Machine Learning

Clustering Algorithms

K-Means Clustering

K-Means clustering is one of the most widely used clustering algorithms in machine learning. It groups data points into a predefined number of clusters based on their features. The key idea is to assign data points to the nearest cluster centroid and then update the centroids iteratively until convergence.

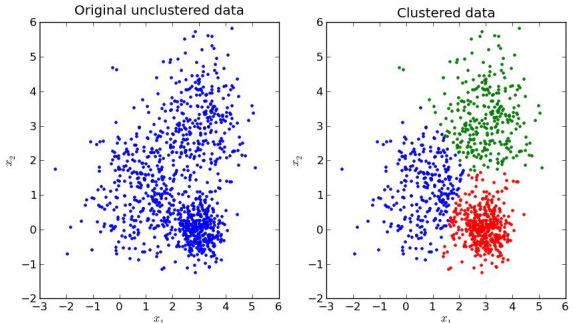
How its work:

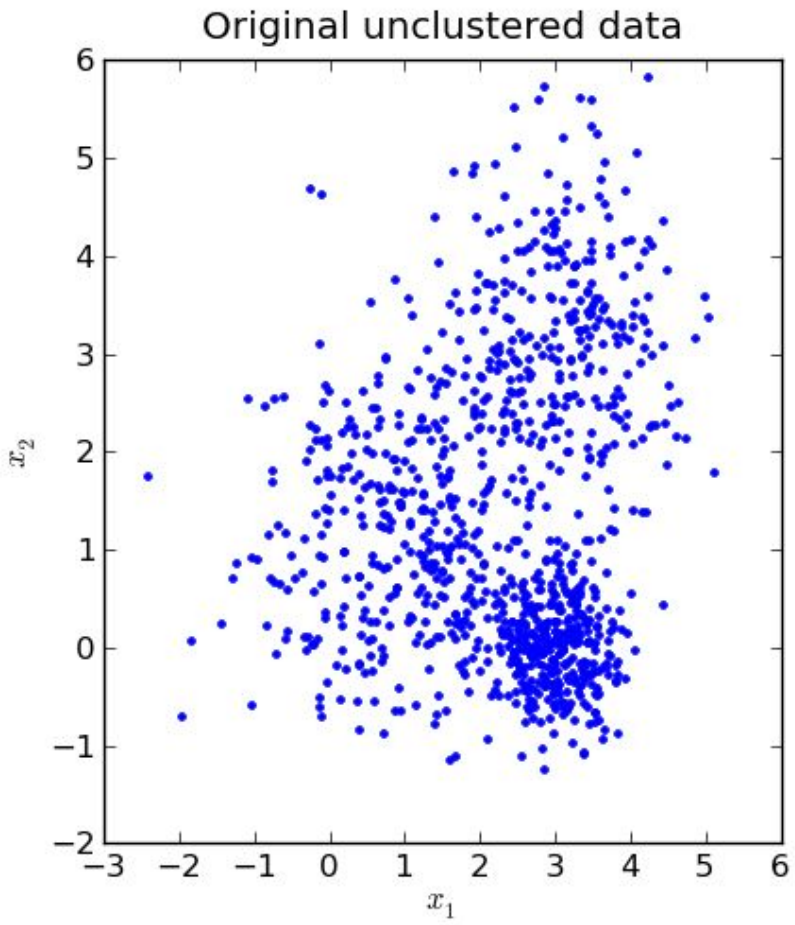
Choose the number of clusters,Randomly initialize the positions of the cluster centroids.

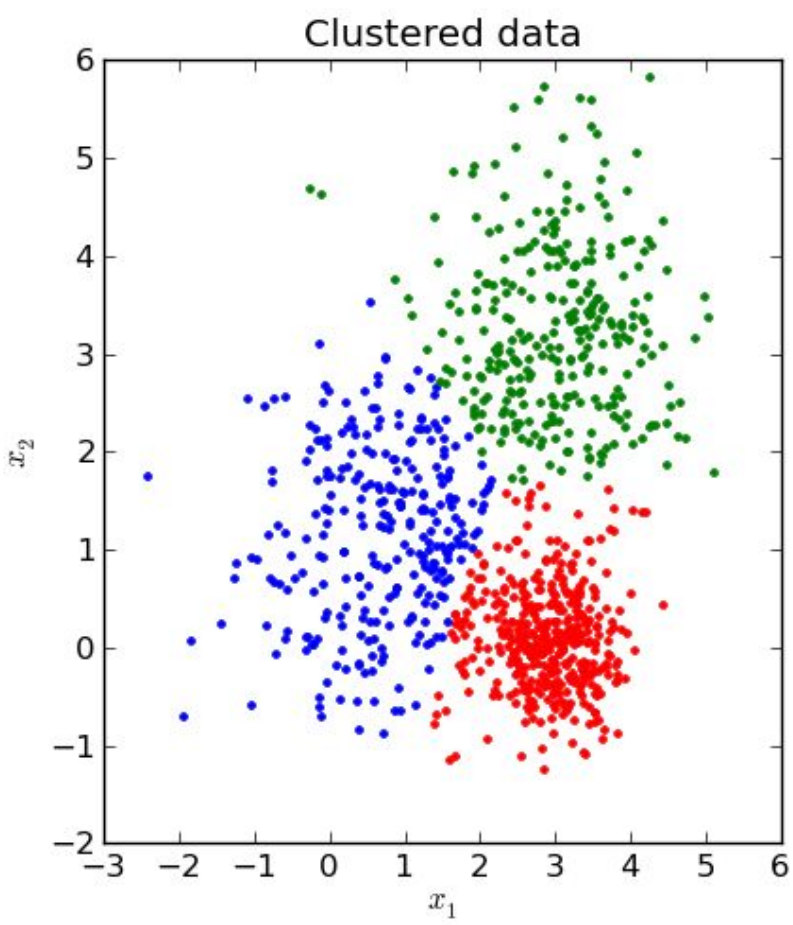
Assign each data point to the nearest centroid.For each data point, calculate the distance (usually Euclidean distance) to each of the centroids.this form a clusters

For each cluster, recalculate the centroid as the mean of all data points assigned to that cluster.The centroid is updated to the new mean position of the cluster

Repeat the process until convergence.




Original unclustered data
x_2
x_1
6
5
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4
5
6


Clustered data
x_2
x_1
6
5
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4
5
6

Advantages of K-Means:

1. **Easy to Implement**: Simple to understand and implement.
2. **Scalable**: Works well with large datasets.
3. **Efficient**: Relatively fast, especially with small numbers of clusters and features.
4. **Interpretability**: The results are intuitive, and the clusters are easy to interpret.

Disadvantages:

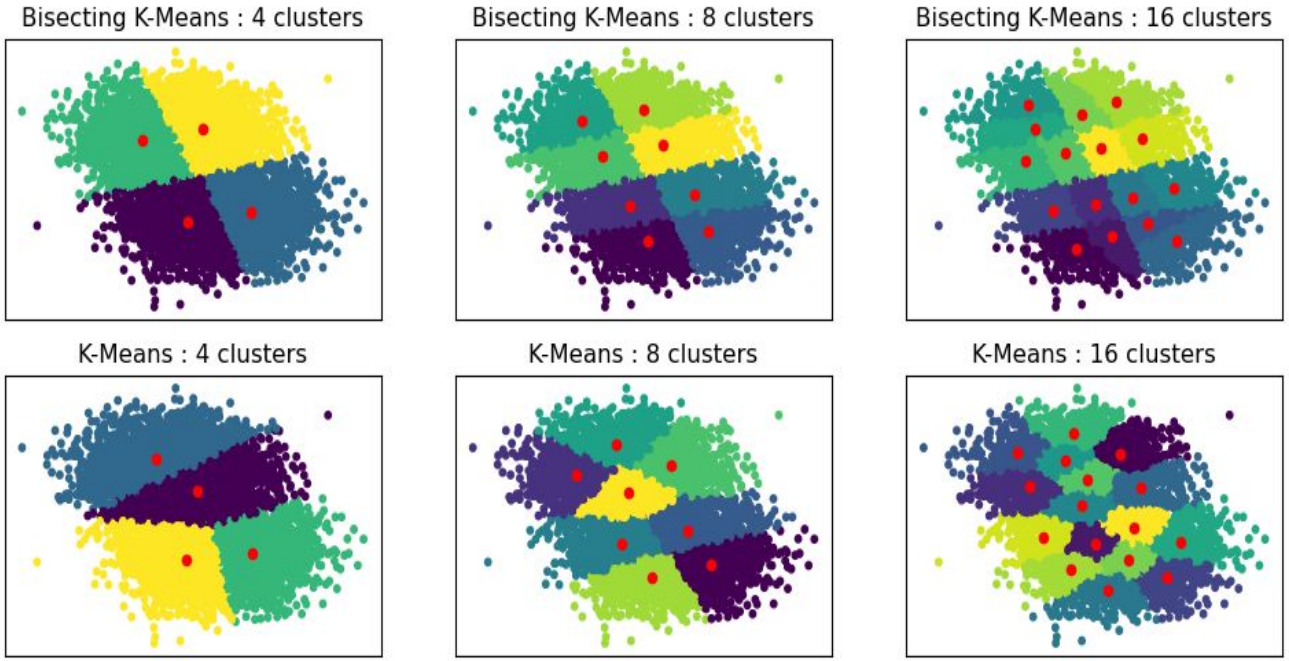
1. **Choosing K**: The number of clusters, **K**, needs to be predefined. This can be tricky and may require experimentation or methods like the **Elbow Method**.
2. **Sensitive to Initial Centroids**: The final clusters can vary depending on how the initial centroids are chosen. The algorithm can converge to a local minimum.
3. **Assumes Spherical Clusters**: K-Means works well only if the clusters are roughly spherical and evenly distributed.
4. **Sensitive to Outliers**: Outliers can significantly affect the centroids and the clustering result.

Bisecting K-Means

Bisecting K-Means is a variation of the standard **K-Means** clustering algorithm that operates by iteratively splitting clusters into two (bisecting) until the desired number of clusters is reached. It combines hierarchical clustering with K-Means and is particularly useful when you want to break the data into clusters in a more controlled manner.

How Bisecting K-Means Works:

1. **Start with All Data in One Cluster**:
   - Initially, all data points are treated as one big cluster.
2. **Bisect (Split) a Cluster**:
   - The cluster is split into two smaller clusters using the standard **K-Means** algorithm. This step involves dividing the cluster into two by minimizing the within-cluster variance (like standard K-Means).
3. **Choose the Best Split**:
   - If multiple splits are tried, the split that results in the least within-cluster variance is chosen.
4. **Repeat the Process**:
   - This process of bisecting clusters is repeated on one of the existing clusters until the desired number of clusters is reached. At each step, the cluster with the largest variance (or the one that is least compact) is chosen for bisecting.
5. **Stop When Desired Clusters are Formed**:
   - The algorithm continues bisecting clusters until the desired number of clusters is achieved.


Bisecting K-Means : 4 clusters
Bisecting K-Means : 8 clusters
Bisecting K-Means : 16 clusters
K-Means : 4 clusters
K-Means : 8 clusters
K-Means : 16 clusters

Advantages of Bisecting K-Means:

1. **Hierarchical and K-Means Hybrid**: Combines the strengths of hierarchical clustering and K-Means, allowing better control over the splitting process.
2. **Efficiency**: Faster than traditional hierarchical clustering because it avoids calculating all pairwise distances.
3. **Better for Certain Datasets**: Works well with large datasets, especially when clusters are well-separated and of different sizes.
4. **Scalable**: It's efficient for large datasets due to the bisecting approach.

Disadvantages of Bisecting K-Means:

1. **Cluster Shape**: Like K-Means, Bisecting K-Means still struggles with non-spherical clusters.
2. **Random Initialization**: The results can vary depending on the initial split, similar to the regular K-Means algorithm.
3. **Choosing the Cluster to Split**: Deciding which cluster to bisect next can affect the final clusters. Different heuristics (e.g., largest cluster) may lead to different results.

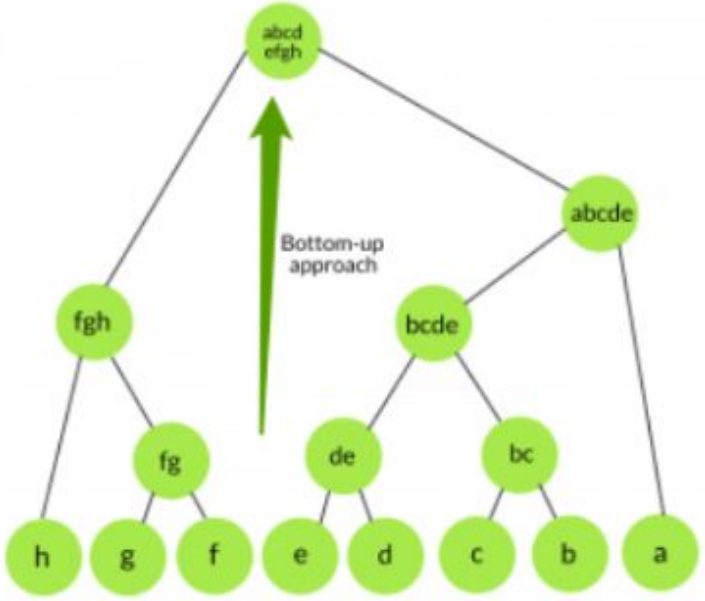
Agglomerative Clustering (Agglomerative Hierarchical Clustering)

Agglomerative Clustering, also known as **Agglomerative Hierarchical Clustering**, is a hierarchical clustering algorithm that builds a nested hierarchy of clusters by iteratively merging smaller clusters into larger ones. It's a **bottom-up approach**, where each data point starts as its own cluster, and clusters are progressively merged based on their similarity until all points belong to a single cluster or a specified number of clusters is reached..

How it works:

1. **Start with individual data points**: Each data point is initially considered as a single cluster.
2. **Calculate similarity (or distance)**: Compute the distance (similarity) between every pair of clusters. Common distance metrics
3. **Merge the closest clusters**: Combine the two clusters that are closest to each other based on the chosen distance metric.
4. **Update the distance matrix**: After merging, recalculate the distances between the new cluster and all remaining clusters.
5. **Repeat**: Continue merging the closest clusters and updating the distance matrix until all points are in a single cluster or until a stopping criterion (e.g., a predefined number of clusters) is met.

Agglomerative Clustering



Hierarchical agglomerative clustering

Advantages:

1. **Doesn’t require a predefined number of clusters**: Unlike K-Means, the number of clusters doesn’t need to be specified upfront. You can cut the dendrogram at different levels to obtain different numbers of clusters.
2. **Flexible distance metrics**: You can choose different distance metrics and linkage criteria depending on the structure of the data.
3. **Hierarchical nature**: Provides insight into the data at different levels of granularity. The dendrogram allows you to visualize the clustering process and explore different numbers of clusters.

Disadvantages:

1. **Computationally expensive**: The algorithm computes pairwise distances between all points or clusters, making it computationally intensive for large datasets.
2. **Sensitive to noise and outliers**: Outliers or noisy data points can distort the clustering structure.
3. **Irreversible merging**: Once two clusters are merged, they cannot be split again. This means early mistakes in the clustering process can affect the final result.

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

DBSCAN is a popular clustering algorithm that identifies clusters in data by finding regions of high density. It is particularly effective in discovering clusters of arbitrary shapes and dealing with noisy data. Unlike K-Means or Agglomerative Clustering, DBSCAN does not require the number of clusters to be specified beforehand and is able to classify outliers (noise).

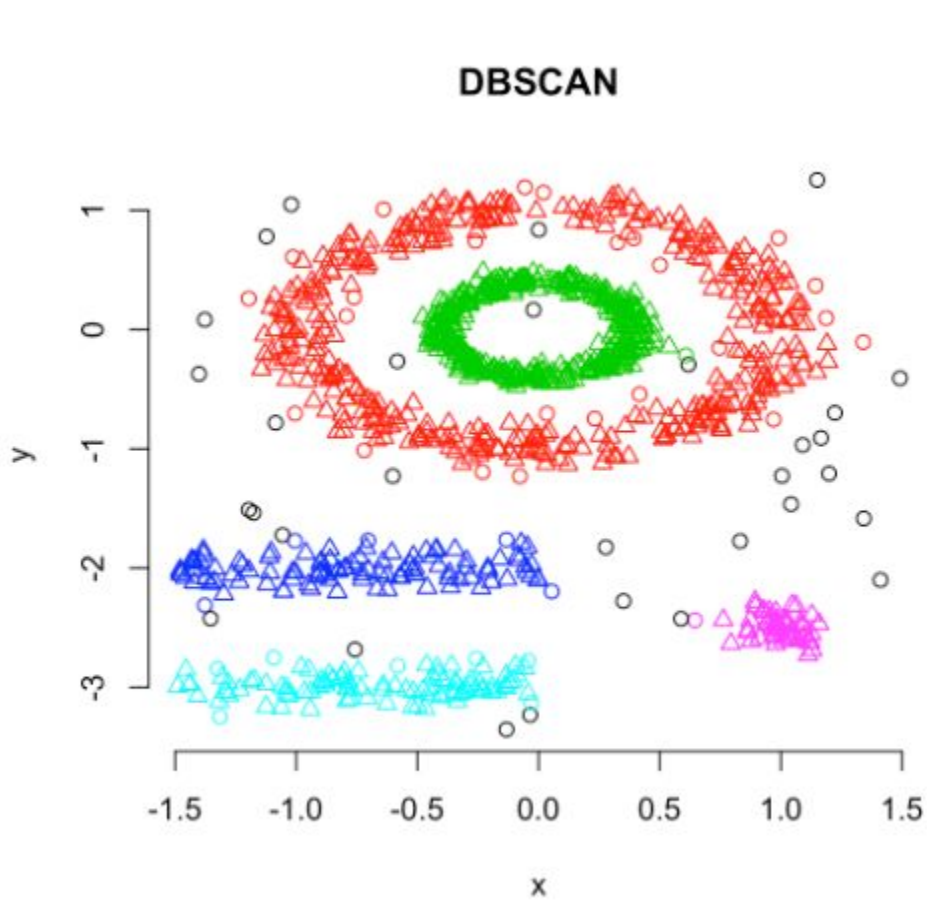
Step 1:For each point in the dataset, DBSCAN checks if it has at least **minimum** points within a distance of neighbor

- If yes, the point is labeled as a **core point** and a new cluster starts.
- If no, the point is labeled as noise or temporarily left unclustered.

Step 2: If a core point is found, DBSCAN expands the cluster by including all points within its neighborhood. It recursively applies the same process to any other core points in the neighborhood, expanding the cluster.

Step 3: DBSCAN stops expanding the cluster when no more core points can be reached.

Step 4: The process continues until all points have been visited, either as part of a cluster or as noise.


DBSCAN
y
1
0
-1
-2
-3
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
x

Advantages of DBSCAN:

1. **Detects clusters of arbitrary shapes**: Unlike K-Means, DBSCAN can identify clusters that are not spherical, including elongated and irregular shapes.
2. **No need to specify the number of clusters**: The algorithm automatically detects the number of clusters, making it more flexible for different types of data.
3. **Robust to noise and outliers**: DBSCAN explicitly labels points as noise when they don't belong to any cluster.
4. **Versatile**: Works well for datasets with clusters of varying sizes and densities.

Disadvantages of DBSCAN:

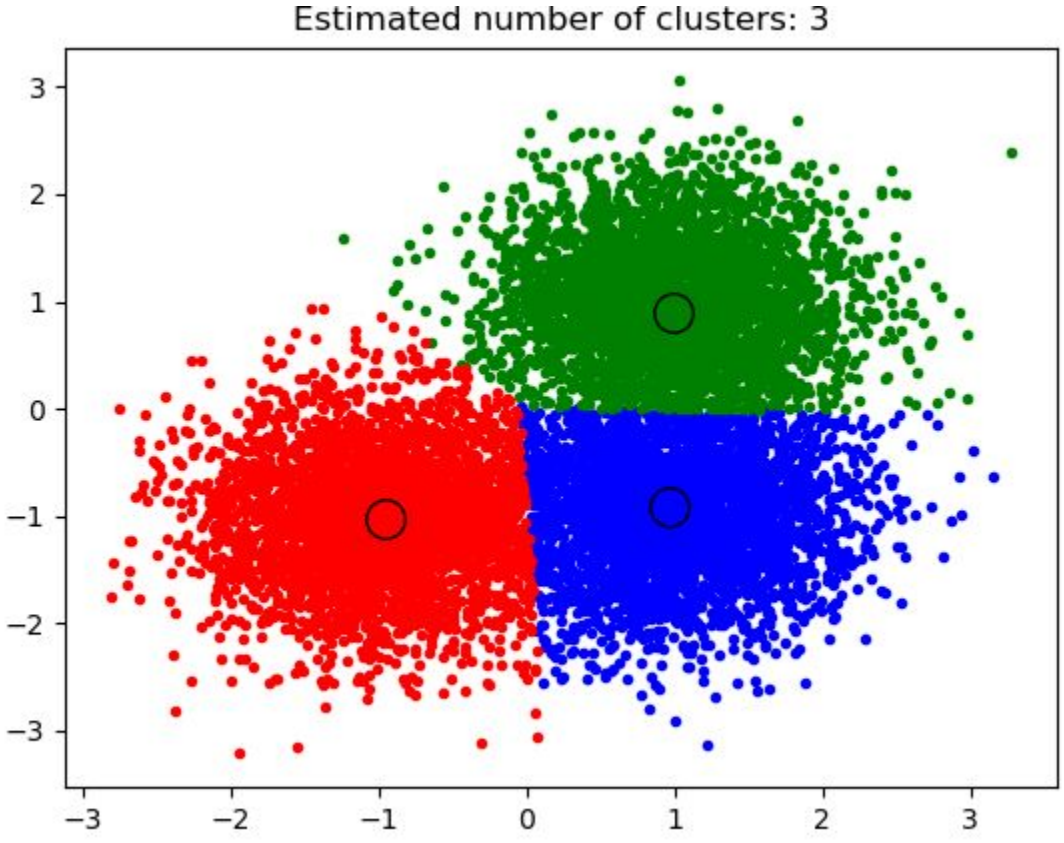
1. **Parameter sensitivity**: DBSCAN's performance is highly dependent on the choice of **ε** and **minPts**. Choosing inappropriate values can lead to poor clustering results.
    - **ε too small**: Many points will be labeled as noise.
    - **ε too large**: Clusters may merge, and the algorithm may fail to distinguish between dense regions.
2. **Struggles with varying densities**: DBSCAN may not perform well if the dataset has clusters with varying densities, as a single **ε** value may not suit all clusters.
3. **High-dimensional data**: In high-dimensional datasets, DBSCAN may struggle to find meaningful clusters due to the curse of dimensionality (the concept of distance becomes less meaningful as dimensionality increases).

Mean Shift Clustering

Mean Shift Clustering is a type of **centroid-based** clustering algorithm that seeks to find dense areas of data points, referred to as "modes," in a feature space. Unlike K-Means or DBSCAN, Mean Shift does not require the number of clusters to be specified beforehand and can automatically determine the number of clusters by finding dense regions in the data.

How Mean Shift Works:

1. **Initialize**: Each point in the dataset is considered as for the centroid of a cluster. Each point starts as its own cluster.
2. **Shift Each Point**: For each point, the algorithm computes the **mean** of the points that fall within the bandwidth around it (i.e., its local neighborhood). It then shifts the point towards the mean (hence the name **Mean Shift**).
3. **Update**: Repeat this process for all points until the points converge to the **modes** (the local maximums of density).
4. **Merge Clusters**: After convergence, points that end up close to each other are grouped into the same cluster. The algorithm may merge these points to form the final clusters.


Estimated number of clusters: 3
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

Advantages:

1. **No need to specify the number of clusters**: Mean Shift can automatically determine the number of clusters based on the data.
2. **Non-linear boundaries**: It can find clusters of arbitrary shapes and sizes, unlike K-Means, which assumes spherical clusters.
3. **Robust to outliers**: Outliers are less likely to affect the result because they are unlikely to form dense regions.

Disadvantages:

1. **Computationally expensive**: Mean Shift can be slow, especially for large datasets, because it requires repeated shifting of all data points.
2. **Sensitive to bandwidth**: The choice of bandwidth is critical and can affect the quality of the clusters. Choosing the wrong bandwidth may lead to over- or under-clustering.
3. **Noisy convergence**: Some points may not converge neatly to a single mode, especially if there are regions of the data with similar densities.

OPTICS (Ordering Points To Identify the Clustering Structure)

OPTICS is a density-based clustering algorithm similar to **DBSCAN**, but it addresses some of DBSCAN's limitations, particularly with respect to handling clusters with varying densities. OPTICS creates an ordering of the data points based on their density-based clustering structure, making it more flexible than DBSCAN when dealing with data of varying densities.

Neighborhood Search:

- For each point, the algorithm computes the core distance and reachability distance with respect to its neighbors, similar to DBSCAN.

Ordering of Points:

- The algorithm starts with an unprocessed point and assigns it a reachability distance. It then updates the reachability distances of its neighbors and moves on to the next point with the smallest reachability distance. This process continues until all points are processed, forming an ordering of the points based on their reachability.

Reachability Plot:

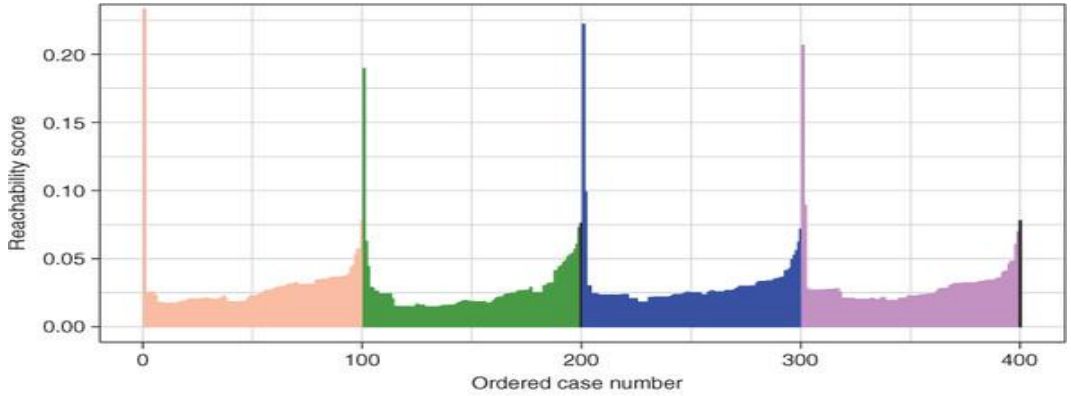
- The output is an ordered list of points with their reachability distances. A reachability plot can then be constructed, where the x-axis represents the ordering of points, and the y-axis represents the reachability distance.

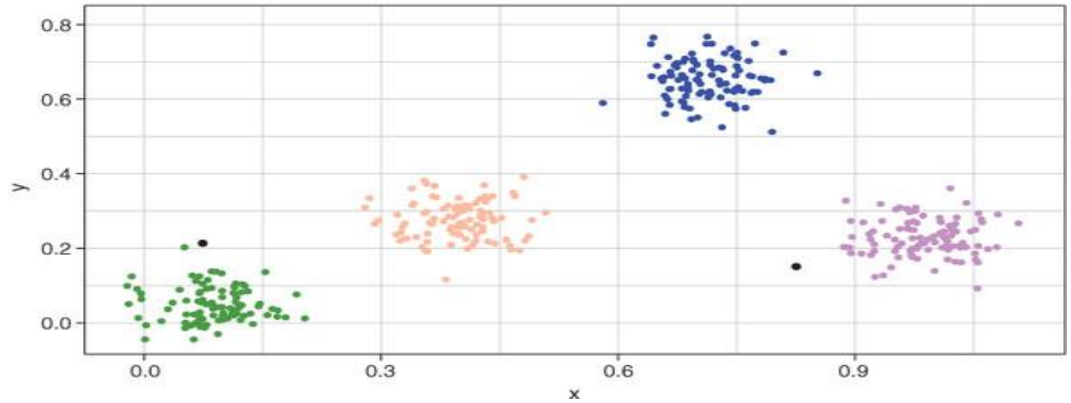
Extracting Clusters:

- Clusters can be identified by analyzing the valleys and peaks of the reachability plot. Valleys represent regions of high density, while peaks represent regions of low density or boundaries between clusters.

Advantages: Handles clusters with different densities, identifies nested clusters.

Disadvantages: Can be difficult to interpret the result, slower than DBSCAN.





Spectral Clustering

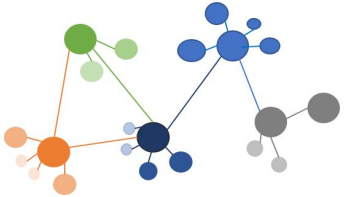
Spectral Clustering is a powerful clustering algorithm that leverages the structure of the data using graph theory. Unlike algorithms like K-Means, which partition data points directly in the original feature space, Spectral Clustering works by representing the data as a graph and using eigenvalues (or "spectrum") of matrices derived from the graph to perform clustering.

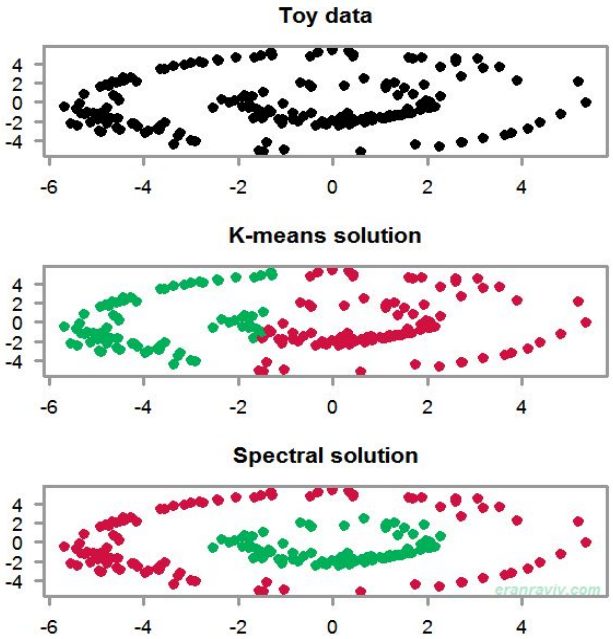
How it works:

- Constructs a similarity graph.
- Uses eigenvalues of this graph to reduce dimensionality.
- Clusters the data in the lower-dimensional space.

Advantages: Effective for data with complex structures and non-convex shapes.

Disadvantages: Computationally intensive, especially for large datasets.





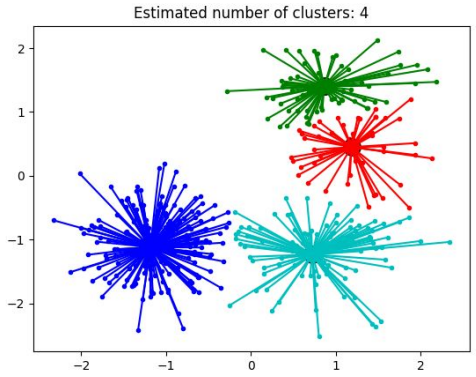
Affinity Propagation

Affinity Propagation is a clustering algorithm that works by exchanging messages between data points, identifying representative examples called "exemplars" for each cluster. Unlike traditional clustering methods like K-Means, Affinity Propagation doesn't require the number of clusters to be specified beforehand. Instead, the algorithm automatically determines the number of clusters based on the input data.

How it works:

- Sends messages between points to decide whether a point should be an exemplar or belong to another exemplar.
- No need to predefine the number of clusters.

Advantages: Finds the number of clusters automatically, works well for non-Euclidean data.
Disadvantages: Computationally expensive, may require tuning of hyperparameters.

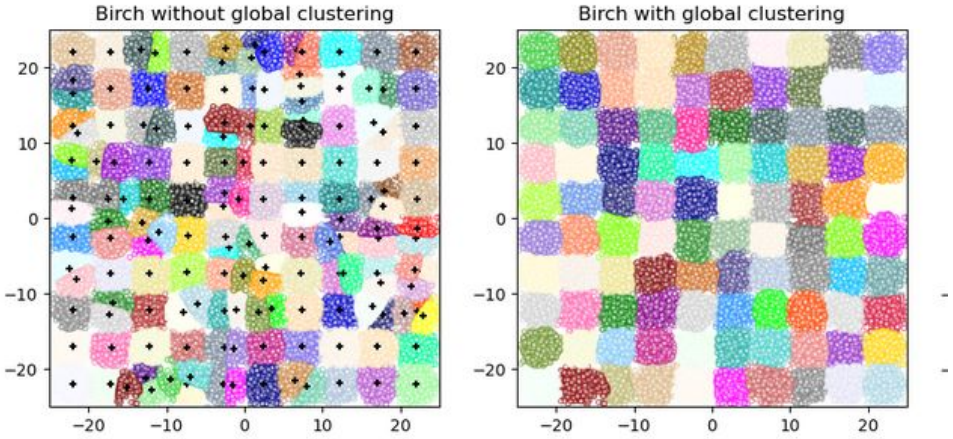


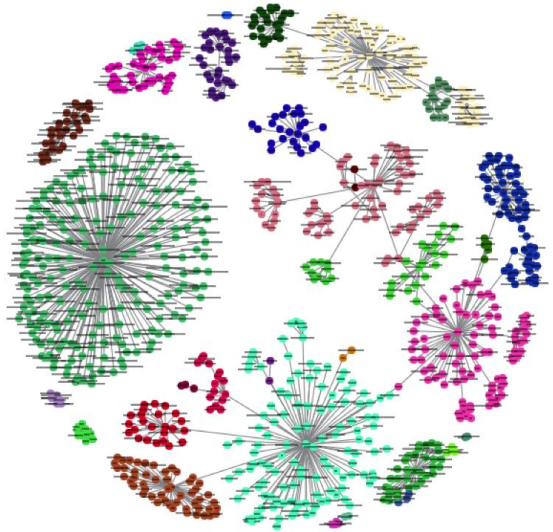
BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

BIRCH is a scalable clustering algorithm designed for large datasets. Unlike traditional clustering algorithms such as K-Means or Agglomerative Clustering, BIRCH is particularly effective for handling large, dynamic datasets where memory efficiency is important.

How BIRCH Works:

1. **Summarizes Data into Small Pieces**:
    - BIRCH groups nearby data points into small clusters, called **Clustering Features (CFs)**. These CFs store information about the group of points without needing to keep track of every individual point.
2. **Builds a Tree (CF Tree)**:
    - These small clusters are organized in a tree-like structure called a **CF Tree**. This makes it easy to add new data points or clusters as the dataset grows.
3. **Uses a Threshold**:
    - A **threshold** controls the maximum size of each small cluster. If adding a point makes the cluster too large, a new one is created.
4. **Refines the Clusters**:
    - After creating the CF Tree, BIRCH can run another algorithm (like K-Means) on the small clusters to get the final set of clusters.


Birch without global clustering
Birch with global clustering
20
10
0
−10
−20
−20
−10
0
10
20



Advantage:

1. BIRCH is designed to handle large datasets efficiently by summarizing them into small clusters (CFs) and building a compact tree structure, reducing memory usage.
2. It can process data incrementally, which means you can add data points one by one without needing to restart the clustering process.
3. BIRCH only stores summaries of data (using CFs) instead of individual points, which makes it memory-efficient and able to handle large-scale data.
4. It can tolerate noise and outliers better than some other clustering algorithms by controlling the size of clusters using the threshold.
5. Unlike K-Means, which works well with spherical clusters, BIRCH can handle clusters of different shapes and sizes.
6. BIRCH is faster than many other clustering algorithms due to its two-phase approach, where it first reduces the data into smaller subclusters and then applies clustering on those.

Disadvantages:

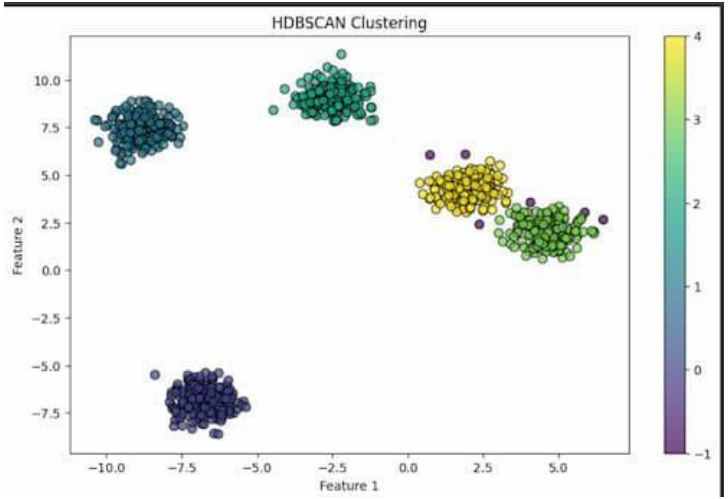
1. The quality of clustering depends heavily on the **threshold** parameter, which controls how much data can fit into each subcluster. Poor choices can lead to too many or too few clusters.
2. BIRCH may struggle with very high-dimensional data where distance measures become less meaningful, affecting clustering quality.
3. BIRCH often needs a second clustering step (like K-Means) to refine the clusters, which adds complexity and processing time.
4. BIRCH is optimized for large datasets, so it may not perform as well on small datasets compared to other clustering algorithm.
5. If the initial CF Tree is not constructed well, the resulting clusters may be poor. BIRCH works best when data is reasonably well-distributed.

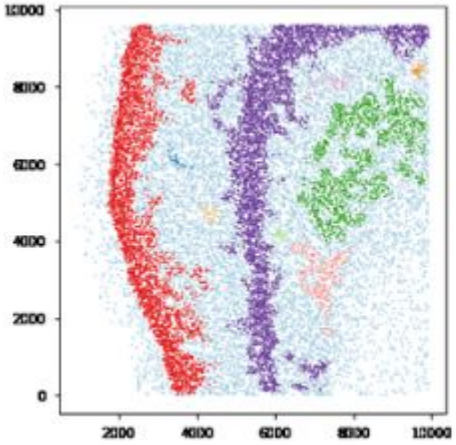
HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise)

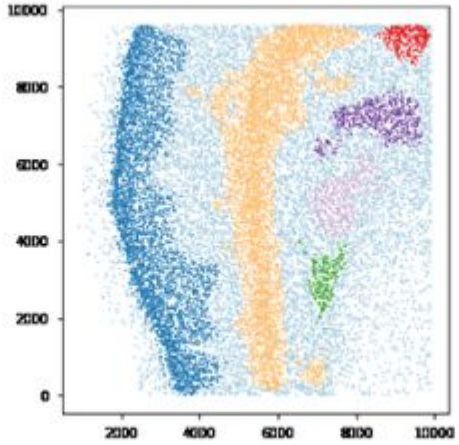
HDBSCAN is a clustering algorithm that builds on the ideas of DBSCAN but improves upon it by allowing for better handling of variable density clusters and producing a hierarchy of clusters. It's particularly useful for finding clusters of different shapes and densities and can also classify points as noise when they don’t belong to any cluster.

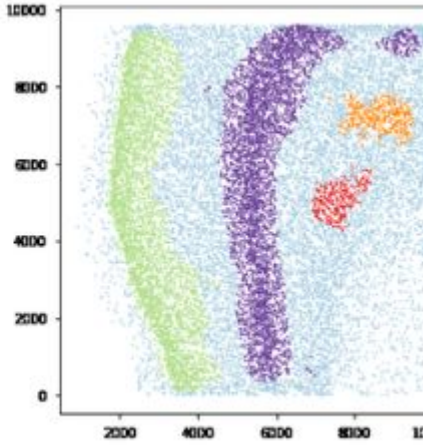
How HDBSCAN Works:

1. **Density-Based Clustering**:
    - Like DBSCAN, HDBSCAN groups points that are close together and have enough neighboring points (based on density). But HDBSCAN allows for clusters with varying densities, which is a limitation in DBSCAN.
2. **Hierarchy of Clusters**:
    - HDBSCAN creates a hierarchy of clusters, where smaller, denser clusters might exist within larger, less dense clusters. This allows it to adapt to the natural structure of the data better than other clustering methods.
3. **Condenses the Hierarchy**:
    - HDBSCAN simplifies the hierarchy by keeping only the most stable clusters. This means it automatically selects the best clustering without needing to define the number of clusters upfront.


HDBSCAN Clustering
Feature 1
Feature 2
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
4
3
2
1
0
−1


10000
8000
6000
4000
2000
0
2000
4000
6000
8000
10000


10000
8000
6000
4000
2000
0
2000
4000
6000
8000
10000


10000
8000
6000
4000
2000
0
2000
4000
6000
8000

Advantages of HDBSCAN:

1. **No Need to Specify the Number of Clusters**:
   - Unlike K-Means, you don't need to specify the number of clusters. HDBSCAN finds the optimal number based on the data.
2. **Handles Variable Density**:
   - HDBSCAN can find clusters of varying density, making it much more flexible than DBSCAN.
3. **Robust to Noise**:
   - HDBSCAN can effectively identify noise points and outliers, which makes it ideal for noisy datasets.
4. **Finds Arbitrarily Shaped Clusters**:
   - It's not restricted to spherical clusters, meaning it can find clusters of complex shapes, just like DBSCAN.
5. **Automatic Hierarchy**:
   - HDBSCAN produces a hierarchy of clusters, allowing for different granularities of clustering.

Disadvantages of HDBSCAN:

1. **Sensitive to Parameters**:
   - The choice of parameters like **min_cluster_size** and **min_samples** can significantly affect the results. It may take some tuning.
2. **Computational Complexity**:
   - HDBSCAN is more computationally expensive than simpler clustering algorithms like K-Means, especially on very large datasets.
3. **Does Not Work Well in High Dimensions**:
   - Like most density-based clustering algorithms, HDBSCAN struggles in high-dimensional data because distance metrics become less meaningful.