MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Dr. Vishwanath Karad

MIT WORLD PEACE UNIVERSITY | PUNE

TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

MACHINE LEARNING

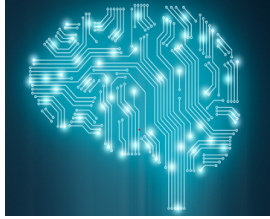
Professional Core(CET3006B)

T. Y. B.Tech AIDS, Sem-V

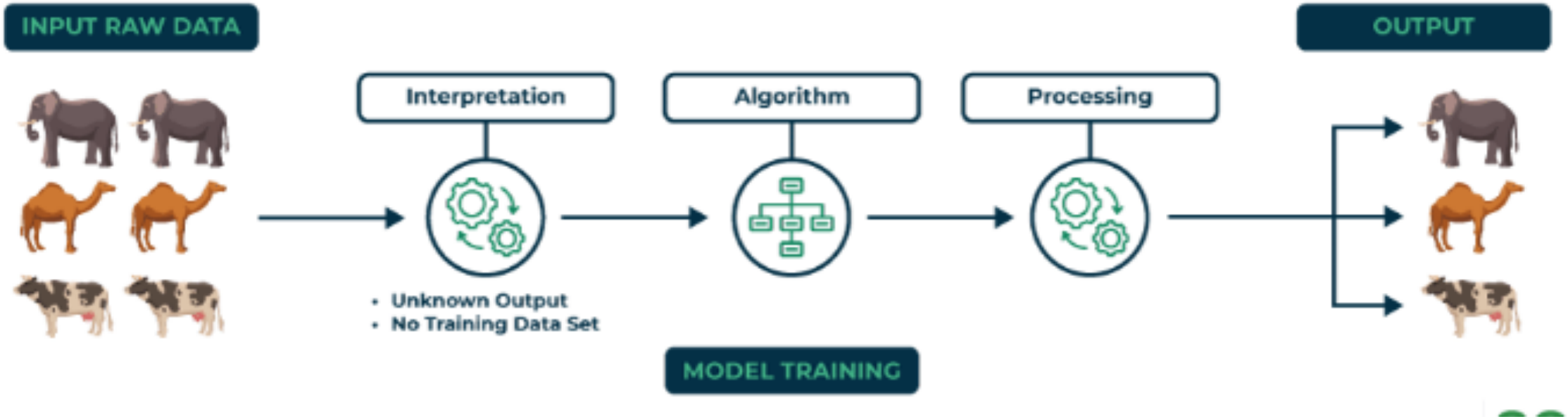
2024-25

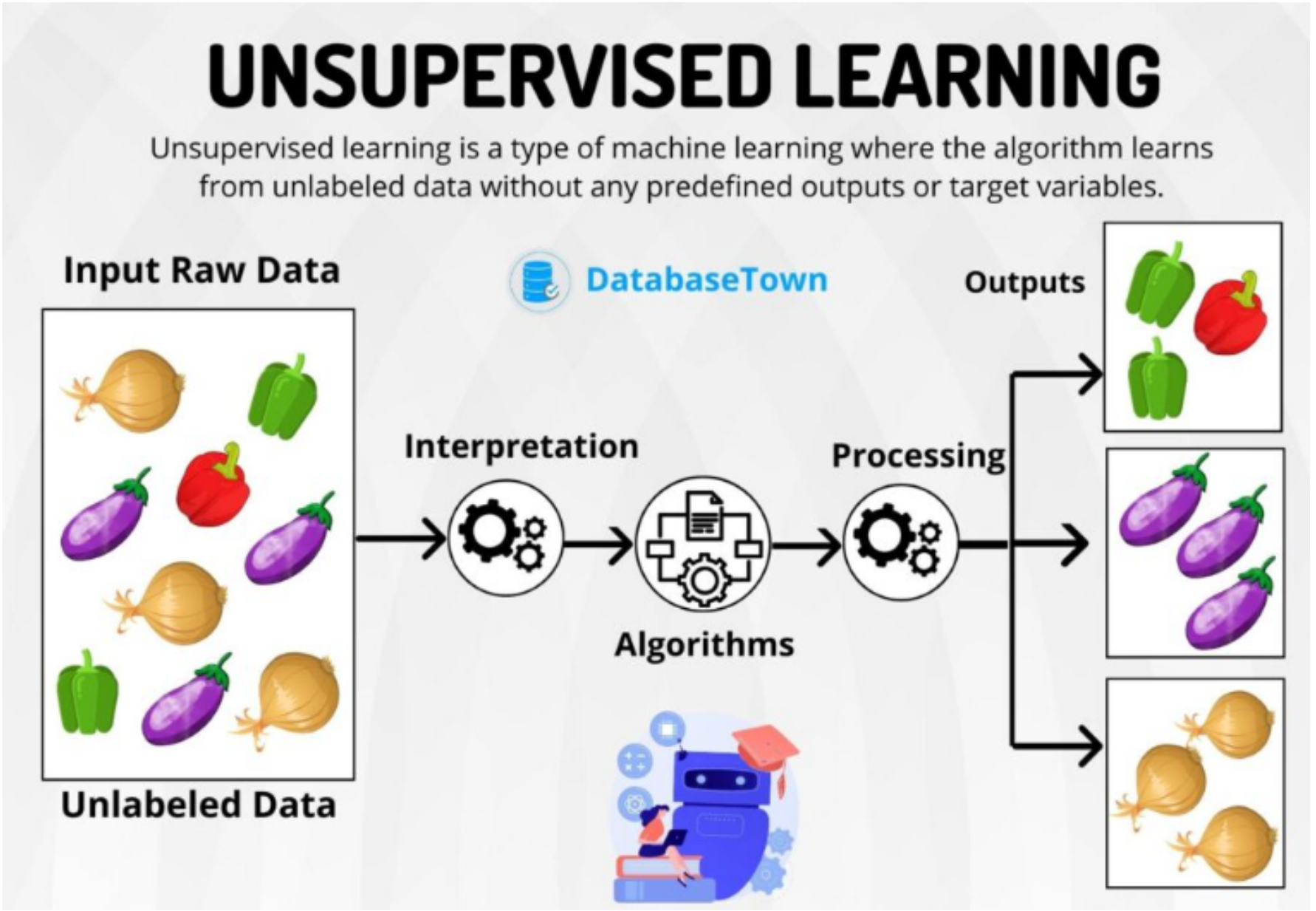
UNIT-III

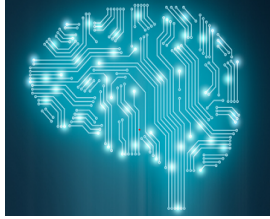
SoCSE – Dept. of Computer Engineering & Technology



Unsupervised Learning

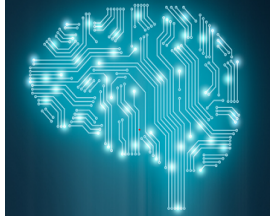



UNSUPERVISED LEARNING
Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any predefined outputs or target variables.
Input Raw Data
DatabaseTown
Outputs
Interpretation
Processing
Algorithms
Unlabeled Data



How does unsupervised learning work?

Unsupervised learning works by analyzing unlabeled data to identify patterns and relationships. The data is not labeled with any predefined categories or outcomes, so the algorithm must find these patterns and relationships on its own. This can be a challenging task, but it can also be very rewarding, as it can reveal insights into the data that would not be apparent from a labeled dataset.



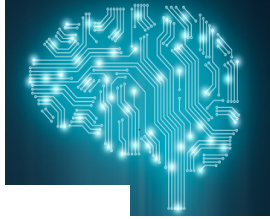
The input to the unsupervised learning models is as follows:

- **Unstructured data**: May contain noisy(meaningless) data, missing values, or unknown data
- **Unlabeled data**: Data only contains a value for input parameters, there is no targeted value(output). It is easy to collect as compared to the labeled one in the Supervised approach.

Unsupervised Learning Algorithms

There are mainly 3 types of Algorithms which are used for Unsupervised dataset.

- Clustering
- Association Rule Learning
- Dimensionality Reduction

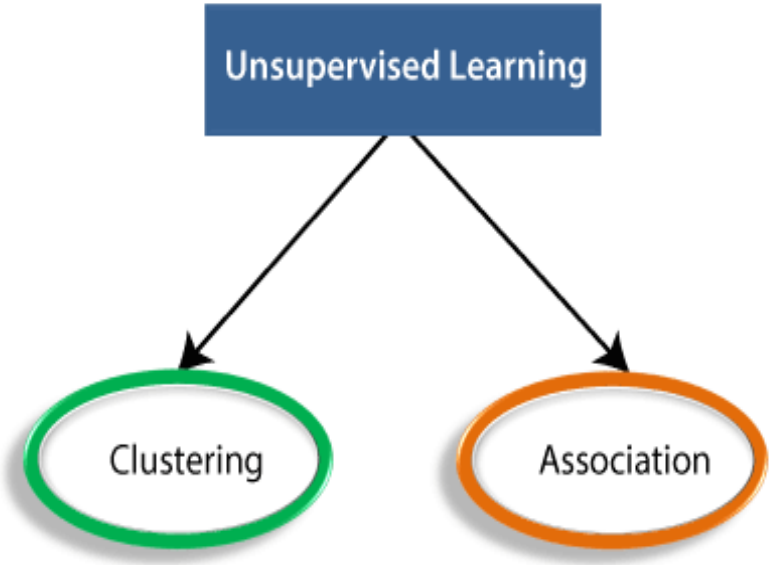


Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

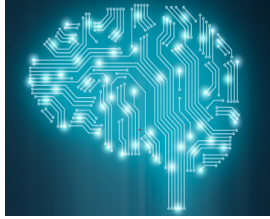
Types of Unsupervised Learning



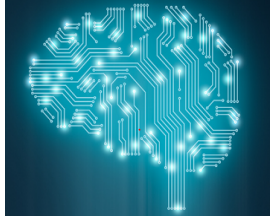
•**Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

•**Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Clustering Types

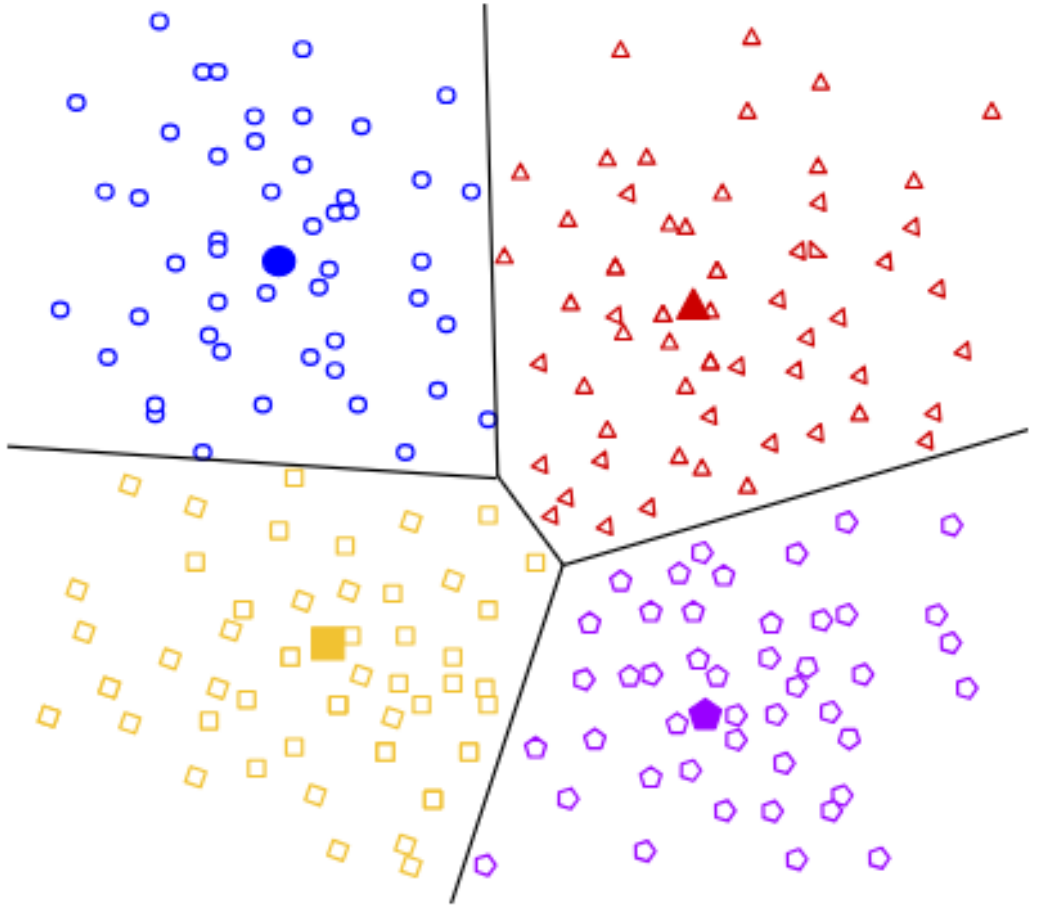


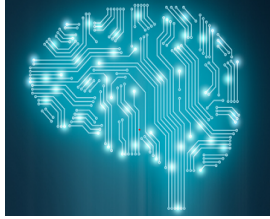
- Centroid Based Clustering
- Density Based Clustering
- Distribution Based Clustering
- Hierarchical Clustering



Centroid-based clustering

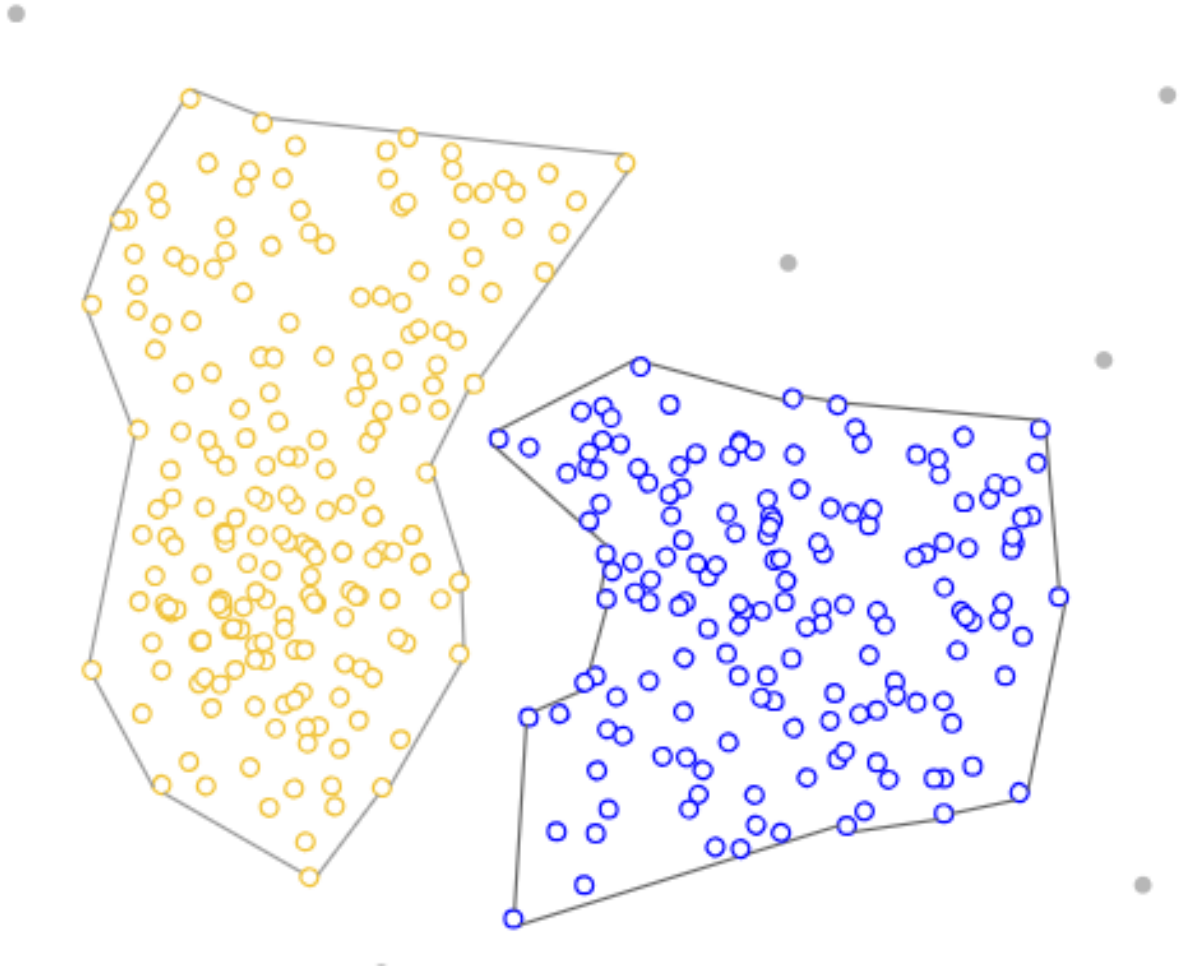
The centroid of a cluster is the arithmetic mean of all the points in the cluster. **Centroid-based clustering** organizes the data into non-hierarchical clusters. Centroid-based clustering algorithms are efficient but sensitive to initial conditions and outliers. Of these, k-means is the most widely used. It requires users to define the number of centroids, *k*, and works well with clusters of roughly equal size.

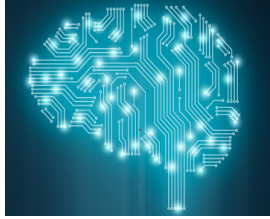




Density-based clustering

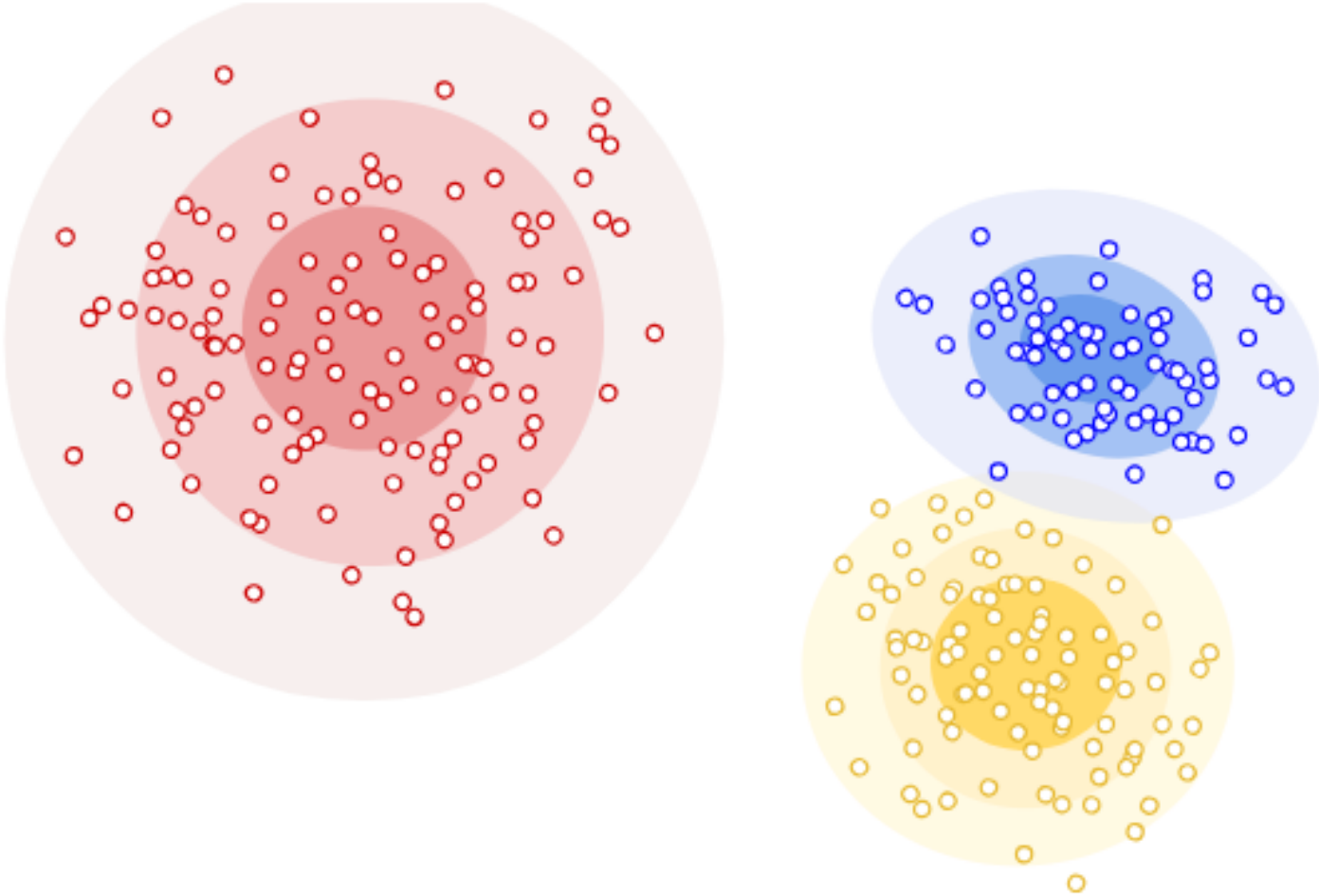
Density-based clustering connects contiguous areas of high example density into clusters. This allows for the discovery of any number of clusters of any shape. Outliers are not assigned to clusters. These algorithms have difficulty with clusters of different density and data with high dimensions.

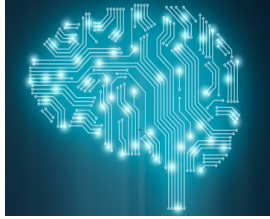




Distribution-based clustering

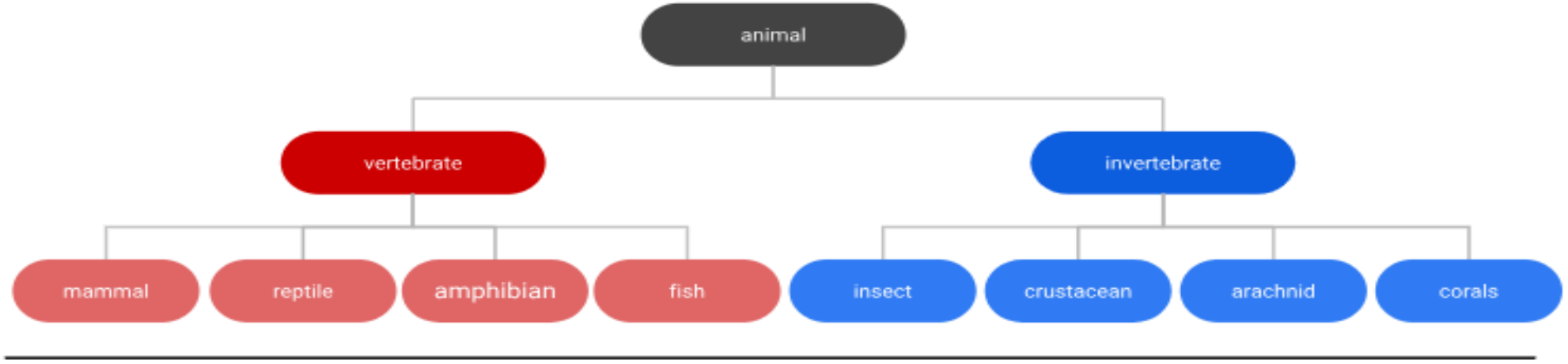
This clustering approach assumes data is composed of probabilistic distributions, such as **Gaussian distributions**. In Figure 3, the distribution-based algorithm clusters data into three Gaussian distributions. As distance from the distribution's center increases, the probability that a point belongs to the distribution decreases. The bands show that decrease in probability. When you're not comfortable assuming a particular underlying distribution of the data, you should use a different algorithm.

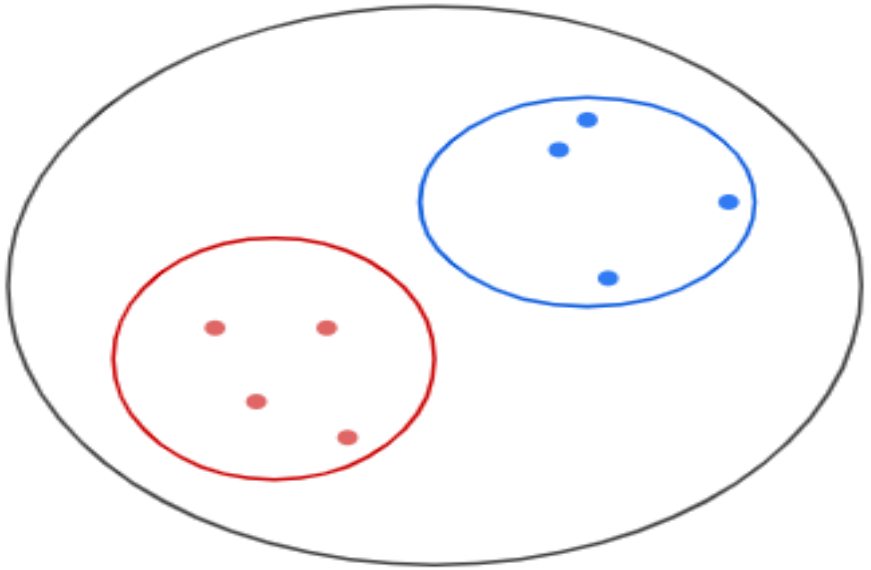


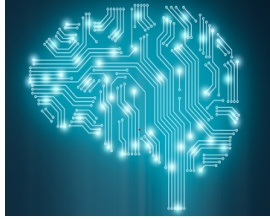


Hierarchical clustering

Hierarchical clustering creates a tree of clusters. Hierarchical clustering, not surprisingly, is well suited to hierarchical data, such as taxonomies. See *Comparison of 61 Sequenced Escherichia coli Genomes* by Oksana Lukjancenko, Trudy Wassenaar & Dave Ussery for an example. Any number of clusters can be chosen by cutting the tree at the right level.

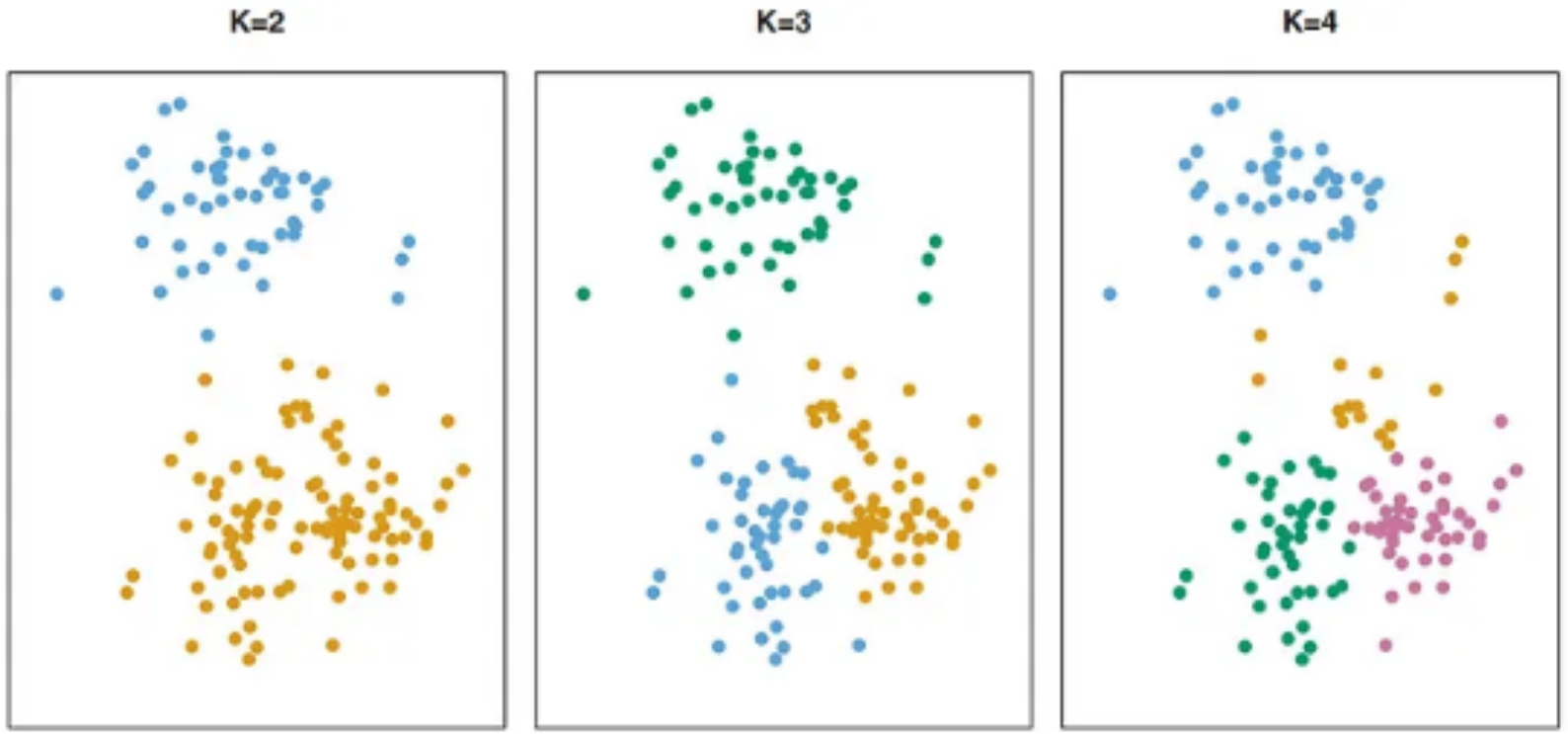




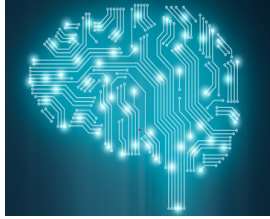


K Means and K Medoids

Clustering is a powerful technique in machine learning and data analysis, used to group similar data points together. Two popular clustering algorithms, K-Means and K-Medoids (PAM), play a significant role in this realm.

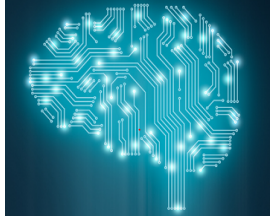


'k-means clustering' minimizes within-cluster variances (squared Euclidean distances)



K Means Clustering

- K-Means clustering aims to partition a dataset into a specified number of clusters, minimizing the within-cluster variances. The process involves randomly assigning observations to clusters, computing cluster centroids, assigning observations to the nearest centroid, and iteratively recalibrating until clusters stabilize around optimal centroids. However, K-Means has its limitations, such as sensitivity to initial cluster assignments, the need to specify the number of clusters (k), and the tendency to produce equal-sized, spherical clusters.
-



K Means Clustering

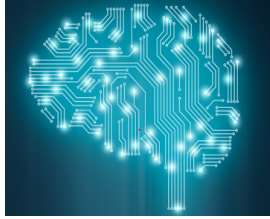
Algorithm 10.1 *K-Means Clustering*

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.

2. Iterate until the cluster assignments stop changing:

   (a) For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.

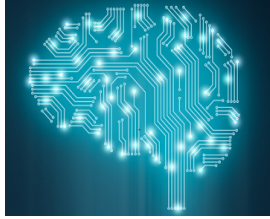
   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).



K Medoids

- Partition Around Medoids (PAM or K-Medoids) is an alternative to K-Means that uses actual data points, known as medoids, as cluster centers. The medoid is defined as the object in a cluster with a minimal sum of dissimilarities to all other objects within the same cluster. Unlike K-Means, K-Medoids are more robust to noise and outliers, making them suitable for certain datasets.

K Medoids



- **K-Medoids Clustering Algorithm** .

1. First, we select K random data points from the dataset and use them as medoids.
2. Now, we will calculate the distance of each data point from the medoids. You can use any of the Euclidean, Manhattan distance, or squared Euclidean distances as the distance measure.
3. Once we find the distance of each data point from the medoids, we will assign the data points to the clusters associated with each medoid. The data points are assigned to the medoids at the closest distance.
4. After determining the clusters, we will calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. Let the cost be C_i.

K-Medoid Clustering

5. Now, we will select a random data point D_j from the dataset and swap it with a medoid M_i. Here, D_j becomes a temporary medoid. After swapping, we will calculate the distance of all the non-medoid data points to the current medoid of each cluster. Let this cost be C_j.

6. If $C_i > C_j$, the current medoids with D_j as one of the medoids are made permanent medoids. Otherwise, we undo the swap, and M_i is reinstated as the medoid.

7. Repeat 4 to 6 until no change occurs in the clusters.

K-Medoid Clustering

K-Medoids Clustering Example

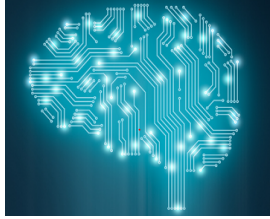
Point	Coordinates
A1	(2, 6)
A2	(3, 8)
A3	(4, 7)
A4	(6, 2)
A5	(6, 4)
A6	(7, 3)
A7	(7,4)
A8	(8, 5)
A9	(7, 6)
A10	(3, 4)

K-Medoid Clustering

- **Iteration 1**
- Suppose that we want to group the above dataset into two clusters. So, we will randomly choose two medoids.
- Here, the choice of medoids is important for efficient execution. Hence, we have selected two points from the dataset that can be potential medoid for the final clusters. Following are two points from the dataset that we have selected as medoids.
- M1 = (3, 4)
- M2 = (7, 3)
- Now, we will calculate the distance between each data point and the medoids using the Manhattan distance measure. The results have been tabulated as follows.

MIT-WPU
॥ विश्वशान्तिर्धुवं ध्रुवा ॥

K-Medoid Clustering



Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (7,3)	Assigned Cluster
A1	(2, 6)	3	8	Cluster 1
A2	(3, 8)	4	9	Cluster 1
A3	(4, 7)	4	7	Cluster 1
A4	(6, 2)	5	2	Cluster 2
A5	(6, 4)	3	2	Cluster 2
A6	(7, 3)	5	0	Cluster 2
A7	(7,4)	4	1	Cluster 2
A8	(8, 5)	6	3	Cluster 2
A9	(7, 6)	6	3	Cluster 2
A10	(3, 4)	0	5	Cluster 1

Iteration-1

K-Medoid Clustering

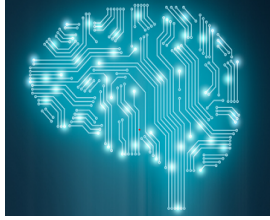
The clusters made with medoids (3, 4) and (7, 3) are as follows.

Points in cluster1= {(2, 6), (3, 8), (4, 7), (3, 4)}

Points in cluster 2= {(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}

After assigning clusters, we will calculate the cost for each cluster and find their sum. The cost is nothing but the sum of distances of all the data points from the medoid of the cluster they belong to.

Hence, the cost for the current cluster will be 3+4+4+2+2+0+1+3+3+0=22.



K-Medoid Clustering

Iteration 2

Now, we will select another non-medoid point (7, 4) and make it a temporary medoid for the second cluster. Hence,

M1 = (3, 4)

M2 = (7, 4)

Now, let us calculate the distance between all the data points and the current medoids.

MIT-WPU
॥ विश्वशान्तिर्धुवं ध्रुवा ॥

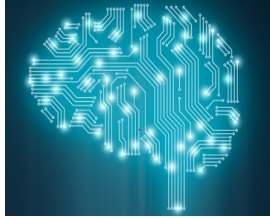
K-Medoid Clustering

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (7,4)	Assigned Cluster
A1	(2, 6)	3	7	Cluster 1
A2	(3, 8)	4	8	Cluster 1
A3	(4, 7)	4	6	Cluster 1
A4	(6, 2)	5	3	Cluster 2
A5	(6, 4)	3	1	Cluster 2
A6	(7, 3)	5	1	Cluster 2
A7	(7,4)	4	0	Cluster 2
A8	(8, 5)	6	2	Cluster 2
A9	(7, 6)	6	2	Cluster 2
A10	(3, 4)	0	4	Cluster 1

K-Medoid Clustering

The data points haven't changed in the clusters after changing the medoids.
Hence, clusters are:
Points in cluster1:{(2, 6), (3, 8), (4, 7), (3, 4)}
Points in cluster 2:{(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}
Now, let us again calculate the cost for each cluster and find their sum. The total cost this time will be 3+4+4+3+1+1+0+2+2+0=20.
Here, the current cost is less than the cost calculated in the previous iteration. Hence, we will make the swap permanent and make (7,4) the medoid for cluster 2. If the cost this time was greater than the previous cost i.e. 22, we would have to revert the change. New medoids after this iteration are (3, 4) and (7, 4) with no change in the clusters.

K-Medoid Clustering

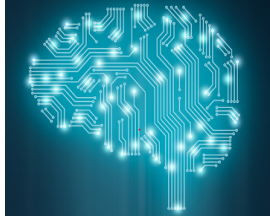


Iteration 3

Now, let us again change the medoid of cluster 2 to (6, 4). Hence, the new medoids for the clusters are M1=(3, 4) and M2= (6, 4).

Let us calculate the distance between the data points and the above medoids to find the new cluster. The results have been tabulated as follows.

MIT-WPU
॥ विश्वशान्तिर्धुवं ध्रुवा ॥



K-Medoid Clustering

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (6,4)	Assigned Cluster
A1	(2, 6)	3	6	Cluster 1
A2	(3, 8)	4	7	Cluster 1
A3	(4, 7)	4	5	Cluster 1
A4	(6, 2)	5	2	Cluster 2
A5	(6, 4)	3	0	Cluster 2
A6	(7, 3)	5	2	Cluster 2
A7	(7,4)	4	1	Cluster 2
A8	(8, 5)	6	3	Cluster 2
A9	(7, 6)	6	3	Cluster 2
A10	(3, 4)	0	3	Cluster 1

Again, the clusters haven't changed. Hence, clusters are:

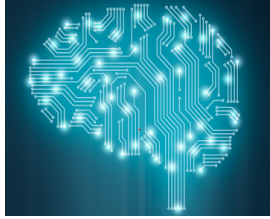
Points in cluster1:{(2, 6), (3, 8), (4, 7), (3, 4)}

Points in cluster 2:{(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}

Now, let us again calculate the cost for each cluster and find their sum. The total cost this time will be 3+4+4+2+0+2+1+3+3+0=22.

The current cost is 22 which is greater than the cost in the previous iteration i.e. 20. Hence, we will revert the change and the point (7, 4) will again be made the medoid for cluster 2.

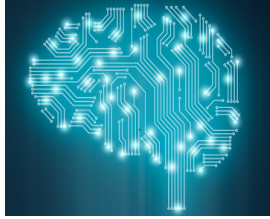
So, the clusters after this iteration will be cluster1 = {(2, 6), (3, 8), (4, 7), (3, 4)} and cluster 2= {(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}. The medoids are (3,4) and (7,4).



Replacing the medoids with a non-medoid data point. The set of medoids for which the cost is the least, the medoids, and the associated clusters are made permanent. So, after all the iterations, you will get the final clusters and their medoids.

The K-Medoids clustering algorithm is a computation-intensive algorithm that requires many iterations. In each iteration, we need to calculate the distance between the medoids and the data points, assign clusters, and compute the cost. Hence, K-Medoids clustering is not well suited for large data sets.

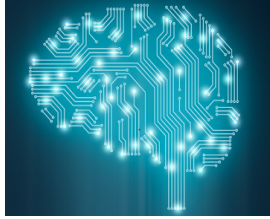
MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥



Advantages of K-Medoids Clustering

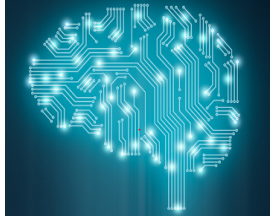
- K-Medoids clustering is a simple iterative algorithm and is very easy to implement.
- K-Medoids clustering is guaranteed to converge. Hence, we are guaranteed to get results when we perform k-medoids clustering on any dataset.
- K-Medoids clustering doesn't apply to a specific domain. Owing to the generalization, we can use k-medoids clustering in different machine learning applications ranging from text data to Geo-spatial data and financial data to e-commerce data.
- The medoids in k-medoids clustering are selected randomly. We can choose the initial medoids in a way such that the number of iterations can be minimized. To improve the performance of the algorithm, you can warm start the choice of medoids by selecting specific data points as medoids after data analysis.
- Compared to other partitioning clustering algorithms such as K-median and k-modes clustering, the k-medoids clustering algorithm is faster in execution.
- K-Medoids clustering algorithm is very robust and it effectively deals with outliers and noise in the dataset. Compared to k-means clustering, the k-medoids clustering algorithm is a better choice for analyzing data with significant noise and outliers.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥



Disadvantages of K-Medoids Clustering

- K-Medoids clustering is not a scalable algorithm. For large datasets, the execution time of the algorithm becomes very high. Due to this, you cannot use the K-Medoids algorithm for very large datasets.
- In K-Medoids clustering, we don't know the optimal number of clusters. Hence, we need to perform clustering using different values of k to find the optimal number of clusters.
- When the dimensionality in the dataset increases, the distance between the data points starts to become similar. Due to this, the distance between a data point and various medoids becomes almost the same. This introduces inefficiency in the execution. To overcome this problem, you can use advanced clustering algorithms like spectral clustering. Alternatively, you can also try to reduce the dimensionality of the dataset while data preprocessing.
- K-Medoids clustering algorithm randomly chooses the initial medoids. Also, the output clusters are primarily dependent on the initial medoids. Due to this, every time you run the k-medoids clustering algorithm, you will get unique clusters.
- The K-Medoids clustering algorithm uses the distance from a central point (medoid). Due to this, the k-medoids algorithm gives circular/spherical clusters. This algorithm is not useful for clustering data into arbitrarily shaped clusters.



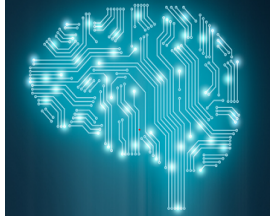
Comparison of K Means and K Medoids

Centroid vs. Medoid

- **K-Means**: Uses the mean of the points in a cluster as the centroid, which may not be an actual data point.
- **K-Medoids**: Uses actual data points as medoids, making it more interpretable.

Distance Measures

- **K-Means**: Typically uses Euclidean distance, which may not be suitable for all data types.
- **K-Medoids**: Can use any distance measure, providing more flexibility.



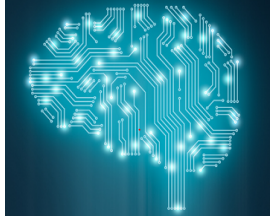
Comparison of K Means and K Medoids

Sensitivity to Outliers

- **K-Means**: Sensitive to outliers, as they can significantly affect the mean.
- **K-Medoids**: More robust to outliers, as medoids are actual data points less influenced by extreme values.

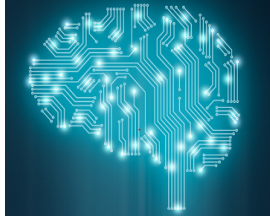
Computational Complexity

- **K-Means**: Generally faster and more efficient, making it suitable for very large datasets.
- **K-Medoids**: Slower due to the need to evaluate all possible swaps, better for smaller datasets or when robustness is crucial.



Difference between K means and K medoids Clustering

Aspect	K -Means Clustering	K-Medoids Clustering
Representation of Clusters	K-Means Clustering uses the mean of points (centroid) to represent a cluster.	It uses the most centrally located point (medoid) to represent a cluster.
Sensitivity to Outliers	Highly sensitive to outliers.	More robust to outliers.
Distance Metrics	K-Means primarily uses Euclidean distance.	Whereas it can use any distance metric.
Computational Efficiency	K-Means is generally faster and more efficient	It is slower due to the need to calculate all pairwise distances within clusters.
Cluster Shape Assumption	It assumes spherical clusters.	It does not make strong assumptions about cluster shapes.



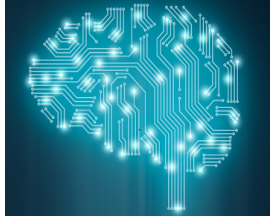
Density Based Clustering-DBScan

DBSCAN is a density-based algorithm.

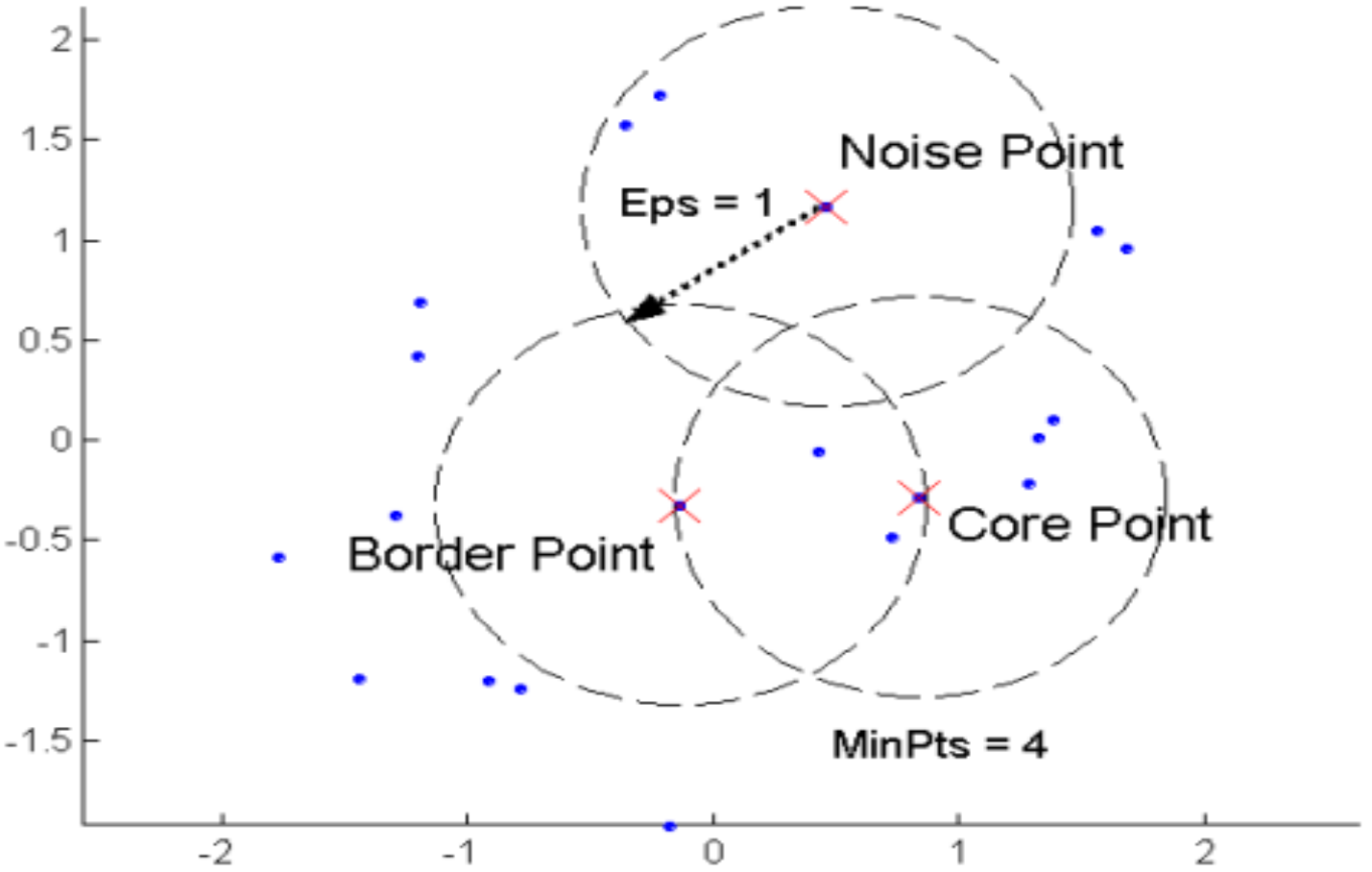
- Density = number of points within a specified radius r (Eps)
- A point is a core point if it has more than a specified number of points (MinPts) within Eps

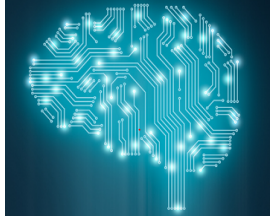
These are points that are at the interior of a cluster

- 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.



DBSCAN: Core, Border, and Noise points





Two parameters (eps and MinPts)***:***

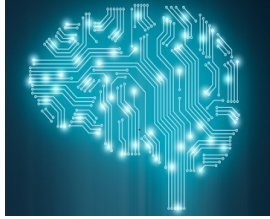
- ***ε***: Maximum radius of the neighbourhood
- ***MinPts***: Minimum number of points in an Eps-neighbourhood of that point
- $N_{\varepsilon}(p)$: ***{q belongs to D | dist(p,q) <= ε}***

Directly density-reachable**:** A point ***p*** is directly density-reachable from a point ***q*** wrt. ***ε***, ***MinPts*** if

1) ***p*** belongs to $N_{\varepsilon}(q)$

2) core point condition:

$|N_{\varepsilon}(q)| >= MinPts$



DBSCAN: Algorithm

Let ClusterCount=0. For every point *p*:

1. If *p* it is not a core point, assign a null label to it [e.g., zero]
2. If *p* is a core point, a new cluster is formed [with label ClusterCount:= ClusterCount+1]

   Then find all points density-reachable from p and classify them in the cluster.
   [Reassign the zero labels but not the others]

Repeat this process until all of the points have been visited.

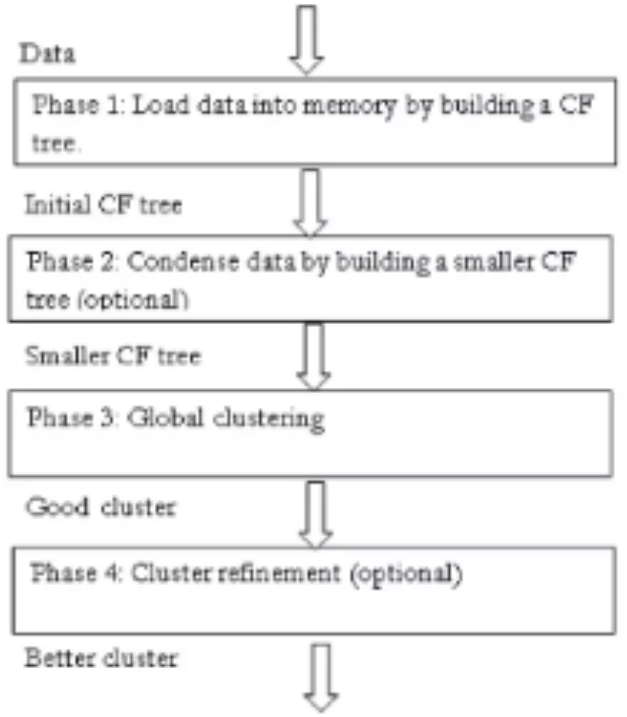
Since all the zero labels of border points have been reassigned in 2, the remaining points with zero label are noise.

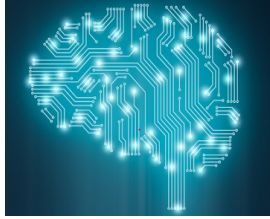
BIRCH clustering

BIRCH (*balanced iterative reducing and clustering using hierarchies*) is an unsupervised data-mining algorithm perform hierarchical-clustering over particularly large data-sets.

- The BIRCH algorithm takes **as input a set of N data points, represented as real-valued vectors, and a desired number of clusters K**. It operates in four phases, the second of which is optional. tree, while removing outliers and grouping crowded subclusters into larger ones.

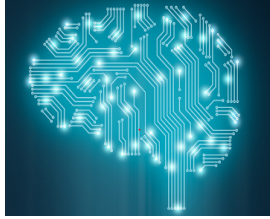
- Phase 1: Load data into memory
  Scan DB and load data into memory by building a CF tree. If memory is exhausted rebuild the tree from the leaf node.
- Phase 2: Condense data
  Resize the data set by building a smaller CF tree
  Remove more outliers
  Condensing is optional
- Phase 3: Global clustering
  Use existing clustering algorithm (e.g. KMEANS, HC) on CF entries
- Phase 4: Cluster refining
  Refining is optional
  Fixes the problem with CF trees where same valued data points may be assigned to different leaf entries.





BIRCH clustering…goals

- Minimize running time and data scans, thus formulating the problem for large databases
- Clustering decisions made without scanning the whole data
- Exploit the non uniformity of data – treat dense areas as one, and remove outliers (noise)

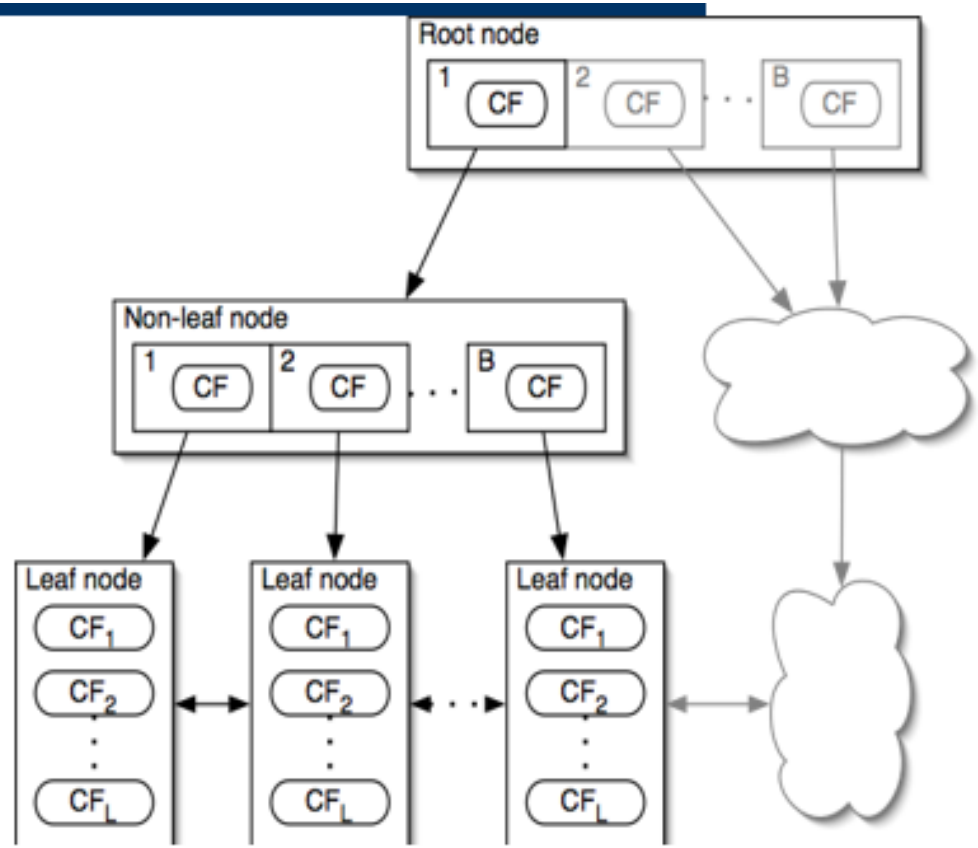


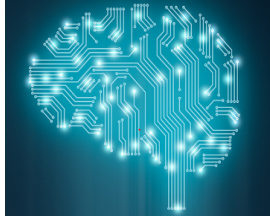
BIRCH clustering…Features

- CF is a compact storage for data on points in a cluster
- Has enough information to calculate the intra-cluster distances
- Additivity theorem allows us to merge sub-clusters

BIRCH clustering…Properties of CF tree

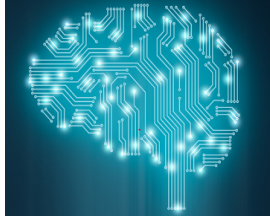
- Each non-leaf node has at most B entries
- Each leaf node has at most L CF entries which each satisfy threshold T
- Node size is determined by dimensionality of data space and input parameter P (page size)





BIRCH clustering…CF tree insertion

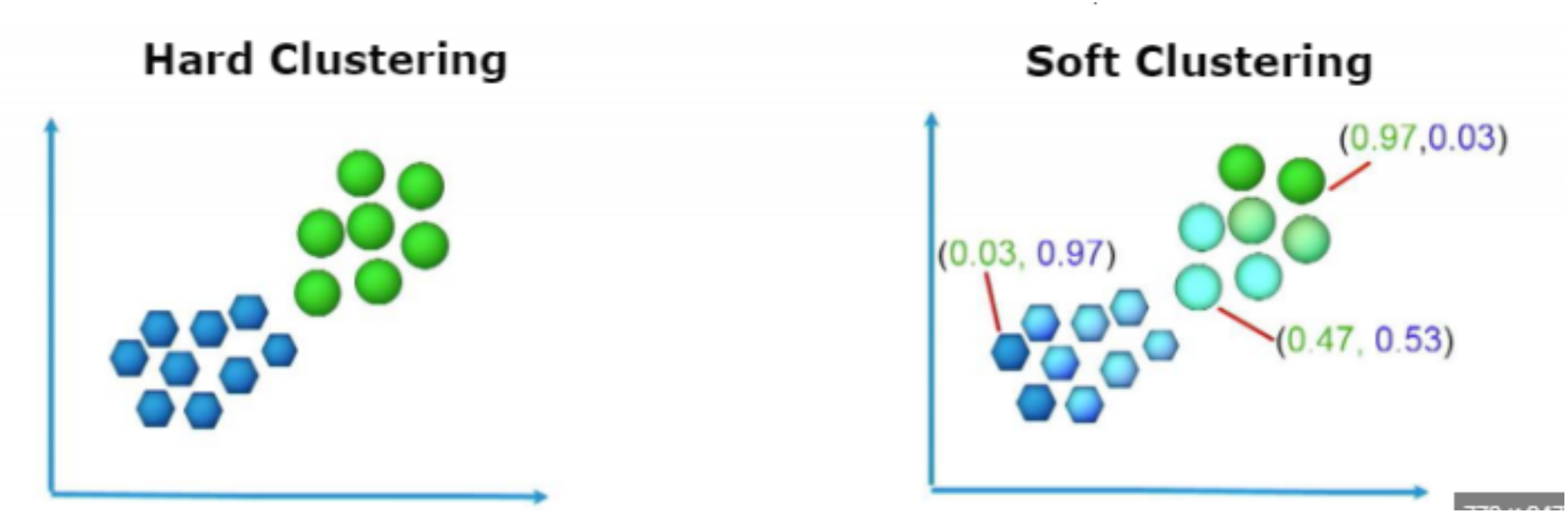
- Identifying the appropriate leaf: recursively descending the CF tree and choosing the closest child node according to a chosen distance metric
- Modifying the leaf: test whether the leaf can absorb the node without violating the threshold. If there is no room, split the node
- Modifying the path: update CF information up the path.



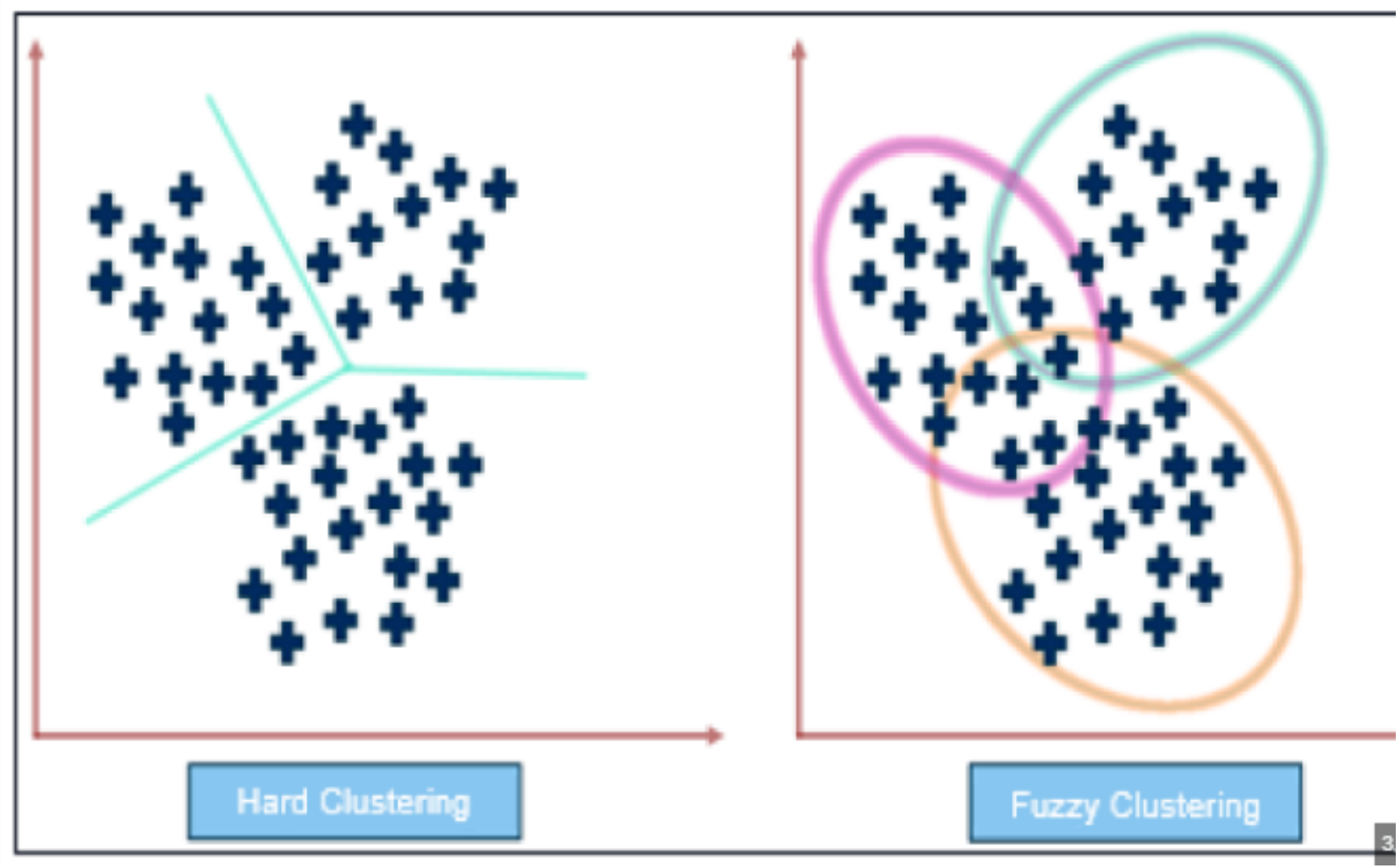
BIRCH clustering…Algorithm

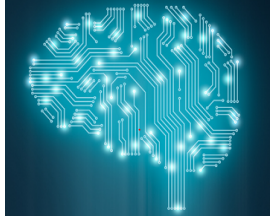
- Phase 1: Scan all data and build an initial in-memory CF tree.
- Phase 2: condense into desirable length by building a smaller CF tree.
- Phase 3: Global clustering
- Phase 4: Cluster refining – this is optional, and requires more passes over the data to refine the results

Fuzzy Clusters



Fuzzy Clusters





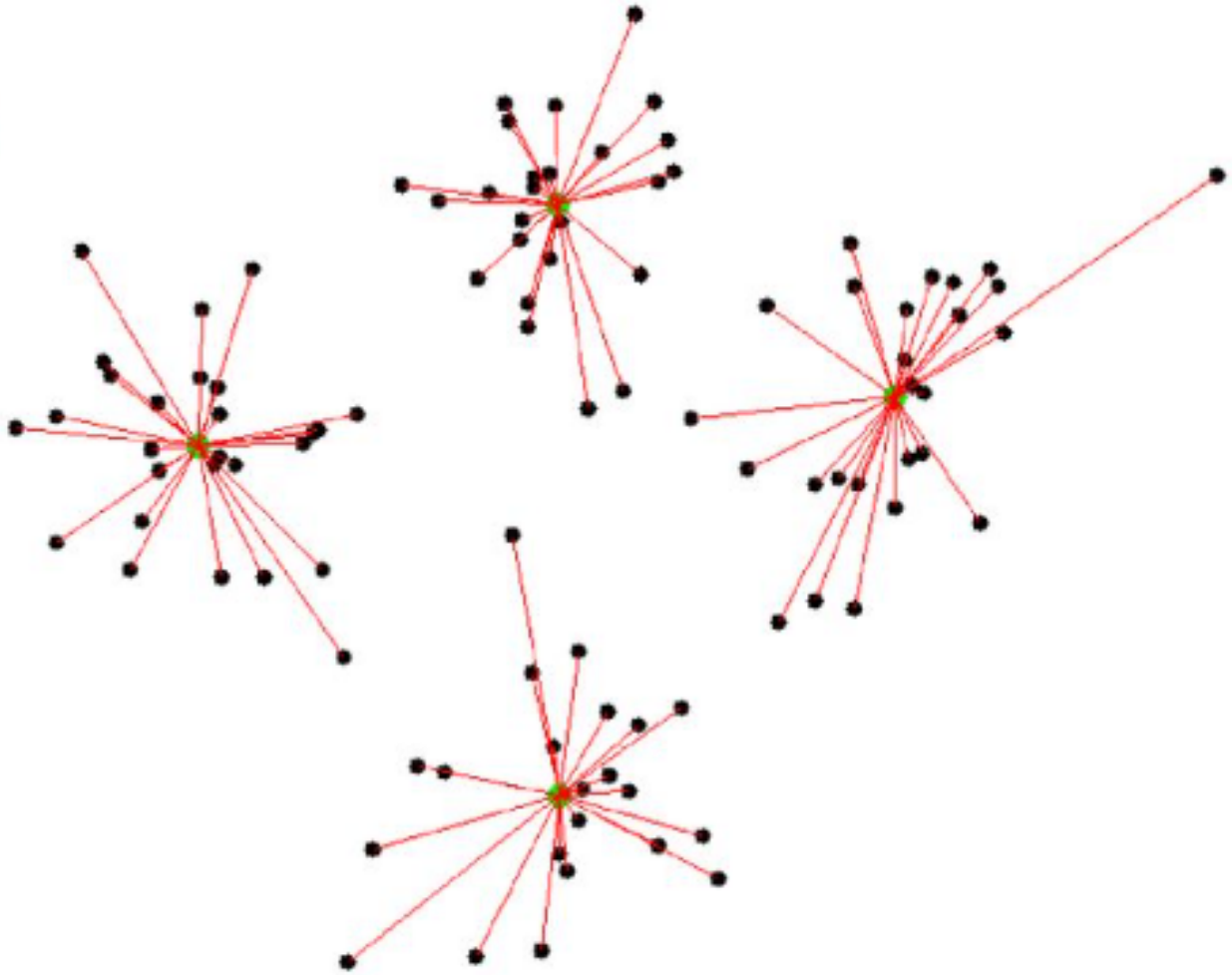
Crisp & Fuzzy Clustering

CRISP	**Each point belongs to exactly one cluster.** C-Means Clustering
FUZZY	**Cluster membership is a matter of degree.** Fuzzy C-Means Clustering (FCM)

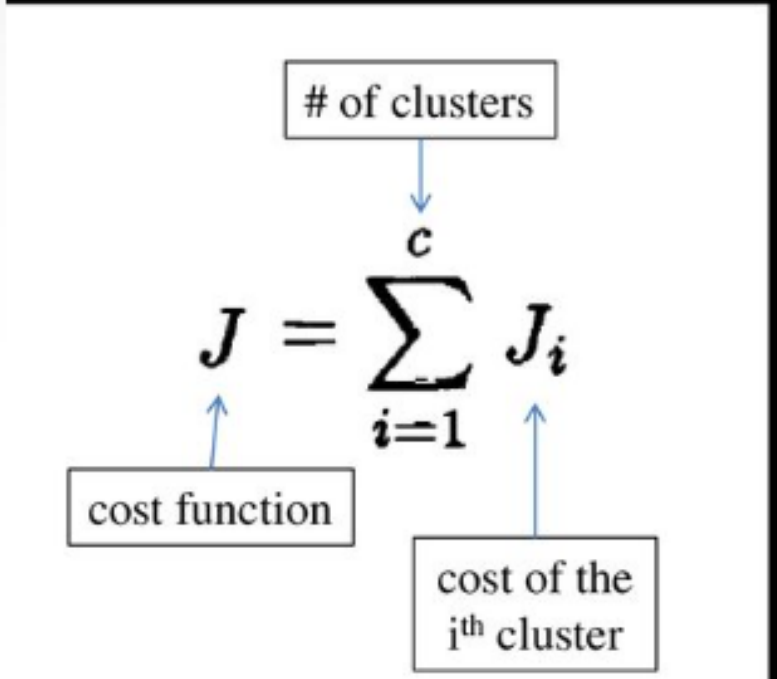
C-Means Clustering

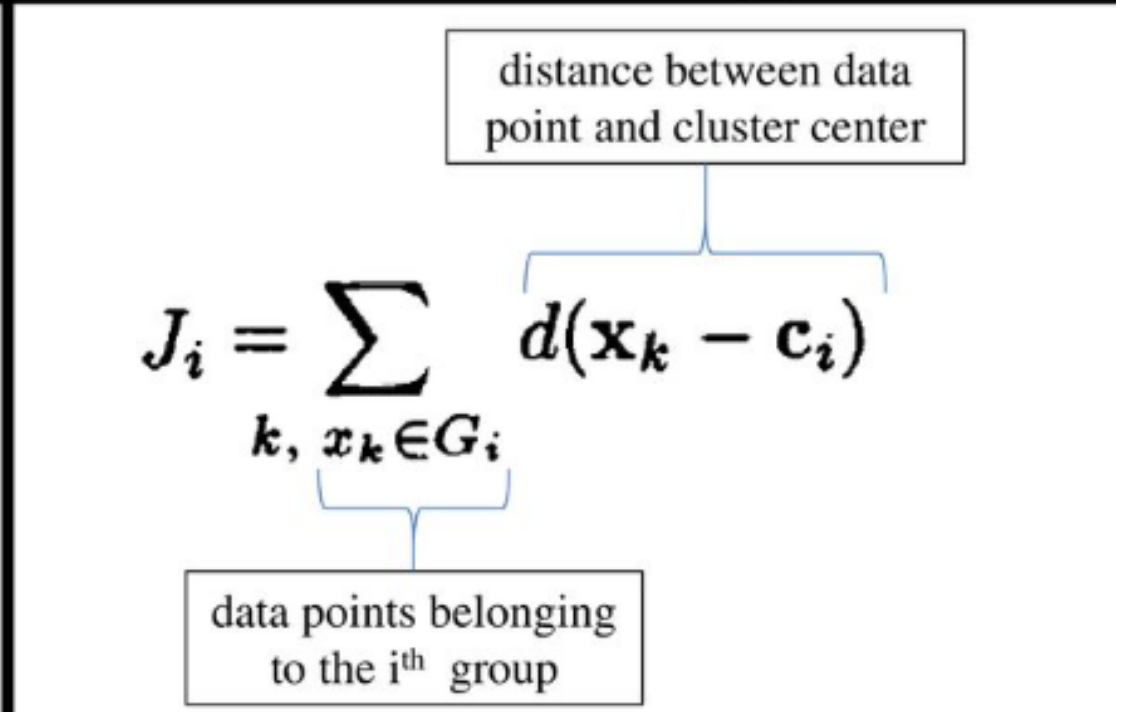
Fixed number of clusters. One centroid per cluster.

Each data point belongs to the cluster corresponding to the closest centroid.



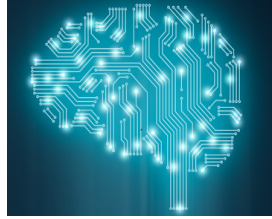
C-Means Clustering





$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k,\ x_k \in G_i} d(\mathbf{x}_k - \mathbf{c}_i) \right)$$

C-Means Clustering



pick c centroids at random

assign each data point to the cluster corresponding to the nearest centroid.

move each centroid to the mean value of its cluster's data points.

$$\mathbf{c}_i = \frac{1}{|G_i|} \sum_{k, \mathbf{x}_k \in G_i} \mathbf{x}_k$$

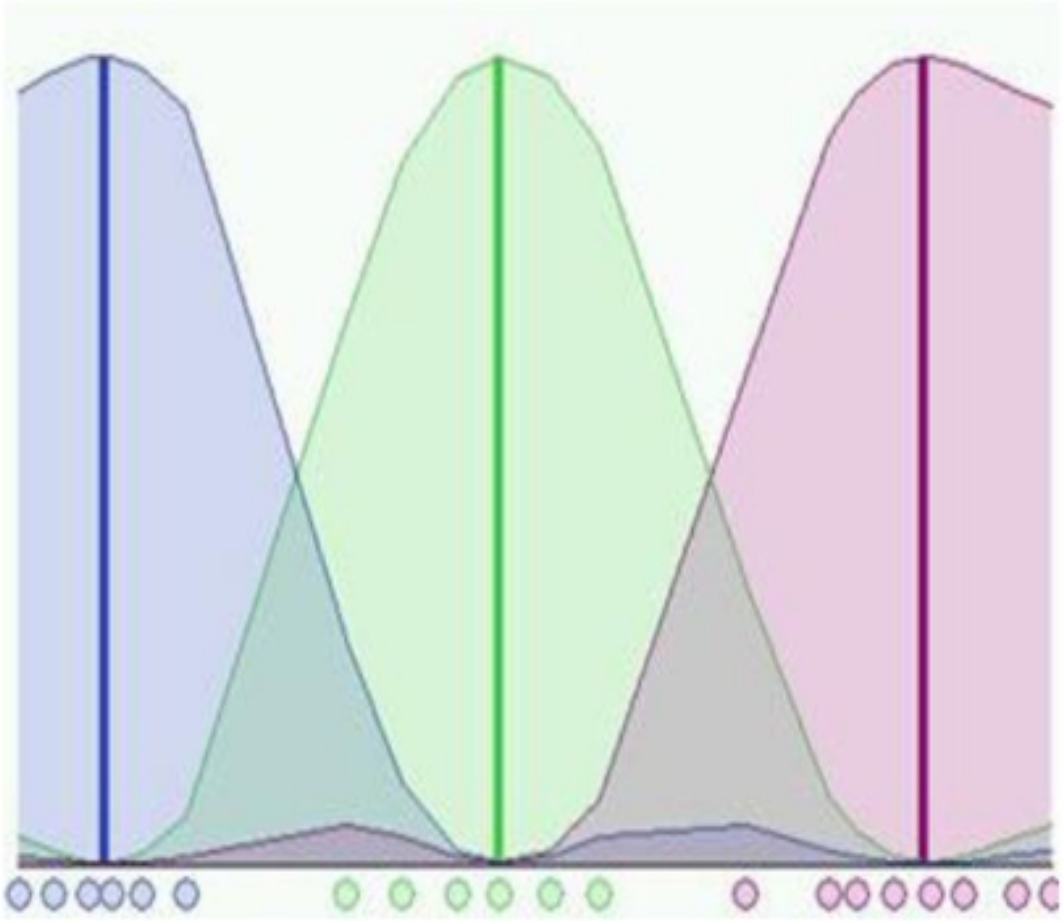
Fuzzy C-Means Clustering

Fixed number of clusters.
One centroid per cluster.

Clusters are fuzzy sets.

Membership degree of a point can be any number between 0 and 1.

Sum of all degrees for a point must add up to 1.



Fuzzy C-Means Clustering

C-Means	$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k,\ x_k \in G_i} d(\mathbf{x}_k - \mathbf{c}_i) \right)$
Fuzzy C-Means (FCM)	$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^{m} d_{ij}^{2}$ summing over all data points fuzziness exponent membership degree

Fuzzy C-Means Clustering

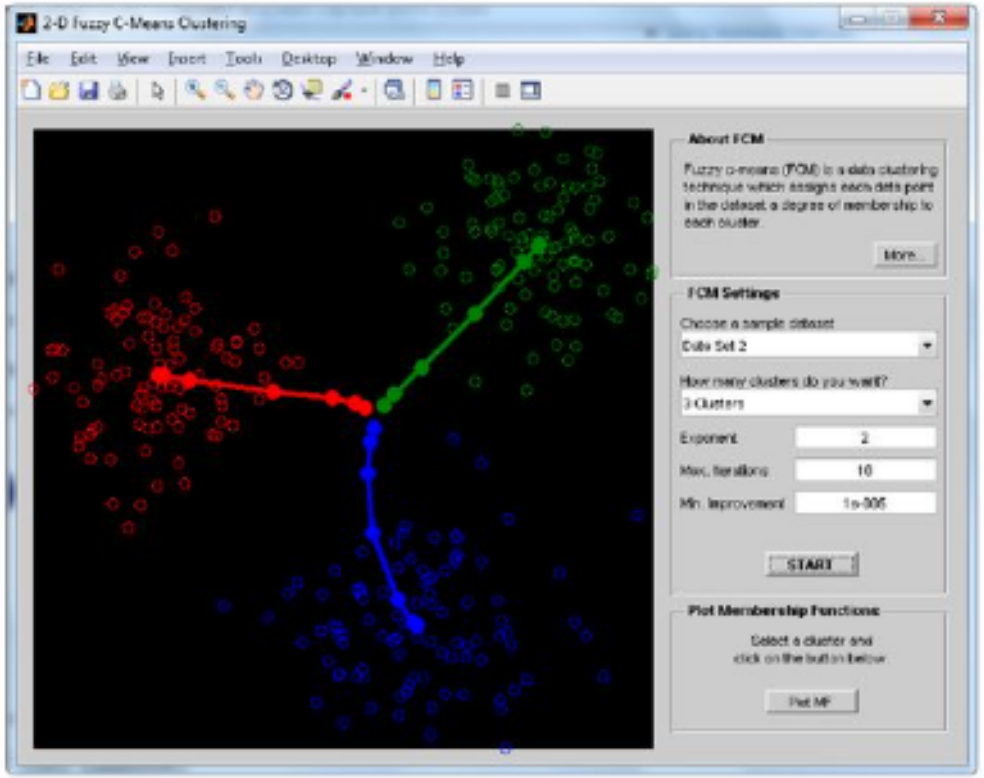
pick c centroids at random

assign membership degrees according to:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}}$$

move each centroid to the following position:

$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} \mathbf{x}_j}{\sum_{j=1}^{n} u_{ij}^{m}}$$



Note: formulas are result of the method of Lagrange multipliers as applied to aforementioned cost function

Fuzzy C-Means Clustering

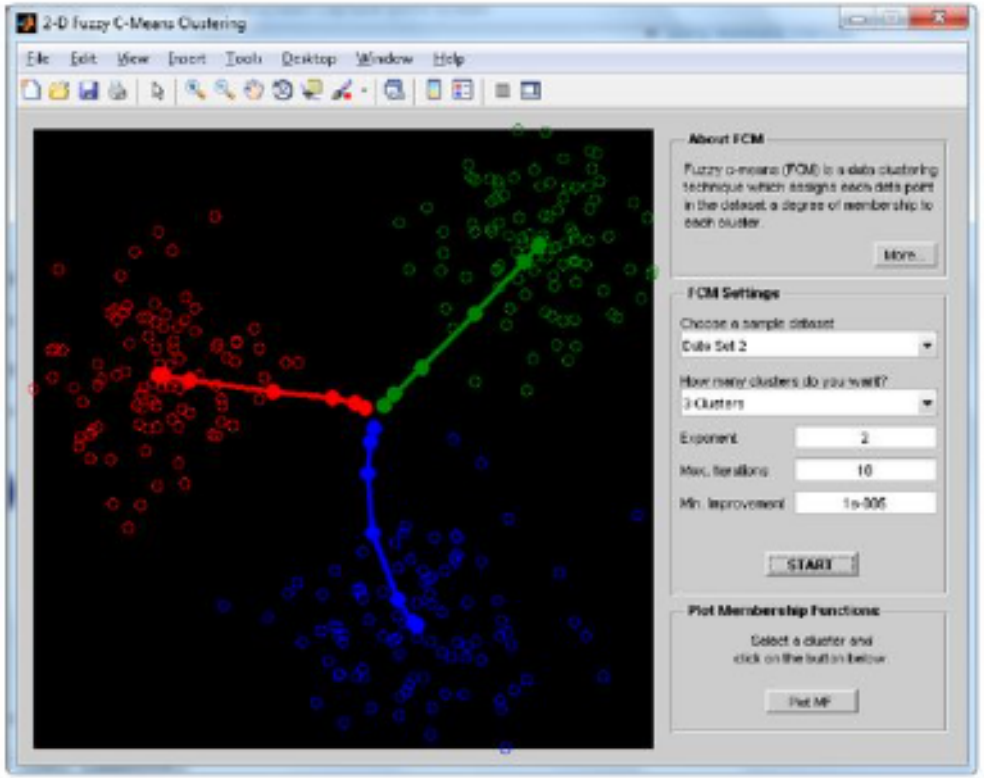
pick c centroids at random

assign membership degrees according to:

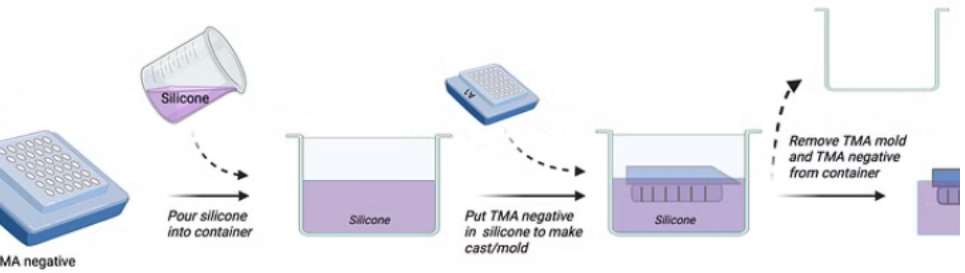
$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}}$$

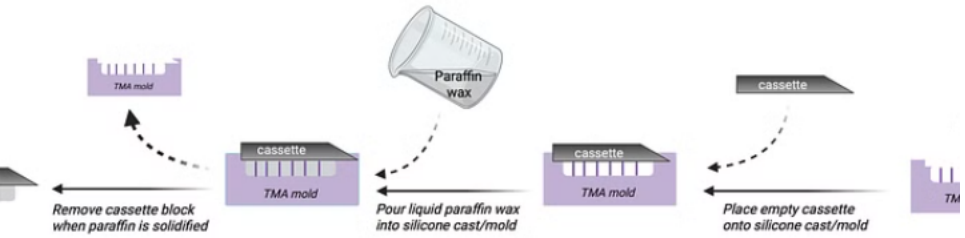
move each centroid to the following position:

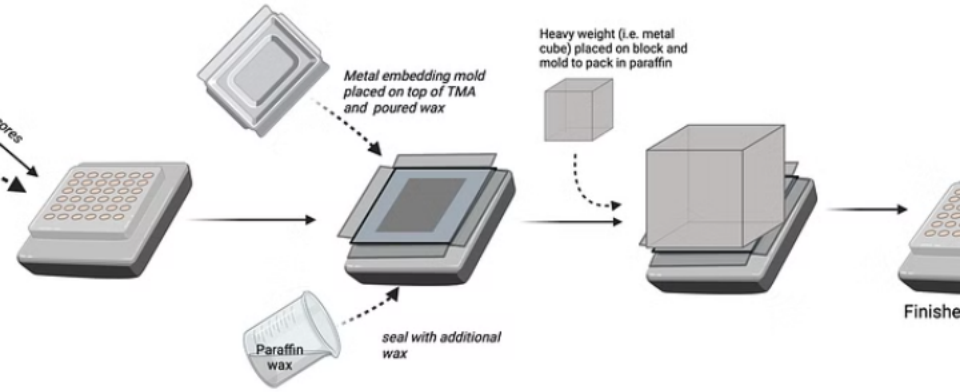
$$\mathbf{c}_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} \mathbf{x}_j}{\sum_{j=1}^{n} u_{ij}^{m}}$$

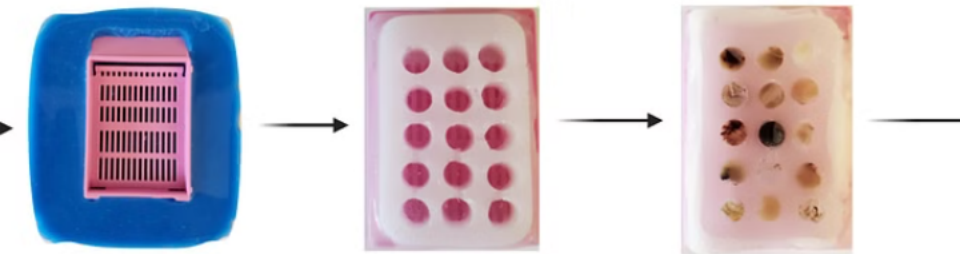


Note: formulas are result of the method of Lagrange multipliers as applied to aforementioned cost function







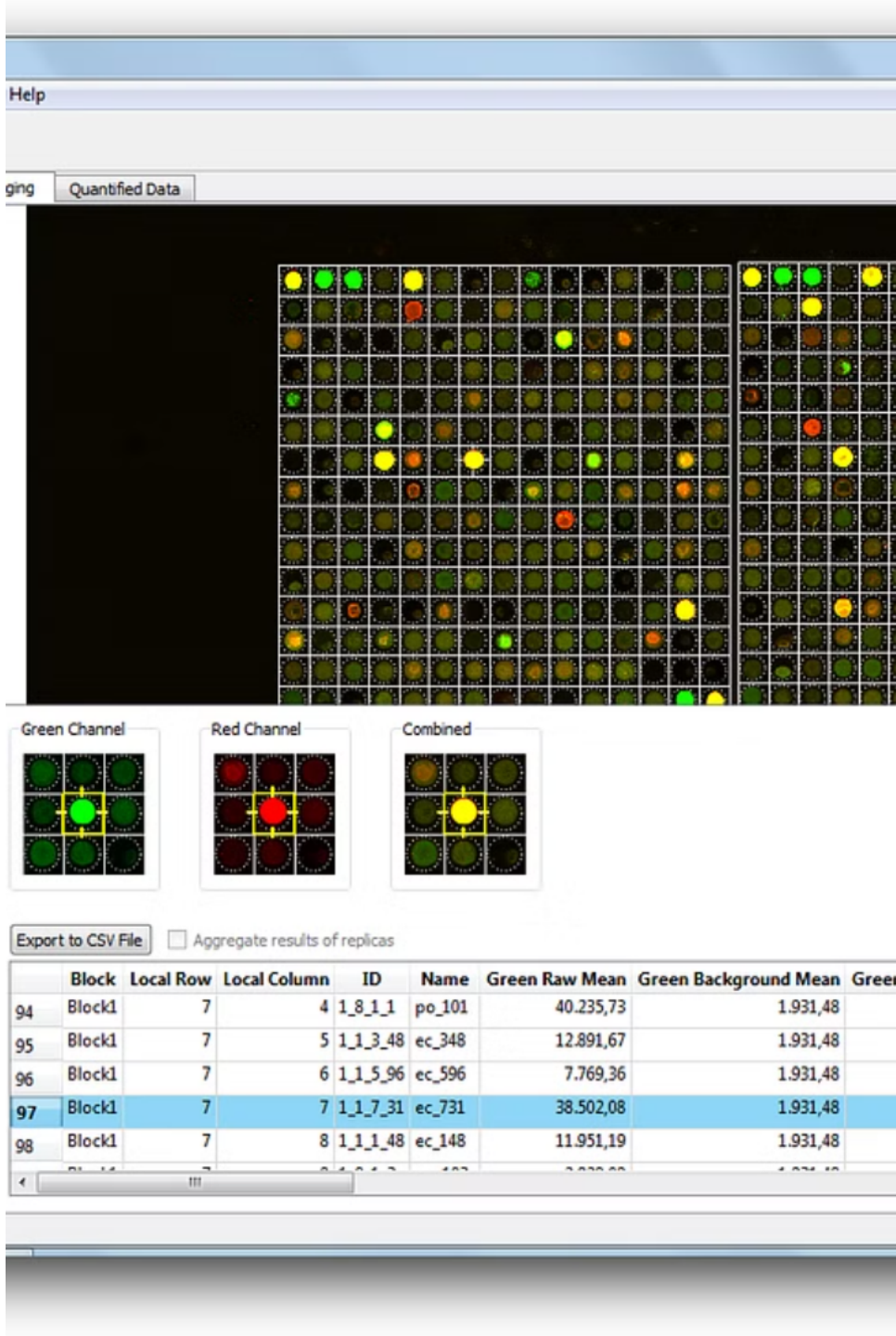


Overview of Tissue Microarray Technology

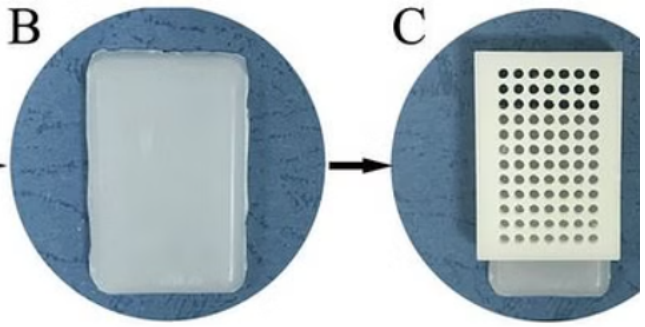
Tissue microarray (TMA) is an innovative technology that allows for high-throughput analysis of tissue samples. In a TMA, hundreds of tiny tissue cores are precisely arrayed on a single paraffin block, enabling **simultaneous immunohistochemical or in situ hybridization analysis** of multiple patient samples on a single slide.

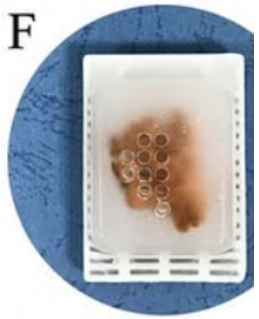
Role of Machine Learning in Tumor Microarray Analysis

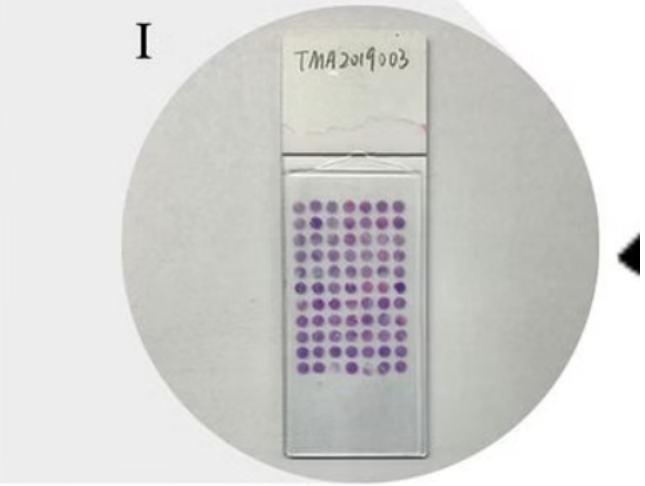
Machine learning algorithms have become invaluable tools for analyzing the vast amounts of data generated by tumor microarray technologies. These powerful techniques can uncover complex patterns and relationships within the data, enabling researchers to better understand tumor biology and develop more targeted therapies.



	Block	Local Row	Local Column	ID	Name	Green Raw Mean	Green Background Mean
94	Block1	7	4	1_8_1_1	po_101	40.235,73	1.931,48
95	Block1	7	5	1_1_3_48	ec_348	12.891,67	1.931,48
96	Block1	7	6	1_1_5_96	ec_596	7.769,36	1.931,48
97	Block1	7	7	1_1_7_31	ec_731	38.502,08	1.931,48
98	Block1	7	8	1_1_1_48	ec_148	11.951,19	1.931,48







Algorithms for Tumor Microarray Analysis

1

Image Segmentation

Applying image segmentation algorithms to isolate individual tissue cores and separate them from the background.

2

Feature Extraction

Extracting quantitative features from the tissue cores, such as staining intensity, texture, and morphological properties.

3

Supervised Classification

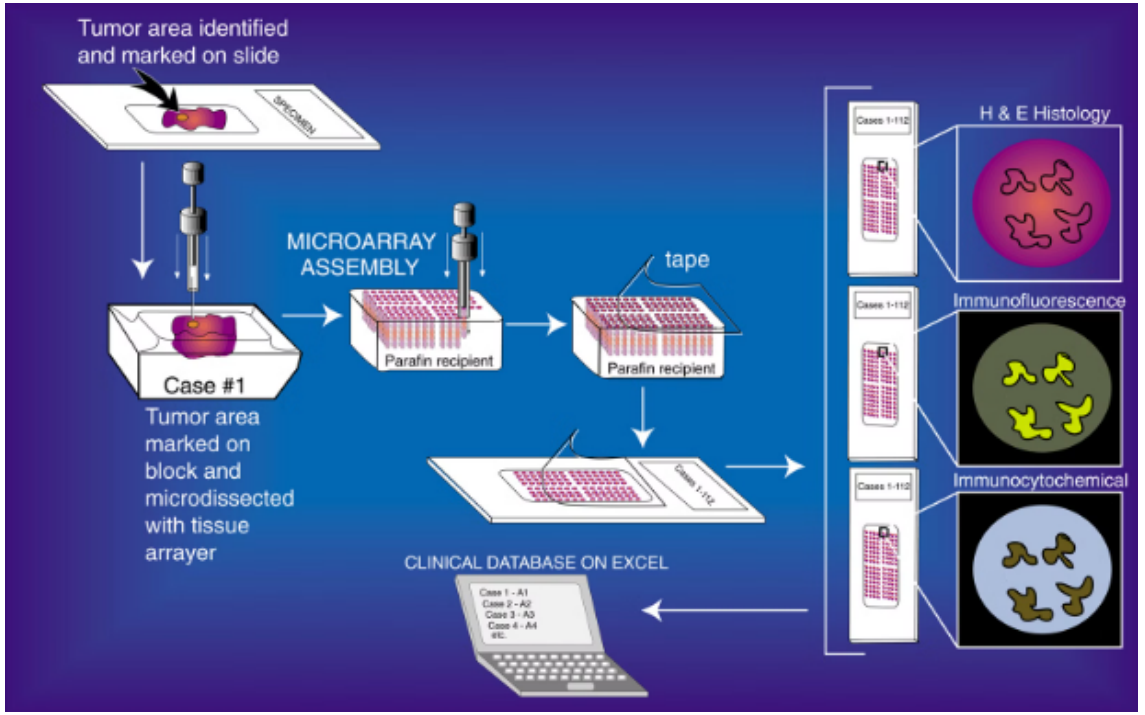
Using machine learning algorithms like decision trees, random forests, and support vector machines to classify tumor samples into different subtypes or grades.

Image Processing and Feature Extraction

Image processing plays a crucial role in analyzing tissue microarray data. Key steps include image segmentation, cell detection, and feature extraction.

Sophisticated algorithms extract quantitative measurements from the images, such as cell counts, protein expression levels, and spatial distributions.

These image-derived features serve as inputs to machine learning models for tumor classification and subtyping.



Supervised Learning Techniques for Tumor Classification

Logistic Regression

Predicts tumor type based on features

Support Vector Machines

Identifies optimal hyperplane to separate tumor classes

Random Forests

Ensemble of decision trees for robust classification

Supervised learning algorithms are powerful tools for classifying tumor types based on the microarray data. Logistic regression, support vector machines, and random forests are commonly used techniques that can accurately predict tumor subtypes by learning from labeled training data. These models excel at identifying complex patterns in the high-dimensional tumor profiles.

Unsupervised Clustering Algorithms for Tumor Subtyping

Unsupervised clustering algorithms play a crucial role in identifying distinct tumor subtypes within a heterogeneous tumor microarray dataset. These algorithms can uncover hidden patterns and groupings without relying on pre-defined labels or classes.

K-Means Clustering	Groups tumors into k distinct clusters based on similarity of gene expression profiles. Useful for identifying broad tumor subtypes.
Hierarchical Clustering	Builds a hierarchy of clusters, allowing visualization of relationships between tumor samples. Helps identify subtypes and their hierarchical structure.
Gaussian Mixture Models	Assumes data is generated from a mixture of Gaussian distributions. Can identify overlapping tumor subtypes with probabilistic assignments.

These algorithms uncover hidden structures in the tumor microarray data, enabling researchers to identify novel tumor subtypes that may have distinct molecular profiles and clinical outcomes. The discovered subtypes can then be further studied and validated using supervised learning techniques.

Applications of Tissue Microarray in Cancer Research

Biomarker Discovery

Tissue microarrays enable rapid screening of tissue samples for potential biomarkers associated with cancer prognosis and treatment response.

Drug Development

Tissue microarrays allow efficient evaluation of drug candidates and their impact on various tumor types, accelerating the drug discovery process.

High-Throughput Analysis

The compact array format enables comprehensive, high-throughput analysis of large cohorts of patient samples, providing valuable insights into tumor biology.

Pathology Validation

Tissue microarrays facilitate the validation of novel immunohistochemical markers and their association with clinicopathological features of cancer.

Advantages of Tissue Microarray

High-Throughput Analysis

Tissue microarray enables the simultaneous analysis of hundreds of tissue samples on a single slide, dramatically increasing the efficiency of cancer research.

Reduced Tissue Consumption

The miniaturized format of tissue microarray requires only small amounts of tissue, preserving precious patient samples for multiple analyses.

Standardized Conditions

Tissue microarray ensures consistent staining and analysis conditions across all samples, improving the reliability and reproducibility of results.

Cost-Effectiveness

The streamlined workflow and reduced tissue requirements of tissue microarray make it a cost-effective tool for cancer research and biomarker discovery.

Disadvantages of Tissue Microarray

Limited Representativeness

Tissue microarrays typically contain small tissue cores from different regions of a tumour or from multiple tumours.

Sampling Bias

The selection of tissue cores for inclusion in a microarray may introduce sampling bias, particularly if cores are selected based on subjective criteria or if certain tissue regions are overrepresented or underrepresented.

Tissue Damage and Degradation

The process of constructing tissue microarrays involves punching small cores from paraffin-embedded tissue blocks.

Loss of Morphological Context

Tissue microarrays sacrifice the spatial context of individual tissue samples, as multiple cores are arranged in a grid-like pattern on a single slide.

Expectation Maximization

STEPS

YOU WOULD FOLLOW

GUESS AND IMPROVE (EXPECTATION STEP)

First, you make a guess about what the missing puzzle pieces might look like. This is like saying, "Hmm, I think the missing pieces could be this color and shape." Your guess doesn't have to be perfect; it's just a starting point.

STEPS

YOU WOULD FOLLOW

MAKE IT BETTER (MAXIMIZATION STEP)

Then, you look at the pieces you have and the ones you guessed. You figure out how to adjust your guess to make it match the pieces you have as closely as possible. This step is like tweaking your guess to fit the puzzle better.

STEPS

YOU WOULD FOLLOW

REPEAT UNTIL DONE

You keep doing these two steps over and over, making your guess better and better each time. It's like refining your guess until the puzzle is complete.

The EM algorithm is like a smart helper that makes educated guesses and keeps improving them until the puzzle is solved. It's great for figuring out things when you don't have all the information you need.

IN

ACTUAL TERMS

The Expectation-Maximization (EM) algorithm is an iterative statistical technique used for estimating parameters of probabilistic models when some of the data is missing or unobserved. EM is particularly useful in situations where you have incomplete or partially observed data, and you want to estimate the underlying hidden variables or parameters of a statistical model.

IN

ACTUAL TERMS

The Expectation-Maximization (EM) algorithm is an iterative optimization method that combines different unsupervised machine learning algorithms to find maximum likelihood or maximum posterior estimates of parameters in statistical models that involve unobserved latent variables.

IN

ACTUAL TERMS

The EM algorithm is commonly used for latent variable models and can handle missing data. It consists of an estimation step (E-step) and a maximization step (M-step), forming an iterative process to improve model fit.

IN

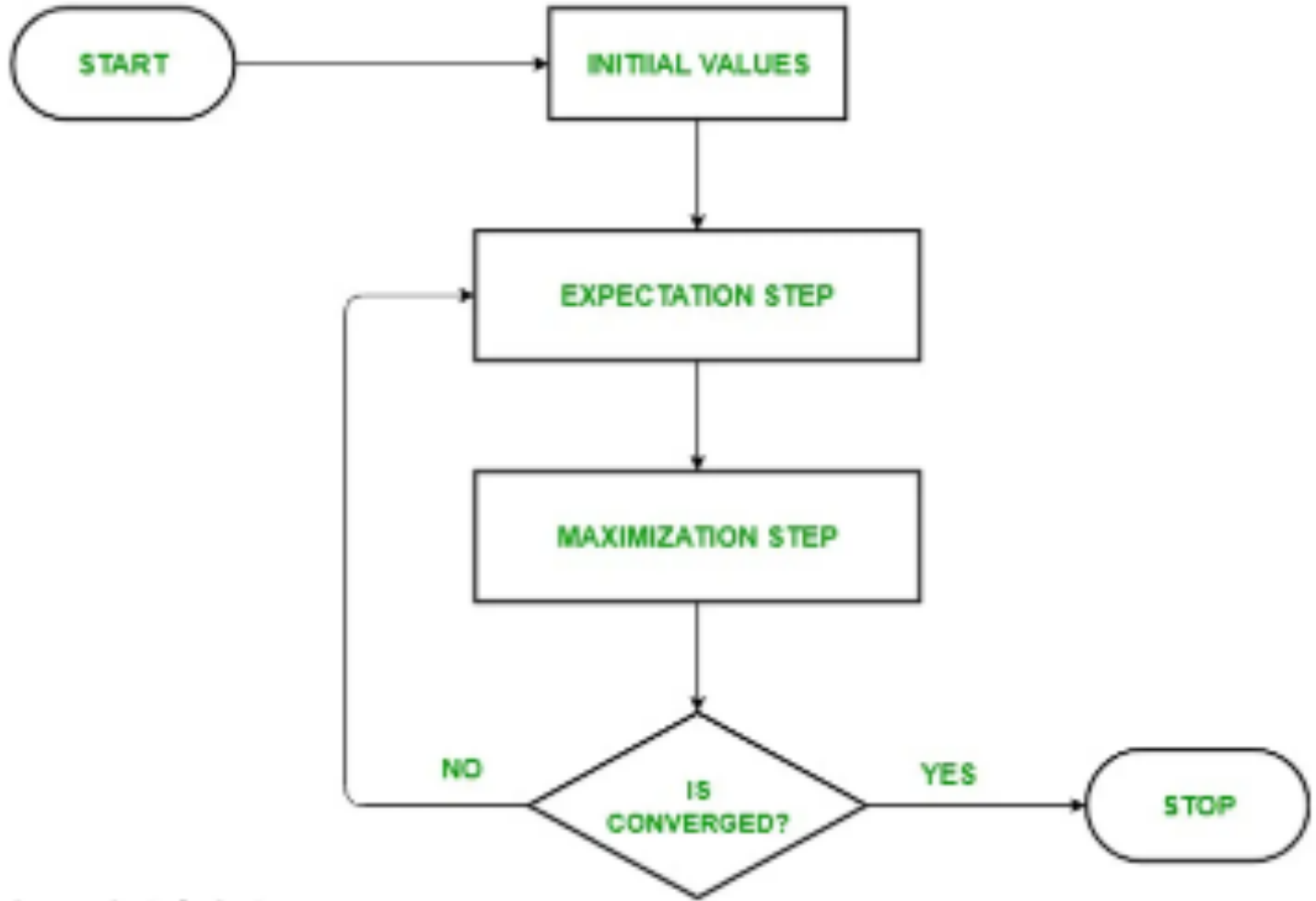
ACTUAL TERMS

- In the E step, the algorithm computes the latent variables i.e. expectation of the log-likelihood using the current parameter estimates.
- In the M step, the algorithm determines the parameters that maximize the expected log-likelihood obtained in the E step, and corresponding model parameters are updated based on the estimated latent variables.

IN

ACTUAL TERMS

By iteratively repeating these steps, the EM algorithm seeks to maximize the likelihood of the observed data. It is commonly used for unsupervised learning tasks, such as clustering, where latent variables are inferred, and has applications in various fields, including machine learning, computer vision, and natural language processing.



Source: GeeksforGeeks

PROBLEM

Imagine you have a bag of colorful candies, but you don't know how many of each color are in the bag. You want to figure this out by using the EM algorithm.

STEP-1 (E STEP)

1. Close your eyes and take out one candy from the bag without looking.
2. Now, you ask your friend to guess the color of the candy.
3. Your friend makes a guess based on their knowledge of candies, but they're not entirely sure because they can't see the candy either. So, they give you their best guess along with how confident they are in their guess.

STEP-2 (M STEP)

1. You collect all the guesses and confidence levels from your friend for the candies you've taken out so far.
2. You count how many times each color was guessed and use the confidence levels to estimate the number of candies of each color in the bag.
3. You adjust your guess of how many candies of each color are in the bag based on this new information.

STEP-3 (REPEAT)

Keep repeating these two steps. Each time you do it, your guess about the candies' colors and amounts gets better and better. After doing this many times, you'll have a very good idea of how many candies of each color are in the bag.

LET'S MAKE IT MATHEMATICAL

Suppose you have a bag with red (R), green (G), and blue (B) candies. You take out one candy at a time and record your friend's guesses. After several candies, you have these guesses:

- For the first candy: 80% chance it's Red, 10% Green, 10% Blue
- For the second candy: 30% Red, 60% Green, 10% Blue
- For the third candy: 20% Red, 10% Green, 70% Blue

LET'S MAKE IT MATHEMATICAL

Suppose you have a bag with red (R), green (G), and blue (B) candies. You take out one candy at a time and record your friend's guesses. After several candies, you have these guesses:

- For the first candy: 80% chance it's Red, 10% Green, 10% Blue
- For the second candy: 30% Red, 60% Green, 10% Blue
- For the third candy: 20% Red, 10% Green, 70% Blue

LET'S MAKE IT MATHEMATICAL

Now, in the M-step, you count the total guesses for each color and update your estimates:

- Red: (0.80 + 0.30 + 0.20) / 3 = 0.43
- Green: (0.10 + 0.60 + 0.10) / 3 = 0.27
- Blue: (0.10 + 0.10 + 0.70) / 3 = 0.30

So, based on these new estimates, you think there are approximately 43% Red candies, 27% Green candies, and 30% Blue candies in the bag.

You repeat this process many times until your estimates become very accurate, and you have a good idea of the candy distribution in the bag. That's how the EM algorithm works to solve problems like this one!

ADVANTAGES

1. Handles data with missing values effectively.
2. Useful for unsupervised learning tasks like clustering.
3. Robust to noisy data.
4. Adaptable to various probabilistic models.
5. Can be applied to large datasets.
6. Estimates model parameters in mixture distributions.
7. Guarantees convergence to a local maximum.
8. Well-founded in statistical theory.
9. Not very sensitive to initial parameter values.
10. Versatile for various machine learning applications.

DISADVANTAGES

1. Sensitive to initial parameter guesses.
2. Slow convergence for high-dimensional data.
3. Limited scalability for very large datasets.
4. Assumes data is generated from a specific model.
5. Convergence is not guaranteed for all cases.
6. Can be computationally intensive for some problems.