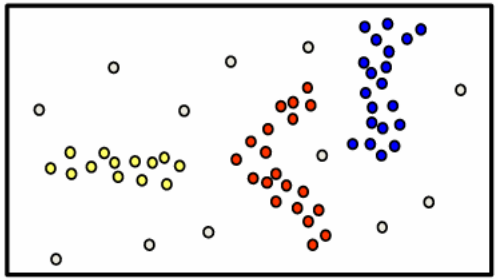
Density Based Clustering (DBSCAN: Density Based Spatial Clustering of Applications with Noise)

Density-based Clustering

- **Basic idea**
  - Clusters are dense regions in the data space, separated by regions of lower object density
  - A cluster is defined as a maximal set of density-connected points
  - Discovers clusters of arbitrary shape
- **Method**
  - DBSCAN

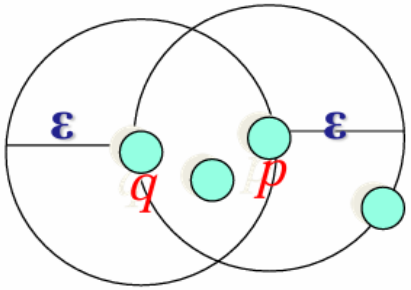


Density Definition

- ε-Neighborhood – Objects within a radius of ε from an object.

$$N_\varepsilon(p) : \{q \mid d(p,q) \le \varepsilon\}$$

- "High density" - ε-Neighborhood of an object contains at least *MinPts* of objects.

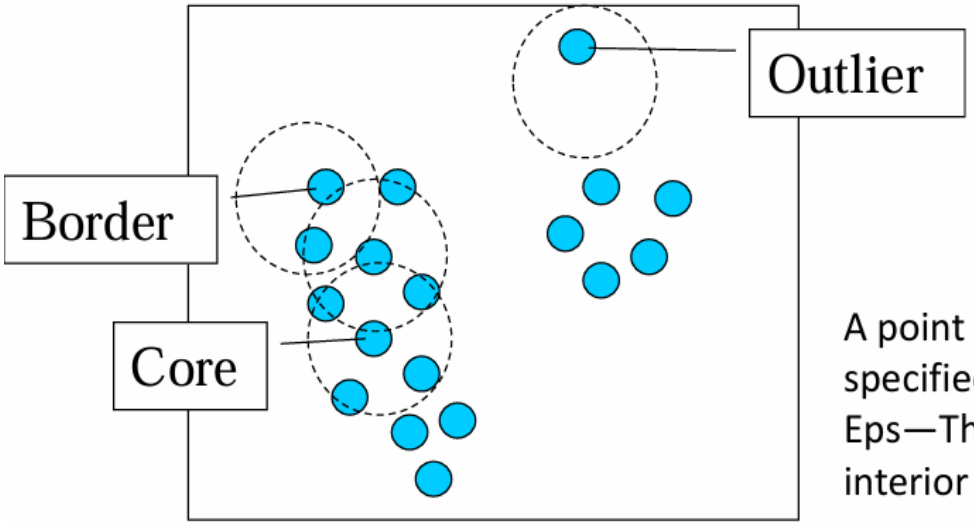


ε-Neighborhood of p
ε-Neighborhood of q

Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

Core, Border & Outlier



Given ε and *MinPts*, categorize the objects into three exclusive groups.

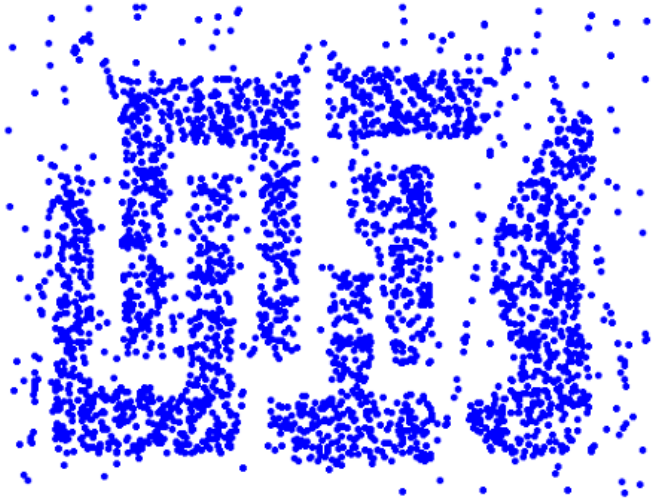
A point is a core point if it has more than a specified number of points (MinPts) within Eps—These are points that are at the interior of a cluster.

$\varepsilon = 1\text{unit}, \text{MinPts} = 5$

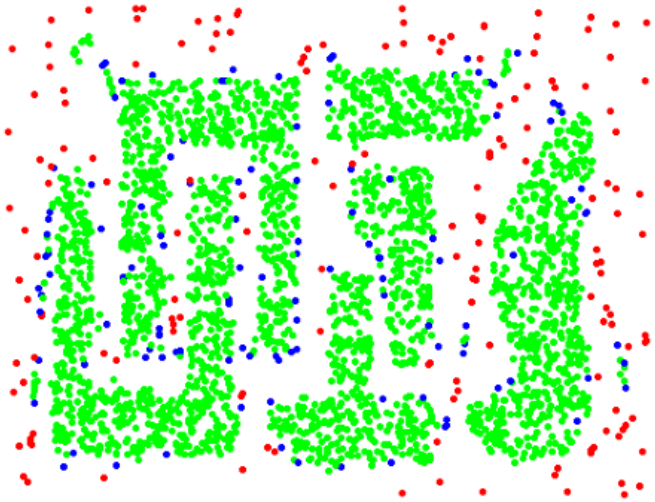
A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

Example



Original Points

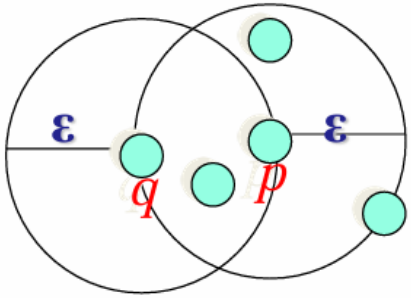


Point types: core, border and outliers

ε = 10, MinPts = 4

Density-reachability

- Directly density-reachable
  - An object *q* is directly density-reachable from object *p* if *p* is a core object and *q* is in p's ε-neighborhood.

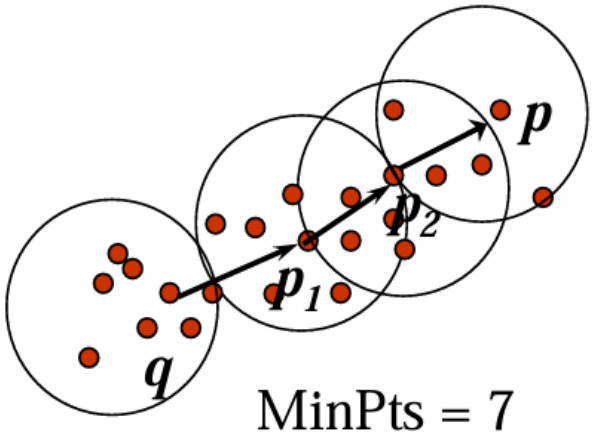


MinPts = 4

- *q* is directly density-reachable from *p*
- *p* is not directly density-reachable from *q*
- Density-reachability is asymmetric

Density-reachability

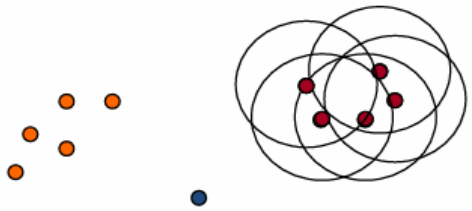
- Density-Reachable (directly and indirectly):
  - A point p is directly density-reachable from p_2
  - p_2 is directly density-reachable from p_1
  - p_1 is directly density-reachable from q
  - $p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a chain



- p is (indirectly) density-reachable from q
- q is not density-reachable from p

DBSCAN Algorithm: Example

- **Parameter**
  - ε = 2 cm
  - *MinPts* = 3



```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

DBSCAN: Algorithm

Let ClusterCount=0. For every point *p*:

1. If *p* it is not a core point, assign a null label to it [e.g., zero]
2. If *p* is a core point, a new cluster is formed [with label ClusterCount:= ClusterCount+1]

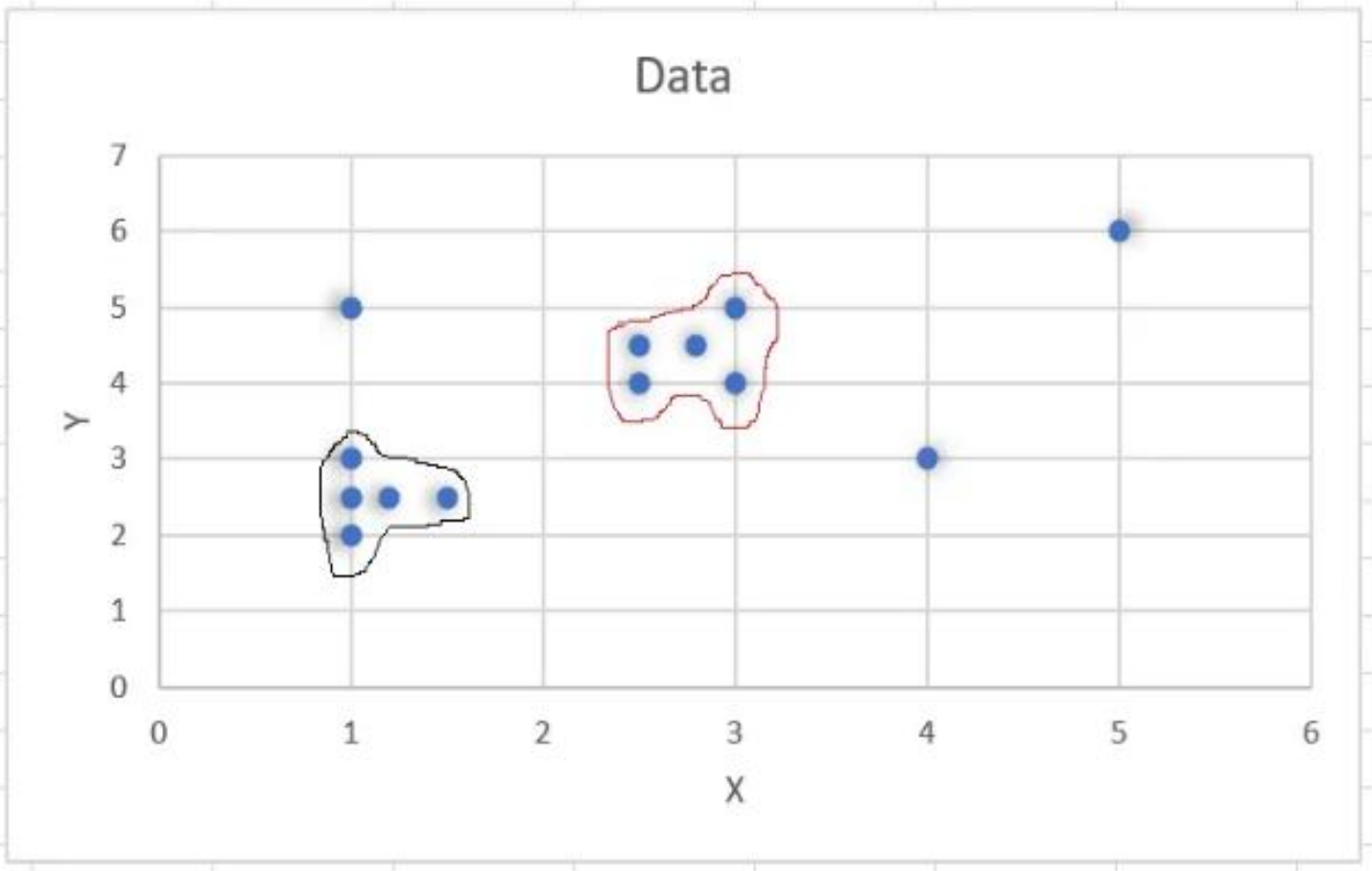
   Then find all points density-reachable from p and classify them in the cluster.
   [Reassign the zero labels but not the others]

Repeat this process until all of the points have been visited.

Since all the zero labels of border points have been reassigned in 2, the remaining points with zero label are noise.

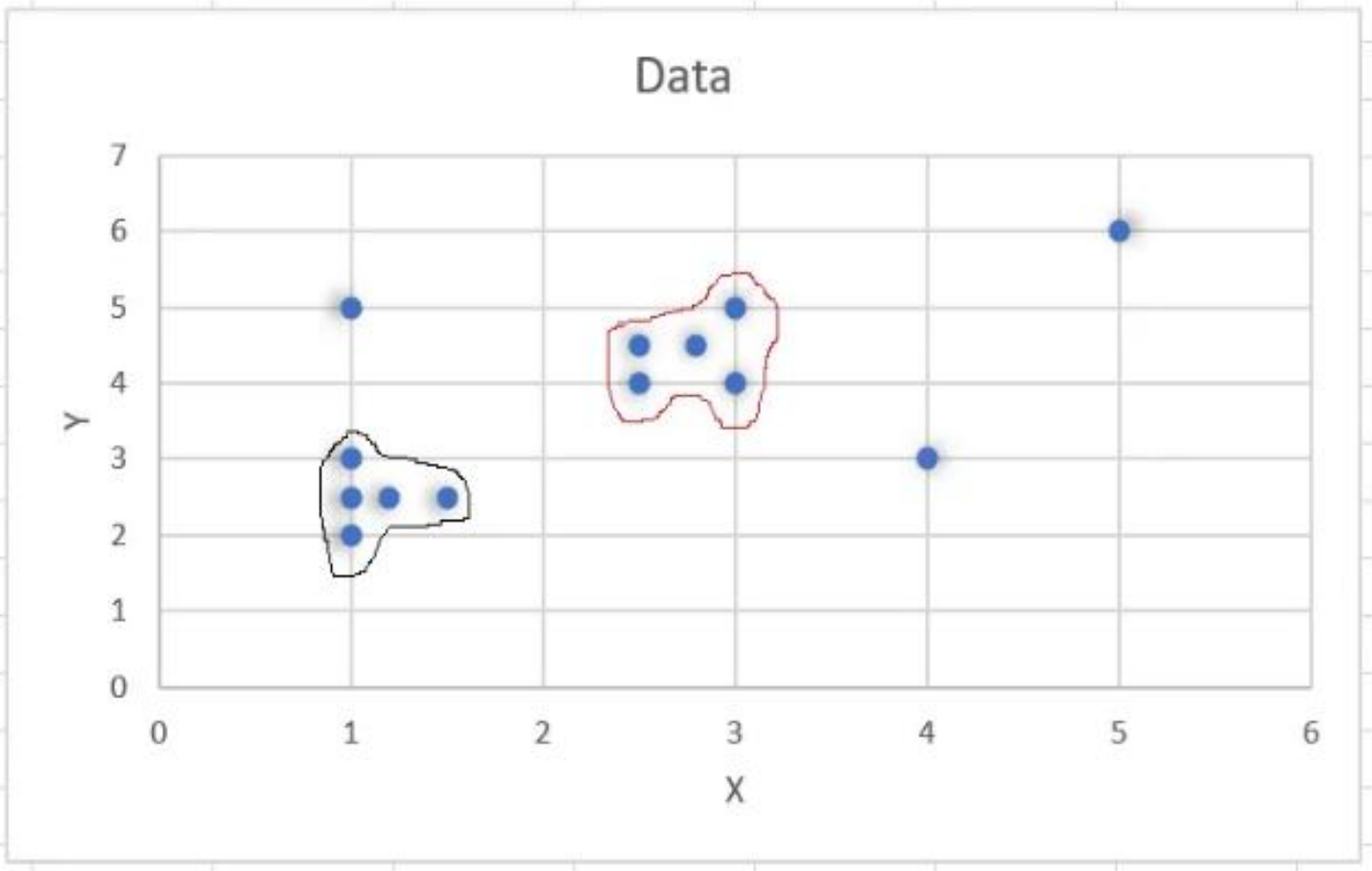
Example

X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3



Example

X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3



Eps=0.6, Minpoints=4

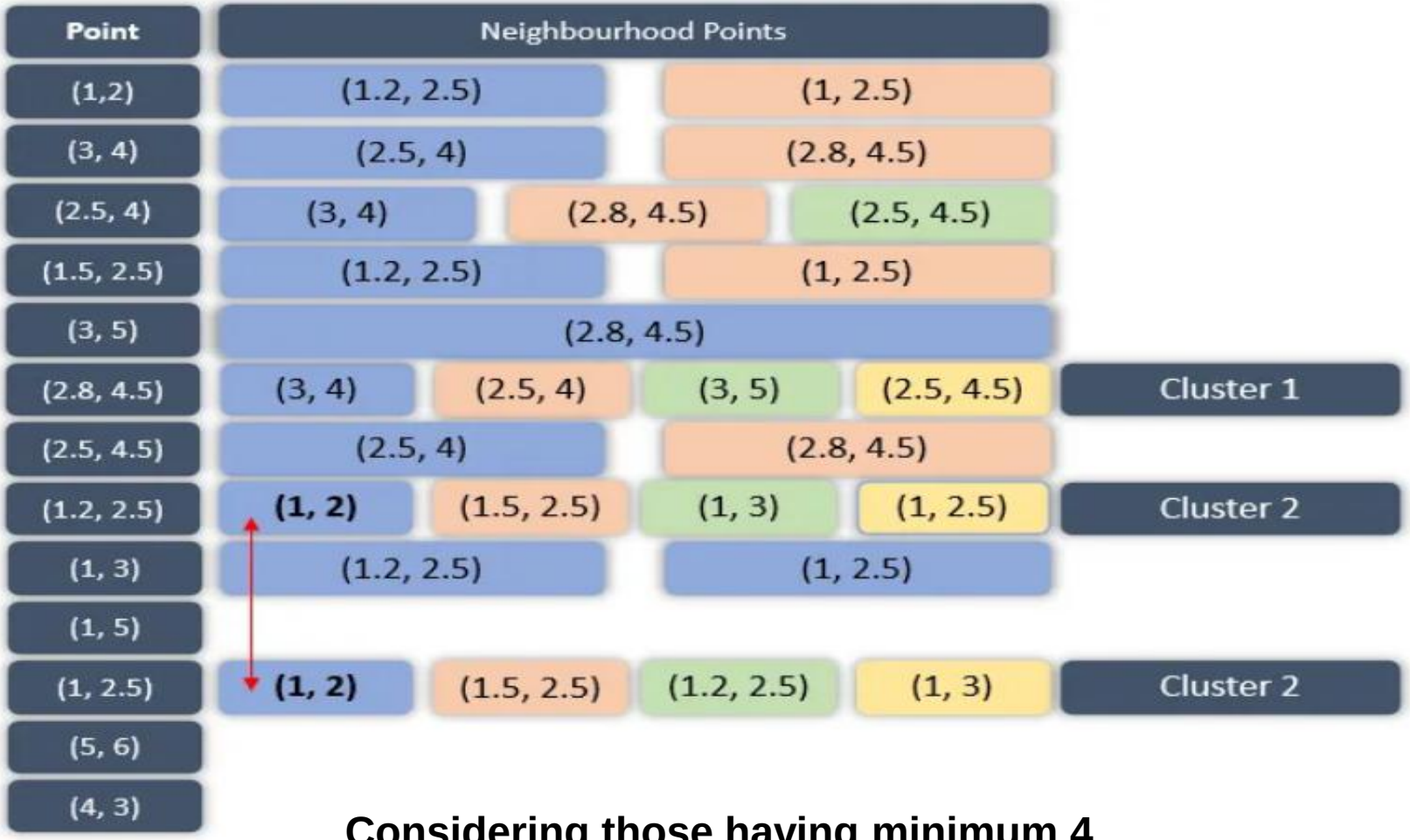
X	Y	Distance from (1,2)
1	2	
3	4	
2.5	4	
1.5	2.5	
3	5	
2.8	4.5	
2.5	4.5	
1.2	2.5	
1	3	
1	5	
1	2.5	
5	6	
4	3	

Calculate Euclidean distance of every point from (1,2)

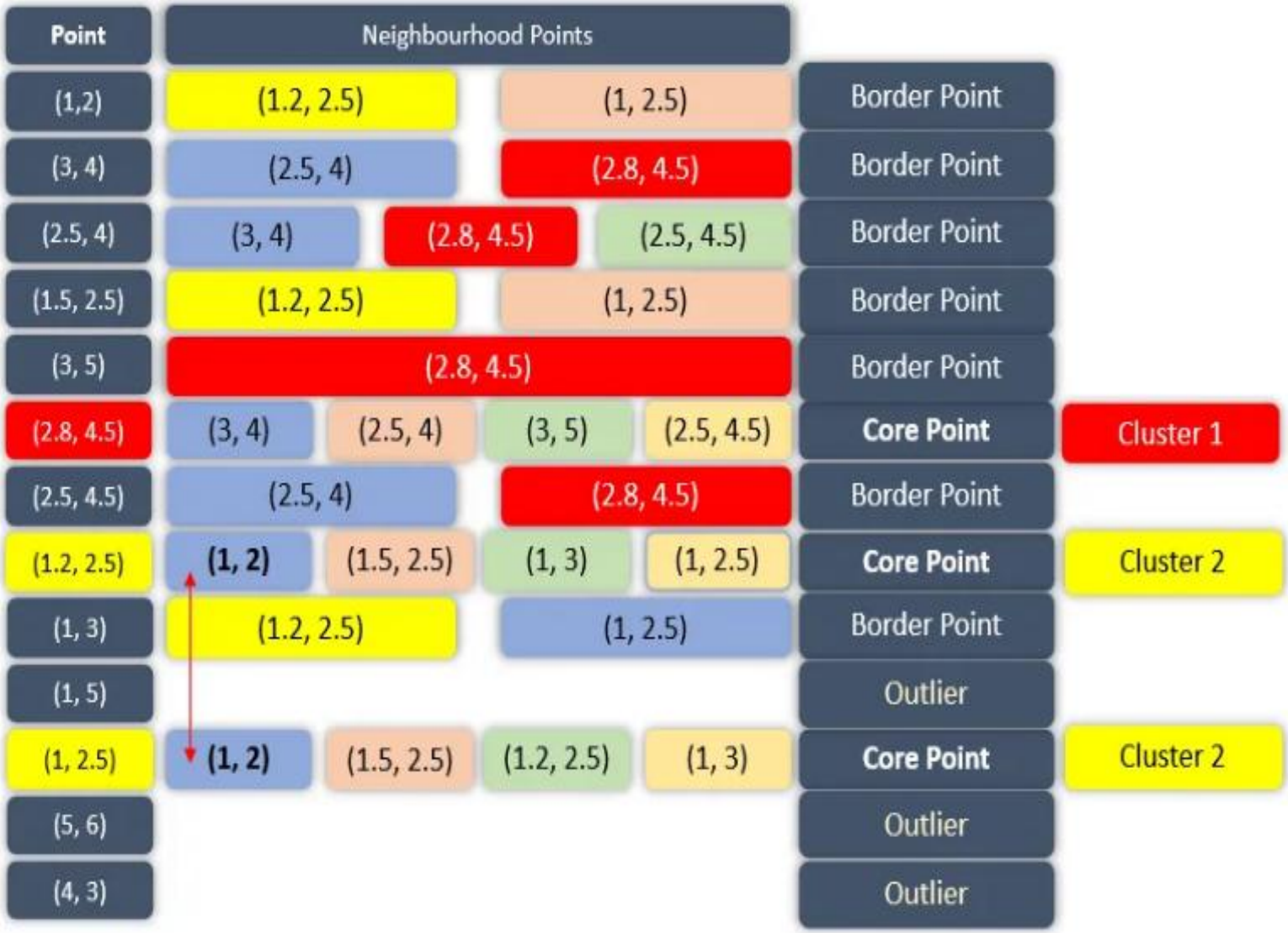
X	Y	Distance from (1,2)
1	2	0
3	4	2.8
2.5	4	2.5
1.5	2.5	0.7
3	5	3.6
2.8	4.5	3.08
2.5	4.5	2.9
1.2	2.5	0.53
1	3	1
1	5	3
1	2.5	0.5
5	6	5.6
4	3	3.1

Only 2 points , less then the minpoints

Same process repeated by taking other points as the reference



Considering those having minimum 4 points in neighbourhood


Point
Neighbourhood Points
(1,2)
(1.2, 2.5)
(1, 2.5)
Border Point
(3, 4)
(2.5, 4)
(2.8, 4.5)
Border Point
(2.5, 4)
(3, 4)
(2.8, 4.5)
(2.5, 4.5)
Border Point
(1.5, 2.5)
(1.2, 2.5)
(1, 2.5)
Border Point
(3, 5)
(2.8, 4.5)
Border Point
(2.8, 4.5)
(3, 4)
(2.5, 4)
(3, 5)
(2.5, 4.5)
Core Point
Cluster 1
(2.5, 4.5)
(2.5, 4)
(2.8, 4.5)
Border Point
(1.2, 2.5)
(1, 2)
(1.5, 2.5)
(1, 3)
(1, 2.5)
Core Point
Cluster 2
(1, 3)
(1.2, 2.5)
(1, 2.5)
Border Point
(1, 5)
Outlier
(1, 2.5)
(1, 2)
(1.5, 2.5)
(1.2, 2.5)
(1, 3)
Core Point
Cluster 2
(5, 6)
Outlier
(4, 3)
Outlier

Cluster 1	Cluster 2	Outliers
(3,4)	(1, 2)	(1, 5)
(2.5, 4)	(1.5, 2.5)	(5, 6)
(3,5)	(1.2, 2.5)	(4, 3)
(2.8, 4.5)	(1, 3)	
(2.5, 4.5)	(1, 2.5)	

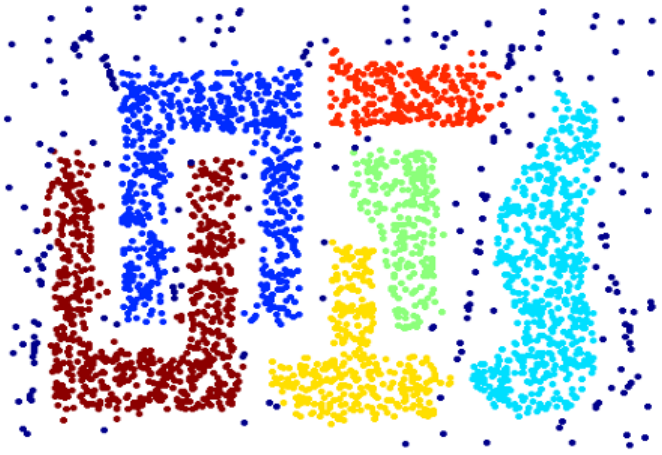
DBSCAN: Complexity

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).

When DBSCAN Works Well

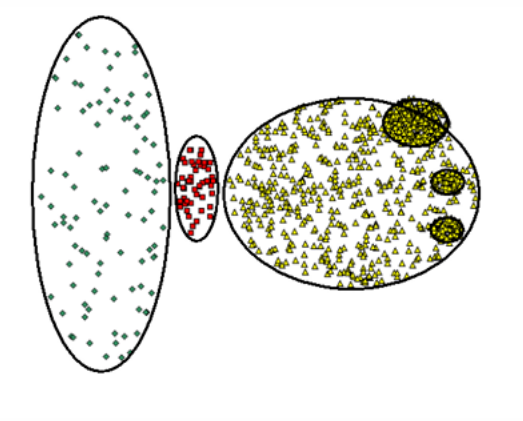
Original Points



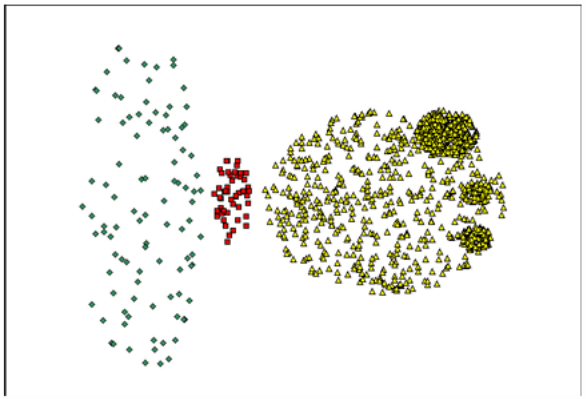
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

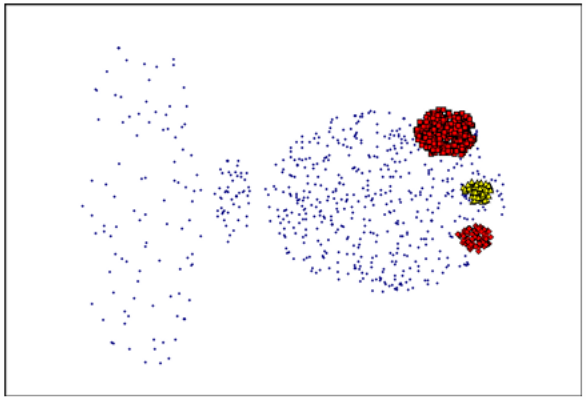
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

- Cannot handle varying densities
- sensitive to parameters—hard to determine the correct set of parameters

Feature	Hierarchical Clustering	Density-Based Clustering
Cluster Structure	Hierarchical, represented by a dendrogram	Flat, based on density regions
Cluster Shape	Typically assumes spherical or compact clusters	Can identify clusters of arbitrary shapes
Noise Handling	May struggle with noisy data, potentially creating spurious clusters	Robust to noise, can identify and label outliers
Number of Clusters	Requires a threshold or stopping criterion to determine the number of clusters	Automatically determines clusters based on density
Algorithm Examples	Agglomerative, Divisive	DBSCAN, HDBSCAN