Valuable Time.