Density-Based Clustering

Boston University CS 506 - Lance Galletti

Density-Based Clustering

Goal: cluster together points that are densely packed together.

How should we define density?

Density-Based Clustering

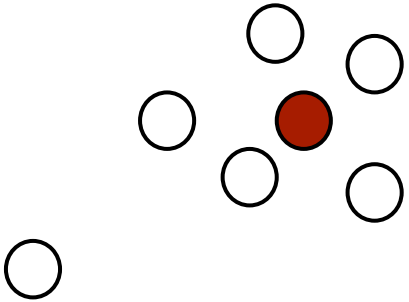
Goal: cluster together points that are densely packed together.

How should we define density?

Given a fixed radius **ε** around a point, if there are at least **min_pts** number of points in that area, then this **area** is dense.

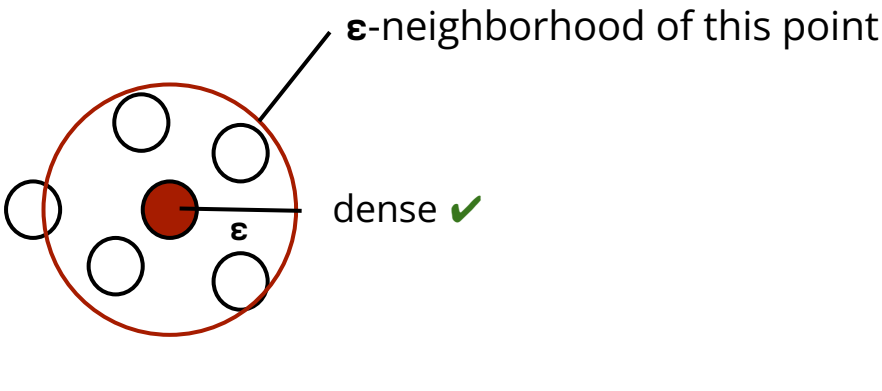
Example

Min_pts = 3



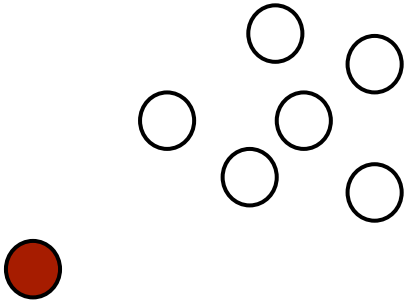
Example

Min_pts = 3



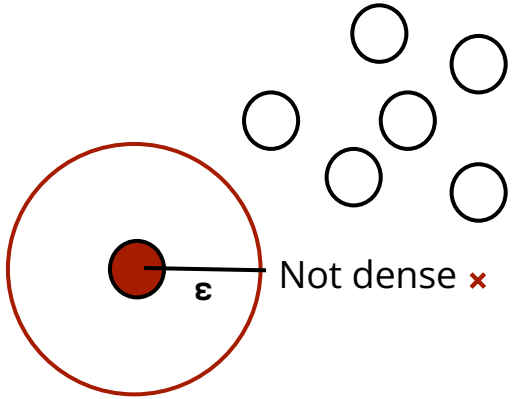
Example

Min_pts = 3



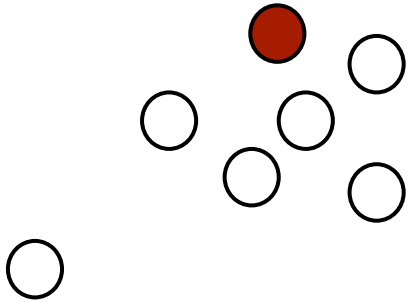
Example

Min_pts = 3



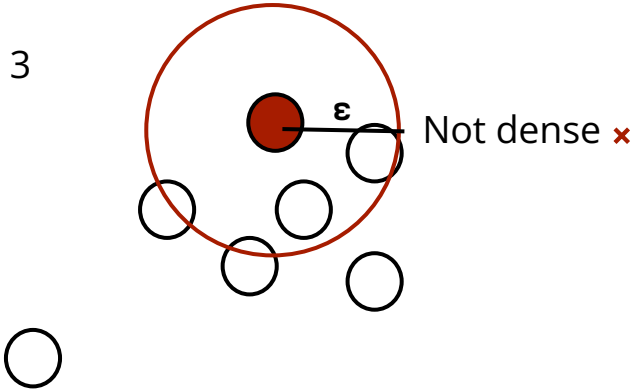
Example

Min_pts = 3



Example

Min_pts = 3



Not dense ×

But… That point was part of a dense section earlier…

weird

Density-Based Clustering

We need to distinguish between points at the core of a dense region and points at the border of a dense region.

Let’s define:

Core point: if its **ε**-neighborhood contains at least **min_pts**

Border point: if it is in the **ε**-neighborhood of a core point

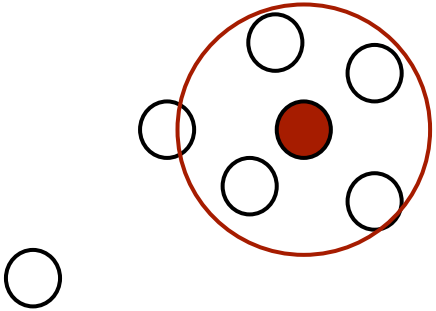
meaning it is in a dense point even if it doesn't generate a dense region

Noise point: if it is neither a core nor border point

Example

core point

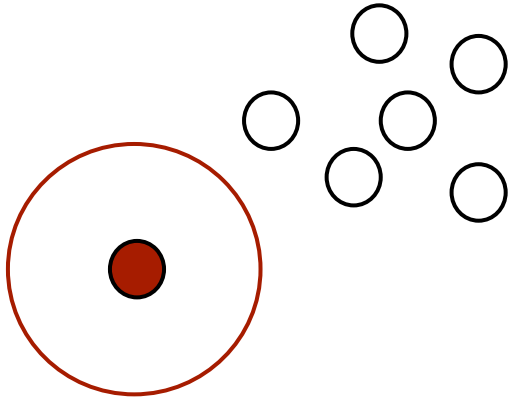
Min_pts = 3



Example

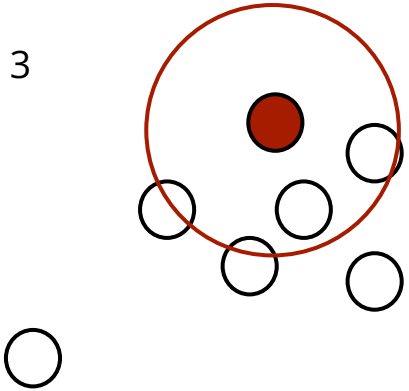
noise point

Min_pts = 3

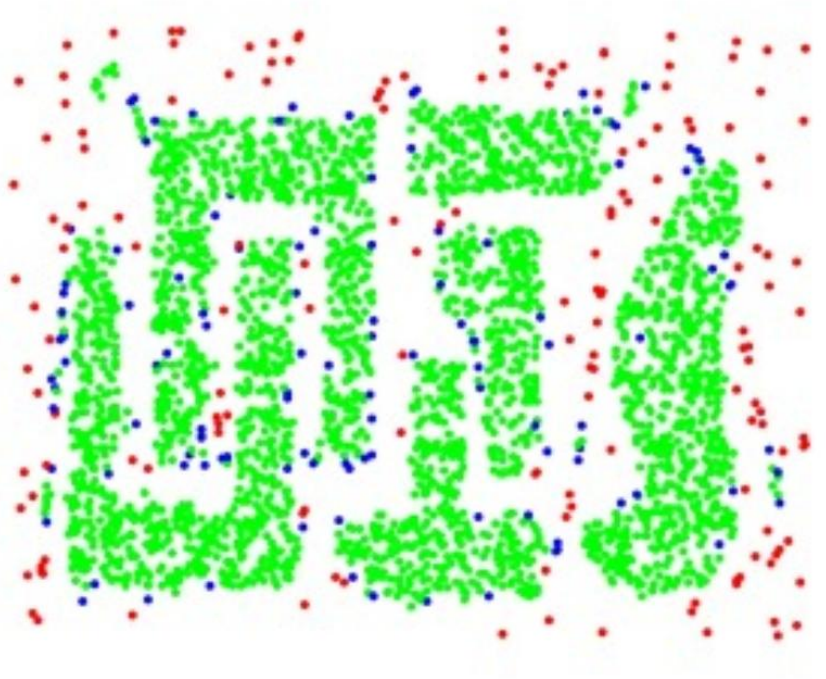


Example

Min_pts = 3

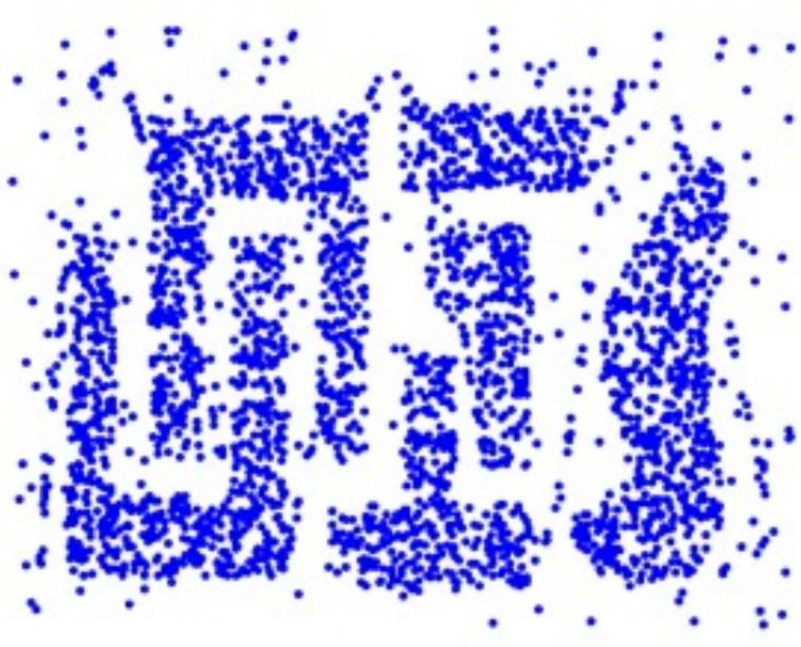


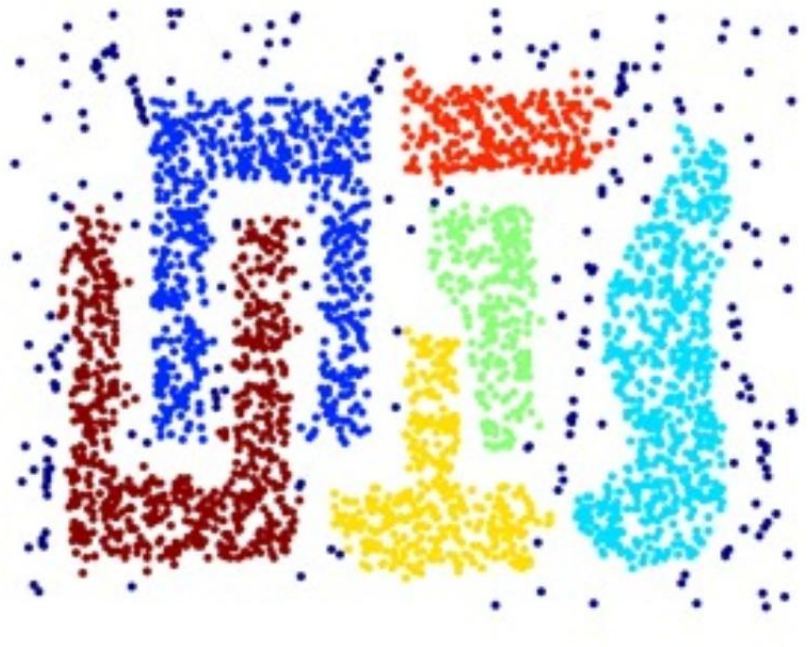
Density-Based Clustering



Core | Border | Noise

Density-Based Clustering





Create clusters by connecting core points

DBScan Algorithm

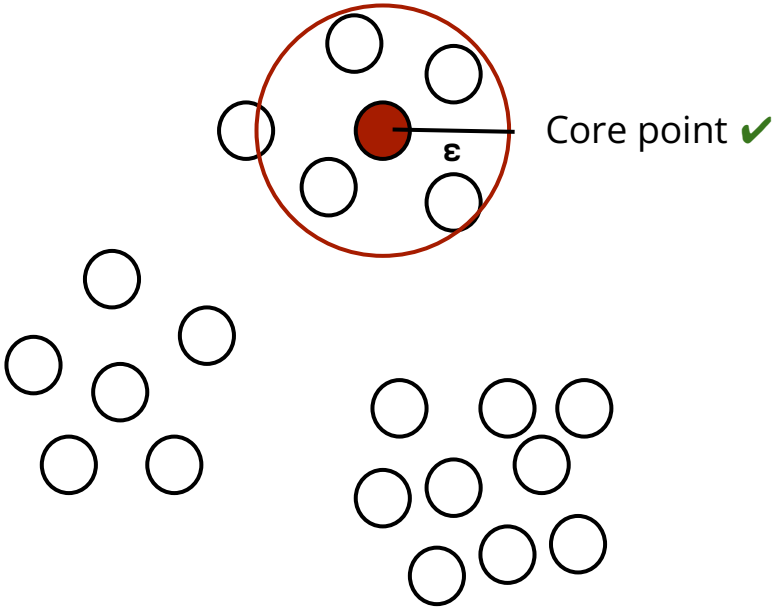
depth-first search algorithm

ε and **min_pts** given:

1. Find the **ε**-neighborhood of each point
2. Label the point as **core** if it contains at least **min_pts**
3. For each **core** point, assign to the same cluster all **core** points in its neighborhood (crux of the algorithm)
4. Label points in its neighborhood that are not **core** as **border**
5. Label points as **noise** if they are neither **core** nor **border**
6. Assign border points to nearby clusters

DBScan visualized

Min_pts = 3



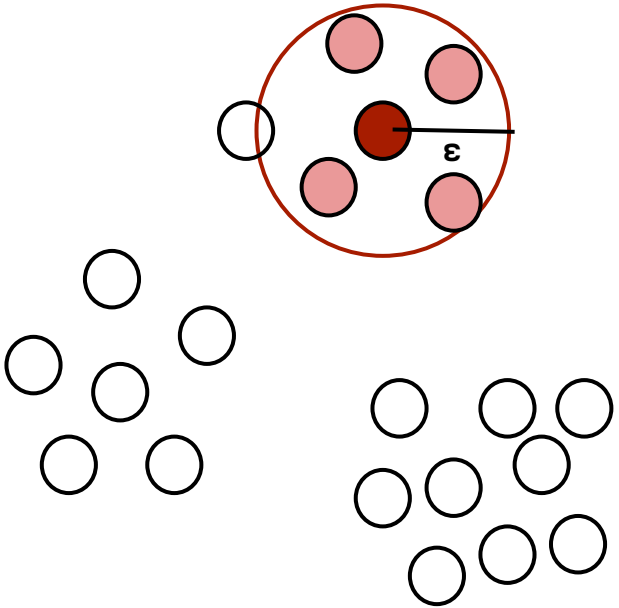
Iterate through the dataset

DBScan visualized

the light red points are part of that cluster - we will look through the neighbors of the light red to see if they should be added to the cluster

we only add those neighbors if those neighbors make the light red into a cluster-generating "core point"

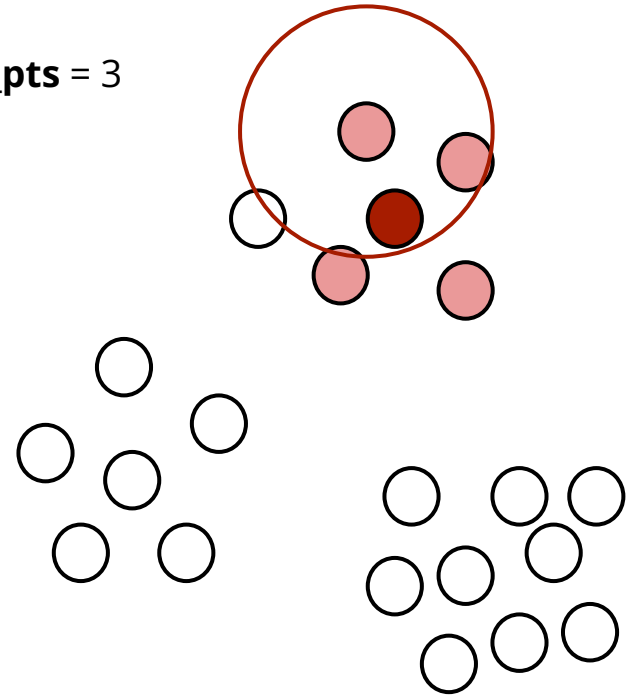
Min_pts = 3



If core point - iterate through its neighborhood to find more core points that should also be part of this cluster

DBScan visualized

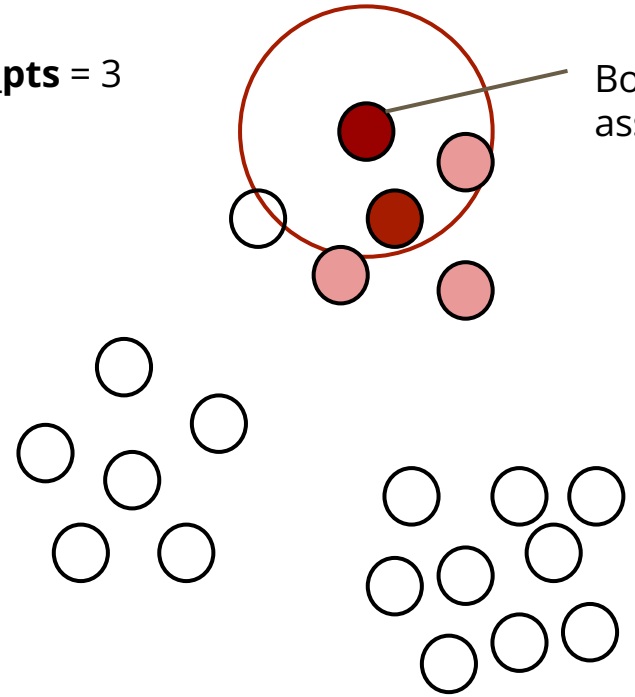
Min_pts = 3



If core point - iterate through its neighborhood to find more core points that should also be part of this cluster

DBScan visualized

Min_pts = 3

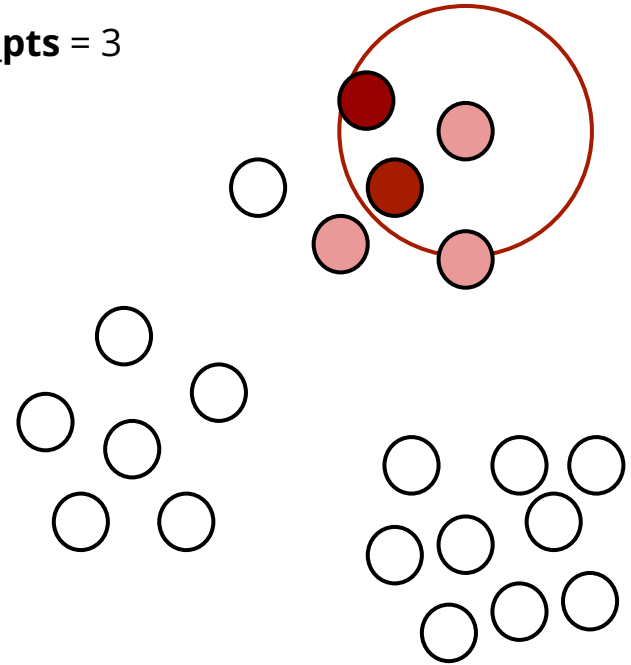


Border point but let's just assign it to this cluster

If core point - iterate through its neighborhood to find more core points that should also be part of this cluster

DBScan visualized

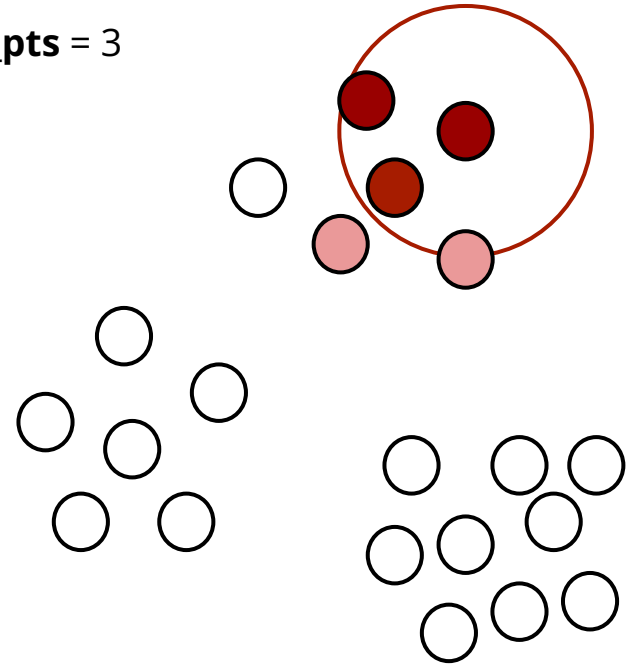
Min_pts = 3



If core point - iterate through its neighborhood to find more core points that should also be part of this cluster

DBScan visualized

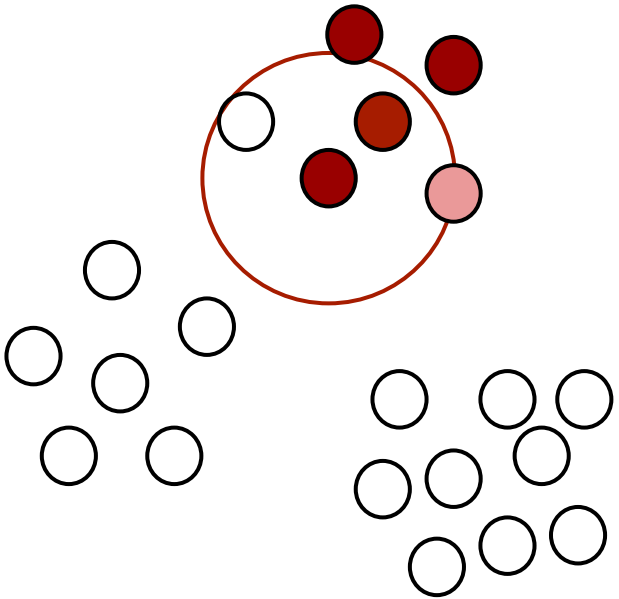
Min_pts = 3



If core point - iterate through its neighborhood to find more core points that should also be part of this cluster

DBScan visualized

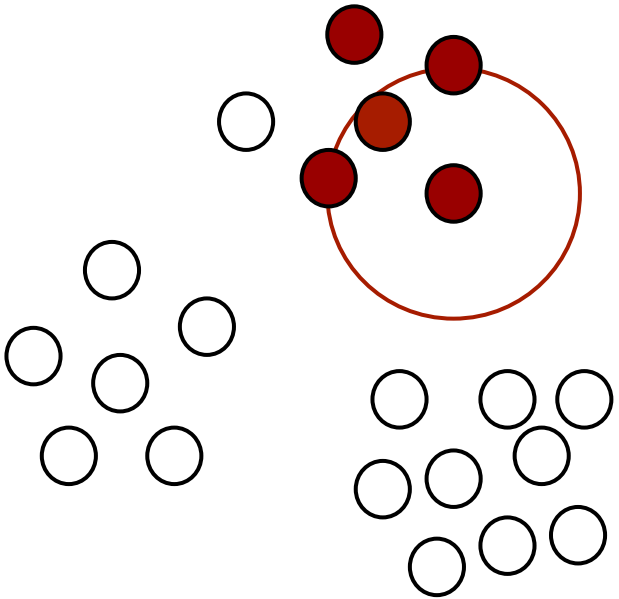
Min_pts = 3



If core point - iterate through its neighborhood to find more core points that should also be part of this cluster

DBScan visualized

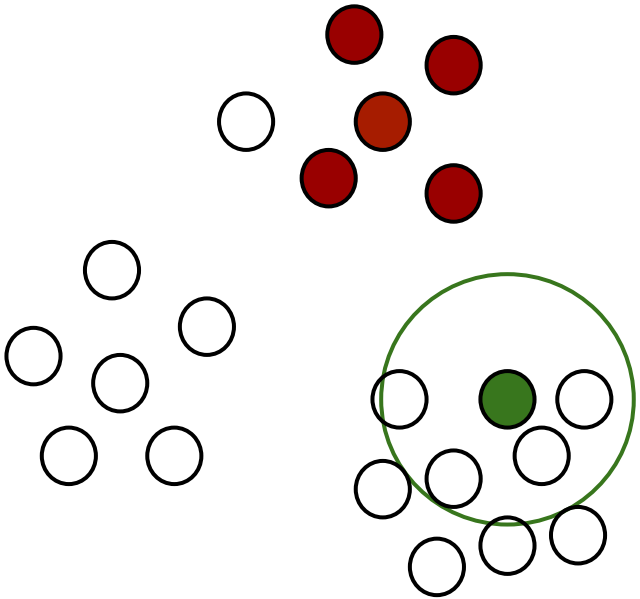
Min_pts = 3



If core point - iterate through its neighborhood to find more core points that should also be part of this cluster

DBScan visualized

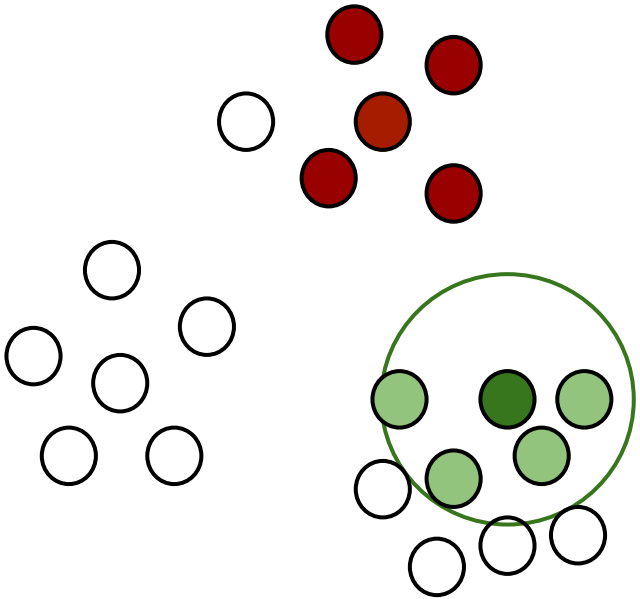
Min_pts = 3



Go to next data point in the dataset

DBScan visualized

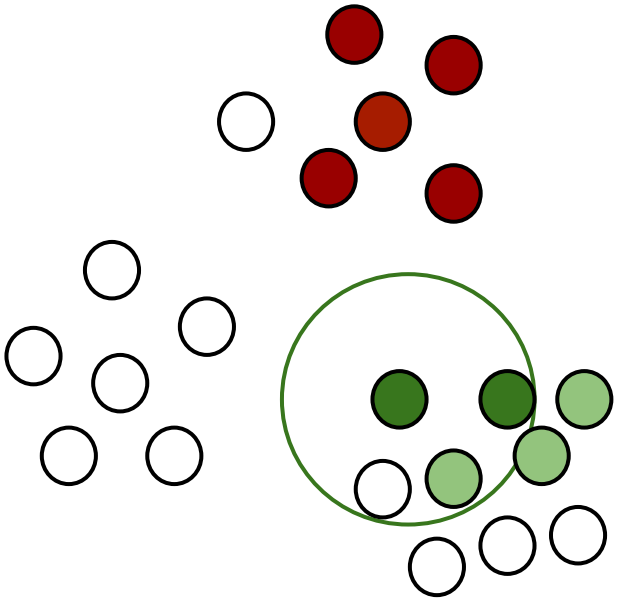
Min_pts = 3



Iterate over its neighborhood since it’s a core point

DBScan visualized

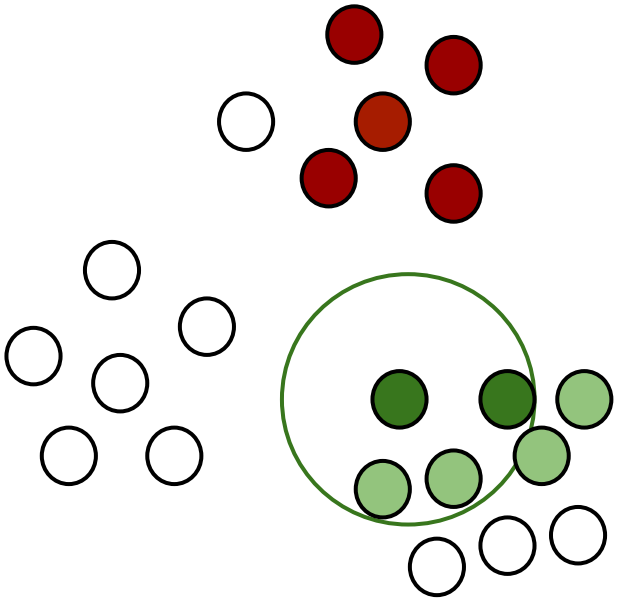
Min_pts = 3



Found another core point so we need to iterate over its neighborhood too

DBScan visualized

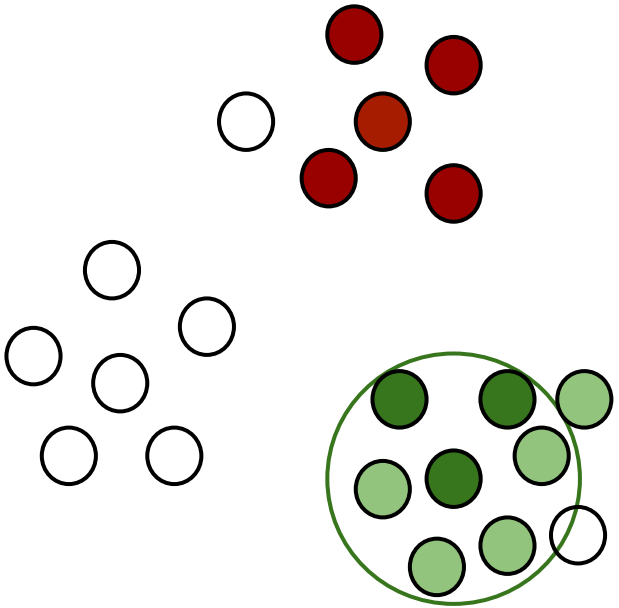
Min_pts = 3



Found another core point so we need to iterate over its neighborhood too

DBScan visualized

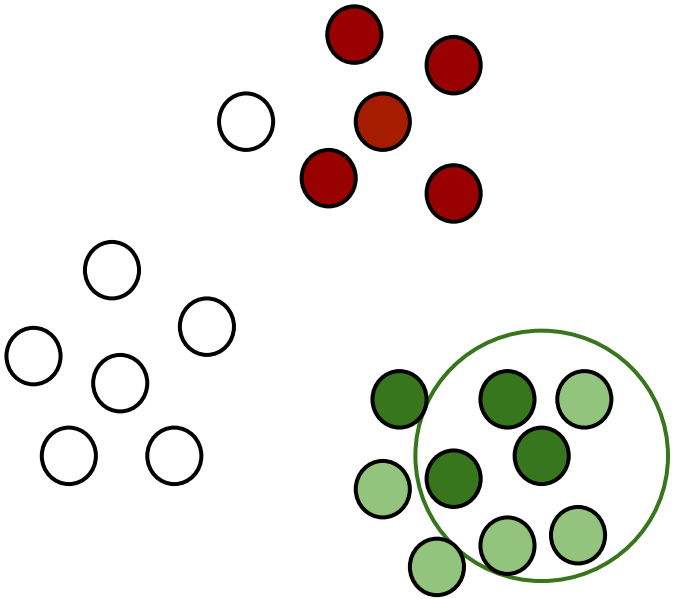
Min_pts = 3



Found another core point so we need to iterate over its neighborhood too

DBScan visualized

Min_pts = 3

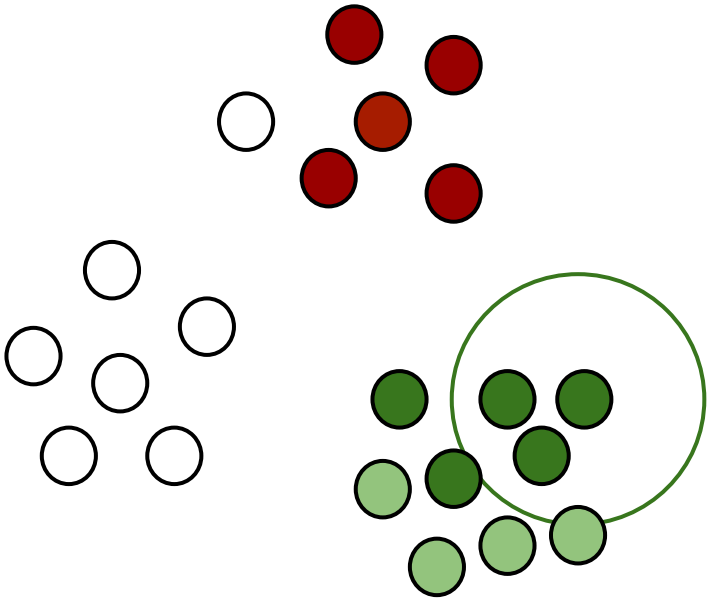


Found another core point so we need to iterate over its neighborhood too

core point is neighbor of another core points which has more core pt neighbors...

DBScan visualized

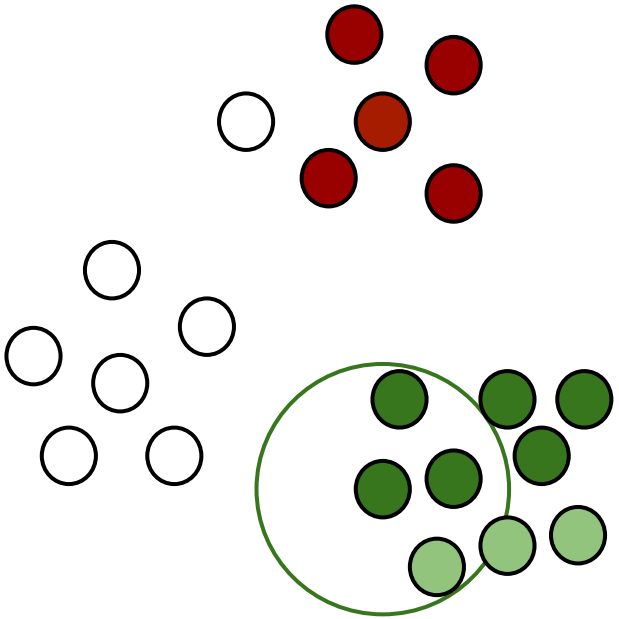
Min_pts = 3



Border point but let's assign it to the cluster now

DBScan visualized

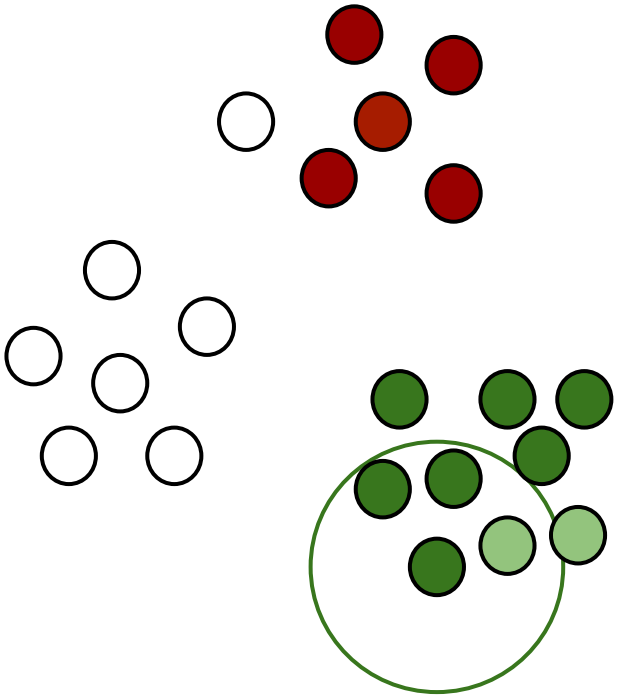
Min_pts = 3



Core point but all its neighborhood is already tracked

DBScan visualized

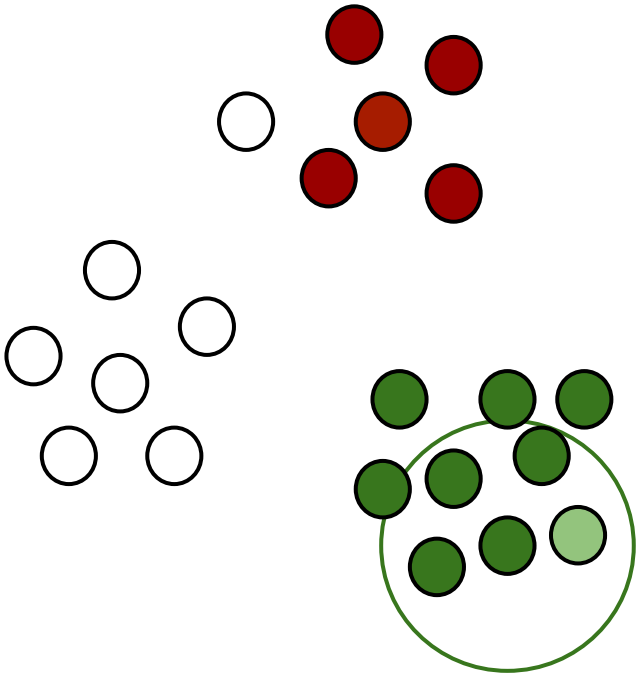
Min_pts = 3



Core point but all its neighborhood is already tracked

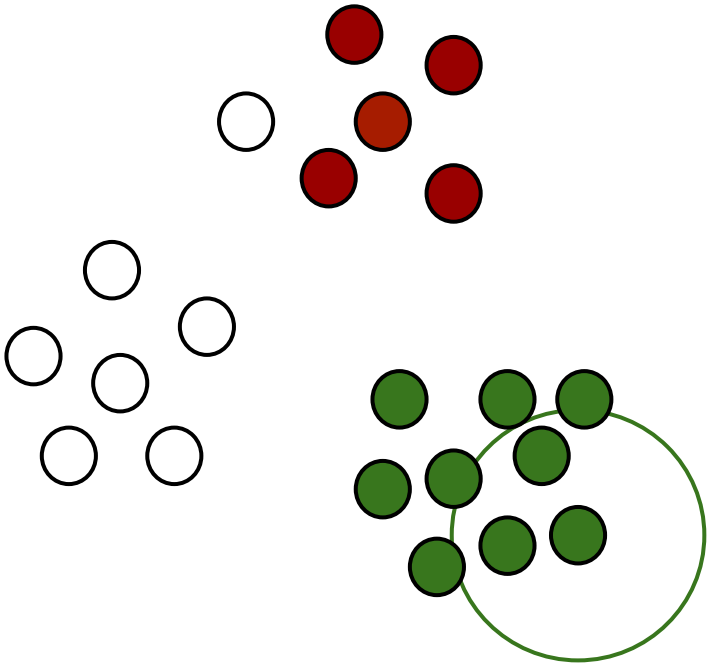
DBScan visualized

Min_pts = 3



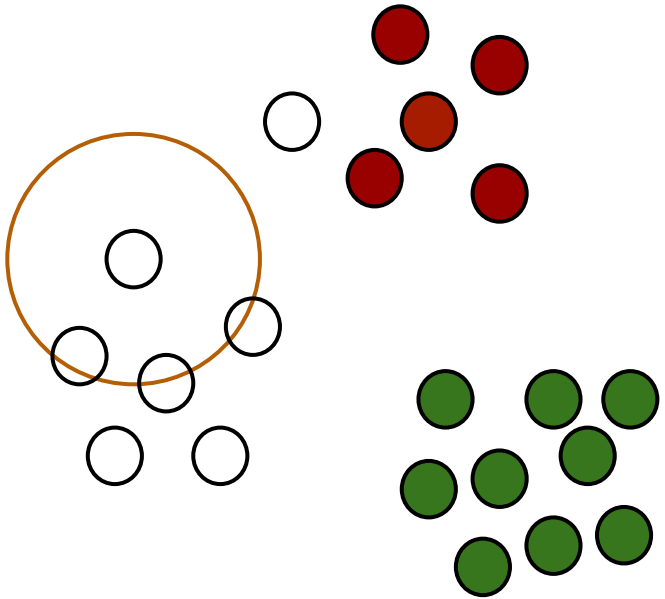
DBScan visualized

Min_pts = 3



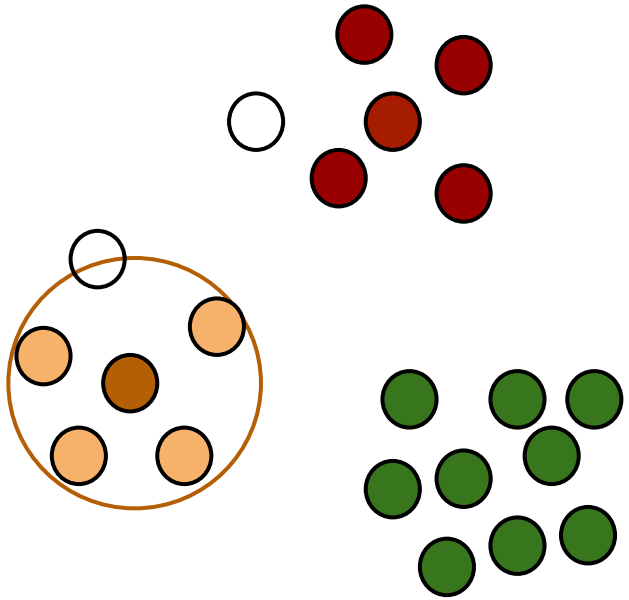
DBScan visualized

Min_pts = 3



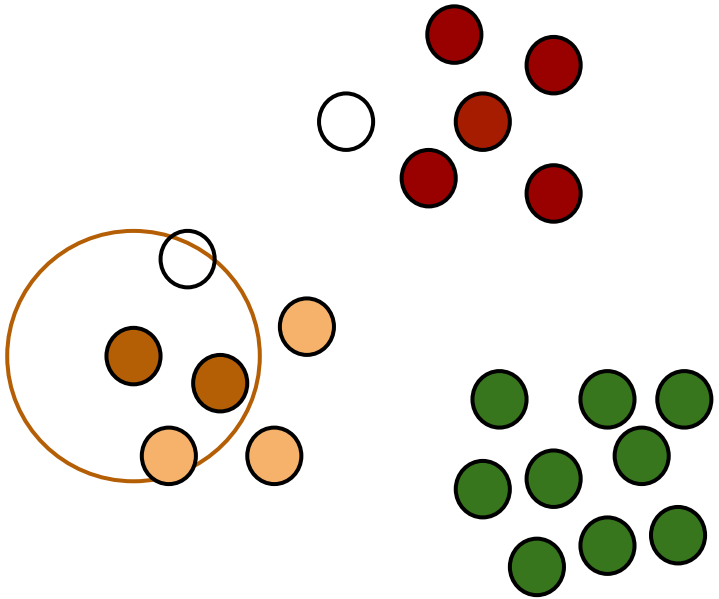
DBScan visualized

Min_pts = 3



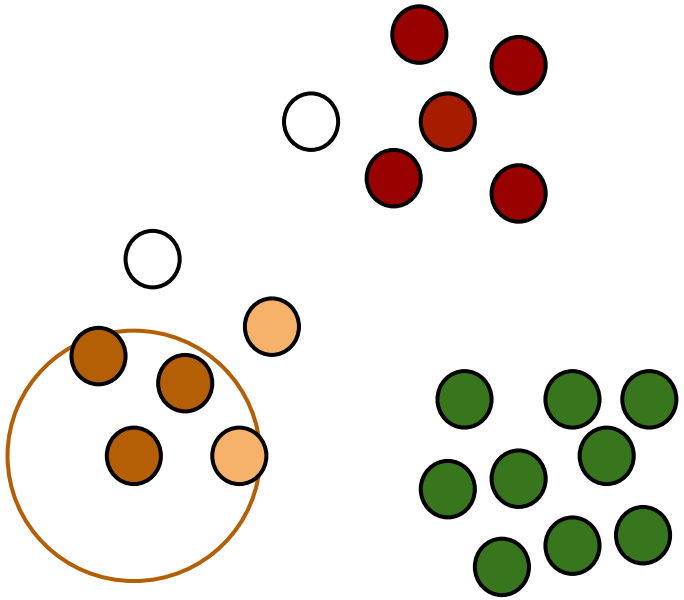
DBScan visualized

Min_pts = 3



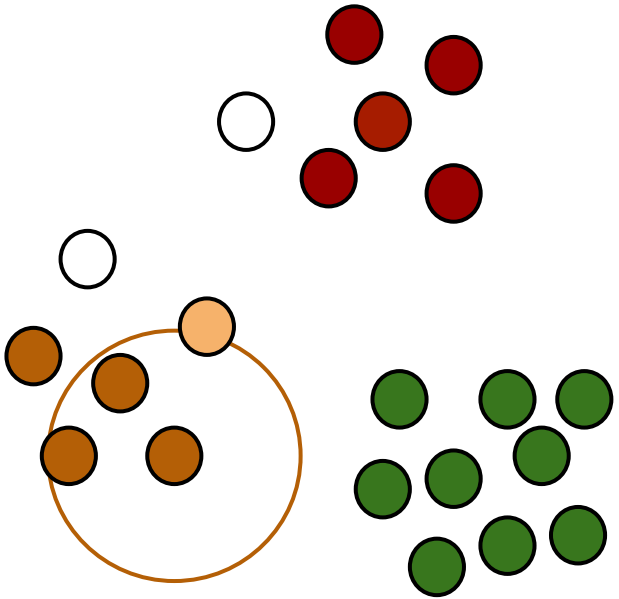
DBScan visualized

Min_pts = 3



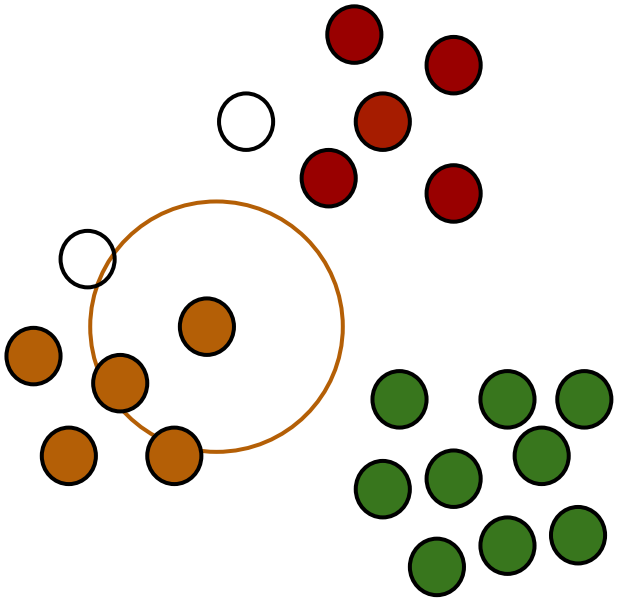
DBScan visualized

Min_pts = 3



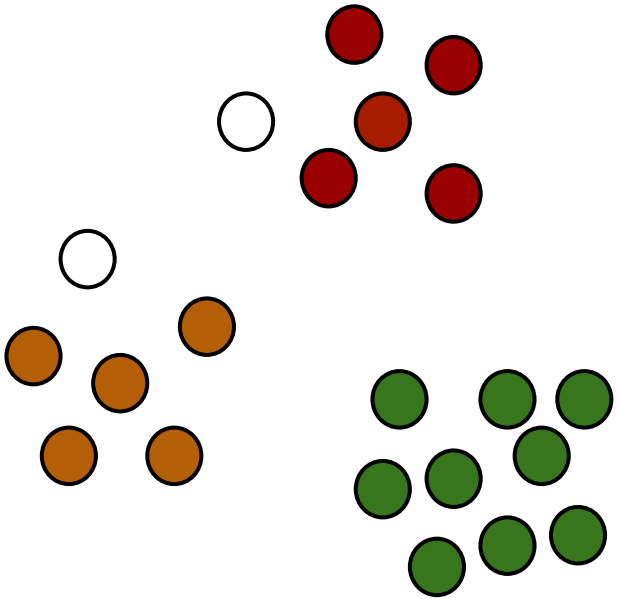
DBScan visualized

Min_pts = 3



DBScan visualized

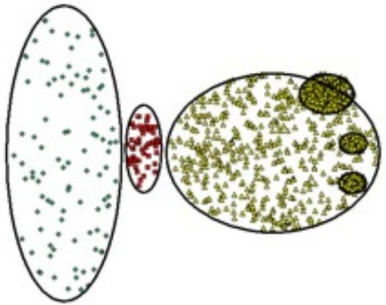
Min_pts = 3



DBScan - Benefits

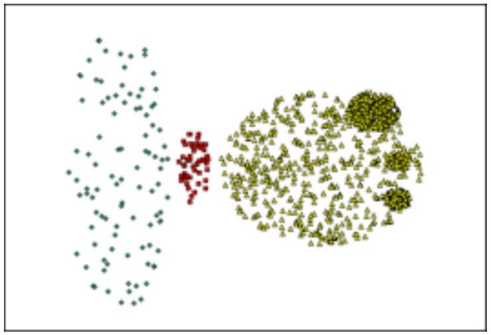
1. Can identify clusters of different shapes and sizes
2. Resistant to noise

DBScan - Limitations

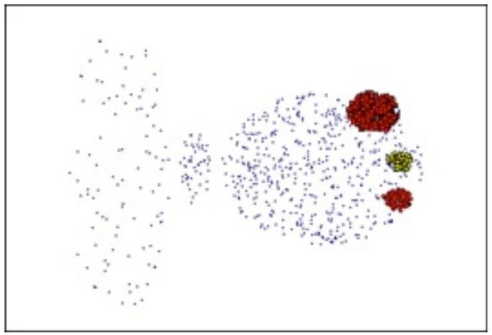


1. Can fail to identify clusters of varying densities.
2. Tends to create clusters of the same density.
3. Notion of density is problematic in high-dimensional spaces

Misses the less dense cluster on the right



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

Demo

Attribute A	Attribute B	Attribute C	Attribute D	Assignment
Yes	Single	125k	No	0
No	Married	100k	No	1
No	Single	70k	No	0
Yes	Married	120k	No	2
No	Divorced	90k	Yes	1
No	Married	60k	No	1
Yes	Divorced	220k	No	3
No	Single	85k	Yes	0
No	Married	75k	No	1
No	Single	90k	Yes	3