République algérienne démocratique et populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Boudiaf de M'sila
Faculté des Mathématiques et de l'Informatique
Département de l'Informatique

THÈSE

Présentée par :
Nasereddine AMROUNE

En vue de l'obtention du diplôme de

Doctorat en Sciences

Filière : Informatique
Spécialité : Informatique

SUJET

Développement d'une nouvelle approche pour la classification non supervisée de données

Soutenue publiquement le : 09/10/2025

Jury :

Président :	Pr.	Debbi Hichem	Univ. M'sila
Promoteur :	Pr.	SAYAD Lamri	Univ. M'sila
Examinateur :	Dr.	Semcheddine Moussa	Univ. Setif 1
Examinateur :	Dr.	Slimani Yacin	Univ. Setif 1

2024/2025

To My Parents

In Memory of My Sister

In Memory of Dr. Lamiche Chaabane

Acknowledgments

First and foremost, I thank Almighty Allah for giving me the strength and courage to complete this thesis to the end. I would like to thank my thesis supervisor, Dr. Lamiche Chaabane, for his supervision and support. May God bless him with His holy mercy.

I express my gratitude to Prof. Lamri SAYAD for agreeing to be my new thesis supervisor after the passing of Mr. Lamiche Chaabane.

I would like also to thank my colleague Maklouf BENAZI for his guidance and help with many issues.

I also want to express my appreciation to the examiners who accepted to sit on the jury for this thesis and for their valuable comments.

Lastly, I thank everyone who contributed, directly or indirectly, to the completion of this work.

ملخص

تطرقنا في هذه الرسالة إلى مشكلة تصنيف البيانات غير الخاضعة للإشراف ، مع التركيز بشكل خاص على خوارزميات التجميع القائمة على الكثافة. بدأنا بمراجعة المفاهيم الأساسية للتجميع وتطبيقاته المتنوعة عبر مختلف المجالات. ثم تعمقنا في تعقيدات خوارزمية DBSCAN ، وهي تقنية قوية للتجميع على أساس الكثافة معروفة بقدرتها على تحديد مجموعات من الأشكال التعسفية والتعامل مع البيانات الصاخبة بشكل فعال.

إدراكًا لقيود اعتماد DBSCAN على قيمة Eps، شرعنا في تطوير نسخة معدلة من الخوارزمية التي تلغي الحاجة إلى تحديد Eps يدويًا. يستخدم نهجنا خوارزمية k-nearest neighbors (kNN) لحساب قيمة Eps لكل نقطة بيانات بشكل تكيفي بناءً على كثافة الجوار المحلي. تمكن هذه الآلية التكيفية خوارزميتنا من تحديد المجموعات ذات الكثافات غير المتجانسة بشكل فعال ، وهي مهمة غالبًا ما تكافح الأساليب التقليدية معها.

من خلال التقييمات التجريبية المكثفة على كل من مجموعات البيانات الاصطناعية والعالمية الحقيقية ، أظهرنا الأداء المتفوق لخوارزمية DBSCAN المعدلة مقارنة بنظيرتها التقليدية. لا يؤدي حساب Eps التلقائي والتكيفي إلى تبسيط عملية التجميع فحسب ، بل يؤدي أيضًا إلى تحديد الكتلة بشكل أكثر دقة وذات مغزى.

بينما حقق عملنا خطوات كبيرة في تطوير التجميع القائم على الكثافة ، لا تزال هناك طرق مثيرة للبحث في المستقبل. أحد الاتجاهات الواعدة هو استكشاف التحديد التلقائي لقيمة MinPts ، مما قد يجعل الخوارزمية خالية تمامًا من المعلمات. بالإضافة إلى ذلك ، يمكن أن يؤدي التحقيق في قابلية تطبيق نهج Eps التكيفي الخاص بنا على خوارزميات التجميع الأخرى القائمة على الكثافة إلى توسيع تأثيره بشكل أكبر. من خلال الاستمرار في تحسين هذا البحث وتوسيعه ، نهدف إلى المساهمة في تطوير أدوات تجميع أكثر قوة ودقة وسهلة الاستخدام يمكنها فتح رؤى قيمة من البيانات المعقدة.

الكلمات المفتاحية: التجميع ، DBSCAN، KNN، التعلم غير الخاضع للإشراف ، التجميع القائم على الكثافة ، ضبط المعلمات ، استخراج البيانات ، التعلم الآلي

Abstract

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a widely used density-based clustering algorithm that can identify clusters of arbitrary shapes and handle noisy data effectively. However, the algorithm's performance hinges on the selection of two crucial parameters: Eps (epsilon) and MinPts (minimum points). Eps defines the radius within which to search for neighboring data points, while MinPts specifies the minimum number of points required to form a dense region.

The challenge of determining the optimal Eps value is a well-known issue in DBSCAN. A fixed Eps value can be inadequate for datasets containing clusters with varying densities. If Eps is set too small, low-density clusters might be misclassified as noise, whereas a large Eps value could lead to the merging of distinct high-density clusters.

In this thesis, we address this challenge by proposing a modified DBSCAN algorithm that eliminates the need for manual Eps specification. Our approach leverages the k-nearest neighbors (kNN) algorithm to adaptively calculate an Eps value for each data point based on its local neighborhood density. This adaptive Eps calculation allows our algorithm to effectively identifies clusters with heterogeneous densities, a task that traditional DBSCAN struggles with.

By automating the Eps selection process and tailoring it to the local density characteristics of the data, our modified DBSCAN algorithm enhances the flexibility and accuracy of density-based clustering. This contribution is particularly valuable for real-world datasets, which often exhibit diverse density distributions. Our approach simplifies the parameter tuning process, making DBSCAN more accessible and user-friendly, while also improving its ability to uncover complex cluster structures in data.

Keywords: clustering, DBSCAN, k-nearest neighbors, unsupervised learning, density-based clustering, parameter tuning, data mining, machine learning.

Résumé

Le partitionnement spatial de données basé sur la densité avec bruit (DBSCAN) est un algorithme de clustering largement utilisé, capable d'identifier des groupes de formes arbitraires et de gérer efficacement les données bruitées. Cependant, la performance de l'algorithme dépend du choix de deux paramètres cruciaux : Eps (epsilon) et MinPts (nombre minimum de points). Eps définit le rayon dans lequel rechercher les points de données voisins, tandis que MinPts spécifie le nombre minimum de points requis pour former une région dense.

La détermination de la valeur optimale d'Eps est un défi bien connu dans DBSCAN. Une valeur fixe d'Eps peut être inadéquate pour les ensembles de données contenant des clusters de densités variables. Si Eps est défini trop petit, les clusters de faible densité pourraient être classés à tort comme du bruit, tandis qu'une grande valeur d'Eps pourrait conduire à la fusion de clusters distincts de haute densité.

Dans cette thèse, nous relevons ce défi en proposant un algorithme DBSCAN modifié qui élimine le besoin de spécification manuelle d'Eps. Notre approche exploite l'algorithme des k plus proches voisins (kNN) pour calculer de manière adaptative une valeur Eps pour chaque point de données en fonction de la densité de son voisinage local. Ce calcul adaptatif d'Eps permet à notre algorithme d'identifier efficacement les clusters avec des densités hétérogènes, une tâche avec laquelle le DBSCAN traditionnel a du mal.

En automatisant le processus de sélection d'Eps et en l'adaptant aux caractéristiques de densité locales des données, notre algorithme DBSCAN modifié améliore la flexibilité et la précision du clustering basé sur la densité. Cette contribution est particulièrement précieuse pour les ensembles de données réels, qui présentent souvent des distributions de densité diverses. Notre approche simplifie le processus de réglage des paramètres, rendant DBSCAN plus accessible et convivial, tout en améliorant sa capacité à découvrir des structures de cluster complexes dans les données.

Mots-clés : clustering, DBSCAN, KNN, apprentissage non supervisé, clustering basé sur la densité, réglage des paramètres, data mining, apprentissage automatique.

Table of contents

List of Figures

List of Algorithms

1. General Introduction

1. 1 Context and Problematic:

Clustering is a fundamental task in data mining and machine learning, aiming to group similar data points together based on their inherent characteristics. This unsupervised learning technique plays a crucial role in uncovering hidden patterns, structures, and relationships within datasets, enabling valuable insights and informed decision-making across various domains. Density-based clustering, a prominent clustering approach, identifies clusters based on the concentration of data points in a region, allowing for the discovery of clusters with arbitrary shapes and sizes. This makes it particularly well-suited for analyzing complex and noisy data, where traditional clustering methods might struggle.

One of the most popular density-based clustering algorithms is DBSCAN (Density-Based Spatial Clustering of Applications with Noise). DBSCAN's effectiveness in handling noisy data and identifying clusters of arbitrary shapes has made it a widely used tool in diverse fields such as customer segmentation, image analysis, and anomaly detection. However, DBSCAN's performance is highly sensitive to the choice of its parameters, particularly the radius parameter (Eps), which defines the neighborhood of a data point for density calculations.

The selection of an appropriate Eps value is crucial for accurate clustering results. If Eps is set too small, low-density clusters might be misclassified as noise, leading to under-clustering. Conversely, if Eps is set too large, distinct high-density clusters might be merged, resulting in over-clustering. This sensitivity to the Eps parameter poses a significant challenge, especially when dealing with datasets containing clusters with heterogeneous densities, where a single fixed Eps value may not be suitable for all clusters.

1.2. Contributions:

This thesis introduces a modified DBSCAN algorithm that addresses the challenge of parameter tuning by eliminating the need for manual Eps specification. Our key contributions include:

Adaptive Eps Calculation: We leverage the k-nearest neighbors (kNN) algorithm to adaptively calculate an Eps value for each data point based on its local neighborhood density. This approach allows the algorithm to dynamically adjust the neighborhood size according to the density characteristics of different regions in the dataset.

Handling Heterogeneous Densities: By tailoring the Eps value to the local density, our algorithm effectively identifies clusters with varying densities, overcoming a limitation of traditional DBSCAN.

Improved Clustering Performance: Through extensive experimental evaluations on synthetic and real-world datasets, we demonstrate that our modified DBSCAN algorithm outperforms the traditional DBSCAN in terms of accuracy, cluster identification, and handling of noisy data.

Simplified Parameter Tuning: Our approach simplifies the parameter tuning process, making DBSCAN more accessible and user-friendly for practitioners.

1.3. Thesis Structure:

The thesis is organized as follows:

Chapter 1: General Introduction: Provides an overview of clustering algorithms, their applications, and the challenges they address.

Chapter 2: Clustering Algorithms: Explores various clustering techniques, including feature selection, probabilistic models, distance-based algorithms, density-based methods, and dimensionality reduction techniques.

Chapter 3: Deep Clustering: provides a comprehensive exploration of deep clustering's principles, architectures, and significance in modern data analysis

Chapter 4: Adaptive DBSCAN for Heterogeneous Density Clustering: Presents our modified DBSCAN algorithm with the adaptive Eps calculation and evaluates its performance on different datasets.

Chapter 5: Conclusion and Perspectives: Summarizes the contributions of the thesis and outlines potential future research directions.

2. Clustering algorithm

2.1. Introduction

The challenge of grouping data points into similar clusters has been extensively explored in data mining and machine learning due to its wide range of applications, including target marketing, segmentation, and summarization . Without specific labels, clustering provides a compact representation of the data, either as a summary or a generative model.

The fundamental task of clustering is to divide a collection of data points into groups that are as similar as possible [1]. This is a broad definition, and the specifics vary significantly based on the chosen model. For instance, similarity is defined using a probabilistic generative mechanism in a generative model, whereas a distance-based approach employs a standard distance function. Additionally, the nature of the data itself greatly influences how the problem is defined. Some common application domains in which the clustering problem arises are as follows [2,3,4]:

• **In collaborative filtering**, clustering groups together users with similar preferences. The ratings users give each other are then used to generate personalized recommendations across various applications.

• **Customer Segmentation:** This approach resembles collaborative filtering by grouping customers with similar preferences. However, instead of relying solely on user ratings, it leverages various attributes of the items themselves to form these clusters.

• **Data Summarization**: Clustering and dimensionality reduction techniques are closely linked, both serving as forms of data summarization. This process creates compact data representations that streamline processing and interpretation across diverse applications.

• **Dynamic Trend Detection**: Real-time clustering via dynamic and streaming algorithms uncovers key patterns and shifts in diverse social networking data, including time series text streams, and multidimensional data. This versatile approach enables the discovery of significant trends and events as they emerge.

• **Multimedia Data Analysis:** Diverse documents —such as images, audio, or video fall under multimedia data. Recognizing similar sections in this data has various applications, such as locating similar music clips or pictures. The challenge intensifies when dealing with multimodal data containing various types.

• **Biological Data Analysis**: With the success of the Human Genome Project and advances in gene expression data collection, biological data have become abundant. Clustering algorithms offer an effective way to analyze this data, often structured as sequences or networks, to uncover key trends and pinpoint atypical sequences

• **Social Network Analysis**: In these situations, analyzing the structure of a social network is used to identify key communities within it. Recognizing these communities is vital in social network analysis because it offers essential insights into the network's community structure. Additionally, clustering is significant in condensing social networks, which is advantageous in various scenarios.

• **A Foundation for Advanced Data Analysis**: Given that clustering condenses data into meaningful groups, it serves as a crucial building block for various core data mining tasks, such as classification and outlier detection. This concise data representation often unlocks valuable, context-specific insights.

The applications mentioned above demonstrate the diverse range of problems that clustering algorithms can address. Research in data clustering generally falls into several broad categories:

• **Technique-centered**: The growing popularity of clustering methodologies has led to the development and adoption of a broad range of diverse approaches, including probabilistic methods, distance-based techniques, spectral algorithms, density-based strategies, and dimensionality reduction-based methods. Each of these distinct approaches has its own set of advantages and disadvantages, making them particularly suitable for specific situations and unique domains. Particular data types, such as high-dimensional data, big data, or streaming data, present unique challenges and frequently require specialized techniques to address their complexities effectively.

• **Data-Type-Centered:** Diverse applications generate various data types with distinct properties. For instance, ECG machines produce highly correlated time series data, while social networks yield a mix of document and structural data. Common examples include categorical, time series, discrete sequences, network, and probabilistic data. The data's nature significantly influences the chosen clustering methodology. Moreover, certain data types pose greater challenges due to the separation between different attribute types, such as behavioral or contextual attributes.

2.2 Typical Approaches for Clustering

Problems related to clustering can be effectively resolved using a wide variety of methods. The data preprocessing phase itself demands specialized techniques. Numerous books and surveys delve into these topics [14, 20, 31, 37]. This section will discuss the most commonly used clustering techniques.

2.2.1 Feature Selection Methods

Feature selection, a crucial preprocessing step, significantly enhances the quality of clustering by removing noisy and irrelevant features. While closely related to dimensionality reduction, feature selection focuses on selecting original feature subsets, whereas dimensionality reduction techniques, such as principal component analysis [50], use linear combinations of features for further enhancement. The former prioritizes interpretability, while the latter requires fewer transformed directions for representation.

When conducting cluster analysis, simply applying global feature selection is often insufficient, especially in high-dimensional data. This is because different parts of the data might be best understood through various sets of features. For instance, some data points might correlate along one set of dimensions, while others correlate along a completely different set. Consequently, removing dimensions broadly can mean losing critical information. The solution lies in integrating feature selection directly into the clustering algorithm, leading to insights that are highly specific to localized data patterns. This approach drives high-dimensional subspace clustering and enables local dimensionality reduction.

.2.2.2 Probabilistic and Generative Approaches

In probabilistic models, the fundamental concept centers on modeling the data as if a specific process generated it. A generative model, such as a mixture of Gaussians, is assumed to represent this process accurately, and its various parameters are estimated systematically using the Expectation Maximization (EM) algorithm [7]. The dataset is used to estimate these parameters, thereby maximizing the likelihood that the data fit the model. This model is then used to determine the generative probabilities (or fit probabilities) for each data point. Data points that fit well into the distribution have high fit probabilities, while anomalies have low fit probabilities.

The fundamental concept of a mixture-based generative model is to assume that the data were generated from a mixture of k distinct distributions, represented as $G_1 \ldots G_k$.We

choose a specific data distribution characterized by an initial probability α_i, where the index i spans from 1 to k, allowing for the selection of one out of the k available distributions.

Generative models are usually approached through an EM method, beginning with either random or heuristic initialization. This process iteratively employs two steps to address the computational circularity:

- **E-Step**: Determine the expected likelihood of assigning each data point to a specific cluster based on current model parameters.
- **M-Step**: Refine the model parameters for each mixture component, utilizing the assignment probabilities as weights.

A key advantage of EM models is their flexibility in handling diverse data types, simply by selecting the appropriate generative model for each component G_r. This adaptability makes them a versatile tool for a wide range of clustering and machine learning tasks.

- **Numerical data:** Gaussian mixture models effectively represent each component of G_r.
- **Categorical data:** Bernoulli models describe the generation of discrete values for G_r.
- **Sequence data:** Hidden Markov Models (HMMs) explain the generation of sequences for G_r. Importantly, HMMs are mixture models themselves, featuring interdependent components connected through transitions. As a result, clustering sequence data with a mixture of HMMs can be viewed as a two-level mixture model.

Generative models are a cornerstone of clustering methods because they focus on understanding the fundamental process behind cluster formation. These models reveal interesting links to other clustering approaches when we examine specific scenarios involving prior probabilities or mixture parameters. For instance, if we assume equal prior probabilities and uniform mixture component radii, we arrive at a softer variant of the k-means algorithm.

2.2.3 Distance-Based Algorithms

Many specialized generative algorithms can be reframed as distance-based algorithms. This connection arises from the frequent use of distance functions within the probability distributions of mixture components in generative models. For instance, the Gaussian distribution expresses data generation probabilities in terms of the distance from

the mixture mean, illustrating the close ties between a Gaussian-based generative model and the k-means algorithm. Indeed, numerous distance-based algorithms can be considered simplifications or reductions of different generative model types.

Distance-based methods are frequently preferred due to their straightforward nature and adaptability to diverse situations. These algorithms can be broadly categorized into two distinct types.

a) **Flat**: In this case, data is divided into multiple clusters simultaneously, usually by employing partitioning representatives. The selection of these representatives and the distance function have a significant influence on the algorithm's behavior. Each iteration involves assigning data points to their nearest representative, followed by adjusting the representative based on the assigned points. Common methods for creating partitions include:
    - **k-Means**: It is a popular partitioning clustering algorithm that aims to group 'n' observations into 'k' clusters. It works by iteratively assigning each data point to the cluster whose mean (centroid) is closest, and then recalculating the centroid based on the new cluster assignments. The centroid is a synthetic point, not an actual data point, representing the average of all points within its cluster. This process minimizes the sum of squared Euclidean distances between data points and their respective centroids, making it efficient for large datasets. Its simplicity and speed contribute to its widespread use in various practical applications
    - **k-Medians**: use the median, rather than the mean, for partitioning representatives, making them more robust to noise and outliers since medians are less affected by extremes. The term "k-Medians" is sometimes confused with "k-Medoids," where representatives are selected from the dataset. While the research often conflates these terms, k-Medians and k-Medoid methods are distinct. The k-Medians approach described is actually a k-Medoids method, chosen for consistency with a specific paper. It is important to note this term confusion to avoid misunderstanding.

- **k-Medoids**: K-Medoids is a clustering algorithm that partitions a dataset into 'k' clusters, but instead of using a calculated mean as the cluster center like K-Means, it selects an actual data point, called a medoid, to represent each cluster. A medoid is chosen as the most centrally located data point within its cluster, minimizing its average distance to all other points in that cluster. This approach makes K-Medoids particularly resilient to outliers and capable of handling various data types, even those where a mean isn't meaningful. Its most common implementation, PAM (Partitioning Around Medoids), iteratively refines cluster assignments, though it can be computationally more intensive for large datasets.

b) Hierarchical: In these techniques, clusters are illustrated hierarchically using a dendrogram at different levels of detail. Based on whether this hierarchical structure is produced from the top-down or bottom-up approach, they can be classified as either divisive or agglomerative.

– **Agglomerative**: In these methods, a bottom-up approach starts with individual data points, successively merging clusters to create a tree structure. Various choices exist for merging clusters, balancing quality and efficiency, such as single-linkage, all-pairs linkage, centroid-linkage, and sampled-linkage clustering. Single-linkage uses the shortest distance between any pair in two clusters. All-pairs linkage averages all pairs, while sampled linkage uses a sample for the average distance. Centroid-linkage uses distances between centroids. Variations may cause chaining, where larger clusters attract more points due to naturally closer distances; single linkage clustering is particularly prone to this. Certain data domains, like network clustering, are also more affected by this.

-Divisive: A top-down approach partitions data into a tree structure, using any flat clustering algorithm for each partitioning step. Divisive partitioning provides flexibility in both tree hierarchy and cluster balance, without requiring perfectly balanced trees or branches of degree two. This enables various tradeoffs in node depths and weights.

2.2.4 Density- and Grid-Based Methods

Density- and grid-based methods represent interconnected approaches that intricately examine the data space at an exceptionally fine-grained level. Density at a specific point in the data space is determined either by counting the number of data points within a predetermined volume surrounding that point or by employing a smoother kernel density estimate to achieve a similar purpose. The data space is typically analyzed with a high level of meticulous detail, which is followed by a post-processing phase aimed at combining these identified dense regions into arbitrary, more significant shapes. Grid-based methods, considered a subset of density-based methods, structure these explored and analyzed regions into a grid-like format, which enhances the overall process. This grid structure substantially simplifies the post-processing step of merging dense blocks into larger regions and is particularly advantageous for high-dimensional data scenarios where lower-dimensional grids can be used to define clusters on subsets of dimensions .

One significant advantage of density- and grid-based methods is their capability to fully recreate the shape of the data distribution through a detailed examination of the data space. DBSCAN is a well-known example of density-based clustering, while STING is a classic instance of grid-based clustering. Nonetheless, density-based techniques encounter a challenge because they are fundamentally suited for continuous spaces, restricting their use in discrete or non-Euclidean spaces unless specific transformations are employed. This limitation makes them less appropriate for certain data types, such as time series, without tailored modifications. Furthermore, as dimensionality increases, the complexity of density calculations also rises due to the growing number of grid cells and the increasing sparsity within the data.

.

2.2.5 Dimensionality Reduction Methods

Dimensionality reduction, feature selection, and clustering are closely interconnected, as they all utilize relationships between dimensions to reduce the data representation. Dimensionality reduction can be considered a type of clustering applied to data features instead of individual data points, employing correlation or proximity analysis. This naturally leads to the question of whether simultaneously clustering both rows and columns would yield better results. This broader idea has led to the development of algorithms like matrix factorization, spectral clustering, probabilistic latent semantic

indexing, and co-clustering, all based on this core concept. These methods share similarities with projected clustering, often used for high-dimensional data.

2.2.5.1 Generative Models for Dimensionality Reduction

In these models, a generative probability distribution is used to model the complex relationships between data points and dimensions simultaneously. For example, a generalized Gaussian distribution can represent a mixture of arbitrarily oriented clusters, and the EM algorithm can be used to learn its parameters. However, the effectiveness of this approach decreases as dimensionality increases, due to the increasing number of parameters involved in the learning process [50]. EM methods, known for their sensitivity to overfitting when dealing with numerous parameters, prioritize retaining all information through soft probabilities rather than making decisive choices about point and dimension selection, unlike nonparametric methods [66]. Nevertheless, generative models have proven successful in learning specific cases across various data types.

Topic modeling, often referred to as PLSI, is a popular dimensionality reduction technique for analyzing text data. It involves clustering both words and documents into meaningful groups called topics. The core parameters learned are the probabilities of words belonging to specific topics and documents belonging to specific topics, resulting in soft clusters from both a word and document perspective. This holistic approach has been highly successful in the field of text mining and has led to the development of various extensions, such as Latent Dirichlet Allocation (LDA), which builds upon this principle by incorporating more flexible assumptions about the data.

2.2.5.2. Spectral Methods

Spectral methods present an alternative to traditional dimensionality reduction techniques. They operate on the similarity or distance matrix of the data, rather than the raw data points and dimensions. This allows for the analysis of arbitrary objects and simultaneously embeds them into Euclidean space while reducing dimensionality. This makes spectral methods popular for clustering complex objects, such as node sets in graphs. However, there are drawbacks[34,35]. The use of an n x n similarity matrix leads to a computational complexity that scales with the square of the number of data points. Additionally, determining the eigenvectors of this matrix can be computationally expensive. Moreover, creating lower-dimensional representations for new data points is

challenging. In cases where data is multidimensional but not excessively noisy, using a large similarity matrix may be unnecessary.

2.3 Data Types Studied in Cluster Analysis

The type of data significantly influences the choice of clustering algorithm. Initially, clustering algorithms were mainly designed for numerical data; however, real-world datasets contain various formats, such as categorical, temporal, and structural data. This section explores how data type affects clustering and offers a brief overview of the different data types found in real-world applications scenarios.

2.3.1 Clustering Categorical Data

Real-world datasets often include categorical data as attributes, such as sex, race, or zip code, which are discrete and unordered. Mixed datasets combine numerical attributes like salary with categorical ones. Market basket data is a form of categorical data where attributes are binary, indicating item presence or absence in transactions. Categorical datasets pose challenges for clustering algorithms: Standard metrics can't be used if algorithms rely on similarity or distance calculations. New similarity measures for categorical data must be used. K-means and k-medians typically form cluster representatives by calculating data point means or medians within clusters. While natural for numerical data, means and medians need modification for discrete data. Mixed data clustering is complex due to attribute variability. Numerical and categorical attributes require distinct similarity functions for effective treatment. A clustering algorithm's effectiveness depends on the data type. Some models are flexible and accommodate various data types with suitable similarity functions; others are specialized. Spectral clustering is versatile and applicable to any data type with a suitable similarity function but its computational cost rises with the square of data points. Generative models are adaptable to different data types if generative models are defined for each component. Specific algorithms developed for categorical data clustering include [18,28]

2.3.2 Clustering Text Data

The growing popularity of the web and social networks has made text data increasingly common. Text data is typically represented in vector space format, where word order is ignored, and the focus is on the words themselves. This representation allows text clustering methods to be applied to other set-based attributes as well. However, text data has specific characteristics that need to be taken into account:

Text data tends to be very high-dimensional and sparse. This means there's a large vocabulary of possible words, but each document only uses a small subset of them. As a result, most word counts in text data are zero.

Word frequencies, represented as attribute values in text data, are naturally non-negative. This inherent non-negativity plays a significant role in the effectiveness of various co-clustering and matrix factorization techniques.

Initial approaches to text clustering, like scatter-gather, relied on distance-based methods, often combining k-means and agglomerative clustering. Topic modeling subsequently gained traction, utilizing the EM framework to create soft clusters of words and documents. PLSI and LDA are prominent examples, considered soft variations of co-clustering and matrix factorization. Spectral methods, frequently employed to partition documents and words simultaneously, construct bipartite similarity graphs to represent their membership relations. This connection isn't surprising, as matrix factorization and spectral clustering are closely related.

2.3.3 Clustering Multimedia Data

Social media's growing popularity has brought with it a wealth of multimedia data, including images, audio, and video, which can be analyzed using clustering methods. Sites like Flickr and YouTube are prime examples of platforms hosting massive amounts of such data. Even on traditional social networks and the wider web, multimedia data of different types is prevalent. In many instances, this multimedia data coexists with more conventional text-based information.

The heterogeneity and context-dependence of multimedia data make clustering a complex task. Often multimodal or contextual, this data includes both content-related attributes (behavior) and attributes defining relationships or positions (context). Image data, for example, contains pixel values (behavior) within a specific spatial context, while video and music data rely on temporal context for meaning. Addressing this complexity requires careful data representation and analysis. The quality of multimedia analysis outcomes is heavily influenced by how the data is represented, making it a key factor in the entire process.

2.3.4 Clustering Time-Series Data

Time-series data is ubiquitous in sensor readings, financial markets, and other applications involving temporal tracking or forecasting. The distinguishing feature of time-series data is the interdependence of data points, where the value at any given time depends on previous values. This dependency is captured by two attributes: a contextual attribute (time) and a behavioral attribute (the data value). Due to the varied nature of time-series problems, there exists a wide array of clustering methods, broadly classified into correlation-based online analysis and shape-based offline analysis, depending on the specific objectives and constraints.

In the context of time-series data, correlation-based online analysis dynamically tracks the correlations among different data streams to create evolving clusters. This methodology proves valuable in applications such as sensor selection and forecasting, especially when there are time lags within the correlated patterns of the clustered streams. In the stock market, this approach enables the real-time maintenance of groups of related stock tickers based on their fluctuating correlation patterns. Distance functions between different series are continuously calculated using their mutual regression coefficients. Examples of this approach include the MUSCLES method and large-scale time-series correlation monitoring techniques.

Unlike correlation-based analysis, shape-based offline analysis of time-series data disregards the specific time of data collection, focusing instead on the overall patterns or shapes of the series. This is particularly relevant in scenarios where the shape, rather than the precise timing, is the key factor for comparison, such as in clustering ECG data from patients. Distance functions play a crucial role in this approach because time series often have varying scales and may exhibit time warping, where temporal adjustments are needed for accurate alignment. As with other time-series methods, the design of the distance function significantly influences the effectiveness of the shape-based approach.

2.5 Evaluation Metrics for Clustering

Evaluating the performance of clustering algorithms is crucial but challenging due to the unsupervised nature of the task. Metrics are broadly categorized into external and internal measures.

2.5.1 External Evaluation Metrics

When ground truth labels are available (e.g., for benchmarking or experimental evaluation), external metrics offer a powerful way to assess how well the clusters align with the true underlying classes.

- **Accuracy** (ACC): Measures the proportion of data points that are correctly clustered. It requires finding the best mapping between cluster labels and true labels.

$$ACC = \frac{\sum_{i=1}^{N} \mathbb{I}(\text{map}(c_i)=l_i)}{N} \quad 2.1$$

where c_i is the cluster label assigned by the algorithm, l_i is the true label, map() is a permutation mapping, and $\mathbb{I}(\ \ )$ is the indicator function.

- **Normalized Mutual Information** (NMI): Measures the mutual dependence between two clusterings (the predicted clustering and the true labeling). It's a normalization of the Mutual Information (MI) score, ranging from 0 (no mutual information) to 1 (perfect correlation). NMI is robust to the number of clusters and cluster sizes [31].

$$NMI(U,V) = \frac{MI(U,V)}{\text{avg}(H(U),H(V))} \quad 2.2$$

where U is the set of true labels, V is the set of predicted cluster labels, MI is Mutual Information, and H is entropy.

- **Adjusted Rand Index** (ARI): Measures the similarity between two clusterings, correcting for chance. It considers all pairs of samples and counts pairs that are placed in the same or different clusters in both the true and predicted clusterings. ARI ranges from -1 (worse than random) to 1 (perfect match), with 0 indicating random assignment.

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]} \quad 2.3$$

where RI is the Rand Index and $E[RI]$ is its expected value.

$$RI = \frac{TP+TN}{TP+FP+FN+TN} \quad 2.4$$

- **Homogeneity:** Measures whether each cluster contains only data points belonging to a single class.

- **Completeness:** Measures whether all data points belonging to a given class are assigned to the same cluster.
- **V-measure:** The harmonic mean of homogeneity and completeness.

Ranges for clustering metrics: ACC ∈ [0,1] (1 = all labels correct), NMI ∈ [0,1] (1 = identical partitions; 0 = independent), ARI ∈ [−1,1] (1 = perfect agreement; 0 = chance-level; negative = worse than chance).

2.5.2 Internal Evaluation Metrics

These metrics assess the quality of a clustering based on the intrinsic properties of the data and the clusters themselves, without relying on external labels. They are crucial for real-world unsupervised scenarios and for hyperparameter tuning.

- **Silhouette Score**: Measures how similar a data point is to its cluster compared to other clusters. For each point, it calculates an average intra-cluster distance (a) and an average nearest-cluster distance (b). The silhouette score ranges from -1 to 1, where higher values indicate better-defined clusters.

  $$s(i) = \frac{b(i)-a(i)}{max(a(i),b(i))} \quad 2.4$$

  The overall score is the mean silhouette score of all samples.
- **Davies-Bouldin Index:** Measures the average "similarity" between clusters, where similarity is a ratio of within-cluster scatter to between-cluster separation. A lower Davies-Bouldin index indicates a better clustering.
- **Calinski-Harabasz Index (Variance Ratio Criterion):** Measures the ratio of between-cluster dispersion to within-cluster dispersion. Higher values correspond to better-defined clusters.

The choice of evaluation metric depends on the specific goals of the clustering task and the availability of ground truth. For clustering, external metrics are often used in research papers for comparative analysis, while internal metrics are practical for real-world applications where labels are scarce.

2.6 Conclusions

Clustering is a fundamental and versatile technique in data mining and machine learning, addressing the challenge of grouping data points based on similarity. Research in data clustering encompasses a wide range of techniques, including probabilistic and distance-based methods, spectral algorithms, density-based strategies, and dimensionality reduction approaches. Each technique offers unique advantages and disadvantages, tailored to specific data types and application scenarios. Data preprocessing, particularly feature selection, plays a crucial role in enhancing the quality of clustering, especially in high-dimensional datasets.

Different data types, such as categorical, text, multimedia, and time-series data, present distinct challenges and require specialized algorithms and similarity measures. Evaluating clustering performance is essential, with external metrics used when ground truth labels are available and internal metrics used in unsupervised scenarios. Both external and internal metrics provide valuable insights into cluster quality, aiding in algorithm selection and hyperparameter tuning. Overall, clustering offers powerful tools for uncovering hidden patterns, summarizing complex data, and deriving valuable insights across diverse domains.

3. Deep Clustering

3.1 Introduction

The exponential growth of data in terms of volume, velocity, and variety has presented both opportunities and significant challenges for data analysis. A pervasive challenge lies in the effective organization and interpretation of high-dimensional, unstructured data, such as images, text, and biological sequences. Traditional clustering algorithms, while powerful for lower-dimensional and well-structured data, often struggle to cope with the inherent complexities of raw, high-dimensional datasets. Their performance usually degrades due to the "curse of dimensionality," where distance metrics become less meaningful, and the sparsity of data makes it challenging to identify dense regions [40].

Deep Clustering emerges as a cutting-edge solution at the intersection of deep learning and unsupervised learning. It leverages the formidable representation learning capabilities of deep neural networks to extract discriminative, low-dimensional features directly from complex raw data. Instead of relying on hand-crafted features or fixed mappings, deep learning models can learn an optimal feature space where data points belonging to the same inherent cluster are brought closer together. In contrast, those from different clusters are pushed apart [41]. This learned representation then facilitates more effective and robust clustering by traditional or specialized clustering techniques. The core idea is to jointly optimize the feature learning process and the clustering objective, leading to an end-to-end learning framework that often outperforms methods relying on separate, sequential steps of feature extraction and clustering. Deep clustering has become indispensable in domains where data complexity and scale are paramount, offering a powerful avenue for unsupervised pattern discovery.

- **Handling High-Dimensionality and Intrinsic Complexity:** Traditional clustering algorithms often struggle with the "curse of dimensionality," where the effectiveness of distance metrics diminishes in high-dimensional spaces, making it difficult to find meaningful clusters. Deep clustering, by learning a lower-dimensional, more abstract, and discriminative feature representation, effectively mitigates this issue. It can uncover intricate, non-linear relationships that are hidden in the raw, high-dimensional data, leading to more accurate and robust clusterings.

- **Automated Feature Learning (End-to-End Optimization):** A major advantage of deep clustering is its capacity for automated feature extraction. Unlike traditional methods that rely on pre-defined or hand-crafted features (which can be time-consuming, domain-specific, and sub-optimal), deep learning models learn the most relevant features directly from the raw data. This end-to-end optimization, where feature learning and clustering are jointly optimized, allows the network to learn representations that are specifically tailored to the clustering task, resulting in superior performance.
- **Scalability to Large Datasets:** Deep learning architectures are inherently designed to handle and learn from massive datasets. This scalability makes deep clustering particularly well-suited for the "big data" era, where traditional algorithms might become computationally prohibitive or ineffective.
- **Superior Performance on Unstructured Data:** For complex unstructured data types like images, video, and text, deep learning has demonstrated unparalleled success in learning rich, semantic representations. Deep clustering extends this success to unsupervised settings, enabling the discovery of meaningful groupings in data where traditional methods would fail due to the lack of a suitable feature space.
- **Reduced Need for Feature Engineering:** By automating the feature learning process, deep clustering significantly reduces the need for laborious and expert-dependent feature engineering, making the clustering pipeline more efficient and accessible.
- **Foundation for Downstream Tasks:** The high-quality, learned representations produced by deep clustering can serve as excellent inputs for various downstream machine learning tasks, even supervised ones, further highlighting its utility beyond just clustering.

In essence, deep clustering is important and prominent because it bridges the gap between the powerful representation capabilities of deep learning and the critical need for unsupervised pattern discovery in the age of complex, high-dimensional data. It provides a robust and often superior solution for organizing and understanding data where traditional methods fall short.

3.2 Deep learning architectures

A solid understanding of deep clustering necessitates a grasp of its two foundational pillars: traditional clustering algorithms and deep learning architectures.

3.2.1 Deep Neural Networks for Representation Learning

Deep learning, a subfield of machine learning, utilizes **Deep Neural Networks (DNNs)**, which are artificial neural networks with multiple hidden layers, to learn complex patterns and representations directly from raw data. These networks excel at automatically extracting hierarchical features, moving from simple low-level features to complex high-level abstractions. The "deep" aspect refers to the number of hidden layers, which allows DNNs to model highly non-linear relationships and learn increasingly abstract representations of the input data.

A typical DNN consists of:

- **Input Layer:** Receives the raw data.
- **Hidden Layers:** One or more layers between the input and output layers, where the actual learning and feature extraction occur. Each neuron in a hidden layer applies a weighted sum of its inputs and passes it through an activation function (e.g., ReLU, sigmoid, tanh).
- **Output Layer:** Produces the final prediction or representation.

The training of DNNs involves **backpropagation** and **gradient descent** (or its variants like Adam, RMSprop) to iteratively adjust the weights and biases of the network to minimize a defined loss function. DNNs form the backbone of many advanced deep learning architectures used for representation learning in deep clustering.

3.2.1 Autoencoders (AEs)

Autoencoders are a type of unsupervised neural network architecture specifically designed to learn a compressed, low-dimensional representation (encoding) of input data. The core idea is to learn an identity function in an unsupervised manner, forcing the network to capture the most salient features of the input data in its bottleneck layer.

An autoencoder consists of two main parts:

- **Encoder:** Maps the high-dimensional input data (x) to a lower-dimensional latent space (or bottleneck layer, z). This process can be represented as $z = f(x)$, where f is the encoder function, typically a neural network.
- **Decoder:** Reconstructs the original input data from the latent space representation (z). This process is $\hat{x} = g(z)$, where g is the decoder function, another neural network.

The autoencoder is trained by minimizing the **reconstruction error** (e.g., Mean Squared Error for continuous data, binary cross-entropy for binary data) between the original input x and its reconstruction $\hat{x}$. The learned latent space z is then considered a robust, dense, and meaningful feature representation, often used as input for subsequent tasks like clustering.

Variants of Autoencoders:

- **Denoising Autoencoders (DAEs):** Learn robust features by reconstructing the original input from a corrupted version (e.g., input with added noise or missing values). This forces the network to learn more robust representations that are less sensitive to input perturbations.
- **Sparse Autoencoders:** Introduce a sparsity constraint on the activations of the hidden layer neurons, forcing only a small number of neurons to be active at any given time. This encourages the network to learn more specialized and interpretable features.
- **Contractive Autoencoders (CAEs):** Add a penalty to the loss function that encourages the learned representation to be robust to small variations in the input data.

3.2.2 Variational Autoencoders

Variational Autoencoders (VAEs) are a powerful type of generative model that extends the concept of autoencoders by introducing a probabilistic approach to the latent space. Unlike standard AEs which learn a deterministic mapping to a point in the latent space, VAEs learn to map the input data to a **distribution** (typically a Gaussian distribution) in the latent space.

The encoder of a VAE outputs two vectors for each input:

- **Mean vector (μ):** Represents the mean of the latent distribution.
- **Standard deviation vector (σ or $\log \sigma^2$):** Represents the variance of the latent distribution.

During training, a latent vector z is sampled from this learned distribution ($z \sim N(\mu, \sigma^2)$) using the reparameterization trick to allow for backpropagation. This sampled z is then fed to the decoder to reconstruct the input.

The VAE's loss function consists of two main components:

1. **Reconstruction Loss:** Measures how well the output of the decoder matches the original input (similar to AEs).
2. **KL Divergence Loss:** Measures the difference between the learned latent distribution and a prior distribution (typically a standard normal distribution). This term acts as a regularizer, forcing the latent space to be continuous and well-structured, which is highly beneficial for generative tasks and can implicitly aid clustering by creating disentangled and meaningful latent dimensions.

VAEs are particularly useful for deep clustering because their structured and continuous latent spaces often lead to more semantically meaningful embeddings where clusters are more naturally formed and separable. They also allow for generative capabilities, enabling the creation of new data points that belong to learned clusters.

3.2.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are a class of deep learning models composed of two competing neural networks: a **generator** and a **discriminator**. They are trained in an adversarial process, akin to a zero-sum game, to learn to generate new data instances that are indistinguishable from real data.

- **Generator (G):** Takes a random noise vector (z) as input and transforms it into a synthetic data sample (e.g., an image). Its goal is to produce data that can fool the discriminator into believing it's real.
- **Discriminator (D):** Takes either a real data sample from the training set or a synthetic sample from the generator as input. Its goal is to accurately distinguish between real and fake data.

During training, the generator and discriminator are trained simultaneously in an alternating fashion:

1. The **discriminator** is trained to correctly classify real samples as real and generated samples as fake.
2. The **generator** is trained to produce samples that the discriminator classifies as real.

This adversarial process drives both networks to improve: the generator learns to produce increasingly realistic data, and the discriminator becomes better at detecting subtle differences between real and fake.

While primarily generative, some GAN variants are designed to learn disentangled representations in their latent space, which can then be used for clustering. For example:

- **InfoGAN:** Extends GANs by maximizing the mutual information between a small subset of the latent variables and the observations. Some of these latent variables can be interpreted as categorical codes, effectively representing clusters. The generator learns to produce images conditioned on these codes, while the discriminator learns to predict these codes in addition to distinguishing real from fake. This intrinsic clustering capability allows for unsupervised classification.
- **CatGAN (Categorical Generative Adversarial Networks):** Aims to discover and separate different categories of data in an unsupervised manner. It modifies the discriminator to output probabilities over predefined or learned categorical labels, in addition to the real/fake probability. The generator learns to produce samples corresponding to these categories.

3.2.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized type of DNN primarily designed for processing grid-like data, such as images (2D grids of pixels), but also applicable to video (3D grids) and time series (1D grids). They are highly effective at automatically and adaptively learning spatial hierarchies of features from raw input data.

Key components of CNNs:

- **Convolutional Layers:** Apply a set of learnable filters (kernels) that slide across the input data, performing convolution operations. Each filter detects specific

patterns (e.g., edges, textures, shapes) at different locations. This process generates feature maps.

- **Activation Functions:** Non-linear functions (e.g., ReLU) applied after convolutional layers to introduce non-linearity, allowing the network to learn complex patterns.
- **Pooling Layers (e.g., Max Pooling, Average Pooling):** Downsample the feature maps, reducing their spatial dimensions and making the learned features more robust to small translations and distortions. This also helps reduce computational complexity.
- **Fully Connected Layers:** Typically found at the end of the CNN, these layers connect every neuron from the previous layer to every neuron in the current layer, similar to traditional neural networks. They are responsible for making the final predictions or generating the learned feature representation.

For image clustering tasks, CNNs are indispensable. The features extracted from the later layers of a pre-trained CNN (e.g., VGG, ResNet, Inception) often serve as highly powerful and discriminative representations for images. These pre-trained features can then be fed into traditional clustering algorithms or integrated into end-to-end deep clustering frameworks to group similar images effectively. The hierarchical feature learning of CNNs allows them to capture abstract visual concepts that are crucial for distinguishing between different image clusters.

3.2.5 Recurrent Neural Networks and Transformers

These architectures are designed for processing sequential data, where the order and context of elements are crucial.

- **Recurrent Neural Networks (RNNs):** Specifically built to handle sequences by maintaining an internal "hidden state" that acts as a memory of previous elements in the sequence. This allows information to persist across time steps. However, basic RNNs suffer from the vanishing/exploding gradient problem, making it difficult to learn long-term dependencies.
    - **Long Short-Term Memory (LSTM) Networks:** A special type of RNN that addresses the vanishing gradient problem through sophisticated "gates" (input, forget, output gates) that control the flow of information into and out of the cell

state. LSTMs are highly effective at learning long-term dependencies in sequential data.

- **Gated Recurrent Units (GRUs):** A simpler variant of LSTMs, offering similar performance with fewer parameters by combining the input and forget gates into an update gate.

- **Transformers:** A revolutionary architecture introduced in 2017, primarily for sequence-to-sequence tasks, especially in Natural Language Processing (NLP). Unlike RNNs, Transformers do not rely on recurrence or convolutions. Instead, they rely heavily on a mechanism called "**self-attention**," which allows the model to weigh the importance of different parts of the input sequence when processing each element, regardless of their distance in the sequence. This parallelizable attention mechanism makes Transformers highly efficient for very long sequences.

For clustering sequential data like text documents, time series, or DNA sequences, RNNs and especially Transformers are crucial. They learn contextual embeddings for words, sentences, or entire sequences that capture semantic meaning and temporal dependencies. These learned embeddings can then be used as input for clustering algorithms to group similar documents, identify patterns in time series data, or cluster biological sequences. Transformers, with their ability to capture long-range dependencies, have been particularly impactful in learning rich representations for large language models, which can then be fine-tuned or used for clustering text data.

3.2.6 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate directly on graph-structured data. Unlike traditional neural networks that assume data points are independent or arranged in grids/sequences, GNNs leverage the relational information present in graphs (nodes and edges). Each node in a graph has features, and its connections to other nodes (its neighborhood) provide contextual information.

GNNs learn node representations (embeddings) by iteratively aggregating information from a node's neighbors. In essence, each node's representation is updated based on its own features and the aggregated features of its neighbors. This process is repeated for several layers, allowing information to propagate across the graph and enabling nodes to learn representations that capture their local and global structural context.

Key GNN Architectures:

- **Graph Convolutional Networks (GCNs):** A popular type of GNN that generalizes the concept of convolution to graph data.
- **Graph Attention Networks (GATs):** Incorporate an attention mechanism to assign different weights to different neighbors, allowing the model to focus on more relevant connections.

When data has an inherent graph structure (e.g., social networks, molecular structures, citation networks, protein-protein interaction networks), GNNs are invaluable for learning node embeddings that capture both node attributes and relational information. These learned embeddings are highly discriminative and can be directly clustered using traditional clustering algorithms (e.g., K-Means, DBSCAN) or integrated into end-to-end deep clustering frameworks specifically designed for graphs. GNNs enable deep clustering to identify communities or groups within complex relational data that would be difficult to discern with methods that ignore the graph structure.

3.3 Categories and Approaches to Deep Clustering

Deep clustering methods can be broadly categorized based on how they integrate deep learning with the clustering objective. The common thread is the joint optimization of feature learning and cluster assignment.

3.3.1 Deep Embedded Clustering

Deep Embedded Clustering (DEC) is a seminal work in end-to-end deep clustering, proposed by Xie, Girshick, and Farhadi [26]. It aims to learn a mapping from the data space to a lower-dimensional feature space where clustering is performed. The key idea is to iteratively refine clusters by optimizing two objectives:

1. **Feature Learning:** Using an autoencoder (or a similar deep network) to learn a robust embedding of the input data. The encoder part of the autoencoder maps the high-dimensional data to the low-dimensional embedding space.
2. **Clustering Objective:** Optimizing a specific clustering loss function directly in the embedded space. DEC utilizes a Kullback-Leibler (KL) divergence-based objective to iteratively refine cluster assignments by moving each data point closer to its assigned cluster centroid while simultaneously adjusting the centroids towards the mean of their designated points.

DEC Process:

- **Initialization:** Pre-train an autoencoder to obtain an initial set of features (embedding) and then initialize cluster centroids by running a standard clustering algorithm (e.g., K-Means) on these initial embeddings.
- **Iterative Refinement:**
  - **Step 1: Soft Assignment:** Compute a soft assignment (probability) for each data point belonging to each cluster using a Student's t-distribution as a kernel, which measures the similarity between an embedded point and a cluster centroid.
  - **Step 2: Target Distribution:** Compute a "target distribution" that emphasizes high-confidence assignments. This target distribution serves as strong regularization, enabling the network to learn more discriminative features and pushing points towards their assigned cluster centroids.
  - **Step 3: KL Divergence Loss:** Minimize the KL divergence between the current soft assignments and the target distribution. This loss function drives the network to learn embeddings that match the desired cluster assignments.
  - **Step 4: Update Network and Centroids:** Update the encoder network's weights and the cluster centroids using gradient descent.

Variations of DEC:

- **Improved Deep Embedded Clustering (IDEC):** Adds a reconstruction loss term back to the DEC objective during the fine-tuning phase. This helps preserve the local structure of the data and prevents degeneration where the embedding space becomes too compact or loses meaningful information for reconstruction [42].
- **Deep Clustering Network (DCN):** Similar to DEC/IDEC, but uses an optimization strategy for the clustering objective, integrating K-Means loss directly into the autoencoder's training [43].

3.3.2 Autoencoder-Based Clustering

This approach involves two distinct steps, though often integrated:

1. **Deep Feature Learning (Pre-training):** Train an autoencoder (or a similar deep neural network) in an unsupervised manner to learn a low-dimensional, dense, and

meaningful representation of the input data. The encoder part of the autoencoder effectively acts as a feature extractor.

2. **Traditional Clustering:** Apply a traditional clustering algorithm (e.g., K-Means, GMM) directly on the learned latent space (the output of the encoder) instead of the original high-dimensional raw data.

Advantages: Simple to implement, leverages powerful deep feature extractors.

Disadvantages: The feature learning is not directly optimized for the clustering task, meaning the learned latent space might not be ideal for separating clusters. The two steps are typically disjoint, potentially leading to sub-optimal overall performance compared to joint optimization. However, it serves as a strong baseline and is often improved upon by integrated methods.

3.3.3 Generative Model-Based Clustering

This category leverages the power of generative models, particularly Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs).

- **GAN-based Clustering (e.g., InfoGAN, CatGAN):** Some GAN variants are designed to learn disentangled representations in their latent space, which can then be used for clustering.
  - **InfoGAN:** Extends GANs by maximizing the mutual information between a small subset of the latent variables and the observations. Some of these latent variables can be interpreted as categorical codes, effectively representing clusters. The generator learns to produce images conditioned on these codes, while the discriminator learns to predict these codes in addition to distinguishing real from fake. This intrinsic clustering capability allows for unsupervised classification.
  - **CatGAN (Categorical Generative Adversarial Networks):** Aims to discover and separate different categories of data in an unsupervised manner. It modifies the discriminator to output probabilities over predefined or learned categorical labels, in addition to the real/fake probability. The generator learns to produce samples corresponding to these categories.
- **Variational Autoencoder (VAE)-based Clustering:** VAEs learn a probabilistic encoding to a structured latent space. By imposing specific priors on this latent space (e.g., a mixture of Gaussians), the VAE can implicitly learn cluster

assignments. The "clusters" correspond to the components of the mixture model in the latent space. Approaches like Deep Gaussian Mixture Models (DGMM) fall into this category, where a deep neural network learns to map data to a latent space where a GMM is then applied.

3.3.4 End-to-End Deep Clustering (Joint Optimization)

This is the most advanced and often best-performing category, where feature learning and cluster assignment are optimized jointly within a single deep learning framework. The loss function directly encourages the network to learn features that are discriminative for clustering while simultaneously refining the cluster assignments.

- **DEC/IDEC :** Excellent examples of end-to-end learning.
- **Contrastive Learning for Clustering:** Recent advancements in contrastive learning, which learns embeddings by pushing similar samples closer and dissimilar samples apart, have been adapted for clustering. The idea is to enforce that embeddings of samples within the same cluster are more similar than those from different clusters, even without explicit labels [29].
- **Adversarial Clustering:** Methods that use adversarial training principles (similar to GANs) where one part of the network aims to learn good representations for clustering, while another part tries to challenge it, leading to more robust and discriminative features.
- **Graph Neural Networks (GNNs) for Clustering:** When data has inherent graph structures (e.g., social networks, molecular structures), GNNs can learn node embeddings that capture structural and relational information. These embeddings can then be directly clustered or used in an end-to-end fashion .

3.3.5 Hybrid Deep Clustering Approaches

These methods combine elements from different categories or integrate deep learning with specific aspects of traditional clustering algorithms beyond simple K-Means.

- **Deep DBSCAN:** Attempts to combine deep feature learning with the density-based approach of DBSCAN. This often involves learning an embedding where density is meaningful for DBSCAN to identify clusters more effectively in complex data.

- **Neural Network with Custom Loss Functions:** Designing specific loss functions that explicitly promote clustering objectives within the deep network's training, beyond simple reconstruction or KL divergence. These losses might involve measures of cluster compactness, separation, or specific properties tailored to the data.

3.4 Challenges in Deep Clustering

Despite its promise, deep clustering presents several unique challenges that differentiate it from traditional clustering and even other deep learning tasks.

- **Joint Optimization Difficulty:** Optimizing both the feature learning (deep network weights) and the clustering objective simultaneously is inherently challenging. The two objectives can sometimes conflict or lead to local optima, making convergence tricky. A sub-optimal embedding can lead to poor clustering, and incorrect cluster assignments can mislead feature learning.
- **Initialization Sensitivity:** Similar to traditional clustering, deep clustering algorithms (especially those based on centroid optimization like DEC) can be sensitive to the initial conditions of the network weights and cluster centroids. Poor initialization can lead to sub-optimal clusterings. Pre-training with autoencoders is a common strategy to mitigate this, but finding the truly optimal starting point remains difficult.
- **Determining the Number of Clusters (k):** Many deep clustering algorithms still require the number of clusters (k) as a hyperparameter. Estimating k in an unsupervised setting is a long-standing challenge in clustering, and deep learning doesn't inherently solve this. Heuristics like the elbow method or silhouette score in the learned embedding space are often used, but they might not be fully reliable.
- **Evaluation in Unsupervised Settings:** Evaluating clustering performance is complex because there are no true labels. Metrics like purity, NMI (Normalized Mutual Information), and ARI (Adjusted Rand Index) require ground truth labels, which are often unavailable in real-world unsupervised scenarios. Intrinsic metrics (e.g., silhouette score, Davies-Bouldin index) can be used but don't always correlate perfectly with perceived cluster quality. This makes objective comparison and hyperparameter tuning difficult.

- **Interpretability of Latent Space:** While deep learning learns powerful representations, the features in the latent space are often abstract and difficult for humans to interpret. Understanding *why* certain points are grouped together or how the network arrived at a particular clustering remains a "black box" problem, hindering trust and debugging.
- **Computational Cost:** Training deep neural networks, especially those with complex architectures or iterative fine-tuning stages, can be computationally very expensive and time-consuming, requiring significant hardware resources (GPUs).
- **Hyperparameter Tuning:** Deep clustering models typically have numerous hyperparameters (e.g., learning rates, network architecture, regularization strengths, clustering-specific parameters like eps for DBSCAN, k for K-Means). Tuning these parameters optimally is a significant challenge.
- **Scalability:** While deep learning can handle large datasets, scaling deep clustering algorithms to truly massive datasets remains an active area of research, especially for those that involve pairwise distance computations or complex iterative updates.

Overcoming these challenges is a primary focus of ongoing research in deep clustering, aiming to develop more robust, generalizable, and user-friendly algorithms.

3.5 Conclusion

Deep clustering represents a powerful and evolving paradigm that addresses the fundamental challenge of uncovering intrinsic structures within complex, high-dimensional, and often unlabeled data. By synergistically integrating the representation learning capabilities of deep neural networks with the objectives of traditional clustering, these methods transcend the limitations of conventional approaches, enabling more accurate, robust, and meaningful data partitions.

This chapter provides a comprehensive exploration of deep clustering, outlining its foundational concepts that draw from both traditional clustering and deep learning. We categorized prevalent approaches, including Deep Embedded Clustering (DEC) and its variants, autoencoder-based strategies, generative model-driven techniques, and advanced end-to-end joint optimization frameworks. The widespread utility of deep clustering was highlighted through its diverse applications across crucial domains such

as image analysis, natural language processing, bioinformatics, and cybersecurity, underscoring its pivotal role in unlocking insights from vast data landscapes.

Crucially, we also addressed the significant challenges that persist within the field, including the complexities of joint optimization, sensitivity to initialization, the pervasive problem of determining the optimal number of clusters, and the inherent difficulties in unsupervised evaluation.

Looking forward, the trajectory of deep clustering research points towards exciting advancements in areas such as enhanced robustness, automated cluster number determination, improved explainability, and greater scalability.

4. Contribution: Adaptive DBSCAN for Heterogeneous Density Clustering

4.1 Introduction

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a widely used density-based clustering algorithm renowned for its ability to identify clusters of arbitrary shapes and effectively handle noisy data [13]. However, the algorithm's performance hinges on the selection of two crucial parameters: Eps (epsilon) and MinPts (minimum points). Eps defines the radius within which to search for neighboring data points, while MinPts specifies the minimum number of points required to form a dense region.

The challenge of determining the optimal Eps value is a well-known issue in DBSCAN. A fixed Eps value can be inadequate for datasets containing clusters with varying densities. If Eps is set too small, low-density clusters might be misclassified as noise, whereas a large Eps value could lead to the merging of distinct high-density clusters.

In this chapter, we address this challenge by proposing a modified DBSCAN algorithm that eliminates the need for manual Eps specification. Our approach leverages the k-nearest neighbors (kNN) algorithm to adaptively calculate an Eps value for each data point based on its local neighborhood density[39]. This adaptive Eps calculation allows our algorithm to effectively identifies clusters with heterogeneous densities, a task that traditional DBSCAN struggles with.

By automating the Eps selection process and tailoring it to the local density characteristics of the data, our modified DBSCAN algorithm enhances the flexibility and accuracy of density-based clustering. This contribution is particularly valuable for real-world datasets, which often exhibit diverse density distributions. Our approach simplifies the parameter tuning process, making DBSCAN more accessible and user-friendly, while also improving its ability to uncover complex cluster structures in data.

4.2 DBSCAN Algorithm

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm, that can identify clusters with arbitrary shapes and different

sizes. Unlike k-means, clusters are not necessarily spheroidal and there is no need to know the number of clusters in advance. Clusters are formed by the identifying points that are density-connected. The algorithm uses two parameters, MinPts and Eps, to define density in data. It can identify three types of points: core points, border points, and noise points [13], which make it more suitable for processing noisy. To assign a point to a cluster, the point must have at least minpts in its radius neighborhood (eps), in this case, we call it core point. Or, the point is within the radius neighborhood of another core point, in this case, we call it border point. Otherwise, it will be considered a noise point. The original DBSCAN algorithm is presented below.

Algorithm 4.1: DBSCAN algorithm

```
Input:
 X     : dataset (observations x1, ..., xn)
 N     : mapping xi ↦ N(xi)   // neighbors of each xi (within Eps)
 MinPts : minimum points

Output:
 Labels : cluster labels for all points in X  // includes NOISE

C := 0
for each observation xi in X do
 if xi is UNCLASSIFIED then
  if |N(xi)| < MinPts then
   mark xi as NOISE
   continue  // go to next xi in X
  end if

  C := C + 1              // start a new cluster
  mark xi as CORE in cluster C
  initialize queue Q and enqueue all points in N(xi)

  while Q is not empty do
   let qi := next UNCLASSIFIED point from Q

   if qi is NOISE or |N(qi)| < MinPts then
    mark qi as BORDER in cluster C
   else
    mark qi as CORE in cluster C
    enqueue all points in N(qi) that are UNCLASSIFIED or NOISE into Q
   end if
  end while
 end if
end for
```

To identify clusters, DBSCAN depends heavily on the parameter eps, which represents the maximum distance to search for neighbors. In addition to being difficult to guess, using a global value for eps can lead DBSCAN to not find the right clusters, especially when data have clusters with different densities. When taking eps with small values, low density clusters may be considered as noise. However, if eps takes a large value, DBSCAN will merge high density clusters

4.3 Related Work

Recent research has been focused on using a single parameter with the widely used clustering algorithm, DBSCAN, to reduce the complexity associated with parameter tuning while maintaining high-quality clustering results. Various papers have proposed different approaches for using a single parameter with DBSCAN, and this approach has gained attention from the research community. In this section, the recent papers that explore these approaches and their potential to achieve high-quality clustering results will be reviewed and discussed.

Bryant and Cios [15] have developed a new clustering algorithm called RNN-DBSCAN, which estimates observation density using reverse nearest neighbor counts and employs a DBSCAN-like approach based on k-nearest neighbor graph traversals through dense observations. This algorithm offers two significant benefits: it reduces problem complexity to a single parameter and enhances the ability to handle large variations in cluster density. However, it may not be effective in distinguishing adjacent clusters with different densities, which could limit its usefulness in some situations.

Hu et al. [16] proposed a new density-based clustering algorithm called KR-DBSCAN. The algorithm is based on the reverse nearest neighbor and influence space. KR-DBSCAN distinguishes adjacent clusters with different densities, reduces computational load, and identifies boundary objects and noise objects. The algorithm defines a new cluster expansion condition using the reverse nearest neighborhood and its influence space and adds core objects to the cluster by breadth-first traversal. However, KR-DBSCAN is more complex than standard DBSCAN due to the use of influence space, which can lead to higher computational cost for large databases.

In addition to these approaches, other papers have proposed novel mechanisms to find the best value of epsilon for a given dataset. For instance, the Multi-verse optimizer

(MVO) algorithm [17] was used to find the Eps interval corresponding to the highest accuracy of DBSCAN. The AE-DBSCAN algorithm [18] uses the first slope, which is greater than the mean plus standard deviation of all non-zero slopes, as Eps. AutoEpsDBSCAN [19] uses the k-dist graph to select several values of the Eps parameter for different densities in the dataset before applying traditional DBSCAN.

Overall, the research reviewed in this section indicates that using a single parameter with DBSCAN can be a promising alternative to the traditional two-parameter approach for DBSCAN. Different approaches have been proposed, each with its strengths and limitations, and the choice of the appropriate approach depends on the characteristics of the dataset and the desired clustering quality and efficiency.

4.4 The Modified DBSCAN

In this section, we explain our proposed algorithm, which can be carried out in two steps. In the first step, the eps values are automatically computed for every point in the data. To do this, we calculate the k-distances of all its k nearest points and then, we calculate the average according to the following formula:

$$eps(i) = 1/k \sum kdist(j) \quad (j \in N(i)) \tag{4.1}$$

where i is the point, whose eps will be calculated, k is the number of points in a neighborhood of i, in DBSCAN algorithm k is called minpts, N(i) is the set of nearest neighbors of the point i, and kdist is the function that calculates the distance between a point j and its k^{th} nearest neighbor.

Algorithm 4.2: computing eps and set of neighbors.

Input:

X : dataset

k : minimum points

Output:

eps : per-point radius ε(xi)

N : neighbors N(xi) for each xi ∈ X

(all points initially marked UNCLASSIFIED)

1) Compute k-distance values for all points in X.

2) For each observation xi ∈ X:

a) Compute eps(xi): the average k-distance of the k nearest neighbors of xi.

b) Find N(xi): the set of neighbors of xi within eps(xi).

c) Mark xi as UNCLASSIFIED.

The second step is to use DBSCAN to find clusters using eps computed in the first step.For a better understanding, the following example illustrates the results obtained by applying formula 1 to a simple mono-dimensional dataset of four points. Fig 1.a shows the circles corresponding to the kth nearest neighbors for each point, here k=3, and Fig 1.b shows circles whose radius are equal to the average k-distance of their k nearest neighbors (formula 4.1).

We can see that radius of the point "x" is extended to be core point, while the radius of the other three points have been reduced to get fewer points in their neighborhoods (less

Algorithm 4.3: Our DBSCAN algorithm

```
Input:
 X     : dataset
 N     : mapping xi ↦ N(xi) (neighbors of each xi)
 MinPts : minimum points
Output:
 Labels : cluster labels for all points in X
C := 0
for each observation xi in X do
 if xi is UNCLASSIFIED then
  if |N(xi)| < MinPts then
   mark xi as NOISE
   continue  // go to next xi in X
  end if
  C := C + 1              // start a new cluster
  mark xi as CORE in cluster C
  initialize queue Q and enqueue all points in N(xi)
   while Q is not empty do
   let qi := next UNCLASSIFIED point from Q
   if qi is NOISE or |N(qi)| < MinPts then
    mark qi as BORDER in cluster C
   else
    mark qi as CORE in cluster C
    enqueue all points in N(qi) that are UNCLASSIFIED or NOISE into Q
   end if
  end while
 end if
end for
```

than k), making the point "w" and the point "y" borders points because they fall within the neighborhood of "x" and make "z" as noise point.

4.5 Experimental Evaluation

In the previous section, we presented a novel version of the DBSCAN algorithm that utilizes only the 'minPts' parameter for clustering. In this section, we aim to provide empirical evidence of the effectiveness of our new algorithm by conducting a comparative analysis with the original DBSCAN algorithm, which uses two parameters.

To assess the effectiveness of our proposed algorithm, we utilized three synthetic 2-D datasets of different shapes and different densities. The first dataset, named 'Compound', comprises 399 points divided into six clusters with varying densities, posing a challenge to accurate clustering. The second dataset, 'asymmetric', is composed of 1000 points divided into five clusters. The third dataset, ‘hard’, includes 1500 points divided into three clusters of different densities.

Figure 4.2 displays the clustering results of the first dataset, where our method successfully identified 6 distinct clusters with only 38 data points labeled as noise. In contrast, the original DBSCAN algorithm detected only 5 clusters and classified the sixth cluster as noise, resulting in 96 noise points.

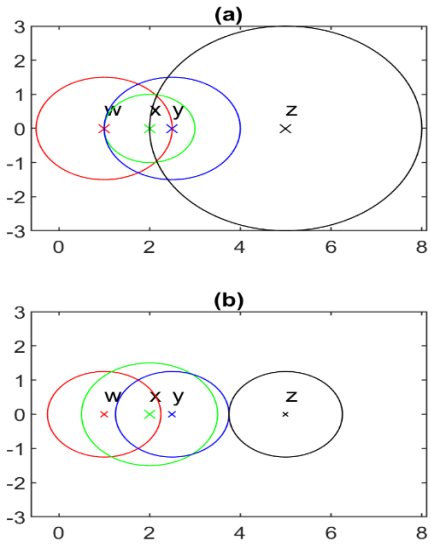
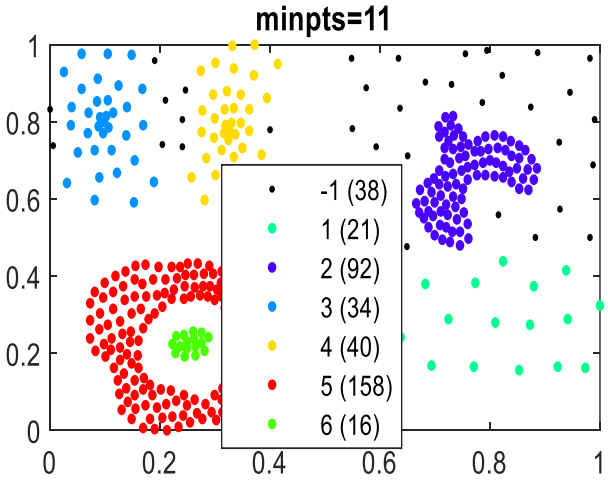


Figure 4.1 Example to illustrate the results obtained by applying formula 1

(a)



(b)

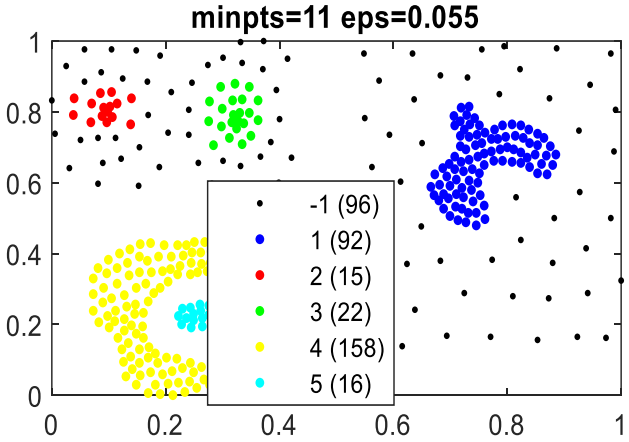


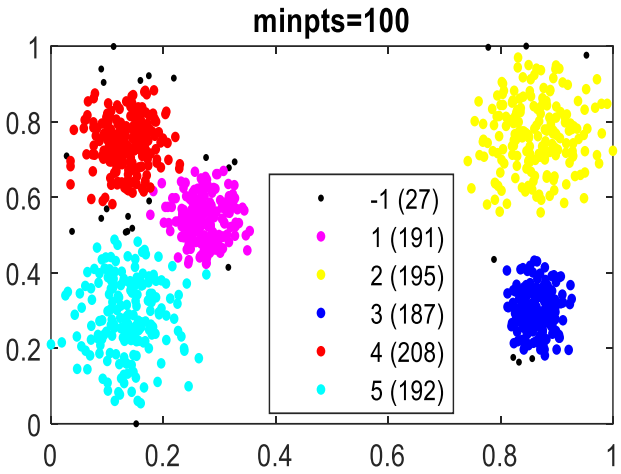
Figure 4.2. Our-DBSCAN (a) and DBSCAN (b) clustering results for the Compound dataset.

.

Figure 4.3 illustrates the clustering results obtained by two algorithms. our algorithm successfully identified five distinct clusters with 27 points labeled as noise. On the other hand, the original DBSCAN algorithm could barely detect five clusters with 225 points marked as noise. If we increase the value of epsilon, it will merge the two groups in the upper left, while decreasing the value of epsilon will classify the group in the upper right as noise.

In reference to the hard database shown in Figure 4.4, it is not possible to determine a suitable value for the parameter eps that would allow us to identify the three clusters using the original DBSCAN algorithm. This is because the algorithm relies on a fixed global value for eps, whereas our modified algorithm was able to successfully detect the three distinct clusters with only 30 points labeled as noise.

(a)



(b)

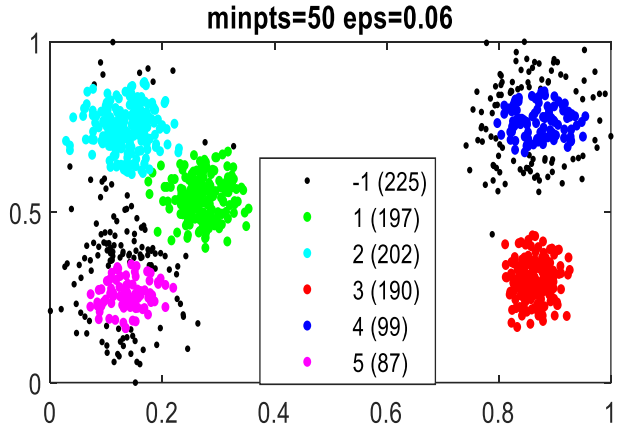
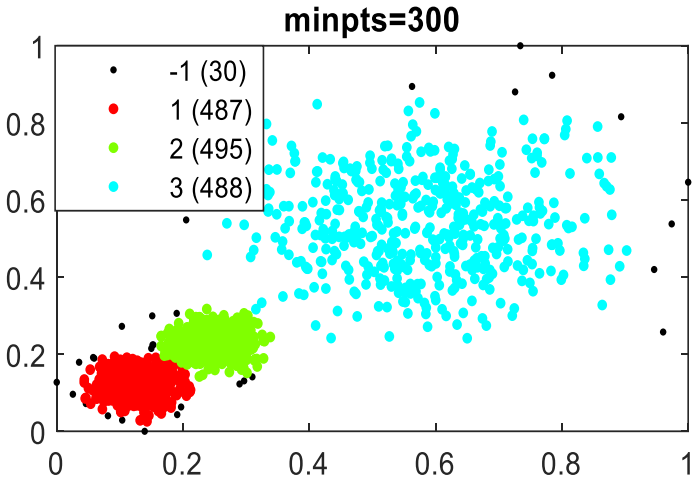


Figure 4.3. Our-DBSCAN (a) and DBSCAN (b) clustering results for the Asymmetric dataset.

(a)



(b)

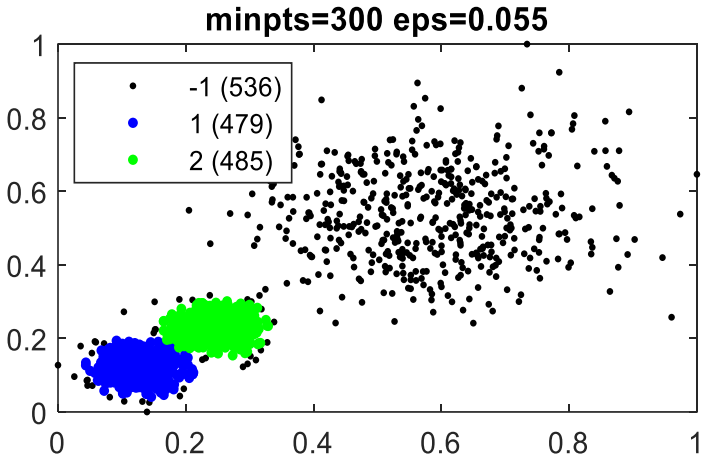


Figure 4.4. Our-DBSCAN (a) and DBSCAN (b) clustering results for the Hard dataset.

To further assess the effectiveness of our approach, we conducted experiments on widely-used datasets such as iris and seed. Since these datasets come with true labels, we were able to measure the accuracy of our method and compare it with that of the original DBSCAN algorithm. Table 4.1 provides a summary of the results we obtained.

Table 4.1: Performance of Our Method on Real-World Datasets

Algorithm	Dataset	Iris	Seed
DBSCAN	Accuracy	0.6667	0.5905
	NMI	0.7612	0.5536
	Cluster / Noise	2/0	3/81
	Minpts/ eps	12/0.025	14/0.19
Our DBSCAN	Accuracy	0.8600	0.8048
	NMI	0.7376	0.6363
	Cluster / noise	3/17	5/15
	Minpts	12	14

As per the data presented in Table 1, it is evident that the proposed method exhibits better performance than the traditional DBSCAN algorithm on the iris and seed datasets.

For the iris dataset, our DBSCAN algorithm achieved an accuracy of 0.86 compared to 0.67 with the traditional DBSCAN algorithm. Additionally, our algorithm was able to detect three clusters with 17 noise points compared to two clusters with no noise points in the traditional algorithm. This indicates that our proposed algorithm is better at detecting the underlying structure in the iris dataset and is able to handle noisy data more effectively.

Similarly, for the seed dataset, our algorithm achieved an accuracy of 0.8048 compared to 0.5905 with the traditional DBSCAN algorithm. Our algorithm was also able to detect five clusters with only 15 noise points compared to three clusters with 81 noise points in the traditional algorithm. This again shows that our proposed algorithm is more robust and accurate in detecting the underlying structure in the data and is more effective in dealing with noisy data.

Furthermore, our algorithm achieved a higher accuracy on both datasets even though the traditional algorithm achieved higher NMI on the iris dataset. This suggests that our proposed algorithm is more effective at finding meaningful clusters rather than just maximizing the agreement with the ground truth.

Overall, the results suggest that the proposed DBSCAN algorithm is superior to the traditional DBSCAN algorithm in terms of accuracy, the number of clusters detected, and handling noisy data.

4.6 Conclusion

In this chapter, we have presented a modified DBSCAN algorithm that addresses the challenge of parameter tuning, particularly the selection of the Eps parameter. By leveraging the k-nearest neighbors algorithm, we have enabled the automatic and adaptive calculation of Eps for each data point, thereby enhancing DBSCAN's ability to detect clusters with heterogeneous densities. Our experimental results on both synthetic and real-world datasets demonstrate the superior performance of our modified algorithm compared to the traditional DBSCAN. The automatic Eps calculation not only simplifies the clustering process but also leads to more accurate and meaningful cluster identification.

This contribution opens up several avenues for future research. One promising direction is to explore the automatic determination of the MinPts parameter, potentially making the algorithm entirely parameter-free. Additionally, investigating the applicability of our adaptive Eps approach to other density-based clustering algorithms could further broaden its impact. By refining and extending this work, we aim to advance the field of density-based clustering and empower practitioners with more robust and user-friendly tools for data analysis.

5.Conclusion and outlook

In this thesis, we have presented a modified DBSCAN algorithm that addresses the challenge of parameter tuning, particularly the selection of the Eps parameter. By leveraging the k-nearest neighbors algorithm, we have enabled the automatic and adaptive calculation of Eps for each data point, thereby enhancing DBSCAN's ability to detect clusters with heterogeneous densities. Our experimental results on both synthetic and real-world datasets demonstrate the superior performance of our modified algorithm compared to the traditional DBSCAN. The automatic Eps calculation not only simplifies the clustering process but also leads to more accurate and meaningful cluster identification.

This contribution opens up several avenues for future research. One promising direction is to explore the automatic determination of the MinPts parameter, potentially making the algorithm entirely parameter-free. Additionally, investigating the applicability of our adaptive Eps approach to other density-based clustering algorithms could further broaden its impact. By refining and extending this work, we aim to advance the field of density-based clustering and empower practitioners with more robust and user-friendly tools for data analysis.

In future work, we would like to explore the following directions:

- **Automatic MinPts Selection:** Investigate methods for automatically determining the optimal MinPts value, potentially making the algorithm completely parameter-free.
- **Extension to Other Algorithms:** Apply the adaptive Eps concept to other density-based clustering algorithms to assess its generalizability and potential benefits.
- **High-Dimensional Data:** Evaluate the performance of the modified DBSCAN algorithm on high-dimensional datasets and explore strategies for dimensionality reduction or feature selection to improve clustering results.
- **Real-Time Applications:** Adapt the algorithm for real-time or streaming data scenarios, where clusters may evolve over time.
- **Scalability:** Develop distributed or parallel implementations of the algorithm to handle large-scale datasets efficiently.
- **Integration with Other Techniques:** Combine the modified DBSCAN algorithm with other data mining or machine learning techniques, such as deep learning or outlier detection, to create more powerful and comprehensive data analysis workflows.

6.References

[1] H. Yu and X. Hou, “Hierarchical clustering in astronomy,” Astronomy and Computing, vol. 41, Art. no. 100662, 2022.

[2] X. Ye and J. W. K. Ho, “Ultrafast clustering of single-cell flow cytometry data using FlowGrid,” BMC Systems Biology, vol. 13, pp. 1–8, 2019.

[3] D. T. Pham and A. A. Afify, “Clustering techniques and their applications in engineering,” Proc. Inst. Mech. Eng., Part C: J. Mech. Eng. Sci., vol. 221, pp. 1445–1459, 2007.

[4] X. Lu, Y. Zhou, L. Qiao, W. Yu, S. Liang, M. Zhao, Y. Zhao, C. Lu, and N. Chi, “Amplitude jitter compensation of PAM-8 VLC system employing time-amplitude two-dimensional re-estimation based on density clustering of machine learning,” Physica Scripta, vol. 94, Art. no. 055506, 2019.

[5] L. Ližbetinová, P. Štarchoň, S. Lorincová, D. Weberová, and P. Průša, “Application of cluster analysis in marketing communications in small and medium-sized enterprises: An empirical study in the Slovak Republic,” Sustainability, vol. 11, Art. no. 2302, 2019.

[6] A. Gramegna and P. Giudici, “Why to buy insurance? An explainable artificial intelligence approach,” Risks, vol. 8, Art. no. 137, 2020.

[7] X. Zhang, D. Wang, and H. Chen, “Improved biogeography-based optimization algorithm and its application to clustering optimization and medical image segmentation,” IEEE Access, vol. 7, pp. 28810–28825, 2019.

[8] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” Pattern Recognition, vol. 36, pp. 451–461, 2003.

[9] T. Kohonen, “Median strings,” Pattern Recognition Letters, vol. 3, pp. 309–313, 1985.

[10] D. Müllner, “Modern hierarchical, agglomerative clustering algorithms,” arXiv:1109.2378, 2011.

[11] D. Defays, “An efficient algorithm for a complete link method,” The Computer Journal, vol. 20, pp. 364–366, 1977.

[12] N. A. Yousri, M. S. Kamel, and M. A. Ismail, “A distance-relatedness dynamic model for clustering high dimensional data of arbitrary shapes and densities,” Pattern Recognition, vol. 42, pp. 1193–1209, 2009.

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD), 1996.

[14] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. S. Sander, “OPTICS: Ordering points to identify the clustering structure,” ACM SIGMOD Record, vol. 28, pp. 49–60, 1999.

[15] A. Bryant and K. Cios, “RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates,” IEEE Trans. Knowl. Data Eng., vol. 30, pp. 1109–1121, 2017.

[16] Hu, L., Liu, H., Zhang, J., and Liu, A., “KR-DBSCAN: A density-based clustering algorithm based on reverse nearest neighbor and influence space,” *Expert Systems with Applications*, vol. 186, Art. no. 115763, Dec. 2021.

[17] A. Baraldi and P. Blonda, “A survey of fuzzy clustering algorithms for pattern recognition. II,” IEEE Trans. Syst., Man, Cybern. B, vol. 29, no. 6, pp. 786–801, 1999.

[18] R. Cordeiro, C. Traina Jr., A. Traina, J. López, U. Kang, and C. Faloutsos, “Clustering very large multi-dimensional datasets with MapReduce,” in Proc. 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), 2011, pp. 690–698.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” J. R. Stat. Soc. B, vol. 39, no. 1, pp. 1–38, 1977.

[20] S. Guha, R. Rastogi, and K. Shim, “ROCK: A robust clustering algorithm for categorical attributes,” Information Systems, vol. 25, no. 5, pp. 345–366, 2000.

[21] J.-G. Lee, J. Han, and K.-Y. Whang, “Trajectory clustering: A partition-and-group framework,” in Proc. ACM SIGMOD Int. Conf. Management of Data (SIGMOD), 2007, pp. 593–604.

[22] T. Liao, “Clustering of time series data—A survey,” Pattern Recognition, vol. 38, no. 11, pp. 1857–1874, 2005.

[23] H. Liu and H. Motoda, Computational Methods of Feature Selection. Boca Raton, FL, USA: Chapman & Hall/CRC, 2007.

[24] L. Liu, R. Jin, C. Aggarwal, and Y. Shen, “Reliable clustering on uncertain graphs,” in Proc. IEEE Int. Conf. Data Mining (ICDM), 2012.

[25] U. von Luxburg, “A tutorial on spectral clustering,” Statistics and Computing, vol. 17, no. 4, pp. 395–416, 2007.

[26] S. C. Madeira and A. L. Oliveira, “Biclustering algorithms for biological data analysis: A survey,” IEEE/ACM Trans. Comput. Biol. Bioinform., vol. 1, no. 1, pp. 24–45, 2004.

[27] Y. Zhao, G. Karypis, and U. Fayyad, “Hierarchical clustering algorithms for document datasets,” Data Mining and Knowledge Discovery, vol. 10, no. 2, pp. 141–168, 2005.

[28] Y. Zhou, H. Cheng, and J. X. Yu, “Graph clustering based on structural/attribute similarities,” Proc. VLDB Endow., vol. 2, no. 1, pp. 718–729, 2009.

[29] Y. Zhu and D. Shasha, “StatStream: Statistical monitoring of thousands of data streams in real time,” in Proc. 28th Int. Conf. Very Large Data Bases (VLDB), 2002, pp. 358–369.

[30] R Core Team, R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing, 2015. [Online]. Available: http://www.r-project.org/

[31] W. Wang, J. Yang, and R. R. Muntz, “STING: A statistical information grid approach to spatial data mining,” in Proc. 23rd Int. Conf. Very Large Data Bases (VLDB), 1997, pp. 186–195.

[32] R. Weber, H.-J. Schek, and S. Blott, “A quantitative analysis and performance study for similarity search methods in high-dimensional spaces,” in Proc. 24th Int. Conf. Very Large Data Bases (VLDB), 1998, pp. 194–205.

[33] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, “SCAN: A structural clustering algorithm for networks,” in Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), 2007, pp. 824–833.

[34] D. R. Karger and M. Ruhl, “Finding nearest neighbors in growth-restricted metrics,” in Proc. 34th ACM Symp. Theory of Computing (STOC), 2002, pp. 741–750.

[35] R. Krauthgamer and J. R. Lee, “Navigating nets: Simple algorithms for proximity search,” in Proc. 15th ACM–SIAM Symp. Discrete Algorithms (SODA), 2004, pp. 798–807.

[36] H.-P. Kriegel, E. Schubert, and A. Zimek, “The (black) art of runtime evaluation: Are we comparing algorithms or implementations?,” Knowledge and Information Systems, 2016.

[37] M. Lichman, “UCI Machine Learning Repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[38] S. Mahran and K. Mahar, “Using grid for accelerating density-based clustering,” in Proc. 8th IEEE Int. Conf. Computer and Information Technology (CIT), 2008.

[39] N. Amroune, M. Benazi, and L. Sayad, “An adaptive Eps parameter of DBSCAN algorithm for identifying clusters with heterogeneous density,” Computación y Sistemas, vol. 28, no. 2, pp. 465–472, 2024.

[40] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a simple greedy layer-wise pretraining,” Journal of Machine Learning Research, vol. 11, pp. 3371–3408, 2010.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, et al., “Generative adversarial nets,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 27, 2014.

[42] J. Xie, R. Girshick, and P. Farhadi, “Unsupervised deep embedding for clustering analysis,” in Proc. Int. Conf. Machine Learning (ICML), 2016.

[43] X. Guo, L. Gao, X. Liu, and J. Yin, “Improved deep embedded clustering with reconstructed loss,” in Proc. Int. Joint Conf. Artificial Intelligence (IJCAI), 2017.

[44] J. Yang, D. Parikh, and D. Batra, “Joint unsupervised learning of deep representations and image clusters,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016.

[45] Y. Tian, C. Chen, and L. Wang, “Rethinking clustering with self-supervision,” in Proc. Int. Conf. Machine Learning (ICML), 2020.

[46] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” Journal of Machine Learning Research, vol. 9, pp. 2579–2605, 2008.

[47] A. Strehl and J. Ghosh, “Cluster ensembles—a knowledge reuse framework for combining multiple partitions,” Journal of Machine Learning Research, vol. 3, pp. 583–617, 2002.