Master Degree in Artificial Intelligence for Science and Technology

Cluster Analysis: Density-Based Clustering

Fabio Stella
Department of Informatics, Systems and Communication
University of Milan-Bicocca
fabio.stella@unimib.it

OUTLOOK

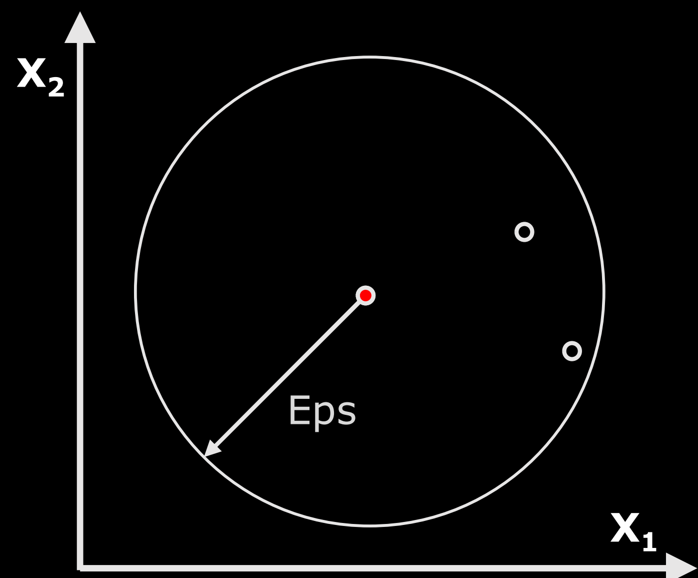
- Concept
- DBSCAN
  - Core point
  - Border point
  - Noise point
- Advantages
- Limitations
- DBSCAN vs K-means
- Additional algorithms

DENSITY-BASED CLUSTERING TECHNIQUES

- Density-based clustering techniques aim to find dense regions of objects that are surrounded by low-density regions.
- **DBSCAN** is a simple and effective density-based clustering algorithm that illustrates a number of important concepts that are typical of the density-based approach.
- Several methods exist to define density, we describe the center-based approach on which DBSCAN is based.

density of the red filled circle object is the number of objects within a specific radius (Eps) of that object

3



density of any point will depend on the specified radius (Eps).

DBSCAN

- Density-based Clustering Techniques allow to classify a point (record) as being:
  - **CORE POINT**
  - **BORDER POINT**
  - **NOISE POINT**

DBSCAN

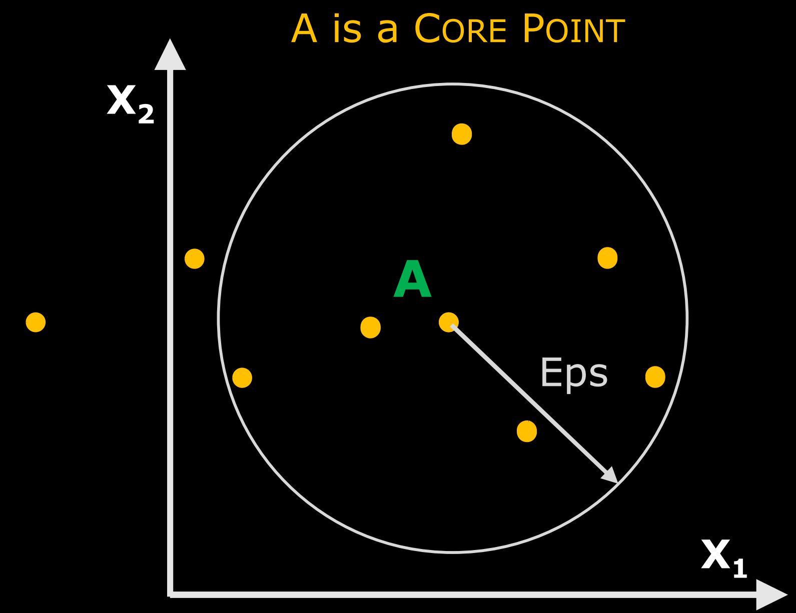
- Density-based Clustering Techniques allow to classify a point (record) as being:

  — **CORE POINT :** is in the interior of a density-based cluster.

  A point is a core point if the number of points within a given neighborhood around the point as determined by the distance function and a user-specified distance parameter, **Eps**, exceeds a certain threshold, **MinPts**, which is also a user-specified parameter.

We set **MinPts=6** and use

the selected **Eps** value



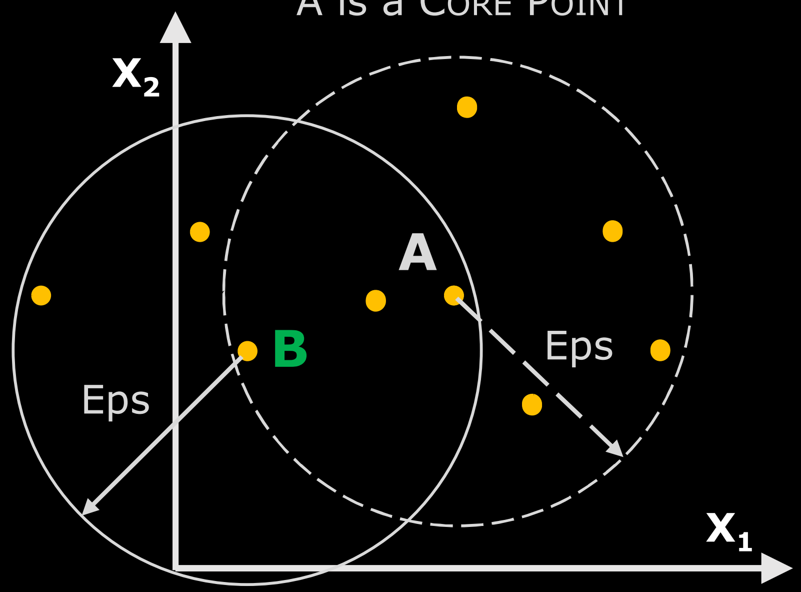
DBSCAN

- Density-based Clustering Techniques allow to classify a point (record) as being:
  - **BORDER POINT :** is not a core point, but falls within the neighborhood of a core point. A border point can fall in the neighborhood of several core points.

We set **MinPts=6** and use the selected **Eps** value

B is a Border Point

A is a Core Point



B is not a core point

B is a border point

DBSCAN

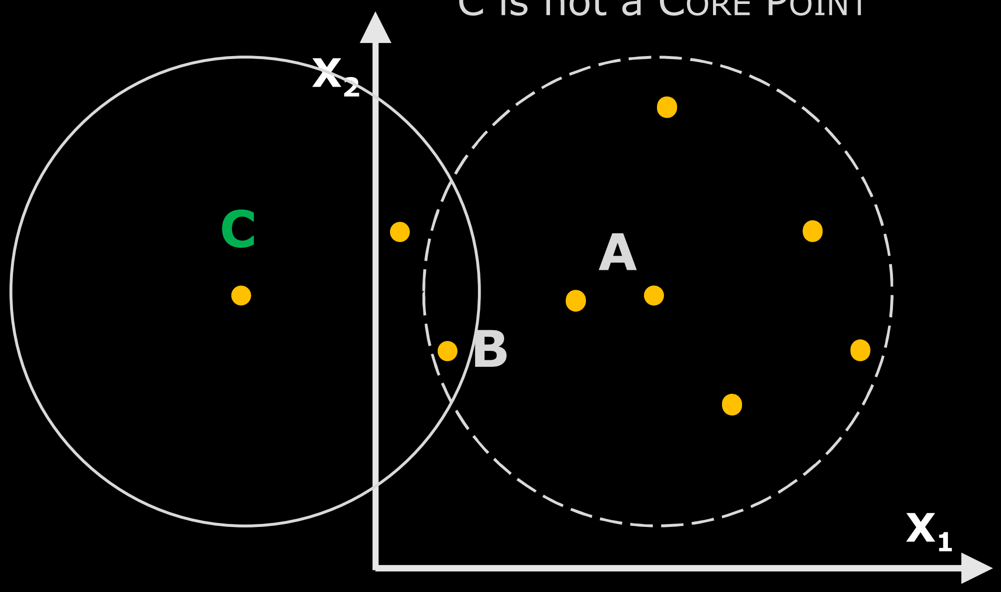
- Density-based Clustering Techniques allow to classify a point (record) as being:
  - **NOISE POINT :** is any point that is neither a core nor a border point

C is a NOISE POINT

C is not a BORDER POINT

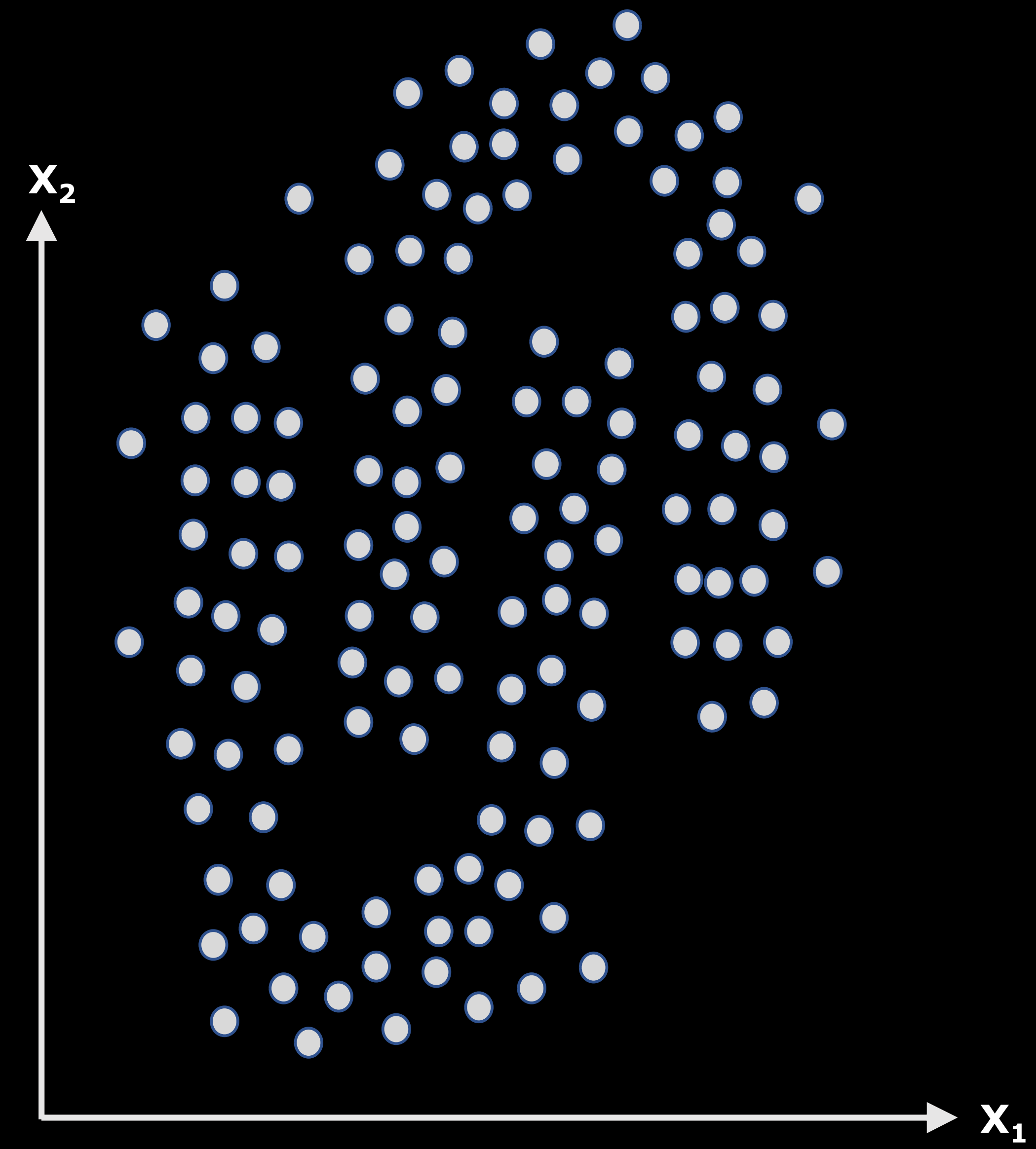
C is not a CORE POINT

We set **MinPts=6** and use the selected **Eps** value



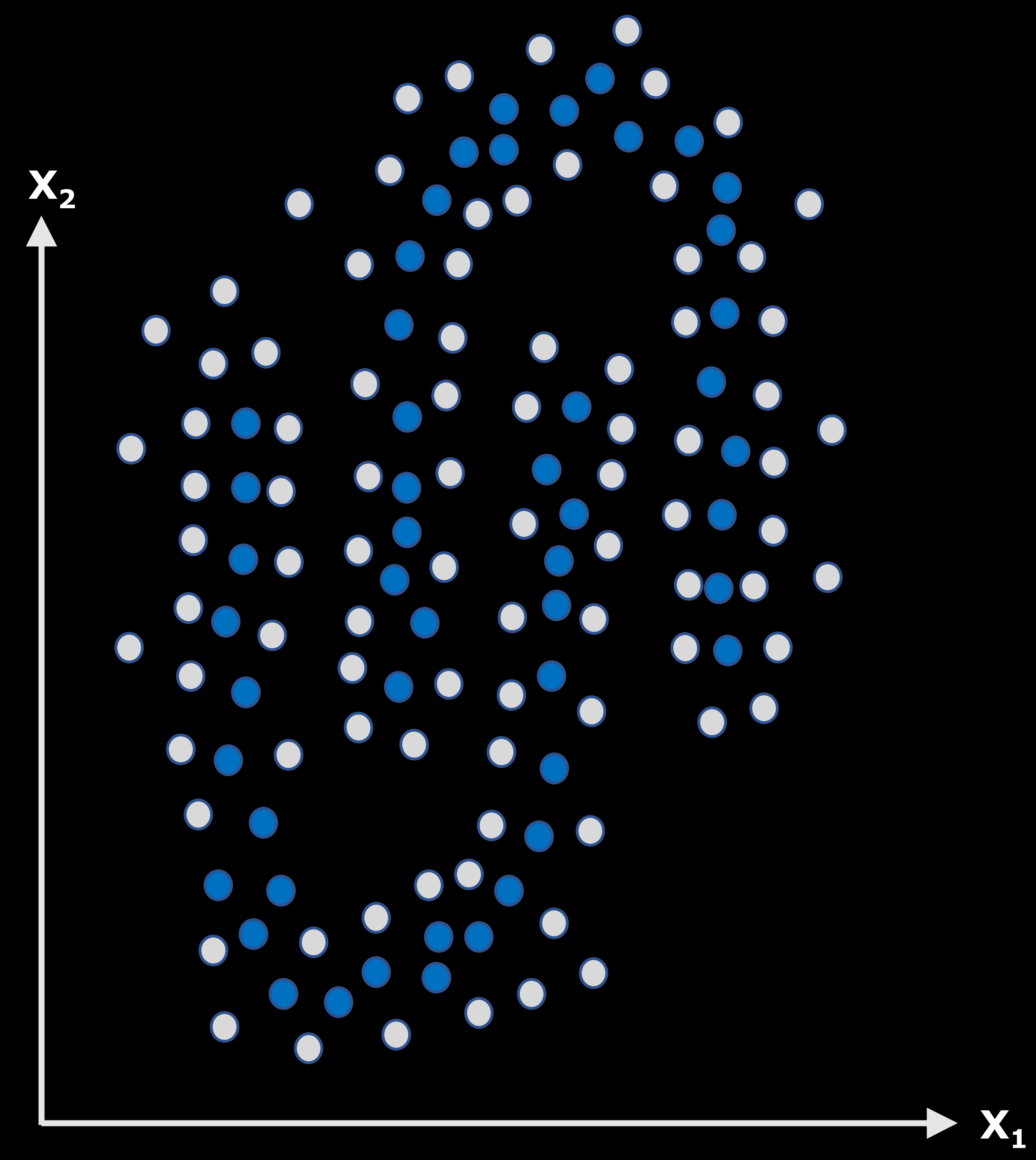
DBSCAN

1. Label all points as core, border, or noise points
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points



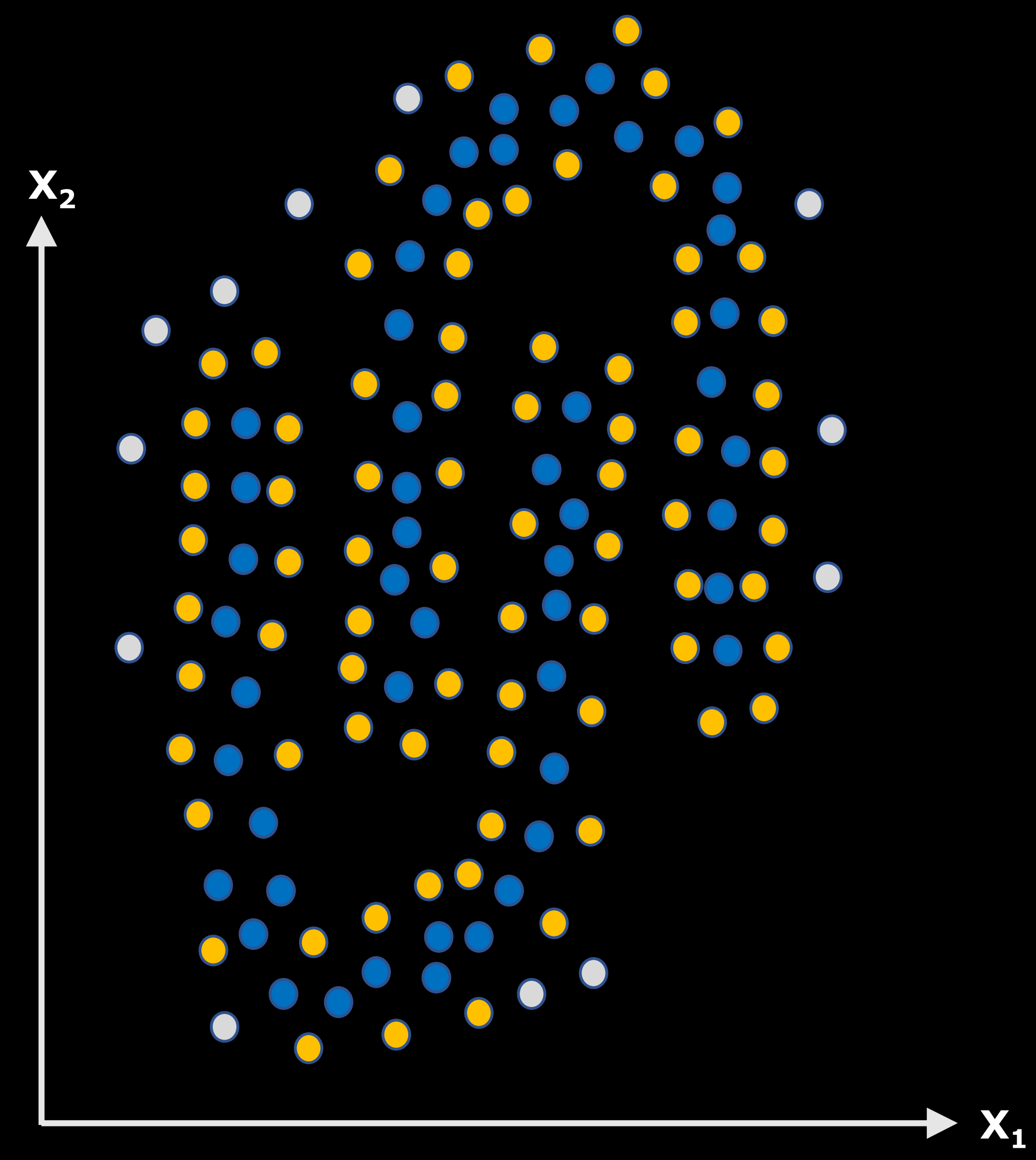
DBSCAN

1. Label all points as core point
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points



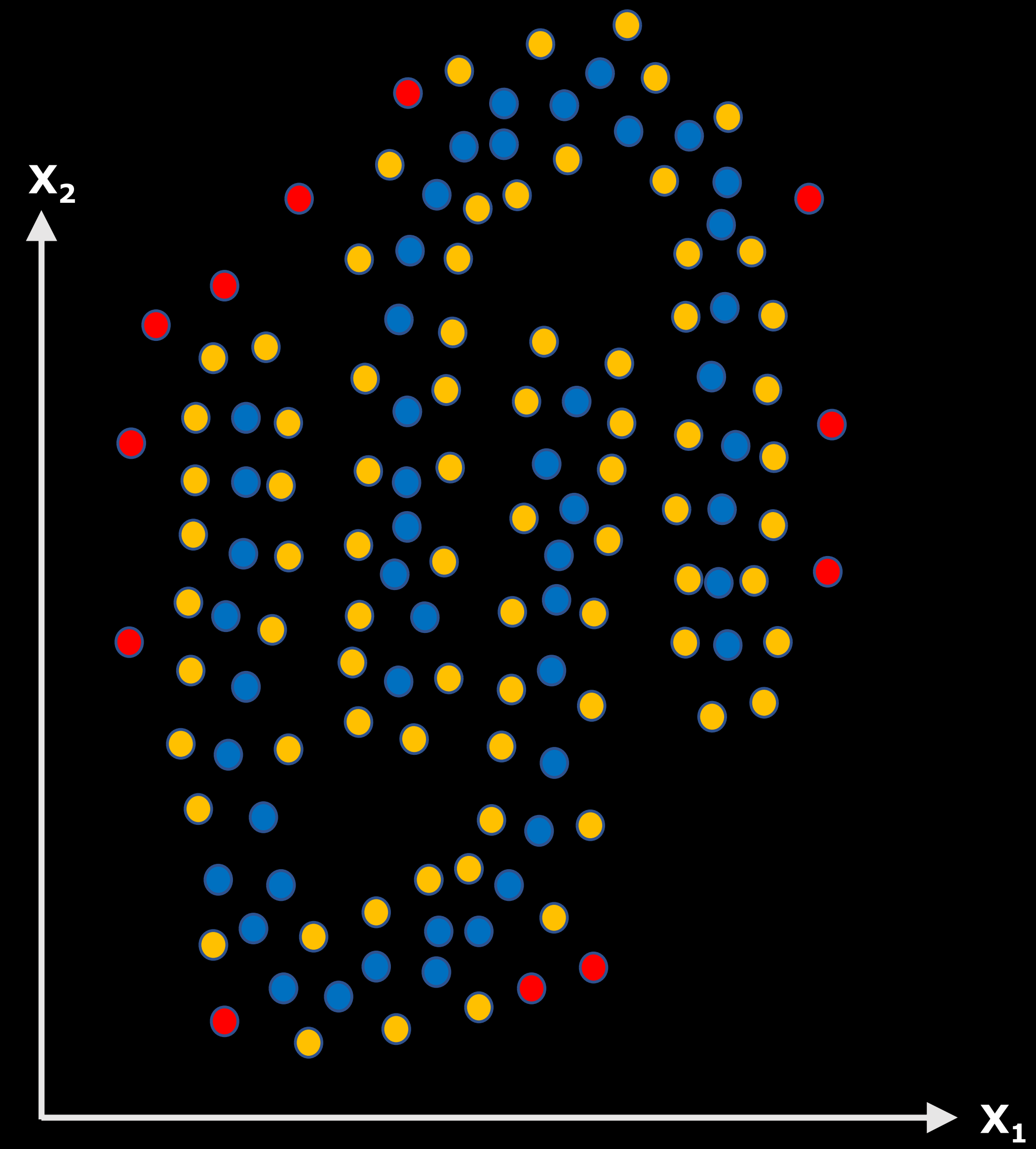
DBSCAN

1. Label all points as border point
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points



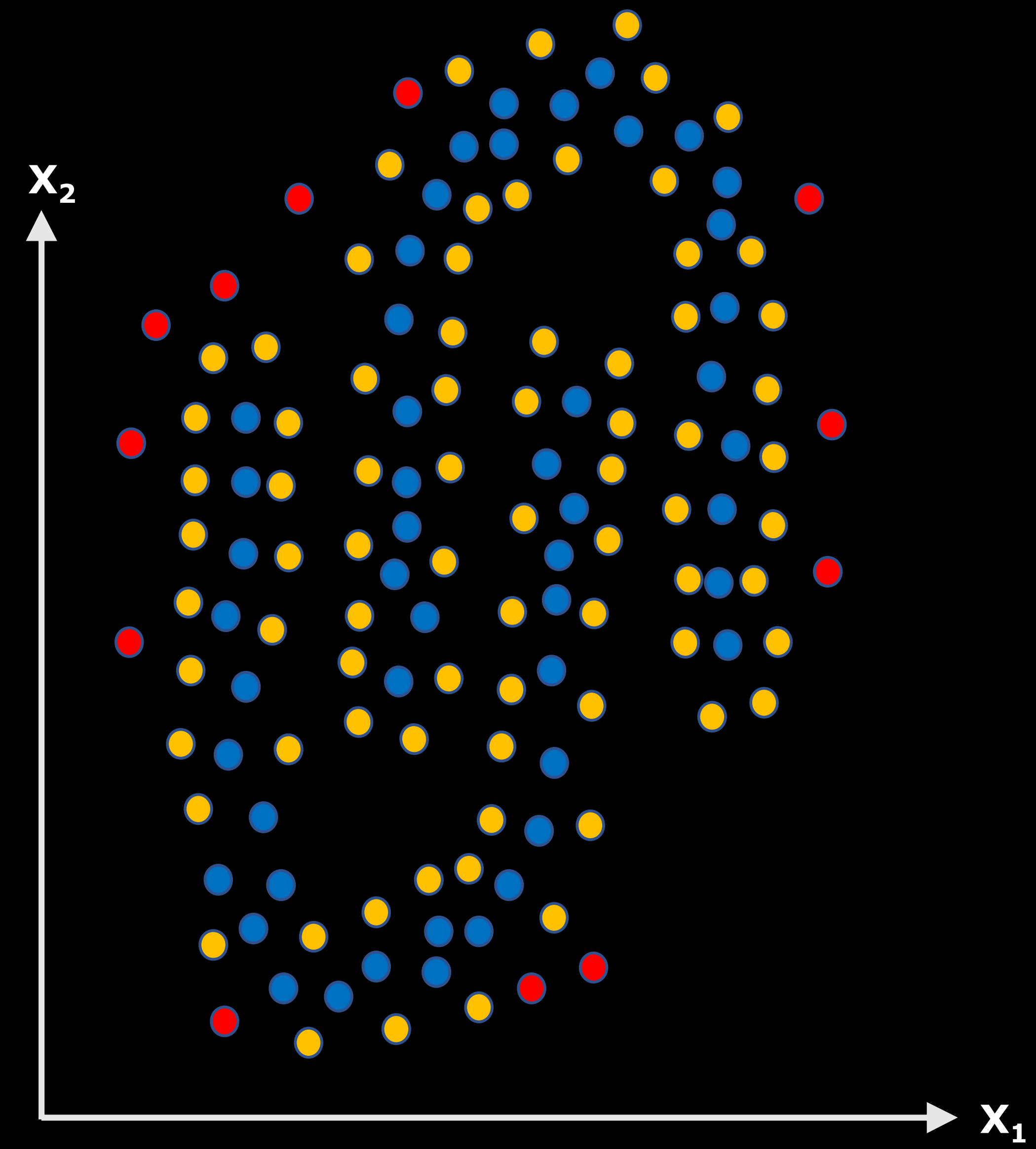
DBSCAN

1. Label all points as noise point
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points



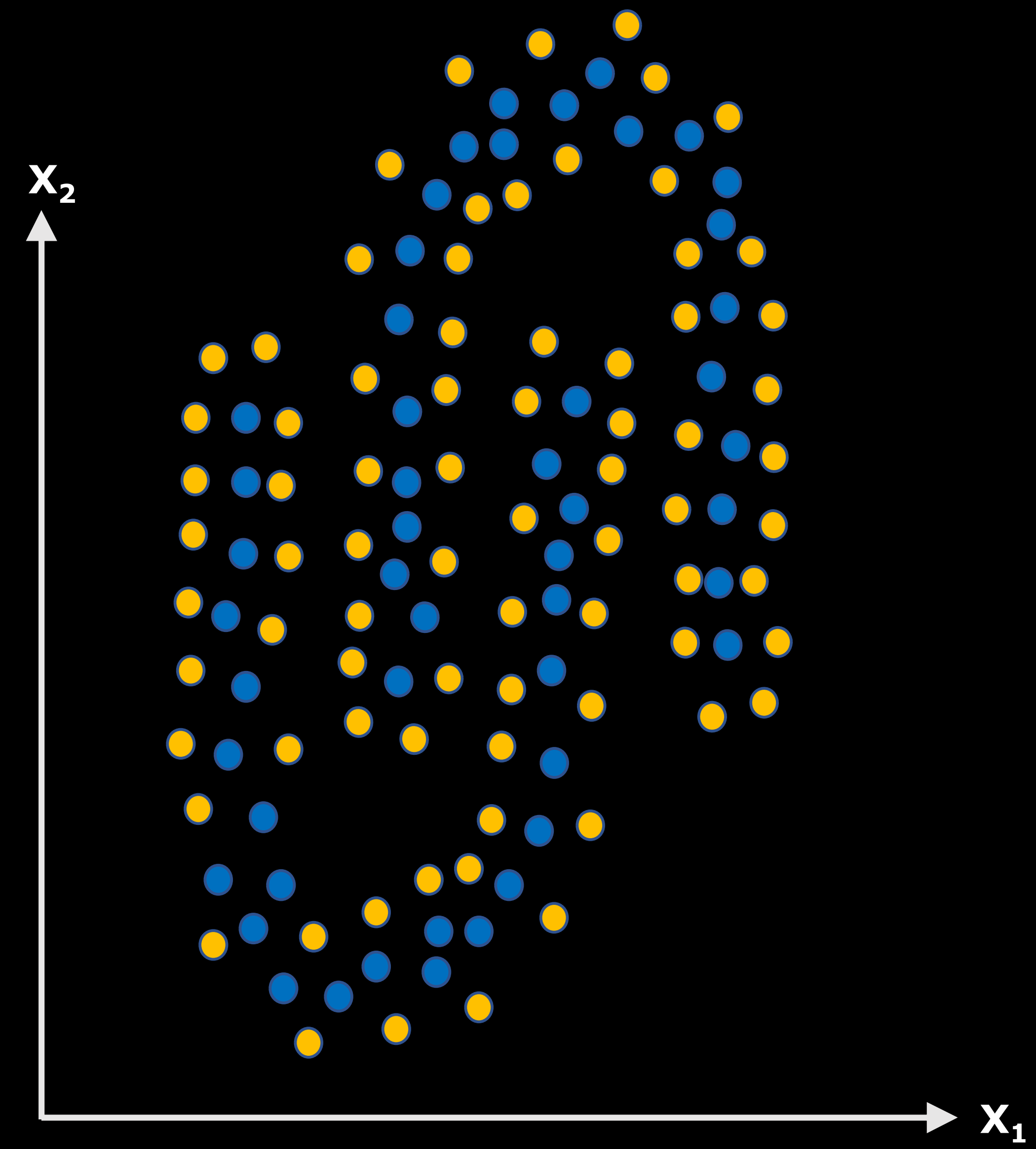
DBSCAN

1. Label all points as core, border, or noise points
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points



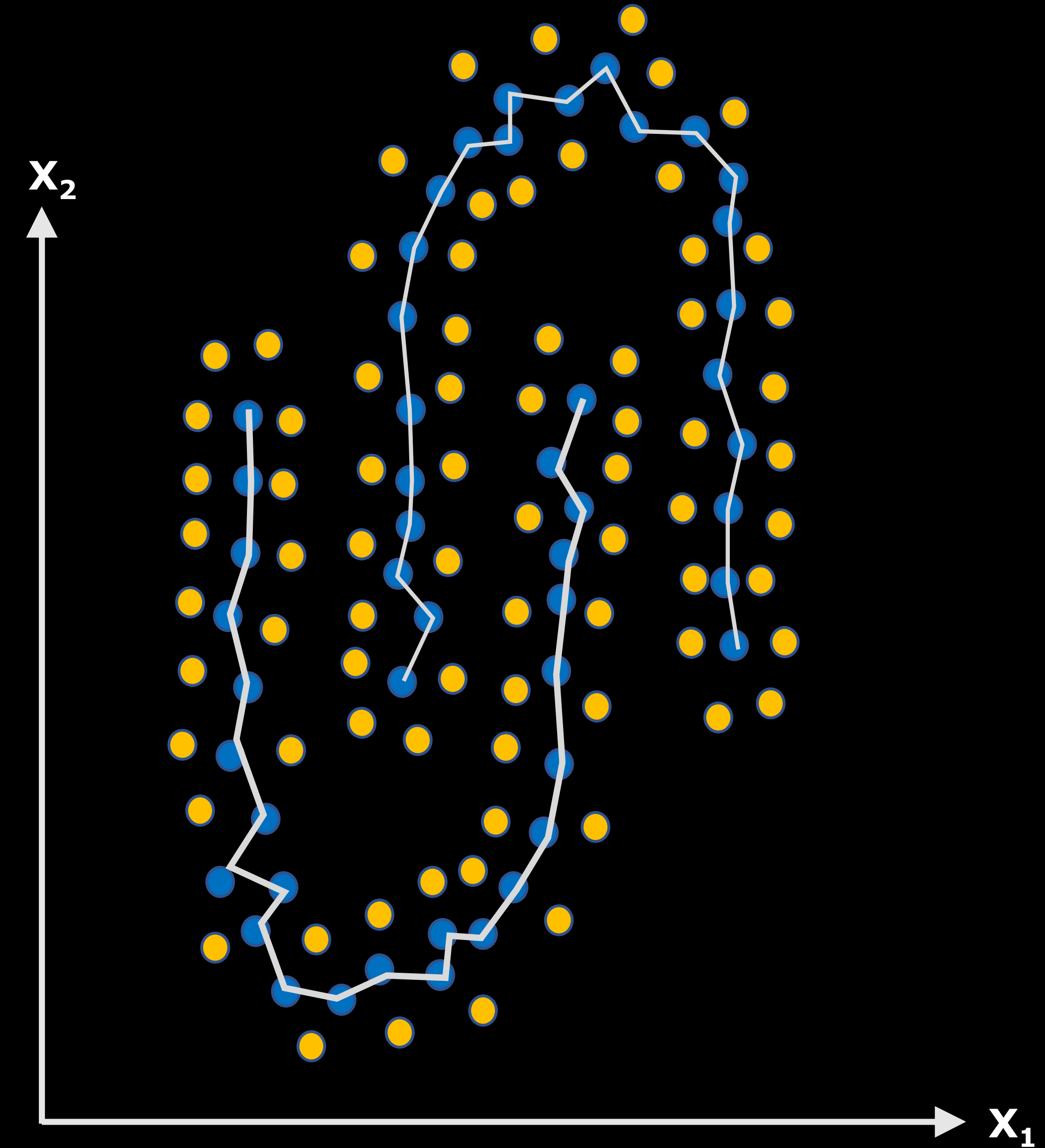
DBSCAN

1. Label all points as core, border, or noise points
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points



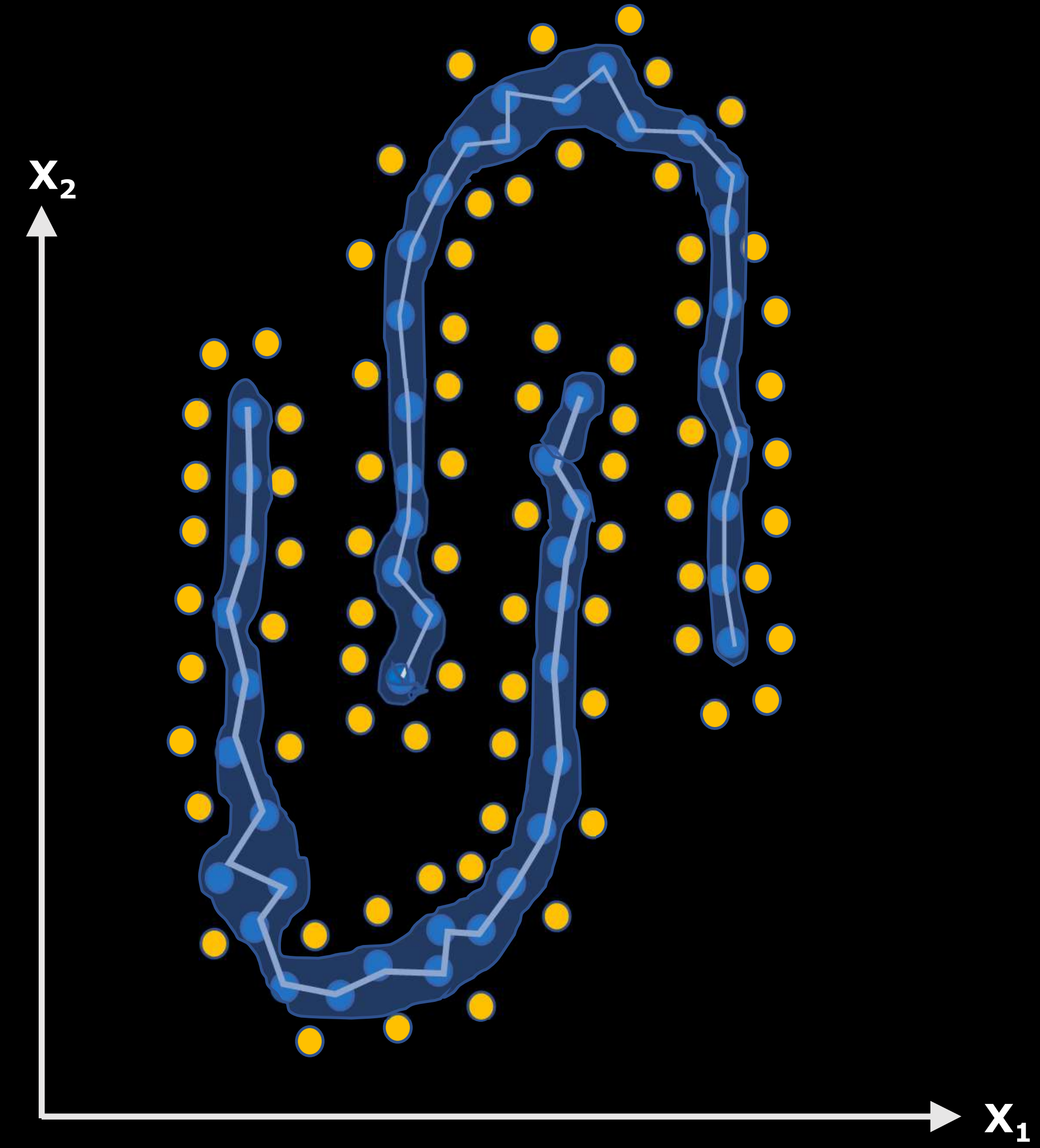
DBSCAN

1. Label all points as core, border, or noise points
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points



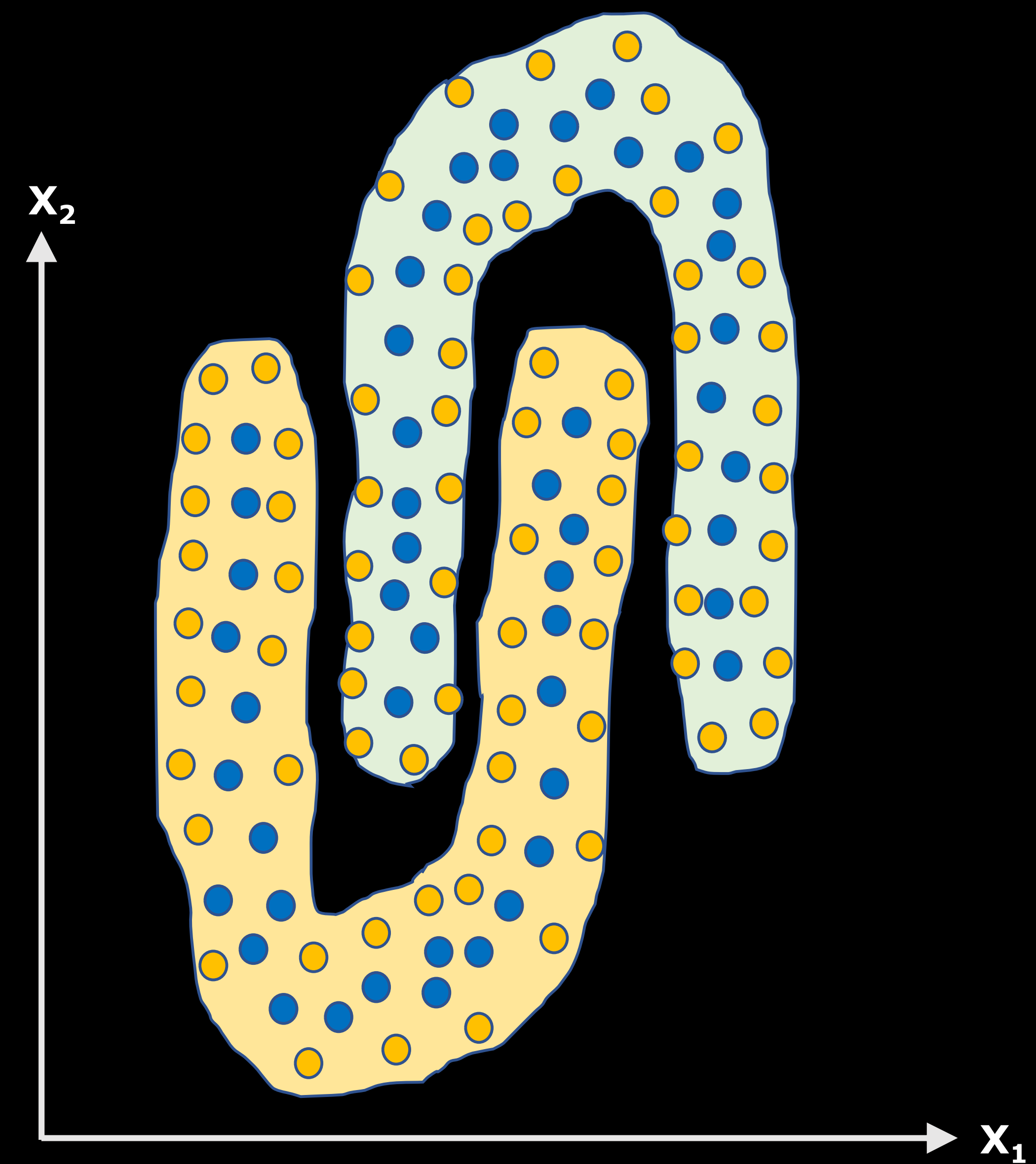
DBSCAN

1. Label all points as core, border, or noise points
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points

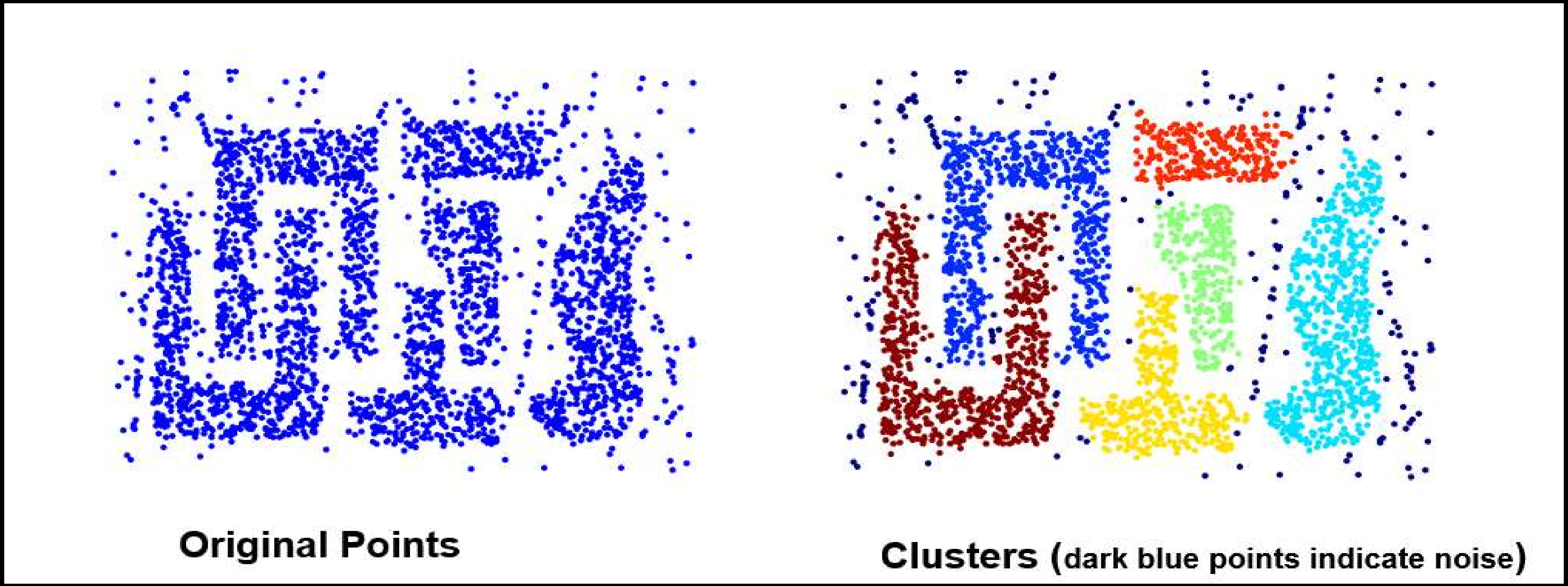


DBSCAN

1. Label all points as core, border, or noise points
2. Eliminate noise points
3. Put an edge between all core points that are within *Eps* of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to one of the clusters of its associated core points

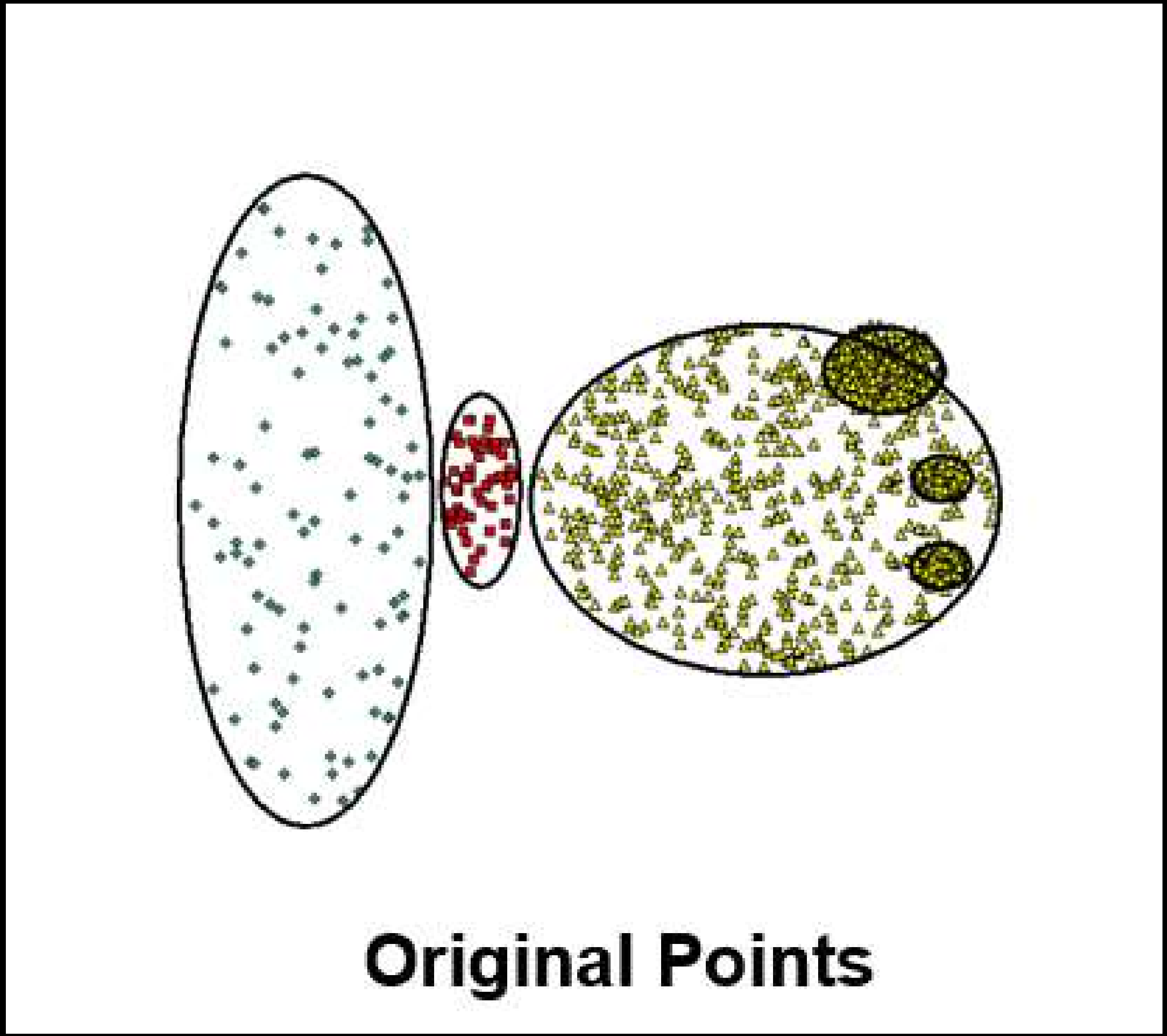


DBSCAN WORKS WELL WHEN …

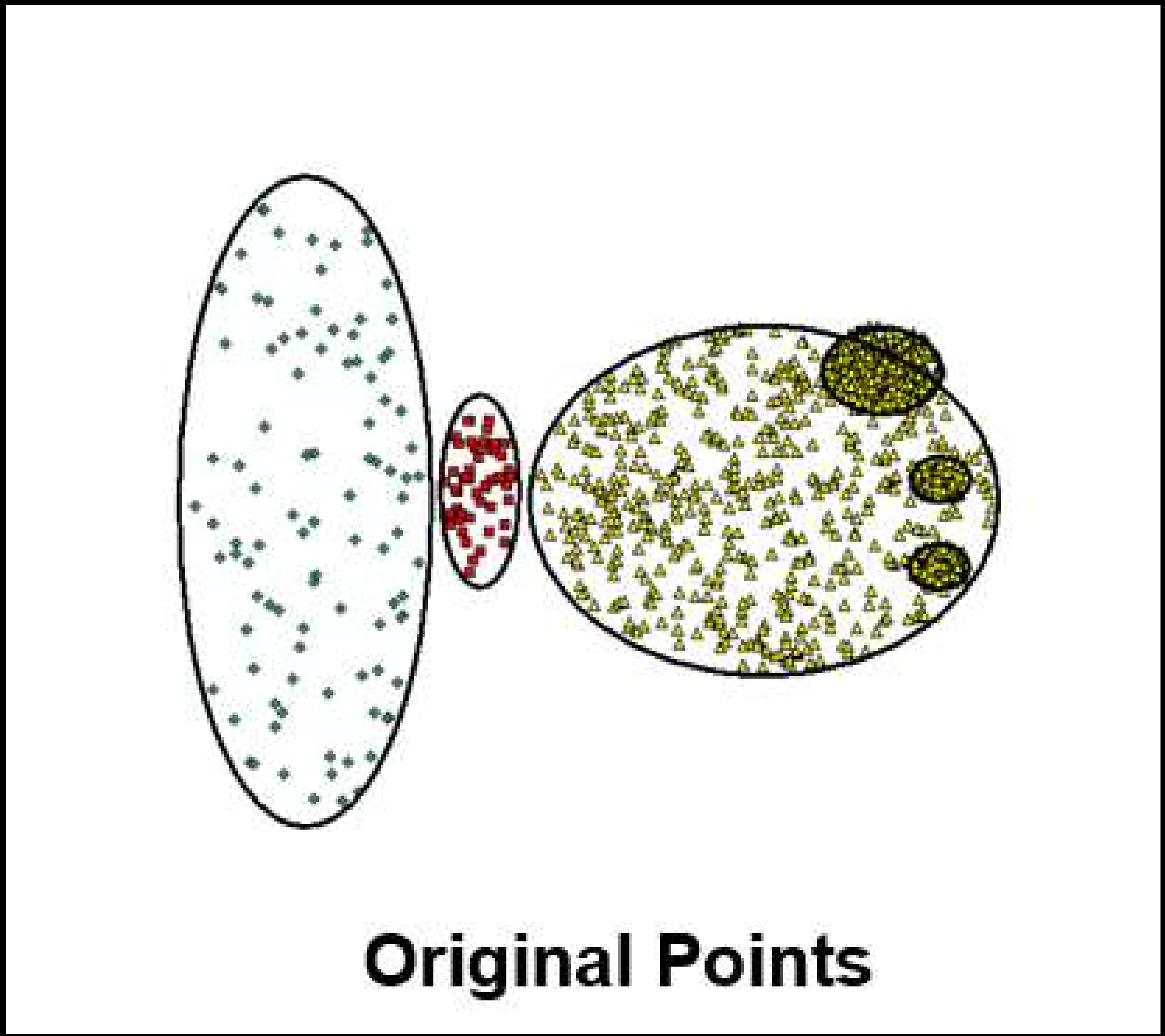


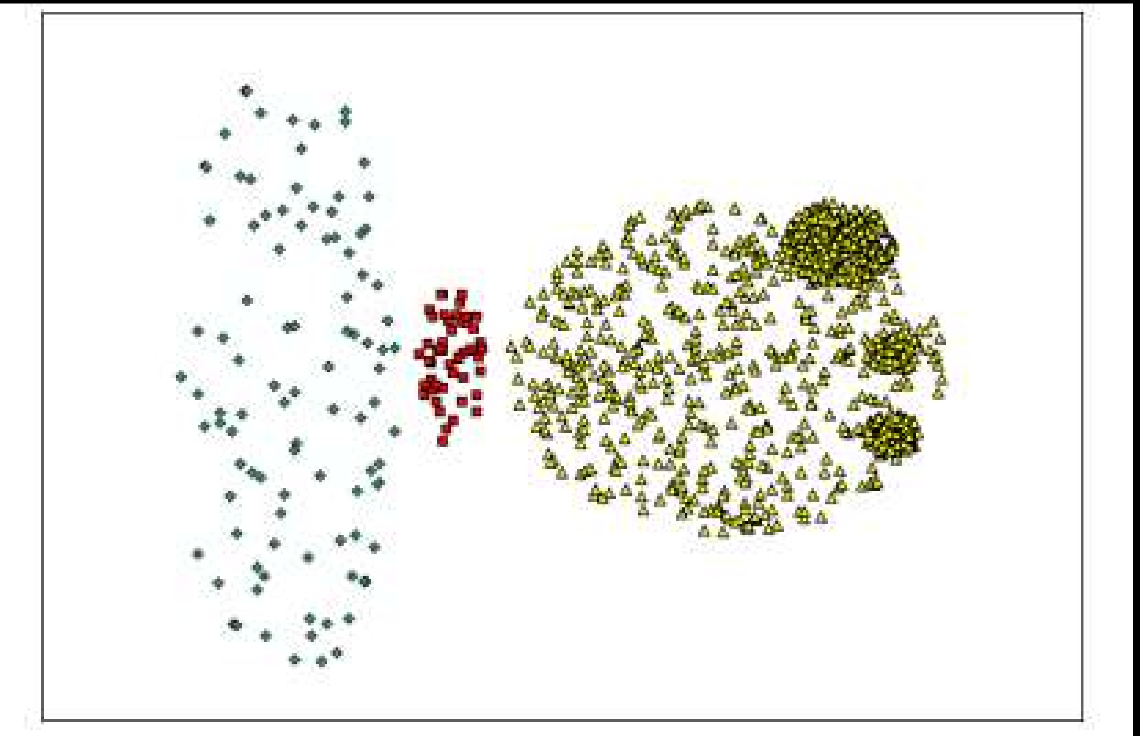
- Can handle clusters of different shapes and sizes
- Resistant to noise

DBSCAN DOES NOT WORK WELL WHEN …



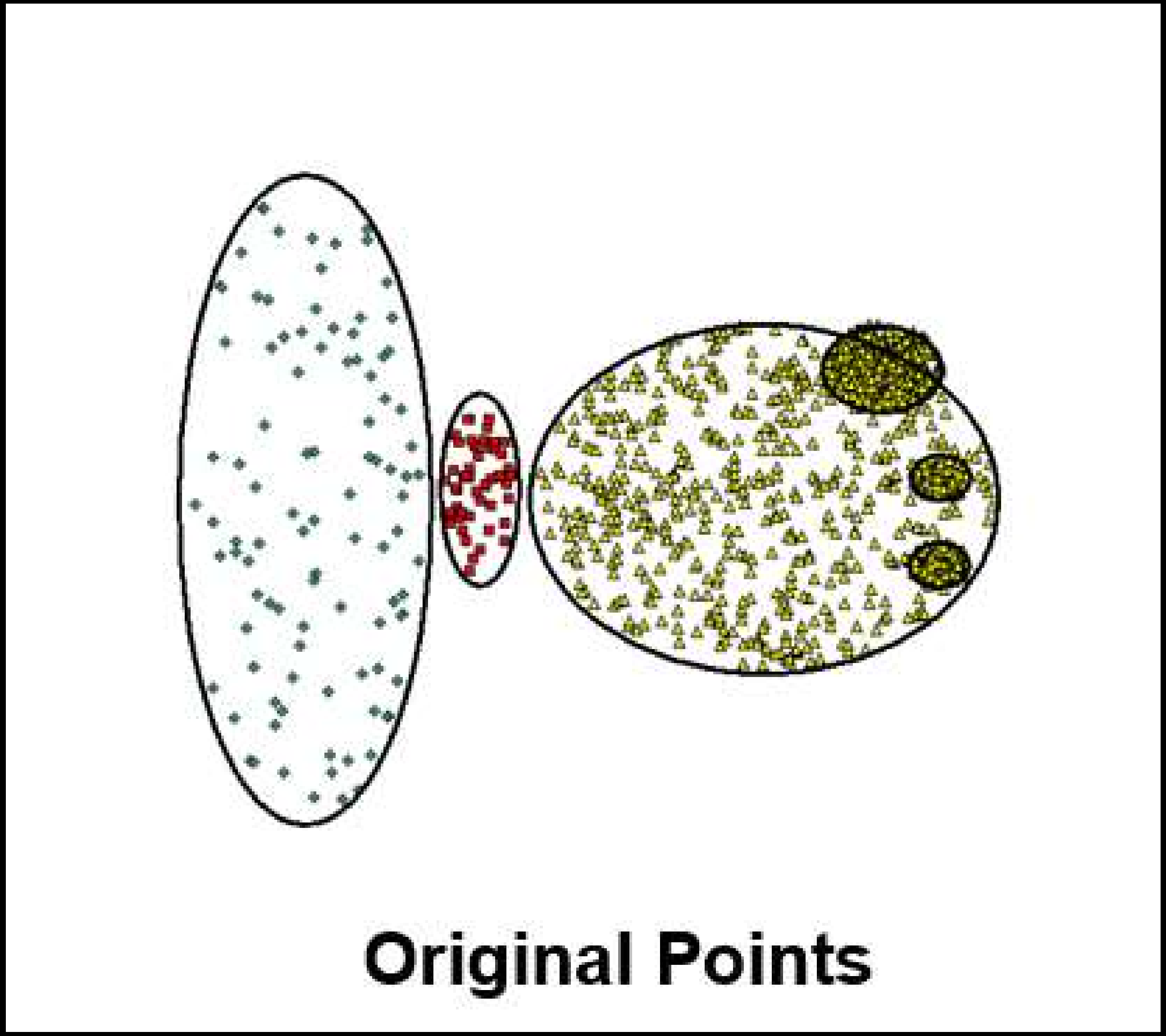
DBSCAN DOES NOT WORK WELL WHEN …

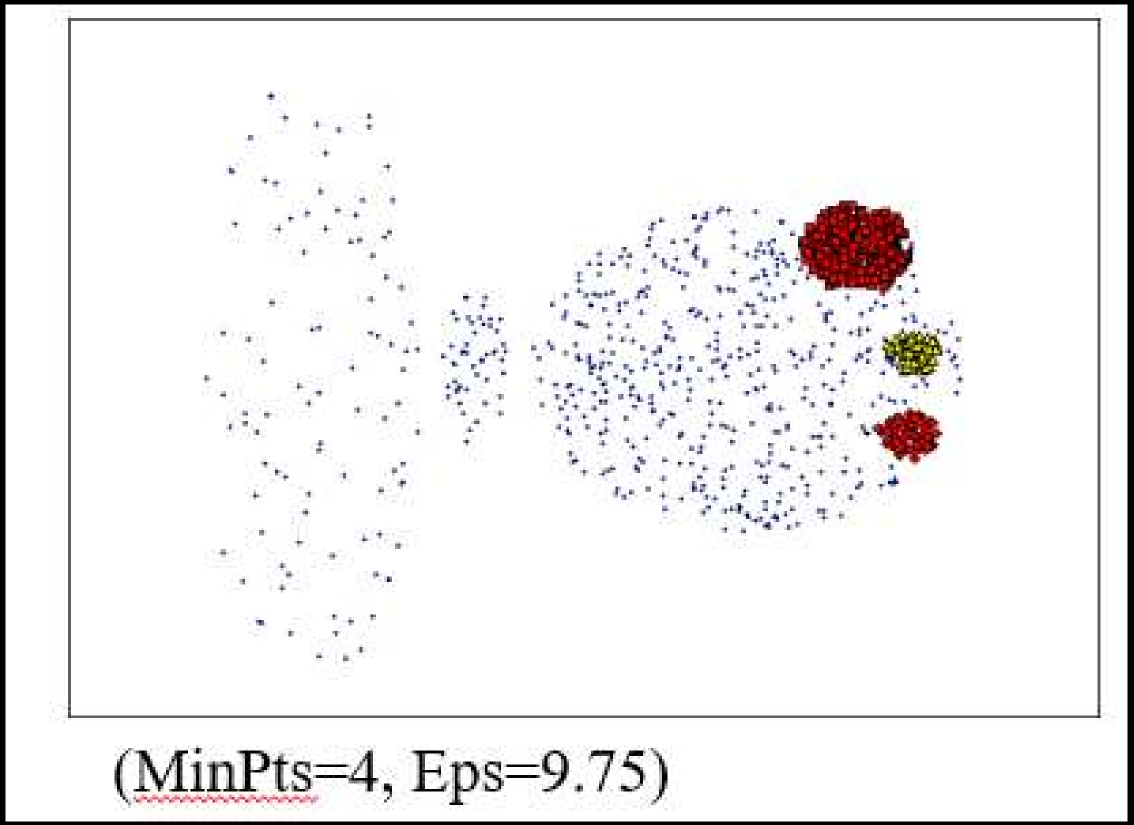




(MinPts=4, Eps=9.92).

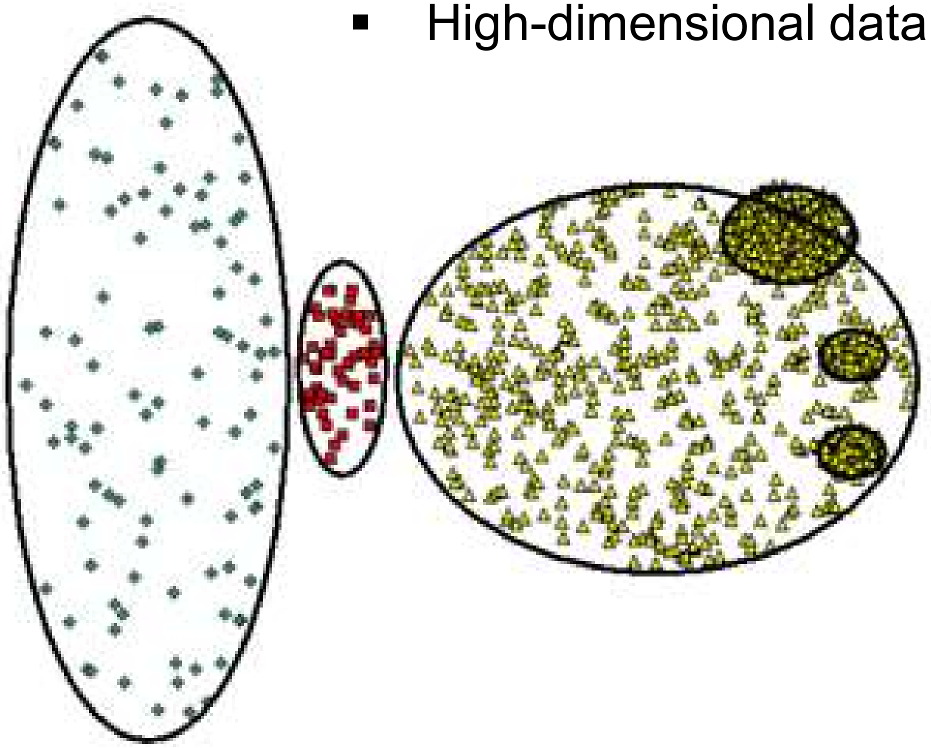
DBSCAN DOES NOT WORK WELL WHEN …



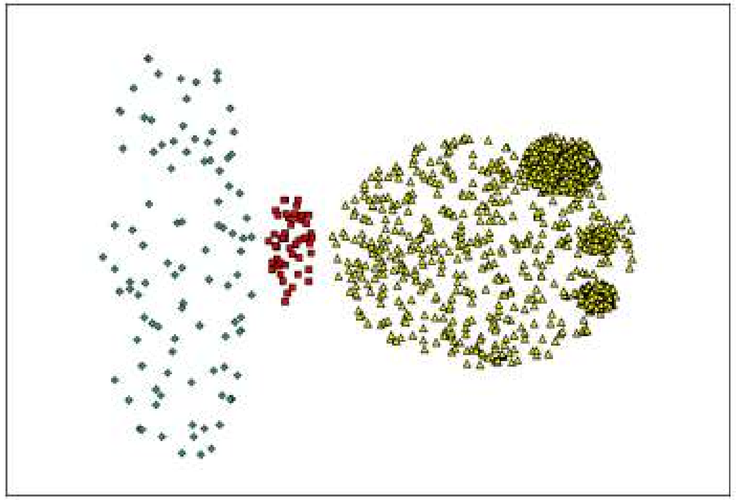


DBSCAN DOES NOT WORK WELL WHEN …

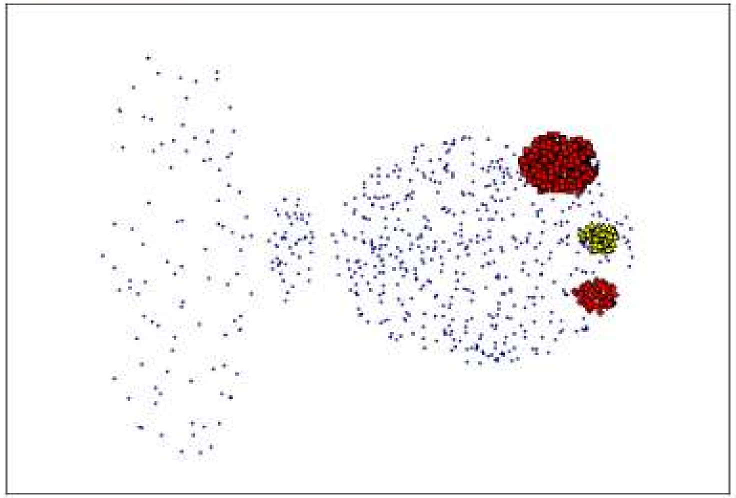
- Varying densities
- High-dimensional data



Original Points



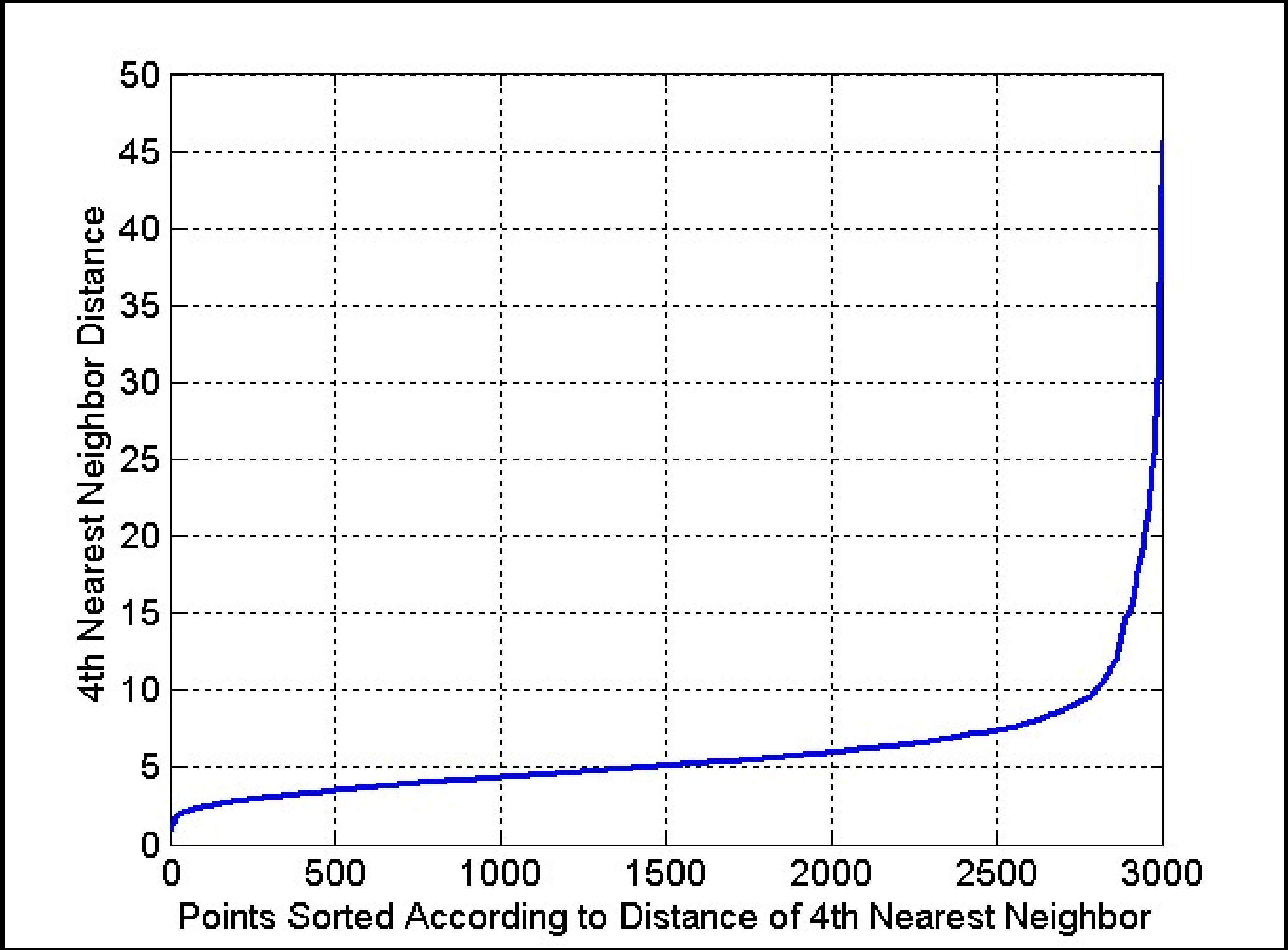
(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

DBSCAN – How to set parameters (Eps and MinPts)?

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at close distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor



DBSCAN VS K-MEANS

We assume that there are no ties in distances for either DBSCAN and K-means, we also assume that DBSCAN always assigns a border point which is associated with more core points to the closest core point.

- DBSCAN and K-means assign objects to a single cluster, but K-means assigns all objects while DBSCAN can discard noise objects
- DBSCAN can handle clusters of different sizes and shapes and it is not strongly affected by noise or outliers. K-means has difficulties with non-globular clusters and clusters of different sizes. Both algorithms perform poorly when clusters have widely differing densities
- K-means can only be used for data that has a well defined centroid, such as mean or median. DBSCAN requires that its definition of density, which is based on the traditional Euclidean notion of density, be meaningful for the data
- K-means can be applied to sparse, high-dimensional data, such as document data. DBSCAN typically performs poorly for such data because the traditional Euclidean definition of density does not work well for high-dimensional data

DBSCAN vs K-means

We assume that there are no ties in distances for either DBSCAN and K-means, we also assume that DBSCAN always assigns a border point which is associated with more core points to the closest core point.

- DBSCAN makes no assumption about the distribution of the data. The basic K-means is equivalent to a statistical clustering approach (Mixture Model) that assumes all clusters come from spherical Gaussian distributions with different means but the same covariance matrix
- DBSCAN and K-means both look for clusters using all attributes, that is, they do not look for clusters that may involve only a sub-set of the attributes
- K-means can find clusters that are not well separated, even if they overlap, but DBSCAN merges clusters that overlap
- K-means has complexity $O(N)$ while DBSCAN is $O(N^2)$ (except for low dimensional Euclidean data.)
- DBSCAN produces the same set of clusters from one run to another while K-means, which is typically used with random initialization of centroids, does not
- DBSCAN automatically determines the number of clusters, for K-means the number of clusters needs to be specified as a parameter

ADDITIONAL DENSITY-BASED CLUSTERNG ALGORITHMS

Many additional Density-based clustering techniques exist that address the issues of efficiency, finding clusters in subspaces, and more accurately modelling density.

They can be grouped in

- **GRID-BASED CLUSTERING**; they break the data space into grid cells and then form clusters from cells that are sufficiently dense. They can be effective and efficient at least for low-dimensional data (**CLIQUE**)
- **SUBSPACE CLUSTERING**; they look for clusters (dense regions) in subsets of all dimensions. For a data space with “n” attributes, potentially 2^{n-1} subspaces need to be searched, and thus an efficient technique is needed to do this (**COSA**)
- **KERNEL DENSITY FUNCTION**; they use kernel density functions to model density as the sum of the influences of individual data objects (**DENCLUE**)

RECAP

- Concept
- DBSCAN
  - Core point
  - Border point
  - Noise point
- Advantages
- Limitations
- DBSCAN vs K-means
- Additional algorithms