Density Clustering

Clustering

Problem description

- Given:

  A data set of N data items which are d-dimensional data feature vectors.

- Task:

  Determine a natural, useful partitioning of the data set into a number of clusters (k) and noise.

Density-Based Clustering

- A cluster is defined as a connected dense component which can grow in any direction that density leads.
- Density, connectivity and boundary
- Arbitrary shaped clusters and good scalability

K-means Clustering	DBScan Clustering
Clusters formed are more or less spherical or convex in shape and must have same feature size.	Clusters formed are arbitrary in shape and may not have same feature size.
K-means clustering is sensitive to the number of clusters specified.	Number of clusters need not be specified.
K-means Clustering is more efficient for large datasets.	DBSCan Clustering can not efficiently handle high dimensional datasets.
K-means Clustering does not work well with outliers and noisy datasets.	DBScan clustering efficiently handles outliers and noisy datasets.
In the domain of anomaly detection, this algorithm causes problems as anomalous points will be assigned to the same cluster as “normal” data points.	DBScan algorithm, on the other hand, locates regions of high density that are separated from one another by regions of low density.
It requires one parameter : Number of clusters (K)	It requires two parameters : Radius(R) and Minimum Points(M)
Varying densities of the data points doesn’t affect K-means clustering algorithm.	DBScan clustering does not work very well for sparse datasets or for data points with varying density.

Two Major Types of Density-Based Clustering Algorithms

- Connectivity based:

  DBSCAN, GDBSCAN, OPTICS and DBCLASD
- Density function based:

  DENCLUE

DBSCAN [Ester et al.1996]

- Clusters are defined as Density-Connected Sets (wrt. Eps, MinPts)
- Density and connectivity are measured by local distribution of nearest neighbor
- Target low dimensional spatial data

DBSCAN

- Definition 1: Eps-neighborhood of a point

  $N_{Eps}(p) = \{q \in D \mid dist(p,q) \leq Eps\}$

- Definition 2: Core point

  $|N_{Eps}(q)| \geq MinPts$

DBSCAN

- Definition 3: Directly density-reachable

  A point p is directly density-reachable from a point q wrt. Eps, MinPts if

  1) $p \in N_{Eps}(q)$ and
  2) $|N_{Eps}(q)| \geq MinPts$ (core point condition).

DBSCAN

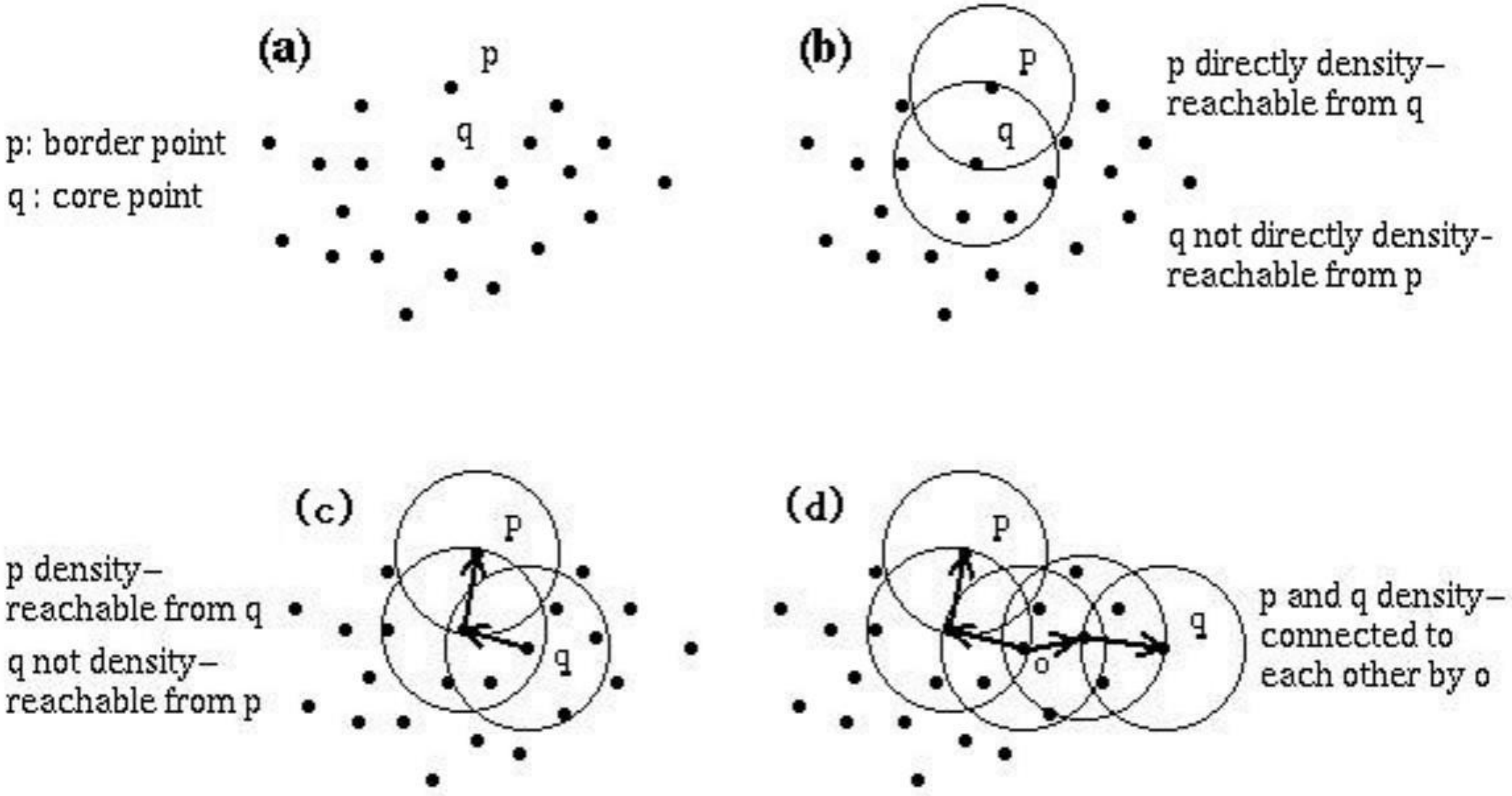
- Definition 4: Density-reachable

  A point p is density-reachable from a point q wrt. Eps and MinPts if there is a chain of points $p_1, ..., p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

- Definition 5: Density-connected

  A point p is density-connected to a point q wrt. Eps and MinPts if there is a point o such that both, p and q are density-reachable from o wrt. Eps and MinPts.

DBSCAN

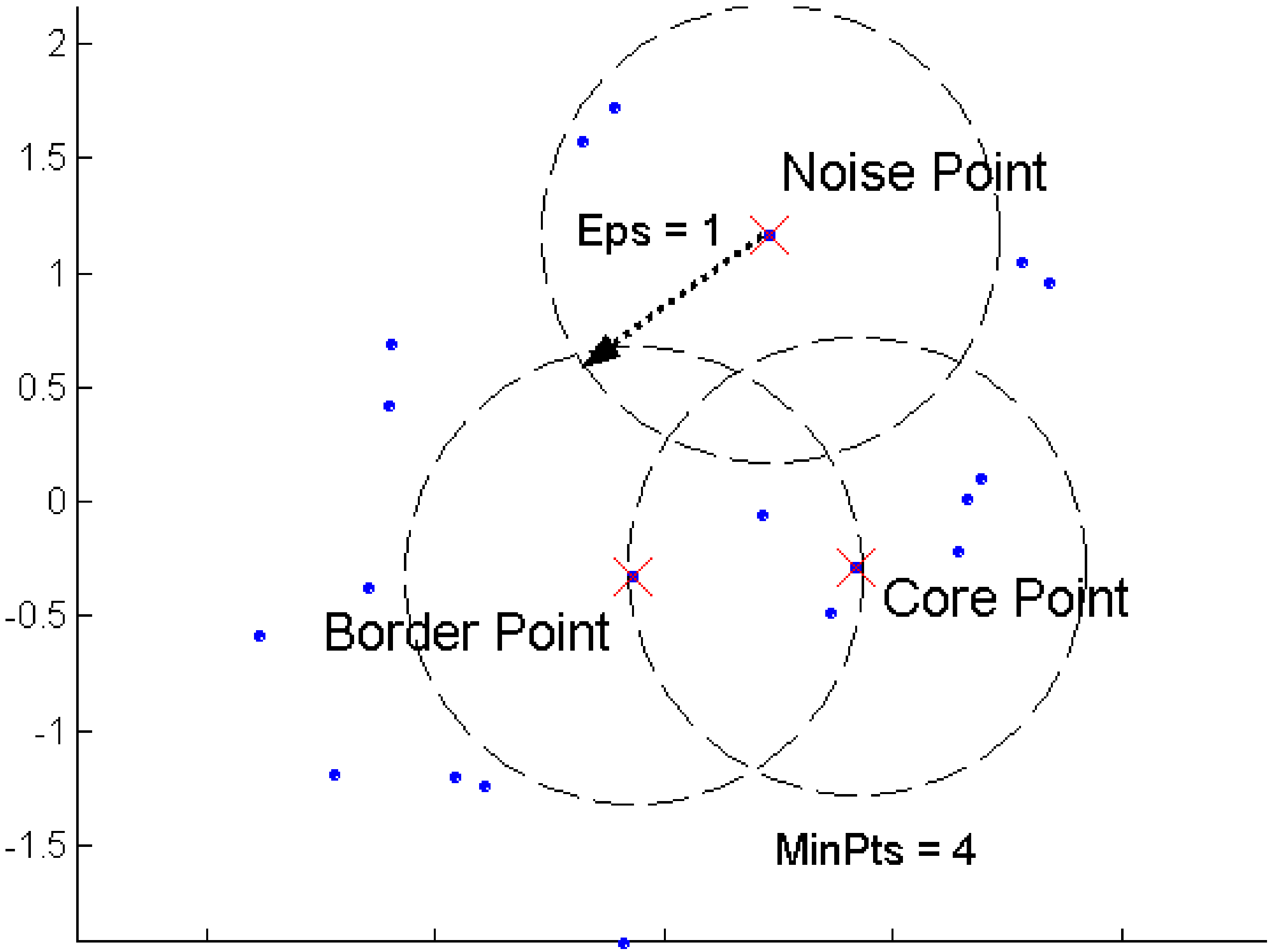

(a)
p
p: border point
q : core point
q
(b)
p
q
p directly density–
reachable from q
q not directly density-
reachable from p
(c)
p
p density–
reachable from q
q not density–
reachable from p
q
(d)
p
o
q
p and q density–
connected to
each other by o

DBSCAN

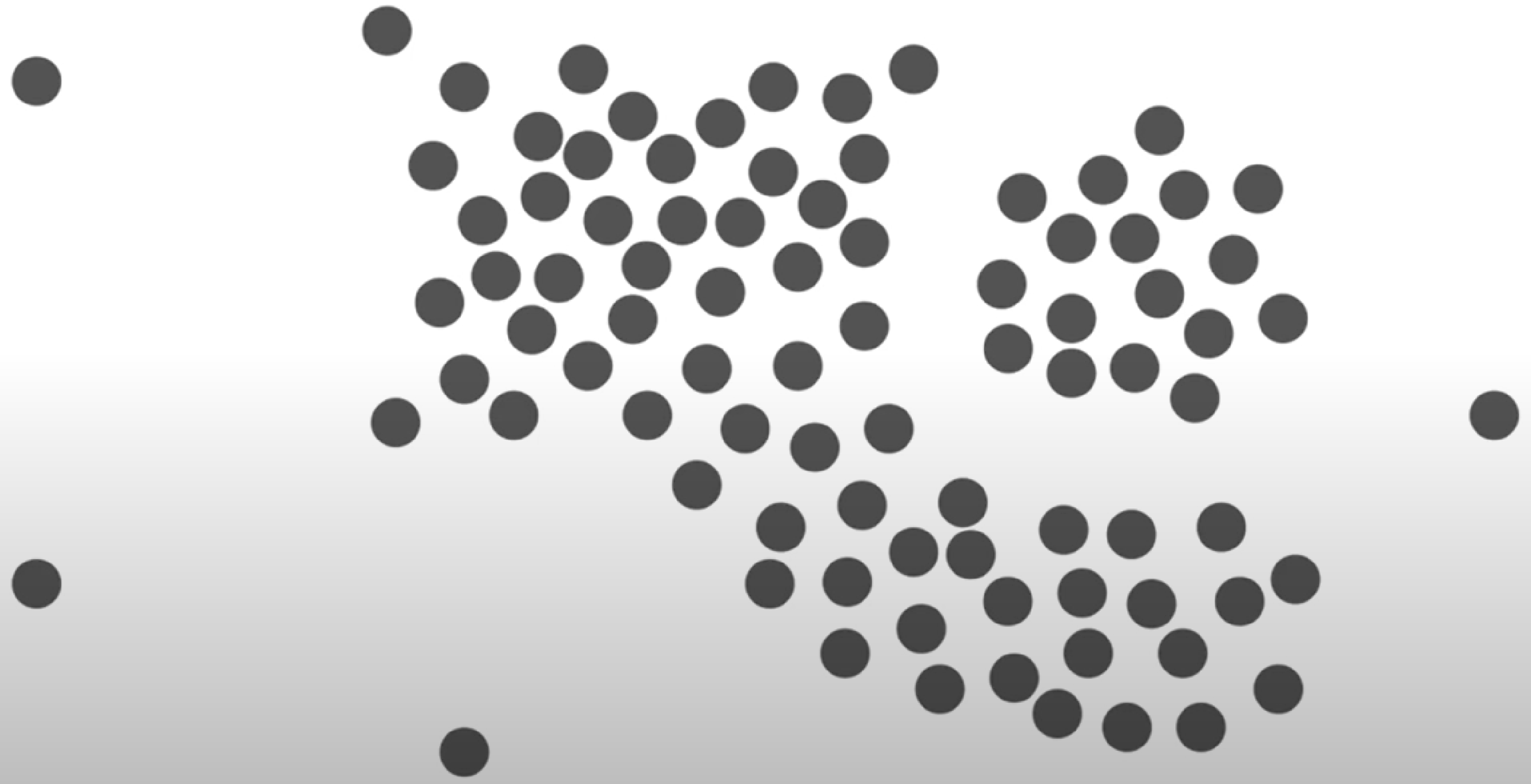
DBSCAN is a density-based algorithm.

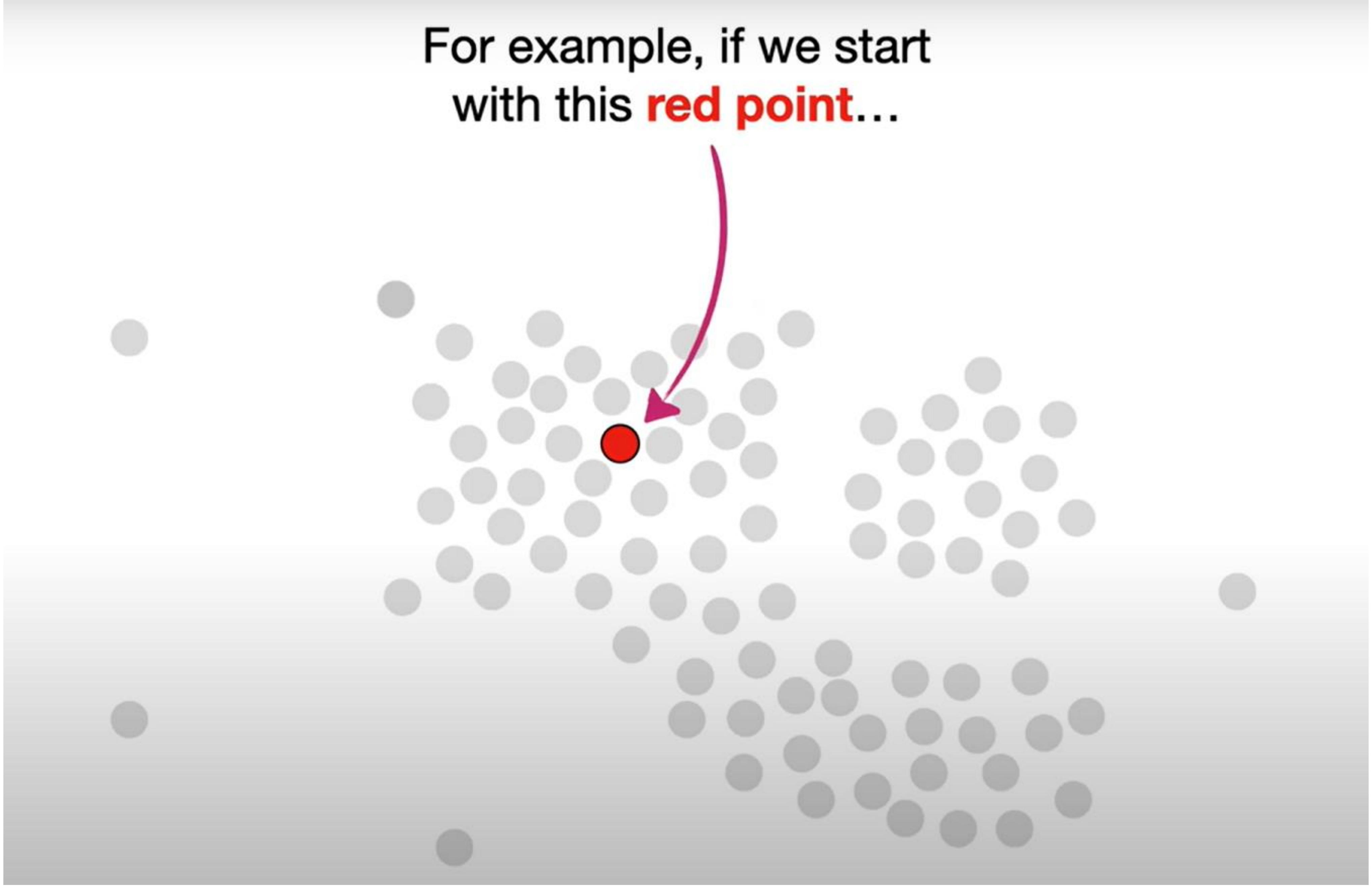
- Density = number of points within a specified radius r (Eps)
- A point is a core point if it has more than a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point.

DBSCAN: Core, Border, and Noise Points

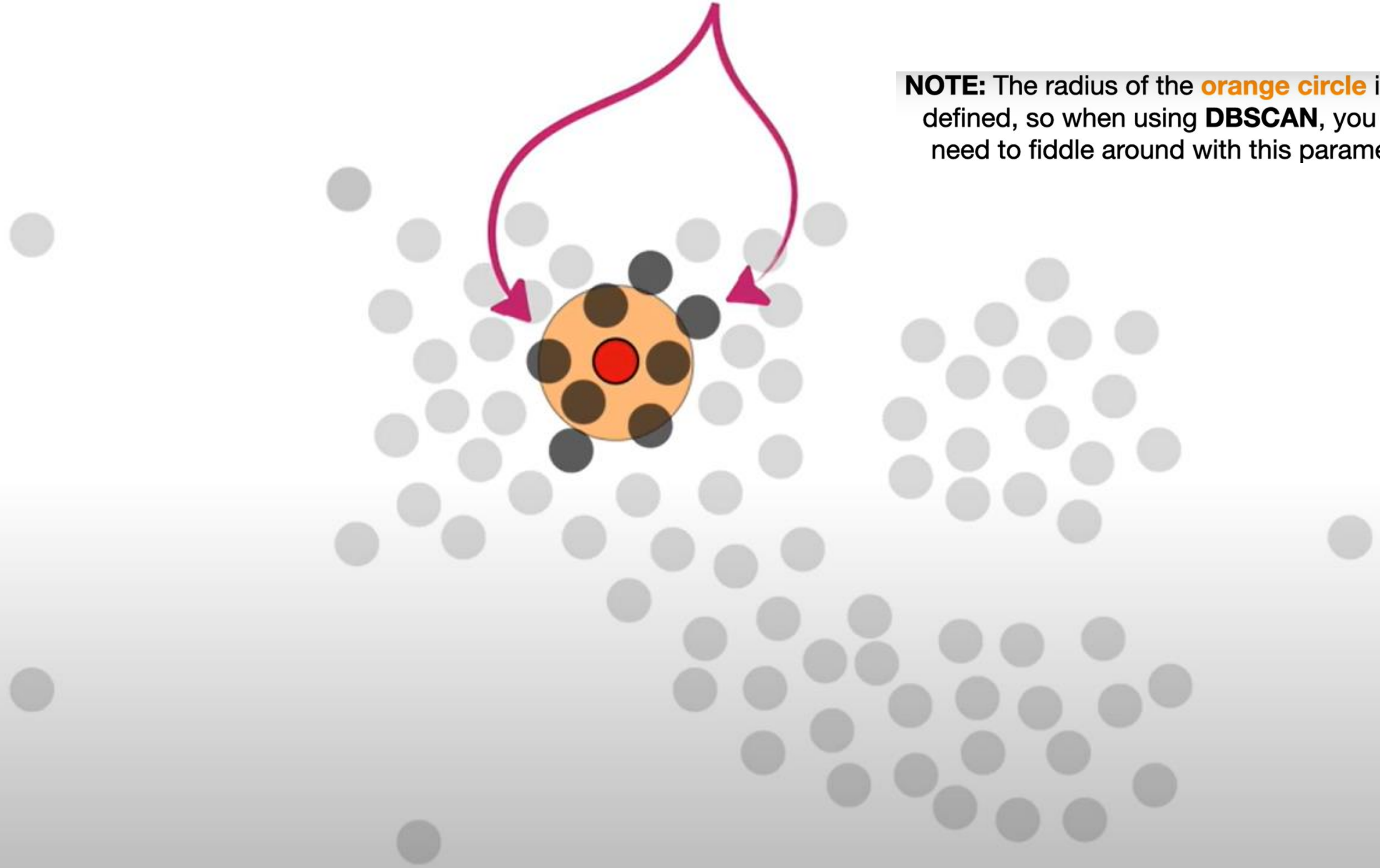


…the first thing we can do is count the number of points close to each point.



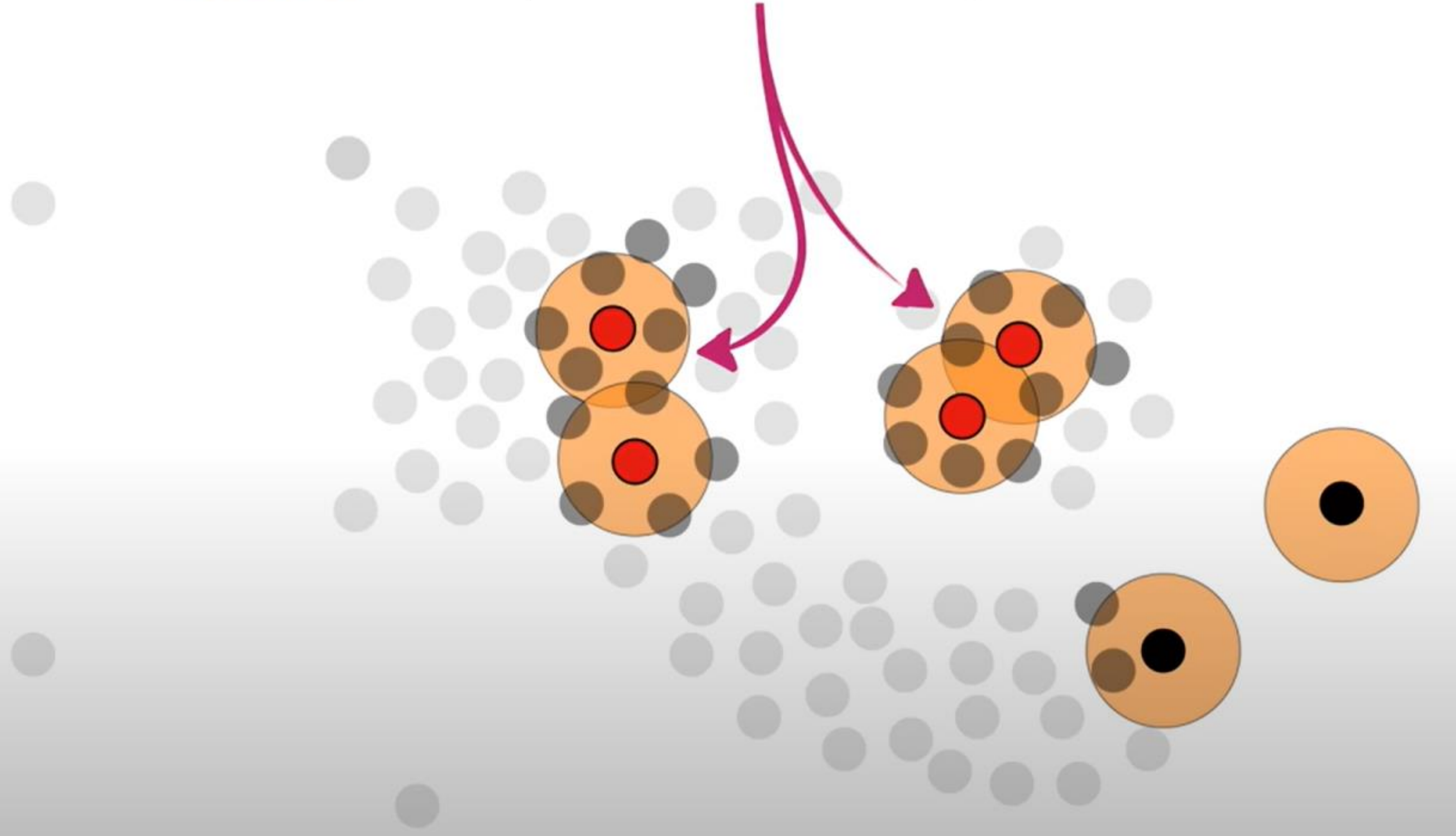

For example, if we start
with this red point…

...then we see that the **orange circle** overlaps, at least partially, **8** other points.

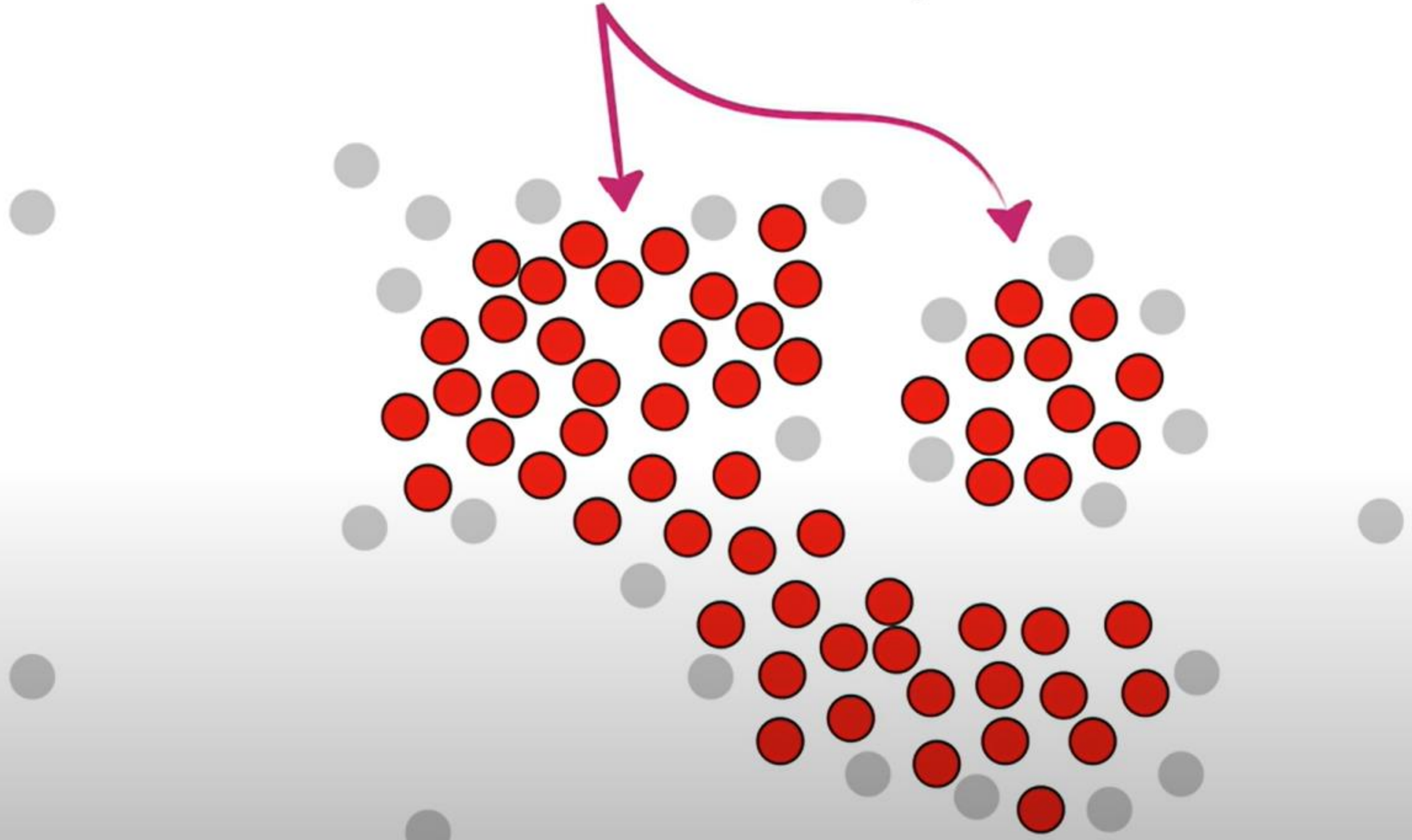


NOTE: The radius of the **orange circle** is user defined, so when using **DBSCAN**, you may need to fiddle around with this parameter.

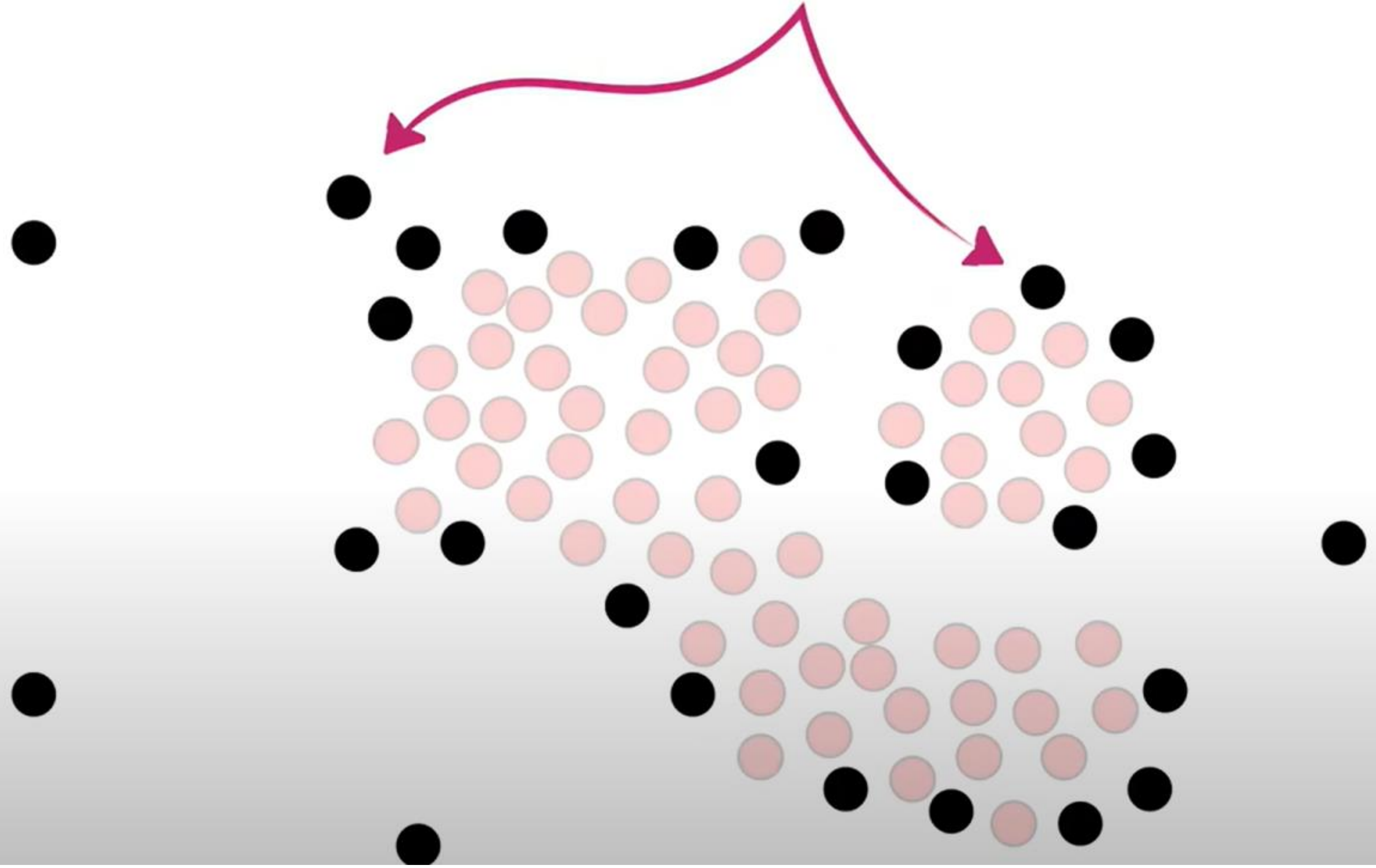
Anyway, these **4** points are some of the **Core Points**, because their **orange circles** overlap at least **4** other points...



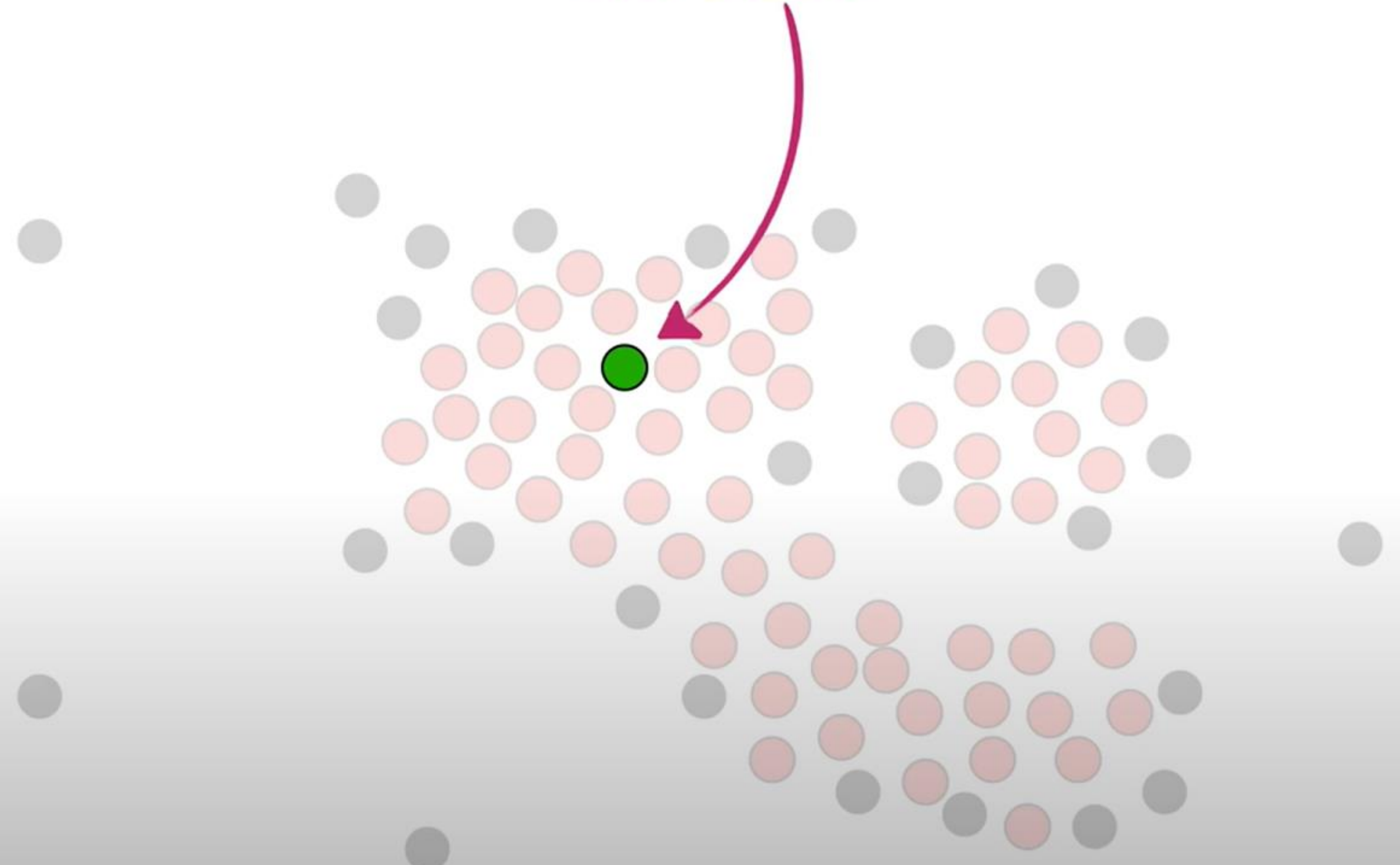
Ultimately, we can call all of these **red points Core Points** because they are all close to **4** or more other points…

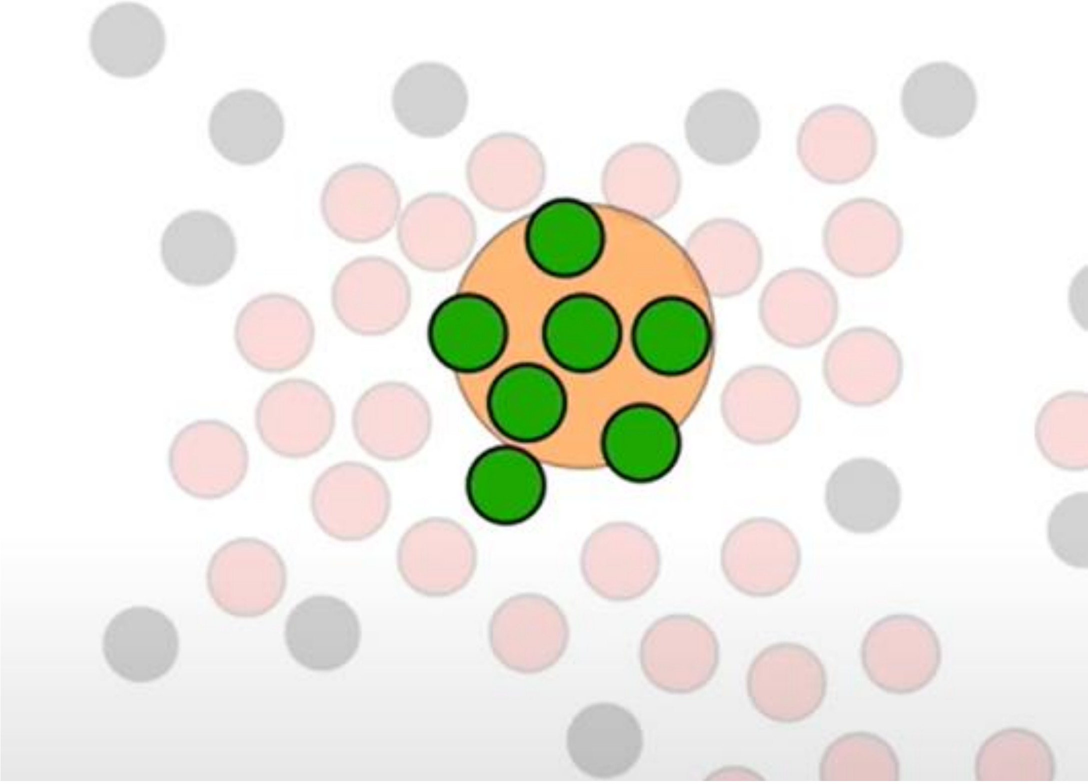


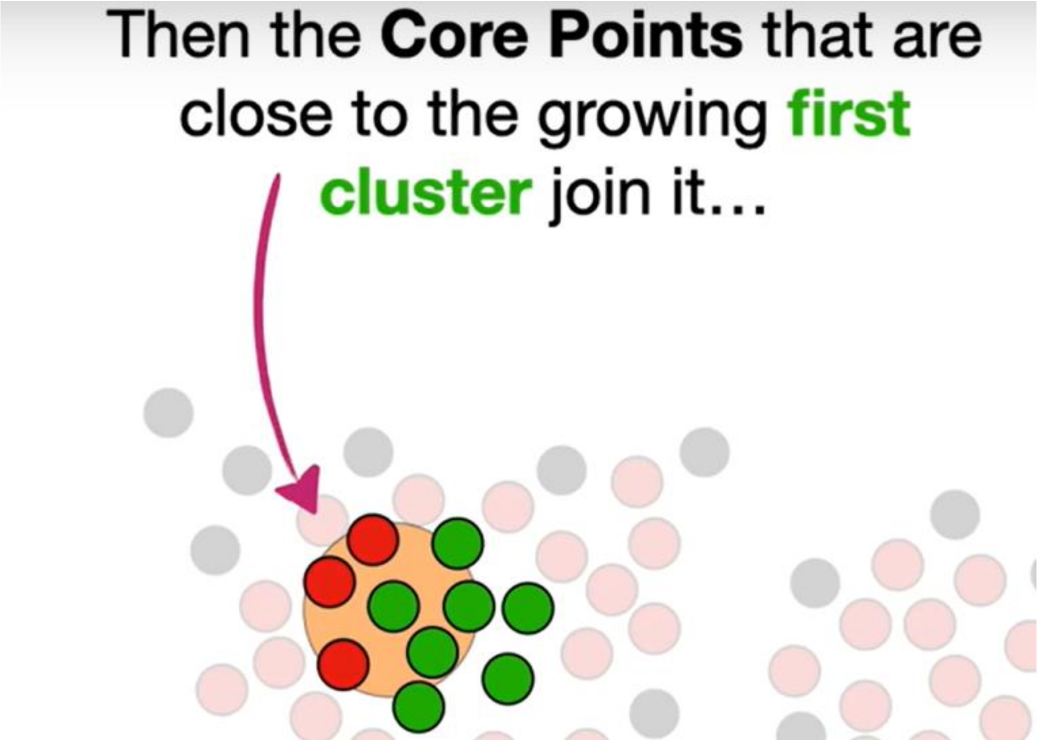
...and the remaining points
are **Non-Core**.



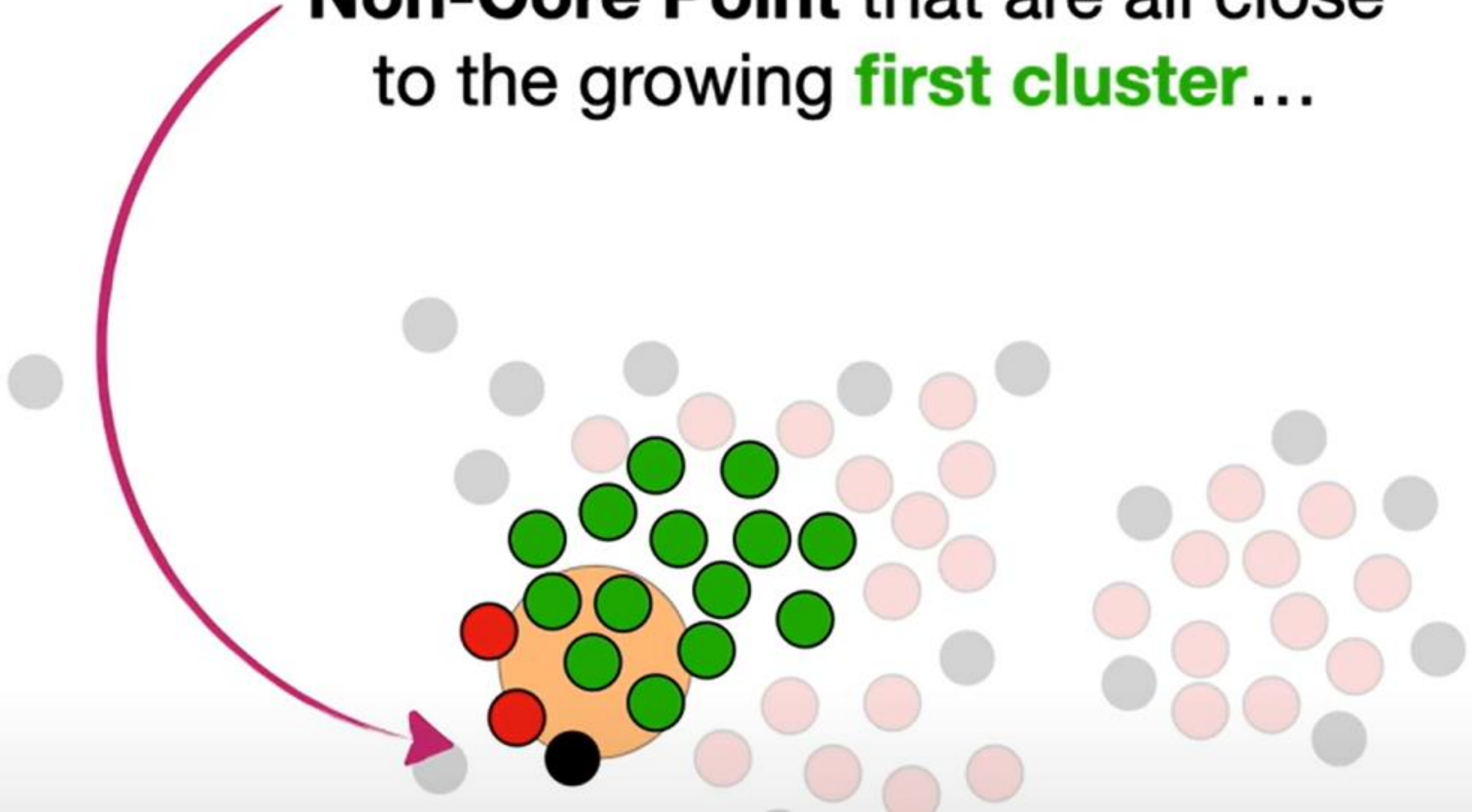
...and assign it to the **first cluster**.



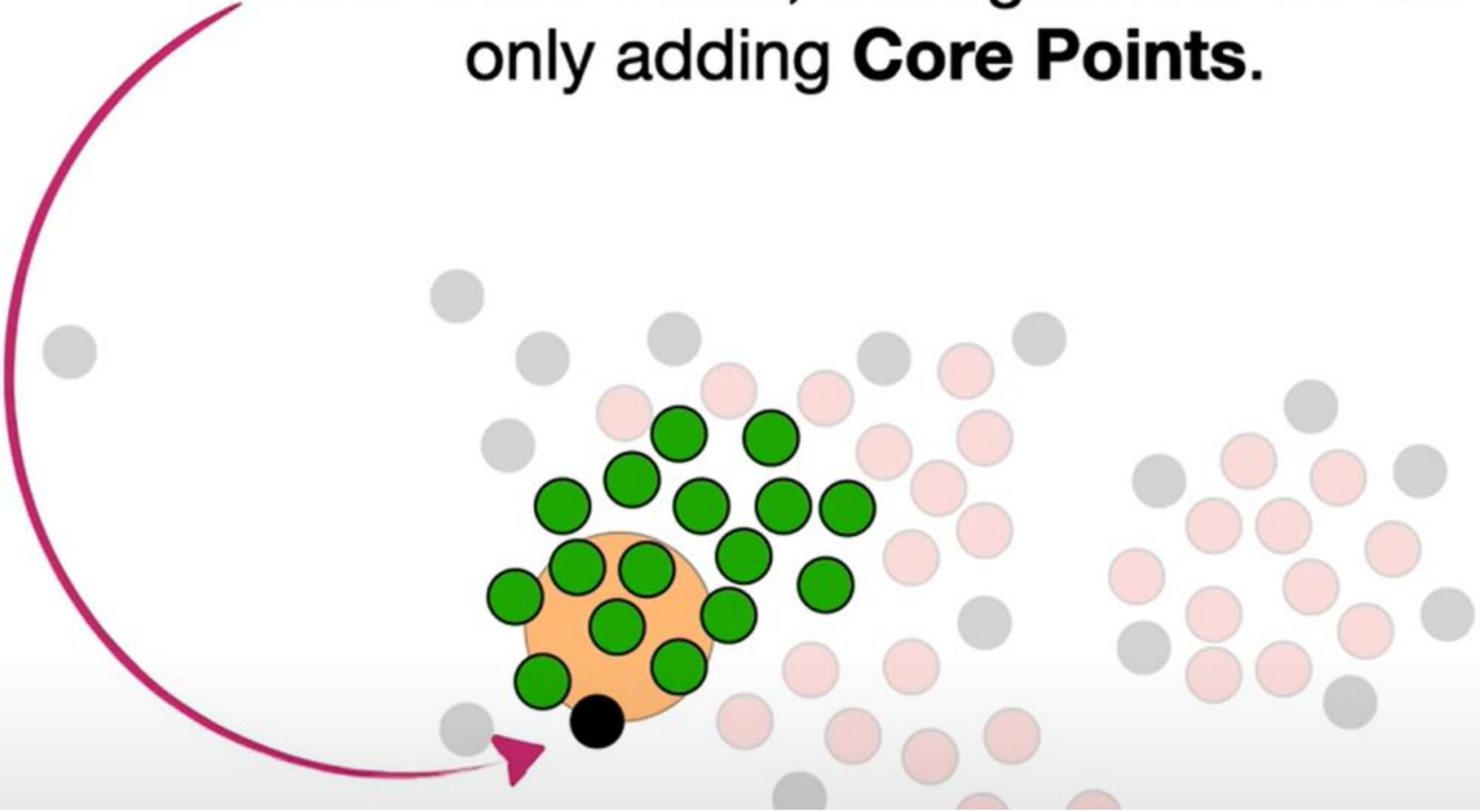



Then the **Core Points** that are close to the growing **first cluster** join it…

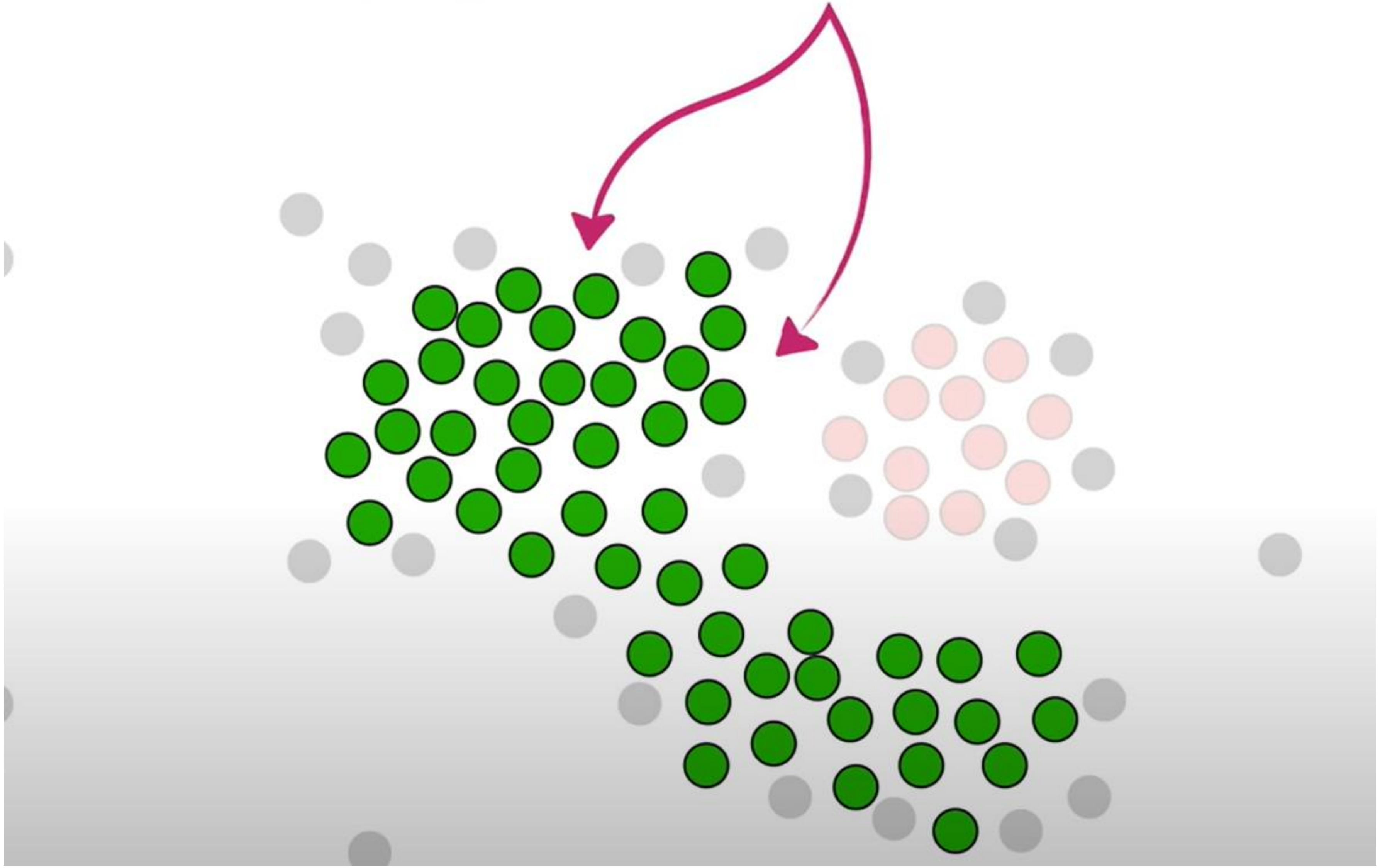
Here we see **2 Core Points** and **1 Non-Core Point** that are all close to the growing **first cluster**…



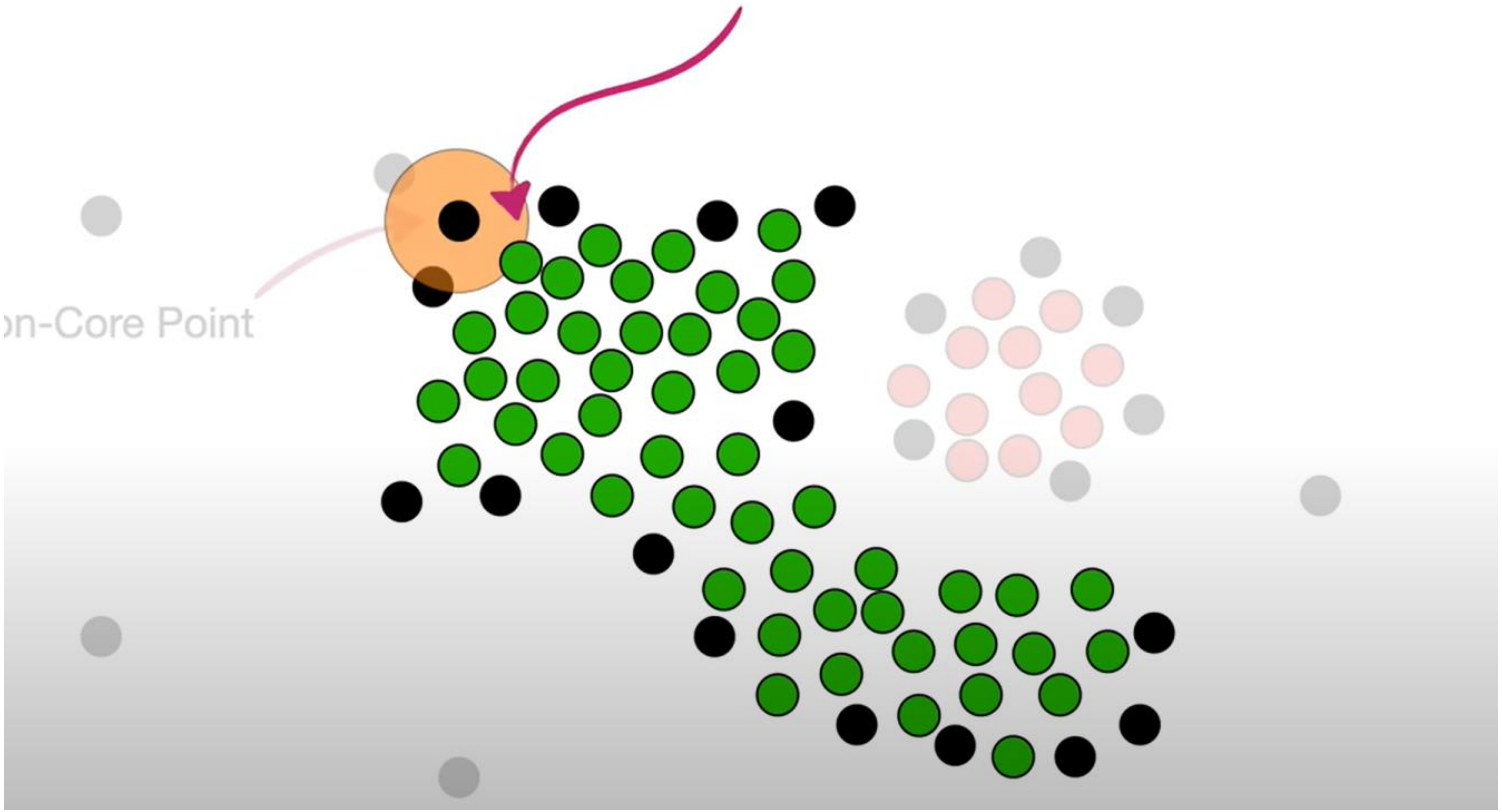
That said, eventually we will add this **Non-Core Point**, but right now we are only adding **Core Points**.



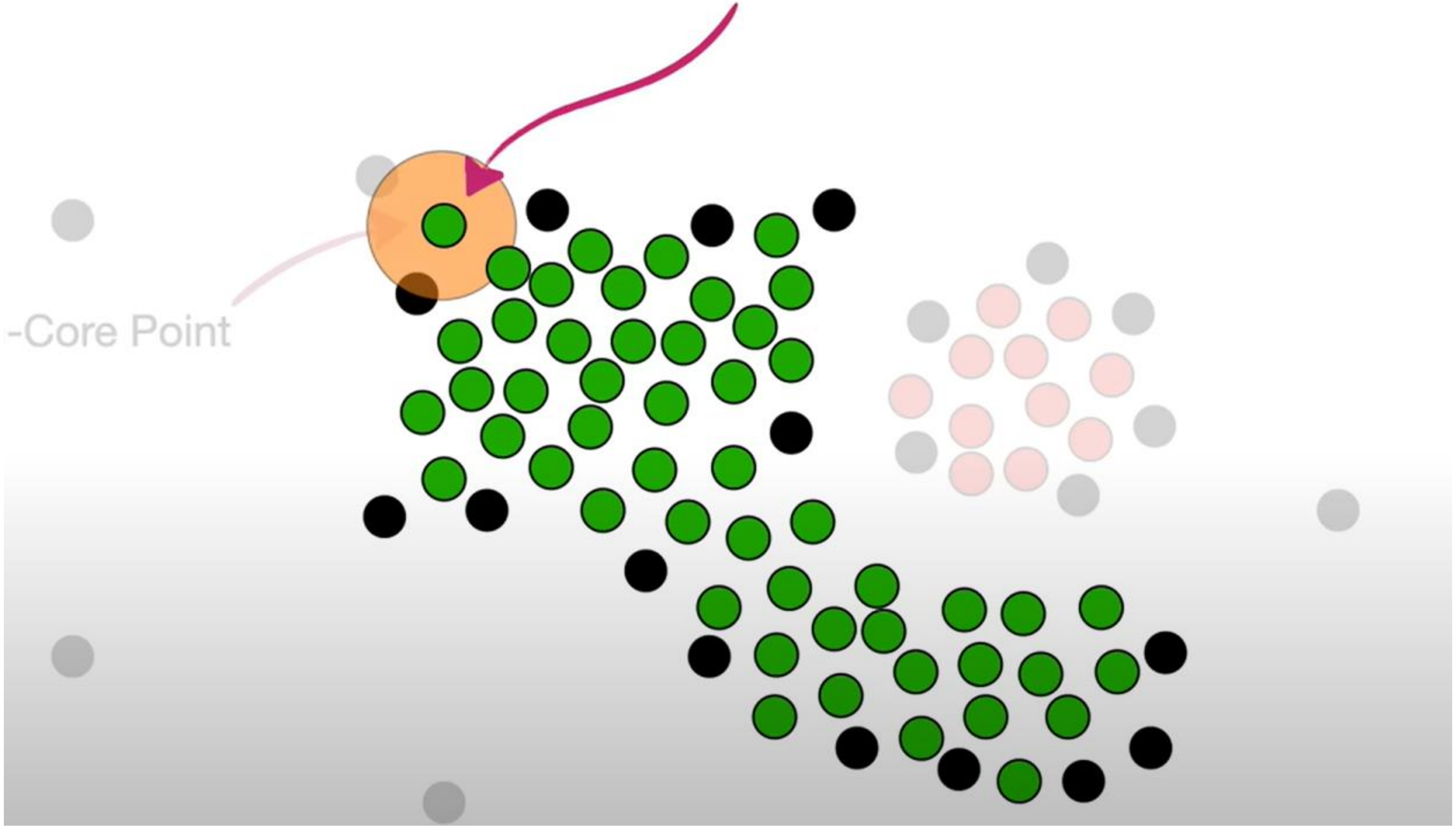
NOTE: At this point, every single point in the **first cluster** is a **Core Point**...



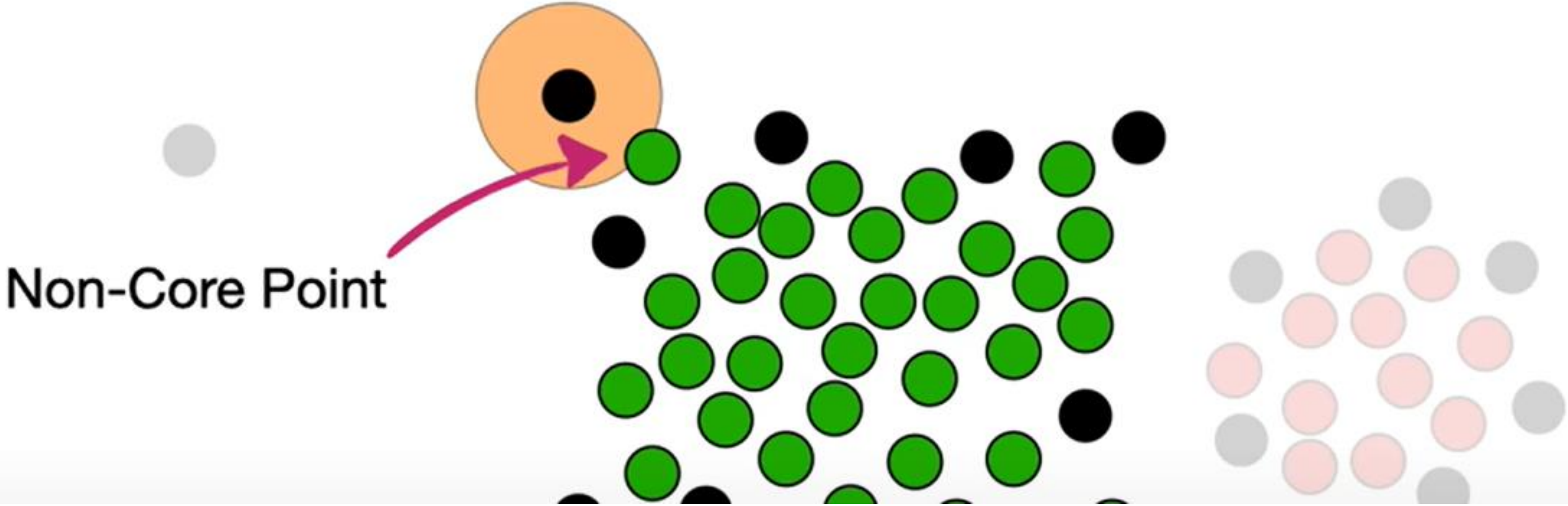
…is close to a **Core Point** in
the **first cluster**…



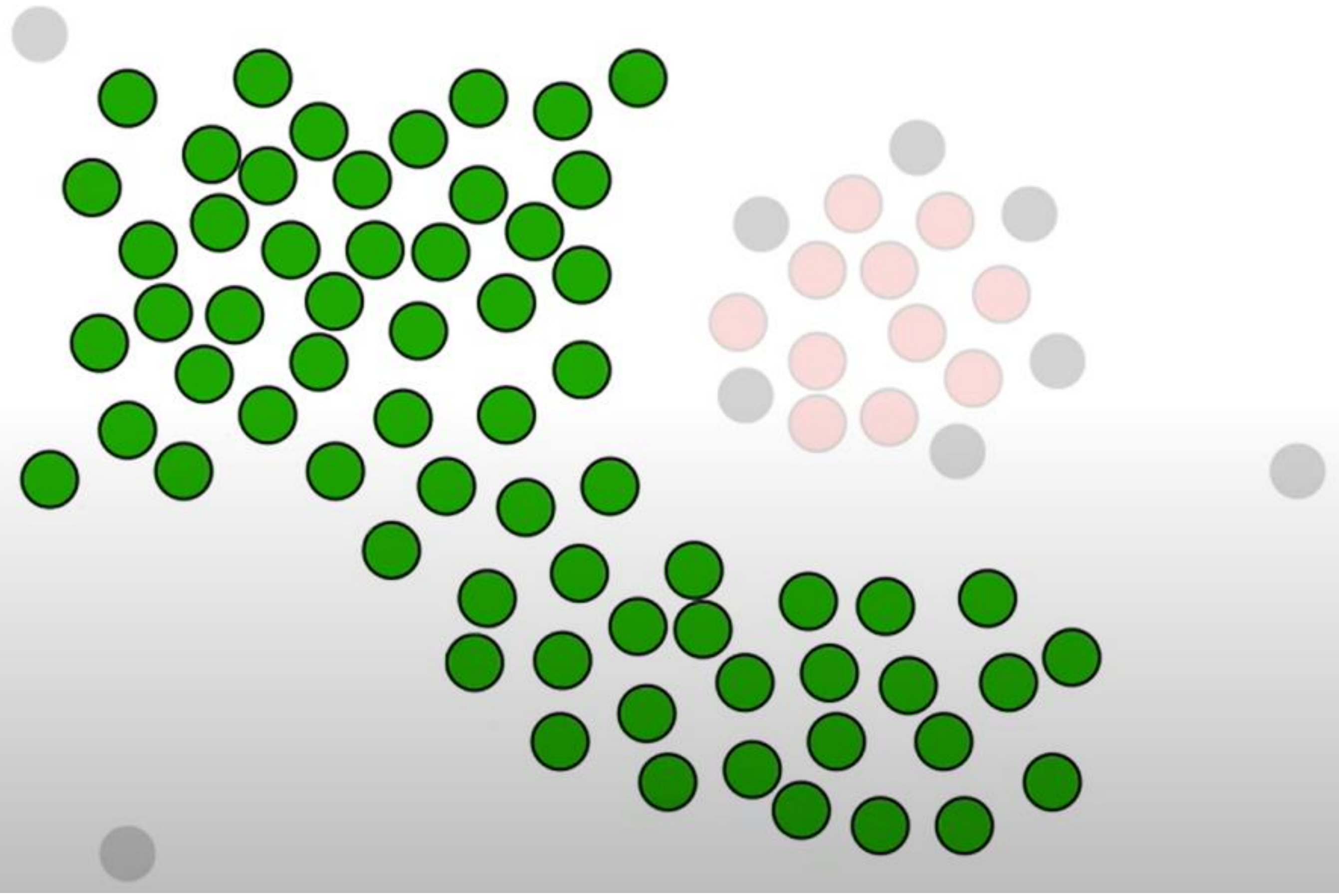
…so we add it to the **first cluster**.



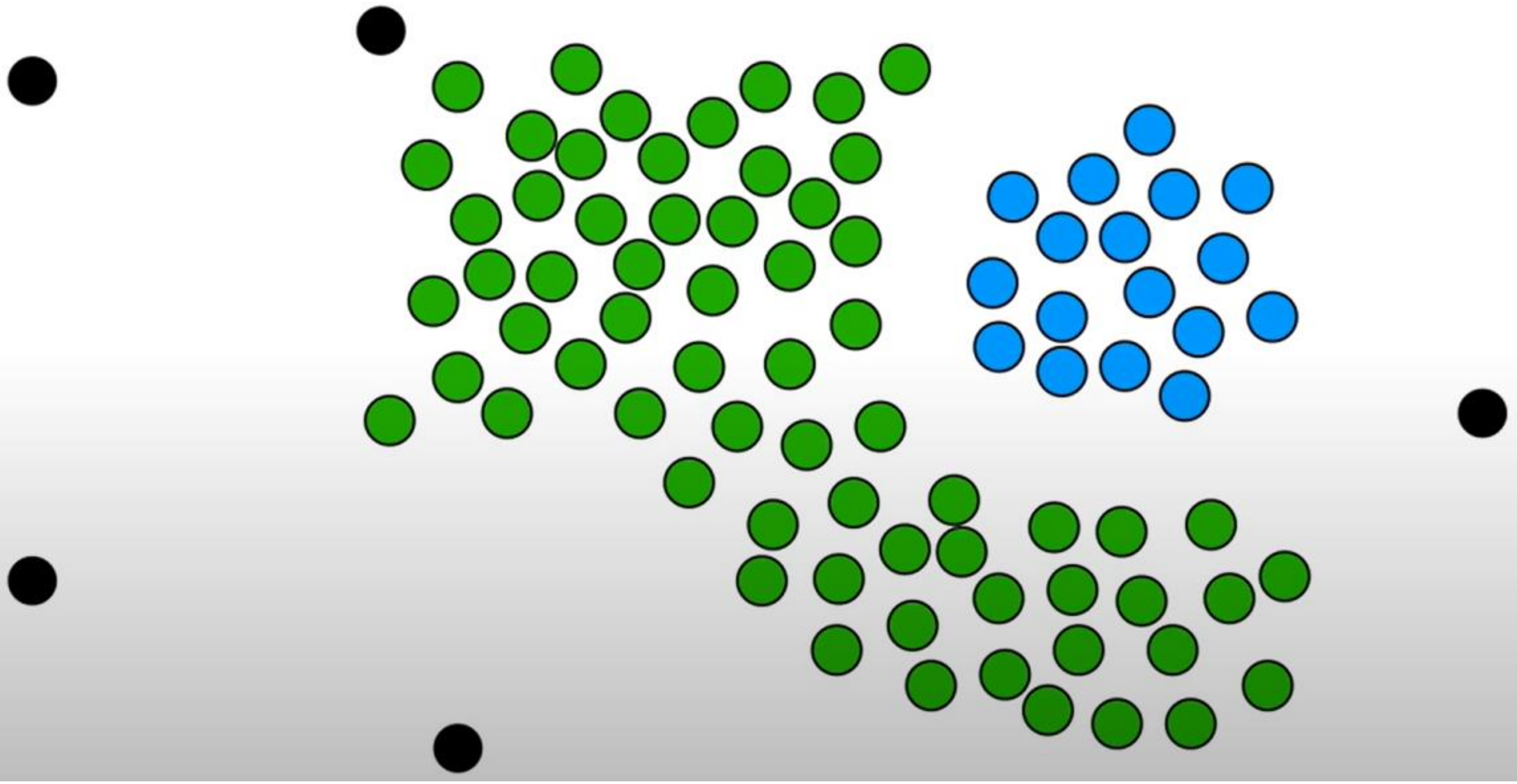
So, unlike **Core Points, Non-Core Points** can only join a cluster. They can not extend it further.



Now we add all of the **Non-Core Points** that are close **Core Points** in the **first cluster** to the **first cluster**.



Lastly, because all of **Core Points** have been assigned to a cluster, we're done making new clusters…



Example

Point	X	y
A	0.5	1.2
B	1.0	1.8
C	1.2	1.5
D	3.8	3.2
E	3.9	3.8
F	4.5	4.2
G	2.0	2.2
H	2.3	2.0

Point	A	B	C	D	E	F	G	H
A	0	0.78	0.76	3.76	4.36	5.02	1.86	2.05
B	0.78	0	0.36	3.10	3.74	4.45	1.28	1.37
C	0.76	0.36	0	2.78	3.41	4.08	0.96	1.06
D	3.76	3.10	2.78	0	0.61	1.22	1.84	1.58
E	4.36	3.74	3.41	0.61	0	0.72	2.50	2.27
F	5.02	4.45	4.08	1.22	0.72	0	3.31	3.05
G	1.86	1.28	0.96	1.84	2.50	3.31	0	0.36
H	2.05	1.37	1.06	1.58	2.27	3.05	0.36	0

minPts=2 and eps=0.8

Example

Point	A	B	C	D	E	F	G	H
A	0	0.78	0.76	3.76	4.36	5.02	1.86	2.05
B	0.78	0	0.36	3.10	3.74	4.45	1.28	1.37
C	0.76	0.36	0	2.78	3.41	4.08	0.96	1.06
D	3.76	3.10	2.78	0	0.61	1.22	1.84	1.58
E	4.36	3.74	3.41	0.61	0	0.72	2.50	2.27
F	5.02	4.45	4.08	1.22	0.72	0	3.31	3.05
G	1.86	1.28	0.96	1.84	2.50	3.31	0	0.36
H	2.05	1.37	1.06	1.58	2.27	3.05	0.36	0

Point	
A	A, B, C
B	A, B, C
C	A, B, C
D	D, E
E	D, E, F
F	E, F
G	G, H
H	G, H

minPts=2 and eps=0.8

Example

Point	
A	A, B, C
B	A, B, C
C	A, B, C
D	D, E
E	D, E, F
F	E, F
G	G, H
H	G, H

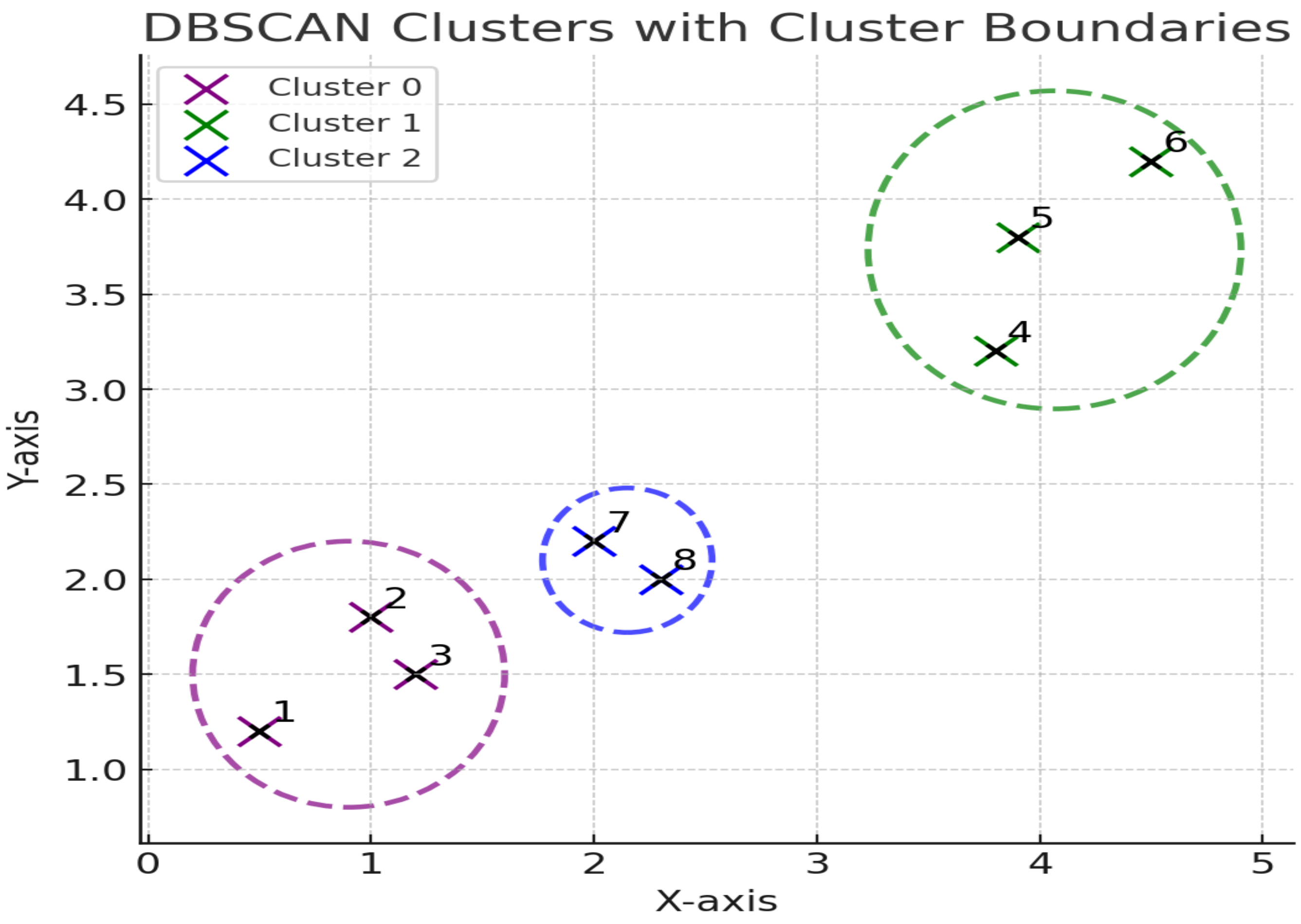
Point	Status	
A	Core	
B	Core	
C	Core	
D	Core	
E	Core	
F	Core	
G	Core	
H	Core	

Cluster	Points
1	A, B, C
2	D,E,F
3	G,H

minPts=2 and eps=0.8

Core Points are A, B, C, D, E, F, G and H

Example



DBSCAN Clusters with Cluster Boundaries
Cluster 0
Cluster 1
Cluster 2
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Y-axis
0
1
2
3
4
5
X-axis
1
2
3
4
5
6
7
8

Example

Point	
A	A, B, C
B	A, B, C
C	A, B, C
D	D, E
E	D, E, F
F	E, F
G	G, H
H	G, H

Point	Status	
A	Core	
B	Core	
C	Core	
D	Noise	Border
E	Core	
F	Noise	Border
G	Noise	Outlier
H	Noise	Outlier

minPts=3 and eps=0.8

D is part of core point (E)

F is part of core point (E)

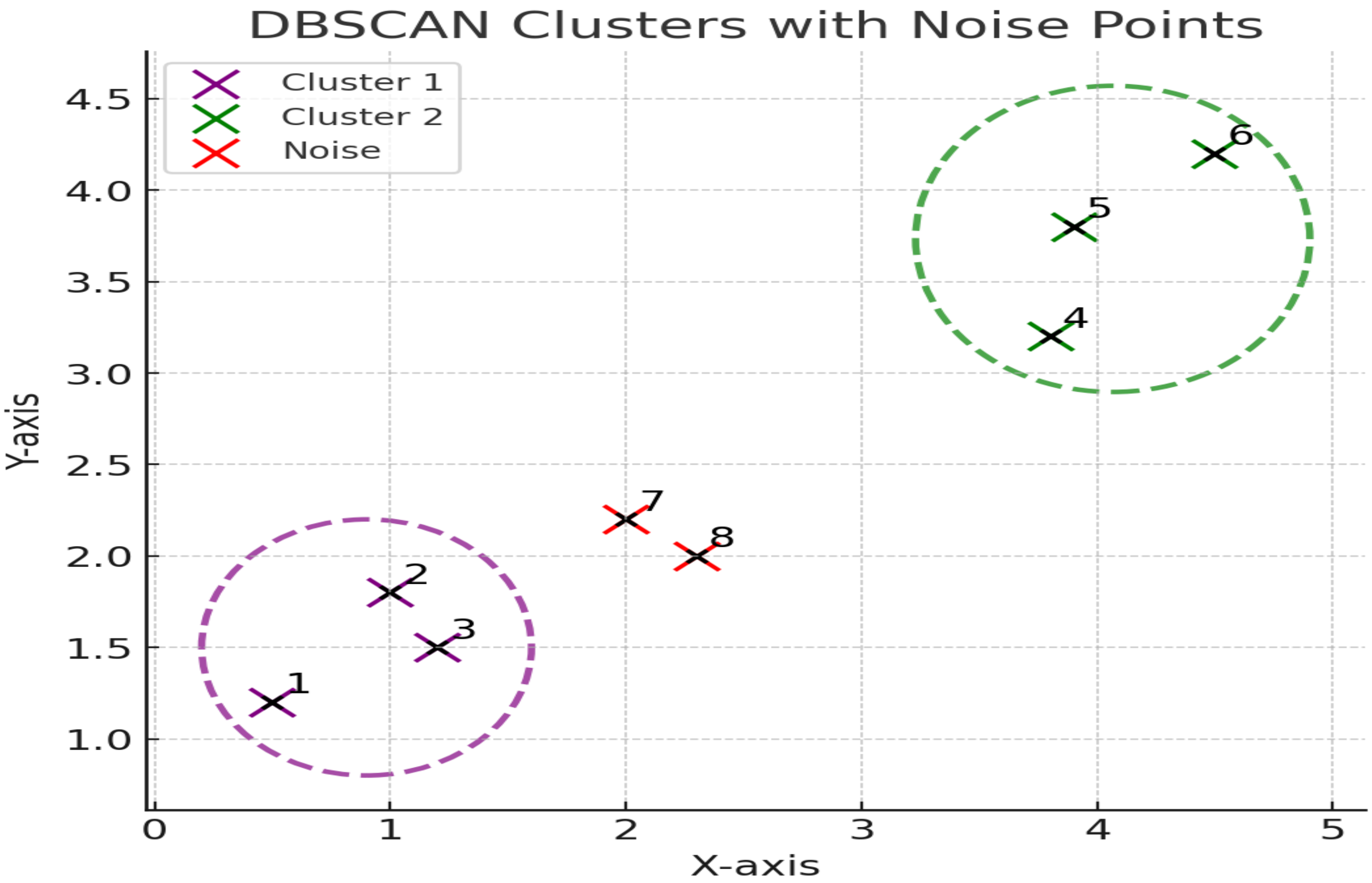
Cluster	Points
1	A, B, C
2	D,E,F
Outlier	G,H

Core Points: A, B, C and E

Border Points: D and F

Outliers/Noise Points: G and H

Example


DBSCAN Clusters with Noise Points
Cluster 1
Cluster 2
Noise
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Y-axis
0
1
2
3
4
5
X-axis
1
2
3
4
5
6
7
8

Practice Problem

- Apply the DBSCAN algorithm to the given data points and
- Create the clusters with
- minPts = 4 and
- epsilon (ε) = 1.9.

Data Points:

P1: (3, 7)	P2: (4, 6)
P3: (5, 5)	P4: (6, 4)
P5: (7, 3)	P6: (6, 2)
P7: (7, 2)	P8: (8, 4)
P9: (3, 3)	P10: (2, 6)
P11: (3, 5)	P12: (2, 4)