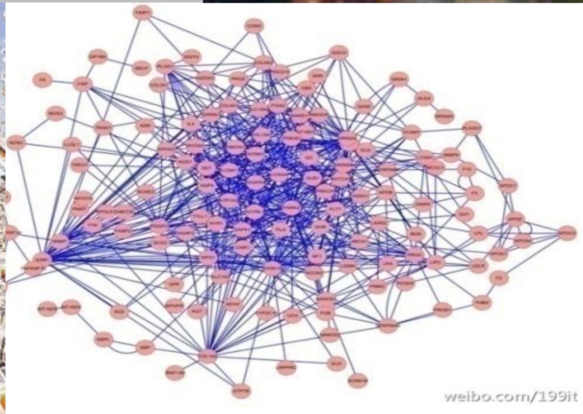




Chapter 11 Korhonen and DBSCAN

2025 Autumn

Lei Sun

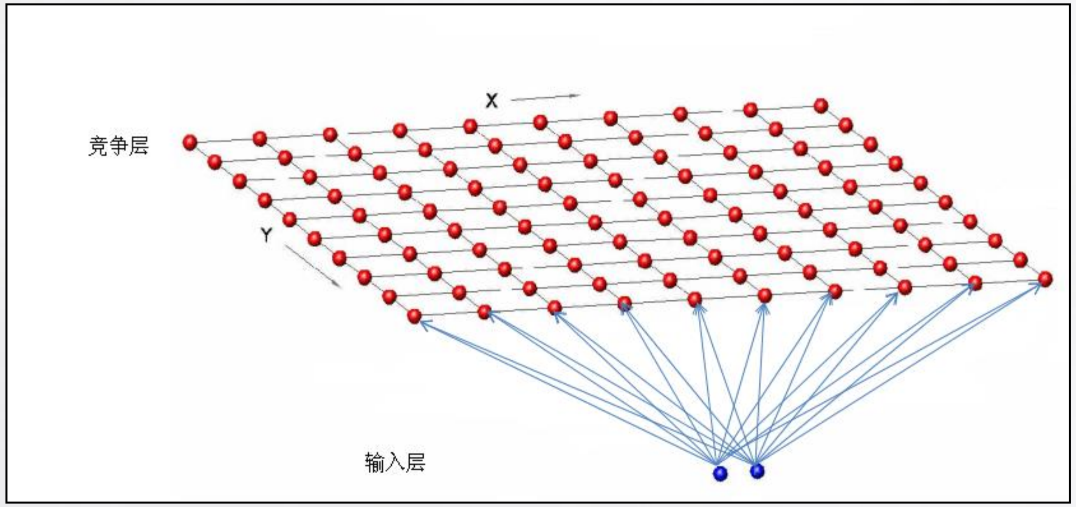
01 SOM topology

02 Steps of Kohonen

03 DBSCAN concepts

04 DBSCAN and K-means

Self-Organizing feature Map



SOM structure

Two layers: input and output

Input layer: the number of nodes is the number of inputted attributes

Output layer: the number of nodes is the number of cluster K

At the output layer, there are connections

w_{jk} means the weight from the node k of input layer to a neuron j of output layer

Clustering steps

Step1. Preprocess data

Based on Euclidean distance, all the attributes have to be normalized to [0,1]

Step2. Identify the initial centers of clusters

Similar to K-Means. Give the cluster number K

That is to have K cluster centers (K output neurons)

$W = [w_1, w_2 \cdots w_k]$ Every output layer neuron is the vector of weights $W_j = [w_{1j}, w_{2j} \cdots w_{pj}]$

Step3. At t moment randomly read observations, respectively compute Euclidean distance d(t) between the data X(t) and K centers. And find the closest center. Now $W_i(t)$ is winner(neuron), which is the best node matching the t th observation.

Clustering steps

Step 4: Adjust winner $W_i(t)$ **and its neighborhoods weights.**
That is to adjust cluster center values.

（1） how to adjust?
（2） what elements are the neighborhoods?

Step 5: If the condition of stop is not satisfied, go to step 3

Loop stop： loops number

the weights does not change any more.

Clustering steps

At t moment (the number of iterations), p weights between p input nodes and the W_i (t) are :

$$W_i(t) = [w_{1i}, (t), w_{2i}(t) \cdots w_{pi}(t)]$$

The winner's weight is adjusted as follows:

The learning rate (weight coefficient) at t moment

$$W_i(t+1) = W_i(t) + \eta(t)[X(t) - W_i(t)]$$

$$\eta(t+1) = \eta(t) - \frac{\eta(0) - \eta_{low}}{c}$$

学习周期数

$$\eta(t) = \eta(0) * (1 - \frac{t}{c})$$

Clustering steps

Due to the links betwee output layer's neurons, the neighborhoods' weights have to be renewed. Typically, need to confirm the radius from the winner to the neighborhoods.

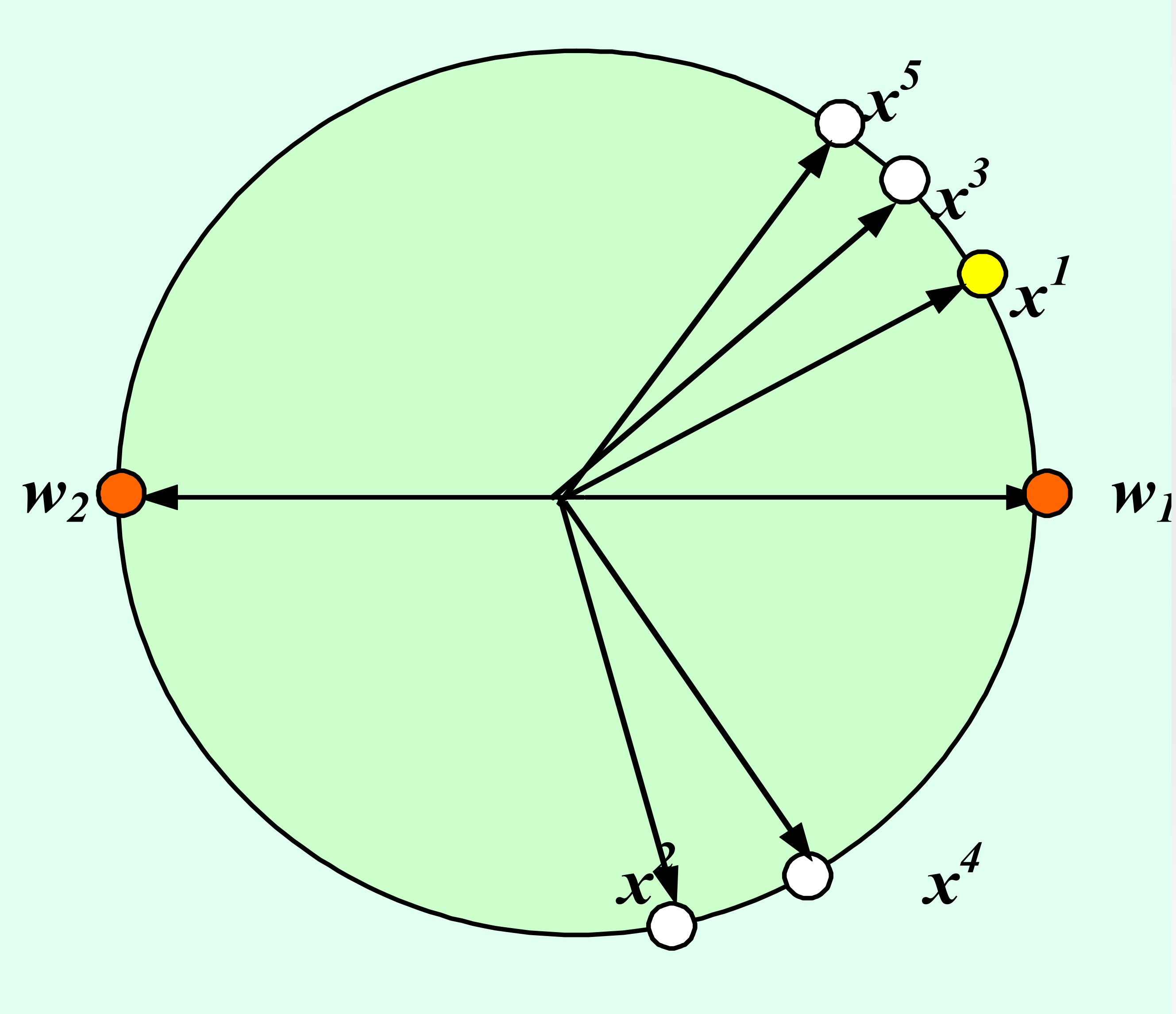
center: $W_i(t)$

The neurons whose distance ($h_{ji}(t)$) to $W_i(t)$ is in the range of predefined radium are the winner's neighborhoods.

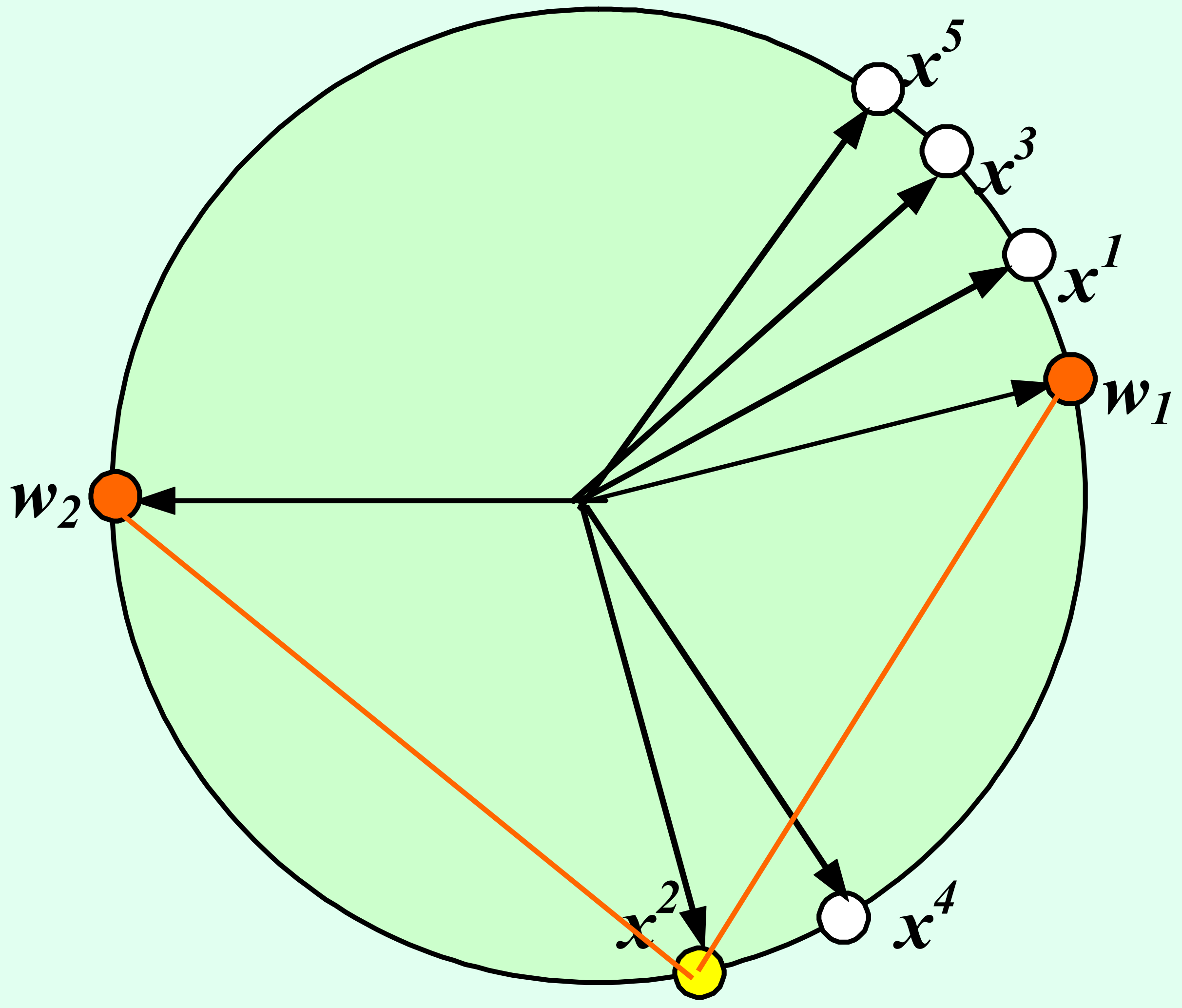
$$w_j(t+1) = w_j(t) + \eta(t)\left[x(t) - w_j(t)\right]$$

Initialization of h is 2/3 of the distance between winner point and the furthest point. And be gradually reduced to zero. The number of output layer nodes within neighborhood gradually

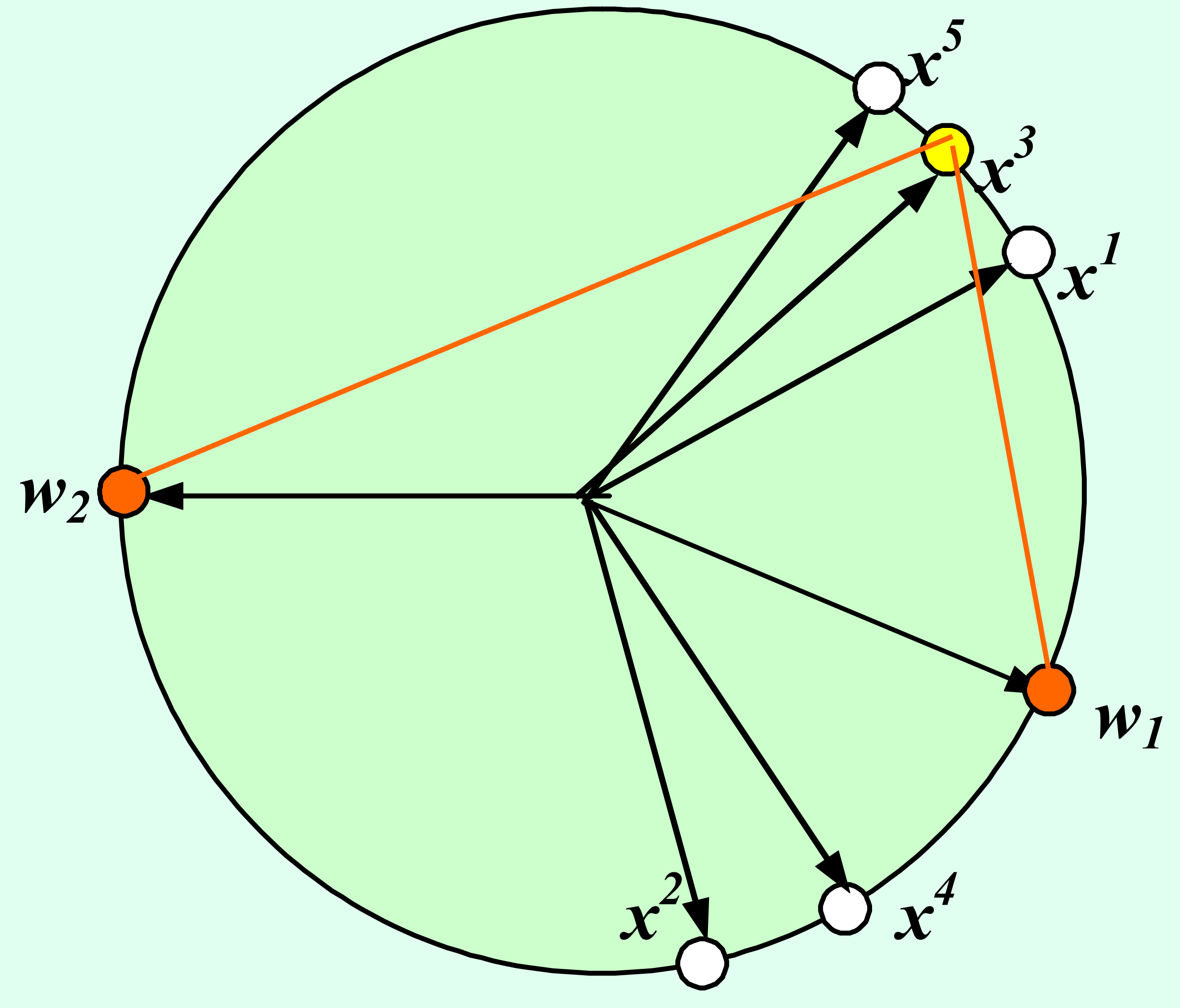
```
som = SOM(m=5, n=5, dim=2, radius=1.0, learning_rate=0.5, topology='rectangular', activation_distance='euclidean')
```



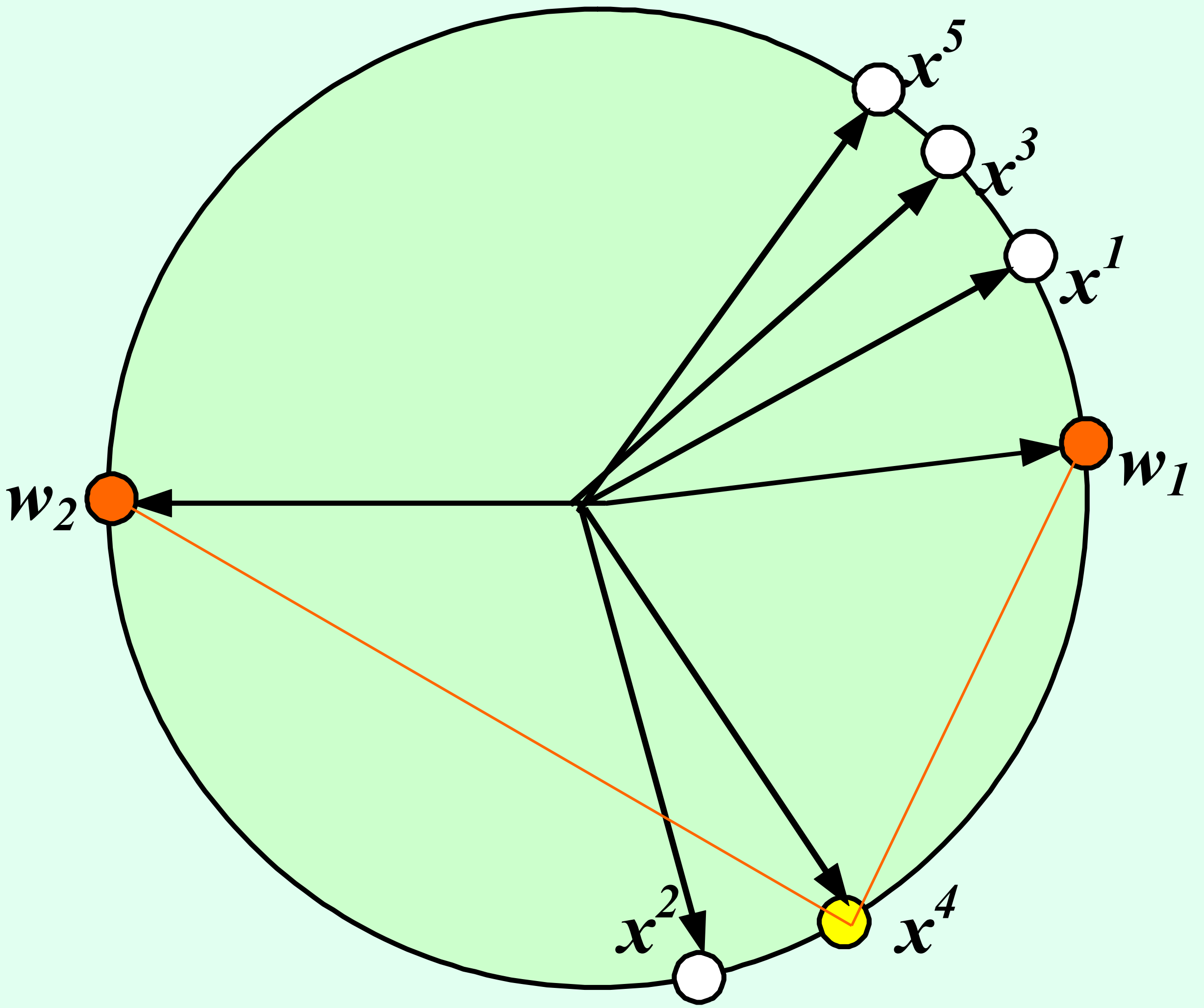
训练次数	W_1	W_2
1	18.43°	−180°
2	−30.8°	−180°
3	7°	−180°
4	−32°	−180°
5	11°	−180°
6	24°	−180°
7	24°	−130°
8	34°	−130°
9	34°	−100°
10	44°	−100°
11	40.5°	−100°
12	40.5°	−90°
13	43°	−90°
14	43°	−81°
15	47.5°	−81°
16	42°	−81°
17	42°	−80.5°
18	43.5°	−80.5°
19	43.5°	−75°
20	48.5°	−75°



训练次数	W_1	W_2
1	18. 43°	−180°
2	−30. 8°	−180°
3	7°	−180°
4	−32°	−180°
5	11°	−180°
6	24°	−180°
7	24°	−130°
8	34°	−130°
9	34°	−100°
10	44°	−100°
11	40. 5°	−100°
12	40. 5°	−90°
13	43°	−90°
14	43°	−81°
15	47. 5°	−81°
16	42°	−81°
17	42°	−80. 5°
18	43. 5°	−80. 5°
19	43. 5°	−75°
20	48. 5°	−75°



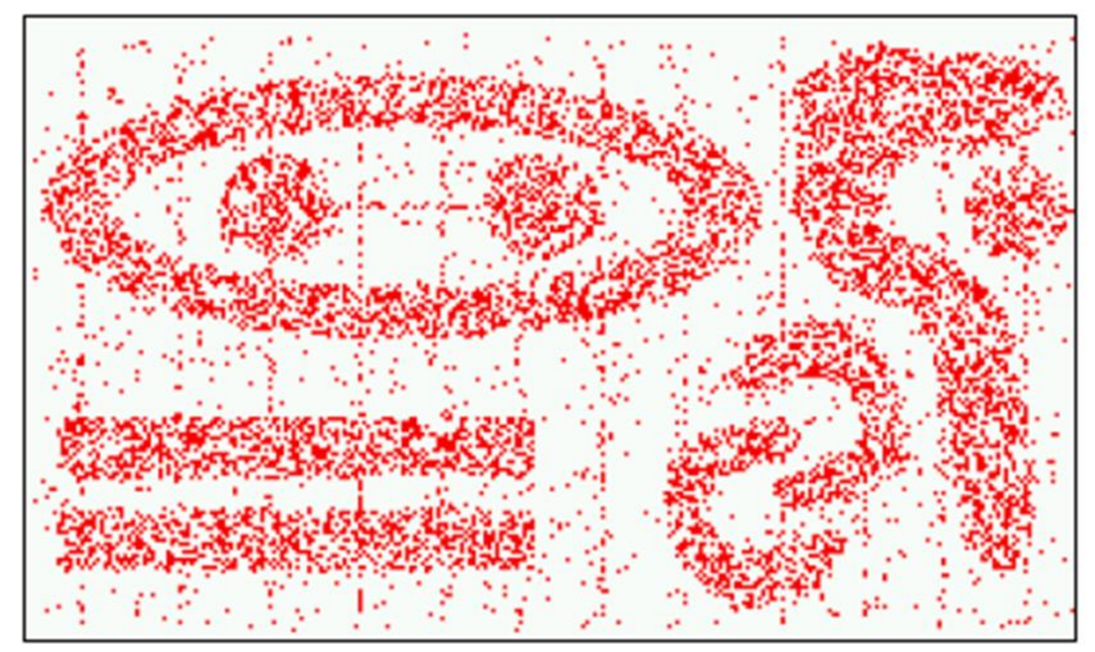
训练次数	W_1	W_2
1	18.43°	−180°
2	−30.8°	−180°
3	7°	−180°
4	−32°	−180°
5	11°	−180°
6	24°	−180°
7	24°	−130°
8	34°	−130°
9	34°	−100°
10	44°	−100°
11	40.5°	−100°
12	40.5°	−90°
13	43°	−90°
14	43°	−81°
15	47.5°	−81°
16	42°	−81°
17	42°	−80.5°
18	43.5°	−80.5°
19	43.5°	−75°
20	48.5°	−75°

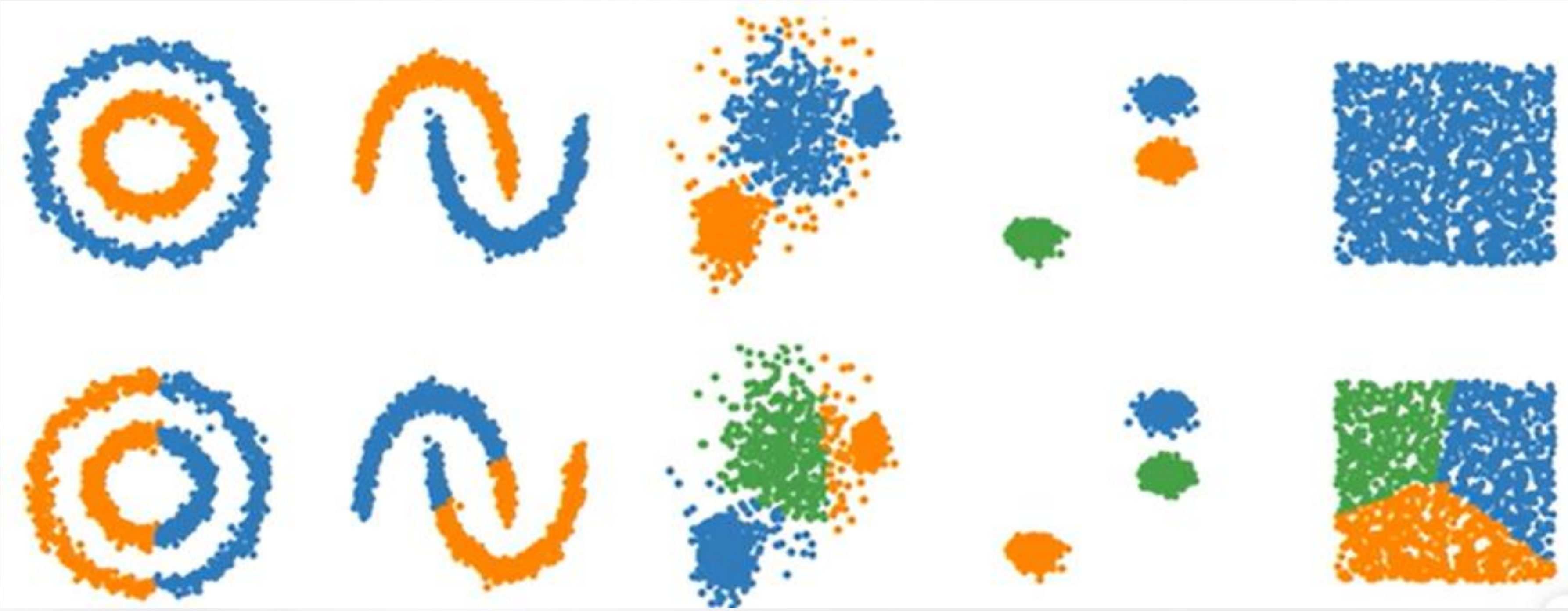


训练次数	W_1	W_2
1	18.43°	−180°
2	−30.8°	−180°
3	7°	−180°
4	−32°	−180°
5	11°	−180°
6	24°	−180°
7	24°	−130°
8	34°	−130°
9	34°	−100°
10	44°	−100°
11	40.5°	−100°
12	40.5°	−90°
13	43°	−90°
14	43°	−81°
15	47.5°	−81°
16	42°	−81°
17	42°	−80.5°
18	43.5°	−80.5°
19	43.5°	−75°
20	48.5°	−75°

Density-Based Spatial Clustering of Application with Noise (DBSCAN)

- ✓ Generate cluster of arbitrary shapes
- ✓ Robust against noise
- ✓ No K value required in advance
- ✓ Somewhat similar to human vision





Visualizing DBSCAN Clustering:

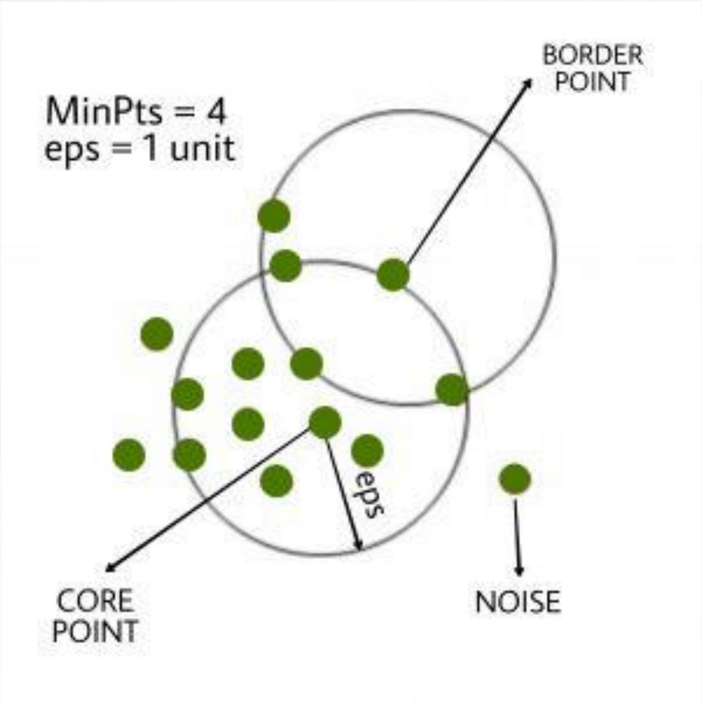
https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

Concepts

Two Parameters Required For DBSCAN Algorithm

epsilon (eps)– defines the neighborhood around a data point i.e. If the distance between two points is lower or equal to 'eps' then they are considered neighbors.

min_samples – Minimum number of neighbors (data points) within eps radius. As a general rule, the minimum (MinPts) can be derived from the number of dimensions D in the dataset as MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

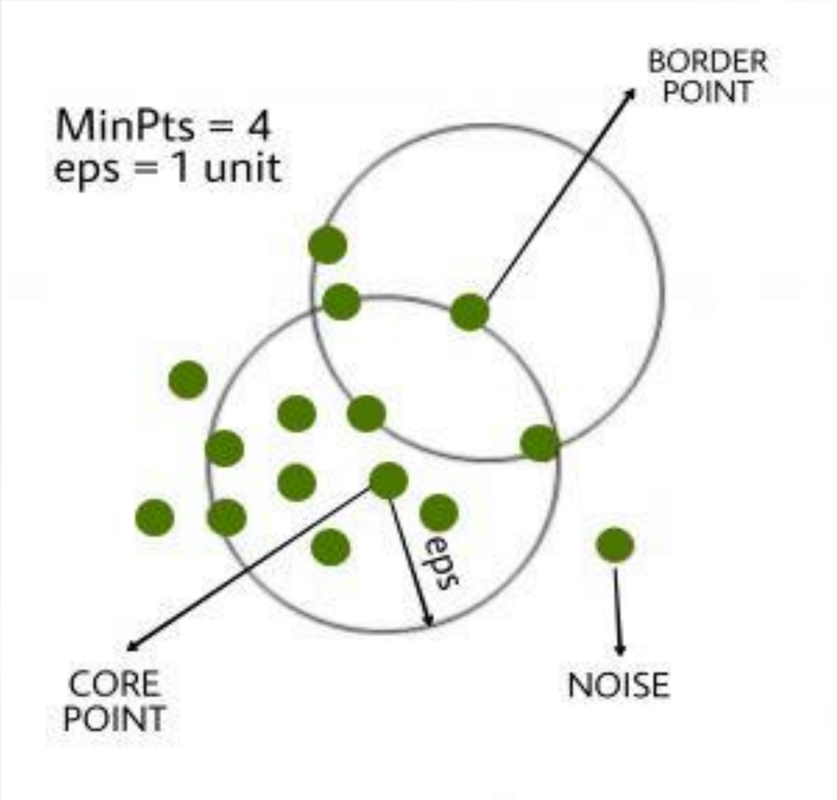


Concepts

Core Point: A point is a core point if it has more than MinPts points within eps.

Border Point: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.

Noise or outlier: A point which is not a core point or border point.

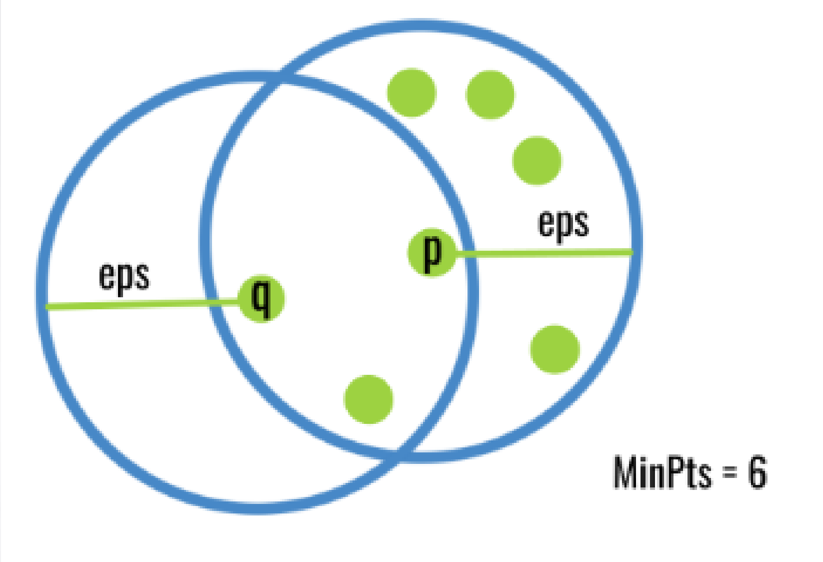


DBSCAN reachability and connectivity

Reachability – 核心点的直接密度可达点

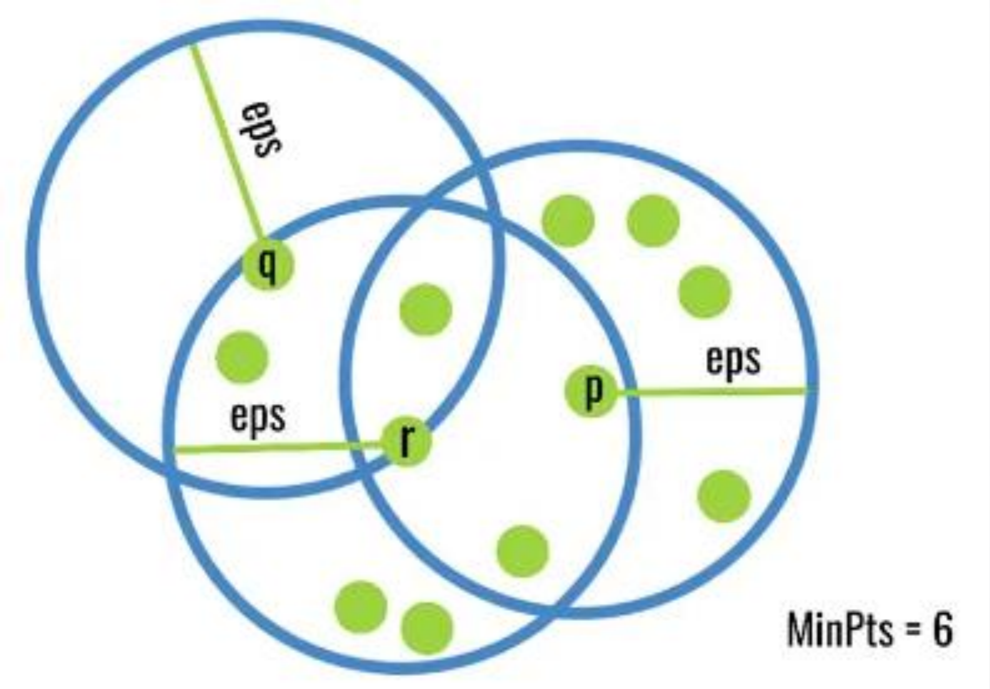
Directly density reachable: An object (or instance) q is directly density reachable from object p if q is within the ε-Neighborhood of p and p is a core object.

directly density reachability is not symmetric. Object p is not directly density-reachable from object q as q is not a core object.



Concepts

Reachability – 核心点的密度可达点

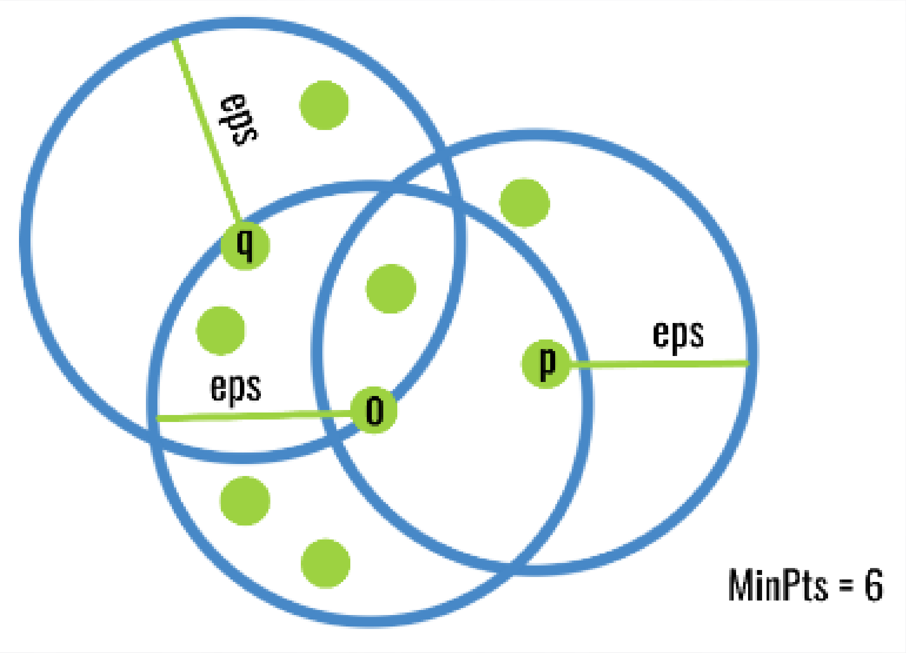


Density reachable: An object q is density-reachable from p w.r.t (with respect to) ε and MinPts , if there is a chain of objects q_1, q_2..., q_n, with q_1=p, q_n=q such that q_{i+1} is directly density-reachable from q_i w.r.t ε and MinPts for all 1 <= i <= n

density reachability is not symmetric. p is not density-reachable from object q.

Concepts

Connectivity – 密度相连



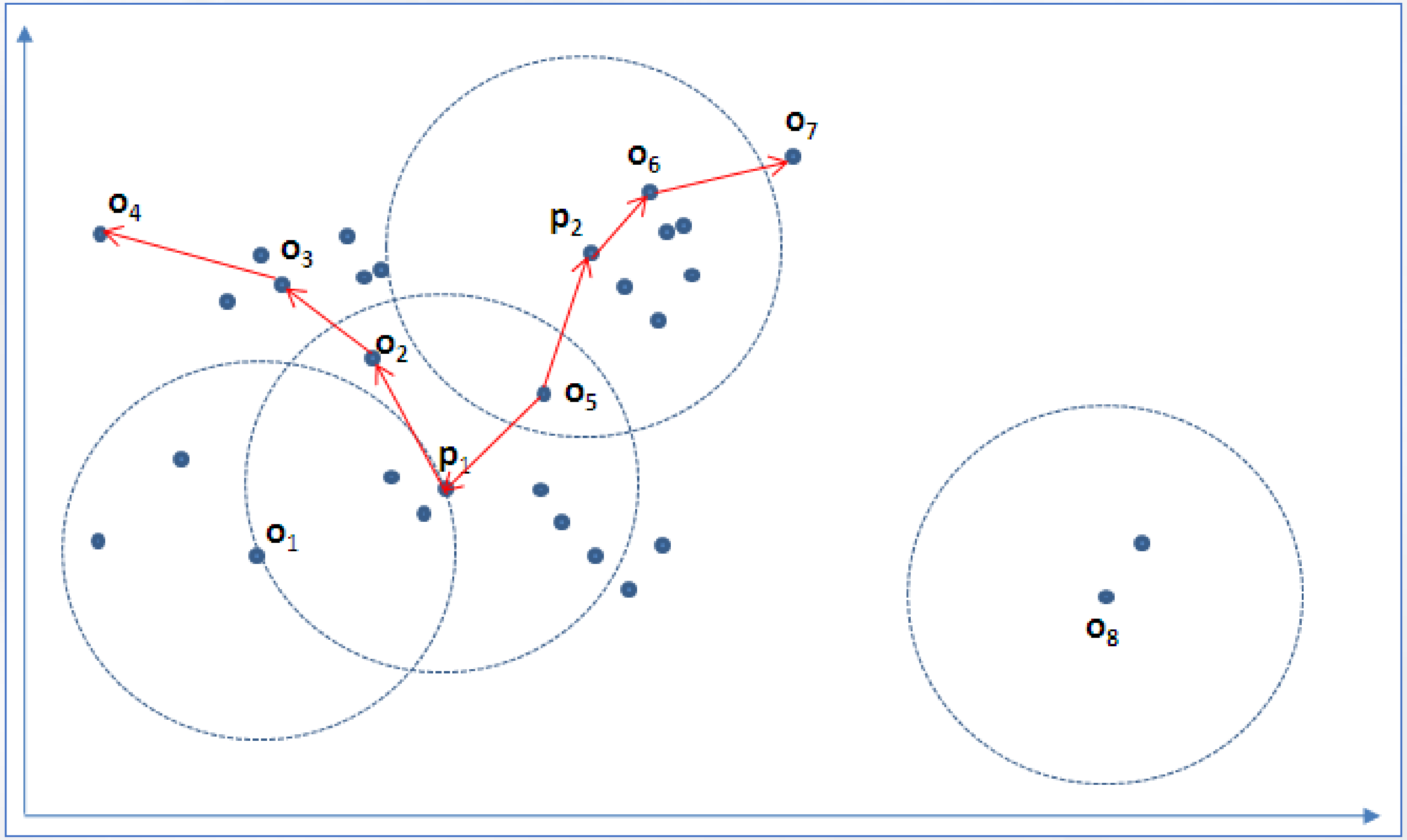
Density connectivity: Object q is density-connected to object p w.r.t ε and *MinPts*, if there is an object o such that both p and q are density-reachable from o w.r.t ε and *MinPts*.

density connectivity is symmetric. If object q is density-connected to object p then object p is also density-connected to object q.

Concepts

ε=1, and *Min* Pts=6

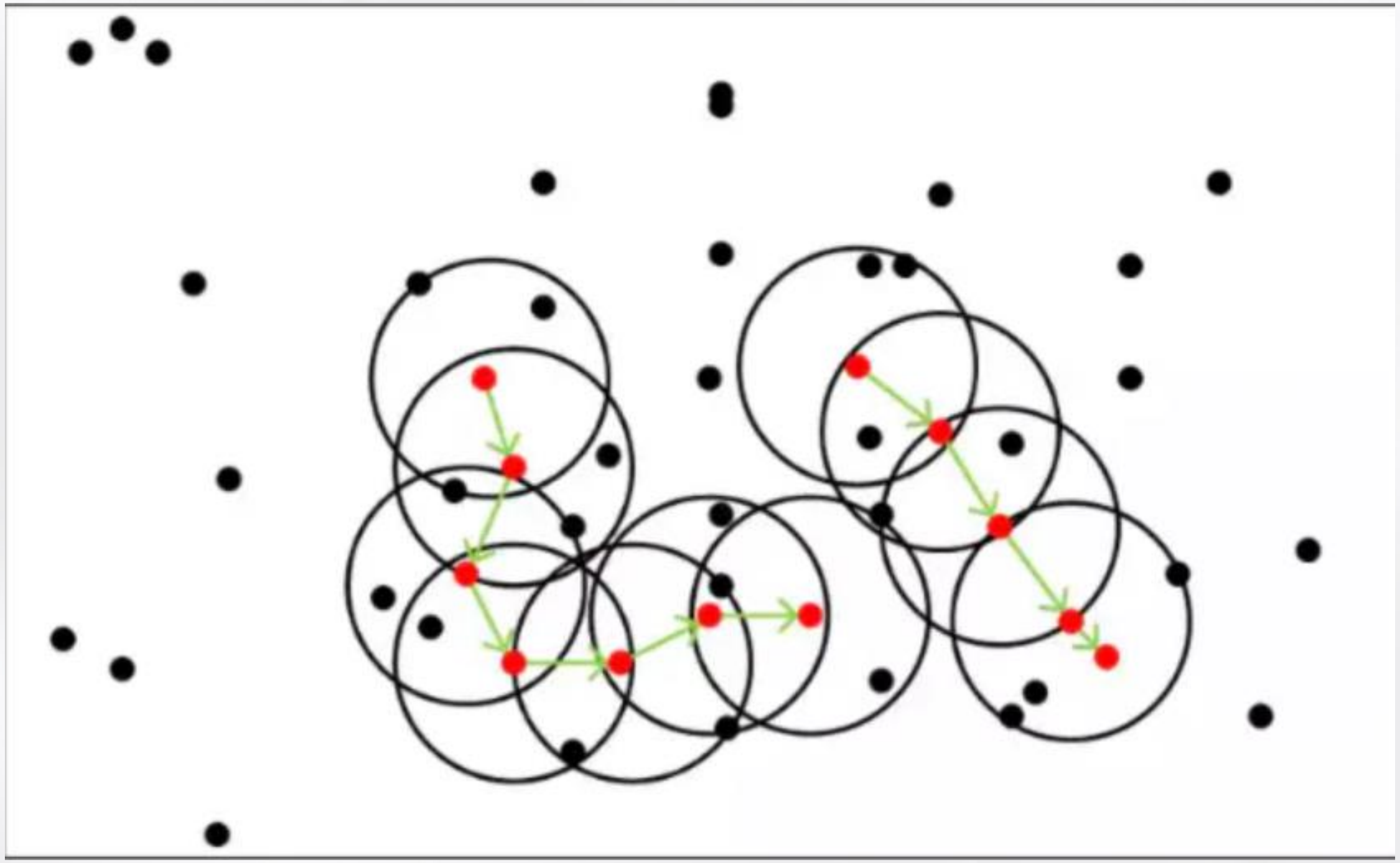
Core: P1,O2,O3,
O5,P2,O6



（O4, O7）?   (O4, P1) ?

P_1 density-reachable to o4

Concepts



All the red dots are core points, and all the black samples aren’t core points.

The core points connected by the green arrow lines are density reachable.

All the **black dots** inside the circle are density connection.

01

Concepts

Outliers:

- All points not reachable(connected) from any other point are outliers.
- Or this is a point that is neither a Core nor a Border. And it has less than m points within distance n from itself.

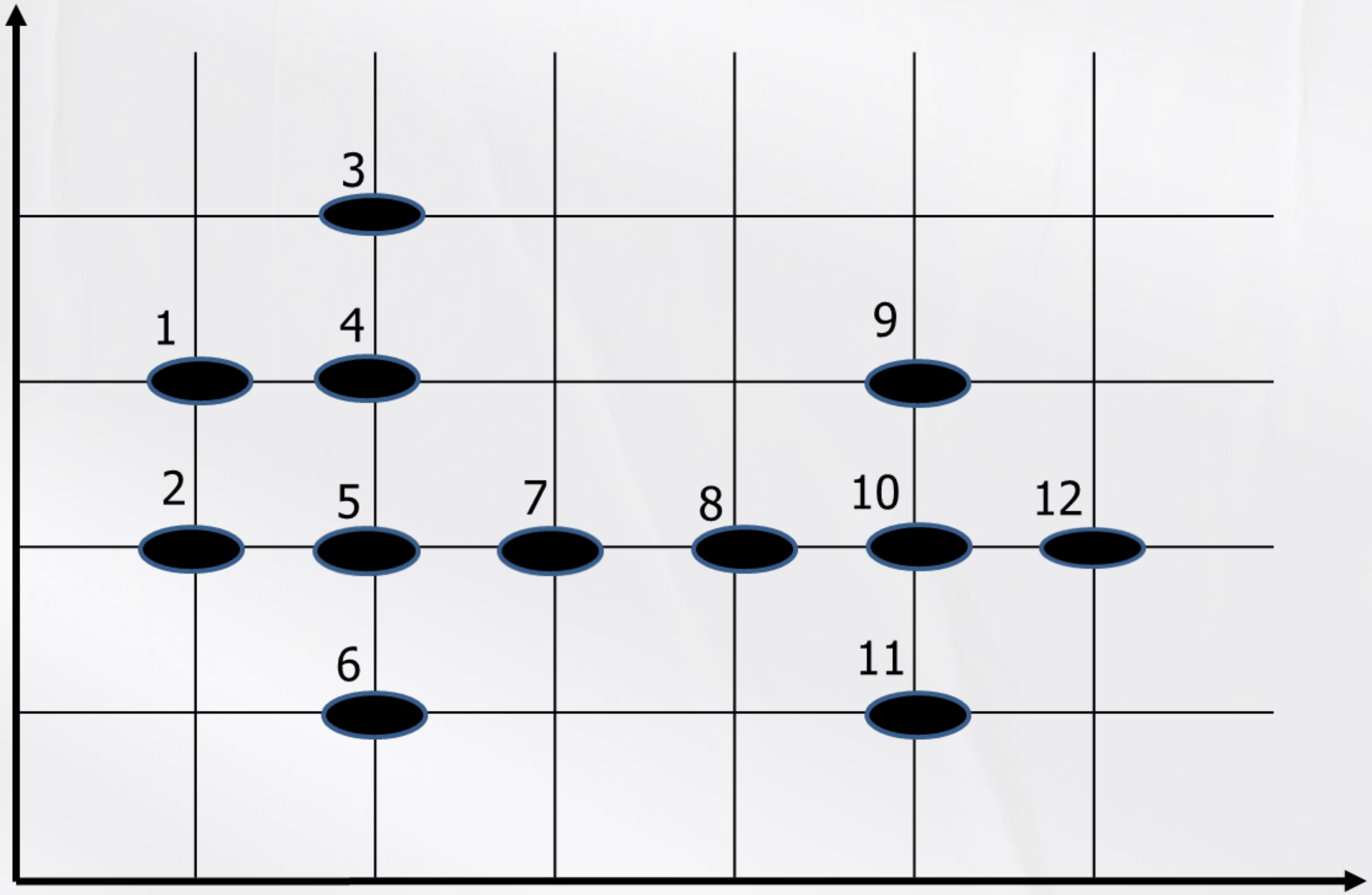
A cluster then satisfies two properties:

- All points within the cluster are at least mutually density-connected.
- If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

DBSCAN Algorithm

Requirements:

ε=1， MinPts=4



DBSCAN Algorithm

Parameter selection

- **k-距离**：给定数据集P={p(i); i=0,1,···n}，对于任意点P(i)，计算点P(i)到其子集S={p(1), p(2), ···, p(i-1), p(i+1), ···, p(n)}中所有点之间的距离，距离按照从小到大的顺序排序，假设排序后的距离集合为D={d(1), d(2), ···, d(k-1), d(k), d(k+1), ···,d(n)}，则d(k)就被称为k-距离。

k-距离是点p(i)到所有点(除了p(i)点)之间距离第k近的距离。对待聚类集合中每个点p(i)都计算k-距离，最后得到所有点的k-距离集合E={e(1), e(2), ···, e(n)}。k-距离变化趋势确定K值从而确定邻域半径（距离值）。

```
Input:
   A dataset D, Epsilon (ε), MinPts

Output:
   A set of clusters C

1. Set C to an empty list
2. For each point p in D NOT marked as visited:
    2.1 Mark p as visited
    2.2 Find all points, points(p), within Epsilon distance from p.
    2.3 If |points(p)|>=MinPts
        2.3.1 Create a new cluster s, and add p to s
        2.3.2 for each point q in points(p)
            2.3.2.1 if q is not marked as visited
                2.3.2.1.1 mark q as visited
                2.3.2.1.2 if |points(q)|>=MinPts then s=union(s, points(q))
            2.3.2.2 if q does not yet have a cluster label, add q to s
        2.3.3 add s to C
    2.4 else mark p as outlier
3. Return C
```

```
DBSCAN(D, eps, MinPts)
  C = 0
  for each unvisited point P in dataset D
    mark P as visited
    NeighborPts = regionQuery(P, eps)
    if sizeof(NeighborPts) < MinPts
      mark P as NOISE
    else
      C = next cluster
      expandCluster(P, NeighborPts, C, eps, MinPts)
```

```
expandCluster(P, NeighborPts, C, eps, MinPts)
  add P to cluster C
  for each point P' in NeighborPts
    if P' is not visited
      mark P' as visited
      NeighborPts' = regionQuery(P', eps)
      if sizeof(NeighborPts') >= MinPts
        NeighborPts = NeighborPts joined with NeighborPts'
    if P' is not yet member of any cluster
      add P' to cluster C
```

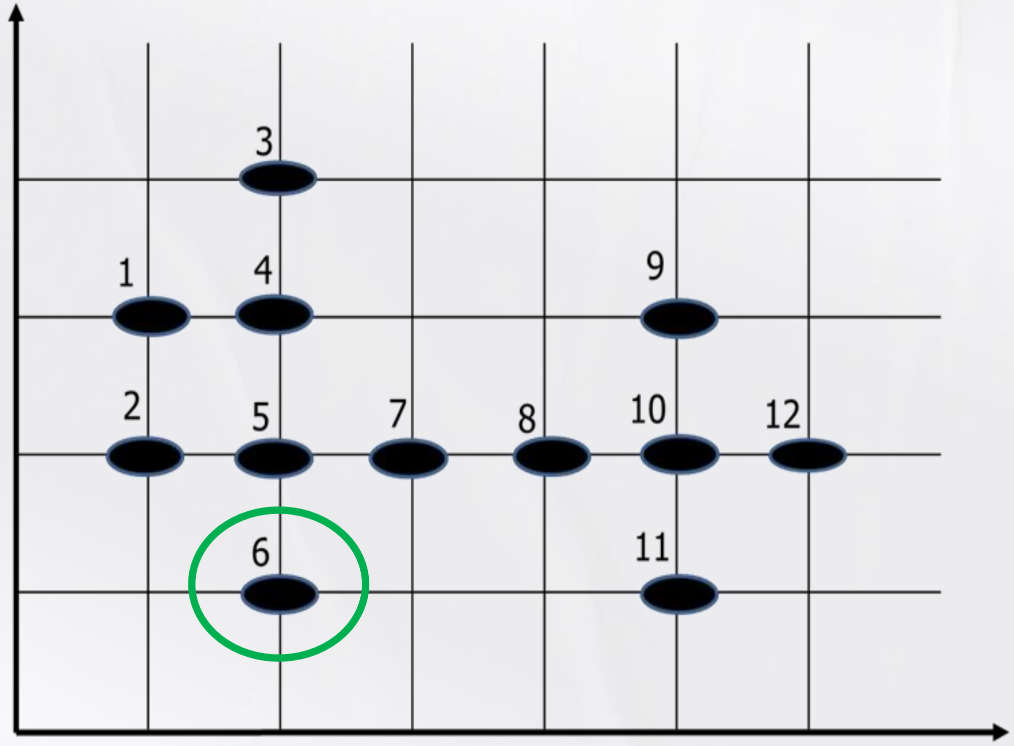
```
regionQuery(P, eps)
  return all points within P's eps-neighborhood (including P)
```

DBSCAN Algorithm

① Mark all points as *unvisited*。

② Randomly select point 6 and mark it as *visited*.

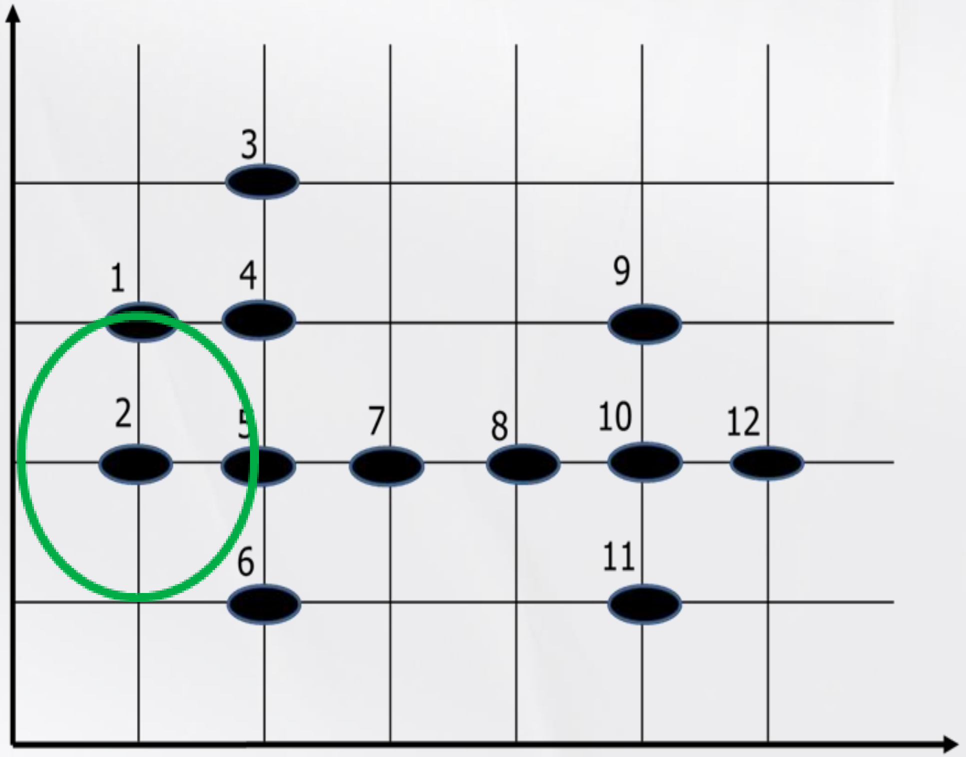
The neighborhood with it as the center and radius 1 contains 2 points, which does not meet the requirement, so it is not a core point and is temporarily marked as noise.



DBSCAN Algorithm

③ Randomly select point 2 and mark it as *visited*.

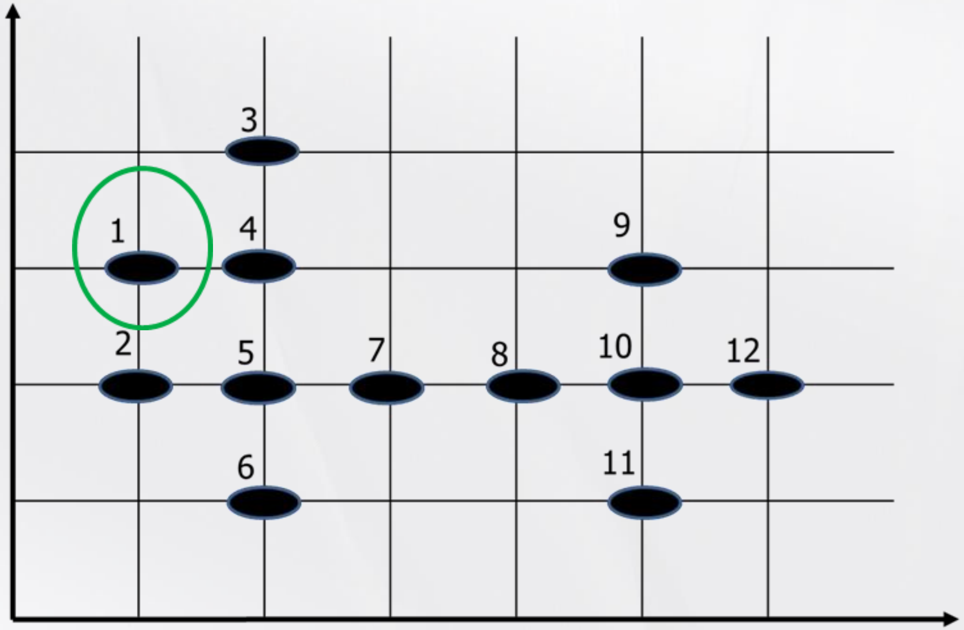
The neighborhood with it as the center and radius 1 contains 3 points, which does not meet the requirement. So it is not a core point and is temporarily marked as *noise.*



DBSCAN Algorithm

④ Randomly select point 1 and mark it as *visited*.

The neighborhood with it as the center and radius 1 contains 3 points, which does not meet the requirement. So it is not a core point and is temporarily marked as *noise.*

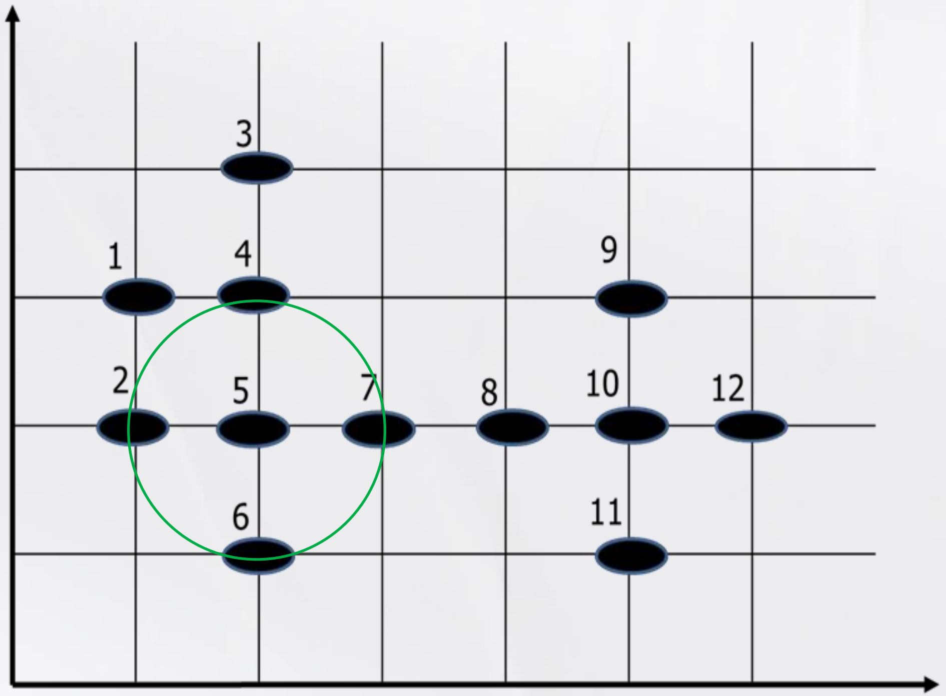


DBSCAN Algorithm

⑤ Randomly select point 5 and mark it as visited.

- Point 5 is the core point. Form a new cluster C1, put point 5 into C1, that is, C1={5}.

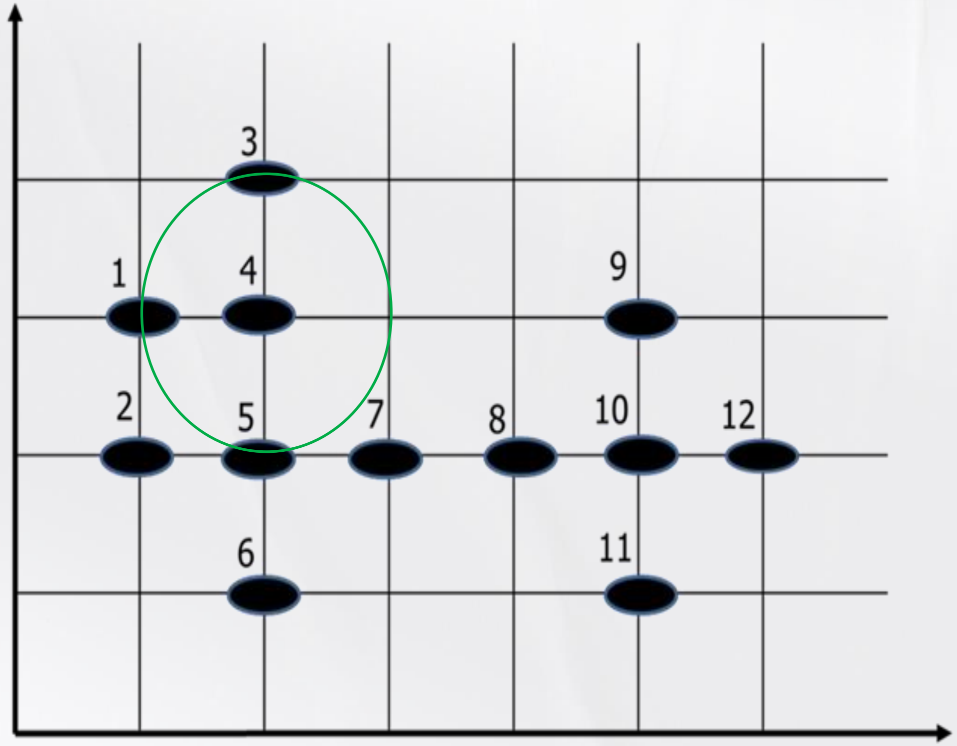
- Put the points in the neighborhood of point 5 with a radius of 1 into the candidate set N, that is, N={2,4,6,7}, where the point 2 and 6 are visited, and points 4 and 7 are unvisited.



DBSCAN Algorithm

- Select the unvisited point 4 in N and mark it as visited. Check if the point 4 is core point. YES!

  put point 4 into C1, C1={4,5}.
- Put the points in the neighborhood of point 4 with a radius of 1 into the candidate set N, that is, N={1,2,3,6,7}, where points 1, 2 and 6 are visited, and points 3 and 7 are unvisited.



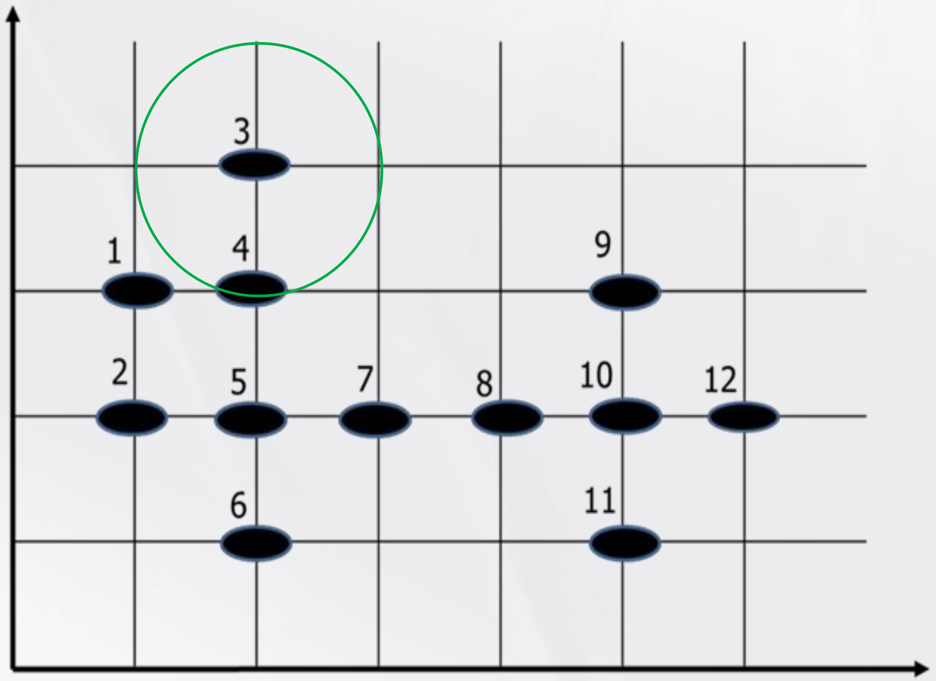
02

DBSCAN Algorithm

- Select the unvisited point 3 in N, mark it as visited.

  Is point 3 a core point? NO!

- And point 3 doesn't belong to other clusters and with in N. Put point 3 into C1, i.e. C1={3,4,5}, N={1,2,6,7}, where point 1, point 2 and point 6 are visited, point 7 is unvisited.

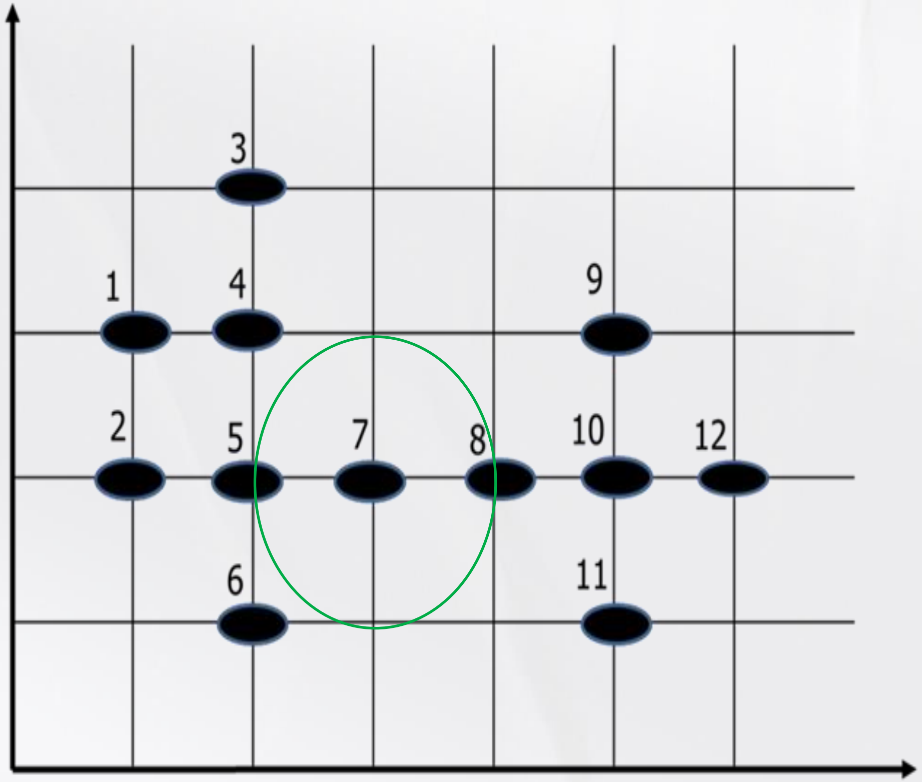


DBSCAN Algorithm

- Select the unvisited point 7 in N, mark it as visited.

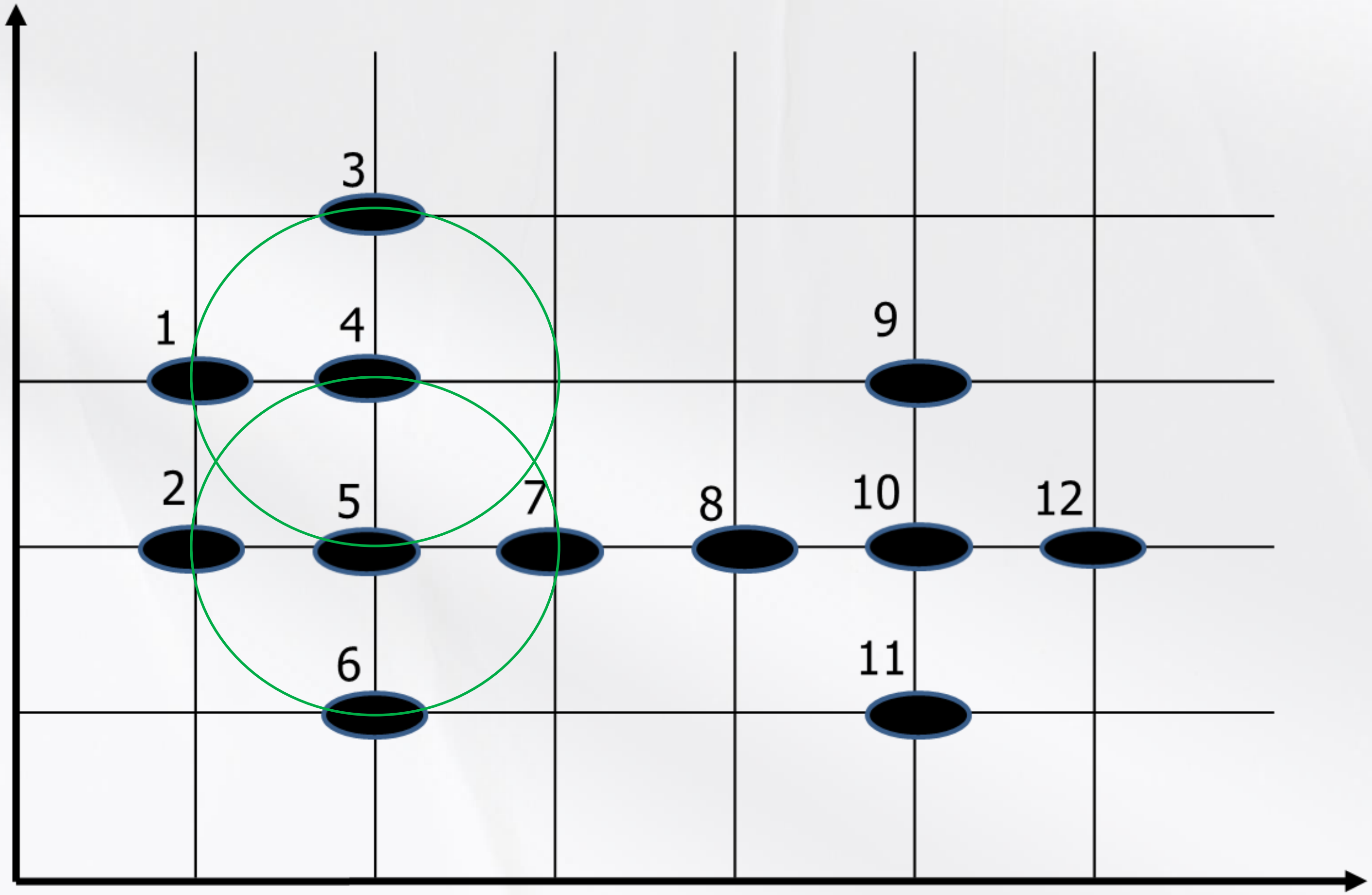
  Check if point 7 is core point. NO!

- point 7 doesn't belong to other clusters, and with in N. So put point 7 into C1, i.e. C1={3,4,5,7}, N={1,2,6}. Although point 1, 2 and 6 in N are visited, they do not belong to other clusters. Put them into C1, C1={1,2,3,4,5,6,7},   N={}.



02

DBSCAN Algorithm



$\varepsilon=1$， MinPts=4

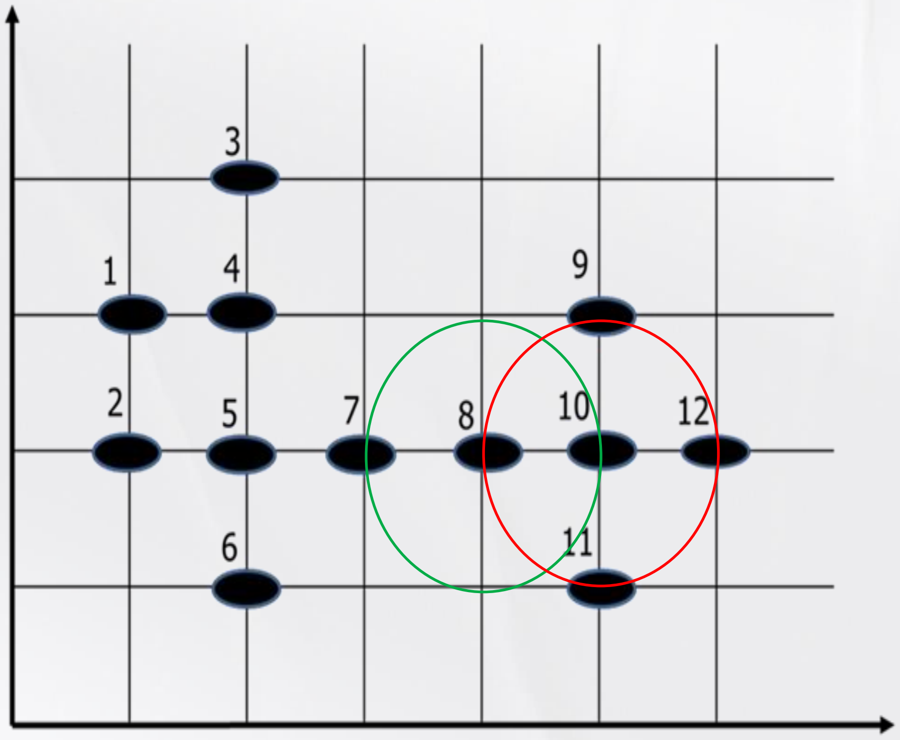
02

DBSCAN Algorithm

⑥ Randomly select point 8 among other unvisited points, mark it as visited. Not the core point, temporarily marked as noise.

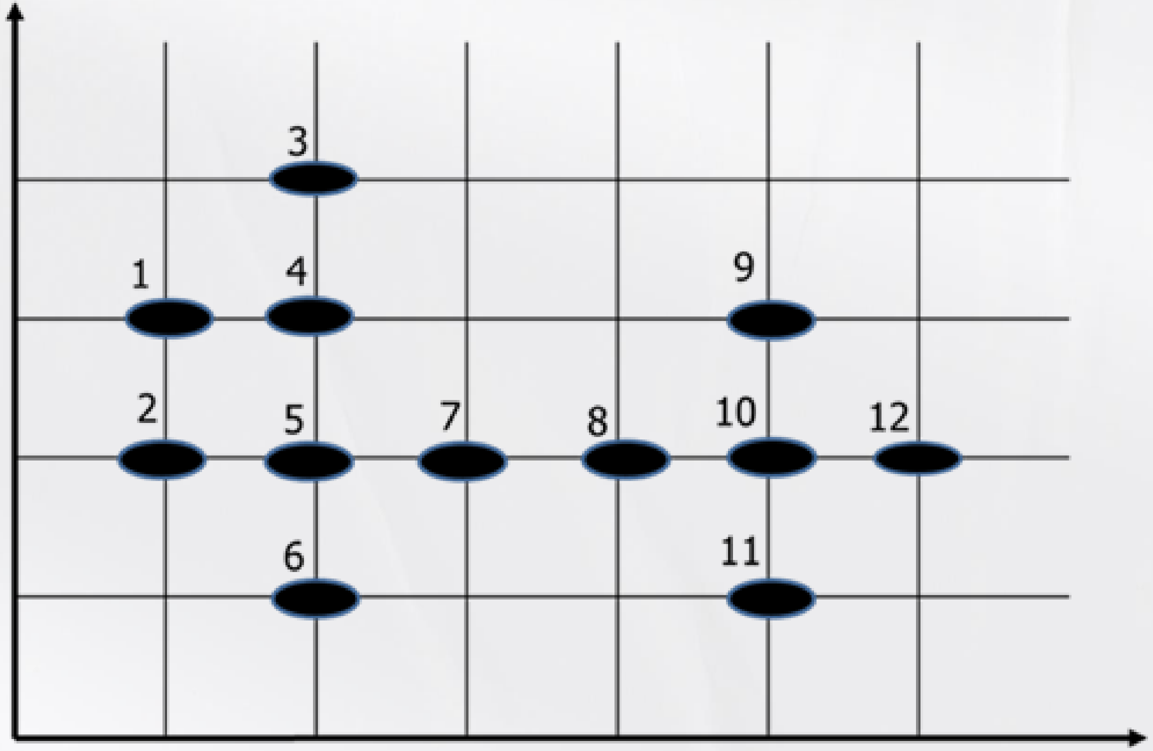
⑦ Randomly select the unvisited point 10 and mark it as visited. Is it a core point. YES!

- Generate a new cluster C2, put point 10 into C2, C2={10}. Put the points in the neighborhood of point 10 into the candidate set N, that is N={8,9,11,12}, where point 8 is visited, and point 9, 11 and 12 are unvisited.



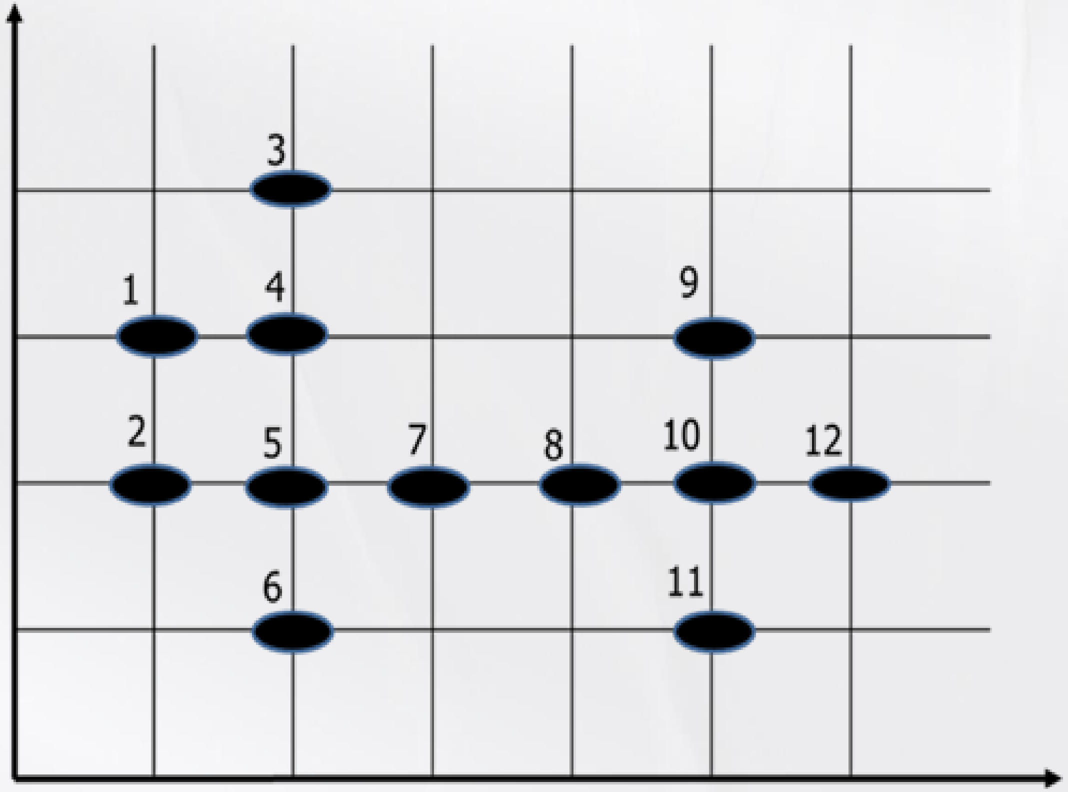
DBSCAN Algorithm

- Select the unvisited point 9 in N and mark it as visited. NOT the core point. Because point 9 does not belong to other clusters, put point 9 into C2, C2={9,10}, N={8,11,12}, where point 8 is visited, points 11 and 12 are unvisited.



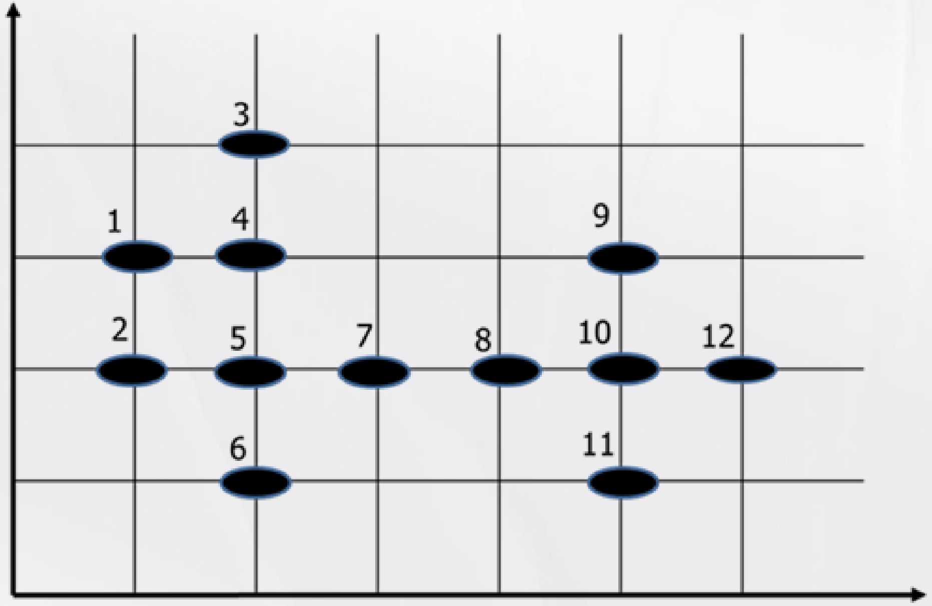
DBSCAN Algorithm

- Select the unvisited point 11 in N, mark it as visited. NOT a core point. Because point 11Does not belong to other clusters, put point 11 into C2, C2={9,10,11}, N={8,12}, where point 8 is visited, point 12 is unvisited.



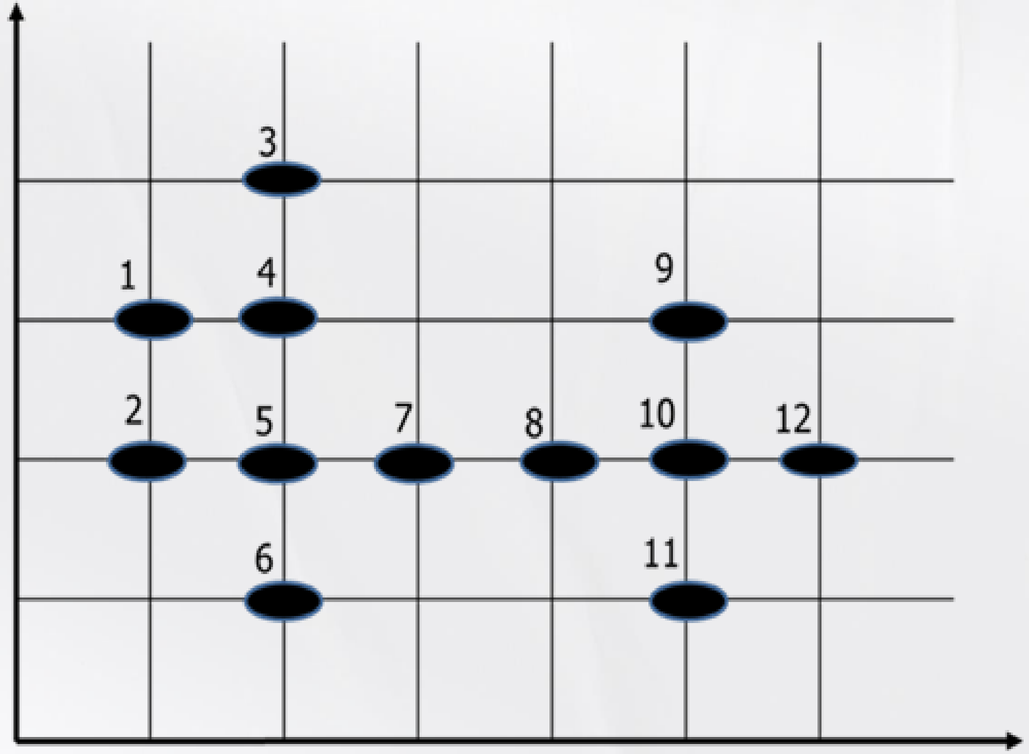
DBSCAN Algorithm

- Select the unvisited point 12 in N, mark it as visited. NOT a core point. Because point 12 does not belong to other clusters, put point 12 into C2, C2={9,10,11,12}, N={8}, where point 8 is visited.



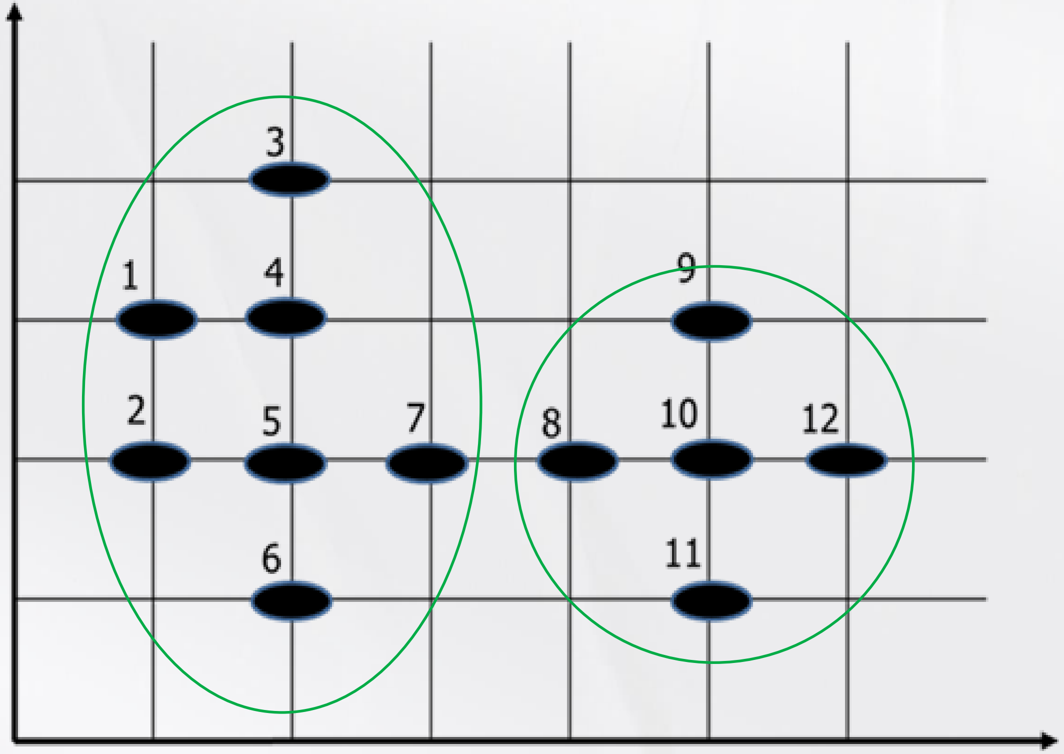
DBSCAN Algorithm

- Although point 8 in N is visited, but it does not belong to other clusters, put it in C2, C2={8,9,10,11,12}, N={}. The new cluster C2={8,9,10,11,12} can be obtained.



DBSCAN Algorithm

- Although point 8 in N is visited, but it does not belong to other clusters, put it in C2, C2={8,9,10,11,12}, N={}. The new cluster C2={8,9,10,11,12} can be obtained.
- All points in the data set D are visited, so the original data set D is divided into two clusters C1={1,2,3,4,5,6,7} and C2={8,9,10,11,12}.



03

Details of determining DBSCAN parameters

k-distance plot

The method computes the k-nearest neighbor distances in a matrix of points.

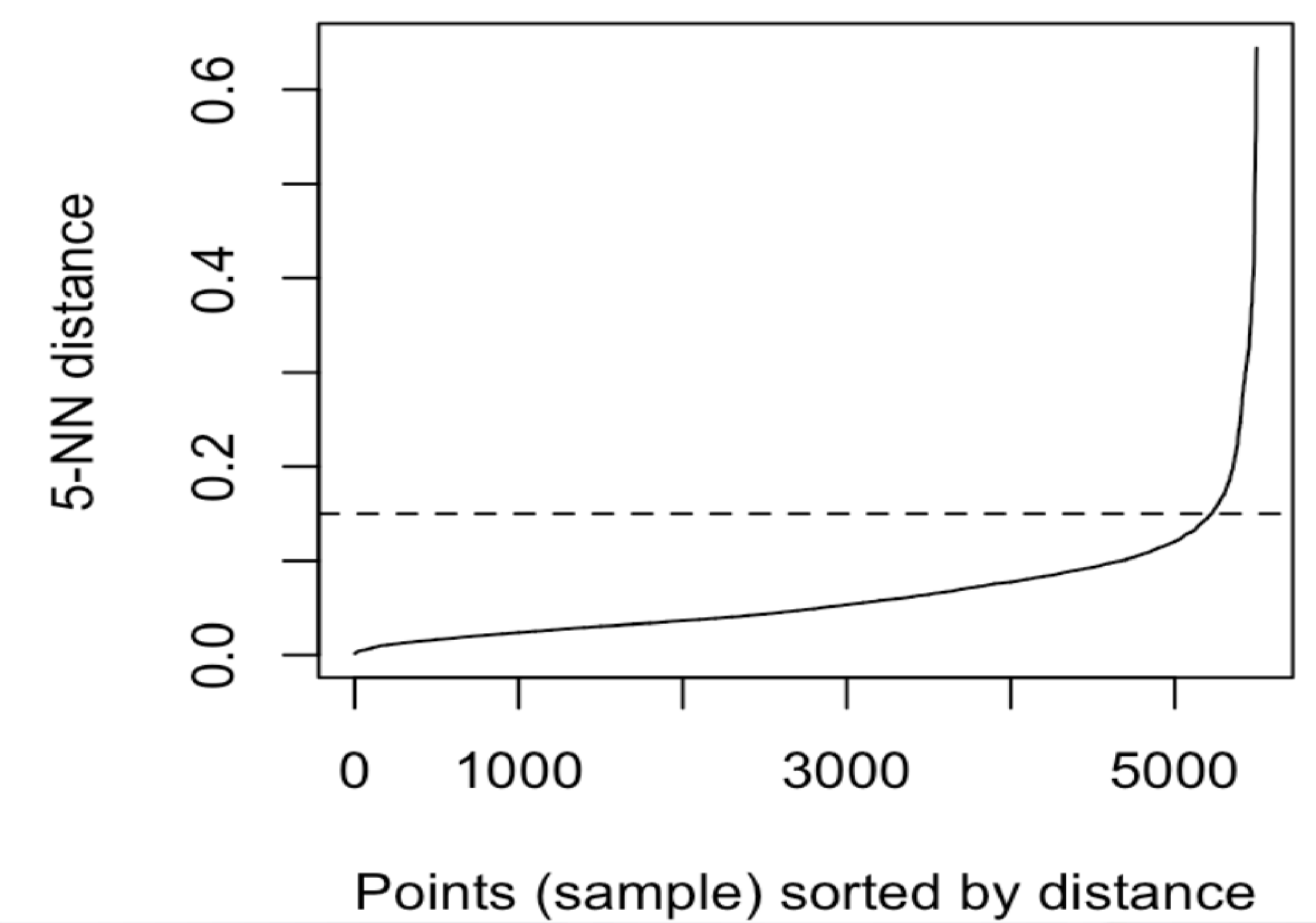
The idea is to calculate the average of the distances of every point to its k nearest neighbors. The value of k will be specified by the user and corresponds to MinPts.

Next, these k-distances are plotted in ascending order. The aim is to determine the “knee”, which corresponds to the optimal epsilon parameter.

A knee corresponds to a threshold where a sharp change occurs along the k-distance curve.

Details of determining DBSCAN parameters

It can be seen that the optimal eps value is around a distance of 0.15.



Details of determining DBSCAN parameters

Some general rules for determining MinPts

The MinPts value is better to be set using domain knowledge and familiarity with the data set. Here are a few rules of thumb for selecting the MinPts value:

- The larger the data set, the larger the value of MinPts should be
- If the data set is noisier, choose a larger value of MinPts
- Generally, MinPts should be greater than or equal to the dimensionality of the data

2*dimension-1   Minpts=k+1

DBSCAN VS K-means

K-means	DBSCAN
Cluster formed are more or less spherical or convex	Arbitrary
Sensitive to the K	not to be specified
Need parameter: K	Radius and minimum points
More efficient for large datasets	Cannot efficiently handle High dimensional datasets
Doesn't work well with outlier and noisy datasets	Efficiently handles outliers and noisy datasets
Varying densities of the data points doesn't affect K-means algorithm	Doesn't work very well for sparse datasets or for data points with varying density.