Clustering

Density-based Clustering [DBSCAN (Density-Based Spatial Clustering of Applications with Noise)]

Presented by: Eng. Doaa Fahmy

Today Outline

Clustering Algorithms.

DBSCAN Algorithm.

What ‘s the DBSCAN meaning ?

Advantages of DBSCAN.

Applications of DBSCAN

How the DBSCAN Algorithm work ?

Common types of clustering algorithms

Centroid-based Clustering:
These algorithms represent each cluster by a central point (centroid).
K-Means is a prominent example, where data points are assigned to the closest centroid, and centroids are iteratively updated based on the mean of their assigned points.

Hierarchical Clustering:
These algorithms build a hierarchy of clusters. Agglomerative hierarchical clustering starts with individual data points as clusters and iteratively merges the closest clusters, while divisive hierarchical clustering begins with all data points in one cluster and recursively splits them.

Density-based Clustering:
These algorithms identify clusters as high-density regions separated by low-density regions. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a well-known example that can discover clusters of arbitrary shapes and identify outliers (noise).

Distribution-based Clustering:

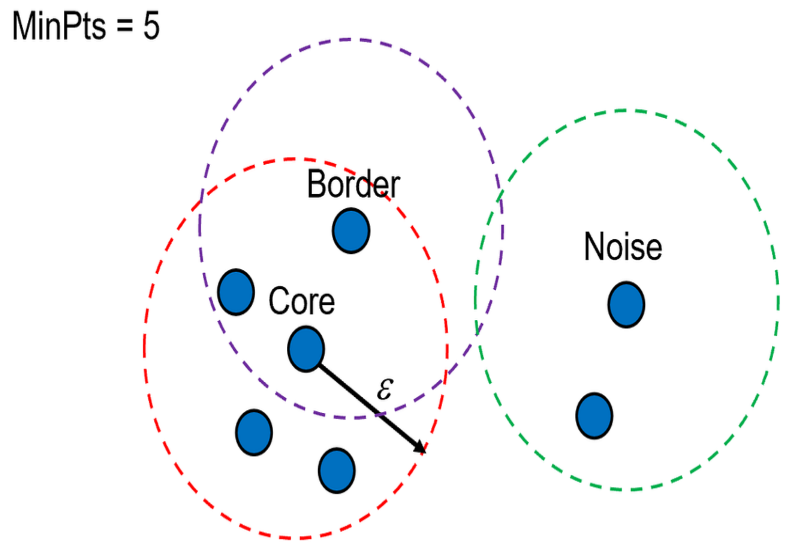
These algorithms model clusters as statistical distributions, often assuming that data points within a cluster belong to a specific probability distribution.
Gaussian Mixture Models (GMM) are a common example, where clusters are modeled as a mixture of Gaussian distributions.

Graph-based Models:

These models represent data points as nodes in a graph, and clusters are identified based on connectivity within the graph.
Spectral clustering is an example that uses the eigenvalues of a similarity matrix to perform dimensionality reduction before clustering.

DBSCAN

is a density-based clustering algorithm that groups data points that are closely packed together and marks outliers as noise based on their density in the feature space.
It identifies clusters as dense regions in the data space separated by areas of lower density.



DBSCAN algorithm creates a circle of *epsilon* radius around every data point and classifies them into **Core** point, **Border** point, and **Noise**. A data point is a **Core** point if the circle around it contains at least ‘*minPoints*’ number of points. If the number of points is less than *minPoints*, then it is classified as **Border** Point, and if there are no other data points around any data point within *epsilon* radius, then it treated as **Noise**.

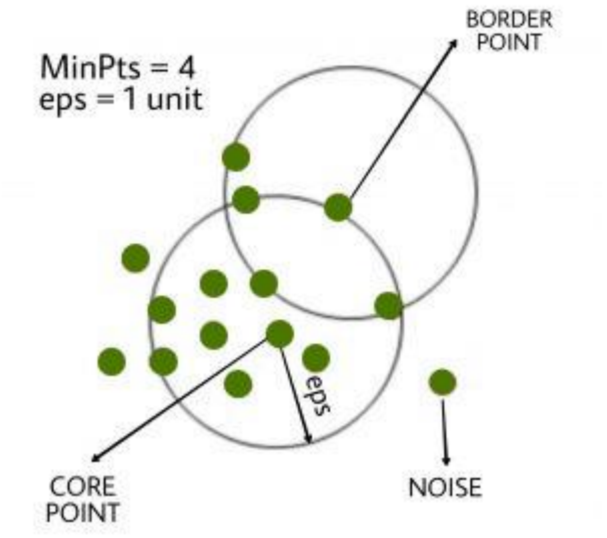
How Does DBSCAN Work?

DBSCAN works by categorizing data points into three types:

Core points : which have a sufficient number of neighbors within a specified radius (eplison).

Border points : which are near core points but lack enough neighbors to be core points themselves.

Noise points : which do not belong to any cluster.



Hyperparameter Tuning (Choosing value of ‘ε’ & min. no. of points)

1- Epsilon (ϵ):
This defines the radius of the neighborhood around a data point. If the distance between two points is less than or equal to **eps** they are considered neighbors

- If **eps** is too small most points will be classified as noise.
- If **eps** is too large clusters may merge, and the algorithm may fail to distinguish between them.

2. MinPts: This is the minimum number of points required within the **eps** radius to form a dense region.
A general rule of thumb is to set MinPts >= D+1 where **D** is the number of dimensions in the dataset.

Advantages of DBSCAN:

- **No need to specify number of clusters:**
  DBSCAN automatically determines the number of clusters.

- **Handles noise and outliers:**
  Outliers are identified as noise points.

- **Discovers clusters of arbitrary shapes :**
  Unlike K-Means, DBSCAN can identify clusters with irregular shape.

Drawbacks of DBSCAN

Difficulty with Varying Densities: DBSCAN may struggle with datasets containing clusters of significantly different densities, as finding an appropriate “MinPts-ε” combination becomes challenging for all clusters.

Curse of Dimensionality:
The Euclidean distance measure, when applied to high-dimensional data, may become less effective due to the curse of dimensionality

Here's a breakdown of DBSCAN's applications

Fraud detection
In financial transactions, DBSCAN can identify outliers or unusual patterns that may indicate fraudulent activity.

Market research
DBSCAN can be used to segment customers based on their purchasing patterns and behaviors.

Earthquake monitoring
DBSCAN can help identify major fault lines by clustering seismic activities and filtering out minor tremor

Urban planning and transportation

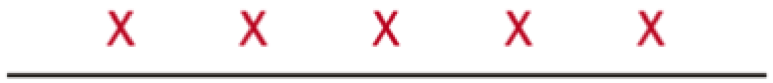
Bioinformatics:
It can be used to analyze gene expression data and identify groups of genes with similar expression patterns

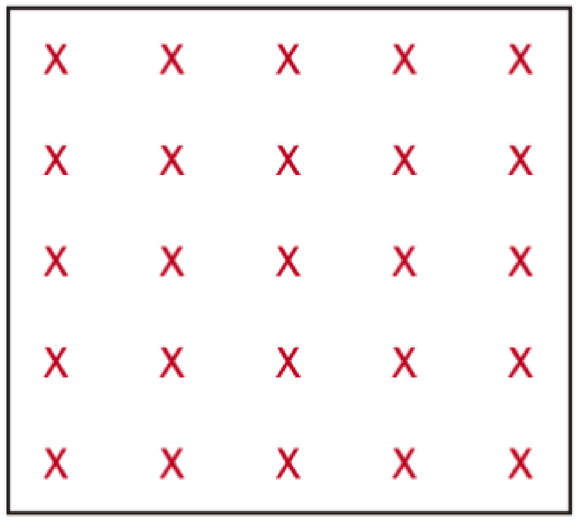
Image segmentation:
DBSCAN can group pixels with similar properties into regions, allowing for image segmentation and object recognition

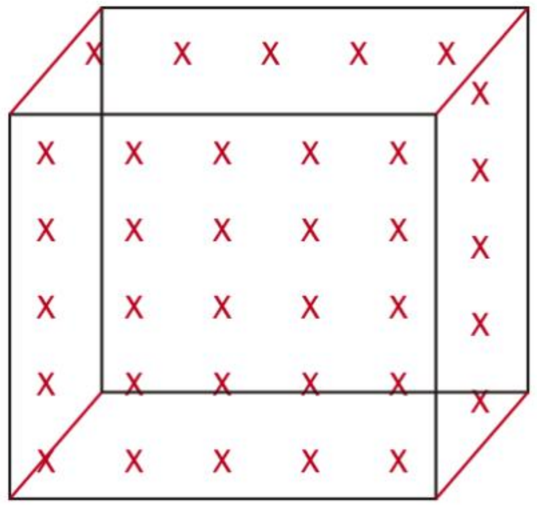
What Is The Curse of Dimensionality?

High Dimensional Data

High dimensional data is when a dataset a number of features (p) that is bigger than the number of observations (N).
 High dimensional data is the problem that leads to the curse of dimensionality.
The equation for high dimensional data is usually written like p >> N.

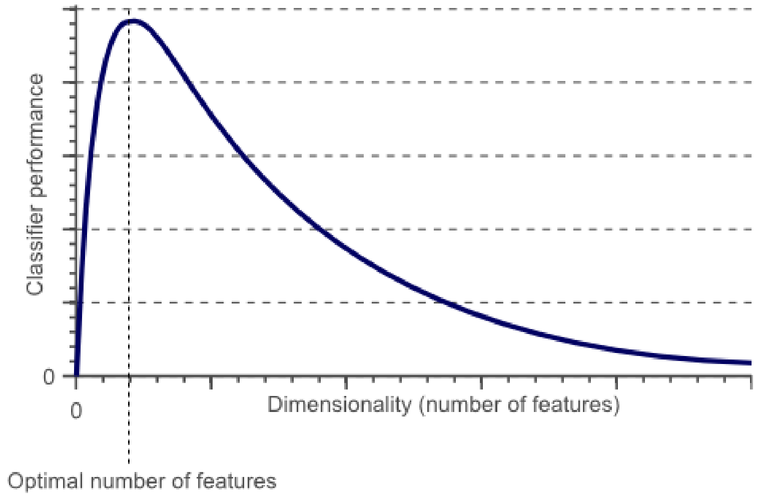

A one-dimensional features space with five data points


A two-dimensional features space with 25 data points


A three-dimensional features space with 125 data points

Hughes Phenomenon

he Hughes Phenomenon shows that as the number of features increases,
the classifier’s performance increases as well until we reach the optimal number of features.
Adding more features based on the same size as the training set will then degrade the classifier’s performance.



Curse of Dimensionality in Distance Function

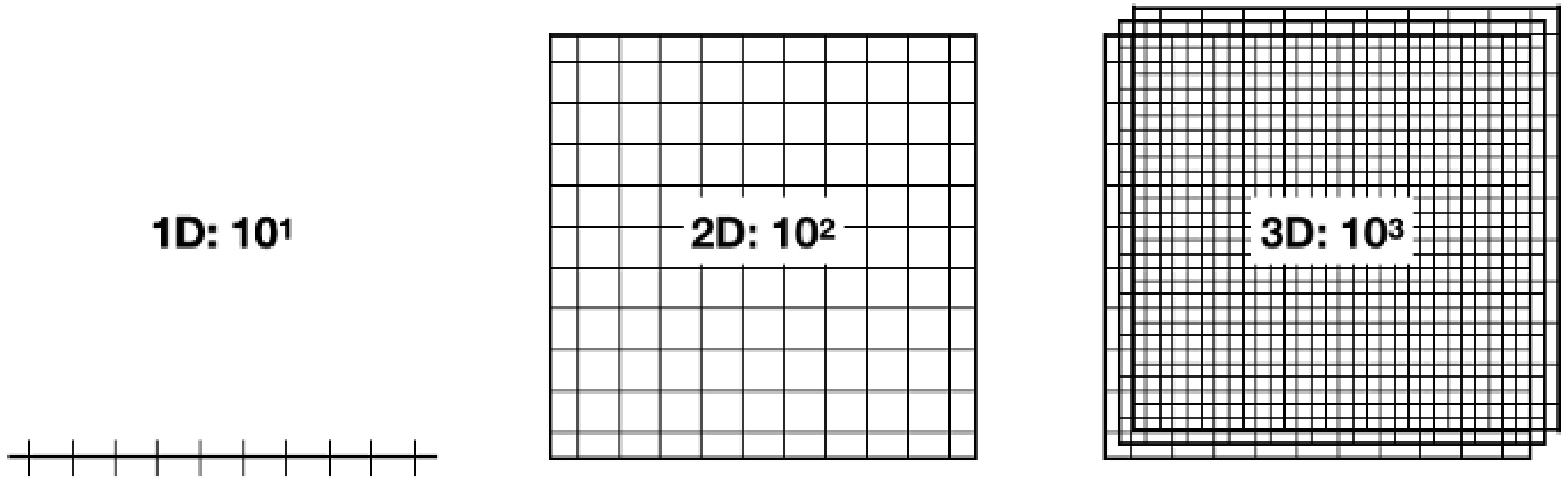
An increase in the number of dimensions of a dataset means there are more entries in the vector of features that represents each observation in the corresponding Euclidean space. We measure the distance in a vector space using Euclidean distance.

The Euclidean distance between 2-dimensional vectors with Cartesian coordinates *p = (p1, p2, …, pn)* and *q = (q1, q2, …, qn)* is computed using the familiar formula:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

with the number of dimensions for distinct vectors. In other words, as the number of features grows for a given number of observations, the feature space becomes increasingly sparse; that is, less dense or emptier. On the flip side, the lower data density requires more observations to keep the average distance between data points the same.

Below figure shows how many data points we need to maintain the average distance of 10 observations uniformly distributed on a line. It increases exponentially from 10^1 in a single dimension to 10^2 in two and 10^3 in three dimensions, as the data needs to expand by a factor of 10 each time we add a new dimension:



The number of features required to keep average distance constant grows exponentially with the number of dimensions.

The number of possible unique rows grows exponentially as the number of features increases, which makes it so much harder to efficiently generalize.
The variance increases as they get more opportunity to overfit to noise in more dimensions, resulting in poor generalization performance.

When Should We Use DBSCAN Over K-Means Clustering?

DBSCAN	K-Means
• In DBSCAN we need not specify the number of clusters.	• It is very sensitive to the number of clusters so it need to specified
• Clusters formed in DBSCAN can be of any arbitrary shape.	• Clusters formed are spherical or convex in shape.
• It can work well with datasets having noise and outliers.	• It does not work well with outliers' data. Outliers can skew the clusters in K-Means to a very large extent.
• In DBSCAN two parameters are required for training the Model	• In K-Means only one parameter is required is for training the model

References

Hands-on Machine Learning, Aurelien Geron

Introduction to Machine Learning with Python, Andreas C. Muller

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/