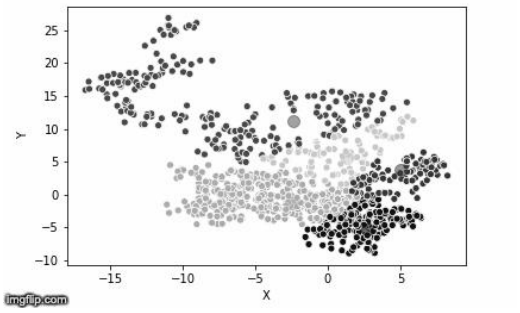


DBSCAN Clustering

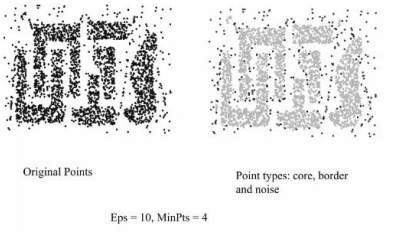
Concept

- **DBSCAN is a density-based algorithm**
- DBScan stands for Density-Based Spatial Clustering of Applications with Noise
- Density-based Clustering locates regions of high density that are separated from one another by regions of low density

Density = number of points within a specified radius (Eps)

eps -Radius around each point
MinPts- Min no of points that should be arround that point within that radius

Concepts: Preliminary



Original Points

Point types: core, border and noise

Eps = 10, MinPts = 4

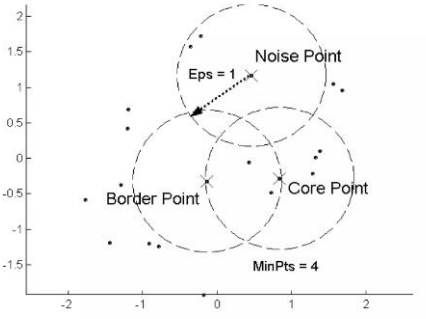
Concepts: Preliminary

- 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

Concepts: Preliminary

- Any two core points are close enough– within a distance *Eps* of one another – are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are discarded

Concepts: Core, Border, Noise



Clustering

Original Points

Clusters

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

DBScan Algorithm

Input: N objects to be clustered and global parameters *Eps*, *MinPts*.
Output: Clusters of objects.

Algorithm:

1) Arbitrary select a point *P*.
2) Retrieve all points density-reachable from *P* wrt ***Eps*** and ***MinPts***.
3) If *P* is a core point, a cluster is formed.
4) If *P* is a border point, no points are density-reachable from *P* and **DBSCAN** visits the next point of the database.
5) Continue the process until all of the points have been processed.

DBSCAN : Advantages

- Does not require one to specify the number of clusters in the data
- Can find arbitrarily shaped clusters. even find a cluster completely surrounded by a different cluster.
- Has a notion of noise, and is robust to outliers.
- Requires just two parameters and is mostly insensitive to the ordering of the points in the database.
- Designed for accelerate region queries.
- minPts and ε can be set by a domain expert

DBSCAN : Disadvantages

- DBSCAN is not entirely deterministic: Border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.
- The quality of DBSCAN depends on the distance measure used in the function regionQuery. (such as Euclidean distance)
- If the data and scale are not well understood, choosing a meaningful distance threshold ε can be difficult.

Good:

- can detect arbitrary shapes,
- not very sensitive to noise,
- supports outlier detection,
- complexity is kind of okay,
- beside K-means the second most used clustering algorithm.

Question: what is Outliers? Outliers are often discarded as noise but some applications these noisy data can be more interesting than the more regularly occurring ones. why ?

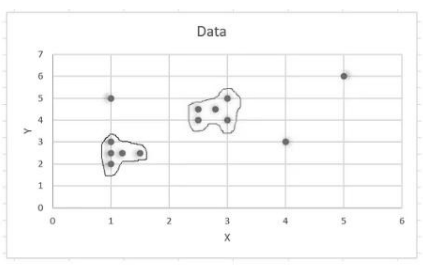
DBSCAN

- DBSCAN stands for Density-Based Spatial Clustering Application with Noise.
- It is an unsupervised machine learning algorithm that makes clusters based upon the density of the data points or how close the data is.
- That said, the points which are outside the dense regions are excluded and treated as noise or outliers.
- This characteristic of the DBSCAN algorithm makes it a perfect fit for outlier detection and making clusters of arbitrary shape.

- The DBSCAN algorithm takes two input parameters.
- Radius around each point (eps) and the minimum number of data points that should be around that point within that radius (MinPts).

DBSCAN

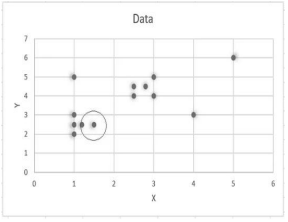
X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3



Logic and Steps:

- For example, consider the point (1.5,2.5),
- if we take eps = 0.3, then the circle around the point with radius = 0.3, will contain only one other point inside it (1.2,2.5)

X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3



Algorithm in action

- Let's choose eps = 0.6 and MinPts = 4.
- Let's consider the first data point in the dataset (1,2) & calculate its distance from every other data point in the data set.

X	Y	Distance from (1,2)
1	2	0
3	4	2.8
2.5	4	2.5
1.5	2.5	0.7
3	5	3.6
2.8	4.5	3.08
2.5	4.5	2.9
1.2	2.5	0.53
1	3	1
1	5	3
1	2.5	0.5
5	6	5.6
4	3	3.1

- As evident from the above table the point (1, 2) has only two other points in its neighbourhood (1, 2.5), (1.2, 2.5) for the assumed value of eps, as its less than MinPts, we can't declare it as a **core point**.

X	Y	Distance from (1,2)
1	2	0
3	4	2.8
2.5	4	2.5
1.5	2.5	0.7
3	5	3.6
2.8	4.5	3.08
2.5	4.5	2.9
1.2	2.5	0.53
1	3	1
1	5	3
1	2.5	0.5
5	6	5.6
4	3	3.1

- Let's repeat the above process for every point in the dataset and find out the neighbourhood of each. The calculations when repeated can be summarized

Point	Neighbourhood Points				
(1,2)	(1.2, 2.5)	(1, 2.5)			
(3, 4)	(2.5, 4)	(2.8, 4.5)			
(2.5, 4)	(3, 4)	(2.8, 4.5)	(2.5, 4.5)		
(1.5, 2.5)	(1.2, 2.5)	(1, 2.5)			
(3, 5)	(2.8, 4.5)				
(2.8, 4.5)	(3, 4)	(2.5, 4)	(3, 5)	(2.5, 4.5)	Cluster 1
(2.5, 4.5)	(2.5, 4)	(2.8, 4.5)			
(1.2, 2.5)	**(1, 2)**	(1.5, 2.5)	(1, 3)	(1, 2.5)	Cluster 2
(1, 3)	(1.2, 2.5)	(1, 2.5)			
(1, 5)					
(1, 2.5)	**(1, 2)**	(1.5, 2.5)	(1.2, 2.5)	(1, 3)	Cluster 2
(5, 6)					
(4, 3)					

- There are three points in the data set, (2.8, 4.5) (1.2, 2.5) (1, 2.5) that have 4 neighbourhood points around them, hence they would be called core points.
- Hence, (2.8, 4.5) is assigned to a new cluster, Cluster 1 and so is the point (1.2, 2.5), Cluster 2. Also observe that the core points (1.2, 2.5) and (1, 2.5) share at least one common neighbourhood point (1,2) so, they are assigned to the same cluster

Point	Neighbourhood Points					
(1,2)	(1.2, 2.5)	(1, 2.5)			Border Point	
(3, 4)	(2.5, 4)	(2.8, 4.5)			Border Point	
(2.5, 4)	(3, 4)	(2.8, 4.5)	(2.5, 4.5)		Border Point	
(1.5, 2.5)	(1.2, 2.5)	(1, 2.5)			Border Point	
(3, 5)	(2.8, 4.5)				Border Point	
(2.8, 4.5)	(3, 4)	(2.5, 4)	(3, 5)	(2.5, 4.5)	**Core Point**	Cluster 1
(2.5, 4.5)	(2.5, 4)	(2.8, 4.5)			Border Point	
(1.2, 2.5)	**(1, 2)**	(1.5, 2.5)	(1, 3)	(1, 2.5)	**Core Point**	Cluster 2
(1, 3)	(1.2, 2.5)	(1, 2.5)			Border Point	
(1, 5)					Outlier	
(1, 2.5)	**(1, 2)**	(1.5, 2.5)	(1.2, 2.5)	(1, 3)	**Core Point**	Cluster 2
(5, 6)					Outlier	
(4, 3)					Outlier	

- There are three types of points in the dataset as detected by the DBSCAN algorithm, core, border and outliers.

Cluster 1	Cluster 2	Outliers
(3,4)	(1, 2)	(1, 5)
(2.5, 4)	(1.5, 2.5)	(5, 6)
(3,5)	(1.2, 2.5)	(4, 3)
(2.8, 4.5)	(1, 3)	
(2.5, 4.5)	(1, 2.5)	