Unsupervised Learning

Nasrul Huda
Department of Informatics
University of Hamburg, Germany
Email: nasrul.huda@studium.uni-hamburg.de

Laiba Qureshi
Department of Informatics
University of Hamburg, Germany
Email: laiba.qureshi@studium.uni-hamburg.de

***Abstract*—This paper presents an overview of foundational techniques in unsupervised learning, a central machine learning approach focused on uncovering patterns and representations from unlabeled data. It explores major categories—including dimensionality reduction, clustering, and association rule learning—by outlining fundamental algorithms, underlying theory, practical applications, and the advantages and limitations of each method. The goal is to to offer a comprehensive entry point for understanding the landscape of unsupervised learning methods and their applications.**

I. Introduction

For decades, researchers have explored techniques to harness the vast quantities of unlabeled data available in real-world applications. As noted by Xiaojin Zhu, "labels are hard to obtain while unlabeled data are abundant" [**?**]. In an era of rapidly increasing data volumes, manual annotation remains a significant bottleneck. This reinforces the importance of unsupervised learning as a powerful framework for uncovering structure, identifying patterns, and generating insights directly from raw data—without the need for labeled examples.

Unsupervised methods have become essential in domains where labeled data is unavailable or expensive to obtain. They are particularly useful in exploratory analysis, anomaly detection, and scenarios where human labeling is infeasible, such as genomics, recommender systems, and social network analysis [**?**], [**?**].

In recent years, generative models have made significant advances in unsupervised learning. By learning to generate data that resembles the input distribution, models such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) uncover meaningful latent structures, enabling tasks like representation learning, synthetic data generation, clustering, and data compression [**?**].

A. Report Overview

This report provides a comprehensive overview of key methods in unsupervised learning. It begins by contrasting the three main machine learning paradigms—supervised, unsupervised, and reinforcement learning—highlighting their core objectives and differences. The focus then shifts to major categories within unsupervised learning, namely ***Dimensionality Reduction***, ***Clustering***, and ***Association Rule Learning***. For each category, we present widely used algorithms, discuss their underlying principles and mathematical formulations, evaluate their strengths and limitations, and explore real-world applications where these methods are effectively employed.

II. The Three Learning Rules

Machine learning is commonly categorized into three main paradigms based on the nature of the data and the learning objective [**?**]:

A. Reinforcement Learning

Reinforcement learning involves an agent that interacts with an environment and learns to make decisions by receiving rewards or penalties. The objective is to learn a policy that maximizes cumulative reward over time. Key algorithms include Q-learning, Deep Q Networks (DQN), and policy gradient methods. Reinforcement learning has been successfully applied in fields such as robotics, autonomous driving, and game playing (e.g., AlphaGo) [**?**].

B. Supervised Learning

Supervised learning involves learning a function that maps inputs to outputs based on labeled data. The goal is typically predictive—either classification or regression. Popular algorithms include linear regression, logistic regression, support vector machines (SVMs), decision trees, and neural networks [**?**]. Example use cases include spam detection, image classification, and medical diagnosis.

C. Unsupervised Learning

Unsupervised learning refers to a class of techniques aimed at identifying patterns or structures in datasets without predefined labels [**?**]. Unlike supervised learning, which relies on input-output pairs, unsupervised learning algorithms operate on unlabeled data, discovering groupings, correlations, or lower-dimensional representations on their own. These models learn to organize data by detecting similarities and differences, often revealing hidden structures that may not be evident through manual analysis. Common techniques include k-means clustering, DBSCAN, hierarchical clustering, Apriori algorithm, principal component analysis (PCA), and t-SNE, all of which are explained in detail below.

III. Dimensionality Reduction

Dimensionality reduction is a technique that reduces the number of features or dimensions in a dataset by creating a smaller set of new variables that retain the essential information. It is is widely used both for facilitating data visualization and as a pre-processing step to improve the performance of supervised learning algorithms.

A. Principal Component Analysis (PCA)

In many real-world datasets, features may be highly correlated, making analysis redundant and visualization difficult in high-dimensional space. Principal Component Analysis (PCA) mitigates this challenge by identifying a lower-dimensional representation of the the original correlated variables into into a smaller set of uncorrelated variables, called principal components, that captures the maximum possible variance in the data, thereby facilitating more efficient analysis and visualization.

PCA is not only useful in data summarization and compression, but also in handling missing values through imputation and for pre-processing the data for any supervised learning model.

1) Mathematical Formulation: Let $\mathbf{X}$ be a dataset with n observations and p features. We assume that the data is *centered*, i.e., the mean of each feature (column) has been subtracted so that:

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0 \quad \text{for all } j = 1, \ldots, p$$

The first principal component is defined as a normalized linear combination of the original features [**?**]:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

or more compactly:

$$Z_1 = \boldsymbol{\phi}_1^\top \mathbf{X}$$

where $\boldsymbol{\phi}_1 = (\phi_{11}, \phi_{21}, \ldots, \phi_{p1})^\top$ is the *loading vector* for the first principal component. The loadings are subject to the constraint [**?**]:

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

This normalization ensures that the variance of the principal component cannot be made arbitrarily large by scaling the loadings.

Each observation $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ is projected onto the principal component axis as:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$

The resulting values z_{i1} are referred to as the *scores* of the first principal component.

The goal is to find the loading vector that maximizes the sample variance of the scores [**?**]:

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1}x_{ij} \right)^2 \right\} \quad \text{subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

This optimization problem can be solved via *eigenvalue decomposition* of the sample covariance matrix:

$$\mathbf{S} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$$

The solution yields:

- **Eigenvectors:** These correspond to the loading vectors $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots$
- **Eigenvalues:** These represent the amount of variance explained by each corresponding principal component

The first principal component corresponds to the eigenvector associated with the largest eigenvalue, as it captures the greatest variance in the data. Subsequent principal components (e.g., $Z_2, Z_3, \ldots$) are derived similarly, with the additional constraint of being orthogonal (i.e., uncorrelated) to the previously computed components.

2) Choosing Appropriate Components: To determine an appropriate number of principal components k, one typically analyzes the proportion of variance each component explains. Let $\lambda_1, \lambda_2, \ldots, \lambda_p$ be the eigenvalues of the covariance matrix, ordered in decreasing magnitude. The cumulative proportion of variance explained by the first k components is given by:

$$\text{Explained Variance Ratio (EVR)} = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{p}\lambda_i}$$

A common heuristic is to choose the smallest k such that the EVR exceeds a threshold—typically 90% or 95%—ensuring that the reduced dimensionality retains most of the original information. This decision is often aided by examining a *scree plot*, which visualizes the eigenvalues in descending order. The "elbow" point in this plot, where the marginal gain in explained variance drops off significantly, suggests a natural cutoff for selecting k (see Fig. **??**).

B. t-distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE), introduced by van der Maaten and Hinton (2008), is a nonlinear dimensionality reduction technique designed to visualize high-dimensional data in a lower-dimensional space (typically 2D or 3D) [**?**]. t-SNE focuses on maintaining the *local structure* of the data — that is, ensuring that data points that are close together in the high-dimensional space remain close in the low-dimensional embedding.

The core idea of t-SNE is to minimize the mismatch between two probability distributions:

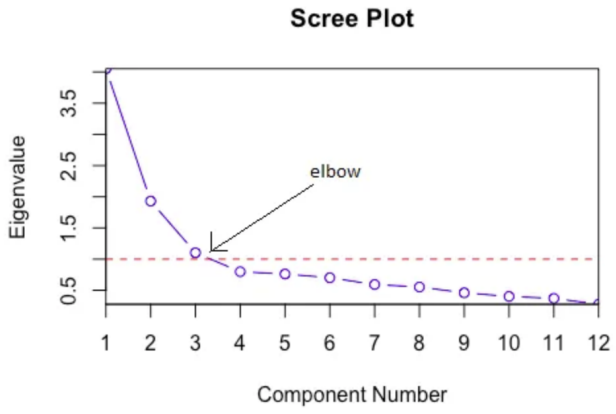

Fig. 1. Example scree plot for selecting the number of principal components [**?**].

- A distribution P that captures pairwise similarities between points in the high-dimensional space.
- A distribution Q that reflects those similarities in the lower-dimensional embedding.

a) High-dimensional similarity: The similarity between two points x_i and x_j in the original space is modeled as a conditional probability $p_{j|i}$ in proportion to the probability density under a Gaussian centered at x_i [**?**]:

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$

b) Low-dimensional similarity: In the embedding space, the similarity q_{ij} between two points y_i and y_j is modeled using a Student's t-distribution with one degree of freedom (equivalent to the Cauchy distribution) [**?**]:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

The use of the t-distribution, with its heavier tails compared to the Gaussian, helps prevent the *crowding problem*, where distant points in the high-dimensional space are erroneously mapped too close together in the low-dimensional space.

Rather than computing separate conditional probabilities $p_{j|i}$ and $q_{j|i}$, t-SNE defines joint probabilities:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}, \quad q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

This leads to a single cost function based on the Kullback-Leibler (KL) divergence which t-SNE uses to minimize the divergence between the high-dimensional and low-dimensional distributions [**?**]:

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

This cost function is minimized using gradient descent. The gradient of the KL divergence with respect to a low-dimensional point y_i guides the optimization: for each pair (i, j), if $p_{ij} > q_{ij}$, then y_i and y_j are pulled closer together; otherwise, they are pushed apart.

C. Additional approaches

In addition to PCA and t-SNE, several other dimensionality reduction techniques are widely used, each with its own strengths. *Uniform Manifold Approximation and Projection (UMAP)* is a recent method that, like t-SNE, preserves local structure but also maintains more of the global relationships [**?**]. It is often faster and more scalable for large datasets. *Isomap* and *Locally Linear Embedding (LLE)* are manifold learning techniques that aim to capture the intrinsic geometry of nonlinear data by preserving geodesic or local linear relationships [**?**], [**?**]. Another powerful family of methods are *Autoencoders*, which are neural network-based architectures that learn compact, meaningful representations by training the network to reconstruct its input [**?**]. These nonlinear techniques are particularly useful when working with complex or high-dimensional datasets where linear methods fall short.

IV. Clustering

Clustering is an unsupervised machine learning technique that partitions a dataset into groups, or clusters, such that observations within the same cluster exhibit high similarity, while those in different clusters are dissimilar. It is widely used to uncover hidden patterns or intrinsic structures in data without relying on labeled outcomes.

A. k-Means Clustering

K-means clustering is a centroid-based method for partitioning a dataset into K distinct, non-overlapping clusters [**?**]. It assumes that the desired number of clusters K is known in advance [**?**]. Each data point is assigned to the cluster with the nearest mean (centroid), and the centroids are iteratively updated to minimize the total within-cluster variation.

Formally, let $C_1, C_2, \ldots, C_K$ be the clusters, where each C_k contains indices of the observations assigned to the k-th cluster. The objective of K-means is to minimize the total within-cluster variation [**?**]:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

where the within-cluster variation $W(C_k)$ is typically defined using the squared Euclidean distance :

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i'\in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

This leads to the overall optimization problem [**?**]:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

A more computationally efficient formulation uses the cluster mean $\bar{x}_{kj}$ for each feature j in cluster C_k:

$$W(C_k) = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

a) Algorithm: The standard K-means algorithm works as follows:

1) Randomly assign each data point to one of the K clusters.
2) Repeat until convergence:
   a) Compute the centroid of each cluster.
   b) Reassign each data point to the cluster with the nearest centroid.

This iterative approach guarantees that the objective function never increases and eventually converges to a local minimum. However, because the solution is sensitive to initialization, it is common practice to run K-means multiple times with different random seeds and select the best outcome based on the lowest objective value.

1) Optimal value of K: Selecting the appropriate value of K in K-means clustering is crucial, as it directly impacts the quality and interpretability of the clustering. Two commonly used methods:

Elbow Method: This method involves plotting the total within-cluster sum of squares (WCSS) against different values of K. Initially, WCSS decreases sharply as K increases, but after a certain point, the rate of decrease slows—forming an "elbow". This elbow indicates the optimal number of clusters, as shown in Fig. **??** below.

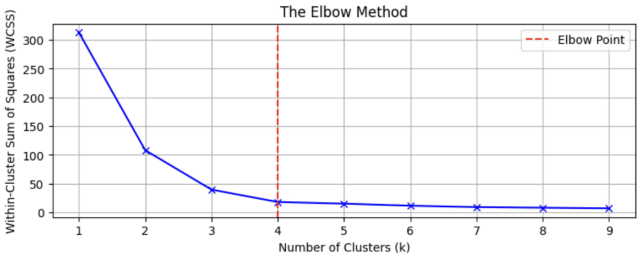

Fig. 2. Elbow method showing distortion score vs. number of clusters [**?**].

Silhouette Score: This metric evaluates the quality of clustering by measuring how similar a point is to its own cluster compared to other clusters. It ranges from -1 to 1, where a higher value indicates better-defined and well-separated clusters. The optimal K is the one with the highest average silhouette score. (Fig. **??**)

Both methods provide heuristic guidelines, and often they are used together or supported with domain knowledge to make a final decision about the number of clusters.

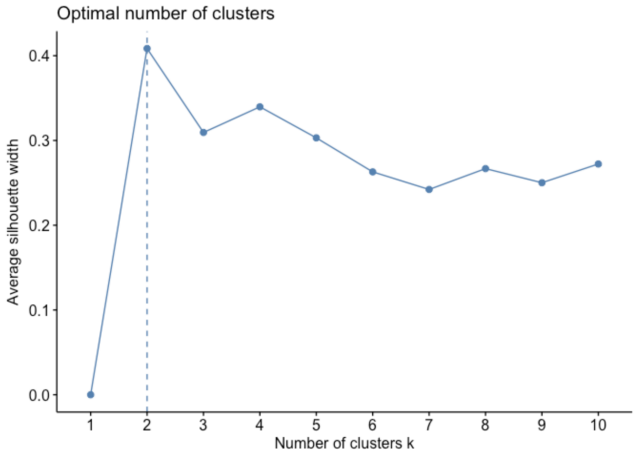

Fig. 3. Avg silhouette scores for different values of K. Highest score indicates optimal cluster count [**?**].

B. Hierarchical Clustering

Hierarchical Clustering is a tree representation of clusters of a given dataset. Leaves of the tree represent individual data points and each internal node defines a cluster on leaf nodes of its descendants. It is used to group similar data points together based on their similarity creating hierarchy or tree-like structure. Hierarchical clustering does not require a prespecified number of clusters. Hierarchical Clustering algorithms fall into two classes - agglomerative hierarchical clustering and divisive clustering [**?**], [**?**].

Agglomerative Clustering: Also known as the bottom-up approach. In Hierarchical Agglomerative Clustering(HAC), the key idea is to begin with each data point and treat them as their own individual clusters and then start merging the nodes or clusters recursively based on their similarity measure.

An HAC clustering is typically visualised as a dendrogram as shown in Fig. **??**. Each merge is represented by a horizontal line. The y-coordinate of the horizontal line is the similarity of the two clusters that were merged, where documents are viewed as singleton clusters. We call this similarity the *combination similarity* of the merged cluster. For example, the combination similarity of the cluster consisting of *Lloyd's CEO questioned* and *Lloyd's chief / U.S. grilling* in the figure **??** ≈ 0.56. We define the combination similarity of a singleton cluster as its document's self-similarity (which is 1.0 for cosine similarity).

A fundamental assumption in HAC is that the merge operation is monotonic, i.e., if $s_1, s_2, ..., s_{K-1}$ are the combination similarities of the successive merges of an HAC, then $s_1 \geq s_2 \geq ... \geq s_{K-1}$ holds. A non-monotonic hierarchical clustering contains at least one inversion $s_i < s_{i+1}$ and contradicts the fundamental assumption that we chose the best merge available at each step.

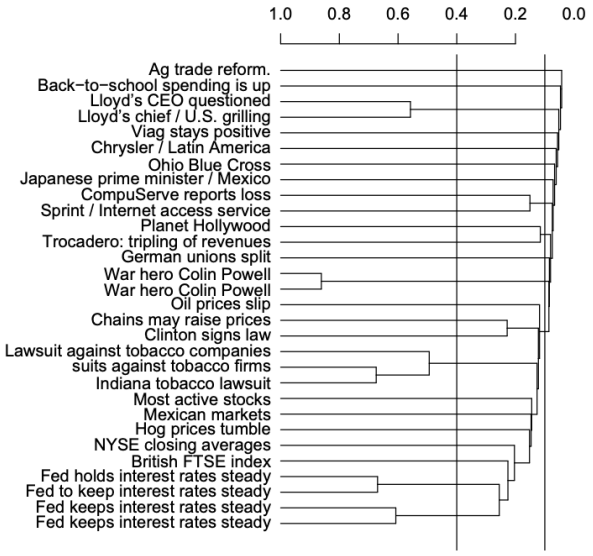

Fig. 4. Dendrogram of a single-linkage clustering of 30 documents from Reuters-RCV1. Two possible cuts of a dendrogram are shown: at 0.4 into 24 clusters and at 0.1 into 12 clusters.

1) Linkage Criteria: The similarity between two clusters is defined as the linkage criteria. There are four common linkage criteria:

- **Single Linkage:** The similarity between two clusters is defined as the similarity between the two most similar points in the two clusters.
- **Complete Linkage:** The similarity between two clusters is defined as the similarity between the two least similar points in the two clusters.
- **Average Linkage:** The similarity between two clusters is defined as the average similarity between all pairs of points in the two clusters.
- **Centroid Linkage:** The similarity between two clusters is defined as the similarity between the centroids of the two clusters.

As earlier mentioned, hierarchical clustering does not require a prespecified number of clusters, however, in some applications we want a partition of disjoint clusters just as in the case of flat clustering methods. How do we determine the number of clusters in a hierarchical clustering? A number of criteria can be used to determine the cutting point:

- Cut at a prespecified level of similarity. For example, we cut the dendrogram at 0.4 if we want clusters with a minimum combination similarity of 0.4. In Figure **??**, we can cut the dendrogram at 0.4 into 24 clusters or at 0.1 into 12 clusters.
- Cut the dendrogram where the gap between two successive combination similarities is largest. Such large gaps arguably indicate "natural" clusterings. Adding one more cluster decreases the quality of the clustering significantly, so cutting before this steep decrease occurs is desirable.
- Apply equation

$$K = \underset{K'}{\operatorname{argmin}} \left(RSS(K') + \lambda K' \right)$$

where K' refers to the cut of the hierarchy that results in K' clusters, RSS is the residual sum of squares and λ is a penalty for each additional cluster. Instead of RSS, another measure of distortion can be used.

- As in flat clustering, we can also prespecify the number of clusters K and select the cutting point that produces K clusters.

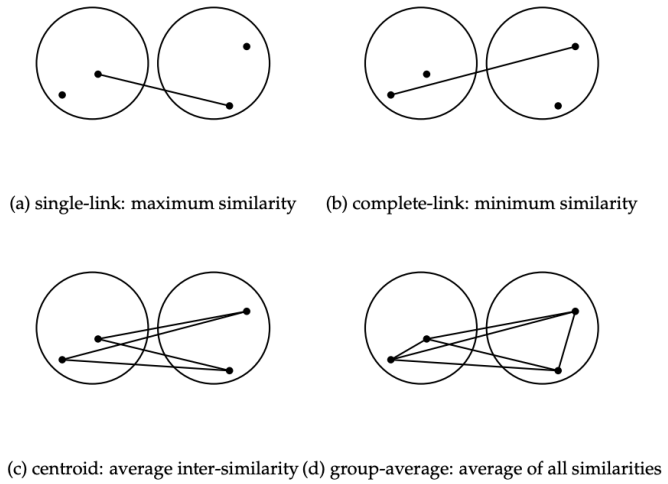

Fig. 5. Linkage criteria for hierarchical clustering.

Divisive Clustering: Also known as the top-down approach. In Divisive Clustering, all of the data points are treated as one single cluster at the beginning, and a cluster hierarchy will be generated top-down. The cluster is split using a flat clustering algorithm. This procedure is applied recursively until each document is in its own singleton cluster.

Conceptually, divisive clustering is the reverse of agglomerative clustering and it is more complex because we need a second, flat clustering algorithm as a "subroutine". It has the advantage of being more efficient if we do not generate a complete hierarchy all the way down to individual document leaves. For a fixed number of top levels, using an efficient flat algorithm like K-means, top-down algorithms are linear in the number of documents and clusters. So they run much faster than HAC algorithms, which are at least quadratic.

There is evidence that divisive clustering produces better and more accurate results than bottom-up algorithms in some circumstances. Bottom-up methods make clustering

decisions based on local patterns without initially taking into account the global distribution. These early decisions cannot be undone. Top-down clustering benefits from the complete information about the global distribution when making top-level partitioning decisions.

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

DBSCAN is a density-based clustering algorithm that groups together points that are closely packed together, based on a distance measure [**?**]. It is a popular algorithm for clustering because it does not require the number of clusters to be specified in advance, and it can find arbitrarily shaped clusters.

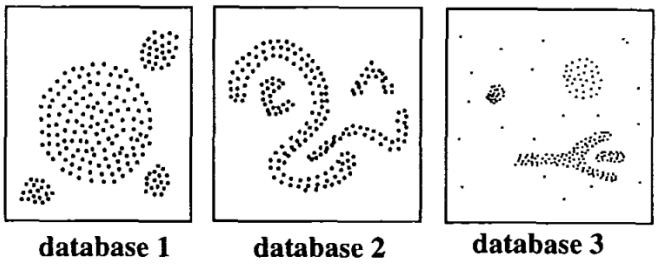

Fig. 6. Sample datasets with arbitrary shapes.

The main reason why the algorithm recognises the clusters is that within each cluster we have a typical density of points which considerably higher than outside of the cluster. Furthermore, the density within the areas of noise is lower than the density within the areas of noise is lower than the density in any of the clusters. Note that the algorithm apply as well to 2D or 3D Euclidean space as to some high dimensional feature space.

The key idea is that for each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points, i.e., the density in the neighborhood has to exceed some threshold. The shape of a neighborhood is determined by the choice of a distance function for two points p and q, denoted by $d(p, q)$. For instance, when using Manhatten distance in 2D space, the shape of the neighborhood is rectangular. Note, that our approach works with any distance function so that an appropriate function can be chosen for some given application. For the purpose of proper visualization, all examples will be in 2D space using the Euclidean distance.

The hyperparamters of the algorithm are:

- ϵ: Eps-neighborhood of a point p is defined as the set of all points within a distance of ϵ from p.
- $MinPts$: Minimum number of points in an eps-neighborhood.

1) Algorithm: To find a cluster, DBSCAN starts with an artbitrary point p and retrieves all points density-reachable from p wrt. ϵ and $MinPts$. If p is a core point, this procedure yields a cluster wrt. ϵ and $MinPts$. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the dataset.

Since the algorithm uses global values for ϵ and $MinPts$, DBSCAN may merge two clusters into one cluster, if two clusters of different density are "close" to each other. Let the *distance between two sets of points* S_1 *and* S_2 be defined as $dist(S_1, S_2) = \min_{p \in S_1, q \in S_2} d(p, q)$. Then, two sets of points having at least the density of the thinnest cluster will be separated from each other only if the distance between the two sets is larger than ϵ. Consequently, a recursive call of DBSCAN may be necessary for the detected clusters with a higher value for $MinPts$. This is however, no disadvantage because the recursive application of DBSCAN yields an elegant and very efficient basic algorithm. Furthermore, the recursive clustering of the points of a cluster is only necessary under conditions that can be easily detected.

If two clusters C_1 and C_2 are very close to each other, it might happen that some point p belongs to both C_1 and C_2. Then p must be border point in both clusters because otherwise C_1 would be equal to C_2 since we use global parameters. In this case, point p will be assigned to the clusters discovered first. Except from these rare situations, the result of DBSCAN is independent of the order in which the points of the dataset are visited.

2) Determining the Parameters (ϵ and $MinPts$): In this section, we will discuss a simple but effective heuristic to determine the parameters ϵ and $MinPts$ of the "thinnest" cluster in the database. This heuristic is based on the following observation. Let d be the distance of a point p to its k-th nearest neighbor, then the d-neighborhood of p contains exactly $k + 1$ points for almost all points p. The d-neighborhood of p contains more than $k + 1$ points only if several points have exactly the same distance d from p which is quite unlikely. Furthermore, changing l for a point in a cluster does not result in large changes of d. This only happends if the k-th nearest neighbors of p for $k = 1, 2, 3, ...$ are located approximately on a straight line which is in general not true for a point in a cluster.

For a given k, a function $k - dist$ from the database D to the real numbers, mapping each point to the distance from its k-th nearest neighbor. When sorting the points of the dataset in descending order of their $k - dist$ values, the graph of this function gives some hints concerning the density distribution in the dataset. The graph is called $sortedk - dist$ graph. If we choose an arbitrary point p, set the parameter ϵ to $k - dist(p)$ and set the parameter $Minpts$ to k, all points with an equal or smaller $k - dist$ value wil be core points. If we could find a *threshold point* with the maximal $k - dist$

value in the thinnest cluster of D we would have the desired parameter values. The threshold point is the first point in the first "valley" of the sorted k-dist graph. All points with a higher k-dist value (left of the threshold) are considered to be noise, all other points (right of the threshold) are assigned to some cluster.

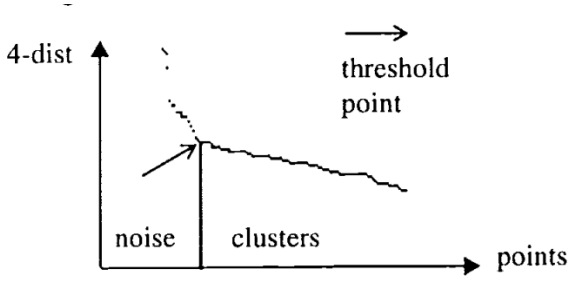

Fig. 7. Sorted 4-dist graph for sample dataset 3.

In general, it is very difficult to detect the first "valley" automatically, but it is relatively simple for a user to see this valley in a graphical representation. Therefore, we propose to follow an interative approach for determining the threshold point.

DBSCAN needs two parameter, ϵ and $MinPts$. However, our experiments indicate that the k-dist graphs for k ¿ 4 do not significantly differ from the 4-dist graph and further, they need considerably more computation. Threrefore, we eliminiate the parameter $MinPts$ by setting it to 4 for all datasets (for 2-dimensional data). We propose the following interactive approach for determining the parameter ϵ of DBSCAN:

1) The system computes and displays the 4-dist graph for the dataset.
2) If the user can estimate the percentage of noise, this percentage is entered and the system derives a proposal for the threshold point from it.
3) The user either accepts the proposed threshold or selects another point as the threshold point. The 4-dist value of the threshold point is used as the ϵ value for DBSCAN.

D. Evaluation Metrics

To assess the quality of clustering results, several evaluation metrics are used. These metrics help determine how well the algorithm has grouped similar data points and separated dissimilar ones. Evaluation can be divided into internal and external metrics.

Internal Metrics: These metrics are used to evaluate the quality of the clustering without reference to external information. They are based on the properties of the clusters themselves, such as the within-cluster similarity and the between-cluster separation.

External Metrics: These metrics are used to evaluate the quality of the clustering with reference to external information. They are based on the properties of the clusters themselves, such as the within-cluster similarity and the between-cluster separation.

Name	Formula or measure method	Explanation
Davies–Bouldin indicator	$DB = \frac{1}{k}\sum_{i=1}^{k}\max_{i\neq j}\left(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\right)$	K stands for the number of clusters, C_x is the center of cluster x, σ_x is the average distance between any data in cluster x and C_x, $d(c_i, c_j)$is the distance between c_i and c_j
Dunn indicator	$D = \min_{1\leq i\leq n}\left\{\min_{1\leq j\leq n, i\neq j}\left\{\frac{d(i,j)}{\max_{1\leq k\leq n} d'(k)}\right\}\right\}$	1. Mainly for the data that has even density and distribution 2. $d(c_i, c_j)$is the distance between c_i and c_j, $d'(k)$ stands for the distance in cluster k
Silhouette coefficient	Evaluate the clustering result based on the average distance between a data point and other data points in the same cluster and average distance among different clusters	

Fig. 8. Internal evaluation metrics [**?**].

Name	Formula or measure method	Explanation
Rand indicator	$RI = \frac{TP+TN}{TP+FP+FN+TN}$	1. TP is the number of true positives 2. TN is the number of true negatives 3. FP is the number of false positives 4. FN is the number of false negatives
F indicator	$F_\beta = \frac{\left(\beta^2+1\right)\cdot P\cdot R}{\beta^2\cdot P+R}$	1. $P = \frac{TP}{TP+FP}$ stands for the accuracy, $R = \frac{TP}{TP+FN}$ stands for the recall rate 2. TP, TN, FP, and FN are defined as before
Jaccard indicator	$J(A, B) = \frac{\lvert A\cap B\rvert}{\lvert A\cup B\rvert} = \frac{TP}{TP+FP+FN}$	1. Measure the similarity of two sets 2. |X| Stands for the number of elements of set X 3. TP, TN, FP, and FN are defined as before
Fowlkes–Mallows indicator	$FM = \sqrt{\frac{TP}{TP+FP}\cdot\frac{TP}{TP+FN}}$	TP, TN, FP, and FN are defined as before
Mutual information	To measure, based on information theory, how much information is shared by two clusters, between which the nonlinear correlation can be detected	
Confusion matrix	To figure out the difference between a cluster and a gold-standard cluster	

Fig. 9. External evaluation metrics [**?**].

V. Association Rule Learning

Association rule learning is a rule-based learning method for discovering interesting relations between variables in large datasets [**?**]. It aims to identify strong rules discovered in databases using measures of interestingness such as support, confidence, and lift.

The main key concepts of association rule learning are:

- **Itemset:** A collection of one or more items.
- **Support:** The support of a rule is the proportion of transactions that contain all the items in the rule.
- **Confidence:** The confidence of a rule is the proportion of transactions that contain all the items in the rule, divided by the proportion of transactions that contain the antecedent of the rule.
- **Lift:** The lift of a rule is the ratio of the confidence of the rule to the expected confidence of the rule if the antecedent and consequent were independent.

A. Apriori Algorithm

The Apriori algorithm is a popular algorithm for mining frequent itemsets and generating association rules [**?**]. It is based on the Apriori principle, which states that if an itemset is frequent, then all of its subsets must also be frequent.

Apriori is the first association rule mining algorithm that prioneered the use of support-based pruning to systematically control the exponential growth of candidate itemsets.

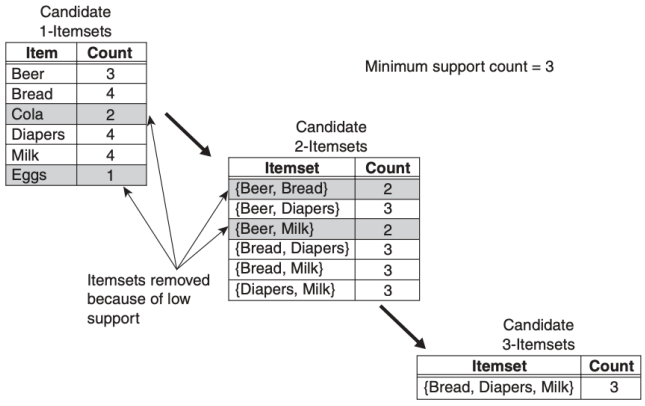

Fig. 10. High-level illustration of the frequent itemset generation part of the Apriori algorithm.

Initially, eveery item is considered as a candidate 1-itemset. After counting their supports, the candidate itemsets $Cola$ and $Eggs$ are discarded because they appear in fewer than three transactions. In the next iteration, candidate 2-itemsets are generated using only the frequent 1itemsets because the Apriori principle ensures that all the supersets of the infrequent 1-itemsets must be infrequent. because there are only four frequent 1-itemsets, the number of candidate 2-itemsets generated by the algorithm is 6. Two of these six candidates, $Beer, Bread$ and $Beer, Milk$ are subsequently found to be infrequent after computing their support values. The remaining four candidates are frequent, and thus will be used to generate candidate 3-itemsets. Without support-based pruning, there are 20 candidate 3-itemsets that can be formed using the six items given in this example. With the Apriori principle, we only need to keep candidate 3-itemsets whose subsets are frequent. The only candidate that has this property is $Bread, Diapers, Milk$.

The effectiveness of the Apriori pruning strategy can be shown by counting the number of candidate itemsets generated. A brute-force strategy of enumerating all itemsets (up to size 3) as candidates will produce

$$\binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 6 + 15 + 20 = 41$$

candidates. With the Apriori principle, this number decreases to

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13$$

candidates, which represents a 68% reduction in the number of candidate itemsets even in this simple example.

1) Algorithm: Let C_k denote the set of candidate k-itemsets and F_k denote the set of frequent k-itemsets. The algorithm works as follows:

- The algorithm initially makes a single pass over the data set to determine the support of each item. Upon completion of this step, the set of all frequent 1-itemsets, F_1, will be known (steps 1 and 2).
- Next, the algorithm will iteratively generate new candidate k-itemsets using the frequent $(k-1)$-itemsets found in the previous iteration (step 5). Candidate generation is implemented using a function called apriori-gen, which is described in Section 6.2.3.
- To count the support of the candidates, the algorithm needs to make an additional pass over the data set (steps 6–10). The subset function is used to determine all the candidate itemsets in C_k that are contained in each transaction t. The implementation of this function is described in Section 6.2.4.
- After counting their supports, the algorithm eliminates all candidate itemsets whose support counts are less than $minsup$ (step 12).
- The algorithm terminates when there are no new frequent itemsets generated, i.e., $F_k = \emptyset$.

The frequent itemset generation part of the Apriori algorithm has two important characteristics. First, it is a **level-wise** algorithm; i.e., it traverses the itemset lattice one level at a time, from frequent 1-itemsets to the maximum size of frequent itemsets. Second, it employs a **generate-and-test** strategy for finding frequent itemsets. At each iteration, new candidate itemsets are generated from the frequent itemsets found in the previous iteration. The support for each candidate is then counted and tested against the $minsup$ threshold. The total number of iterations needed by the algorithm is $k_{max}+1$, where k_{max} is the maximum size of the frequent itemsets.

B. Additional approaches

There are several other approaches to association rule learning. They are as follows:

1) FP-Growth Algorithm: FP-Growth avoids Apriori's expensive candidate generation by building a compact data structure called an FP-tree. It compresses the dataset by storing frequent patterns in a tree format and recursively mines the tree to find frequent itemsets, making it much faster and more scalable for large datasets.

2) Eclat Algorithm: Eclat uses a vertical data layout, representing items with lists of transaction IDs. It finds frequent itemsets by intersecting these ID lists, which is very efficient, especially on dense data, because it avoids generating large numbers of candidates explicitly.

3) FP Max Algorithm: FPMax is designed to find maximal frequent itemsets, which are the largest itemsets meeting the minimum support threshold without any frequent supersets. This approach reduces the number of patterns returned, simplifying the output without losing important information about the data's structure.

VI. Real World Applications

Some of the real world applications for unsupervised learning methods are:

A. Dimensionality Reduction

- Data visualization: Reducing high-dimensional data to 2D or 3D (e.g. t-SNE, PCA) for exploratory analysis.
- Noise reduction: Removing irrelevant features in signal processing or images.
- Feature extraction for machine learning: Improving model training by reducing overfitting and computation.
- Genomics: Simplifying gene expression data for biomarker discovery.

B. Clustering

- Customer segmentation: Marketers group customers based on purchasing behavior to tailor promotions.
- Image segmentation: Grouping pixels into regions for object recognition.
- Document clustering: Grouping news articles or search results by topic for easier navigation.
- Anomaly detection: Identifying fraud or network intrusions by spotting outlier clusters.

C. Association Rule Learning

- Market basket analysis: Retailers find products frequently bought together (e.g. diapers and beer).
- Cross-selling strategies: Banks identify which financial products are bought together to target offers.
- Web usage mining: Understanding navigation patterns to improve site design and recommendations.
- Medical diagnosis: Discovering symptom–disease relationships in patient records.

VII. Conclusion

LQ -

References

[1] X. Zhu, "Semi-supervised learning literature survey," *Comput Sci, University of Wisconsin-Madison*, vol. 2, 07 2008.

[2] R. Yu, H. Qiu, Z. Wen, C. Lin, and Y. Liu, "A survey on social media anomaly detection," *SIGKDD Explor. Newsl.*, vol. 18, no. 1, p. 1–14, Aug. 2016. [Online]. Available: https://doi.org/10.1145/2980765.2980767

[3] N. Verbeeck, R. Caprioli, and R. Van de Plas, "Unsupervised machine learning for exploratory data analysis in imaging mass spectrometry," *Mass Spectrometry Reviews*, vol. 39, no. 3, pp. 245–291, 2019. [Online]. Available: https://doi.org/10.1002/mas.21602

[4] J. C. Ye, "Generative models and unsupervised learning," in *Geometry of Deep Learning: A Signal Processing Perspective*. Singapore: Springer Nature Singapore, 2022, pp. 267–313.

[5] S. Dridi, "Unsupervised learning - a systematic literature review," 04 2022.

[6] J. Hui, "Alphago: How it works technically? - jonathan hui," *Medium*, May 2018. [Online]. Available: https://jonathan-hui.medium.com/alphago-how-it-works-technically-26ddcc085319

[7] M. Iqbal and Z. Yan, "Supervised machine learning approaches: A survey," *International Journal of Soft Computing*, vol. 5, pp. 946–952, 04 2015.

[8] G. James, D. Witten, T. Hastie, and R. Tibshirani, *Unsupervised Learning*. New York, NY: Springer US, 2021, pp. 497–552.

[9] S. Mangale, "Scree plot," https://sanchitamangale12.medium.com/scree-plot-733ed72c8608, 2021, accessed: 2025-07-03.

[10] L. van der Maaten and G. Hinton, "Viualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 11 2008.

[11] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018. [Online]. Available: https://arxiv.org/abs/1802.03426

[12] B. Tripathy, S. Anveshrithaa, and S. Ghela, *Isomap*, 07 2021, pp. 53–65.

[13] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[14] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science (New York, N.Y.)*, vol. 313, pp. 504–7, 08 2006.

[15] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, vol. 622, pp. 178–210, 2023.

[16] L. Rokach and O. Maimon, *Clustering Methods*. Boston, MA: Springer US, 2005, pp. 321–352.

[17] GeeksforGeeks, "Elbow method for optimal value of k in kmeans," 2021, accessed: 2025-07-03.

[18] University of Cincinnati Business Analytics R Programming Guide, "K-means clustering," 2020, accessed: 2025-07-03.

[19] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/pdf/17hier.pdf

[20] P.-N. Tan, M. Steinbach, and V. Kumar, *Cluster Analysis: Basic Concepts and Algorithms*. Boston, MA: Pearson, 2005, ch. 8. [Online]. Available: https://www-users.cse.umn.edu/ kumar/dmbook/ch8.pdf

[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD '96. AAAI Press, 1996, pp. 226–231. [Online]. Available: https://cdn.aaai.org/KDD/1996/KDD96-037.pdf

[22] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, 2017. [Online]. Available: https://link.springer.com/article/10.1007/s40745-015-0040-1

[23] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Record*, vol. 22, no. 2, pp. 207–216, 1993. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/170035.170072

[24] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. Morgan

Kaufmann Publishers Inc., 1994, pp. 487–499. [Online]. Available: https://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/vldb94$_rj.pdf$