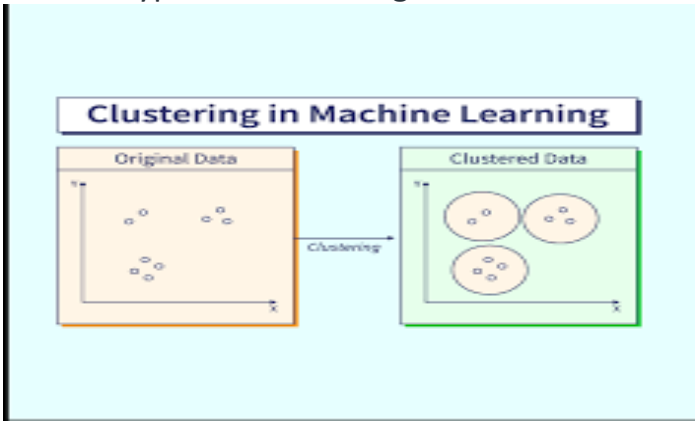
Clustering is an unsupervised machine learning technique used to group similar data points together without using labelled data. It helps discover hidden patterns or natural groupings in datasets by placing similar data points into the same cluster.

- Discover the natural grouping or structure in unlabelled data without predefined categories.
- Data points are assigned to clusters based on similarity or distance measures.
- Uses Euclidean distance, cosine similarity or other metrics depending on data type and clustering method.
- 

Types of Clustering

1. Hard Clustering

Hard clustering assigns each data point to exactly one cluster. A data point cannot belong to multiple clusters, making the grouping clear and easy to interpret.

- Each data point belongs to only one cluster
- No overlap between clusters
- Simple and easy to interpret

Example

If customers are divided into two clusters, each customer belongs completely to either Cluster 1 or Cluster 2. A customer cannot belong to both clusters at the same time.

Common Uses

- **Market segmentation:** Businesses group customers with similar buying behaviour to design targeted marketing strategies.
- **Customer grouping:** Companies organize customers into clear categories for better service and analysis.

- **Document clustering:** Documents with similar topics or keywords are grouped together for easier organization.

Limitation

Cannot represent overlapping groups: Hard clustering cannot handle situations where a data point may logically belong to multiple groups.

2. Soft Clustering

Soft clustering allows a data point to belong to multiple clusters with different probabilities. Instead of assigning a strict cluster, it gives a degree of membership to each cluster.

Example

A data point may belong 70% to Cluster 1 and 30% to Cluster 2, indicating that it shares characteristics with both groups.

Use Cases

- **Overlapping class boundaries:** Useful when data points cannot be clearly separated into distinct groups.
- **Customer personas:** Helps represent customers who share traits with multiple behavioral groups.
- **Medical diagnosis:** Patients may show symptoms related to multiple conditions.

Benefits

- **Captures ambiguity:** Represents uncertainty when cluster boundaries are not clear.
- **Models gradual transitions:** Allows smooth transitions between clusters instead of strict separation.

Applications

Clustering is widely used in data analysis and machine learning to identify patterns in unlabelled data.

- **Customer Segmentation:** Group customers based on behaviour or demographics.
- **Anomaly Detection:** Detect unusual activities in finance, security or sensor data.
- **Image Segmentation:** Divide images into meaningful regions for computer vision tasks.

- **Recommendation Systems:** Group similar users or items for personalized suggestions.
- **Market Basket Analysis:** Identify products frequently purchased together.

Hierarchical Clustering in Machine Learning

Hierarchical Clustering is an unsupervised learning technique that groups data into a hierarchy of clusters based on similarity. It builds a tree-like structure (dendrogram) that helps visualize relationships and decide the optimal number of clusters.

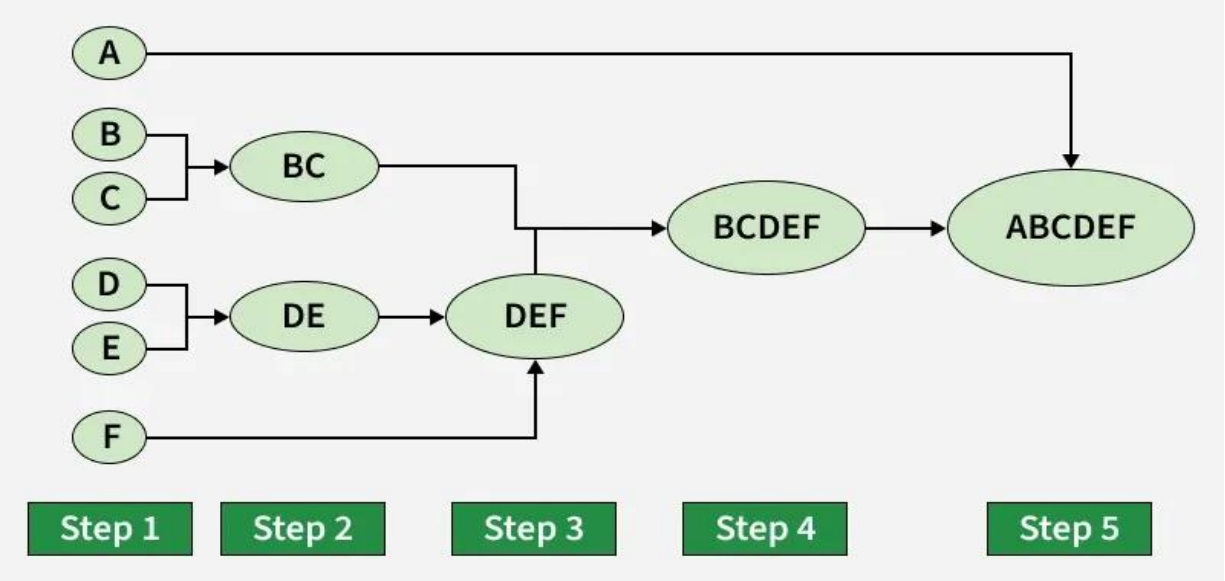
- Does not require pre-selecting the number of clusters.
- Uses agglomerative or divisive approaches.
- Commonly applied in data exploration and pattern discovery.
- It is commonly used in pattern recognition, customer segmentation and image grouping.

There are two types of hierarchical clustering:

1. Agglomerative Clustering
2. Divisive clustering

1. Hierarchical Agglomerative Clustering

It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.



Workflow for Hierarchical Agglomerative clustering

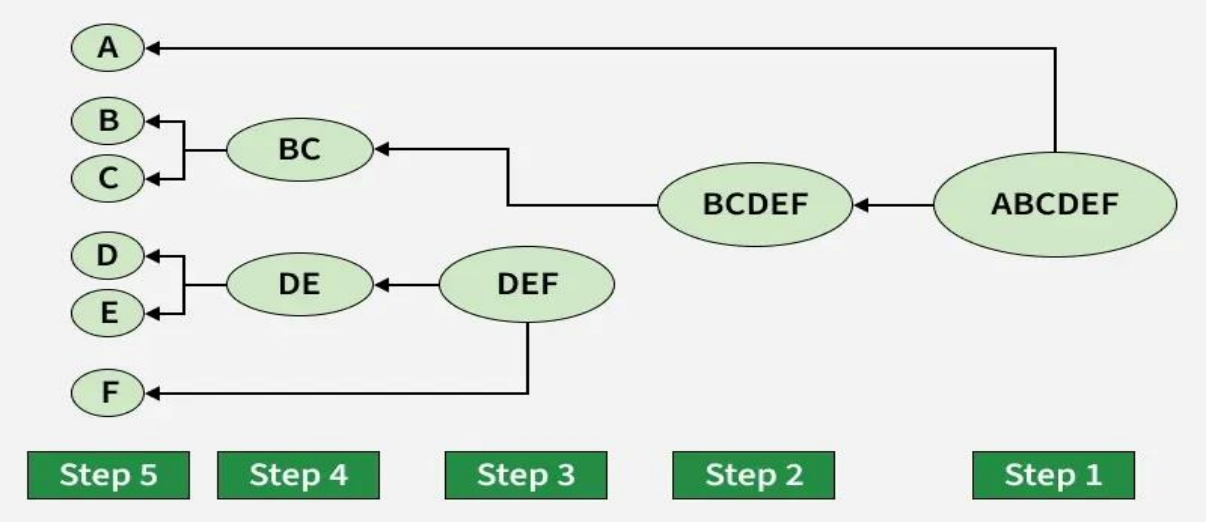
1. **Start with individual points:** Each data point is its own cluster. For example if we have 5 data points we start with 5 clusters each containing just one data point.
2. **Calculate distances between clusters**: Calculate the distance between every pair of clusters. Initially since each cluster has one point this is the distance between the two data points.
3. **Merge the closest clusters**: Identify the two clusters with the smallest distance and merge them into a single cluster.
4. **Update distance matrix**: After merging we now have one less cluster. Recalculate the distances between the new cluster and the remaining clusters.
5. **Repeat steps 3 and 4**: Keep merging the closest clusters and updating the distance matrix until we have only one cluster left.
6. **Create a dendrogram**: As the process continues we can visualize the merging of clusters using a tree-like diagram called a dendrogram. It shows the hierarchy of how clusters are merged.

2. Hierarchical Divisive clustering

Divisive clustering is also known as a top-down approach. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.

Workflow for Hierarchical Divisive clustering :

1. **Start with all data points in one cluster:** Treat the entire dataset as a single large cluster.
2. **Split the cluster**: Divide the cluster into two smaller clusters. The division is typically done by finding the two most dissimilar points in the cluster and using them to separate the data into two parts.
3. **Repeat the process**: For each of the new clusters, repeat the splitting process: Choose the cluster with the most dissimilar points and split it again into two smaller clusters.
4. **Stop when each data point is in its own cluster**: Continue this process until every data point is its own cluster or the stopping condition (such as a predefined number of clusters) is met.



Applications

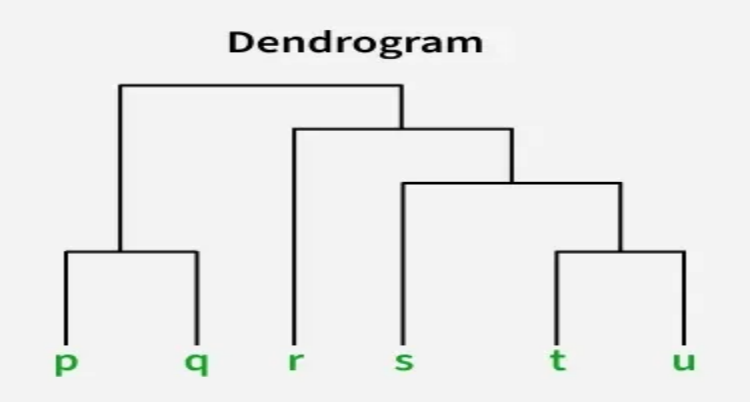
There are many real-life applications of Hierarchical clustering. They include:

- Bioinformatics: grouping animals according to their biological features to reconstruct phylogeny trees
- Business: dividing customers into segments or forming a hierarchy of employees based on salary.
- Image processing: grouping handwritten characters in text recognition based on the similarity of the character shapes.
- Information Retrieval: categorizing search results based on the query.

Dendrogram

A dendrogram is like a family tree for clusters. It shows how individual data points or groups of data merge together. The bottom shows each data point as its own group and as we move up, similar groups are combined.

The lower the merge point, the more similar the groups are. It helps us see how things are grouped step by step.



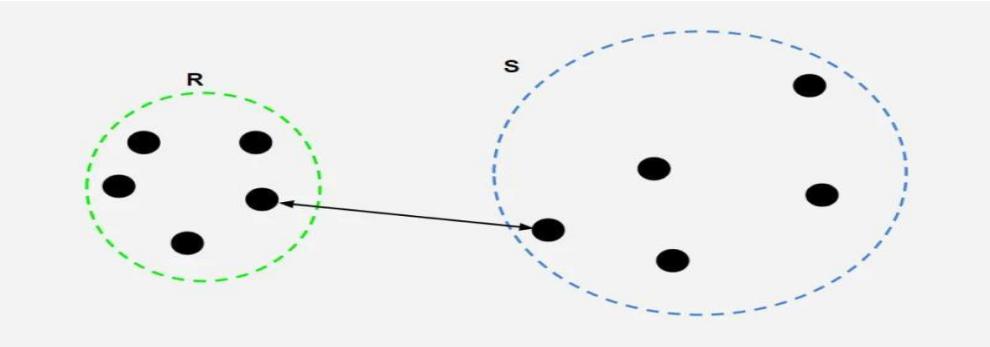
Types of Linkages in Hierarchical Clustering

Hierarchical clustering is used to group similar data points and organise data in a tree-like structure. Key part of this process is **linkage** which **calculates the distance between clusters before they are merged or divided**. Different types of linkage is used measure this distance differently. In this article, we'll look at different linkage methods and see how they affect the cluster formation.

1. Single Linkage

For two clusters **R** and **S** the **single linkage returns the minimum distance between two points**. This method creates long, chain-like clusters because it is **sensitive to outliers** and can **connect clusters based on a very small number of close points**.

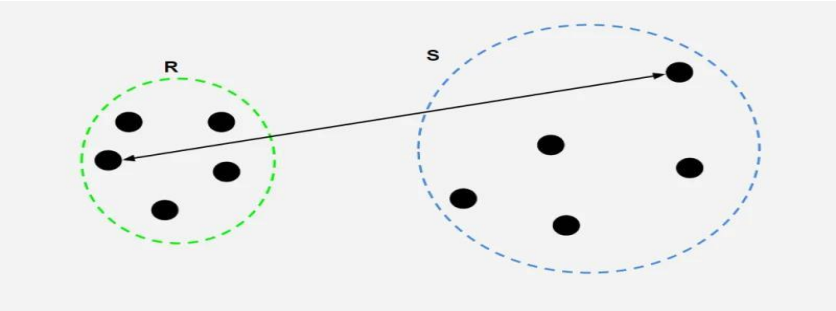
$L(R,S)=min(D(i,j)),i\epsilon R,j\epsilon S$



2. Complete Linkage

For two clusters **R** and **S** the **complete linkage returns the maximum distance between two points**. It tends to create compact and spherical clusters because it is **more sensitive to outliers and tries to make sure that the clusters are not too far**.

L(R,S)=max(D(i,j)),i∈R,j∈



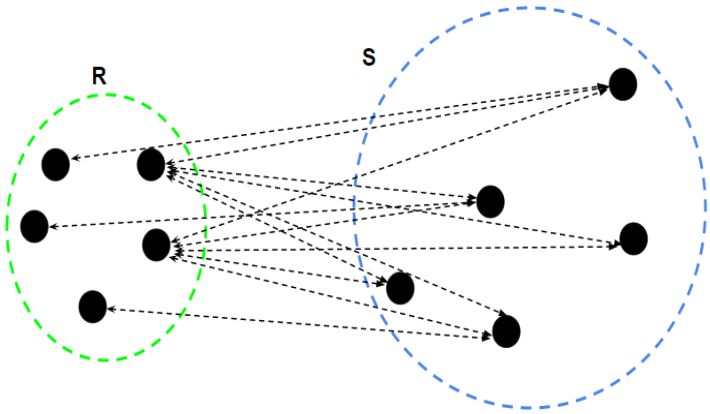
3. Average Linkage

It returns the **average distance between all pairs of points from two clusters**. This method maintain a **balance between single and complete linkage** by considering all pairs of points not just the closest or farthest point. It usually **results in clusters that are moderately compact**.

L(R,S)=1nR×nS∑i=1nR∑j=1nSD(i,j),i∈R,j∈S$L(R,S) = \frac{1}{nR \times nS} \sum_{i=1}^{nR} \sum_{j=1}^{nS} D(i,j), i \in R, j \in S$

where

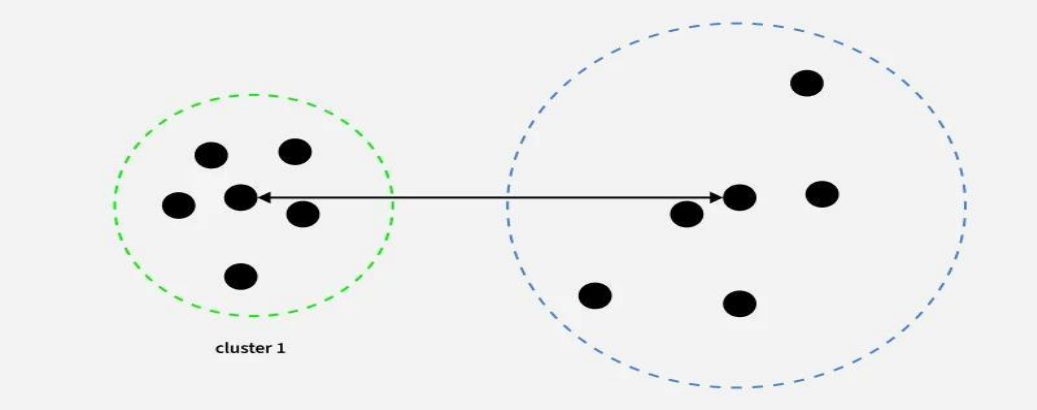
- nRnR : Number of data-points in R
- nSnS : Number of data-points in S



Centroid Linkage

It calculates the distance between two clusters based on the **distance between their central points i.e the average of all points in the cluster**. This method works well when **clusters are round or evenly shaped** but it **may not be the best for irregularly shaped clusters**.

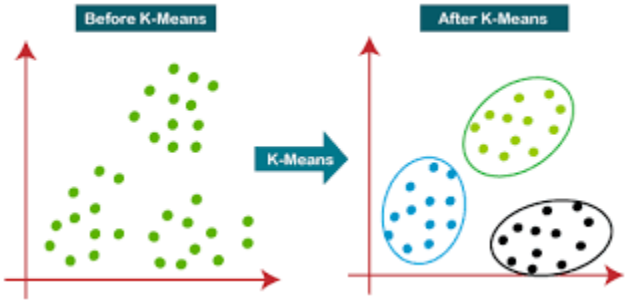
$L(R,S)=D(\bar{R},\bar{S})$



k-means clustering algorithm:

K-Means Clustering groups similar data points into clusters without needing labeled data. It is used to uncover hidden patterns when the goal is to organize data based on similarity.

- Helps identify natural groupings in unlabeled datasets
- Works by grouping points based on distance to cluster centers
- Commonly used in customer segmentation, image compression, and pattern discovery
- Useful when you need structure from raw, unorganized data



Working of K-Means Clustering

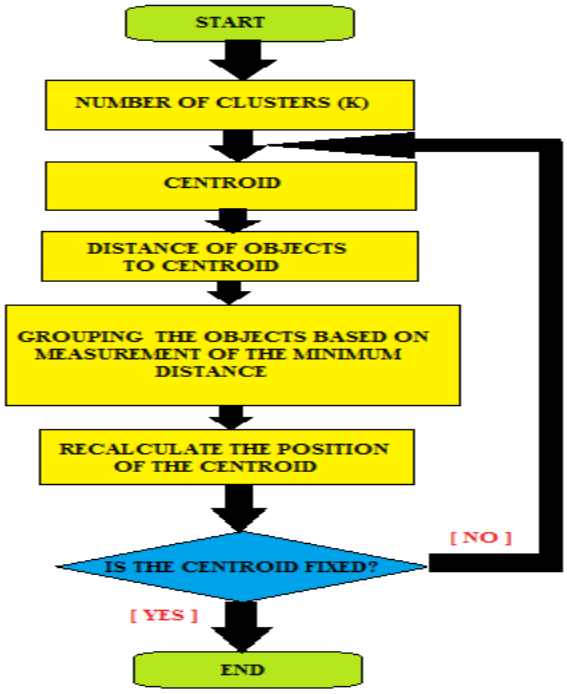
Suppose we are given a data set of items with certain features and values for these features like a vector. The task is to categorize those items into groups. To achieve this we will use the K-means algorithm. "k*k*" represents the number of groups or clusters we want to classify our items into.

1. **Initialization:** We begin by randomly selecting k cluster centroids.
2. **Assignment Step:** Each data point is assigned to the nearest centroid, forming clusters.
3. **Update Step:** After the assignment, we recalculate the centroid of each cluster by averaging the points within it.
4. **Repeat:** This process repeats until the centroids no longer change or the maximum number of iterations is reached.

The way k-means algorithm works is as follows:

1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.



Challenges with K-Means Clustering

K-Means algorithm has the following limitations:

- **Choosing the Right Number of Clusters** (k*k*): One of the biggest challenges is deciding how many clusters to use.
- **Sensitive to Initial Centroids:** The final clusters can vary depending on the initial random placement of centroids.
- **Non-Spherical Clusters:** K-Means assumes that the clusters are spherical and equally sized. This can be a problem when the actual clusters in the data are of different shapes or densities.
- **Outliers**: K-Means is sensitive to outliers, which can distort the centroid and, ultimately, the clusters.

Key Applications:

- **Customer Segmentation:** Grouping customers by purchasing behavior.
- **Image Segmentation:** Dividing images into regions by color similarity.
- **Anomaly Detection:** Identifying outliers that do not fit into any cluster.
- **Document Clustering:** Organizing documents by content similarity.

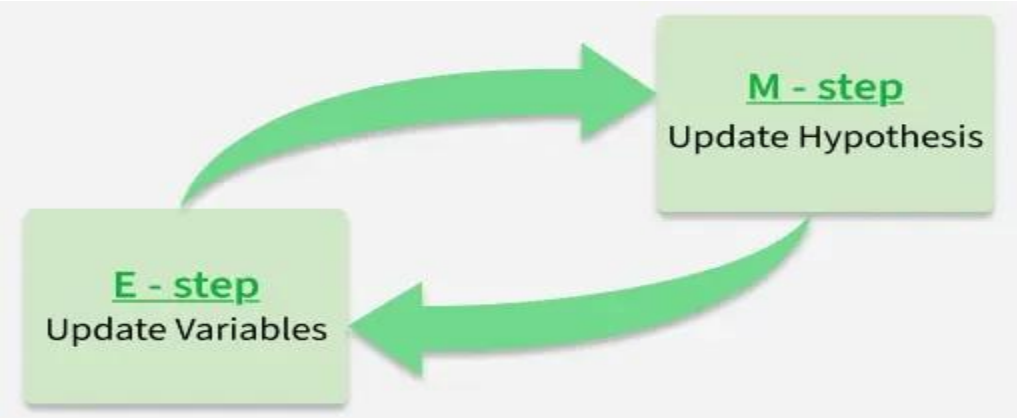
Expectation-Maximization Algorithm – ML

The Expectation-Maximization (EM) algorithm is a powerful iterative optimization technique used to estimate unknown parameters in probabilistic models, particularly when the data is incomplete, noisy or contains hidden (latent) variables. It works in two steps:

- **E-step (Expectation Step):** Using the current parameter estimates, the algorithm calculates the expected values of the missing or hidden variables. Essentially, it assigns probabilities or "responsibilities" to different hidden outcomes given the observed data.
- **M-step (Maximization Step):** With these updated expectations from the E-step, the algorithm then re-estimates the model parameters by maximizing the expected log-likelihood. This improves how well the model explains the observed data.

These two steps are repeated until convergence, which typically means that:

- The parameter values stop changing significantly, or
- The log-likelihood improves only by a negligible amount.



ey Terms in Expectation-Maximization (EM) Algorithm

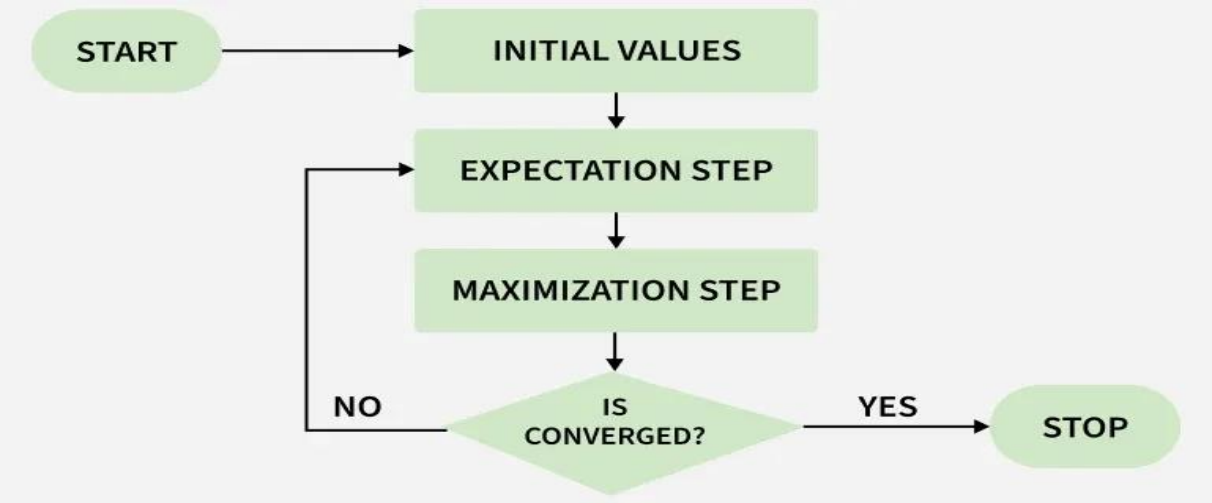
Lets understand about some of the most commonly used key terms in the Expectation-Maximization (EM) Algorithm:

- **Latent Variables**: Variables that are not directly observed but are inferred from the data. They represent hidden structure (e.g., cluster assignments in Gaussian Mixture Models).
- **Likelihood**: The probability of the observed data given a set of model parameters. EM aims to find parameter values that maximize this likelihood.

- **Log-Likelihood**: The natural logarithm of the likelihood function. It simplifies calculations (turning products into sums) and is numerically more stable when dealing with very small probabilities.
- **Maximum Likelihood Estimation (MLE)**: A statistical approach to estimating parameters by choosing the values that maximize the likelihood of observing the given data. EM extends MLE to cases with hidden or missing variables.
- **Posterior Probability**: In Bayesian inference, this represents the probability of parameters (or latent variables) given the observed data and prior knowledge. In EM, posterior probabilities are used in the E-step to estimate the "responsibility" of each hidden variable.
- **Convergence**: The stopping criterion for the iterative process. EM is said to converge when updates to parameters or improvements in log-likelihood become negligibly small, meaning the algorithm has reached a stable solution.

Working of Expectation-Maximization (EM) Algorithm

- Here's a step-by-step breakdown of the process:



1. Initialization: The algorithm starts with initial parameter values and assumes the observed data comes from a specific model.

2. E-Step (Expectation Step):

- Find the missing or hidden data based on the current parameters.
- Calculate the posterior probability of each latent variable based on the observed data.
- Compute the log-likelihood of the observed data using the current parameter estimates.

3. M-Step (Maximization Step):

- Update the model parameters by maximize the log-likelihood.
- The better the model the higher this value.

4. Convergence:

- Check if the model parameters are stable and converging.
- If the changes in log-likelihood or parameters are below a set threshold, stop. If not repeat the E-step and M-step until convergence is reached

Applications

- **Clustering**: Used in Gaussian Mixture Models (GMMs) to assign data points to clusters probabilistically.
- **Missing Data Imputation**: Helps fill in missing values in datasets by estimating them iteratively.
- **Image Processing**: Applied in image segmentation, denoising and restoration tasks where pixel classes are hidden.
- **Natural Language Processing (NLP):** Used in tasks like word alignment in machine translation and topic modeling (LDA).
- **Hidden Markov Models (HMMs):** EM's variant, the Baum-Welch algorithm, estimates transition/emission probabilities for sequence data.

Advantages

- **Monotonic improvement**: Each iteration increases (or at least never decreases) the log-likelihood.
- **Handles incomplete data well**: Works effectively even with missing or hidden variables.
- **Flexibility**: Can be applied to many probabilistic models, not just mixtures of Gaussians.
- **Easy to implement**: The E-step and M-step are conceptually simple and often have closed-form updates.

Disadvantages

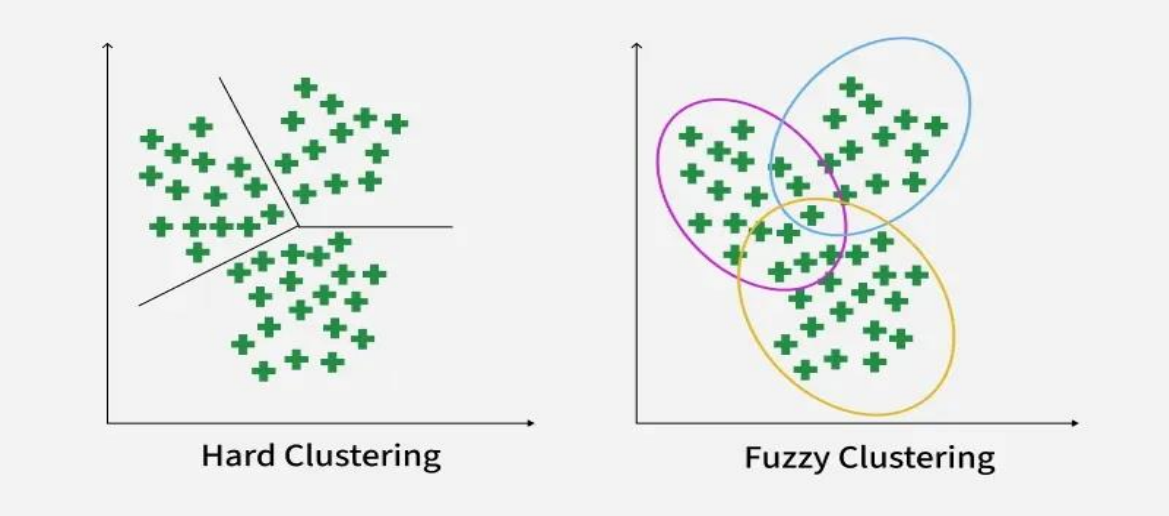
- **Slow convergence**: Convergence can be very gradual, especially near the optimum.
- **Initialization sensitive**: Requires good initial parameter guesses; poor choices may yield bad solutions.

- **No guarantee of global best solution**: Unlike some optimization methods, EM doesn't guarantee reaching the absolute best parameters.
- **Computationally intensive**: For large datasets or complex models, repeated iterations can be costly.

Fuzzy Clustering - ML

Fuzzy clustering allows each data point to belong to multiple clusters with different membership values. Instead of assigning a point to just one group, it captures how strongly a point relates to each cluster.

- Uses membership scores between 0 and 1
- Handles overlapping or unclear cluster boundaries
- More flexible than hard clustering methods
- Useful when data points don't fit neatly into a single group



Working of Fuzzy Clustering

Fuzzy clustering assigns each data point a degree of membership for every cluster and updates these values through an iterative process. Here's how it works:

Step 1: Initialize Membership Values Randomly: Each data point is given random membership scores for all clusters. A point can partially belong to multiple clusters.

Step 2: Compute Cluster Centroids: Centroids are calculated as weighted averages, where weights are membership values raised to the fuzziness parameter m:

Step 3: Calculate Distance Between Data Points and Centroids: Compute the Euclidean distance between each point and every centroid to determine proximity, which will be used to update memberships

Step 4: Update Membership Values: Membership values are updated inversely proportional to these distances. Points closer to a centroid get higher membership.

Step 5: Repeat Until Convergence: Steps 2–4 are repeated until the membership values stabilize meaning there are no significant changes from one iteration to the next. This indicates that the clustering has reached an optimal state.

Applications

- **Image Segmentation**: Handles noise and overlapping regions efficiently.
- **Pattern Recognition**: Identifies ambiguous patterns in speech, handwriting, etc.
- **Customer Segmentation**: Groups customers with partial membership for flexible marketing.
- **Medical Diagnosis**: Analyzes patient or genetic data with uncertain boundaries.
- **Bioinformatics**: Captures multifunctional gene roles by assigning genes to multiple clusters.

Advantages

- **Flexibility**: Allows overlapping clusters representing ambiguous or complex data.
- **Robustness**: More resilient to noise and outliers by soft memberships.
- **Detailed Insights**: Membership degrees give richer understanding of data relationships.
- **Better Representation**: Suitable when strict cluster boundaries are unrealistic.

limitations

- **Computationally Intensive**: More expensive than hard clustering due to membership optimization.
- **Parameter Sensitivity**: Choosing number of clusters and fuzziness parameter needs expertise.
- **Complexity in Interpretation**: Results can be harder to interpret than crisp clusters.

- ## Spectral Clustering:

- Spectral clustering is an advanced, graph-based technique for identifying complex, non-convex, or nested clusters by analyzing the eigenvalues (spectrum) of a similarity matrix. It projects data into a lower-dimensional subspace based on connectivity, then uses algorithms like Means to partition the data.

Spectral Clustering is a variant of the clustering algorithm that uses the connectivity between the data points to form the clustering. It uses eigenvalues and eigenvectors of the data matrix to forecast the data into lower dimensions space to cluster the data points. It is based on the idea of a graph representation of data where the data point are represented as nodes and the similarity between the data points are represented by an edge.

Advantages of Spectral Clustering:

1. Scalability: Spectral clustering can handle large datasets and high-dimensional data, as it reduces the dimensionality of the data before clustering.
2. Flexibility: Spectral clustering can be applied to non-linearly separable data, as it does not rely on traditional distance-based clustering methods.
3. Robustness: Spectral clustering can be more robust to noise and outliers in the data, as it considers the global structure of the data, rather than just local distances between data points.

Disadvantages of Spectral Clustering:

1. Complexity: Spectral clustering can be computationally expensive, especially for large datasets, as it requires the calculation of eigenvectors and eigenvalues.
2. Model selection: Choosing the right number of clusters and the right similarity matrix can be challenging and may require expert knowledge or trial and error.