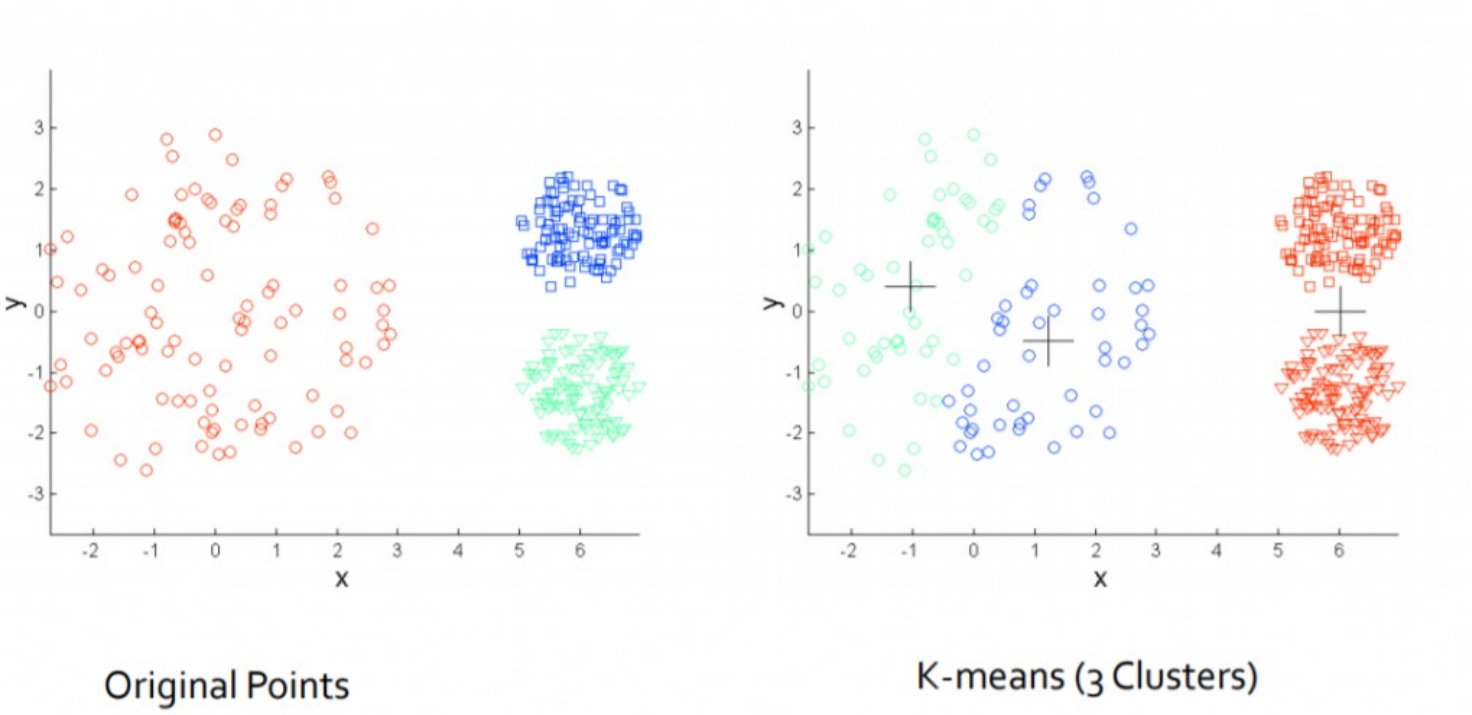
# Advanced Features Clustering

PLAKSHA UNIVERSITY

# K-means Clustering

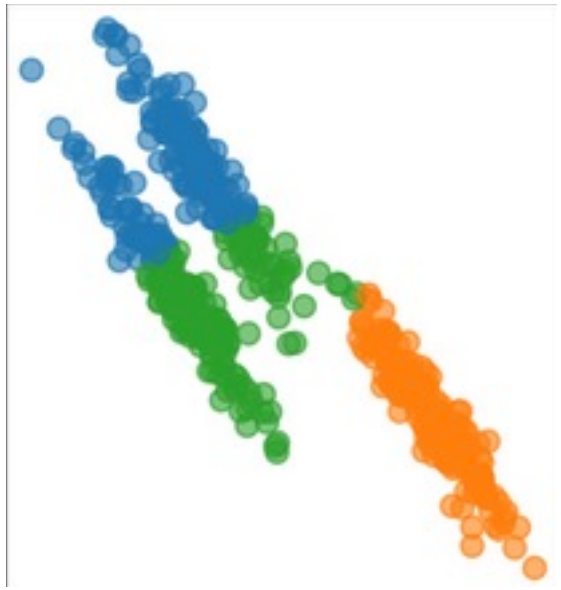
K-means Clustering is quite easy to implement and is intuitive.

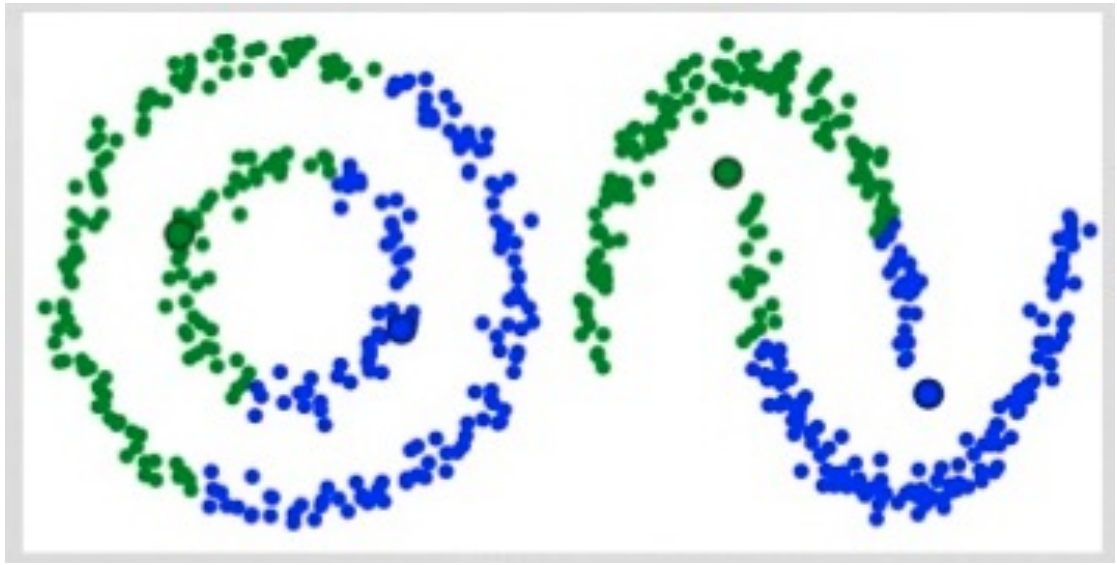
But, that doesn't mean that it's very efficient for all kinds of distributions.

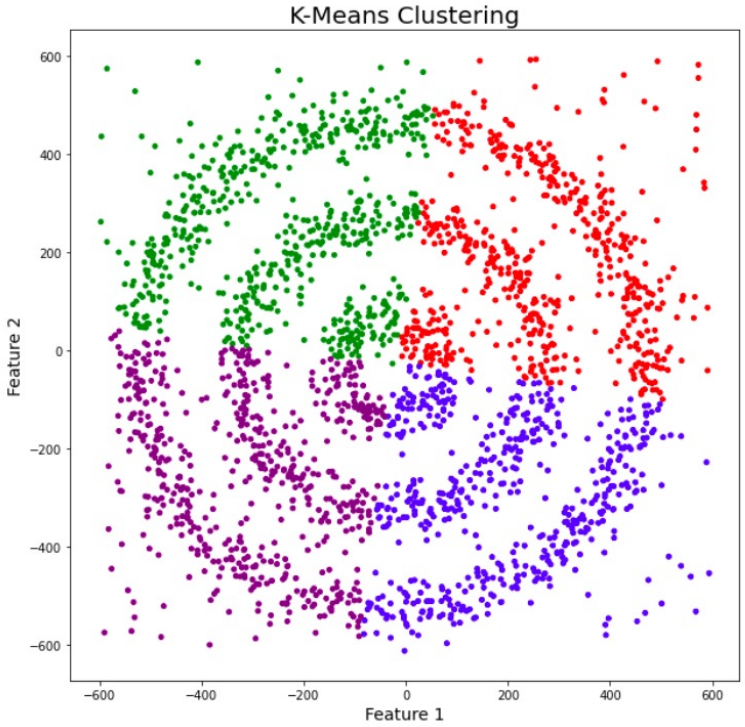


# K-means Clustering

In fact, there are many cases where K-means clustering method fails.








This is because it is purely a distance-based model.

# GMM Clustering

Thus, let's look at a more sophisticated clustering method called Gaussian Mixture Model (GMM) clustering.

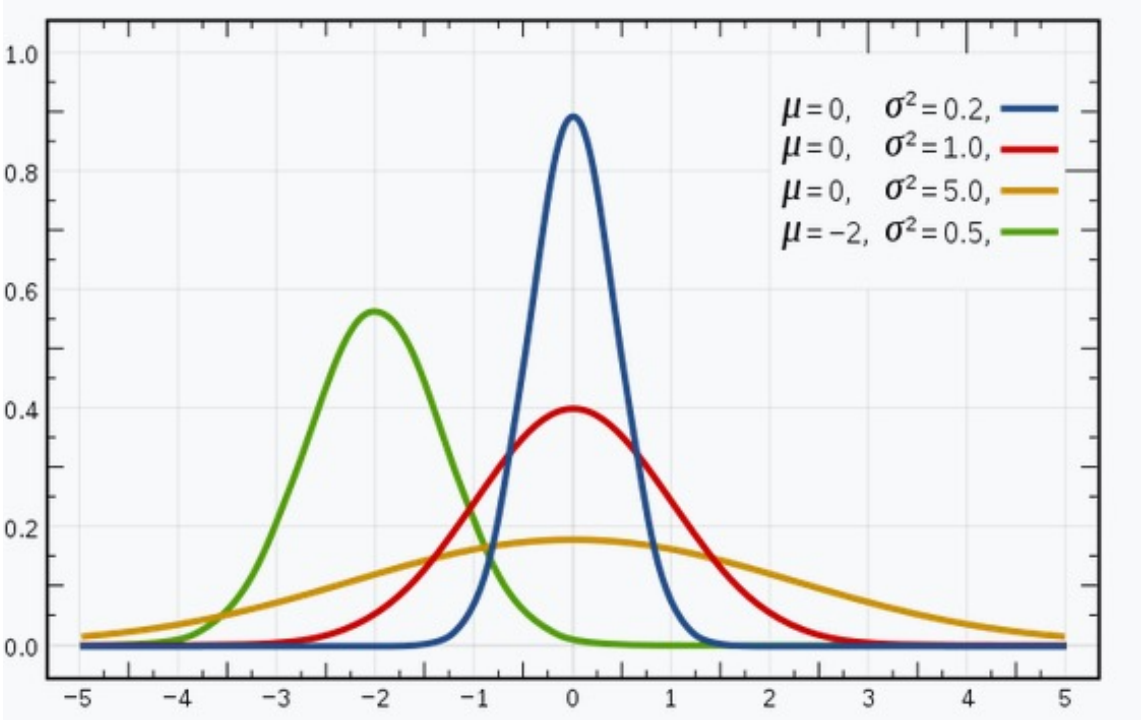
GMM is a distribution-based model rather than a distance-based model.

GMM is a probabilistic model which assumes that the data is made up of multiple Gaussian distributions which form individual clusters.

Since it is a probabilistic model, it allows each datapoint to be a part of multiple clusters simultaneously with different probabilities.

# GMM Clustering

The idea in GMM Clustering is to find each pair of (μ, σ²) that form Gaussian distributions representing the data and then find the probability of each point belonging to these distributions.




Remember: Higher the value of σ², higher will be the spread of the Gaussian distribution.

# Gaussian Distribution Revisit

Recall that for a one-dimensional Gaussian distribution, the probability density function is given by:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where $\mu$ is the mean of the distribution and $\sigma^2$ is the variance.

# Gaussian Distribution Revisit

Recall that for a two-dimensional Gaussian distribution, the probability density function is given by:

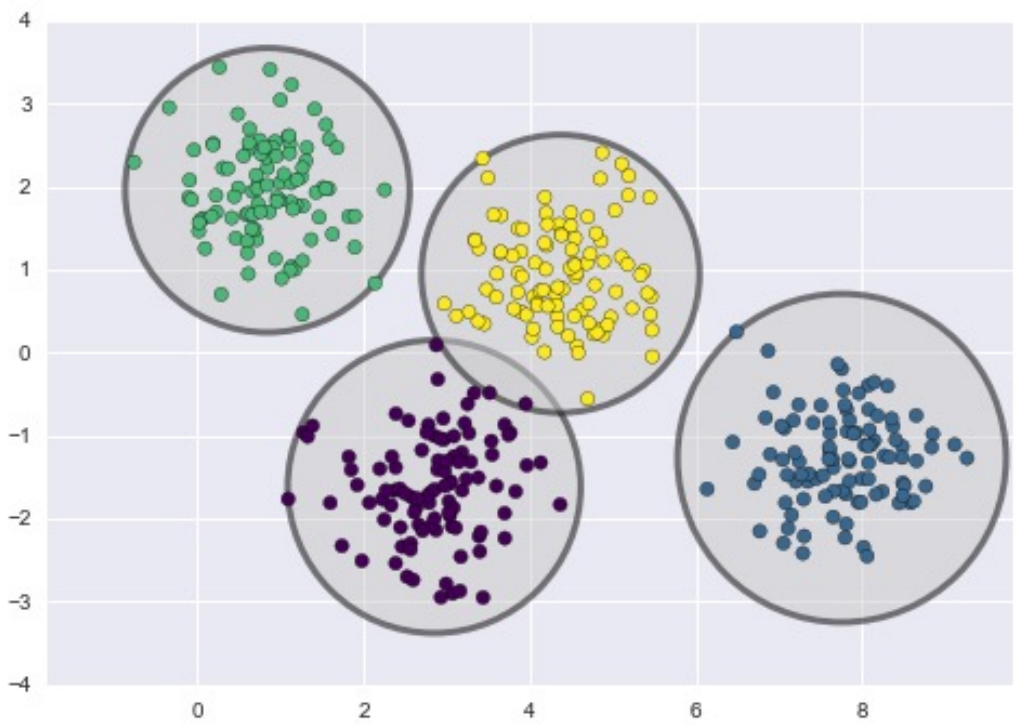
$$f(x \mid \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

where μ is the 2D mean vector and ∑ is 2x2 covariance matrix.

Thus, for multivariate Gaussian distributions, μ would be a N-dimensional mean vector and ∑ will be a NxN covariance matrix.

# GMM Problem Statement

Given K distributions, how do we find the best pairs of $(\mu, \sigma^2)$ that form those distributions?



# GMM Algorithm

GMM prerequisites

a) First, we choose a value of K (just like we did for K-means clustering).

b) Then, we randomly assign K number of means ($\mu_1, \mu_2, ..., \mu_K$)

c) Then, we randomly assign K number of covariances ($\Sigma_1, \Sigma_2, ..., \Sigma_K$)

d) Finally, for each such distribution, we calculate the value of $\Pi_i$ which is the density of that distribution i.e., how many datapoints belong to that distribution as ratio of total datapoints.

Now, we follow the two-step Expectation-Maximization algorithm until convergence to keep updating the values of ($\mu_1, \mu_2, ..., \mu_K$) and ($\Sigma_1, \Sigma_2, ..., \Sigma_K$).

The final values of ($\mu_1, \mu_2, ..., \mu_K$) and ($\Sigma_1, \Sigma_2, ..., \Sigma_K$) after convergence would be the distributions we are looking for.

# GMM Algorithm

Step 1: Expectation

For each point $x_i$, we calculate the probability that it belongs to cluster/distribution $c_1, c_2, \ldots c_K$.

$$r_{ic} = \frac{\text{Probability Xi belongs to c}}{\text{Sum of probability Xi belongs to } c_1, c_2, .. c_k} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$

Our aim is to maximize this probability.

# GMM Algorithm

## Step 2: Maximization

After calculating the probability of each point $x_i$ of belonging to each cluster in the Expectation step, we update the values of $\Pi$, $\mu$ and $\Sigma$ as below.

$$\Pi_c = \frac{\text{Number of points assigned to cluster}}{\text{Total number of points}}$$

$$\boldsymbol{\mu}_c = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} x_i$$

$$\Sigma_c = \frac{1}{\text{Number of points assigned to cluster}} \sum_i r_{ic} (x_i - \boldsymbol{\mu}_c)^T (x_i - \boldsymbol{\mu}_c)$$

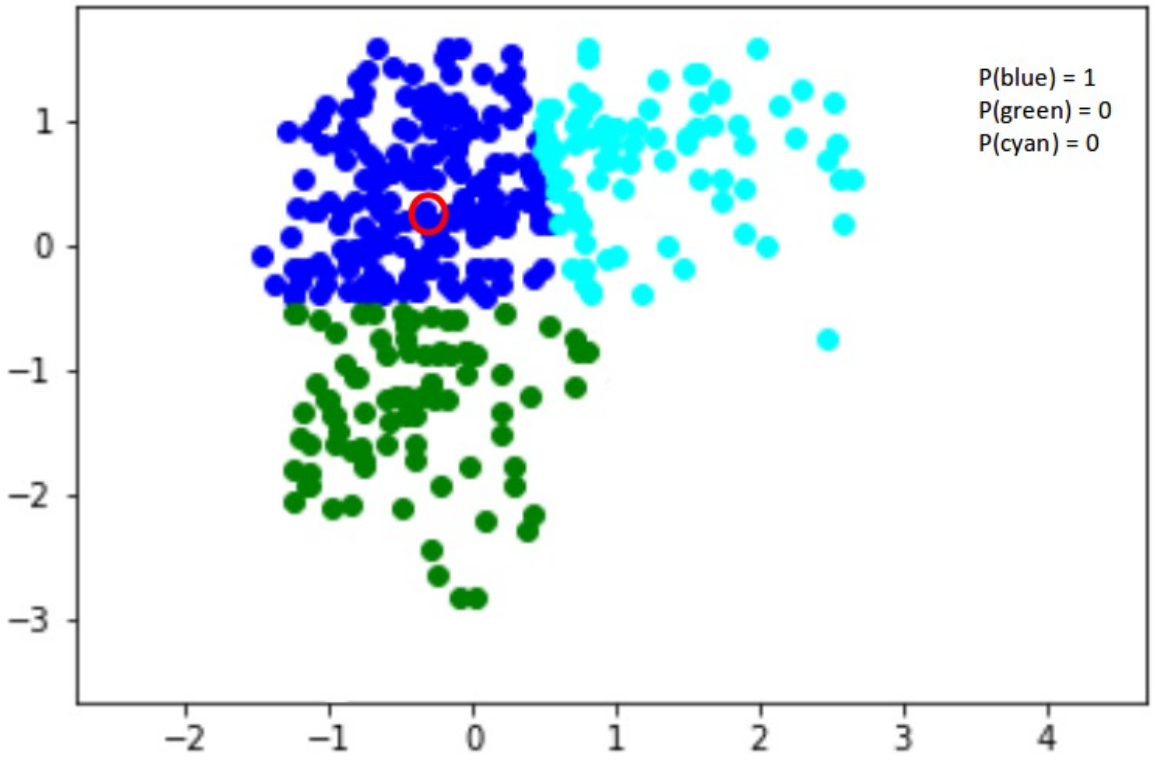
# GMM Algorithm

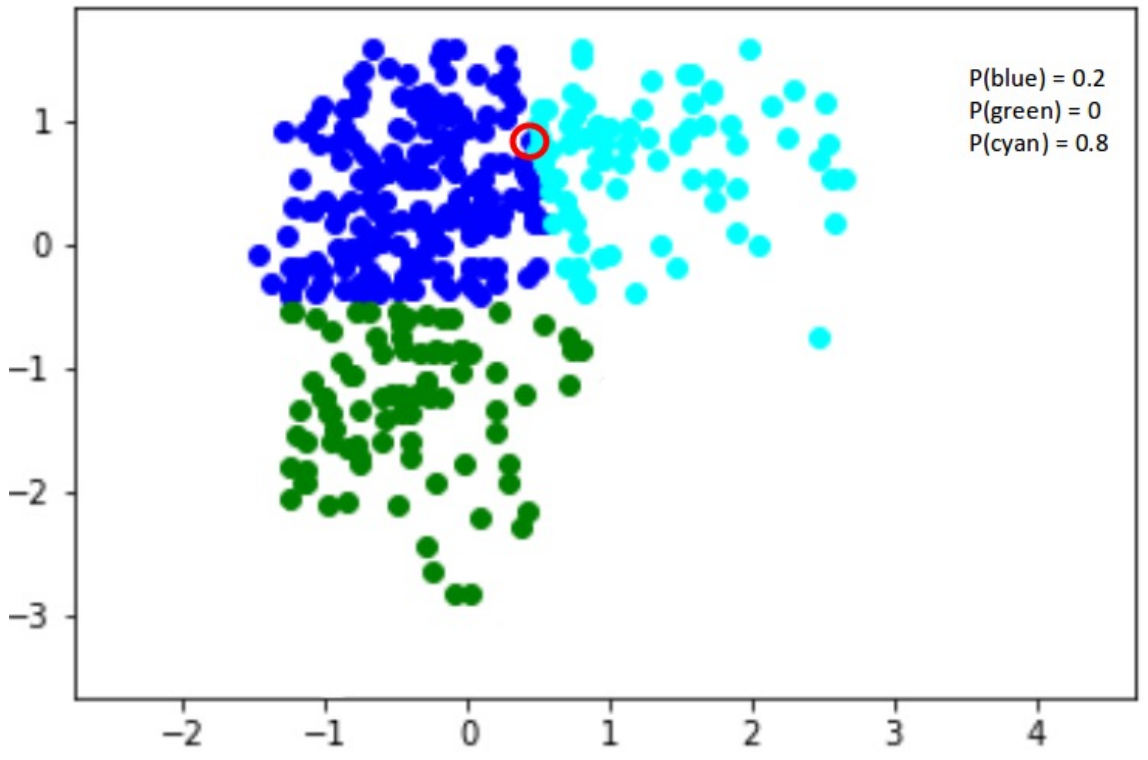
Just like K-means, we keep following the E-step and the M-step until convergence.

Convergence in GMM happens when either the change in $\mu$ and $\Sigma$ parameters falls below a threshold or when a total number of iterations has been reached.

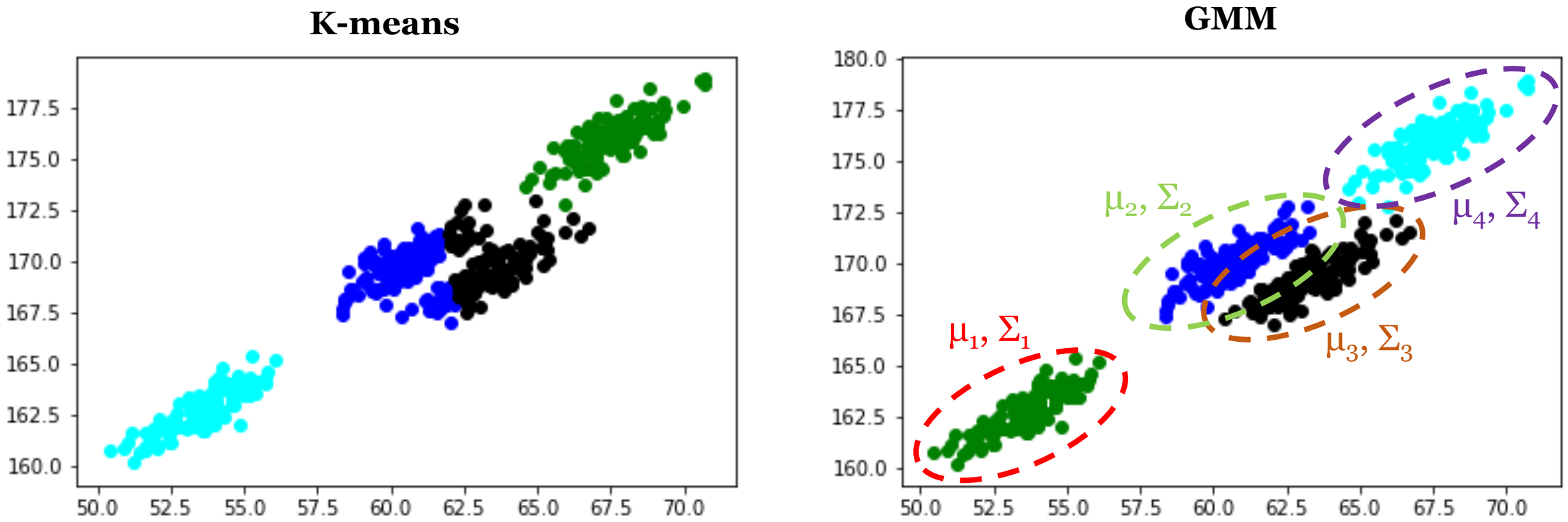
In effect, K-means work only by taking into account the distance from means of data, while GMM works by taking into account both means and variances of the distribution.

# GMM Algorithm



# K-Means vs. GMM Example



# DBScan Algorithm

Let's now look at another powerful clustering algorithm called DBScan i.e., Density-Based Spatial Clustering of Applications with Noise algorithm.

This algorithm is quite powerful because:

a) Unlike K-Means and GMM, one does not need to specify the value of K before-hand.

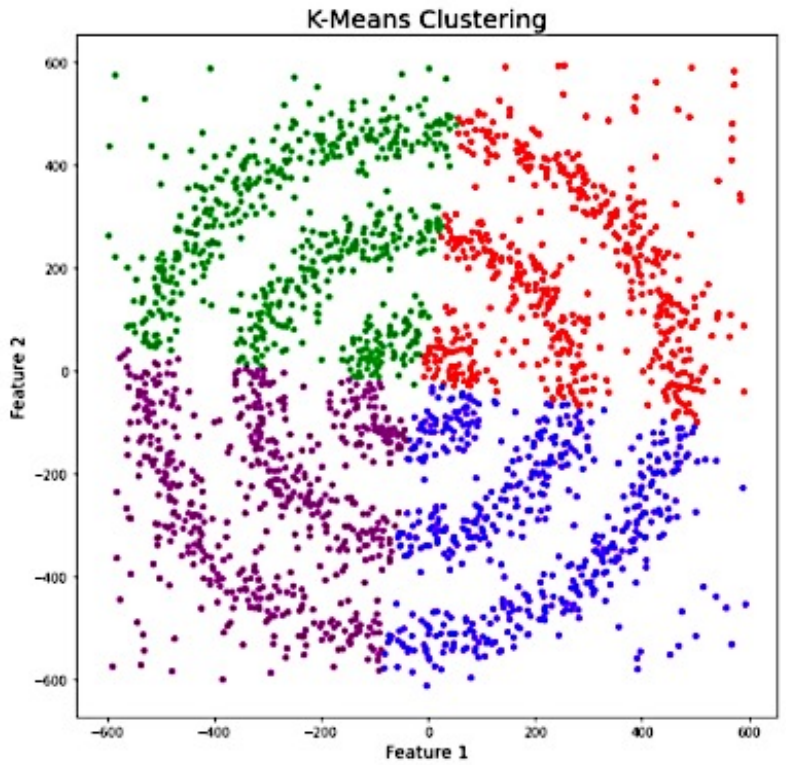
b) Unlike K-means and GMM, this algorithm can also find noise in data and does not cluster those noisy datapoints into one of the clusters.

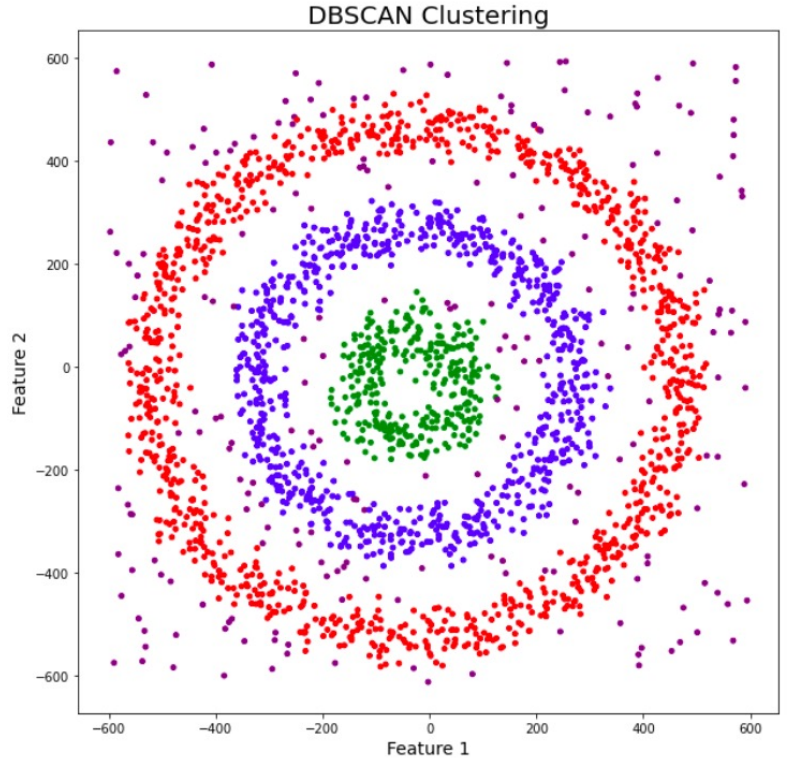
# DBScan Algorithm

DBScan is quite useful because real-life data may contain irregularities

a) Clusters can be of arbitrary shape.

b) Data may have noise.

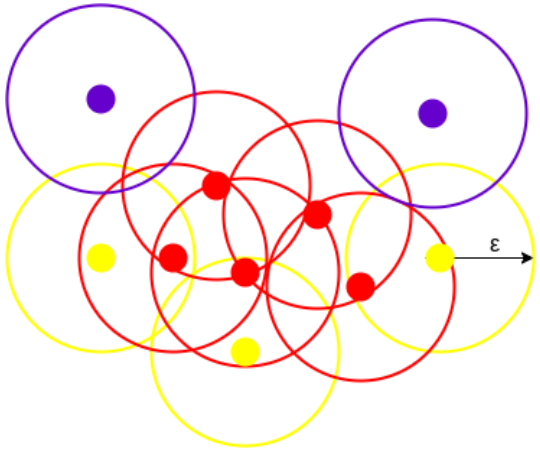


# DBScan Algorithm

DBScan uses two parameters, *minPoints* and *Epsilon* (ε).

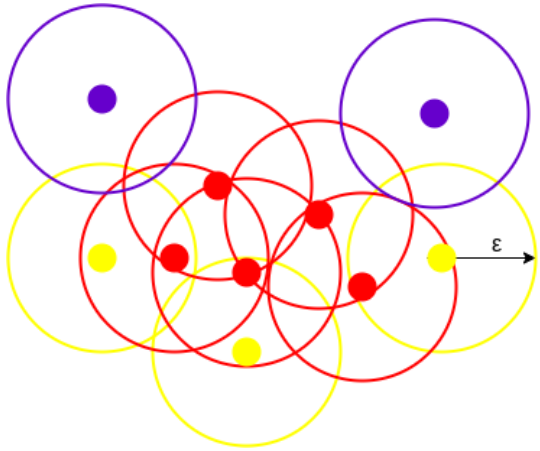
ε is the radius of the circle to check the density of the distribution and *minPoints* is the minimum number of data points inside that circle.




In higher dimensions, a circle becomes a hypersphere, ε becomes the radius of that hypersphere and *minPoints* is the number of points in the hypersphere.

# DBScan Algorithm

DBScan works by creating a circle of ε radius around every datapoint and classifies them into a Core point, Border point, or Noise.




DBScan with *minPoints* = 3
Red = core point, Yellow = border point, Purple = Noise

If the circle around a point has at least *minPoints*, then it is labeled a Core point.

If the circle around a point has less than *minPoints*, then it is labeled a Border point.

If there are no other data points around a point, then it is labeled Noise.

# DBScan Algorithm

Choosing the right values of *minPoints* and ε are key to efficient working of the DBScan algorithm.

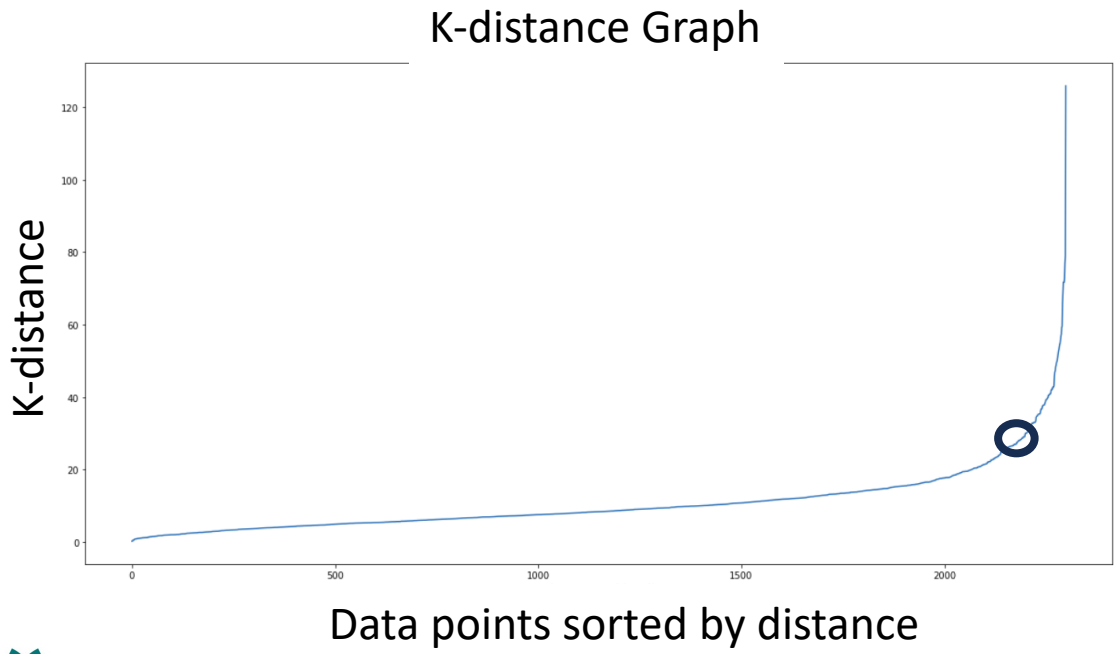
In general, *minPoints* >= *Dimensions* + 1

If *minPoints* is taken as too small, each data point will become its own cluster!

# DBScan Algorithm

Choosing the right values of *minPoints* and ε are key to efficient working of the DBScan algorithm.

The optimal value of ε is determined by the elbow point on the K-distance graph.




X-axis: Number of datapoints in the dataset that are within K-distance from each other (sorted)

Y-axis: K-distance.

In this graph, elbow point is around K-distance value of 30, so a good value of ε could be 30.

# DBScan Algorithm

Reachability and Connectivity

- Reachability refers to the state when a data point can be accessed from another data point directly or indirectly.
- Connectivity refers to the state when two data points are a parts of the same cluster or not.
- Any two data points can be either:
  - Directly density-reachable
  - Density-reachable
  - Density-connected

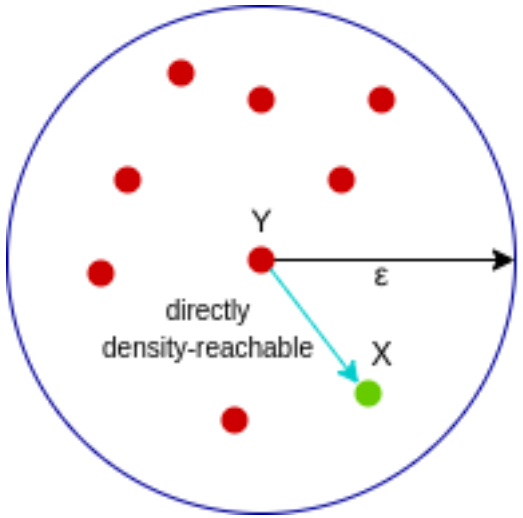
# DBScan Algorithm

Directly Density-reachable

A point X is Directly Density-reachable from a point Y if

a) X belongs to the neighborhood of Y, i.e., *dist(X, Y)* <= ε

b) Y is a core point.



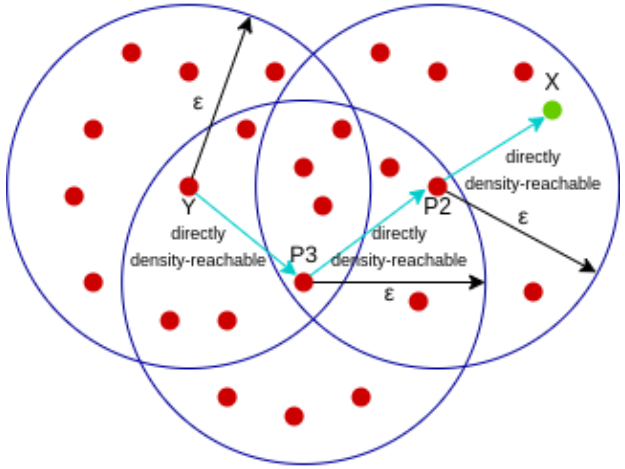

Here, X is Directly Density-reachable from Y.

# DBScan Algorithm

Density-reachable

A point X is Directly Density-reachable from a point Y if

There are a chain of points $P_1$, $P_2$, $P_3$..., $P_N$ where $P_1 = X$, and $P_N = Y$ such that $P_{i+1}$ is directly density-reachable from $P_i$



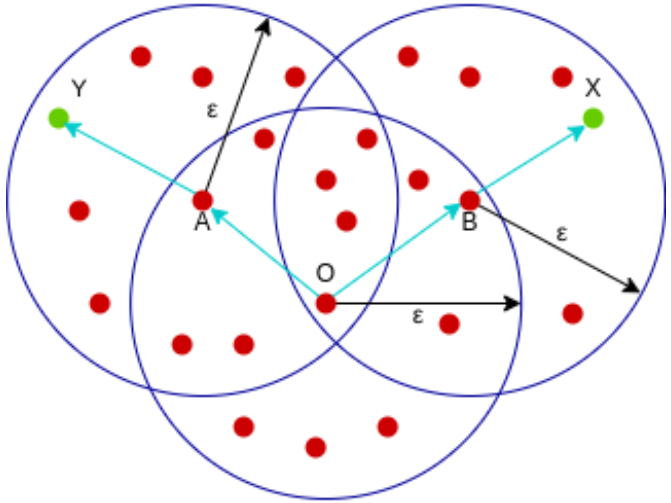

Here, X is density-reachable from Y.
X is directly density-reachable from P2.
P2 is directly density-reachable from P3.
P3 is directly density-reachable from Y.

# DBScan Algorithm

Density-connected

A point X is density-connected from a point Y if

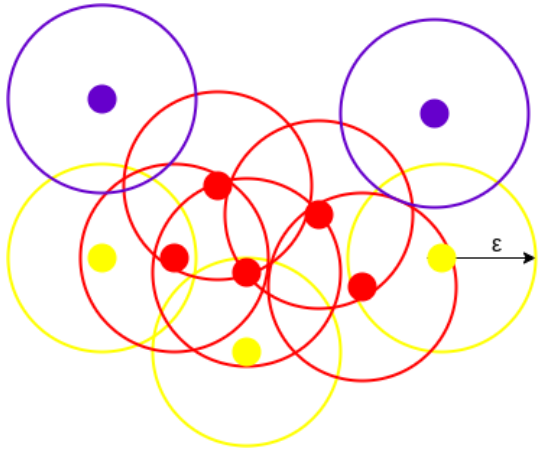
There exists a point O such that both X and Y are density-reachable from O.




Here, both X and Y are density reachable from O via points B and A respectively.
Thus, X is density-connected from point Y.

# DBScan Algorithm

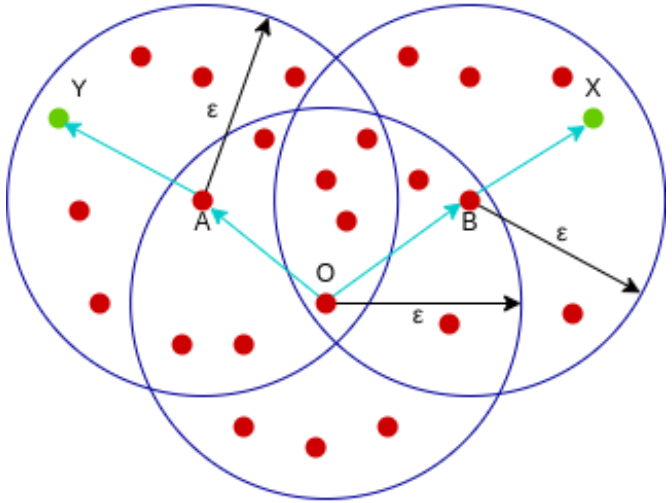
Step 1: Identify all the core points in the data using *MinPoints* and ε.




Step 2: For each core point, if it is not already assigned a cluster, create a new cluster.

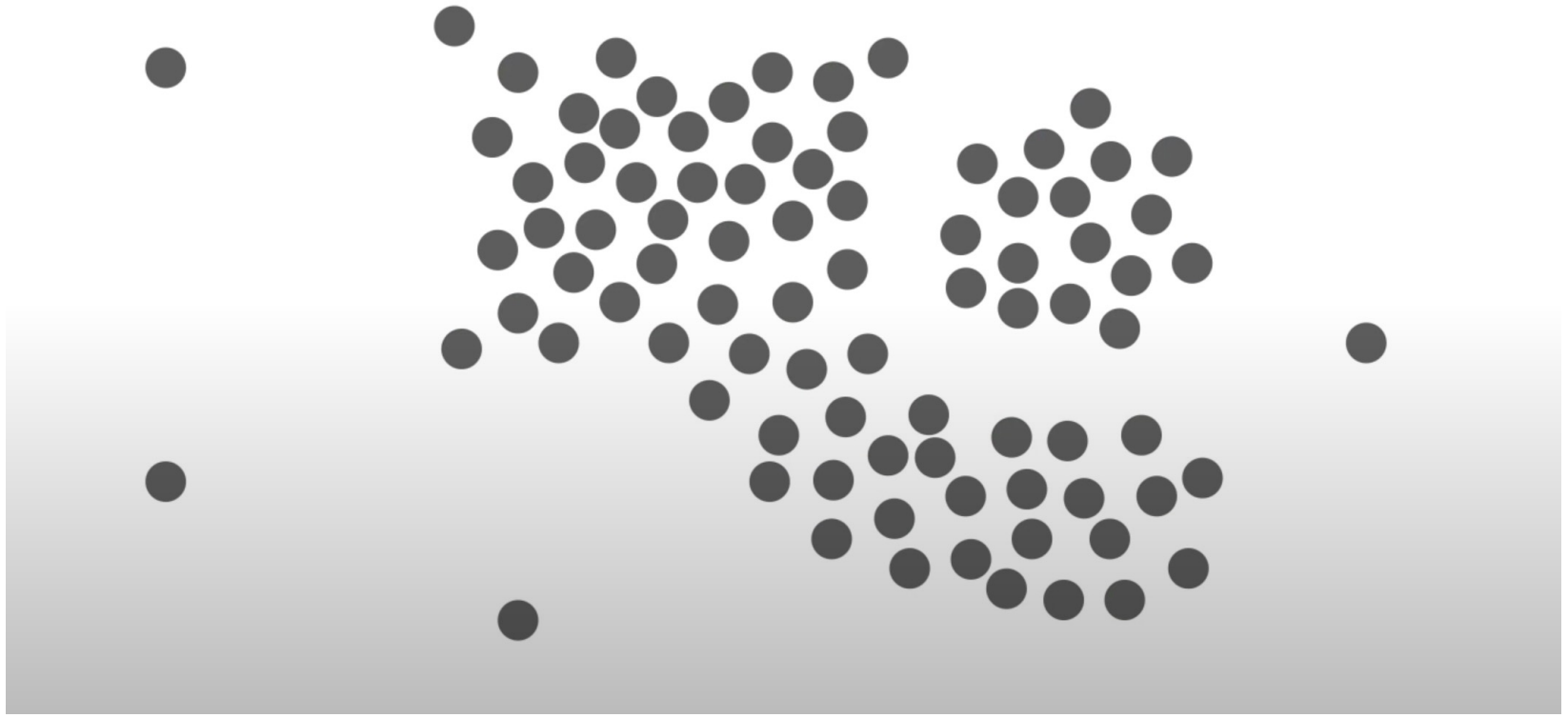
# DBScan Algorithm

Step 3: Find recursively all density-connected points and assign them to the same cluster as the core point. Clustering has begun.



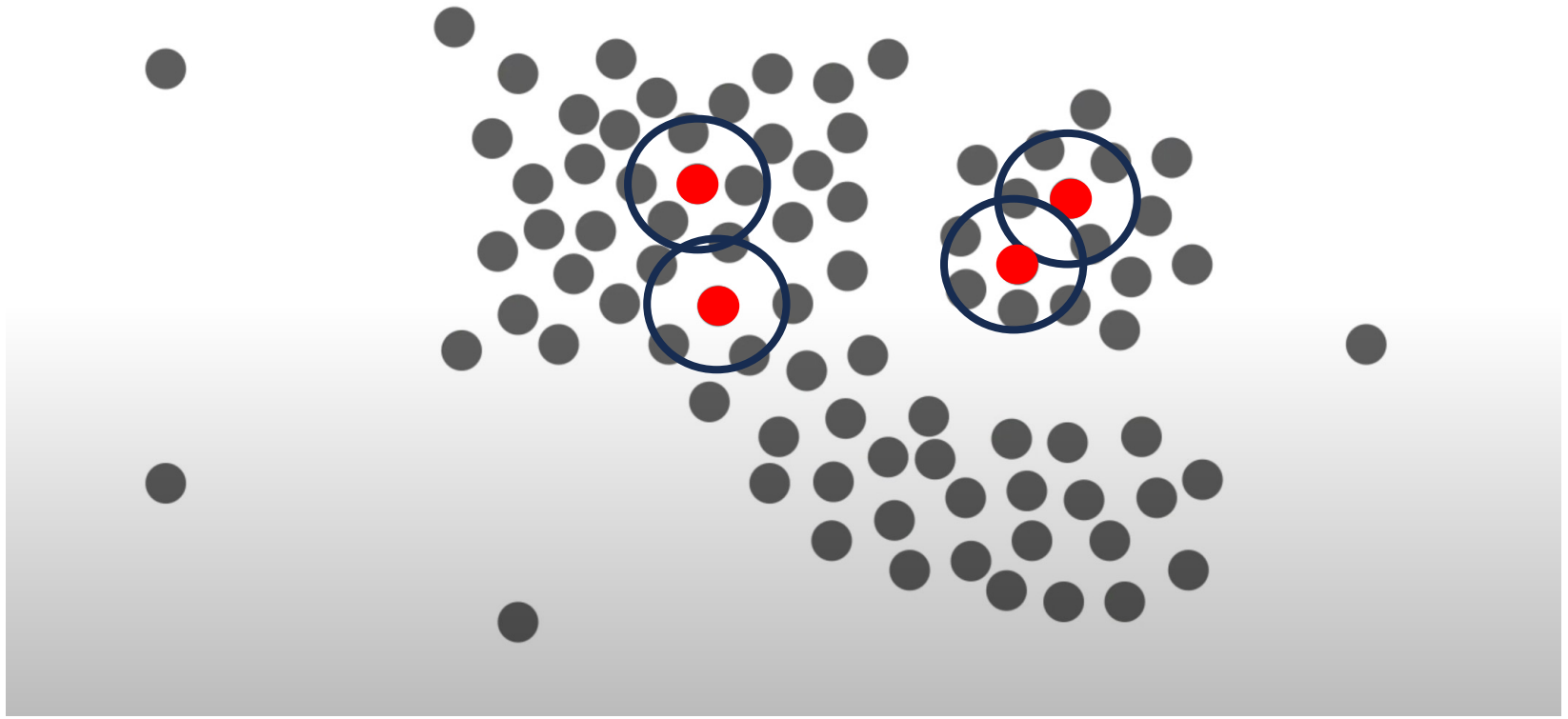

Step 4: Iterate through all points in the dataset. Those points that do not belong to any cluster are labeled as noise.

# DBScan Algorithm in Action



**Raw Data**

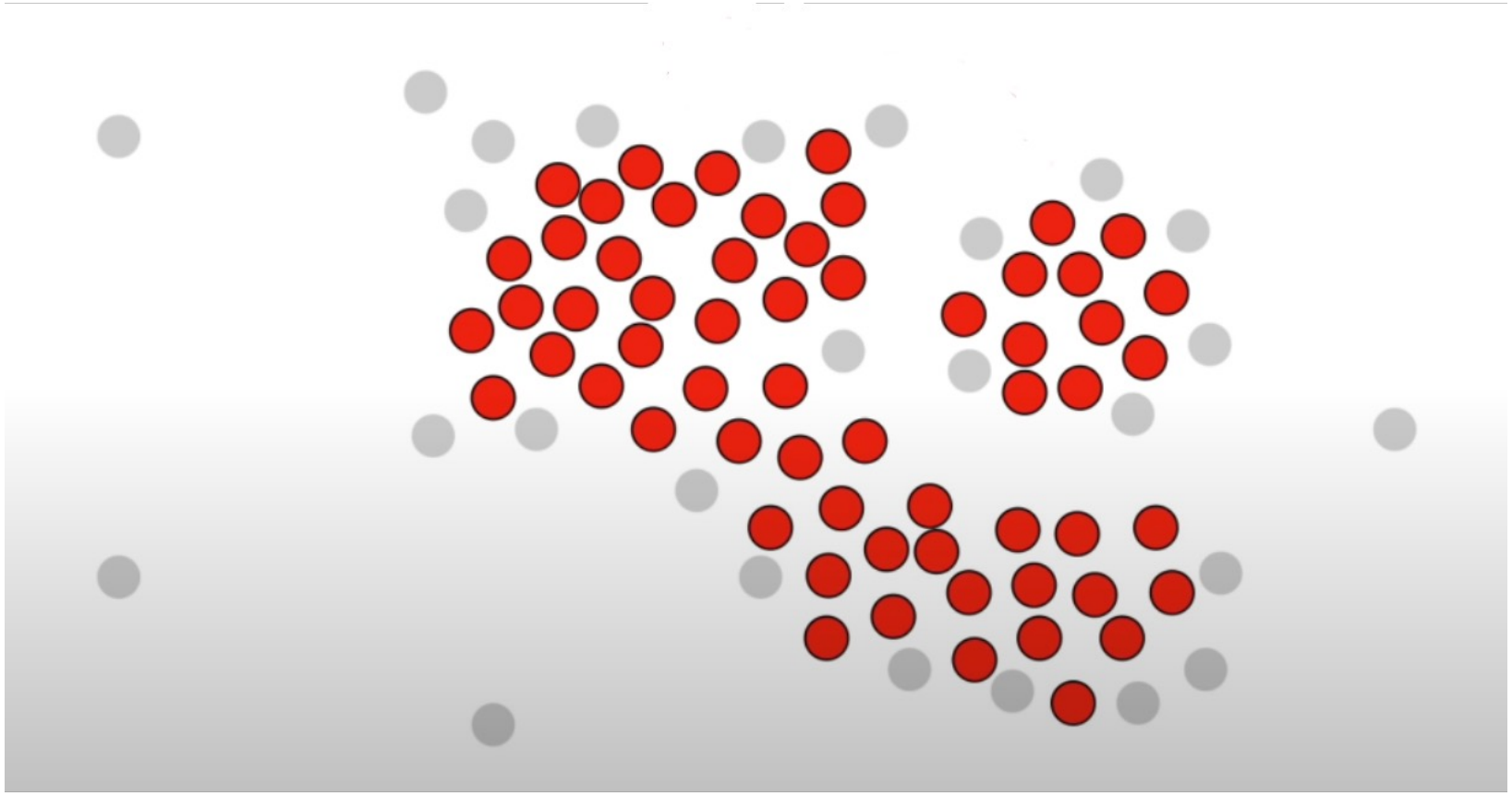
# DBScan Algorithm in Action



**Start finding the Core Points**

Let's say
*MinPoints* = 4
ε = circle on the left

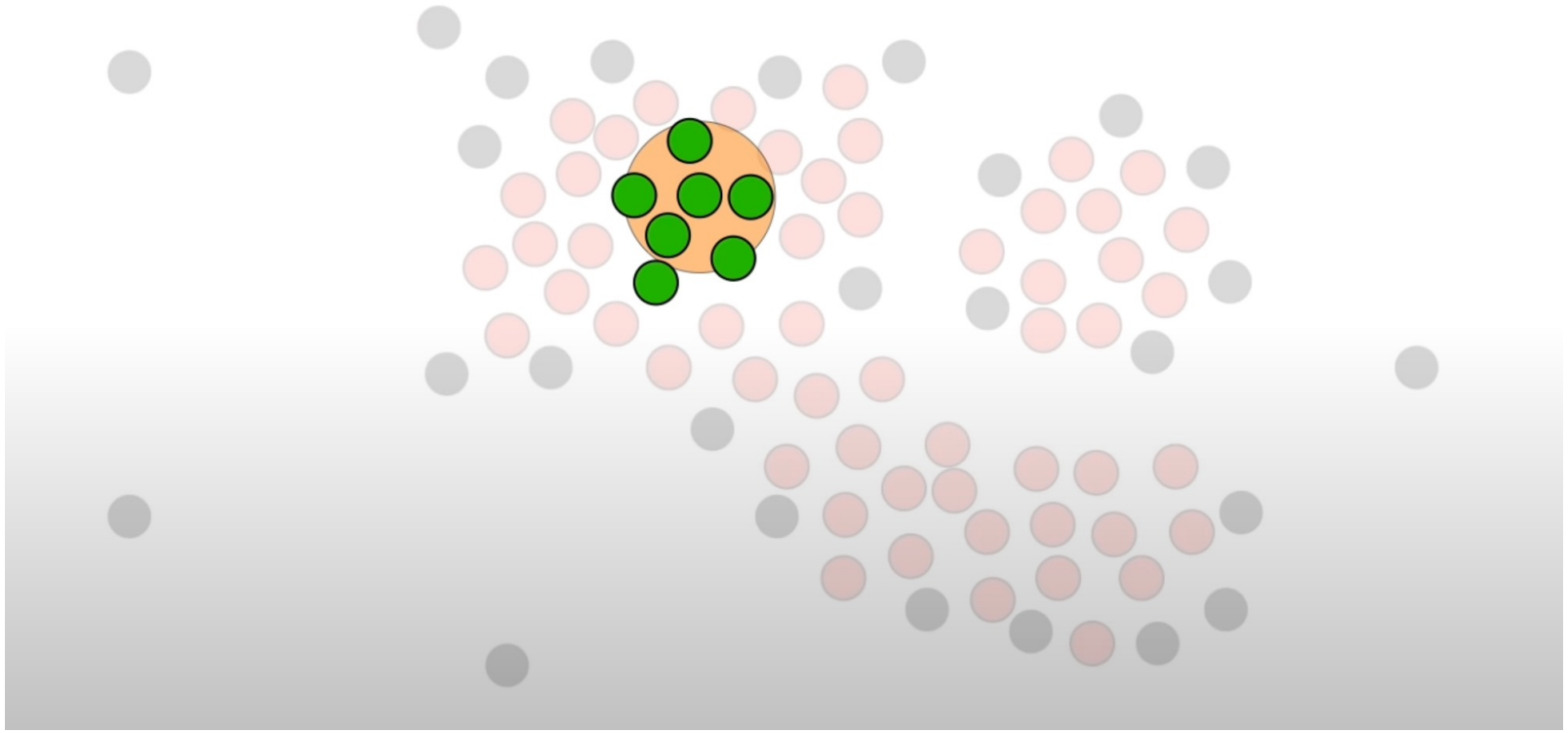
# DBScan Algorithm in Action



**Done finding the Core Points**

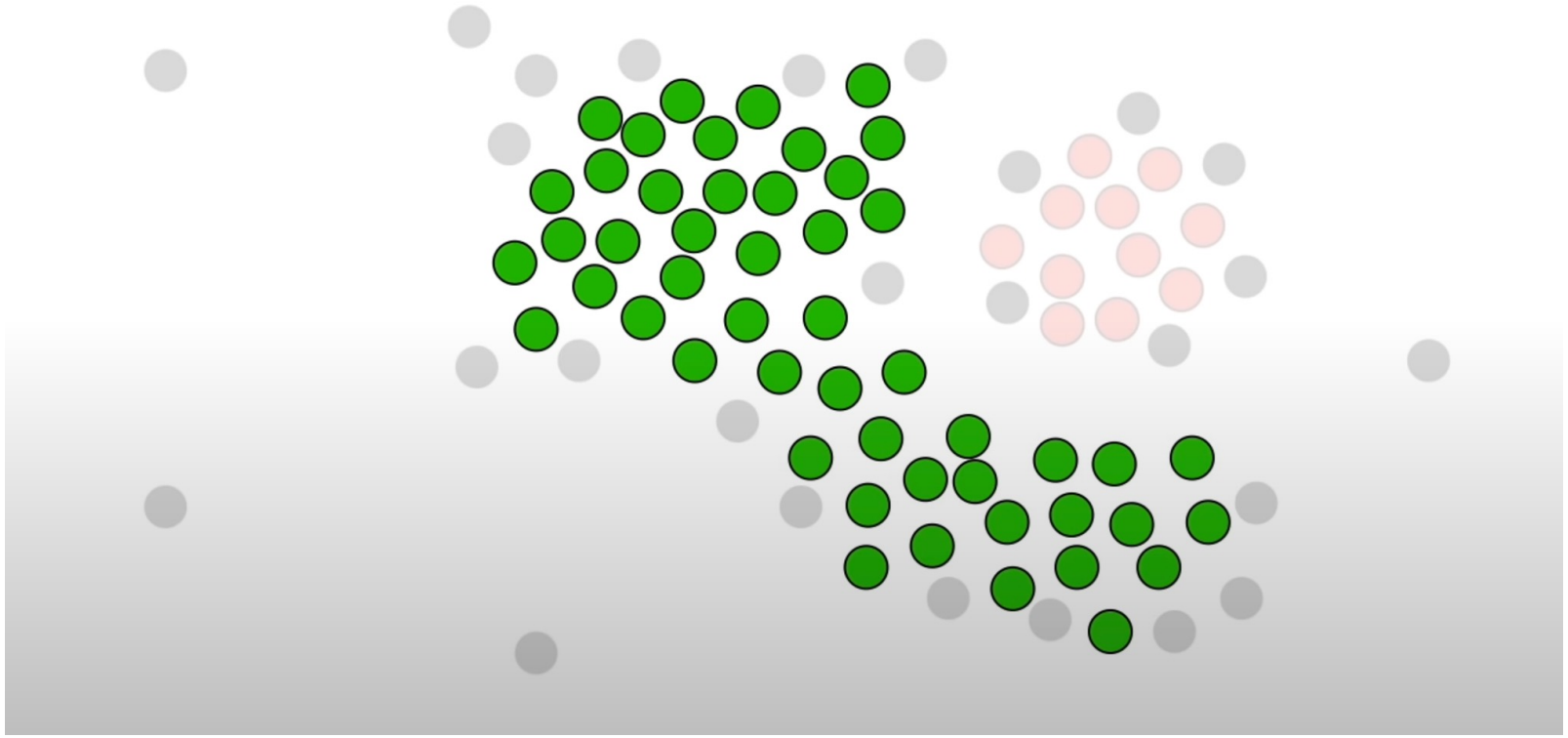
Let's say
*MinPoints* = 4
ε = circle on the left

# DBScan Algorithm in Action



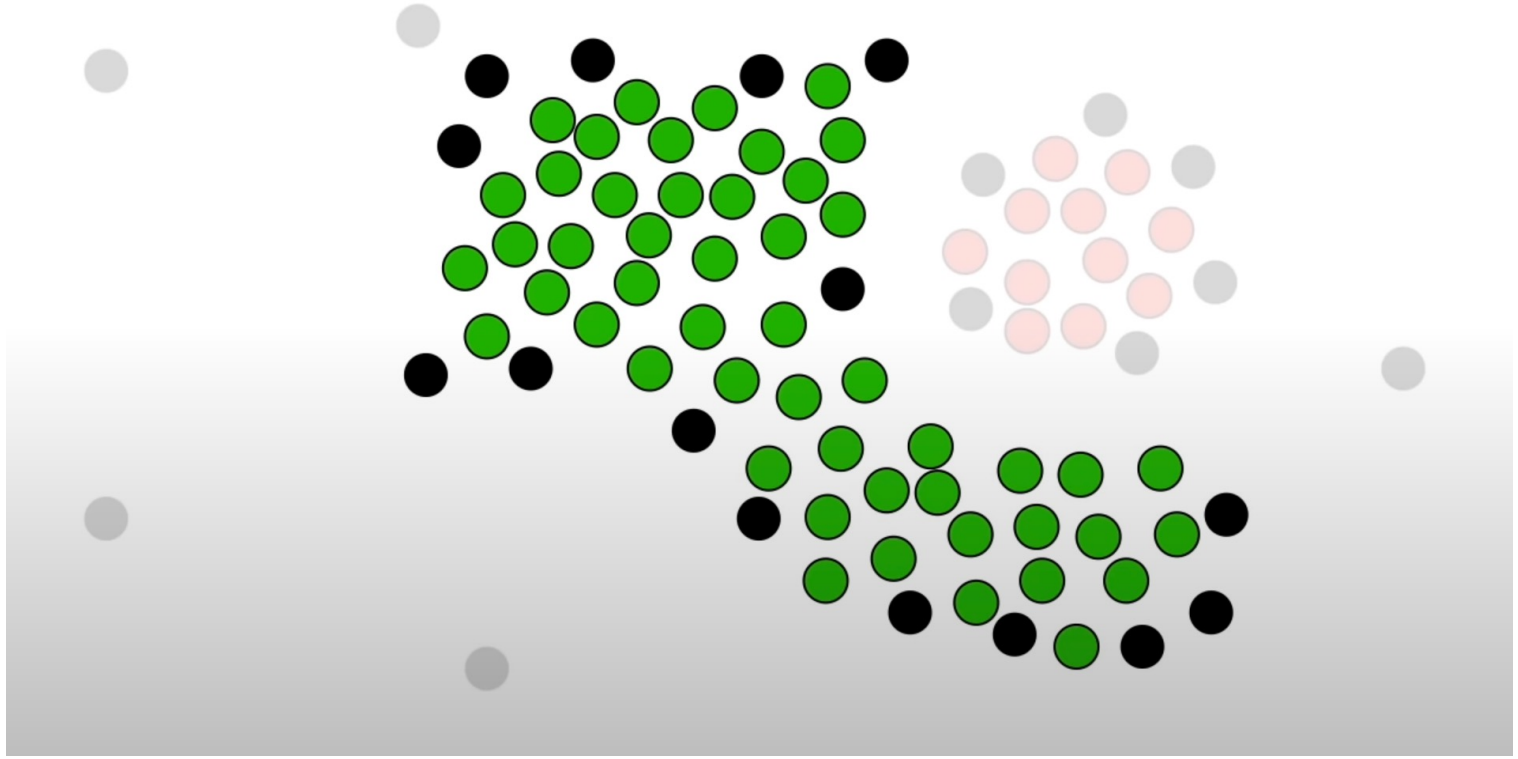
**Randomly choose a Core Point**
**Start adding neighboring Core Points to a cluster**

# DBScan Algorithm in Action



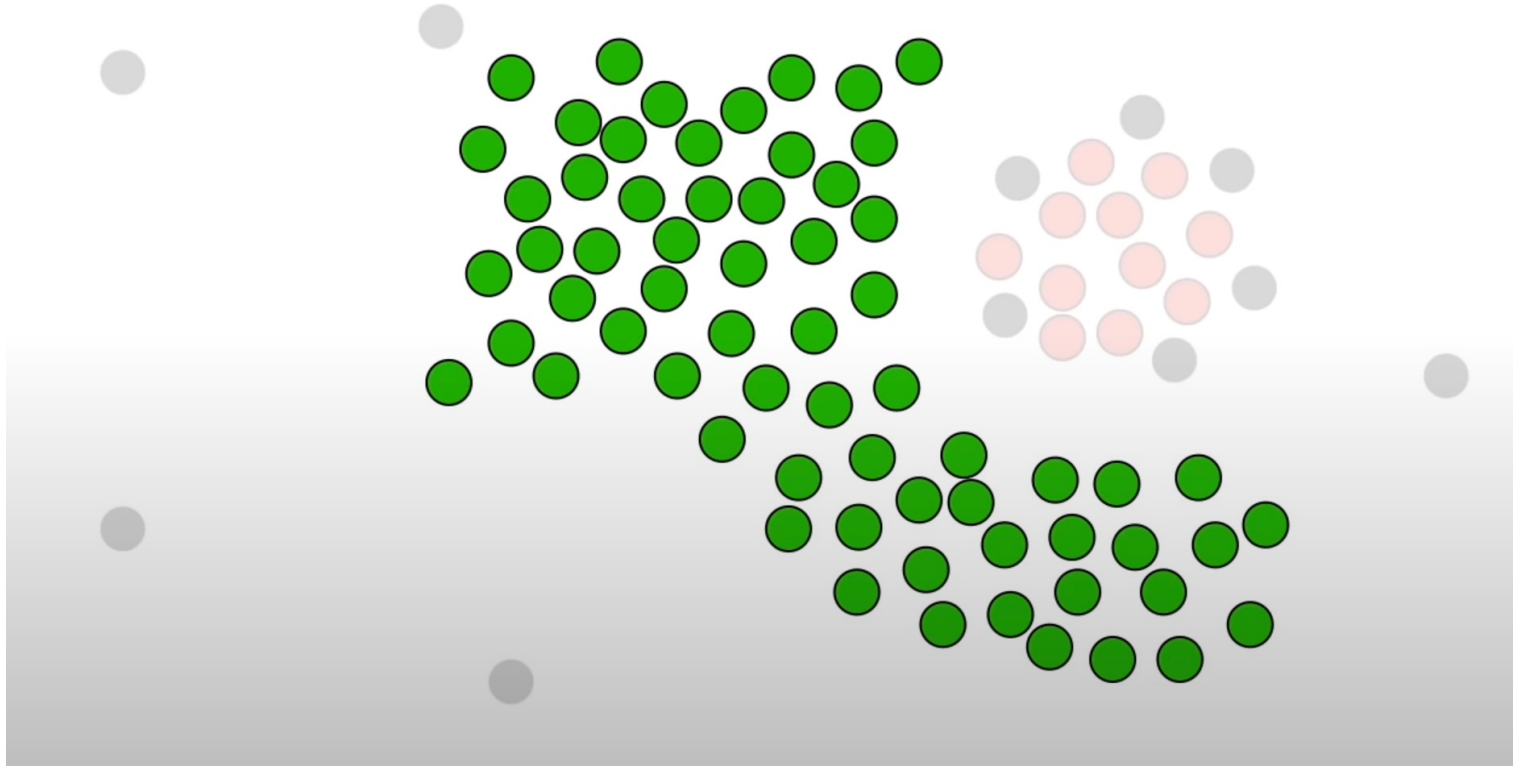
**Done adding all core points that could be added to the first cluster. All other core points will form the second cluster and so on.**

# DBScan Algorithm in Action



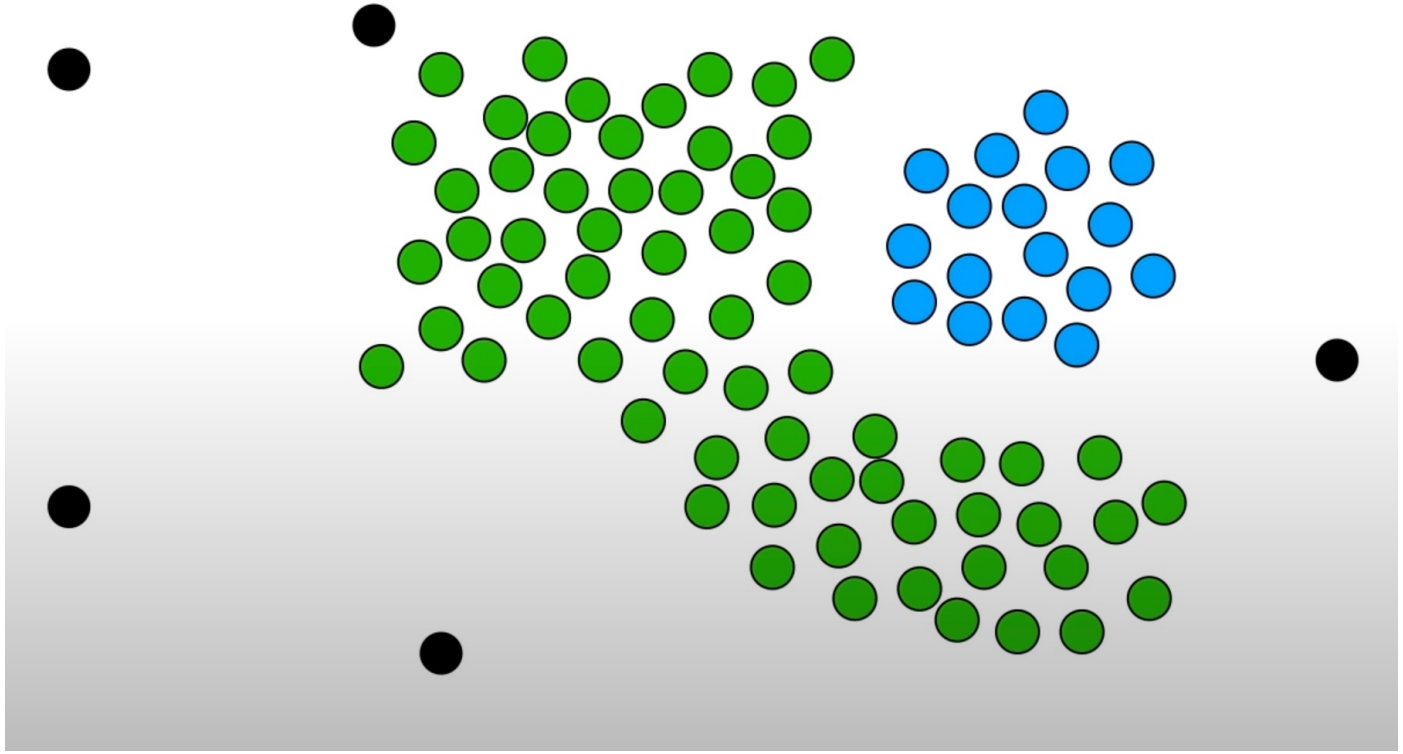
**Identify all the density connected points of the first cluster**

# DBScan Algorithm in Action



**Add all density connected points to the first cluster**

# DBScan Algorithm in Action



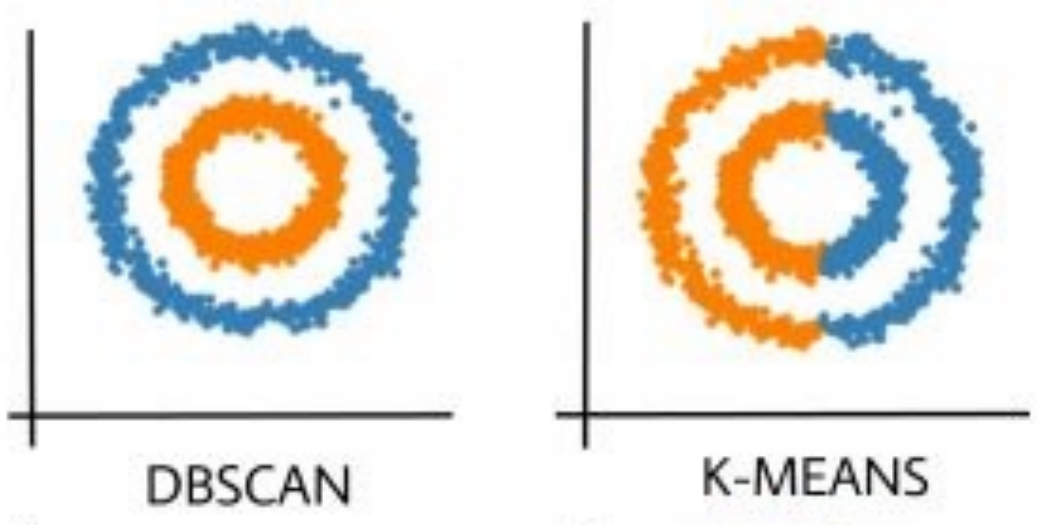
**The points that could not be added to either cluster are the outliers!**

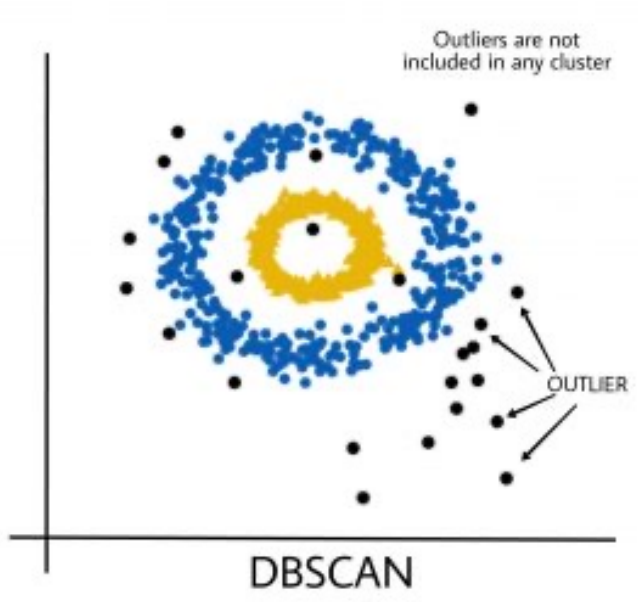
**Do the same with core points and density connected points of the second cluster**
**The points that could not be added to any clusters**

# DBScan vs. K-Means

| DBScan | K-Means |
| --- | --- |
| • K does not need to be specified. | • K has to be user-provided. |
| • Clusters can be of arbitrary shape. | • Clusters are spherical or convex. |
| • Works well with datasets containing noise and outliers. | • Outliers can skew the clustering to a large extent. |
| • Two parameters to train the model. | • Only one parameter required. |

# DBScan vs. K-Means



# Summary

This lecture

- GMM Clustering Method

- DBScan Clustering Method and advantages over K-Means Clustering Method

Next lecture

- We will proceed from Features Clustering to Features Dimensionality Reduction

- We will study the Curse of Dimensionality and Principal Component Analysis (PCA) algorithm

# Questions?


Intro to ML
Features Extraction
Features Clustering
Features Dim Reduction
Features Classification
ML Systems Challenges
ML Real-world Applications
Intro to NNs
Multi-layer NNs
“Deep” NNs
Other ML Methods