Tutorial

Clustering Techniques K-Means vs. DBSCAN

Introduction

A key unsupervised learning method is clustering, which groups data based on shared characteristics to reveal hidden patterns in unlabelled datasets. One of its many uses include market segmentation, image processing, etc. The methodologies and applications of K-Means and DBSCAN make them main clustering techniques. DBSCAN is a density-based method that works well with noisy datasets and can detect any kind of cluster. On the other hand, K-Means is well recognised for being easy to use and successful with properly separated circular clusters.[3]

This tutorial discusses the parameters, how these two algorithms work, and possible uses for them. To create clusters, K-Means divides the data by minimising the difference between each cluster. Techniques like the elbow method are used to find the optimal number of clusters. DBSCAN instead uses density measures (epsilon and minimum points) to characterise clusters. While this expands your options, you need to be careful while modifying the settings. Using hypothetical datasets such as "moons" or "blobs" this tutorial demonstrate method to handle structured and noisy data. Because of this, users may intelligently classify work in different circumstances.

What is K-Means?

A dataset is divided into k clusters using the centroid-based clustering method K-Means, which allocates every data point within the cluster with the closest centroid. In order to guarantee compact, distinct clusters, the goal is to minimise the within-cluster sum of squares (WCSS). [1].

Mathematical Intuition:

K-Means minimizes the following WCSS objective function:

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} \left|\left| x - \mu_i \right|\right|^2$$

Here:

C_i : the set of data points in cluster i.

μ_i : the centroid (mean) of cluster i.

$||x - \mu_i||^2$: the squared Euclidean distance between a data point x and its cluster centroid μ_i.

K-Means efficiently creates compact, unique clusters by minimising WCSS, which guarantees that data units are as near to their associated centroids as possible.

Steps

The k-means method starts by initialising the k cluster centroids at random. After that, every data point is paired with the closest centroid to create clusters. Recalculating the centroids as the average of the points allocated to each cluster updates them. These steps are performed repeatedly, till the centroids are stabilised or a certain number of iterations are achieved.

Parameter Selection

The only main parameter in K-Means is the number of clusters, k. It is usually selected with the help of techniques such as silhouette analysis and the elbow approach. They assess clustering quality by estimating WCSS or inter-cluster distances.[4]

Edge Cases

Approximately equal-sized spherical clusters are assumed by K-Means. Clusters that overlap or have unusual shapes are frequently divided or merged erroneously. It is also susceptible to outliers, that can cause cluster centroids to be distorted.

Visual Impact of Parameters

Over-clustering (too many little clusters) and under-clustering (blending distinct clusters) can occur from improperly selecting k. The best way to choose parameters is to visualise WCSS plots (elbow approach) or silhouette scores.

Computational Complexity and Optimizations:

K-Means has a time complexity of O(n×k×i), where n is the number of data points, k is the number of clusters, and i is the number of iterations. By improving initialisation, optimisations like K-Means++ lower the number of iterations required for convergence. In high-dimensional data, dimensionality reduction methods like PCA can also increase efficiency.[2]

Understanding DBSCAN

What is DBSCAN?

A density-based technique called Density-Based Spatial Clustering of Applications with Noise also known as DBSCAN finds clusters by separating high-density data areas with lower-density ones. It is appropriate for datasets with unpredictable geometries and noise since it doesn't need a predetermined number of clusters.

Mathematical Intuition

DBSCAN uses the following criteria to categorise points according to density, the minimal number of points needed to produce a dense zone is minPts, and the neighbourhood radius surrounding a point is ϵ.[3]

Key Steps of DBSCAN

DBSCAN uses the following criteria to categorise points according to density, the minimal number of points needed to produce a dense zone is minPts, and the neighbourhood radius surrounding a point is ϵ. By locating core points with minimum minPts neighbours within a certain radius ϵ, DBSCAN clusters data. While locations that cannot be reached from any core point are classified as noise, density-connected points are grouped into clusters.

Parameter Selection

In DBSCAN, the selection of minPts (minimum points) and ϵ (neighbourhood radius) is crucial. The "elbow point" on the plot of a k-distance graph, which is frequently used to calculate ϵ, shows a suitable value. The dimensionality of the data usually determines the minPts parameter; a popular heuristic is minPts > d+1, in which d is the number of dimensions.

Edge Cases

Since just one ϵ value could not be effective for all clusters, DBSCAN has trouble with datasets that contain clusters of different densities. While excessive ϵ values may wrongly fuse discrete clusters, little ϵ values may produce a lot of noise point.

Visual Impact of Parameters

Clustering results are strongly impacted by varying ϵ and minPts. For example, a big ϵ with low minPts could over-cluster the data, whereas a small ϵ with high minPts create moderate clusters with more noise.

Computational Complexity and Optimizations

Effective spatial indexing structures such as k-d trees or R-trees can lower the temporal complexity of DBSCAN from O(n2) for naïve implementations to O (n log n). DBSCAN can expand to bigger datasets while still being successful because to these optimisations.[3]

Comparing K-Means and DBSCAN

Comparison Table

Feature	K-Means	DBSCAN
Clustering Approach	Centroid-based	Density-based
Cluster Shape	Spherical	Arbitrary
Noise Handling	Poor	Excellent
Parameter Selection	Number of clusters (k)	Epsilon (ε), MinPts
Scalability	Scalable to large datasets	Less scalable to large datasets
Outlier Sensitivity	High	Low

K-Means and DBSCAN are two most used clustering techniques, each offering unique benefits. K-Means is ideal for large datasets due to their uniform distribution and spherical clusters. Nonetheless, it faces challenges with noise in the data and the cluster having unusual shapes. In contrast, DBSCAN excels in handling noisy data and identifying clusters of various shapes, making it appropriate for complex datasets, including crescent-shaped clusters. Hybrid approaches can use the benefits of both techniques. The choice of approach depends on the dataset characteristics and the specific requirements of the problem.

Implementation of K-Means and DBSCAN

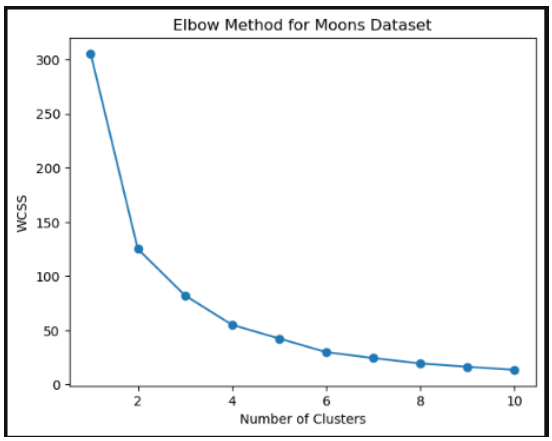
Dataset

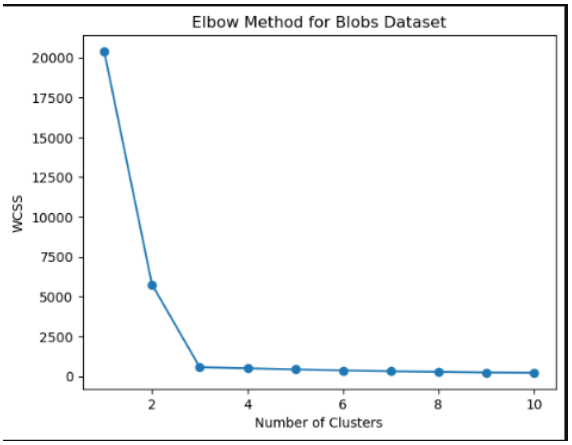
Synthetic Moons and Blobs

We illustrate the clustering behavior of K-Means & DBSCAN in noisy settings using artificial datasets (blobs and moons). There are three spherical clusters with different standard deviations in the blobs dataset, and 2 crescent-shaped clusters with additional noise in the moons dataset.

Elbow Method for K-Means

The ideal number of clusters in K-Means is found using the elbow approach. The 'elbow point' is the point at which the rate of reduction considerably slows down, as determined by analyzing WCSS versus the number of clusters. This provides the dataset's ideal number of clusters.



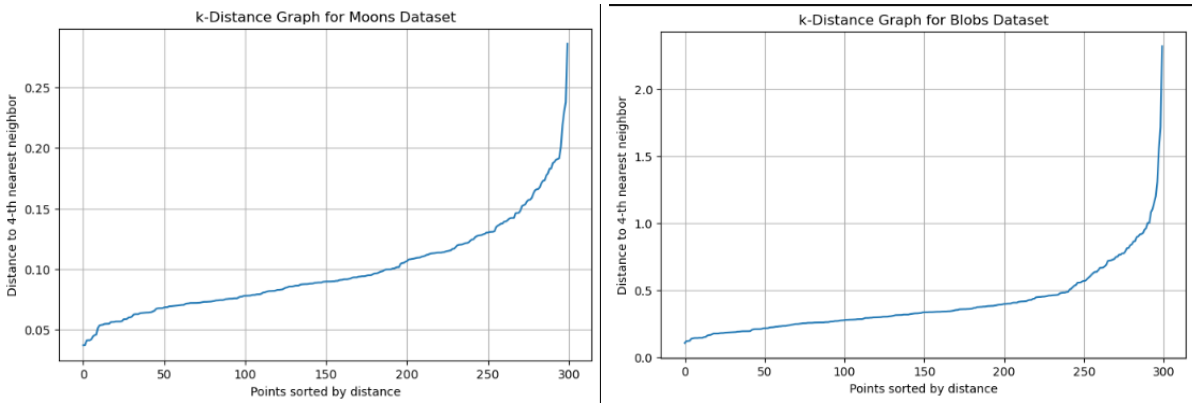


K-Means Implementation

The elbow technique indicates that two clusters are optimal in the moon dataset and three for the blobs dataset. The instances of the randomizer have been fixed using random state. These parameters are used to apply K-Means clustering, and the outcomes are displayed.

k-distance Method

The k-distance approach plots the distances from to the k-th nearest neighbour (k=minPts−1) in an ascending sequence to assist find the ideal ϵ for DBSCAN. The graph's "elbow point" denotes an appropriate ϵ, guaranteeing distinct clusters and efficient noise reduction.



DBSCAN Implementation

Both datasets are subjected to DBSCAN using adjusted variables (epsilon and min_samples). Epsilon is determined to be 0.2 for the moons dataset & 1.0 for the blobs dataset using the K-distance approach. For both datasets, a minimum of 5 samples is used. These parameters are used to implement DBSCAN, and the outcomes are displayed.

Evaluation with Silhouette Scores

SThe clustering performance of both methods was assessed using silhouette scores. Better-defined clusters are indicated by a higher silhouette score. The scores are shown below:

Silhouette Score (K-Means on Moons): 0.49
Silhouette Score (DBSCAN on Moons): 0.65
Silhouette Score (K-Means on Blobs): 0.78
Silhouette Score (DBSCAN on Blobs): 0.72

Analysis of Results: K-Means and DBSCAN

Moons Dataset

Below are the observations for K-Means and DBSCAN:

K-Means

As K-Means assumes spherical cluster forms, it tends to divide linked clusters. The crescent forms from the moons dataset are mistakenly divided into many pieces. The algorithm's efficiency for this set of data is further reduced by its sensitivity to noise & overlapping areas.

DBSCAN

DBSCAN demonstrates its proficiency in managing non-linear cluster borders by successfully identifying the two crescent-shaped clusters. Additionally, it exhibits resilience in noisy settings by effectively detecting noise position.

Blobs Dataset

Here are the observations for K-Means and DBSCAN:

K-Means

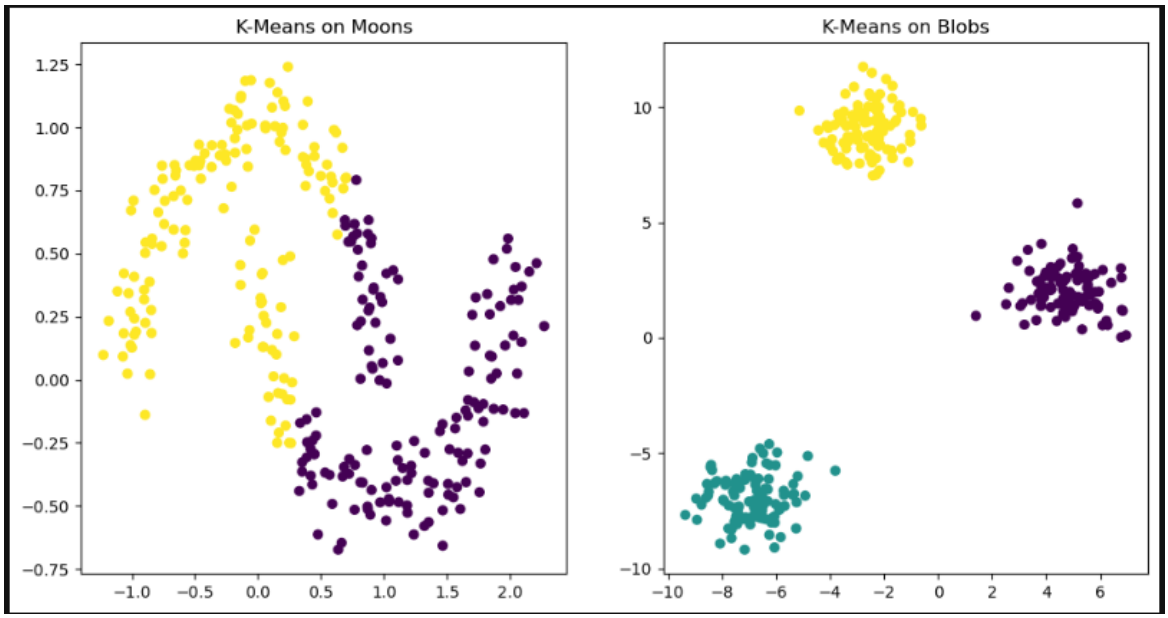
On this dataset, K-Means works particularly well because of the spherical cluster morphologies. The data is effectively and with few mistakes divided into discrete clusters. Due to the algorithm's sensitivity to initialisation, poor clustering may occasionally result.

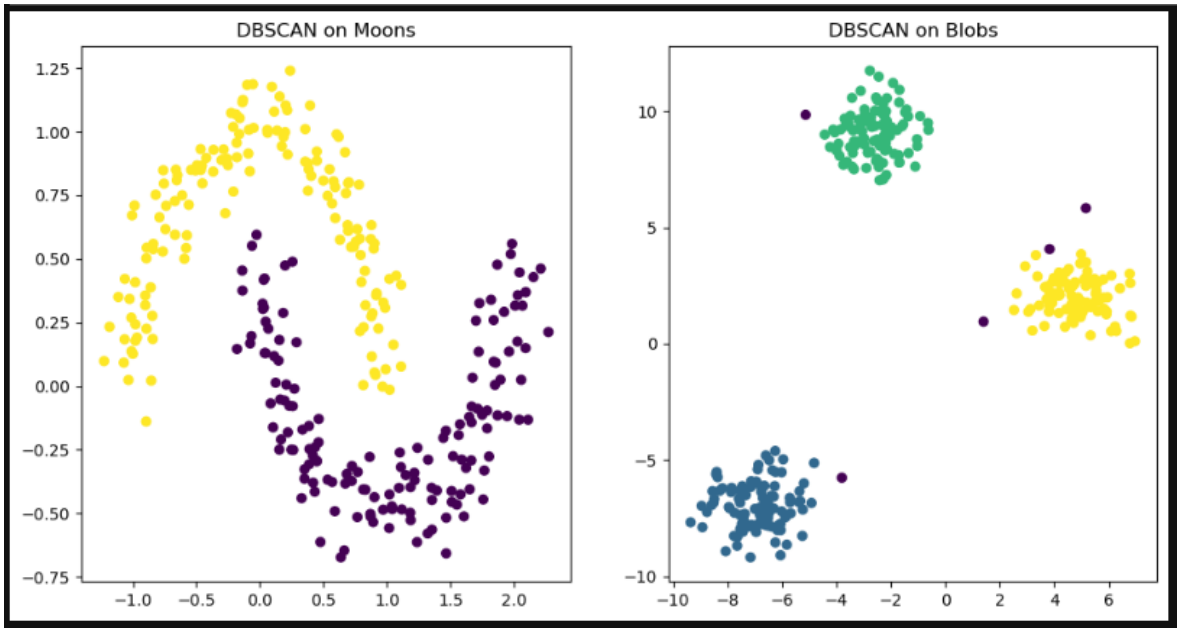
DBSCAN

It is necessary to carefully adjust the epsilon value (ε) to correlate with the blob densities for DBSCAN to cluster each blob properly. Certain locations close to the cluster borders could be mistakenly categorised as noise, particularly if there are large differences in density across clusters.

Graphical Comparisons

For the moons & blobs datasets, the clustering outputs for both techniques are shown in the following visualisations:





Strengths and Weaknesses

K-Means

Large datasets with spherically distributed well-separated clusters are best suited for the effective clustering technique K-Means. Applications like where documents are categorised and market segmentation frequently use it. As it is not able to handle noise or overlapping clusters or irregular form of data, as well as its requirement to predetermine the number of clusters, makes it not useful for complicated datasets.[2]

DBSCAN

DBSCAN is a very reliable technique. It can handle noisy datasets and can form clusters of various shapes. It performs well in jobs such as anomaly detection and geospatial analysis. Unlike K-Means, it provides flexibility for complex data, and it also doesn't demand a predetermined number of clusters. However, it is less effective in big datasets having different densities and those who are sensitive to parameter selection.[3]

Conclusion

The advantages and disadvantages of K-Means and DBSCAN, two well-known clustering methods, have been emphasised in this tutorial. Applications like separating markets and document clustering can benefit from K-Means' efficiency and suitability for spherically distributed, well-separated clusters. It has trouble detecting irregular forms and noise, though. On the other hand, DBSCAN is well appropriate for spatial evaluation and anomaly detection as it performs well in data that is noisy and can find clusters of any shapes. It still has issues with computational inefficiency with big datasets and sensitivity to parameter selection.

New developments in clustering algorithms overcome earlier drawbacks including non-linear separations and different data density. HDBSCAN uses hierarchical density-based clustering to improve DBSCAN, while spectral clustering uses graph theory to handle high-dimensional data and non-linear separations. By using these techniques, clustering may be used to more complicated, large-scale issues. Future studies might test sophisticated methods like Gaussian mixture models, spectral clustering, and HDBSCAN on a variety of datasets, including time-series data and Iris. Furthermore, combining clustering with deep learning or dimensionality reduction may help address increasingly difficult data science problems.

References

1. MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*.

2. Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*.

3. Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

4. Scikit-learn Documentation. (2023). Clustering Algorithms. Available at: https://scikit-learn.org