GATE in Data Science and AI Study Materials
Machine Learning
By Piyush Wairale

Instructions:

- Kindly go through the lectures/videos on our website www.piyushwairale.com
- Read this study material carefully and make your own handwritten short notes. (Short notes must not be more than 5-6 pages)
- Attempt the question available on portal.
- Revise this material at least 5 times and once you have prepared your short notes, then revise your short notes twice a week
- **If you are not able to understand any topic or required detailed explanation, please mention it in our discussion forum on webiste**
- **Let me know, if there are any typos or mistakes in the study materials. Mail me at piyushwairale100@gmail.com**

1 Intro to UnSupervised Learning

- Unsupervised learning in artificial intelligence is a type of machine learning that learns from data without human supervision.
- Unlike supervised learning, unsupervised machine learning models are given unlabeled data and allowed to discover patterns and insights without any explicit guidance or instruction.
- Unsupervised learning is a machine learning paradigm where training examples lack labels, and clustering prototypes are typically initialized randomly. The primary goal is to optimize these cluster prototypes based on similarities among the training examples.
- Unsupervised learning is a machine learning paradigm that deals with unlabeled data and aims to group similar data items into clusters. It differs from supervised learning, where labeled data is used for classification or regression tasks. Unsupervised learning has applications in text clustering and other domains and can be adapted for supervised learning when necessary.
- Unsupervised learning is the opposite of supervised learning. In supervised learning, training examples are labeled with output values, and the algorithms aim to minimize errors or misclassifications. In unsupervised learning, the focus is on maximizing similarities between cluster prototypes and data items.
- Unsupervised learning doesn't refer to a specific algorithm but rather a general framework. The process involves deciding on the number of clusters, initializing cluster prototypes, and iteratively assigning data items to clusters based on similarity. These clusters are then updated until convergence is achieved.
- Unsupervised learning algorithms have various real-world applications. For instance, they can be used in text clustering, as seen in the application of algorithms like AHC and Kohonen Networks to manage and cluster textual data. They can also be used for tasks like detecting redundancy in national research projects.
- While unsupervised learning focuses on clustering, it is possible to modify unsupervised learning algorithms to function in supervised learning scenarios.

1.1 Working

As the name suggests, unsupervised learning uses self-learning algorithms—they learn without any labels or prior training. Instead, the model is given raw, unlabeled data and has to infer its own rules and structure the information based on similarities, differences, and patterns without explicit instructions on how to work with each piece of data.

Unsupervised learning algorithms are better suited for more complex processing tasks, such as organizing large datasets into clusters. They are useful for identifying previously undetected patterns in data and can help identify features useful for categorizing data.

Imagine that you have a large dataset about weather. An unsupervised learning algorithm will go through the data and identify patterns in the data points. For instance, it might group data by temperature or similar weather patterns.
While the algorithm itself does not understand these patterns based on any previous information you provided, you can then go through the data groupings and attempt to classify them based on your understanding of the dataset. For instance, you might recognize that the different temperature groups represent all four seasons or that the weather patterns are separated into different types of weather, such as rain, sleet, or snow.

Figure 1. An ML model clustering similar data points.

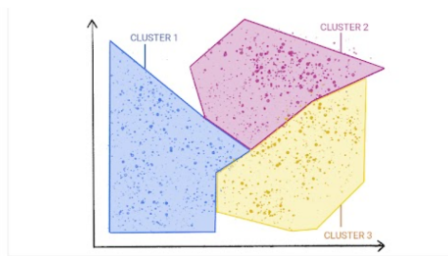


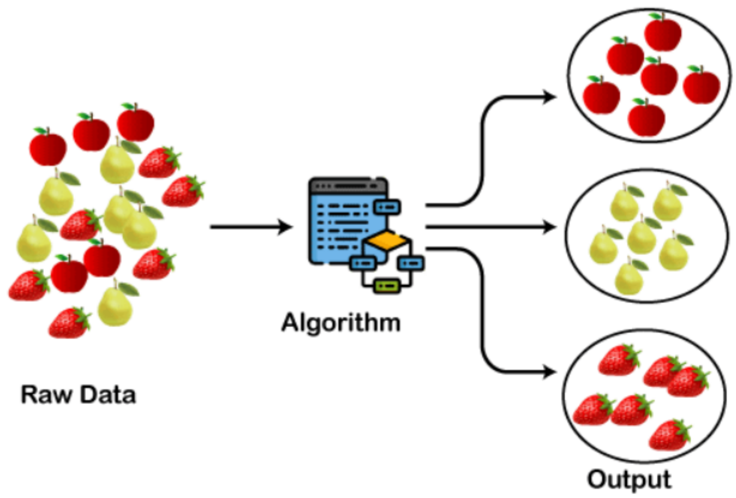
Figure 2. Groups of clusters with natural demarcations.

1.2 Unsupervised machine learning methods

In general, there are three types of unsupervised learning tasks: clustering, association rules, and dimensionality reduction.

1. **Clustering**

   Clustering is a technique for exploring raw, unlabeled data and breaking it down into groups (or clusters) based on similarities or differences. It is used in a variety of applications, including customer segmentation, fraud detection, and image analysis. Clustering algorithms split data into natural groups by finding similar structures or patterns in uncategorized data.



Clustering is one of the most popular unsupervised machine learning approaches. There are several types of unsupervised learning algorithms that are used for clustering, which include exclusive, overlapping, hierarchical, and probabilistic.

- **Exclusive clustering:** Data is grouped in a way where a single data point can only exist in one cluster. This is also referred to as "hard" clustering. A common example of exclusive clustering is the K-means clustering algorithm, which partitions data points into a user-defined number K of clusters.
- **Overlapping clustering:** Data is grouped in a way where a single data point can exist in two or more clusters with different degrees of membership. This is also referred to as "soft" clustering.
- **Hierarchical clustering:** Data is divided into distinct clusters based on similarities, which are then repeatedly merged and organized based on their hierarchical relationships. There are two main types of hierarchical clustering: agglomerative and divisive clustering. This method is also referred to as HAC—hierarchical cluster analysis.
- **Probabilistic clustering:** Data is grouped into clusters based on the probability of each data point belonging to each cluster. This approach differs from the other methods, which group data points based on their similarities to others in a cluster.

2. **Association**

Association rule mining is a rule-based approach to reveal interesting relationships between data points in large datasets. Unsupervised learning algorithms search for frequent if-then associations—also called rules—to discover correlations and co-occurrences within the data and the different connections between data objects.

It is most commonly used to analyze retail baskets or transactional datasets to represent how often certain items are purchased together. These algorithms uncover customer purchasing patterns and previously hidden relationships between products that help inform recommendation engines or other cross-selling opportunities. You might be most familiar with these rules from the "Frequently bought together" and "People who bought this item also bought" sections on your favorite online retail shop.

Association rules are also often used to organize medical datasets for clinical diagnoses. Using unsupervised machine learning and association rules can help doctors identify the probability of a specific diagnosis by comparing relationships between symptoms from past patient cases.

Typically, Apriori algorithms are the most widely used for association rule learning to identify related collections of items or sets of items. However, other types are used, such as Eclat and FP-growth algorithms.

3. **Dimensionality reduction**

   Dimensionality reduction is an unsupervised learning technique that reduces the number of features, or dimensions, in a dataset. More data is generally better for machine learning, but it can also make it more challenging to visualize the data.

   Dimensionality reduction extracts important features from the dataset, reducing the number of irrelevant or random features present. This method uses principle component analysis (PCA) and singular value decomposition (SVD) algorithms to reduce the number of data inputs without compromising the integrity of the properties in the original data.

1.3 Supervised learning vs. unsupervised learning

The main difference between supervised learning and unsupervised learning is the type of input data that you use. Unlike unsupervised machine learning algorithms, supervised learning relies on labeled training data to determine whether pattern recognition within a dataset is accurate.

The goals of supervised learning models are also predetermined, meaning that the type of output of a model is already known before the algorithms are applied. In other words, the input is mapped to the output based on the training data.

Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in Classification and Regression problems.	Unsupervised Learning can be classified in Clustering and Associations problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

1.4 Types of Clustering

Clustering is a type of unsupervised learning wherein data points are grouped into different sets based on their degree of similarity.

The various types of clustering are:

- **Hierarchical clustering**
- **Partitioning clustering**

Hierarchical clustering is further subdivided into:

- Agglomerative clustering
- Divisive clustering

Partitioning clustering is further subdivided into:

- K-Means clustering
- Fuzzy C-Means clustering

There are many different fields where cluster analysis is used effectively, such as

- **Text data mining:** this includes tasks such as text categorization, text clustering, document summarization, concept extraction, sentiment analysis, and entity relation modelling
- **Customer segmentation:** creating clusters of customers on the basis of parameters such as demographics, financial conditions, buying habits, etc., which can be used by retailers and advertisers to promote their products in the correct segment
- **Anomaly checking:** checking of anomalous behaviours such as fraudulent bank transaction, unauthorized computer intrusion, suspicious movements on a radar scanner, etc.
- **Data mining:** simplify the data mining task by grouping a large number of features from an extremely large data set to make the analysis manageable

2 K-Means Clustering

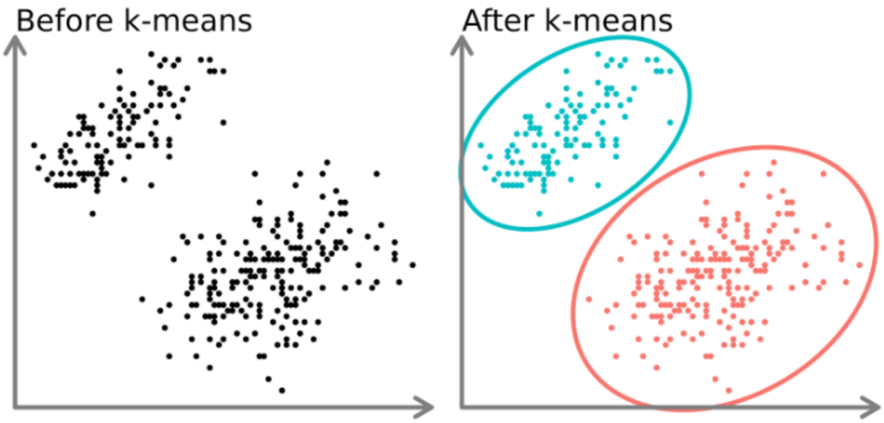
K-Medoids and K-Means are two types of clustering mechanisms in Partition Clustering. First, Clustering is the process of breaking down an abstract group of data points/ objects into classes of similar objects such that all the objects in one cluster have similar traits. , a group of **n** objects is broken down into **k** number of clusters based on their similarities.

- The K-Means algorithm is a popular clustering algorithm used in unsupervised machine learning to partition a dataset into K distinct, non-overlapping clusters.
- It aims to find cluster centers (centroids) and assign data points to the nearest centroid based on their similarity.
- K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.
- Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.
- A cluster refers to a collection of data points aggregated together because of certain similarities.
- K-means is a very simple to implement clustering algorithm that works by selecting k centroids initially, where k serves as the input to the algorithm and can be defined as the number of clusters required. The centroids serve as the center of each new cluster.
- We first assign each data point of the given data to the nearest centroid. Next, we calculate the mean of each cluster and the means then serve as the new centroids. This step is repeated until the positions of the centroids do not change anymore.
- The goal of k-means is to minimize the sum of the squared distances between each data point and its centroid.
- The algorithm aims to minimize the sum of squared distances between data points and their respective cluster centroids.

$$J = \sum_{i=1}^{K} \sum_{j=1}^{n_i} \|x_{ij} - c_i\|^2$$

where:

$$\begin{aligned} K &\text{ is the number of clusters,} \\ n_i &\text{ is the number of data points in cluster } i, \\ x_{ij} &\text{ is the } j\text{-th data point in cluster } i, \\ c_i &\text{ is the centroid of cluster } i. \end{aligned}$$



Here's an overview of the K-Means algorithm:

- **Initialization:** Choose the number of clusters (K) you want to create. Initialize K cluster centroids randomly. These centroids can be selected from the data points or with random values.
- **Assignment Step:** For each data point in the dataset, calculate the distance between the data point and all K centroids. Assign the data point to the cluster associated with the nearest centroid. This step groups data points into clusters based on similarity.
- **Update Step:** Recalculate the centroids for each cluster by taking the mean of all data points assigned to that cluster. The new centroids represent the center of their respective clusters.
- **Convergence Check:** Check if the algorithm has converged. Convergence occurs when the centroids no longer change significantly between iterations. If convergence is not reached, return to the Assignment and Update steps.
- **Termination:** Once convergence is achieved or a predetermined number of iterations are completed, the algorithm terminates. The final cluster assignments and centroids are obtained.
- **Results:** The result of the K-Means algorithm is K clusters, each with its centroid. Data points are assigned to the nearest cluster, and you can use these clusters for various purposes, such as data analysis, segmentation, or pattern recognition.

Key Points and Considerations:

- K-Means is sensitive to the initial placement of centroids. Different initializations can lead to different results. Therefore, it's common to run the algorithm multiple times with different initializations and select the best result.

- The choice of the number of clusters (K) is a critical decision and often requires domain knowledge or experimentation. Various methods, such as the elbow method or silhouette score, can help in determining an optimal K value.
- K-Means is computationally efficient and works well for large datasets, but it may not perform well on data with irregularly shaped or non-convex clusters.
- The algorithm may converge to a local minimum, and it's not guaranteed to find the global optimum.

Strengths	Weaknesses
• The principle used for identifying the clusters is very simple and involves very less complexity of statistical terms • The algorithm is very flexible and thus can be adjusted for most scenarios and complexities • The performance and efficiency are very high and comparable to those of any sophisticated algorithm in term of dividing the data into useful clusters	• The algorithm involves an element of random chance and thus may not find the optimal set of cluster in some cases • The starting point of guessing the number natural clusters within the data requires some experience of the user, so that the final outcome is efficient

Strength and Weakness of K-means

2.1 K-medoids algorithm

- K-medoids is an unsupervised method with unlabelled data to be clustered. It is an improvised version of the K-Means algorithm mainly designed to deal with outlier data sensitivity.
- Compared to other partitioning algorithms, the algorithm is simple, fast, and easy to implement.
- K-medoids clustering method but unlike k-means, rather than minimizing the sum of squared distances, k-medoids works on minimizing the number of paired dissimilarities.
- We find this useful since k-medoids aims to form clusters where the objects within each cluster are more similar to each other and dissimilar to objects in the other clusters. Instead of centroids, this approach makes use of medoids.
- **Medoids** are points in the dataset whose sum of distances to other points in the cluster is minimal.

There are three types of algorithms for K-Medoids Clustering:

1. PAM (Partitioning Around Clustering)
2. CLARA (Clustering Large Applications)
3. CLARANS (Randomized Clustering Large Applications)

PAM is the most powerful algorithm of the three algorithms but has the disadvantage of time complexity. The following K-Medoids are performed using PAM. In the further parts, we'll see what CLARA and CLARANS are.

3 Hierarchical clustering

Till now, we have discussed the various methods for partitioning the data into different clusters. But there are situations when the data needs to be partitioned into groups at different levels such as in a hierarchy. The hierarchical clustering methods are used to group the data into hierarchy or tree-like structure.
For example, in a machine learning problem of organizing employees of a university in different departments, first the employees are grouped under the different departments in the university, and then within each department, the employees can be grouped according to their roles such as professors, assistant professors, supervisors, lab assistants, etc. This creates a hierarchical structure of the employee data and eases visualization and analysis. Similarly, there may be a data set which has an underlying hierarchy structure that we want to discover and we can use the hierarchical clustering methods to achieve that.

- Hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters (or groups) in a given dataset.

- The hierarchical clustering produces clusters in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level.

- At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all of the data.

- The decision regarding whether two clusters are to be merged or not is taken based on the measure of dissimilarity between the clusters. The distance between two clusters is usually taken as the measure of dissimilarity between the clusters.

- Hierarchical clustering is more interpretable than other clustering techniques because it provides a full hierarchy of clusters.

- The choice of linkage method and distance metric can significantly impact the results and the structure of the dendrogram.

- Dendrograms are useful for visualizing the hierarchy and helping to decide how many clusters are appropriate for a particular application.

- Hierarchical clustering can be computationally intensive, especially for large datasets, and may not be suitable for such cases.

3.1 Hierarchical Clustering Methods

There are two main hierarchical clustering methods: **agglomerative clustering** and **divisive clustering.**
Agglomerative clustering is a bottom-up technique which starts with individual objects as clusters and then iteratively merges them to form larger clusters. On the other hand, the **divisive method** starts with one cluster with all given objects 2 and then splits it iteratively to form smaller clusters.
In both these cases, it is important to select the split and merger points carefully, because the subsequent splits or mergers will use the result of the previous ones and there is no option to perform any object swapping between the clusters or rectify the decisions made in previous steps, which may result in poor clustering quality at the end.

A dendrogram is a commonly used tree structure representation of step-by-step creation of hierarchical clustering. It shows how the clusters are merged iteratively (in the case of agglomerative clustering) or split iteratively (in the case of divisive clustering) to arrive at the optimal clustering solution.

3.1.1 Dendrogram

- Hierarchical clustering can be represented by a rooted binary tree. The nodes of the trees represent groups or clusters. The root node represents the entire data set. The terminal nodes each represent one of the individual observations (singleton clusters). Each nonterminal node has two daughter nodes.
- The distance between merged clusters is monotone increasing with the level of the merger. The height of each node above the level of the terminal nodes in the tree is proportional to the value of the distance between its two daughters (see Figure 13.9).
- A dendrogram is a tree diagram used to illustrate the arrangement of the clusters produced by hierarchical clustering.
- The dendrogram may be drawn with the root node at the top and the branches growing vertically downwards (see Figure 13.8(a)).
- It may also be drawn with the root node at the left and the branches growing horizontally rightwards (see Figure 13.8(b)).
- In some contexts, the opposite directions may also be more appropriate.
- Dendrograms are commonly used in computational biology to illustrate the clustering of genes or samples.

Example Figure 13.7 is a dendrogram of the dataset $\{a, b, c, d, e\}$. Note that the root node represents the entire dataset and the terminal nodes represent the individual observations. However, the dendrograms are presented in a simplified format in which only the terminal nodes (that is, the nodes representing the singleton clusters) are explicitly displayed. Figure 13.8 shows the simplified format of the dendrogram in Figure 13.7. Figure 13.9 shows the distances of the clusters at the various levels. Note that the clusters are at 4 levels. The distance between the clusters $\{a\}$ and $\{b\}$ is 15, between $\{c\}$ and $\{d\}$ is 7.5, between $\{c, d\}$ and $\{e\}$ is 15 and between $\{a, b\}$ and $\{c, d, e\}$ is 25.

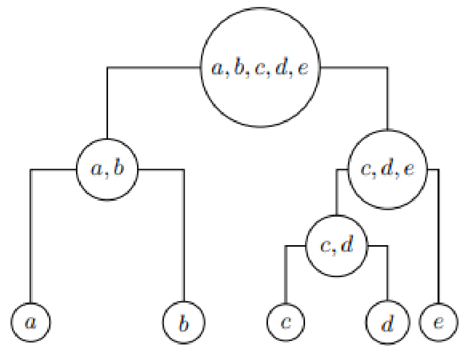


Figure 13.7: A dendrogram of the dataset $\{a, b, c, d, e\}$

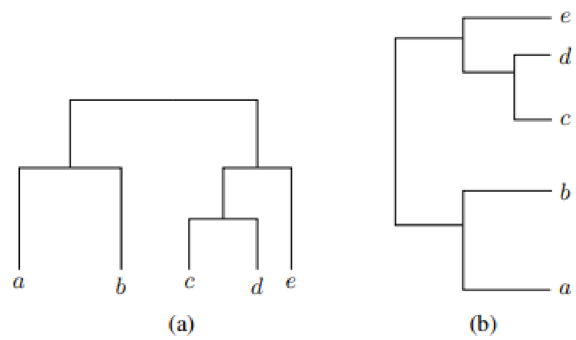


Figure 13.8: Different ways of drawing dendrogram

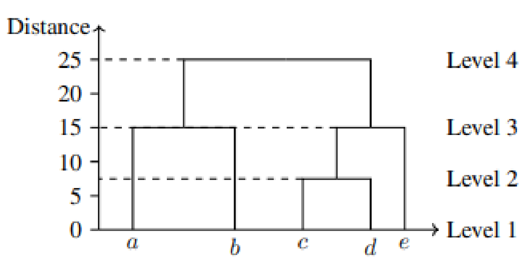
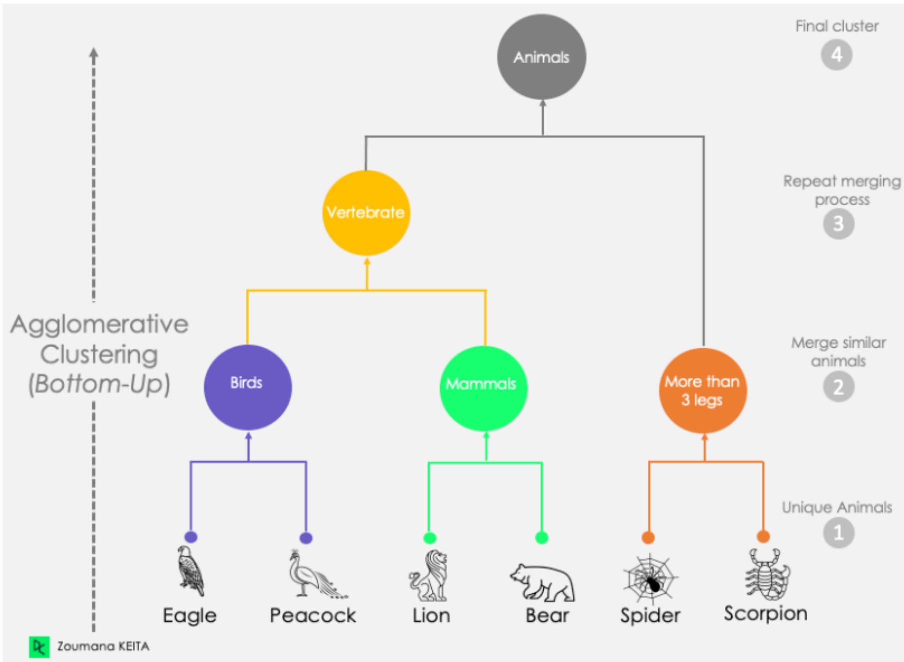


Figure 13.9: A dendrogram of the dataset a, b, c, d, e showing the distances (heights) of the clusters at different levels

3.1.2 Agglomerative Hierarchical Clustering (Bottom-Up)

The agglomerative hierarchical clustering method uses the bottom-up strategy. It starts with each object forming its own cluster and then iteratively merges the clusters according to their similarity to form larger clusters. It terminates either when a certain clustering condition imposed by the user is achieved or all the clusters merge into a single cluster.



If there are N observations in the dataset, there will be $N - 1$ levels in the hierarchy. The pair chosen for merging consists of the two groups with the smallest "intergroup dissimilarity". Each nonterminal node has two daughter nodes. The daughters represent the two groups that were merged to form the parent.

Here's a step-by-step explanation of the agglomerative hierarchical clustering algorithm:

- Step 1: Initialization Start with each data point as a separate cluster. If you have N data points, you initially have N clusters.
- Step 2: Merge Clusters Calculate the pairwise distances (similarity or dissimilarity) between all clusters. Common distance metrics include Euclidean distance, Manhattan distance, or other similarity measures. Merge the two closest clusters based on the distance metric. There are various linkage methods to define cluster distance:
  - **Single Linkage:** Merge clusters based on the minimum distance between any pair of data points from the two clusters.

  - **Complete Linkage:** Merge clusters based on the maximum distance between any pair of data points from the two clusters.
  - **Average Linkage:** Merge clusters based on the average distance between data points in the two clusters.
  - **Ward's Method:** Minimize the increase in variance when merging clusters. Repeat this merging process iteratively until all data points are in a single cluster or until you reach the desired number of clusters.

- Step 3: Dendrogram During the merging process, create a dendrogram, which is a tree-like structure that represents the hierarchy of clusters. The dendrogram provides a visual representation of how clusters merge and shows the relationships between clusters at different levels of granularity.
- Step 4: Cutting the Dendrogram To determine the number of clusters, you can cut the dendrogram at a specific level. The height at which you cut the dendrogram corresponds to the number of clusters you obtain. The cut produces the final clusters at the chosen level of granularity.
- Step 5: Results The resulting clusters are obtained based on the cut level. Each cluster contains a set of data points that are similar to each other according to the chosen linkage method.

For example, the hierarchical clustering shown in Figure 13.7 can be constructed by the agglomerative method as shown in Figure 13.10. Each nonterminal node has two daughter nodes. The daughters represent the two groups that were merged to form the parent.

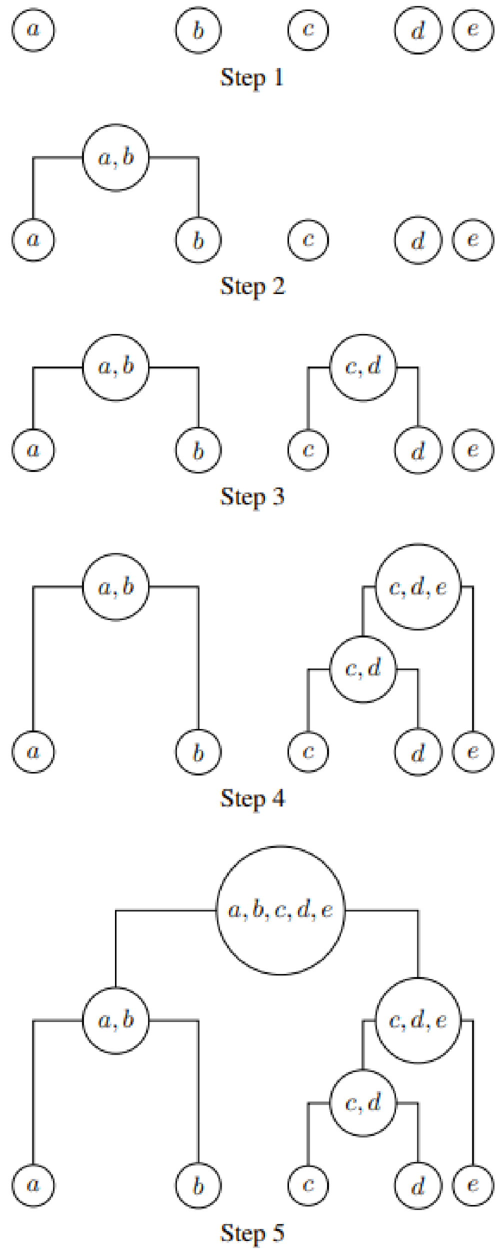


Figure 13.10: Hierarchical clustering using agglomerative method

3.1.3 Divisive Hierarchical Clustering (Top-Down)

- Top-down hierarchical clustering, also known as divisive hierarchical clustering, is a clustering algorithm that starts with all data points in a single cluster and recursively divides clusters into smaller sub-clusters until each data point forms its own individual cluster or a specified number of clusters is reached.
- The divisive hierarchical clustering method uses a top-down strategy. The starting point is the largest cluster with all the objects in it, and then, it is split recursively to form smaller and smaller clusters, thus forming the hierarchy. The end of iterations is achieved when the objects in the final clusters are sufficiently homogeneous to each other or the final clusters contain only one object or the user-defined clustering condition is achieved.
- The divisive method starts at the top and at each level recursively split one of the existing clusters at that level into two new clusters.
- If there are N observations in the dataset, there the divisive method also will produce $N - 1$ levels in the hierarchy. The split is chosen to produce two new groups with the largest "between-group dissimilarity".

  Here's a step-by-step explanation of the divisive hierarchical clustering algorithm:

  - Step 1: Initialization
    Start with all data points as members of a single cluster. If you have N data points, you initially have one cluster with N members.
  - Step 2: Split Clusters
    * Calculate the within-cluster variance or a similar measure for the current cluster. This measure represents the compactness of the cluster.
    * Divide the cluster into two sub-clusters in a way that minimizes the within-cluster variance. There are various methods to achieve this, such as k-means or hierarchical clustering on the sub-cluster.
    * Repeat the splitting process recursively for each sub-cluster until you reach the desired number of clusters.
  - Step 3: Dendrogram (Optional)
    While divisive hierarchical clustering often doesn't produce a dendrogram like agglomerative clustering, you can still record the hierarchy of cluster splits for analysis if needed.
  - Step 4: Results
    The resulting clusters are obtained based on the recursive splits. Each cluster contains a set of data points that are similar to each other according to the splitting criteria.

For example, the hierarchical clustering shown in Figure 13.7 can be constructed by the divisive method as shown in Figure 13.11. Each nonterminal node has two daughter nodes. The two daughters represent the two groups resulting from the split of the parent.

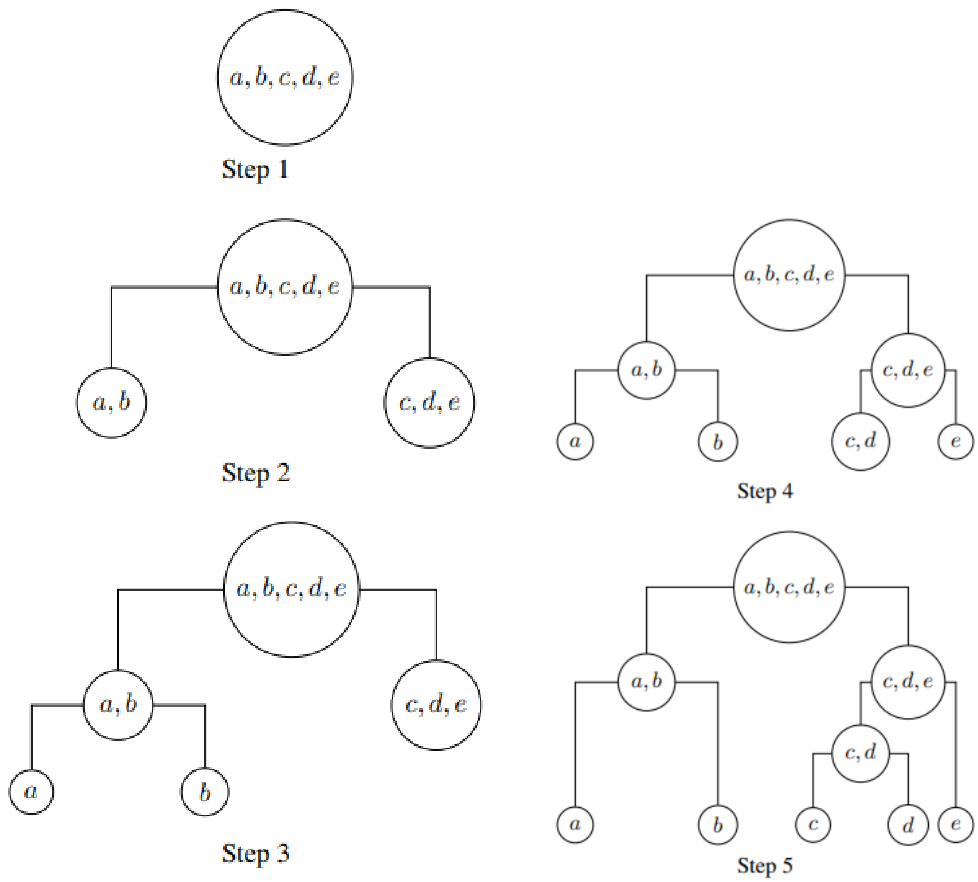


Figure 13.11: Hierarchical clustering using divisive method

In both these cases, it is important to select the split and merger points carefully, because the subsequent splits or mergers will use the result of the previous ones and there is no option to perform any object swapping between the clusters or rectify the decisions made in previous steps, which may result in poor clustering quality at the end.

3.2 Measures of dissimilarity

In order to decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, the dissimilarity between two groups of observations is measured by using an appropriate measure of distance between the groups of observations. The distance between two groups of observations is defined in terms of the distance between two observations. There are several ways in which the distance between two observations can be defined and also there are also several ways in which the distance between two groups of observations can be defined

3.2.1 Measures of distance between data points

One of the core measures of proximities between clusters is the distance between them. There are four standard methods to measure the distance between clusters: Let C_i and C_j be the two clusters with n_i and n_j respectively. p_i and p_j represents the points in clusters C_i and C_j respectively. We will denote the mean of cluster C_i as m_i.

$$\text{Minimum distance } D_{\min}(C_i, C_j) = \min_{p_i \in C_i,\, p_j \in C_j}\{|p_i - p_j|\}$$

$$\text{Maximum distance } D_{\max}(C_i, C_j) = \max_{p_i \in C_i,\, p_j \in C_j}\{|p_i - p_j|\}$$

$$\text{Mean distance } D_{\text{mean}}(C_i, C_j) = \{|m_i - m_j|\}$$

$$\text{Average distance } D_{\text{avg}}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p_i \in C_i,\, p_j \in C_j} |pi - pj|$$

Distance measure in algorithmic methods

Often the distance measure is used to decide when to terminate the clustering algorithm. For example, in an agglomerative clustering, the merging iterations may be stopped once the MIN distance between two neighboring clusters becomes less than the user-defined threshold.

So, when an algorithm uses the minimum distance D_{min} to measure the distance between the clusters, then it is referred to as nearest neighbour clustering algorithm, and if the decision to stop the algorithm is based on a user-defined limit on D_{min} , then it is called a **single linkage algorithm.**

On the other hand, when an algorithm uses the maximum distance D_{max} to measure the distance between the clusters, then it is referred to as furthest neighbour clustering algorithm, and if the decision to stop the algorithm is based on a userdefined limit on D_{max} then it is called **complete linkage algorithm.**

As minimum and maximum measures provide two extreme options to measure distance between the clusters, they are prone to the outliers and noisy data. Instead, the use of mean and average distance helps in avoiding such problem and provides more consistent results.

3.3 Single linkage

Single linkage, also known as single-link clustering, is a hierarchical agglomerative clustering method used in unsupervised machine learning. It is one of the linkage methods used in agglomerative hierarchical clustering. Single linkage defines the distance between two clusters as the minimum distance between any two data points, one from each cluster.

Single linkage works:

1. **Initialization:** Start with each data point as an individual cluster. If you have N data points, you initially have N clusters.
2. **Cluster Distance:** Calculate the pairwise distances between all clusters. The distance between two clusters is defined as the minimum distance between any two data points, one from each cluster.
3. **Merge Clusters:** Merge the two clusters with the shortest distance, as defined by single linkage. This creates a new, larger cluster.
4. **Repeat:** Continue steps 2 and 3 iteratively until all data points are part of a single cluster, or you reach a specified number of clusters.
5. **Dendrogram:** During the process, create a dendrogram, which is a tree-like structure that represents the hierarchy of clusters. It records the sequence of cluster mergers.
6. **Cutting the Dendrogram:** To determine the number of clusters, you can cut the dendrogram at a specific level. The height at which you cut the dendrogram corresponds to the number of clusters you obtain.

Single linkage has some characteristics:

- It is sensitive to outliers and noise because a single close pair of points from different clusters can cause a merger.
- It tends to create elongated clusters, as it connects clusters based on single, nearest neighbors.
- It is fast and can handle large datasets, making it computationally efficient.
- Single linkage is just one of several linkage methods used in hierarchical clustering, each with its own strengths and weaknesses. The choice of linkage method depends on the nature of the data and the desired clustering outcome.

3.4 Multiple Linkage

Multiple linkage, also known as complete linkage or maximum linkage, is a hierarchical clustering method used in agglomerative hierarchical clustering. It defines the distance between two clusters as the maximum distance between any two data points, one from each cluster. In contrast to single linkage, which uses the minimum distance, complete linkage aims to minimize the maximum distance between any two data points within the merged clusters.

Multiple linkage works:

1. **Initialization:** Start with each data point as an individual cluster. If you have N data points, you initially have N clusters.

2. **Cluster Distance:** Calculate the pairwise distances between all clusters. The distance between two clusters is defined as the maximum distance between any two data points, one from each cluster.

3. **Merge Clusters:** Merge the two clusters with the shortest (maximum) distance, as defined by complete linkage. This creates a new, larger cluster.

4. **Repeat:** Continue steps 2 and 3 iteratively until all data points are part of a single cluster or you reach a specified number of clusters.

5. **Dendrogram:** Create a dendrogram to represent the hierarchy of cluster mergers. The dendrogram records the sequence of cluster merges.

6. **Cutting the Dendrogram:** To determine the number of clusters, you can cut the dendrogram at a specific level. The height at which you cut the dendrogram corresponds to the number of clusters you obtain.

Multiple linkage has some characteristics:

- It tends to produce compact, spherical clusters since it minimizes the maximum distance within clusters.

- It is less sensitive to outliers than single linkage because it focuses on the maximum distance.

- It can handle elongated or irregularly shaped clusters effectively.

The choice of linkage method (single, complete, average, etc.) depends on the nature of the data and the desired clustering outcome. Different linkage methods can produce different cluster structures based on how distance is defined between clusters.

3.5 Dimensionality Reduction

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of features (dimensions) in a dataset. It involves transforming a high-dimensional dataset into a lower-dimensional representation while retaining the most relevant information. Dimensionality reduction is useful for several reasons:

- **Curse of Dimensionality:** High-dimensional data can suffer from the curse of dimensionality, where the data becomes sparse and noisy, making it challenging to analyze and model effectively. Reducing dimensionality can help mitigate this issue.

- **Computation and Storage:** High-dimensional data requires more computational resources and storage. Reducing dimensionality can make algorithms faster and more memory-efficient.

- **Visualization:** It is difficult to visualize data in more than three dimensions. Dimensionality reduction techniques can project data into a lower-dimensional space for visualization purposes.

- **Feature Engineering:** Dimensionality reduction can be a form of automated feature selection, helping to identify the most important features and reducing the risk of overfitting.

There are two main approaches to dimensionality reduction:

- **Feature Selection:** In this approach, you select a subset of the original features and discard the rest. This subset contains the most relevant features for the task at hand. Common methods for feature selection include correlation analysis, mutual information, and filter methods.

- **Feature Extraction:** In this approach, you create new features that are combinations or transformations of the original features. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are popular techniques for feature extraction. They find linear combinations of the original features that capture the most significant variation in the data.

Popular algorithms used for dimensionality reduction include principal component analysis (PCA).These algorithms seek to transform data from high-dimensional spaces to low-dimensional spaces without compromising meaningful properties in the original data. These techniques are typically deployed during exploratory data analysis (EDA) or data processing to prepare the data for modeling.

It's helpful to reduce the dimensionality of a dataset during EDA to help visualize data: this is because visualizing data in more than three dimensions is difficult. From a data processing perspective, reducing the dimensionality of the data simplifies the modeling problem.

3.6 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in the fields of statistics, machine learning, and data analysis. It aims to transform high-dimensional data into a lower-dimensional representation while retaining the most important information and patterns.

PCA achieves this by finding a set of orthogonal axes, known as principal components, in the high-dimensional data space. Each principal component is a linear combination of the original features.

Here's a step-by-step explanation of how PCA works:

1. **Data Standardization:** Start with a dataset of high-dimensional data, where each column represents a feature or variable. Standardize the data by subtracting the mean and dividing by the standard deviation for each feature. This step ensures that all features have the same scale.

2. **Covariance Matrix Calculation:** Calculate the covariance matrix of the standardized data. The covariance matrix is a square matrix where each element represents the covariance between pairs of features.

3. **Eigenvalue Decomposition:** Perform eigenvalue decomposition on the covariance matrix to extract eigenvalues and eigenvectors. The eigenvalues represent the amount of variance explained by each principal component. The eigenvectors represent the directions in the high-dimensional space that maximize the variance.

4. **Sort Eigenvalues:** Order the eigenvalues in descending order. The first eigenvalue corresponds to the most significant variance, the second eigenvalue to the second-most significant variance, and so on.

5. **Select Principal Components:** Decide how many principal components you want to retain in the lower-dimensional representation. This choice can be based on the proportion of explained variance or specific requirements for dimensionality reduction.

6. **Projection:** Use the selected principal components (eigenvectors) to transform the data. Each data point is projected onto the subspace defined by these principal components. This transformation results in a lower-dimensional representation of the data.

7. **Variance Explained:** Calculate the proportion of total variance explained by the retained principal components. This information can help you assess the quality of the dimensionality reduction.

8. **Visualization and Analysis:** Visualize and analyze the lower-dimensional data to gain insights, identify patterns, or facilitate further data analysis. Principal components can be interpreted to understand the relationships between features in the original data.

9. **Inverse Transformation (Optional):** If necessary, you can perform an inverse transformation to map the reduced-dimensional data back to the original high-dimensional space. However, some information may be lost in this process.

10. **Application:** Use the lower-dimensional data for various tasks, such as visualization, clustering, classification, or regression, with reduced computational complexity and noise.

PCA provides several benefits:

- **Dimensionality Reduction:** By selecting a subset of principal components, you can reduce the dimensionality of your data while retaining most of the variance. This is especially useful when dealing with high-dimensional data.

- **Noise Reduction:** PCA can help filter out noise in the data, leading to cleaner and more interpretable patterns.

- **Visualization:** PCA facilitates the visualization of data in lower dimensions, making it easier to understand and explore complex datasets.

- **Feature Engineering:** PCA can serve as a form of feature engineering, identifying the most important features for a given task.

References

- Lecture Notes in MACHINE LEARNING, by Dr V N Krishnachandran
- Machine Learning by Amit Kumar Das Saikat Dutt, Subramanian Chandramouli
- https://alexjungaalto.github.io/MLBasicsBook.pdf
- Taeho Jo Machine Learning Foundations Supervised, Unsupervised, and Advanced Learning Springer book
- IIT Madras BS Degree Lectures and Notes
- NPTEL Lectures and Slides
- www.medium.com
- geeksforgeeks.org/
- javatpoint.com/