Attribute Selection Measures:

Attribute selection measures, also known as feature selection measures, are techniques used to evaluate and select the most relevant and informative features or attributes in a dataset. These measures help in identifying the subset of features that contribute the most to the prediction or classification task at hand. In simple terms, they help us determine which features are the most important in making accurate predictions or classifications.

They are classified into several types they are:

- **Information gain**: This measure calculates the amount of information that is gained by knowing the value of an attribute. The higher the information gain, the more important the attribute is for predicting the target variable.

  The expected information needed to classify a tuple in D is given by:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

Where

Info(D) = entropy of D.

"Pi" is the nonzero probability that an arbitrary tuple in D belongs to class Ci

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

The term $\frac{|D_j|}{|D|}$ acts as the weight of the jth partition.

- **Gini index**: This measure calculates the probability that a randomly chosen data point will be misclassified if it is classified according to the attribute's values. The lower the Gini index, the more important the attribute is for predicting the target variable.

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

- **Gain ratio:** This measure is a combination of information gain and Gini index. It is calculated by dividing information gain by the Gini index. The higher the gain ratio, the more important the attribute is for predicting the target variable.

$$Gain(A) = Info(D) - Info_A(D).$$

Types of Clustering:

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
- Link-based clustering:
  - Objects are often linked together in various ways
  - Massive links can be used to cluster objects: SimRank, LinkClus

Hierarichal clustering:

Hierarchical clustering is a popular algorithm used in data analysis and machine learning to group similar data points together based on their similarities or dissimilarities. It creates a hierarchy of clusters, organizing the data into a tree-like structure known as a dendrogram.

The process of hierarchical clustering can be summarized as follows:

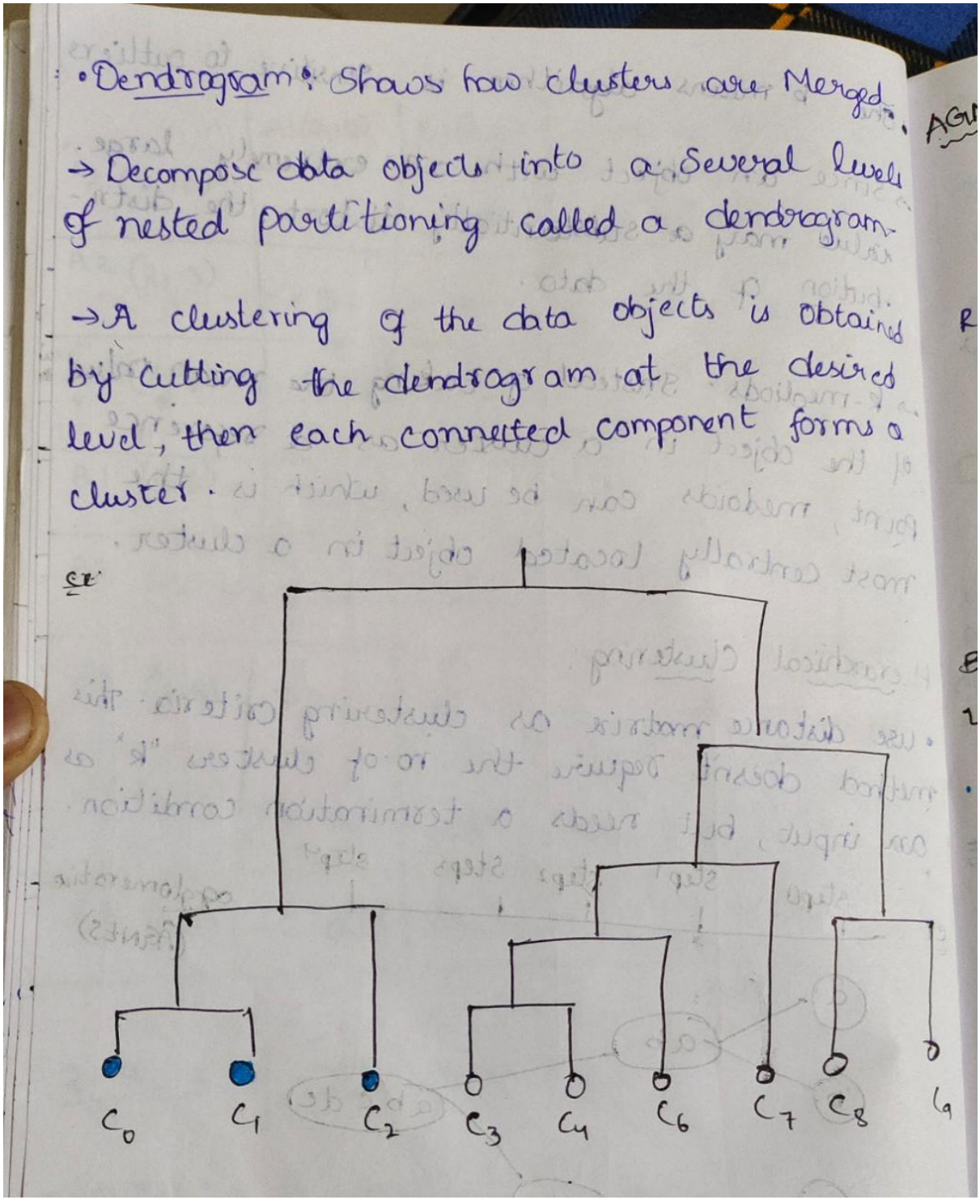
1. Initial Step: Each data point is considered as an individual cluster.

2. Distance Calculation: The algorithm calculates the distance or dissimilarity between each pair of data points. The choice of distance metric depends on the nature of the data (e.g., Euclidean distance for numerical data, Hamming distance for categorical data).

3. Agglomeration: The two closest data points or clusters are merged together to form a new cluster. This step is repeated iteratively until all data points or clusters are combined into a single cluster.

4. Dendrogram Construction: As the merging process continues, a dendrogram is built. The dendrogram represents the hierarchy of clusters and shows the order in which clusters are merged. The vertical axis of the dendrogram represents the dissimilarity between clusters.

5**. Cutting the Dendrogram:** To determine the number of clusters, a cut is made at a specific height on the dendrogram. The height represents a threshold or similarity level. The number of resulting branches below the cut represents the number of clusters.

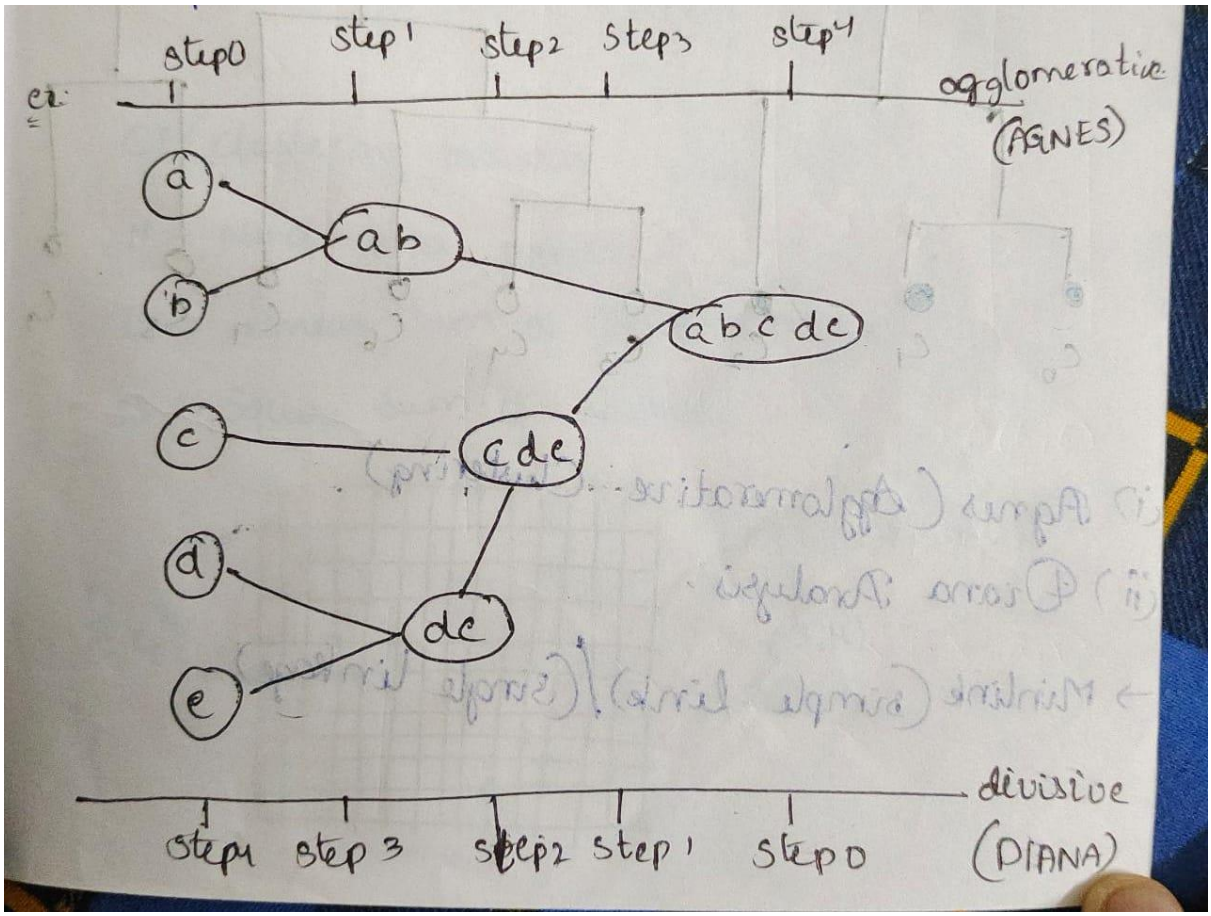


There are two types of hierarchical clustering approaches: agglomerative and divisive.

- **Agglomerative Hierarchical Clustering**: It starts with each data point as a separate cluster and gradually merges the closest clusters until

all data points belong to a single cluster. This bottom-up approach is computationally efficient and widely used.

- **Divisive Hierarchical Clustering**: It begins with all data points in a single cluster and splits them recursively into smaller clusters based on their dissimilarities. This top-down approach can be more computationally intensive and less commonly used.



Hierarchical clustering has several advantages. It does not require specifying the number of clusters in advance, as it generates a dendrogram that allows for visual interpretation and determination of the optimal number of clusters. It can also handle different types of data and is robust to noise and outliers.

However, hierarchical clustering can be computationally expensive for large datasets, and the choice of distance metric and linkage method (criteria for merging clusters) can affect the results. Additionally, it is a deterministic algorithm, meaning it will always produce the same clustering solution given the same input data.

Overall, hierarchical clustering provides a flexible and intuitive approach to discovering structure and patterns within data by organizing them into a hierarchy of clusters.

DBSCAN:

Density-based methods are a type of clustering algorithm that group data points based on their proximity and density. These methods identify areas of high-density regions and separate them from low-density regions. Density-based methods are particularly useful for discovering clusters with arbitrary shapes, handling noise and outliers, and scaling well to large datasets.

One popular density-based clustering algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise). DBSCAN works by defining a dense region as a cluster and a sparser region as noise. The key parameters for DBSCAN are the radius (eps) and the minimum number of points (minPts) required to form a dense

region.

Here's how DBSCAN works:

Finding Core Points:

DBSCAN starts by randomly selecting an unvisited data point and checks if it has at least minPts neighboring points within the radius eps. If so, the point is considered a core point, and all its neighbors are

added to its cluster.

Finding Border Points:

If a data point has fewer than minPts neighbors but belongs to the radius eps of a core point, it is considered a border point. Border points are assigned to the same cluster as their corresponding core point.

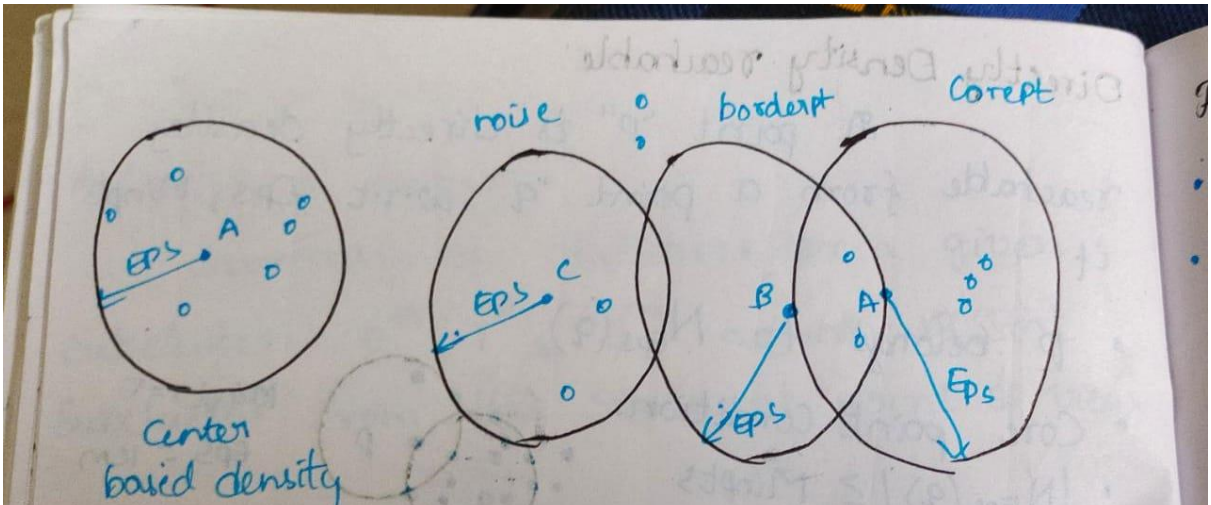
Finding Noise Points:

If a data point does not belong to any core point's radius, it is considered a noise point and is not assigned to any cluster.

Merging Clusters:

DBSCAN repeats the above steps until all points are visited. After the clusters are formed, they may contain outliers or noise. DBSCAN provides an additional step to merge clusters that are close enough, based on their minimum distance.

Example:



Rule-Based Classification

Using IF-THEN Rules for Classification

A rule-based classifier uses a set of IF-THEN rules for classification. An IF-THEN rule is an expression of the form

“IF condition THEN conclusion.”

An example is rule R1,

R1: IF age =youth AND student =yes THEN buys computer = yes.

The “IF” part (or left side) of a rule is known as the rule antecedent or precondition. The “THEN” part (or right side) is the rule consequent. In the rule antecedent, the condition consists of one or more attribute tests (e.g., age = youth and student = yes) that are logically ANDed. The rule’s consequent contains a class prediction (in this case, we are predicting whether a customer will buy a computer). R1 can also be written as

R1: (age = youth) ^ (student = yes)

(buys_computer = yes).

If the condition (i.e., all the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent is satisfied (or simply, that the rule is satisfied) and that the rule covers the tuple.

A rule R can be assessed by its coverage and accuracy. Given a tuple, X, from a classlabeled data set,D, let ncovers be the number of tuples covered by R; n correct be the number of tuples correctly classified by R; and |D|1 be the number of tuples in D. We can define the coverage and accuracy of R as

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

Rule Extraction from a Decision Tree:

To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent ("IF" part). The leaf node holds the class prediction, forming the rule consequent ("THEN" part).

Example 8.7 **Extracting classification rules from a decision tree.** The decision tree of Figure 8.2 can be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree. The rules extracted from Figure 8.2 are as follows:

*R*1: IF *age* = *youth*	AND *student* = *no*	THEN *buys_computer* = *no*
*R*2: IF *age* = *youth*	AND *student* = *yes*	THEN *buys_computer* = *yes*
*R*3: IF *age* = *middle_aged*		THEN *buys_computer* = *yes*
*R*4: IF *age* = *senior*	AND *credit_rating* = *excellent*	THEN *buys_computer* = *yes*
*R*5: IF *age* = *senior*	AND *credit_rating* = *fair*	THEN *buys_computer* = *no*

■

A disjunction (logical OR) is implied between each of the extracted rules. Because the rules are extracted directly from the tree, they are mutually exclusive and exhaustive. Mutually exclusive means that we cannot have rule conflicts here because no two rules will be triggered for the same tuple. (We have one rule per leaf, and any tuple can map to only one leaf.) Exhaustive means there is one rule for each possible attribute–value combination, so that this set of rules does not require a default rule. Therefore, the order of the rules does not matter—they are unordered.

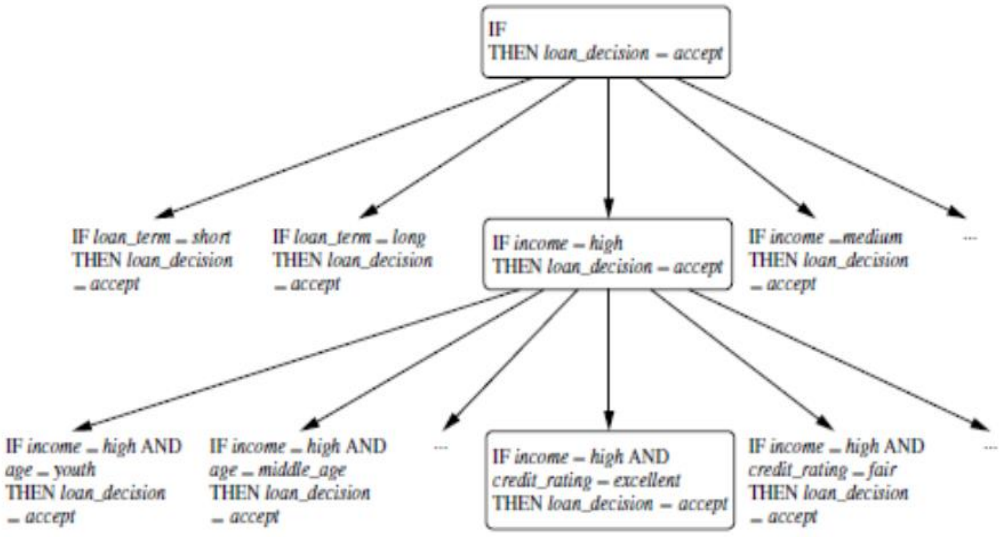


Figure 8.11 A general-to-specific search through rule space.

OLAP VS OLTP:

Feature	OLAP (Online Analytical Processing)	OLTP (Online Transaction Processing)
Purpose	Supports complex analysis and decision-making based on historical data.	Supports day-to-day transactional operations and real-time data processing.
Data Type	Handles large volumes of historical data (typically read-intensive).	Handles smaller, current data sets (both read and write operations).
Database Design	Uses a multidimensional model (star or snowflake schema) for analytical queries.	Uses a relational model (normalized schema) for efficient transaction processing.
Query Complexity	Executes complex, ad-hoc queries involving aggregations, drill-downs, and calculations.	Performs simple, predefined queries (inserts, updates, deletes) efficiently.
Performance	Optimized for query performance and response time.	Optimized for transaction throughput and response time.
Data Latency	Data can have latency, as it involves processing historical information.	Data is up-to-date in real-time or near-real-time.
User Role	Primarily used by analysts, managers, and decision-makers.	Primarily used by operational staff, including employees, customers, and suppliers.

Feature	OLAP (Online Analytical Processing)	OLTP (Online Transaction Processing)
Data Modification	Rarely modifies data, mostly focuses on analysis and reporting.	Frequently modifies data due to frequent transactions and updates.
Data Integrity	Emphasizes data accuracy and consistency, with less focus on strict concurrency controls.	Requires strong concurrency controls and ensures data integrity in a transactional environment.
Data Storage	Optimized for read-intensive operations, may involve denormalization for faster query performance.	Optimized for read and write operations, adhering to normalization principles.
Example Applications	Business intelligence, data mining, forecasting, and trend analysis.	E-commerce, banking transactions, inventory management, and order processing.