Advanced Clustering

by Jie Yang

Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

under the supervision of
Principal Supervisor: Prof. Chin-Teng Lin
Co-Supervisor: Dr. Yu-Kai Wang

University of Technology Sydney
Faculty of Engineering and Information Technology

May 2022

Certificate of Original Authorship

Required wording for the certificate of original authorship

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Jie Yang*, declare that this thesis is submitted in fulfilment of the requirements for the award of the *Doctoral Degree*, in the *Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.
**If applicable, the above statement must be replaced with the collaborative doctoral degree statement (see below).*

**If applicable, the Indigenous Cultural and Intellectual Property (ICIP) statement must be added (see below).*

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 2/April/2023

Collaborative doctoral research degree statement

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with *[insert collaborative partner institution]*.

Indigenous Cultural and Intellectual Property (ICIP) statement

This thesis includes Indigenous Cultural and Intellectual Property (ICIP) belonging to *[insert relevant language, tribal or nation group(s) or communities]*, custodians or traditional owners. Where I have used ICIP, I have followed the relevant protocols and consulted with appropriate Indigenous people/communities about its inclusion in my thesis. ICIP rights are Indigenous heritage and will always remain with these groups. To use, adapt or reference the ICIP contained in this work, you will need to consult with the relevant Indigenous groups and follow cultural protocols.

Acknowledgement

I would like to express my heartfelt appreciation to my principal supervisor, Professor CT Lin, for his unwavering guidance, encouragement, and support throughout my Ph.D. journey. I am grateful for his generosity in allowing me to pursue research topics that interest me. During my time with Professor CT Lin, I have gained valuable insights and learned a great deal. I would also like to extend my thanks to my co-supervisor, Dr. YK Wang, for his invaluable assistance, guidance, and mentorship throughout my Ph.D. studies.

My deepest gratitude goes to the CIBCI Lab and its members, who have provided me with various resources and selfless support during my research. I would like to extend a special thank you to Xiaofei Wang, Jia Liu, Fred Chang, and Liang Ou for their help and support in overcoming the challenges that I faced.

I also want to thank my friends at TechLab for the fun and memorable times we shared together, and for their continued support in my academic journey. Furthermore, I am grateful to all the friends I have made at UTS, who have provided various forms of assistance and support.

I would like to acknowledge the financial support of the Australian Research Council (ARC) under discovery grants DP180100656 and DP210101093. Additionally, I would like to express my gratitude to the UTS International Research Scholarship for covering my tuition fees.

My heartfelt appreciation goes to my parents, who have been a constant source of love, support, and guidance. I am grateful for their unwavering encouragement and understanding.

Finally, I would like to thank myself for my hard work and persistence, especially during the challenging times caused by the COVID-19 pandemic. Despite the difficult circumstances, I remained committed to my research and was able to complete this project.

Published and Under Review Papers Related to This Thesis

[1] **J. Yang** and C.-T. Lin, “Multi-View Adjacency-Constrained Hierarchical Clustering”, *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. Early Access, pp. 1-13, 2022 **[Chapter 4]**

[2] **J. Yang**, Y.-K. Wang, X. Yao, and C.-T. Lin, “Adaptive Initialization Method for K-Means Algorithm,” *Frontiers in Artificial Intelligence*, vol. 4, 2021 **[Chapters 1-2]**

[3] **J. Yang** and C.-T. Lin, “Multi-View Adjacency-Constrained Nearest Neighbor Clustering (Student Abstract),” *AAAI-2022*, Vol. 36, No. 11, pp. 13097-13098, 2022 **[Chapter 4]**

[4] **J. Yang** and C.-T. Lin, “Autonomous clustering by fast find of mass and distance peaks”, submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Major Revision) **[Chapter 3]**

[5] **J. Yang** and C.-T. Lin, “PSO-based Multi-View Nearest Neighbor Clustering”, submitted to *IEEE Computational Intelligence Magazine* (Under Review) **[Chapter 5]**

[6] **J. Yang** and C.-T. Lin, “Enhanced Adjacency-constrained Hierarchical Clustering using Fine-grained Pseudo labels”, submitted to *IEEE Transactions on Emerging Topics in Computational Intelligence* (Under Review)

[7] **J. Yang** and C.-T. Lin, “Almost Ultrametric Learning using Pseudo Labels from Clustering” (Draft) **[Chapter 6]**

[8] **J. Yang** and C.-T. Lin, “Improve Torque Clustering by optimizing linkage” (Draft)

[9] **J. Yang** and C.-T. Lin, “Distributed Torque Clustering” (Draft)

Abstract

Clustering is a classical technique in the field of data mining. It has played a key role in domains such as biology, medicine, business, and climatology, and is employed in nearly all scientific and social sciences. Despite the significance and pervasiveness of clustering and the plethora of existing algorithms, the current clustering methods suffer from a variety of drawbacks. For example, standard hierarchical clustering has an excessive computational overhead and requires some manually determined conditions. Partition clustering, such as K-means, demands that the number of clusters must either be known or estimated in advance and cannot detect non-convex clusters of varying size or density. Density clustering typically requires a suite of thresholds to be set in advance, such as cut-off distance. Model-based clustering generally relies on prior knowledge of many parameter settings, which is often very difficult to acquire in practice. Classic grid clustering also depends on many user-provided parameters, such as interval values to divide space and density thresholds.

On the other hand, in recent years, multi-view clustering has become a new research hotspot. Essentially, multi-view clustering arises from the combination of clustering problems and multi-view learning. Different from the various conventional single-view clustering methods mentioned above, as an extension of single-view clustering, multi-view clustering is used to handle multi-view data gathered from numerous feature collectors or collected from various sources in various domains. However, most current multi-view clustering approaches suffer from the following three problems: a) parameter tuning, b) significant computational cost, and c) difficulty in finding globally optimal view weights.

To solve the above problems, this thesis first proposes a brand-new efficient parameter-free autonomous clustering algorithm called Torque Clustering (TC). The proposed TC overcomes almost all the shortcomings in previous clustering methods. Furthermore, considering the good performance of the proposed TC, this thesis extends TC to two multi-view clustering algorithms, containing multi-view adjacency-constrained hierarchical clustering (MCHC) and particle swarm optimization (PSO)-based multi-view nearest neighbor clustering (PMNNC). MCHC tries to solve two problems in current multi-view clustering methods: a) parameter tuning and b) significant computational cost. PMNNC focuses on solving the third problem: c) difficulty in finding globally optimal view weights. Finally, we further apply the pseudo labels generated by TC to propose a new metric learning framework, named almost ultrametric learning using pseudo labels of torque clustering (AUMLTC), which can help other algorithms improve performance in a parameter-free and unsupervised manner.

This Ph.D. thesis contains seven chapters. Chapter 1 introduces the background, objectives, scope, organization, and contributions of the thesis. Chapter 2 presents the literature review of the research. Chapter 3 proposes a new parameter-free autonomous clustering, i.e., TC. Chapter 4 exploits the partial mechanism of TC in Chapter 3 as a backbone to propose a new parameter-free multi-view clustering with low computational overhead, i.e., MCHC. Chapter 5 also exploits the partial mechanism of TC in Chapter 3 as a backbone to propose a novel multi-view clustering based on an evolutionary algorithm, i.e., PMNNC. Chapter 6 leverages the pseudo labels of TC in Chapter 3 to propose a new metric learning framework, i.e., AUMLTC. Chapter 7 includes an overview of the thesis's contents and some suggestions for future works.

Keywords: Clustering, Parameter-free, Multi-view Clustering, Autonomous, Metric Learning

Contents

Chapter 1. Introduction

1.1 Background

Clustering is a classical and well-known algorithm in the field of unsupervised data mining. It has been instrumental in domains such as biology, medicine, business, and climatology, and is employed in nearly all scientific and social sciences [1]. For example, in the commercial field, Horng-Jinh Chang et al. introduced a clustering analysis-based model that predicts the buying behavior of prospective customers [2]. In the biology field, clustering is of central importance for the analysis of genetic data, as it is used to identify putative cell types [3]. In addition, clustering can also be applied to image segmentation, object or character recognition [4], [5], data reduction [6], [7], and representation learning [8].

From the perspective of clustering principles, the clustering algorithm mainly includes hierarchical algorithms, partition-based algorithms, density-based algorithms, model-based algorithms, and grid-based algorithms [9]. However, existing clustering algorithms suffer from various inherent shortcomings.

With a typical time cost of $O(n^3)$, standard hierarchical clustering is computationally expensive; hence, large-scale data sets are not suited for these techniques. Furthermore, stopping the clustering process requires manually determined conditions, such as "Stop at K number of clusters". Some of the newer algorithms circumvent these shortcomings by either using a fast approximate nearest neighbor method to accelerate the clustering process [10]–[13] or by pruning the cluster trees to determine the correct cluster number [14], [15].

Partition clustering algorithms like K-means [16] require either prior knowledge or estimation of the number of clusters. Additionally, these algorithms cannot detect clusters that are not convex in shape and have varying sizes or densities, and most of them are highly sensitive to noise, outliers, and getting the initialization phase "right". Improvements over these algorithms include the X-means clustering [17], which is able to predict the approximate number of clusters automatically, and kernel K-means and its variant, spectral clustering, which solve the non-convex problem [18], [19]. Further methods address the initialization problem [20], [21].

In the past, density clustering typically required a suite of thresholds to be set in advance – for example, the cut-off radius distance used to calculate the density of data points, the defined cluster needs to contain at least how many points, and so on. Today's density-based algorithms employ a variety of tricks and techniques to depend less on these thresholds [22]–[30]. For example, Liang et al. [24] proposed the 3DC strategy based clustering that can automatically identify the optimal number of clusters. Du et al. [30] introduced *k* nearest neighbors (KNN) and principal component analysis (PCA) into density peak clustering to predict the number of clusters with greater precision and achieve better results on high-dimensional data sets.

Model-based clustering generally relies on prior knowledge of many parameter settings, such as the distribution of each cluster, even though this information is often very difficult to acquire in practice. Hence, solutions to alleviate this disadvantage have also emerged [31]–[33]. For example, Scrucca, L. et al. [31] proposed an enhanced model-based clustering based on data transformations, which can lead to improved model fitting and more accurate clustering results. O'Hagan, A. et al. [32] proposed the Bayesian initialization averaging method to generate high-quality initial parameter settings for the expectation–maximization algorithm.

Last, classic grid clustering also depends on many user-provided parameters, such as interval values to divide space and density thresholds, and most algorithms are not suitable for high-dimensional data sets. Many variant algorithms have been proposed to solve these problems [34]–[36]. For example, Chen, J. et al. [34] proposed a new grid-based clustering algorithm for mixed data streams that can automatically determine the number of clusters, their centers, and radii.

On the other hand, in the past few years, multi-view clustering has become a new research filed. Essentially, multi-view clustering arises from the combination of clustering problems and multi-view learning [37]. Different from the various conventional single-view clustering methods mentioned above, as an extension of single-view clustering, multi-view clustering can process multi-view data that are gathered from diverse sources in different domains or acquired through various feature extraction methods [38]. For instance, multiple heterogeneous features can be used to characterize an image, such as scale-invariant feature transform (SIFT) descriptors [39], GIST descriptors [40], local binary patterns (LBP) [41], etc. Multiple compatible and complementary features are combined in multi-view clustering algorithms to enhance clustering performance.

Almost all multi-view clustering methods exploit single-view clustering methods as the backbones to learn the complementary representation of multiple views to enhance clustering accuracy. Multi-view clustering, for example, consists mostly of multi-view subspace clustering and multi-view spectral clustering [42], which leverage single-view subspace clustering and single-view spectral clustering as backbones, respectively. Despite the importance of multi-view clustering and the abundance of existing algorithms in previous decades, most contemporary approaches in multi-view clustering generally have the following three issues: a) parameter tuning, b) high computational cost, and c) difficulty in finding globally optimal view weights. For most multi-view clustering algorithms, e.g., multi-view spectral clustering algorithms [43]–[46] and multi-view subspace clustering algorithms [47]–[50], the ultimate performance of the models is significantly influenced by adjusting parameters. For example, Zong et al. proposed a multi-view spectral clustering algorithm based on distinct view weights, which has two parameters that need to be set to assign an optimal weight to each view [46]. Zheng et al. proposed a constrained bilinear factorization multi-view subspace clustering algorithm, which also has two prior information-related parameters to tune to obtain competitive performance [49]. For current multi-view clustering, prior knowledge, such as noise level and label information, is required to guide the specific parameter choice steps, which is troublesome. Furthermore, the computational overhead of most prior multi-view clustering algorithms is also high; multi-view clustering based on subspace learning and spectral representation learning, for example, both have time complexities of $O(n^3)$. Finally, many multi-view clustering algorithms employ gradient-based optimization methods to find optimal view weights. However, one of the limitations of gradient-based methods is the existence of numerous local optima, which can result in solutions where achieving global optimality is challenging [51]. For example, Erlin Pan et al. proposed a multi-view contrastive graph clustering algorithm, which exploits the gradient-based method to learn a consensus graph shared by all views [52]. Salima Ouadfel et al. suggested a weighted multi-view clustering algorithm that utilizes a multi-objective gradient optimizer approach, which identifies an appropriate consensus clustering that takes into account both the disparity between views and the significance of characteristics in each view [53]. Most multi-view clustering algorithms have one or more of the above three shortcomings. These three shortcomings also greatly hinder the application of multi-view clustering in practical scenarios.

1.2 Insights and our solutions

As stated in the background section, from the perspective of clustering principles, a good clustering solution would therefore be able to recognize various types of clusters with different shapes, sizes, or densities; be parameter-free; not depend on a priori knowledge; have a relatively low computational overhead and a reasonable time complexity; be robust to noise and outliers; not require initialization; be able to determine the number of clusters automatically; and preclude the need for a manually specified stopping condition. Therefore, this thesis first proposes a parameter-free autonomous clustering algorithm with all the above advantages: Torque Clustering (TC). Note that, like most previous clustering algorithms, TC is a single-view clustering algorithm.

As an extension of single-view clustering, we have discussed the three main limitations of the majority of current multi-view clustering techniques in the background section: a) parameter choice, b) high computational cost, and c) difficulty in finding globally optimal view weights. Considering that the abovementioned proposed TC has good clustering performance, does not need to be adjusted with parameters, and has a low computational cost, it is therefore natural that we further extend TC to two multi-view clustering algorithms, including multi-view adjacency-constrained hierarchical clustering (MCHC) and particle swarm optimization (PSO)-based multi-view nearest neighbor clustering (PMNNC). The MCHC attempts to solve two problems in current multi-view clustering methods: a) parameter tuning and b) significant computational expense. PMNNC focuses on solving the third problem: c) difficulty in finding globally optimal view weights.

Thus far, this thesis has proposed three clustering algorithms, including TC, MCHC, and PMNNC. The clustering algorithms proposed in this thesis can be used in various scenarios. For example, these three clustering algorithms are exploited to process data sets in various fields during our experimental study, including image recognition, biology, medicine, physics, astronomy, etc. In addition, due to the superior performance of the proposed TC, at the end of this thesis, we further apply the pseudo labels generated by TC to learn a new distance metric to help other algorithms improve performance in a parameter-free and unsupervised manner.

1.3 Research objectives

The following are the thesis's key research objectives.

i. To propose a brand-new parameter-free autonomous clustering algorithm

Despite the importance and ubiquity of clustering and the plethora of existing algorithms, the current clustering methods suffer from a variety of drawbacks. For example, almost all clustering algorithms need predetermined selection of the desired number of clusters. The majority of partition-based clustering approaches cannot identify clusters that are non-convex and have varying sizes or densities. Conventional hierarchical clustering has high computational overhead. To achieve this objective, we propose a novel autonomous clustering algorithm called Torque Clustering (TC). The proposed TC overcomes almost all the shortcomings in previous clustering methods.

ii. To propose a new parameter-free multi-view clustering with low computational overhead

Most currently available multi-view clustering algorithms have issues with computational complexity and parameter tuning. Multi-view clustering that is based on subspace and spectral representation, for example, both have time complexities of $O(n^3)$. In this objective, exploiting partial mechanism of TC in objective 1 as a backbone, we propose a new parameter-free multi-view clustering with little computing expense, named multi-view adjacency-constrained hierarchical clustering (MCHC).

iii. To propose a novel multi-view clustering combined an evolutionary algorithm

Many existing multi-view clustering algorithms employ gradient-based optimization methods to find optimal view weights. However, these methods may become trapped in a local minimum, resulting in poor performance. In contrast, evolutionary optimization algorithms, such as particle swarm optimization (PSO) [236], are more likely to reach the global optimum [54], [55]. In this objective, exploiting partial mechanism of TC in objective 1 as a backbone, a new multi-view clustering has been presented, named PSO-based multi-view nearest neighbor clustering (PMNNC).

iv. To apply pseudo labels of clustering to metric learning

The application range of clustering is very broad. In previous studies, clustering can also be applied to representation learning [8] or metric learning [56]. In this objective, exploiting TC in objective 1 as a backbone, we propose a new metric learning framework, named almost ultrametric learning using pseudo labels of torque clustering (AUMLTC).

1.4 Research scope

Clustering is a very classic machine learning or data mining paradigm. The main goal of this thesis is to propose a relatively general clustering algorithm (i.e., TC) to meet the needs of users or researchers in most cases. In addition, as a more in-depth study, we also leverage TC or its clustering mechanism as the backbone and extend it into two new multi-view clustering algorithms to solve related problems in the field of multi-view clustering. In addition, we also combine TC's pseudo labels and the concept of ultrametric, build a bridge between TC and metric learning, and propose a new metric learning framework. Therefore, this thesis includes the related research and discussion on clustering, multi-view clustering, metric learning based on pseudo labels of clustering, and other topics. In fact, it is not difficult to further extend or improve TC. For example, following extending TC to the field of multi-view clustering, researchers can also consider extending TC to deep clustering, distributed clustering, semi-supervised clustering, and other related fields. Due to space limitations, this thesis doesn't include other extension content.

1.5 Thesis overview

The thesis is organized as follows:

- *Chapter 1:* This chapter presents the background, objectives, scope, organization, and contributions of the thesis.
- *Chapter 2:* The research literature review is presented in this chapter.
- *Chapter 3:* This chapter proposes a new parameter-free autonomous clustering, i.e., TC.
- *Chapter 4:* This chapter exploits the partial mechanism of TC in Chapter 3 as a backbone to propose a new parameter-free multi-view clustering with low computational overhead, i.e., MCHC.
- *Chapter 5:* This chapter also exploits the partial mechanism of TC in Chapter 3 as a backbone to propose a new multi-view clustering based on an evolutionary algorithm, i.e., PMNNC.
- *Chapter 6:* This chapter leverages the pseudo labels of TC in Chapter 3 to propose a new metric learning framework, i.e., AUMLTC.
- *Chapter 7:* The final chapter contains an overview of the thesis's contents and contributions. There are also suggestions for future works.

Fig. 1.1 shows the relationship between chapters. We can see that Chapters 4-6 can be regarded as extensions or applications of Chapter 3. Both Chapter 4 and Chapter 5 inherit the partial clustering mechanism of TC introduced in Chapter 3; Chapter 6 adopts the pseudo labels of TC in Chapter 3 to perform metric learning.

Since this thesis is organized by the compilation of papers (Chapters) and each paper

(Chapter) has a different focus, to ensure that each chapter is self-contained, there may be some repetitions between them, such as introduction, methodology, data sets, etc. In addition, the expression of TC or its mechanism in each paper (Chapter) will be different. For example, in Chapter 4 and Chapter 5, the partial clustering mechanism of TC is reformulated as adjacency-constrained nearest neighbor clustering (CNNC).

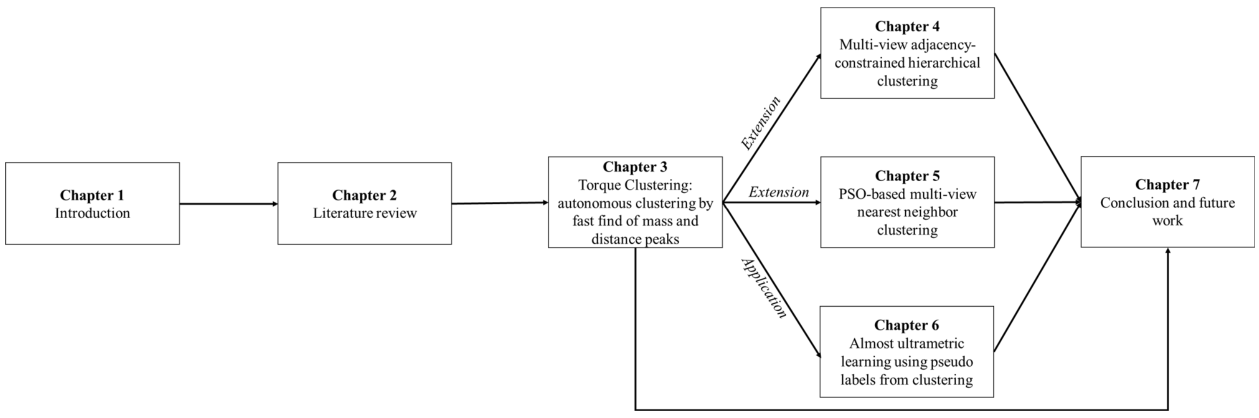

Chapter 1
Introduction
Chapter 2
Literature review
Chapter 3
Torque Clustering: autonomous clustering by fast find of mass and distance peaks
Extension
Extension
Application
Chapter 4
Multi-view adjacency-constrained hierarchical clustering
Chapter 5
PSO-based multi-view nearest neighbor clustering
Chapter 6
Almost ultrametric learning using pseudo labels from clustering
Chapter 7
Conclusion and future work

Figure 1.1. The relationship between Chapters.

1.6 Key contributions

In Chapter 3, we propose a brand-new clustering algorithm (Torque Clustering, TC) derived from the natural idea that a cluster and its nearest cluster with higher mass ought to be merged into one cluster unless they both have relatively large masses and the distance between them is also relatively large. The finding of mass and distance peaks reveals the mergers that do not conform to the rule and should be removed. The TC algorithm is parameter-free and harnesses this idea to recognize any cluster and find the proper number of clusters and noise autonomously. The performance of the proposed TC algorithm was evaluated on 76 synthetic and real-world data sets, demonstrating its remarkable versatility and superiority over the top competing algorithm. Additionally, we also compare it with the latest state-of-the-art deep clustering algorithms on several challenging image data sets. The proposed TC algorithm without any deep representation achieves better or close performance compared to deep clustering algorithms on image clustering.

In Chapter 4, we propose a simple but efficient framework: Multi-view adjacency-Constrained Hierarchical Clustering (MCHC). Specifically, MCHC mainly consists of three parts: the Fusion Distance matrices with Extreme Weights (FDEW); adjacency-Constrained Nearest Neighbor Clustering (CNNC); and the internal evaluation Index based on Rawls' Max-Min criterion (MMI). FDEW aims to learn a fusion distance matrix set, which uses both consensus information among multiple views and the information from each individual view. CNNC is utilized to generate multiple partitions based on FDEW, and MMI is designed for choosing the best one from the multiple partitions. In addition, we propose a parameter-free version of MCHC (MCHC-PF). Without any parameter selection, MCHC-PF can give partitions at different granularity levels with a low time complexity. Comprehensive evaluations on eight real-world data sets indicate that the proposed MCHC (-PF) approach outperforms the 10 most advanced existing methods.

In Chapter 5, we propose a particle swarm optimization (PSO)-based Multi-view Nearest Neighbor Clustering (PMNNC) algorithm. Different from previous spectral (or subspace)-based multi-view clustering, we leverage adjacency-constrained nearest neighbor clustering (CNNC) to enhance the clustering performance on fusion data from multiple views. Furthermore, we propose a new fitness function based on the internal validity index to help learn parameters more accurately. Ultimately, we integrate PSO and CNNC to acquire a fusion distance matrix from multiple views, which enhances the clustering outcomes. Comprehensive evaluations using seven real-world data sets illustrate the advantages of the proposed PMNNC over the top 10 most advanced existing techniques.

In Chapter 6, we first introduce the difference between metric space and ultrametric space. Then, we propose a novel metric called Almost UltraMetric (AUM) and prove that under weak conditions, it will be a true ultrametric. Since the learning of the proposed AUM requires the guidance of ground truth labels, we further propose using pseudo labels of TC to approximate ground truth labels, thus making the learning process completely unsupervised. We call this whole metric learning framework Almost UltraMetric Learning using Torque Clustering's pseudo labels (AUMLTC). It is worth mentioning that, unlike most previous methods, the proposed AUMLTC is unsupervised and parameter-free. The proposed framework's superiority is demonstrated by comparison and ablation experiments conducted on multiple data sets.

Chapter 2. Literature review

2.1 Clustering

Clustering refers to the procedure of dividing a population or a data set into numerous groups based on their similarities, such that the data points within the same group exhibit greater similarities than those in other groups. In other words, the goal is to sort groups with similar characteristics into clusters. The clustering algorithms primarily comprise hierarchical algorithms, partition-based algorithms, density-based algorithms, model-based algorithms, and grid-based algorithms [9].

2.1.1 Hierarchical clustering

Clusters can be created using hierarchical clustering algorithms by iteratively partitioning patterns using one of two strategies: top-down or bottom-up. There are two types of hierarchical clustering: agglomerative and divisive [9]. The bottom-up technique is used in agglomerative clustering to create clusters by combining small individual clusters to form larger and more complex clusters until particular termination conditions are met. The top-down technique is used in divisive hierarchical clustering to obtain clusters by breaking up clusters containing atomic objects into smaller clusters until particular termination requirements are met. Dendrograms are commonly formed using hierarchical approaches, as seen in Fig. 2.1.

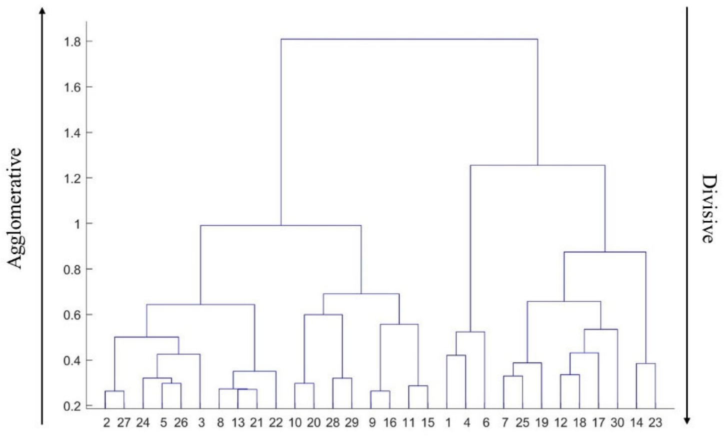


Figure 2.1. Dendrogram of hierarchical clustering [9].

The steps of agglomerative clustering can be condensed as follows [9].

```
1. Make a separate cluster for each point
2. Continue the step 3 until the clustering meets the desired level of satisfaction
3. Fuse the pair of clusters that have the smallest distance between them
4. End
```

The steps of divisive clustering can be condensed as follows [9].

```
1. Make a single big cluster with all of the points
2. Continue the step 3 until the clustering meets the desired level of satisfaction
3. Sever the cluster that results in the formation of two sub-clusters with the highest inter-
   cluster distance
4. End
```

2.1.1.1 Linkage-based hierarchical clustering algorithms

Hierarchical clustering algorithms can be divided into different types based on various measures of inter-cluster distance [9].

1) Single-linkage

The single-linkage method defines the distance between two clusters as the minimum distance between any sample in one cluster and any sample in the other cluster [9] [57]. The following is the distance between clusters A and B:

$$d(A,B) = \min_{a\in A, b\in B} d(a,b) \tag{2.1}$$

2) Complete-linkage

In the complete-linkage method, the distance between two clusters is defined as the maximum distance between any point in one cluster and any point in the other cluster [9] [58]. The following is the distance between clusters A and B:

$$d(A,B) = \max_{a\in A, b\in B} d(a,b) \tag{2.2}$$

3) Average-linkage

The average-linkage method defines the distance between two clusters as the average distance between any sample in one cluster and any sample in the other cluster [9] [59]. The following is the distance between clusters A and B:

$$d(A,B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b) \qquad (2.3)$$

4) Other linkages

In previous studies, various other methods have been proposed to define the distance between clusters in hierarchical clustering. For example, Joe H. Ward, Jr. proposed the ward-linkage algorithm [60]. Ward proposed a general agglomerative hierarchical clustering process in which the optimal value of an objective function is used to determine which pair of clusters to merge at each phase. "Any function that reflects the investigator's purpose" could be this objective function [60]. Michael B. Eisen et al. [61] proposed the centroid-linkage method to analyze genome-wide expression data from DNA microarrays. The distance between two clusters in the centroid method is the distance between the clusters' two mean vectors, and the process combines the two clusters with the least centroid distance at each stage. Wei Zhang et al. proposed the graph degree linkage method for high-dimensional data sets, which investigates the functions of indegree and outdegree, two important concepts in graph theory, in the context of clustering [62].

2.1.1.2 Other hierarchical clustering algorithms

Conventional linkage-based hierarchical clustering has some drawbacks. For example, standard hierarchical clustering requires significant computational resources and has a typical time complexity of $O(n^3)$; hence, large-scale data sets are not suited for these techniques. Furthermore, stopping the clustering process requires some manually determined condition, such as “Stop at *K* number of clusters”. Finally, noise and outliers are problematic for most traditional hierarchical clustering techniques [63]. Some of the newer algorithms circumvent these shortcomings by either using a fast approximate nearest neighbor method to accelerate the clustering process [10]–[13] or by pruning the cluster trees to estimate the proper number of clusters [14], [15]. For example, Manoranjan Dash et al. utilized the "90-10 rule" to decrease the computational consumption of traditional hierarchical clustering significantly [11]. Hisashi Koga et al. utilized the locality-sensitive hashing algorithm, which is a fast search algorithm for the nearest neighbor of each point, to reduce the computational cost of the single-linkage method by quickly identifying nearby clusters to be connected [12]. M. Saquib Sarfraz et al. leveraged the first neighbor relations to make the time complexity of conventional hierarchical clustering $O(nlogn)$ [13]. Kamalika Chaudhuri et al. proposed a robust single-linkage method and a density-based method to predict the correct number of clusters by pruning the cluster trees [15]. In addition, some studies have focused on proposing robust hierarchical clustering algorithms [63], [64]. For example, Maria-Florina Balcan et al. introduced and evaluated a novel and robust algorithm for bottom-up agglomerative clustering procedures, which outperformed traditional agglomerative algorithms [8].

2.1.2 Partition-based clustering

Partition-based clustering generally assigns all samples from data to K clusters by optimizing a specific criterion function [65]. An often-used optimization criterion is the total distance between each data sample and its corresponding cluster centroid. K-means [16], PAM [66], CLARA [66], CLARANS [67], fuzzy c-means [68], and other algorithms [69] have been researched in this category.

2.1.2.1 K-means clustering algorithm

The K-means algorithm [16] is a well-tested, and simple partition-based clustering algorithm that is commonly used to tackle clustering problems. A user-defined number of clusters, K, needs to be set in advance to partition the given data set in this technique. The next key step is to set a centroid for each cluster respectively, with a total of K centroids. The optimization function *J* of K-means is:

$$\boldsymbol{Min}\, J = \sum_{j=1}^{K} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \tag{2.4}$$

where $\left\| x_i^{(j)} - c_j \right\|$ denotes the distance between the cluster centroid c_j and a data point $x_i^{(j)}$. The flow diagram of the K-means algorithm is shown in Figure 2.2.

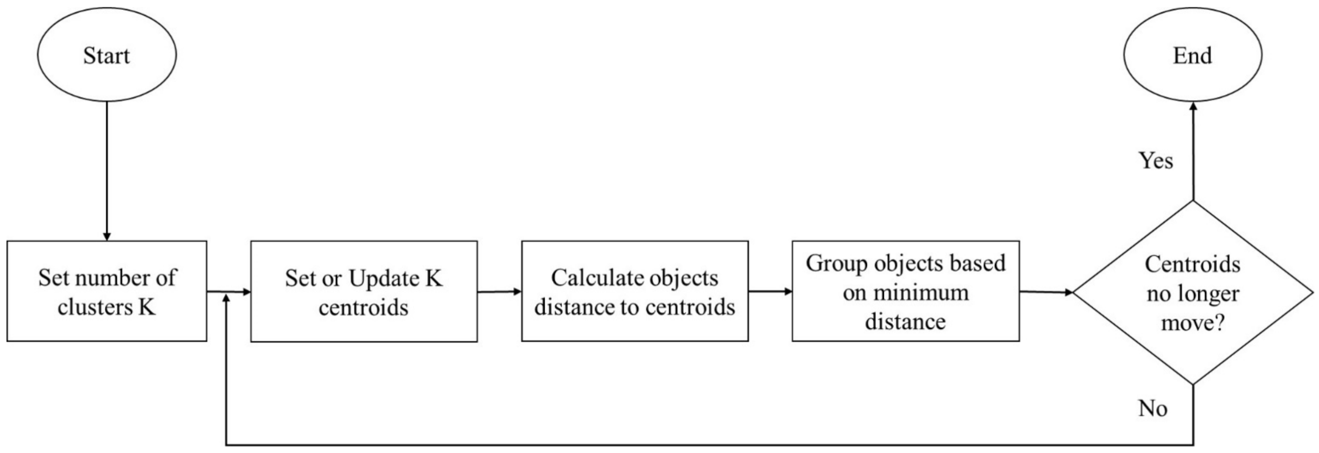

Start
Set number of clusters K
Set or Update K centroids
Calculate objects distance to centroids
Group objects based on minimum distance
Centroids no longer move?
Yes
No
End

Figure 2.2. Flow diagram of the K-means algorithm [9].

2.1.2.2 Other partition-based clustering algorithms

Similar to the K-means algorithm, in general, most partition clustering algorithms demand that the number of clusters must either be known or estimated in advance. Furthermore, these algorithms are incapable of identifying clusters that have irregular shapes and differ in size or density, and most of them are highly sensitive to noise, outliers, and getting the initialization phase "right". Therefore, previous studies have proposed many improved partition-based clustering algorithms. For example, Dan Pelleg et al. proposed the X-means clustering algorithm, which leverages the Bayesian information criterion (BIC) and the Akaike information criterion (AIC) to estimate the optimal number of clusters [17]. Argyris Kalogeratos et al. proposed the dip-means algorithm, which exploits the dip-dist criterion to estimate the correct number of clusters [70]. Grigorios Tzortzis et al. proposed the global kernel K-means clustering algorithm to solve the non-convex problem in the conventional K-means algorithm [71]. Andrew Y. Ng et al. combined the K-means algorithm and graph partitioning to propose the spectral clustering algorithm, which solves the non-convex problem and noise robustness problem [72]. David Arthur et al. presented the K-means++ clustering algorithm to solve the initialization issue in the original K-means algorithm, which is ensured to produce a solution that is competitive with the optimal K-means solution within $O(\log k)$ bounds [20]. Jie Yang et al. proposed the hybrid distance model combining Euclidean distance and density to solve the initialization problem, which makes the K-means algorithm achieve better performance [21].

2.1.3 Density-based clustering

The main idea of density-based clustering is to find the density of a region [73]. Density-based clustering algorithms can discover clusters at various levels of granularity using proper noise reduction. The density concept within these algorithms allows compact sections in the data space to be separated from noise. Clusters are detected in density-based clustering as places with a higher density than the rest of the data space. Density-based clustering aids in the detection of arbitrary form clusters. Many density-based clustering approaches have been presented throughout the previous two decades. These methods are designed to find clusters of densities that are reasonably uniform across the data space. The most representative density-based clustering algorithms include density-based spatial clustering of applications with noise (DBSCAN) [74], density peak clustering (DPC) [75], and their variants.

2.1.3.1 Density peak clustering (DPC) algorithm

In 2014, the DPC algorithm is presented by Rodriguez et al. in Science [75]. DPC is a relatively new density-based technique for clustering. This method is based on the idea that cluster centers are surrounded by neighbors with lower local densities and are separated from any points with higher densities by a greater distance. Each data point *i* is described by two important metrics: its local density ρ_i and its distance from the nearest greater density point δ_i [76].

There are two local density estimators introduced in the DPC algorithm, including the cut-off distance method and the kernel distance method [76]. The local density ρ_i of a data point *i* is calculated using the cut-off distance approach in Eq. (2.5) and the kernel distance method in Eq. (2.6).

$$\rho_i = \sum_{i \neq j} \chi(d_{ij} - d_c),\ \chi(x) = \begin{cases} 1, x < 0 \\ 0, x \geq 0 \end{cases} \tag{2.5}$$

$$\rho_i = \sum_{i \neq j} exp\left[-\left(\frac{d_{ij}}{d_c}\right)^2\right] \quad (2.6)$$

where d_{ij} denotes the distance between two data points, *i* and j. The cut-off distance d_c, is the user-defined neighborhood radius of each data point, where d_c>0. As a result, the number of points with a distance from *i* of less than d_c is positively associated with the local density ρ_i. The most evident distinction between the two techniques is that ρ_i in Eq. (2.5) is a discrete value, and ρ_i in Eq. (2.6) is a continuous value [76].

Following that, DPC defines another property δ_i for each point as in Eq. (2.7).

$$\delta_i = \min_{j:\rho_j > \rho_i}\left(d_{ij}\right) \quad (2.7)$$

As demonstrated in Eq. (2.7), δ_i is the shortest distance between point *i* and another point *j* with a ρ_j greater than ρ_i. Furthermore, the δ_i of point *i* with the largest ρ_i is often defined as Eq. (2.8).

$$\delta_i = \max_{i \neq j}\left(\delta_j\right) \quad (2.8)$$

The points with global or local maxima in terms of ρ_i have the maximum δ_i, as demonstrated in Eqs. (2.7) and (2.8).

In brief, finding the cluster centers (i.e., density peaks) from all the points, and allocating the other points to their associated clusters are the two main steps in the DPC clustering process. For the first step, the tuple (ρ_i, δ_i) for each point *i* is generated in DPC algorithm. The decision graph is then created using these tuples, with the X-axis being ρ_i and the Y-axis being δ_i. Then, as cluster centers, the points with relatively large ρ_i and δ_i values are picked. For the second step, following the selection of cluster centers, the remaining points will be given to the clusters that contain their nearest neighbors with a higher density [76].

Like other clustering methods based on density, DPC also splits data points into three kinds: boundary points, core points, and noise points [76]. The collection of points that belong to a cluster but are separated from points in other clusters by less than the cut-off distance d_c form the border region for each cluster. The highest ρ_i within a cluster's border region is designated as ρ_b. The point i with $\rho_i \geq \rho_b$ in the cluster is regarded as the core point, and all other points are the cluster halo, which is suitable to be noise.

2.1.3.2 Other density-based clustering algorithms

Density clustering typically requires a suite of thresholds to be set in advance, for example, the cut-off distance used to calculate the density of points in DPC and how many points does a defined cluster contain at least in DBSCAN [74]. Today's density-based algorithms employ a variety of tricks and techniques to depend less on these thresholds [22]–[30]. For example, Liang et al. [24] proposed the 3DC strategy based clustering, which can automatically predict the proper number of clusters. Du et al. [30] introduced *k* nearest neighbors (KNN) and principal component analysis (PCA) into density peak clustering to estimate the number of clusters more accurately and achieve better results on high-dimensional data sets. Mariad'Errico et al. applied a non-parametric density estimator PAk to the DPC algorithm to make it fully automatic [77]. Leland McInnes et al. extended DBSCAN by converting it into a hierarchical clustering algorithm called HDBSCAN, which reduces two hyper-parameters in DBSCAN to only one [78]. Zohreh Akbari et al. proposed a parameter-free DBSCAN algorithm based on the statistical technique for outlier detection [79]. Jianghong Zhao et al. proposed the AQ-DBSCAN algorithm, which is a segmentation method for density clustering that involves the use of Gaussian projection [80]. The DBSCAN algorithm can be improved by tackling the challenge of automatically estimating the parameter for neighborhood radius. This approach specifically targets this problem, resulting in a refined and enhanced version of the algorithm.

2.1.4 Model-based clustering

Traditional clustering algorithms such as K-means [16] and hierarchical clustering are heuristic-based algorithms that draw clusters directly from the data rather than including a measure of probability or uncertainty in the cluster allocations. Model-based clustering aims to address this issue by providing a soft assignment in which observations have a chance of belonging to one of the clusters. In general, model-based clustering methods use mathematical models to optimize and determine the appropriateness of given data. Model-based clustering methods, like traditional clustering techniques, uncover informative features for each cluster. In this case, each cluster is indicative of a distinct concept or class [9]. Model-based clustering assumes that the data come from a variety of latent probability distributions. The Gaussian mixture model (GMM) [81] is the most often used method, in which each observation is assumed to be distributed as one of K multivariate-normal distributions, with K denoting the number of clusters (commonly referred to as components in model-based clustering).

However, model-based clustering generally relies on prior knowledge of many parameter settings, such as the distribution of each cluster, even though this information is often very difficult to acquire in practice. Hence, solutions to alleviate this disadvantage have also emerged [31]–[33]. For example, Scrucca, L. et al. [31] proposed an improved model-based clustering using data transformations, which can lead to improved model fitting and more accurate clustering results. O'Hagan, A. et al. [32] proposed the Bayesian initialization averaging method to generate high-quality initial parameter settings for the expectation-maximization algorithm.

2.1.5 Grid-based clustering

Grid-based clustering methods partition the feature space into a grid system, on which all clustering steps are conducted [9]. The approach's key benefit is its quick processing time, the lack of distance computations, and the ease with which clusters are defined as neighbors [9]. Outlined below are the fundamental stages of a grid-based algorithm [9].

1. Create a grid cell set
2. Calculate the density of each grid cell by allocating data samples to the appropriate grid cell
3. Eliminate cells that have a density below a certain threshold.
4. Group dense cells together to form clusters

Wang et al. [82] developed the STING (statistical information grid method) in 1997, which is an algorithm with high scalability, capable of breaking down a data set into multiple levels of complexity. It takes in geographical data and segments it into rectangular cells with varying levels of resolution [9].

However, classic grid clustering also depends on many user-provided parameters, such as interval values to divide space and density thresholds, and most algorithms do not scale to high-dimensional data sets. Many variant algorithms have been proposed to overcome these problems [34]–[36]. For example, Chen, J. et al. [34] proposed a new grid-based clustering algorithm for the mixed data stream, which can automatically discover the properties of clusters.

Table 2.1 summarizes the ideas and disadvantages of the above five types of clustering algorithms. In addition to the above five categories of clustering algorithms, there are also some algorithms based on other ideas or theories. For instance, D.W. van der Merwe et al. proposed a particle swarm optimization (PSO)-based data clustering algorithm [83]. Ehsan Elhamifar et al. combined subspace learning theory and clustering to propose subspace clustering [84]. Moreover, over the past few years, there has been a surge of interest in utilizing deep neural networks to acquire a low-dimensional representation that is conducive to clustering. As a result, there has been a substantial improvement in the performance of clustering algorithms [85]. This type of clustering algorithm is called deep clustering.

Table 2.1. The ideas and disadvantages of the five types of clustering algorithms.

Types	Ideas	Disadvantages
Hierarchical clustering	Combine the two closest sub-clusters; Separate cluster into the two farthest sub-clusters.	High computational cost, require manually determined conditions, sensitive to noise, etc.
Partition-based clustering	Assign all data samples to K clusters by optimizing a specific criterion function.	Require setting number of clusters, cannot detect non-convex clusters, etc.
Density-based clustering	Find the density of regions, and regard those with high density as clusters.	Requires a suite of thresholds to be set in advance, etc.
Model-based clustering	Employ mathematical models to evaluate the suitability of provided data and perform a soft assignment for every sample.	Rely on prior knowledge of many parameter settings, etc.
Grid-based clustering	Divide the feature space into a grid configuration and assign samples to specific cells within the structure.	Depend on many user-provided parameters, etc.

2.2 Multi-view clustering

To improve clustering performance, multi-view clustering is used to learn compatible and complementary information from multi-view data [43]–[47], [86]–[96]. Multi-view data pertains to data that is acquired from disparate sources in distinct domains or obtained from different feature collectors [38]. Essentially, multi-view clustering arises from the combination of clustering problems and multi-view learning [37]. From the perspective of basic clustering principles, there are three kinds of methods that almost constitute a complete family of multi-view clustering: the first type includes the algorithms based on multi-view spectral representation learning, namely multi-view spectral clustering methods; the second type includes the algorithms based on multi-view subspace learning, i.e., multi-view subspace clustering methods; the third type combines various other theories and is often referred to other multi-view clustering methods [42].

2.2.1 Multi-view spectral clustering

By combining information from multiple graphs, multi-view spectral clustering can learn latent cluster structures [97]. Multi-view spectral clustering is based on learning a consensus and complementary graph that contains information from multi-view data and then using the spectral clustering approach on the learned graph to generate clustering results [42]. For example, an illustration of the spectrum perturbation theory of spectral clustering was utilized by Zong and colleagues to develop a weighted multi-view spectral clustering algorithm [46]. The proposed algorithm employs spectral perturbation to imitate the weights of various views. To distinguish the clustering capacity differences of different views, Nie et al. developed an adaptively weighted Procrustes technique, where an indicator matrix can be generated [91]. The proposed method in [92] enhances clustering accuracy by utilizing multiple graphs with distinct weights to integrate complementary data perspectives. The iterative training of a unified graph using mutual reinforcement leads to promising improvements in clustering performance when compared to existing methods [42]. Chang Tang et al. proposed a one-step multi-view spectral clustering method that combines spectral embedding and K-means into a single framework to provide discrete clustering labels in a single step [98]. Shaojun Shi et al. applied the nuclear norm to multi-view spectral clustering to further improve the clustering results, where the nuclear norm makes the view-specific information better explored [99].

2.2.2 Multi-view subspace clustering

Multi-view subspace clustering enhances clustering performance by leveraging multiple data perspectives and learning a uniform subspace representation that captures both shared and unique information across views, making it suitable for complex and high-dimensional data and providing a more complete understanding of the data structure compared to single-view clustering methods [42] [100]. Zheng et al., for example, introduced a multi-view constrained bilinear factorization subspace clustering method that improves clustering results by performing constrained bilinear factorization on the low-rank representation of multiple views [49]. By concatenating multi-view features into a joint representation, Zheng et al. introduced feature concatenation based multi-view subspace clustering to explore the consensus information of multi-view data [50]. The proposed method in [101] for multi-view subspace clustering improves clustering performance by jointly learning a latent representation from multiple views while considering the consensus and complementary information, resulting in effective handling of high-dimensional data and improved accuracy. Guang-Yu Zhang et al. proposed kernelized multi-view subspace clustering based on auto-weighted graph learning, which uses kernel-induced functions to transform multi-view data from linear to nonlinear space [102].

2.2.3 Other multi-view clustering

Furthermore, various other multi-view clustering algorithms have recently been presented [48], [87], [94], [103]. For example, by introducing a collaborative deep matrix decomposition framework, the method proposed in [87] attempts to learn the hidden representations from multi-view data. Xu et al. proposed a deep autoencoder-based method to learn embedded representations, which takes both complementary and consensus information of multiple views into account [94]. The multi-view clustering method proposed in [104] obtains clustering results automatically by considering geometric consistency and cluster assignment consistency.

Multi-view clustering is a relatively new subfield compared to traditional single-view clustering. However, despite the importance of multi-view clustering and the abundance of existing algorithms in previous decades, the most recent approaches in multi-view clustering suffer from the following three problems: 1) parameter choice, 2) high computational cost, and 3) difficulty in finding globally optimal view weights. Most multi-view clustering techniques suffer one or more of the three flaws listed above. These three flaws also make it difficult to use multi-view clustering in real-world applications. Table 2.2 summarizes the ideas and disadvantages of the above three types of multi-view clustering algorithms.

Table 2.2. The ideas and disadvantages of the three types of multi-view clustering algorithms.

Types	Ideas	Disadvantages
Multi-view spectral clustering	Learn a consensus and complementary graph that contains information from multi-view data.	Need to tune hyperparameters, require high computational cost (e.g., $O(n^3)$), have difficulty in finding globally optimal view weights, etc.
Multi-view subspace clustering	Learn a uniform subspace representation from multiple views.	
Other multi-view clustering	Learn embedded representations based on the autoencoder framework; Learn the hidden representations based on the collaborative deep matrix decomposition framework; etc.	

2.3 Distance metric learning

Distance metric learning has gained significant attention in recent years for improving the performance of distance-based methods like KNN [105] and K-means [16], after being first introduced in 2003. The primary objective of metric learning is to reduce intra-class distance while increasing inter-class distance, resulting in each point being closer to other points with the same label and farther away from those with different labels [106]. Distance metric learning can be mainly divided into three categories: supervised metric learning, semi-supervised metric learning, and unsupervised metric learning [106]. The difference between the three lies in the use of ground truth labels.

2.3.1 Supervised metric learning

Supervised metric learning learns distance metrics based on the information from data points and all their labels. For example, one of the most often used supervised linear embedding algorithms is linear discriminant analysis (LDA) [107]. It looks for projection paths that allow data from different classes to be segregated well. Relevant component analysis (RCA) is another well-known supervised distance metric learning approach that makes use of paired data restrictions [108]. RCA is used to find a transformation that amplifies key variability while decreasing irrelevant variability. Average neighborhood margin maximization (ANMM) is a local supervised method that attempts to discover projection paths that maximize local class discriminability [109]. The Mahalanobis metric for clustering (MMC) learns an useful and friendly distance metric for clustering under similarity-relative constraints using semidefinite programming [110]. Furthermore, in recent years, some studies have combined deep learning and metric learning to propose (supervised) deep metric learning models, which learn the higher level of nonlinear characteristics of data directly in the classification structure [111]. For instance, Elad Hoffer et al. proposed the triplet network model for deep metric learning, which aims to learn useful representations by distance comparisons [112].

2.3.2 Semi-supervised metric learning

Semi-supervised methods in machine learning aim to learn distance measures from data sets where labeled data is limited, and most of the data is unlabeled. These algorithms use both labeled and unlabeled data during the learning process. One example is MPCK-Means, proposed in Ref. [113], which combines K-Means clustering and distance metric learning with known pairwise constraints [106]. MPCK-Means partitions the data set into homogeneous clusters using K-means while simultaneously learning a generalized Mahalanobis distance metric with a distinct precision matrix for each cluster [106]. Semi-supervised metric learning using pairwise constraints (SMLPC) is a kernel-based metric learning approach that delivers a nonlinear transformation by considering the topological structure of data as well as both positive and negative constraints [114]. Similarly, based on the strong representation learning ability of deep neural networks, semi-supervised metric learning has also been further extended to semi-supervised deep metric learning [115].

2.3.3 Unsupervised metric learning

Any supervised information is generally not needed for unsupervised distance metric learning methods; instead, they attempt to learn an optimum distance metric or low-dimensional embedding solely from the data matrix or original data, achieving some geometric or discriminative optimality. For example, a popular technique for extracting projection directions from data to achieve the highest variance is known as Principal Component Analysis (PCA) [116]. PCA is commonly employed in high-dimensional data clustering pre-processing. The t-distributed stochastic neighbor embedding (t-SNE) technique is another popular nonlinear dimensionality reduction technique for embedding high-dimensional data in a two- or three-dimensional space for visualization [117]. t-SNE also represents each high-dimensional object as a point in either two or three dimensions, in a way that ensures that similar objects are modelled by nearby points with a high probability, while dissimilar objects are modelled by more distant points. Uniform manifold approximation and projection (UMAP) is another modern method for reducing the dimensionality of data that can be utilized for both visualization purposes and general nonlinear dimension reduction [118]. Recently, some deep metric learning methods have tried to replace ground truth labels with pseudo labels from clustering, making the whole metric learning process unsupervised, which is called unsupervised deep metric learning [119], [120]. For instance, Ujjal Kr Dutta et al. proposed an unsupervised deep metric learning model via orthogonality-based probabilistic loss [120].

Most previous metric learning methods also have some flaws. For example, many methods, such as UMAP, require hyperparameter tuning. In particular, for unsupervised deep metric learning, uninterpretability and the inaccessibility of high-quality pseudo labels are two major obstacles. Table 2.3 summarizes the ideas and disadvantages of the above three types of metric learning algorithms.

Table 2.3. The ideas and disadvantages of the three types of metric learning algorithms.

Types	Ideas	Disadvantages
Supervised metric learning	Learn distance metrics based on the information from data points and all their labels.	Require hyperparameter tuning, inaccessibility of high-quality pseudo labels, uninterpretability, etc.
Semi-supervised metric learning	Learn distance measures from data where supervised information is limited.	
Unsupervised metric learning	Learn an optimum distance metric or low-dimensional embedding solely from the data matrix or original data, achieving some geometric or discriminative optimality.	

2.4 Summary

This chapter first introduces five types of clustering algorithms and several representative algorithms and analyzes their respective shortcomings. Second, this chapter also introduces three common types of multi-view clustering algorithms and summarizes the shortcomings of most current methods. Finally, this chapter also briefly introduces the concept, common classifications of metric learning, and some shortcomings of current methods.

Chapter 3. Torque Clustering: Autonomous clustering by fast find of mass and distance peaks

3.1 Introduction

Grouping similar objects to derive insights from classes of things is a fundamental tool in the search for knowledge. It is used in virtually all the natural and social sciences and plays a central role in biology, astronomy, psychology, medicine, and chemistry [1]. Like many disciplines, grouping objects in data science is called clustering, and, as one of the three broadest categories of machine learning algorithms, clustering in one form or another is the important method of learning from unlabeled data.

Yet, despite the importance and ubiquity of clustering, and the plethora of existing algorithms, the current clustering methods suffer from a variety of drawbacks [1]. Much work has been done to overcome, circumvent, or minimize these problems. Many strategies are targeted, many are ingenious, and several tackles more than one problem. Yet none dispense with enough issues to be considered a universal clustering choice, because optimal clustering is typically an NP-hard problem. Often this means, researchers and analysts must test and tune several alternatives to determine which best suits their needs.

From the literature review in Chapter 2, a good solution would therefore: be able to recognize various types of clusters with different shapes, sizes or densities; be parameter-free; not depend on a priori knowledge; have a relatively low computational overhead and a reasonable time complexity; be robust to noise and outliers; not require initialization; be able to automatically determine the number of clusters; and preclude the need for a manually-specified stopping condition.

To achieve these goals, we propose a novel clustering algorithm, called Torque Clustering (TC), derived from the natural idea that a cluster and its nearest cluster with higher mass ought to be merged into one cluster unless they both have relatively large masses and the distance between them is also relatively large.

The merger process of this algorithm is inspired by the gravitational interactions when galaxies merge. In previous studies, the evolution of galaxies was described as a hierarchical process by astronomers using galaxy merger trees [121]–[124]. Galaxy mergers can occur when two or more galaxies come close enough to each together, and can be classified into two types due to their comparative size of the merging galaxies, including minor mergers and major mergers. According to the predictions of merger rates of dark matter haloes, minor mergers are expected to be much more common than major mergers [125]. TC simulates the process of galaxy minor mergers, so that clusters with larger masses continuously merge adjacent clusters with smaller masses. Similarly, the TC algorithm generates a hierarchical tree that reflects the natural structure of the data set.

After generating the hierarchical tree, TC estimates the correct number of clusters by pruning the cluster tree. However, unlike the existing methods based on probability density [14], [15], TC reveals the reasonable partition by the find of mass and distance peaks. This idea is inspired by the nature of intergalactic distances in the universe [126]. The galaxies usually have very large masses and the intergalactic distances between them are also very large. Regard m_1 and m_2 (i.e., galaxy masses) as the number of samples in two data clusters, and r^2 as the distance between them. TC exploits the two properties, $m_1 m_2$ and r^2, to describe the merger of each pair of clusters. As a result, reasonable cluster (or galaxy) partitions can be obtained by removing the mergers with relatively larger $m_1 m_2$ and r^2.

We evaluated TC on 20 data sets across five different domains: image recognition, biology, medicine, physics, and astronomy, and 19 state-of-the-art algorithms were included in the experiment for performance comparison. Regarding accuracy, TC obtained the top position in 15 out of the 19 data sets (excluding one that lacked ground-truth labels), surpassing the best algorithm compared by an average factor of 4 in ranking. Furthermore, when compared to the best previous automatic clustering algorithm, TC accurately identified the exact number of ground-truth clusters in 15 out of the 20 data sets, whereas the previous algorithm achieved perfect accuracy in only 10 of the data sets. Moreover, we conducted an additional comprehensive evaluation on 56 data sets with noise, outliers, overlaps, imbalance, and high dimensionality. TC still retained a great performance advantage on these data sets.

Finally, we also compared TC with latest state-of-the-art deep clustering algorithms on several challenging image data sets. Interestingly, TC without any deep representation can achieve better or close performance than deep clustering algorithms on image clustering.

3.2 Proposed method

Loosely based on conventional hierarchical clustering structures [127], the TC algorithm generates a hierarchical tree that reflects the natural structure of the data set. However, unlike most existing hierarchy-based algorithms, TC reaches higher accuracy with a significantly smaller number of merger steps and is robust to noise and outliers. In addition, no manual stopping condition is required; the final number of clusters does not need to be defined in advance; the density of each data sample does not need to be estimated, nor does the distribution of each cluster; and the feature space does not need to be divided into distributions. The comparison in mass and distance governs the merger process of TC, while the find of mass and distance peaks reveals the mergers that should be removed to leave a reasonable cluster partition. The TC algorithm is described in sections 3.2.1-3.2.7 below.

3.2.1 Define clusters and form connections between them

Consider a data set denoted as $X = \{x_1, x_2, \dots, x_n\}$, where $x_i \in R^D$. The first step is to determine the initial "mass" of the data set, which is simply the number of samples. Thus, initially, each data sample x_i is considered to be its own cluster ζ_i, which yields an initial cluster set of $\Gamma = \{\zeta_1, \zeta_2, \dots, \zeta_n\}$. This forms the first layer of the hierarchical tree. A count of the set gives us the initial mass, denoted as $\Theta = \{\theta_1, \theta_2, \dots, \theta_n\}$. At this initial step, each $\theta_i = 1$. The following rule is then applied to form connections between clusters:

$$\zeta_i \rightarrow \zeta_i^N, \quad if\ \theta_i \leq \theta_i^N \tag{3.1}$$

where ζ_i^N denotes the 1-nearest cluster to ζ_i, and θ_i^N denotes the number of samples ζ_i^N contains. The symbol " $\rightarrow$ " denotes the connection (i.e., merger) C_i between ζ_i and ζ_i^N. Regarding each cluster as a vertex, then a connected graph *G* can be obtained.

A new set of clusters Γ' can be formed by

$$\Gamma' = \Phi(G) \tag{3.2}$$

where $\Phi(\,)$ identifies the samples contained in each connected component as a new cluster.

Then, applying Eq. (3.1) to Γ' generates new connections C_i, which alters the connected graph *G*. Eq. (3.2) on *G* generates the next cluster set Γ', and the cycle continues until there is only one all-encompassing cluster at the top of a hierarchical tree. Different from the classic agglomerative hierarchical clustering, this constrained method of merging avoids yielding undesirable elongated clusters and can be performed in parallel as long as two neighboring clusters satisfy the requirement of Eq. (3.1). Additionally, this exercise of treating each data sample as a cluster and taking the set through a series of mergers until all have fully merged does three things: 1) It builds a hierarchical map of partitioning clusters at different granularities; 2) it provides a map for the algorithm to choose the most appropriate clustering scheme; and 3) it gives analysts the choice to manually override the automatic selections and choose a different granularity if desired. This leaves the question of what criteria the algorithm uses to determine the "most appropriate" scheme. Based on the above idea, if a connection has both a relatively large mass and stretches over a long distance, it is "abnormal"; removing it should reveal a more reasonable partition structure. The following steps set out the mechanisms for detecting and removing abnormal connections.

3.2.2 Define two properties of each connection to construct the decision graph

Abnormal connections can be identified by observing two intuitive properties of the connection C_i. One of the properties is the product of the mass value of the two clusters it connects

$$M_i = \theta_i \times \theta_i^N \tag{3.3}$$

the other is the square of the distance between the two clusters it connects

$$D_i = d^2(\zeta_i, \zeta_i^N) \tag{3.4}$$

Plotting all the connections on a two-dimensional graph of the two properties, called a decision graph, will reveal that the mass and distance of the abnormal connections are abnormally larger than, and further away from, those of the normal connections. Fig. 3.1 provides an example to illustrate the core idea of the proposed TC algorithm.

There are many studies on defining the distance between two clusters [128]. Here, we simply leverage the minimum distance from any sample in one cluster to any sample in the other cluster as the distance between the two clusters, i.e., $d(\zeta_i, \zeta_i^N) = min\big(d(x_a, x_b)\big), x_a \in \zeta_i, x_b \in \zeta_i^N$. With large-scale data, a fast approximate nearest neighbor method like k-d tree or locality-sensitive hashing may be a more appropriate choice to search nearest cluster since the distance computation approach used in these methods negates the need to actually know the distances between any two clusters [12], [13]. Further, the computation costs would be low and the complexity could be kept to *O(nlog(n))* as compared to the complexity of standard hierarchical clustering algorithms, which is $O(n^3)$. In this way, the proposed TC algorithm is highly scalable. A detailed analysis of the time and space complexity of TC is presented in section 3.2.6.

The detail of how TC works is best explained through an example, which is set out step-by-step in Fig. 3.2 and Table 3.1.

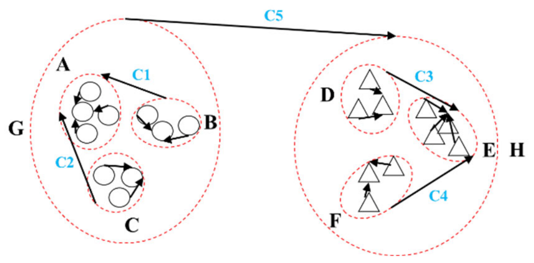


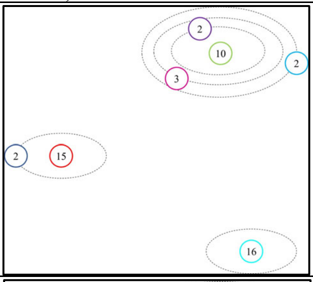
Figure 3.1. The core idea of the proposed algorithm. The red dotted lines delineate the clusters A-H in this two-dimensional data distribution, derived from Eq. (3.1) and Eq. (3.2). The black lines C_1-C_5 indicate the connections from each cluster to its nearest cluster with a length of L_i, where L_5 is the longest. Each cluster is at one end of a connection C_i, and contains several samples. For example, the clusters A and E each have four samples, the clusters B, C, D, and F each have three samples, and the clusters G and H each have 10 samples. Our goal is to find abnormal connections, which are defined as those with both a relatively large distance (i.e., D_i in Eq. (3.4)) and a relatively large number of samples (i.e., M_i in Eq. (3.3)). Obviously, connection C_5, with 10 samples in each of the clusters it connects, is carrying the greatest mass and it is the longest L_5. But what is key is the relativity. C_5 is unique in that it is markedly longer than the other sets of connections. Removing C_5 and calculating the connected components according to Eq. (3.2) results in a final, more reasonable, set of clusters. This approach is consistent with human intuition as well as the natural laws of gravitational interactions.

Figure 3.2. A step-by-step example of how TC works. Consider a data set where, initially, each sample has a mass of 1 and is treated as its own cluster. Applying Eq. (3.1) to

each cluster establishes connections between clusters, resulting in a connected graph. Applying Eq. (3.2) to the graph, new clusters begin to emerge (indicated in different-colored circles) with a mass equal to the number of samples within them (the value in the circles).

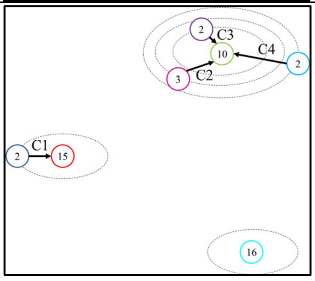
A

Applying Eq. (3.2) to the connected graph of the initial clusters reveals seven new larger clusters.



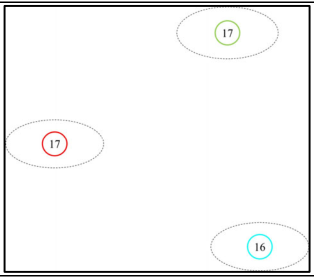
B

Connections C_1 - C_4 can then be added according to the adjacency relationship given by Eq. (3.1). Now, the two properties M_i and D_i of C_1 - C_4 can be calculated according to Eqs. (3.3)- (3.4), as shown in Table 3.1A.



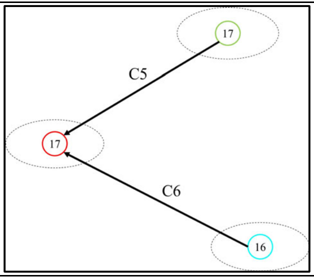
C

Again, applying Eq. (3.2) to the connected graph of Fig. 3.2B reveals three new larger clusters. The mass of each new cluster is equal to the sum of the masses of the sub-clusters it contains.



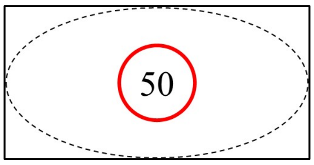
D

Connections C_5 - C_6 can then be added according to the adjacency relationship given by Eq. (3.1). Now, the two properties M_i and D_i of C_5 - C_6 can be calculated Eqs. (3.3)- (3.4), as shown in Table 3.1B.

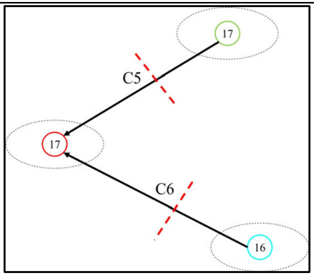


E

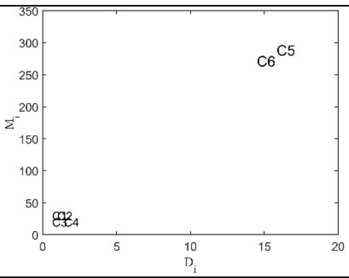
Again, applying Eq. (3.2) to the connected graph of Fig. 3.2D, we now have one big cluster, and the merging process is complete. Steps A-E show that this process has established a hierarchical tree of clustering partitions at different granularities, as illustrated in Fig. 3.2H.



F
Returning to Fig. 3.2D for a moment, it is easy to see from the decision graph in Fig. 3.2G that the relative maxima of the two properties that need to be removed are at C_5, C_6. These connections are identified as abnormal, as indicated by the red dotted lines.



G
Plotting all six connections on a two-dimensional graph of the properties, i.e., the decision graph, indeed shows that C_5, C_6 are abnormally further away and larger than C_1-C_4.



H
Hence, connections C_5, C_6 are removed to arrive at the final partitioning scheme. This entire clustering process can be represented as a hierarchical tree, as the dendrogram to the right shows.

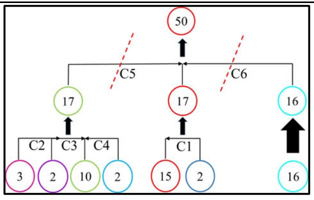


Table 3.1A. Properties of the clusters and connections in Figs. 3.2A and 3.2B.

Cluster	Mass	Nearest Cluster	Mass	Connect?	M_i	D_i	Connection Number
🔴	15	🔵	2	×	–	–	–
🔵	2	🔴	15	✓	30	0.64	C1
🟢	10	🟣	2	×	–	–	–
🟣	3	🟢	10	✓	30	1.00	C2
🟣	2	🟢	10	✓	20	0.64	C3
🔵	2	🟢	10	✓	20	1.44	C4
🔵	16	🔴	15	×	–	–	–

Table 3.1B. Properties of the clusters and connections in Figs. 3.2C and 3.2D.

Cluster	Mass	Nearest Cluster	Mass	Connect?	M_i	D_i	Connection Number
🔴	17	🔵	16	×	–	–	–
🟢	17	🔴	17	✓	289	15.83	C5
🔵	16	🔴	17	✓	272	14.50	C6

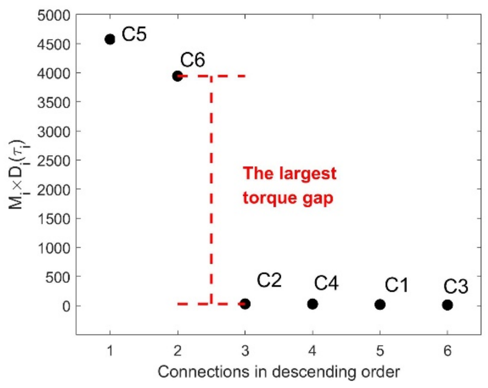


Figure 3.2I. After calculating the torque of each connection generated from Figs. 3.2A-E, all connections are sorted in the order of decreasing torque to obtain the torque sorted connection list (TSCL, i.e., $C_5, C_6, C_2, C_4, C_1, C_3$). Furthermore, Eqs. (3.6)-(3.9) are applied to find the largest torque gap between two adjacent connections in the TSCL (i.e., the torque gap between C_6 and C_2). As a result, C_5 and C_6 are regarded as the abnormal connections to be removed.

3.2.3 Define the torque of each connection and sort the connections in descending order

The decision graph produced by TC provides an efficient visualization tool to determine and remove abnormal connections. However, because a manual inspection of the 2D plot would be both prone to error and time-consuming, we propose an automatic method to determine abnormal connections based on a metric to indicate the gaps between connections. We call this metric the Torque Gap (*TGap*) because of its similarity in mathematical expression. The *TGap* is calculated by first calculating the torque τ_i of all connections, where

$$\tau_i = M_i \times D_i \tag{3.5}$$

Obviously, if the two clusters connected by a connection have relatively large masses and the distance between them is relatively large, the τ_i of the connection must also be large.

Then we sort all connections in descending order according to their corresponding torque values, and call it the torque sorted connections list (TSCL). The connection in TSCL and its torque are denoted as $\grave{C}_i$ and $\grave{\tau}_i$, respectively.

According to our core idea above, abnormal connections must be the top several connections in the TSCL, because they have the largest torque values among all the connections. But how to specify the "several" ones? This requires us to calculate *TGap* on TSCL.

3.2.4 Define torque gap and find the largest gap to determine abnormal connections

The $TGap_i$ between each connection along with its following connection in the TSCL is calculated next. The formula for computing $TGap_i$ is

$$TGap_i = \omega_i \frac{\grave{\tau}_i}{\grave{\tau}_{i+1}},\ \grave{\tau}_{i+1} \neq 0 \tag{3.6}$$

where ω_i is a weighted value that indicates the proportion of connections among the top *i* connections of TSCL that have relatively large M_i, D_i, and τ_i values.

The process for defining ω_i is as follows: Eq. (3.1) will reveal many connections C_i throughout the entire clustering process and, as we know, each C_i has two properties, M_i and D_i. Therefore, the set of connections that have relatively large M_i, D_i, and τ_i values among all the connections (denoted as $Large\_C$) can be defined as:

$$Large\_C = \{C_j | (\tau_j \geq mean\_\tau) \cap (M_j \geq mean\_M) \cap (D_j \geq mean\_D)\} \tag{3.7}$$

where $mean\_\tau$ is the mean value of all τ_i, $mean\_M$ is the mean value of all M_i, and $mean\_D$ is the mean value of all D_i.

Top_C_i is the set of the top *i* connections of TSCL, and can be defined as

$$Top\_C_i = \{\grave{C}_1, \grave{C}_2, \dots, \grave{C}_i\} \tag{3.8}$$

Based on $Large\_C$ and Top_C_i, ω_i is defined as:

$$\omega_i = \frac{|Large\_C \cap Top\_C_i|}{|Large\_C|} \tag{3.9}$$

The largest $TGap_i$ is denoted as $TGap_L$, and the L connections at the top of the TSCL (i.e., $\{\grave{C}_1, \grave{C}_2, \dots, \grave{C}_L\}$) are regarded as abnormal connections to be removed. Fig. 3.2I simulates the above process.

The *TGap* in Eq. (3.6) considers two important factors in determining the abnormal connections at the same time: $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$ is the natural torque gap between adjacent connections in TSCL, and ω_i represents the natural clustering resolution. The larger the torque gap, the more suitable it is as a cutting point. The purpose of calculating $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$ is to find a fracture between the connections with larger torque values and the connections with smaller torque values among the sorted connections. ω_i exists to defend against the uneven distribution or imbalance of clusters in data set. For example, there is a data set containing three relatively balanced ground-truth clusters, {A}, {B}, and {C}. After performing TC on this data set, we get the connection between A and B, and the connection between B and C, denoted as C_{AB} and C_{BC}, respectively. Suppose the distance between A and B is much larger than the distance between B and C, and the distance between B and C is much larger than the distances between sub-clusters in A, B, and C. If we only rely on $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$ to determine the abnormal connections, it is likely to just remove C_{AB} to get the partition: {A}, {B, C}. However, we can consider both $\frac{\grave{\tau}_i}{\grave{\tau}_{i+1}}$ and ω_i at the same time. Since the M_i, D_i, and τ_i of C_{BC}, are also large relative to other connections except C_{AB} (i.e., $\tau_i \geq mean\_\tau, M_i \geq mean\_M, D_i \geq mean\_D$), then TC is more likely to remove both C_{AB} and C_{BC} to get the correct partition, {A}, {B}, and {C}.

3.2.5 Define halo connections to determine the noise

In cluster analysis, noise detection is also an important step. In our algorithm, we define another type of connection to determine the noise, which is called "halo connections" (denoted as *Halo_C*). Halo connections are characterized by a relatively large D_i and a relatively small M_i. The formula for computing halo connections is

$$Halo\_C = \left\{C_k | (M_k \leq mean\_M) \cap (D_k \geq mean\_D) \cap \left(\frac{D_k}{M_k} \geq mean\_\frac{D}{M}\right)\right\} \tag{3.10}$$

where the $mean\_\frac{D}{M}$ is the mean value of all $\frac{D_i}{M_i}$.

In section 3.2.4, after removing the *L* abnormal connections, *L*+1 clusters can be obtained. In this step, the halo connections are further removed, and then some small sub-clusters in the *L*+1 clusters can be found, which are considered as part of the cluster halo (suitable to be considered as noise). Fig. 3.3 provides an example to illustrate the power of abnormal connections and halo connections. The pseudocode of TC is provided in Algorithm 3.1.

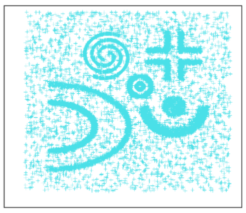
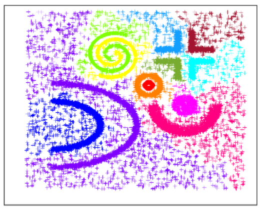
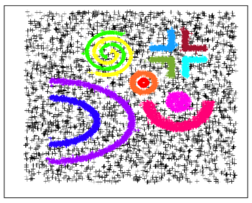
A. Original data distribution	**B.** Removed abnormal connections	**C.** Removed halo connections
		

Figure 3.3. TC on the synthetic data set with 30% uniform noise. **(A)** is the original data distribution, including convex and non-convex clusters with 30% uniform noise; **(B)** illustrates the results of TC after removing the abnormal connections; **(C)** illustrates the results of TC after removing the halo connections, which is also the final partition.

3.2.6 Complexity analysis

- **Time complexity**

According to the pseudocode of TC in Algorithm 3.1, if the input is a distance matrix, the time complexity of Step 1 is $O(n^2)$. Steps 3 and 4 each require $O(n)$. Steps 6, 8 and 9 are in the loop. Suppose the loop needs to be executed m times, where $m \ll n$, the total time cost of steps 6, 8 and 9 is $O(3mn)$, because each of them requires $O(n)$. Step 7 also needs to be executed m times, and its time cost in each loop is $O(l^2)$ due to computing distances

Algorithm 3.1: Algorithm of the proposed TC

```
1: Input: Distance matrix $S \in R^{n \times n}$ or data set $X \in R^{n \times D}$.
2: Output: Cluster partition $\phi = \{\zeta_1, \zeta_2, \dots, \zeta_{L+1}\}$ and cluster halo.
3: Initializing connected graph $G$.
4: Constructing cluster set $\Gamma = \{\zeta_i\}_{i=1}^{l}$ (Initially, regard each sample as a cluster, i.e., $l = n$).
5: while cluster set $\Gamma$ have more than two clusters do
6 : Computing the mass $\theta_i$ of each cluster in $\Gamma$, where $\Theta = \{\theta_i\}_{i=1}^{l}$ .
7: Searching the nearest cluster of $\zeta_i$ according to $S$ or by using a fast approximate nearest neighbor method, e.g. kd-tree.
8: Generating the connections $C_i$ and Updating $G$ by Eq. (3.1).
9: Computing the two properties $M_i$ and $D_i$ of $C_i$ by Eqs. (3.3)-(3.4), and save these to $M_{all}$ and $D_{all}$, respectively.
10: Computing the connected components of $G$ to update the cluster set $\Gamma$ by Eq. (3.2).
11: end
12: Computing the torque $\tau_i$ of each connection based on $M_{all}$ and $D_{all}$ by Eq. (3.5).
13: Sorting all connections in descending order according to their corresponding torque values to get TSCL.
14: Computing $TGap_i$ between each consecutive connection in the TSCL by Eqs. (3.6)-(3.9).
15: Finding the largest $TGap_i$ denoted as $TGap_L$ and treat the $L$ connections at the top of the TSCL as abnormal.
16: Updating $G$ by removing the $L$ abnormal connections, and then compute the connected components of $G$ to obtain the final cluster partition $\phi = \{\zeta_1, \zeta_2, \dots, \zeta_{L+1}\}$.
17: Finding the halo connections by Eq. (3.10).
18: Updating $G$ by removing the halo connections, and then compute the connected components of $G$ to obtain the cluster halo.
```

between neighboring clusters. However, initially, we regard each sample as its own cluster, so the distances between neighboring data samples can be regarded as the distances between neighboring clusters in the first loop without extra computing. Therefore, the total cost of step 7 is $O\big((m-1)l^2\big)$. For step 10, since m loops generate a total of n-1 connections, its time cost is $O(mn+n-1)=O\big((m+1)n-1\big)$. Steps 12, 14, 15 and 17 each require $O(n)$, and step 13 requires $O(nlogn)$. The time cost of step 16 is $O(n+n-1-L)$, approximately equal to $O(2n)$. Similarly, step 18 also requires $O(2n)$. Hence, with a distance matrix as the input, TC's total time cost approximately equal to $O(n^2)+O(nlogn)+O\big((4m+11)n\big)+O\big((m-1)l^2\big)$, its time complexity is approximately $O(n^2)$. However, when using a fast approximate nearest neighbor method, we don't need to compute the distance matrix *S*, so the time cost of step 1 becomes 0 and $O(ml*logl)$ for step 7, giving a total time cost of $O(nlogn)+O\big((4m+11)n\big)+O(ml*logl)$. Therefore, the time complexity here is approximately $O(nlogn)$.

- **Space complexity**

Over the entire algorithm, the following items need to be stored: the cluster set Γ with a mass of Θ, the sparse adjacency matrix for *G*, two properties of each connection M_{all} and D_{all}, the torque τ_i of each connection, and the torque gap $TGap_i$ between each connection in TSCL. Therefore, base space requirement is about $O(7n)$, the space complexity is

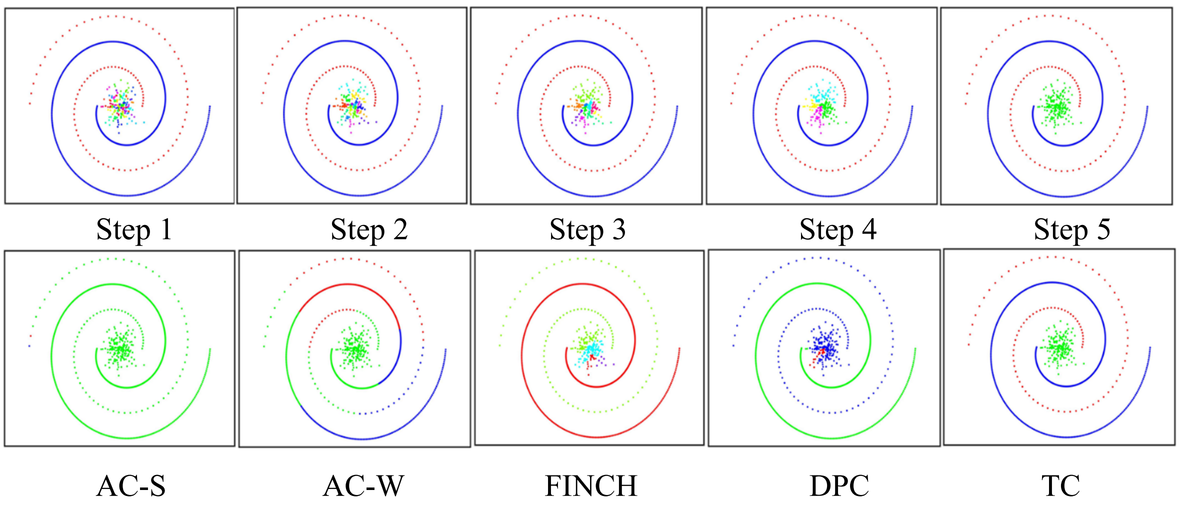


Figure 3.4. Visualization of the step-by-step results of TC (top) and the final results of related methods (bottom) on a synthetic data set.

approximately $O(n)$. This requirement does not change when using a fast approximate nearest neighbor method. However, if the input is a distance matrix, which needs to be stored additionally, the space cost increases to $O(n^2) + O(7n)$, and the space complexity here is approximately $O(n^2)$.

3.2.7 Algorithm analysis

For better clarity, we visualize the step-by-step results of TC and the final results of several related methods on a synthetic data set, as shown in Fig. 3.4. TC follows the rule of Eq. (3.1) to gradually complete the merging of clusters, and finally automatically determines the exact number of clusters. In addition, we can see that in step 1, the red cluster and the blue cluster have been formed, which matches the ground truth. In steps 2-5, due to the constraint in Eq. (3.1), the red and blue clusters do not further merge their 1-nearest clusters but are "waiting" for other sub-clusters to complete the merging, where this process prevents wrong merging in conventional agglomerative methods.

On the other hand, according to Fig. 3.4, agglomerative clustering single-linkage (AC-S) is sensitive to outliers, leading to wrong results. Agglomerative clustering ward-linkage (AC-W) cannot detect clusters with complex shapes. FINCH [13] is an agglomerative method based on nearest neighbor statistics completely without any constraints. Obviously, the result of FINCH contains some wrong mergers. Density peak clustering (DPC) [75] is not robust to the varied density data sets, also leading to wrong results. Besides, all other methods need to manually set the number of clusters (or granularity levels) except for TC.

3.3 Experiments and results

To evaluate the performance of TC, we measured its performance on numerous synthetic and real-world data sets and compared its performance to other 19 well-known or latest algorithms. These algorithms include: K-means++ (K-M++) [20], GMM, Fuzzy clustering (Fuzzy) [68], Spectral clustering (SC) [72], [129], Hierarchical agglomerative clustering single-linkage (AC-S), complete-linkage (AC-C), average-linkage (AC-A), ward-linkage

(AC-W), centroid-linkage (AC-CR) [127]; Density peak clustering (DPC) [75] and its three latest variants, Dynamic graph-based label propagation for density peak clustering (DPCLP) [27], Shared-nearest-neighbor-based density peak clustering (SNNDPC) [76], and Automatic density peak clustering (DPA) [77]; Efficient parameter-free clustering using first neighbor relations (FINCH) [13], DBSCAN (DB) [74], Mean-shift (MS) [130], Affinity Propagation (AP) [131], Border-Peeling clustering (BP) [132], and Robust continuous clustering (RCC) [1]. Among them, DPA, FINCH, DB, MS, AP, BP, and RCC can automatically determine the number of clusters.

In each experiment, the optimal values for the free parameters of all the compared methods were selected based on their best performance across a wide range of possible settings or runs. This approach provided a significant advantage to the compared methods. Implementation details on each of these baselines are provided in the section 3.6. Contrarily, the reported performance of TC is from just a single run.

All experiments were evaluated in terms of the two commonly-used external indices: normalized mutual information (NMI) [133] and accuracy (ACC). Additionally, we also compared TC with other automatic clustering algorithms for their ability to determine the optimal number of clusters. We counted the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters, denoted as NGC.

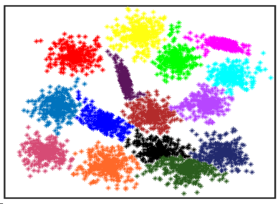
3.3.1 Evaluation on nine synthetic data sets

Fig. 3.5 presents the results of tests with nine different synthetic data sets reflecting seven challenges commonly faced in clustering. These data sets have been widely used as benchmark comparisons for many clustering algorithms [134]. Table 3.2 provides the descriptive statistics of these data sets.

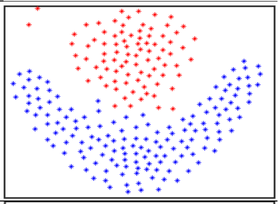
As the tests in Figs. 3.5A-3.5I show, the TC algorithm conquered every trial. In addition, TC automatically found the exact number of clusters for all nine data sets, matching the ground-truth numbers perfectly. As a means of visual comparison, we also conducted these same tests with K-means [16]. As shown in Fig. S3.1 in the section 3.6, K-means failed on eight of the nine data sets, the exception being the first. Additionally, see Table 3.3 for the full quantitative comparison with the 19 state-of-the-art clustering algorithms on these synthetic data sets.

Figure 3.5. Results with seven different clustering challenges. As the results show, the proposed TC algorithm recognized all the clusters regardless of their shape, size, or density.

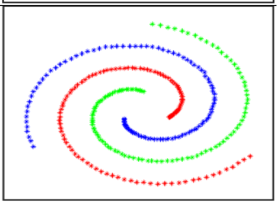
A.[135] **Highly overlapping:** TC was easily able to recognize the 15 clusters in this data set with substantial overlaps.



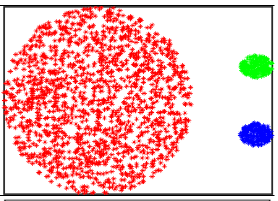
B.[136] **FLAME:** TC was able to find the two clusters in this case designed to test fuzzy clustering by local approximation of membership (FLAME).



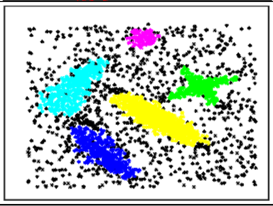
C.[137] **Spectral-path:** This data set was used to illustrate the performance of a path-based spectral clustering algorithm. TC was perfectly able to identify the three clusters without the need to generate a path-based connectivity graph.



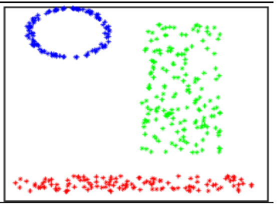
D.[138] **Unbalanced:** Severe imbalances in the data did not present a problem to TC as the hugely disproportionate clusters to the right show.



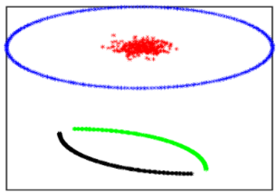
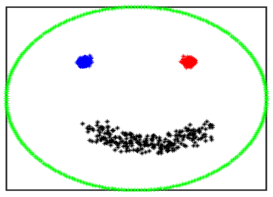
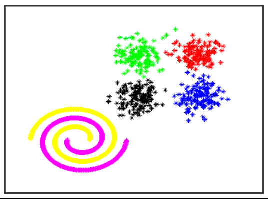
E.[75] **Noisy:** This data was originally used to showcase how a density peak clustering handles noise. TC was able to detect the five clusters with lots of noise.



F.[139] **Heterogeneous geometric:** TC intuitively found the three clusters without the need to calculate point symmetry distances, as was required in.



Multi-objective: Figs. 3.5G-3.5I show examples of multi-objective clustering. With these types of tasks, more than one type of clustering algorithm is needed to reveal all the different types of cluster structures in the data [140]. The

current standard is to use ensemble learning to optimize multiple objective functions. TC was able to identify the different structures naturally.	
G.[141]	**H.** [142]
	
	I. [143]
	

3.3.2 Evaluation on 11 real-world data sets

In this section, we evaluated TC on a further 11 real-world clustering tasks (data sets) across five different domains: image recognition, biology, medicine, physics, and astronomy. Full descriptive statistics for the 11 data sets used are provided in Table 3.2. Here, we report the details of the tasks and TC's individual performance with each of them. Furthermore, Table 3.3 gives the full quantitative comparison with the 19 state-of-the-art clustering algorithms on these real-world data sets.

The image recognition experiments comprised handwritten digit recognition, face recognition, object recognition, and Pose, Illumination, Expression (PIE) recognition as four independent tasks. We used the popular benchmark data set MNIST [144] for the digit recognition task, where it was preprocessed using the method in [13]. Each digit from 0 to 9 should be clustered together to form a total of 10 clusters. TC correctly classified the digits with an accuracy of 99.22%.

The face recognition task was performed on the Olivetti Face Database (OFD) [145]. The results are shown in Fig. S3.2 color-coded by cluster. For ease of reporting, we have only included the first 100 images, denoted as OFD-F100. Using the similarity measure outlined in Ref. [146], TC completed the task with 92% accuracy.

The object recognition was conducted on the extremely high-dimensional data set

Table 3.2. Statistics of the 20 data sets. The Noisy data set has no ground truth labels.

Data sets	Instances	Dimensions	Clusters	Imbalance
Highly overlapping	5000	2	15	~1
FLAME	240	2	2	~2
Spectral-path	312	2	3	~1
Unbalanced	2000	2	3	8
Noisy	4000	2	5	-
Heterogeneous geometric	400	2	3	~2
Multi-objective 1	1000	2	4	1
Multi-objective 2	1000	2	4	1
Multi-objective 3	1500	2	6	~4
OFD-F100	100	10304	10	1
MNIST	10000	4096	10	~1
COIL-100	7200	49152	100	1
Shuttle	58000	9	7	4558
RNA-seq	801	20531	5	~4
Haberman	306	3	2	~3
Zoo	101	16	7	~10
Atom	800	3	2	1
Soybean	47	35	4	~2
Cell-track	40	40	2	1
CMU-PIE	2856	1024	68	1

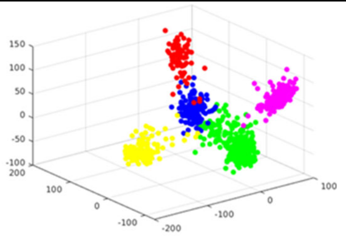


Figure 3.6. Projection of the five clusters (tumors) of the RNA-seq data set found by TC in a three-dimensional subspace.

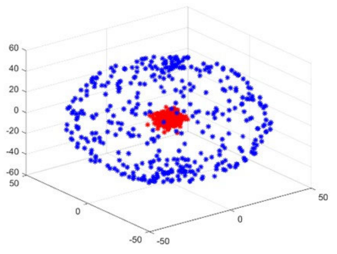


Figure 3.7. The result for the Atom data set. TC correctly identified the two clusters: the atom's kernel and hull.

COIL-100 (the Columbia University Image Library) [147] comprising 72 viewpoints on 100 objects, making a total of 7,200 samples and 49,152 pixels. TC completed the task with an NMI of 97.2%.

The PIE recognition was conducted on the CMU Pose, Illumination, and Expression (CMU-PIE) data set [148], which contains 13 different poses, under 43 different illumination conditions, and with 4 different expressions. TC completed the task with a perfect accuracy of 100%.

The biology experiments comprised tasks in gene expression analysis, cell tracking analysis, and animal recognition. The gene expression task [3] was conducted with the RNA-seq data set [149], which is a random extraction of gene expressions in patients with five different types of tumors. TC correctly recognized all five with 99.88% accuracy. A more intuitive representation appears in Fig. 3.6 with a plot of the 20,531-dimensional feature space distilled to 3D space using PCA [116].

For the cell tracking task, we used the cancer cell tracking (Cell-track) data set from GitHub [150]. The goal is to use cell movements to determine whether they are in the RGDS or FSL layer. TC correctly tracked the cells to each of the two layers with 87.5% accuracy.

The last task, animal recognition, with the Zoo data set from the UCI Machine Learning Repository [151] was completed by TC with an accuracy of 92.08%.

The two tasks in the medicine domain were soybean disease diagnosis, and diagnosis of survival time in breast cancer patients.

The Soybean data set [152] contains observations of four diseases present in soybean plants. TC identified the diseases with a perfect accuracy of 100%.

Last in this domain was the survival time for breast cancer patients task using Haberman's Survival (Haberman) data set [153]. Survival times in this data are categorized into two groups – less or more than 5 years – based on three surgical features: the age of the patient at the time of their operation, the year of the operation, and the number of positive axillary nodes detected. TC identified three clusters with comparatively high accuracy, which is very close to the ground-truth of two clusters. Closer inspection of the additional cluster revealed samples with a reasonable survival time prediction in between the two bipartite clusters.

In the physics domain, we performed one task with the Atom data set [154], which contains 3D data similar to an atom kernel and hull. The data set contains two clusters in R^3 with a completely overlapping convex hull and has 800 samples in total [155]. This task is to discriminate between the kernel and hull of the atom. As shown in Fig. 3.7, this was another task TC performed with perfect accuracy.

Table 3.3A. Performance comparison of all algorithms on all data sets, measured by NMI. The Noisy data set does not contain ground-truth labels, we removed it in the comparison. “NA” means not applicable.

Data sets/Methods	K-M++	GMM	Fuzzy	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	MS	AP	BP	RCC	TC
Highly overlapping	.9652	.9713	.9537	.0128	.9596	.9354	.0346	.8867	.9455	**.9747**	.8497	.9572	.9705	.8665	.2830	.9474	.7542	.8460	.7887	.9568
FLAME	.4843	.4477	.4422	.0479	.4832	.3297	.0479	.0770	.0479	.4132	.7937	.8288	.5805	.4896	.8374	.8673	.4408	.9083	.6492	**1.0000**
Spectral-path	.0012	.0678	.0003	1.0000	.0031	.0068	1.0000	.0106	.0119	1.0000	.3037	1.0000	.3903	.5359	1.0000	.3999	.5546	.2056	.5940	**1.0000**
Unbalanced	.4453	1.0000	.4429	1.0000	.6108	.6109	1.0000	.4406	.6351	1.0000	.6228	1.0000	.6526	.3720	1.0000	.6962	.3841	.5134	.3310	**1.0000**
Heterogeneous geometric	.8089	1.0000	.8016	1.0000	1.0000	1.0000	1.0000	.4611	1.0000	.7445	1.0000	1.0000	1.0000	.5583	1.0000	.8143	.5855	.8017	.5648	**1.0000**
Multi-objective 1	.8357	.9696	.6008	1.0000	.6895	.5981	.8633	.7008	.6636	.8044	.9673	1.0000	.8766	.6995	.9977	.7071	.6180	.8750	1.0000	**1.0000**
Multi-objective 2	.6807	.9448	.6072	1.0000	.6894	.6130	1.0000	.6920	.6775	.6663	1.0000	1.0000	.8660	.6846	1.0000	.7968	.6517	.7999	.8781	**1.0000**
Multi-objective 3	.7065	.8272	.5319	.7881	.7020	.7229	.8341	.7052	.7204	**.9950**	.6868	.7304	.7030	.7947	.9709	.8253	.5988	.9092	.5956	.9925
OFD-F100	.9057	.8225	.4923	.8218	.7278	.7927	.5846	.6063	.6744	.8666	.5195	.9136	.8538	.8746	.5195	.8166	.8379	.0000	.0000	**.9362**
MNIST	.9741	.8396	.3338	.9761	.9751	.9714	.0143	.9741	.9754	.9751	.0000	.9765	.8555	.9755	.9686	.8543	.7364	.7888	.7774	**.9767**
COIL-100	.8281	NA	.2866	.8564	.7256	.8353	.6990	.7428	.6577	.8657	.0760	.6919	.8504	.7897	.7223	.0270	.8388	.5510	.9628	**.9720**
Shuttle	.3247	.3888	.2415	.5860	.0277	.2671	.0385	.0516	.0285	.5769	NA	NA	.3700	.0368	.5113	.6060	NA	.3273	.4858	**.6389**
RNA-seq	.9808	.8400	.5711	.9948	.0464	.9860	.0318	.7252	.0320	.8348	.0000	.9895	.9745	.8785	.5712	.5716	.6633	.8960	.9287	**.9948**
Haberman	.0785	.0706	.0000	.0057	.0006	.0204	.0387	.0006	.0083	.0114	.0118	.0000	.0344	.0295	.0448	.0736	.0653	.0249	.0438	**.0847**
Zoo	NA	.8587	NA	.8438	.8728	.8640	.5025	**.9023**	.8916	.4937	.3530	.8082	.5257	.8725	.8605	NA	.7928	.5643	.8518	.8988
Atom	.3060	.9681	.3082	.0158	.2257	.2257	1.0000	.2107	.0620	.2735	1.0000	1.0000	.8236	.5560	1.0000	.6249	.4345	.6382	.4492	**1.0000**
Soybean	NA	.9441	NA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	.8025	.0000	.7928	1.0000	.7539	.8489	NA	.7914	.6179	1.0000	**1.0000**
Cell-track	.4835	**.5617**	.3793	.3351	.0620	.5466	.0620	.0620	.0620	.3464	.0631	.3988	.3744	.3986	.3793	.3474	.3817	.1281	.3857	.4592
CMU-PIE	.5550	.4284	.1297	.8123	.5060	.5982	.9897	.4920	.3859	.9532	.3857	.3930	.9152	.7977	.7589	.0205	.8365	.2099	.3411	**1.0000**
Rank	10.5	7.9	16.2	7.3	11.7	10.3	9.5	12.9	12.4	8.7	12.7	6.7	7.7	10.5	7.1	11.1	12.9	12.1	10.4	**1.6**

Table 3.3B. Performance comparison of all algorithms on all data sets, measured by ACC.

Data sets/Methods	K-M++	GMM	Fuzzy	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC	DPCLP	SNNDPC	DPA	FINCH	DB	MS	AP	BP	RCC	TC
Highly overlapping	.9808	.9842	.9561	.0773	.9754	.9570	.0738	.8514	.9138	**.9858**	.8548	.9740	.9842	.7468	.1334	.9682	.3124	.7318	.4734	.9714
FLAME	.8583	.8417	.8500	.6458	.8333	.7208	.6458	.5167	.6458	.7875	.9625	.9708	.5375	.2292	.9375	.9792	.1375	.9833	.6167	**1.0000**
Spectral-path	.3526	.4295	.3401	1.0000	.3590	.3750	1.0000	.3878	.4038	1.0000	.5769	1.0000	.4744	.1571	1.0000	.3462	.1571	.5032	.2692	**1.0000**
Unbalanced	.6150	1.0000	.4732	1.0000	.5435	.5440	1.0000	.5320	.6795	1.0000	.7215	1.0000	.5160	.1015	1.0000	.7445	.0670	.3510	.1150	**1.0000**
Heterogeneous geometric	.9325	1.0000	.9325	1.0000	1.0000	1.0000	1.0000	.6075	1.0000	.8500	1.0000	1.0000	1.0000	.1350	1.0000	.8225	.1675	.7700	.1700	**1.0000**
Multi-objective 1	.8530	.9900	.6055	1.0000	.5890	.6580	.7490	.6180	.5760	.6560	.9890	1.0000	.7450	.4280	.9990	.5880	.1320	.8620	1.0000	**1.0000**
Multi-objective 2	.7910	.9820	.6860	1.0000	.8070	.7140	1.0000	.8090	.7830	.7320	1.0000	1.0000	.7500	.5490	1.0000	.7750	.2280	.7100	.8660	**1.0000**
Multi-objective 3	.7280	.7607	.5179	.5072	.7153	.7840	.7520	.7093	.7833	**.9987**	.6547	.7213	.5860	.6133	.9833	.6907	.1160	.9193	.1460	.9980
OFD-F100	.9100	.8200	.2990	.7593	.5900	.7600	.3300	.5200	.4400	.7800	.4100	.8200	.7700	.7700	.3700	.5900	.7000	.1000	.1000	**.9200**
MNIST	.9915	.7842	.2086	.9921	.9917	.9903	.1141	.9912	.9919	.9916	.1135	.9922	.6925	.9918	.9838	.8918	.2662	.6625	.5497	**.9922**
COIL-100	.5992	NA	.0233	.6368	.2794	.6053	.3504	.3521	.2175	.5482	.0187	.3089	.5385	.3922	.4313	.0107	.3529	.2532	.8307	**.8633**
Shuttle	.5456	.5116	.3000	.7581	.7834	.7663	.7862	.6386	.7837	.8893	NA	NA	.2912	.5788	.8232	.8855	NA	.0669	.6619	**.9063**
RNA-seq	.9950	.8939	.5546	.9988	.3645	.9963	.3758	.7403	.3758	.7990	.3745	.9975	.9850	.8240	.6067	.6242	.2784	.9288	.9001	**.9988**
Haberman	**.7582**	.6667	.5098	.5229	.7320	.7288	.7386	.7320	.7353	.5425	.7353	.6863	.6209	.4771	.7386	.7418	.1242	.7386	.3301	.7549
Zoo	NA	.8812	NA	.7228	.8614	.8812	.5248	.9109	.8911	.4752	.4752	.8515	.4950	.8713	.8317	NA	.6733	.5941	.8020	**.9208**
Atom	.7212	.9962	.7362	.5025	.6575	.6575	1.0000	.6450	.5250	.6963	1.0000	1.0000	.8163	.5675	1.0000	.6300	.0975	.6250	.2863	**1.0000**
Soybean	NA	.9787	NA	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	.8085	.3617	.7021	1.0000	.5106	.7872	NA	.5106	.5745	1.0000	**1.0000**
Cell-track	.8750	**.9000**	.8250	.8000	.5250	.9000	.5250	.5250	.5250	.8250	.6250	.8500	.7750	.7750	.8250	.5250	.6750	.5750	.6000	.8750
CMU-PIE	.2479	.2255	.0516	.6524	.1961	.2272	.9496	.1719	.1457	.7892	.1089	.1499	.7539	.3592	.6933	.0259	.3631	.0525	.0882	**1.0000**
Rank	8.9	7.7	14.8	7.4	10.2	8.0	8.5	11.5	10.4	7.8	11.3	6.1	9.7	13.6	6.6	12.6	16.6	12.6	13.1	**1.5**

To evaluate the performance of our TC in the astronomy domain, we tested the NASA Shuttle data set, which comprises 58,000 instances generated by seven unique conditions in the radiator subsystem [1]. TC completed this task with an accuracy of 90.63%.

3.3.3 Results and analysis

3.3.3.1 Performance advantage

According to the full quantitative comparison (see Table 3.3), TC indisputably outperformed all other state-of-the-art algorithms, given its best-in-show performance on 15 of the data sets.

In terms of the rank in Table 3.3, the next-best algorithm, SNNDPC, was about four times higher than TC in its ranking scores both on NMI and ACC, even then, needs to set the ground truth number of clusters in advance.

Moreover, beyond simple ranking metrics, there are some important quality-of-life issues to note:

- Most algorithms that outperformed TC on several data sets are sensitive to initialization/parameters, so the reported accuracies are the highest of many runs with different initializations or with different parameters. TC, however, is parameter-free and does not need any initialization, so the accuracy levels reported are from just a single run.
- Most algorithms require the analyst to specify the ground truth number of clusters before running the clustering procedure, whereas TC can automatically determine the number of clusters.

3.3.3.2 Automatic determination of number of clusters

For a completely unsupervised clustering algorithm, it is important to be able to automatically determine the number of clusters. The TC algorithm, applied to the 20 data sets above, returned the exact or close to the exact number of clusters across the board without human intervention (15 exact, 5 close to). Table 3.4 highlights TC's performance against the seven comparators that can also automatically determine the number of clusters. TC was 50% more accurate than the next-best algorithm, DB, which only identified the correct number of clusters on 10 data sets even parameters tuning to get these results required many runs. The others ranged from 0 to 6 sets.

Table 3.4. The performance comparison of eight automatic methods on predicting the ground-truth number of clusters. #C means the ground-truth number of clusters. NGC indicates the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters.

Data sets/Methods	#C	DPA	FINCH	DB	MS	AP	BP	RCC	TC
Highly overlapping	**15**	**15**	56	2	**15**	116	29	10	**15**
FLAME	**2**	6	19	**2**	**2**	21	**2**	5	**2**
Spectral-path	**3**	5	56	**3**	6	35	**3**	33	**3**
Unbalanced	**3**	4	128	**3**	**3**	80	12	744	**3**
Noisy	**5**	4	29	**5**	**5**	144	54	176	**5**
Heterogeneous geometric	**3**	**3**	36	**3**	**3**	24	7	33	**3**
Multi-objective 1	**4**	6	26	**4**	2	40	6	**4**	**4**
Multi-objective 2	**4**	2	63	**4**	**4**	37	8	14	**4**
Multi-objective 3	**6**	10	7	7	5	75	**6**	104	**6**
OFD-F100	**10**	11	8	3	13	17	1	1	12
MNIST	**10**	42	**10**	**10**	15	80	20	149	**10**
COIL-100	**100**	129	44	64	11	568	32	90	129
Shuttle	**7**	336	16	8	56	-	204	144	4
RNA-seq	**5**	6	4	3	63	36	**5**	4	**5**
Haberman	**2**	5	3	**2**	5	19	1	6	3
Zoo	**7**	12	8	3	-	11	3	14	6
Atom	**2**	**2**	29	**2**	13	55	9	80	**2**
Soybean	**4**	**4**	12	3	-	9	2	**4**	**4**
Cell-track	**2**	3	4	1	10	9	1	11	**2**
CMU-PIE	**68**	61	271	67	4	231	3	6	**68**
NGC	-	4	1	10	6	0	4	2	**15**

It is worth pointing out, however, that in many fields, choosing the right number of clusters can be subjective, depending more on the user's requirements than a ground-truth. This is why many clustering algorithms, including the well-known clustering method [75], DPC, all adopt similar kinds of decision graphs to visualize the cluster structure of the data set. Visualization helps users to estimate the ideal number of clusters, and offering a choice helps users better meet their own needs. However, because there is no objective and agreed definition of a cluster, people use different, subjective determinations of where the borders between clusters are [156]. The research by Balcan et al. on the problem of discovering ground-truth clustering revealed that using a list of partitions or a hierarchy instead of a single flat partition is preferable [157]. Decision graphs do not identify a specific number of clusters, but they do provide a customized way for users to choose for themselves, which complies with the above views.

The decision graphs of the first 100 images of the Olivetti Face Database (see Fig. S3.4A) and the nine data sets in Figs. 3.5A-I (see Fig. S3.3) clearly show the cluster structure of each data set as compared to the decision graph for the DPC algorithm (see Fig. S3.4B). Further, the ease of estimating the ideal number of clusters with TC is clear. But these decision graphs provide another benefit. On the Olivetti Face Database data set, the TC algorithm identified 11 abnormal connections and removed them to leave 12. Yet, when looking at the decision graph, the actual number of abnormal connections is obviously nine. Removing these leaves 10 clusters, which is exactly the ground-truth number. The final recognition accuracy therefore moves from 92% with TC up to 95%. Thus, the decision graph is also helpful for manually correcting the TC algorithm in situations of excessive sensitivity.

Further, in the cases where the ground-truth number of clusters K is known, then K-1 connections with the largest τ_i in Eq. (3.5) can simply be regarded as abnormal connections.

Table 3.5. Runtime comparison of TC with the state-of-art clustering algorithms on COIL-100 data set. The results are reported in HH:MM:SS.

K-M++	Fuzzy	GMM	SC	AC-A	AC-W	AC-S	AC-C	AC-CR	DPC
0:00:28	00:12:16	NA	00:03:41	00:02:02	00:02:27	00:01:53	00:01:57	00:02:07	00:02:11
DPCLP	SNNDPC	DPA	FINCH	DB	MS	AP	BP	RCC	TC
00:07:25	00:15:08	00:02:28	00:00:48	00:04:03	00:07:09	00:07:59	11:47:26	06:16:51	00:00:31

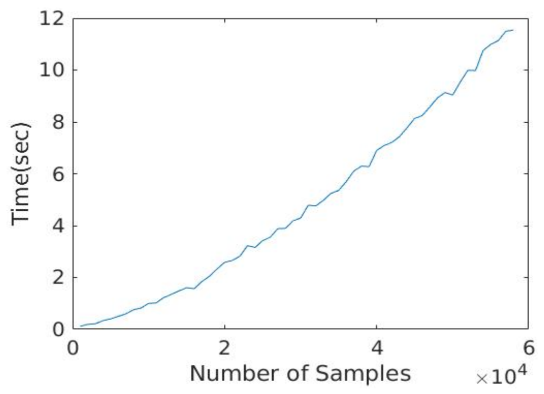


Figure 3.8. Runtime of TC. We assess the runtime of TC by considering the data set size as a variable, which is accomplished by randomly selecting subsets of varying sizes from the Shuttle data set for analysis.

For example, with the COIL-100 data set, removing the 99 connections with the largest τ_i generates 100 clusters and 89.51% accuracy – higher than TC's at 86.33%. What this demonstrates is that if the ground-truth number of clusters is known in advance, TC's performance further improves.

3.3.4 Runtime

To reflect the effectiveness of TC more intuitively, we compared the running time of TC with that of all 19 algorithms. In general, three attributes of the data set, i.e., the number of samples, dimensions, and clusters, all affect the execution time. Therefore, we chose COIL-100 as the test data set because all three attributes are reflected in relatively large numbers. All algorithms were implemented in Matlab or Python. All of the tests were run on a workstation with two 14-core Intel Xeon 6132 CPUs running at 2.6 GHz and 3.7 GHz, as well as 96GB of RAM. The average execution time is reported for the clustering algorithms that need to be executed multiple times. The code provided by the author for GMM fails to function correctly when applied to this data set. The results in Table 3.5 show the running time of TC is less than that of all other algorithms except K-means++.

Additionally, Fig. 3.8 illustrates the time taken by TC to run on subsets of the Shuttle data set that were randomly selected. The proposed TC can produce clustering results in roughly 12 seconds for all subsets.

3.3.5 Further evaluation on 56 data sets with peculiar characteristics

In this section, we conducted an additional comprehensive evaluation on 56 data sets in total that includes 27 data sets with noise, outliers, overlaps, or other peculiar distributions, nine synthetic data sets with unbalanced clusters, five synthetic data sets with uniform noise and 15 poorly-separated, high-dimensional gene expression data sets. The results are reported in Tables S3.4-S3.7 and Figs S3.5-S3.7. Overall, TC still retained a great performance advantage on these data sets.

3.3.6 Comparison to deep clustering algorithms on challenging image data sets

Image data sets usually have very high dimensions, and traditional clustering algorithms often cannot achieve good results on them. Numerous recent investigations have centered on utilizing deep neural networks to train a low-dimensional representation that is conducive to clustering. This kind of approach, often called deep clustering, has led to a substantial enhancement in clustering performance, particularly for image data sets [85]. Hence, in this section, we also compared TC with the latest state-of-the-art deep clustering algorithms on several challenging image data sets, including UMIST [158], FRGC-v2.0 [159], COIL-20 [160], COIL-40 [147], Pendigits [161], and the two mentioned above, COIL-100 [147] and CMU-PIE [148]. We run TC directly on the raw features (or pixels) of these image data sets without any other representation. Further, we combined the leaderboards of the "papers with code", a website that ranks the performance of open-source algorithms, and the latest papers on deep clustering algorithms, to list the results of the top six deep clustering algorithms that perform best on these seven image data sets (measured by NMI), as shown in Table 3.6. On CMU-PIE, COIL-40, and UMIST data sets, TC outperforms all state-of-the-art deep clustering algorithms. Besides, the results of TC are also competitive on other data sets.

In summary, TC without any deep representation can achieve better or close performance on challenging image data sets, compared with state-of-the-art deep clustering

Table 3.6. Comparison to Deep Clustering algorithms, measured by NMI.

Data sets	CMU-PIE	COIL-100	COIL-40	COIL-20	FRGC-v2.0	UMIST	Pendigits
Rank 1	1 JULE [162] 2016	**.985** JULE [162] 2016	.967 A-DSSC [163] 2020	**1** JULE [162] 2016	**.651** DNB [164] 2021	.917 DSC-FEDL [165] 2020	**.868** EAEDC [166] 2021
Rank 2	1 DDSNnet [167] 2021	.946 A-DSSC [163] 2020	.963 J-DSSC [163] 2020	.981 DSC-FEDL [165] 2020	.610 DEPICT [168] 2017	.893 S^2DSCAG [169] 2020	.863 N2D [170] 2019
Rank 3	.970 DAutoED [171] 2021	.943 J-DSSC [163] 2020	.951 DSC-FEDL [165] 2020	.979 SADSC [172] 2021	.580 MI-ADM [173] 2021	.890 DSC-DAG [169] 2020	.820 DnC-SC[174] 2021
Rank 4	.965 MI-ADM [173] 2021	.910 DGMM [175] 2021	.928 RGRL [176] 2020	.974 S^2DSCAG [169] 2020	.574 JULE [162] 2016	.881 RGRL [176] 2020	.817 DipDECK [177] 2021
Rank 5	.964 DEPICT [168] 2017	.905 DBC [178] 2018	.920 DASC [179] 2018	.958 DSC-DAG [169] 2020	.544 DPSC [180] 2021	.877 JULE [162] 2016	.814 GCML [181] 2022
Rank 6	.925 DPSC [180] 2021	.886 DDSNnet[167] 2021	.916 DSC-DAG [169] 2020	.910 DGMM [175] 2021	.522 DDSNnet [167] 2021	.851 DNB [164] 2021	.801 AESC[182] 2020
TC	**1**	.972	**.989**	.960	.586	**.931**	.849

algorithms. Deep clustering algorithms also face some challenges. For example, they have several hyper-parameters that are non-trivial to set, lack interpretability, and have high computational complexity.

3.4 Discussion

3.4.1 Differences between TC and other hierarchical clustering algorithms

Although TC appears to be a hierarchy-based clustering algorithm, it is different from the existing algorithms in several major ways.

First, most of the previous hierarchical clustering algorithms are completely based on the nearest neighbors' statistics without constraints. However, we introduce a simple but very effective constraint (i.e., the requirement in Eq. (3.1)) in TC, which prevents wrong merging usefully (see Fig. 3.4). This idea is inspired by the gravitational interactions of galaxy minor mergers. Second, in each step of TC, if any two neighboring clusters satisfy the requirement of Eq. (3.1), a connection can be formed, and thus mergers can be performed in parallel. That means a large cluster can form within very few steps (see Table S3.2), greatly improving clustering efficiency and reducing the algorithm's execution time (Table 3.5 and Fig. 3.8). However, standard hierarchical clustering algorithms need to perform merging at least n-K times to get K clusters. Third, TC algorithm automatically determines the number of clusters by removing abnormal connections based on a new *TGap* metric. Instead, most of the existing hierarchical algorithms still need to manually set the number of clusters (or granularity levels), even if dendrograms are provided. Finally, TC is robust to noise and outliers and can identify noise clusters. However, many of the classic agglomerative clustering algorithms are not robust to noise [63]. As the case study in Fig. 3.4 and the empirical results in Table 3.3, TC outperforms other hierarchy-based clustering algorithms.

3.4.2 Differences between TC and density peak clustering algorithms

Even though the decision graph of TC is similar to that of DPC techniques, it is also different from the existing algorithms in several major ways.

First, decision objects are different. TC determines which connections between neighboring clusters are abnormal connections, and then prunes the clustering tree by removing them to get the results. The DPC is to decide which data samples are cluster centers, and then complete the label assignments of the remaining samples according to them. Excluding the distance between clusters (or between samples), as the basis for determining abnormal connections, TC only needs to count the number of samples (i.e., mass) contained in each cluster to obtain M_i in Eq. (3.3), without any hyper-parameters. While DPC uses some density estimators to estimate the local density of each sample to determine the cluster centers, where these density estimators usually contain hyper-parameters such as cutoff distance. Second, label assignment strategies are different. TC leverages the constrained method of merging in Eq. (3.1) to assign labels to samples in clusters, which can effectively improve accuracy. DPC, on the other hand, assigns labels to samples based on selected cluster centers (i.e., density peaks). However, once the cluster centers are wrongly chosen, then there may be many more samples subsequently misassigned. For example, in the varied density data set of Fig. 3.4, DPC erroneously selects two cluster centers (i.e., the blue and red ones) in a ground truth cluster, which eventually leads to wrong label assignment. Finally, robustness is different. DPC is not robust to the varied density data sets, since it assumes that density peaks must be cluster centers. While TC has parallelism in the merging process, that is, if any two neighboring clusters satisfy the requirement of Eq. (3.1), a connection can be formed, which can effectively reduce the sensitivity to varied density in conventional hierarchical clustering (see Fig. 3.4).

3.4.3 Differences between TC and subspace clustering algorithms

Clustering algorithms attempt to classify elements into categories, or clusters, on the basis of their similarity metric [75]. Currently, there are some well-known methods that combine clustering and subspace learning (or metric learning), such as subspace clustering (SSC) [84], which achieve good results on high-dimensional data sets. However, like most classical clustering algorithms, such as K-means, linkage methods, the proposed TC tries to find possible clusters based on the commonly used similarity metric (e.g., Euclidean, cosine), without learning a new subspace or new similarity metric.

3.4.4 Potential limitations of TC

On the one hand, the merging process of TC relies on nearest-neighbor statistics and leverages a method like single-linkage to measure the distance between clusters. Therefore, the performance of TC on some high-dimensional and sparse data sets is not particularly satisfactory (see Table S3.7), even if still outperforming related methods. On the other hand, since TC relies on the global mean variable values (i.e., mean_M, mean_D, and mean_$\frac{D}{M}$) to detect cluster halo, it is more suitable to be used to identify uniform noise, and may not be able to accurately identify all non-uniform noise. We will address these issues in future work.

3.5 Conclusion

Whether compared to the classic or very recent methods, TC demonstrates itself to be a highly accurate algorithm, superior in performance to all its counterparts and with an unprecedented level of versatility. Overall, we have presented a clustering algorithm that is: parameter-free, can recognize various types of clusters with different shapes, sizes or densities; does not depend on a priori knowledge; is robust to noise and outliers; does not need any initialization; automatically determines the number of clusters, and does not demand a manually-specified stopping condition. Moreover, the ability to use any desired method of 1-nearest-cluster computation means TC is scalable to large data sets with a relatively low computational overhead and a reasonable time complexity, especially when choosing an approximate nearest neighbor search method, such as k-d tree or locality-sensitive hashing [12], [13]. In this chapter, we presented many test cases to showcase TC's versatility. Even more experimental details and comparisons of clustering quality versus state-of-the-art methods are provided in Tables S3.1-S3.7 and Figs. S3.5-S3.9 in the section 3.6.

3.6 Experimental details and more results

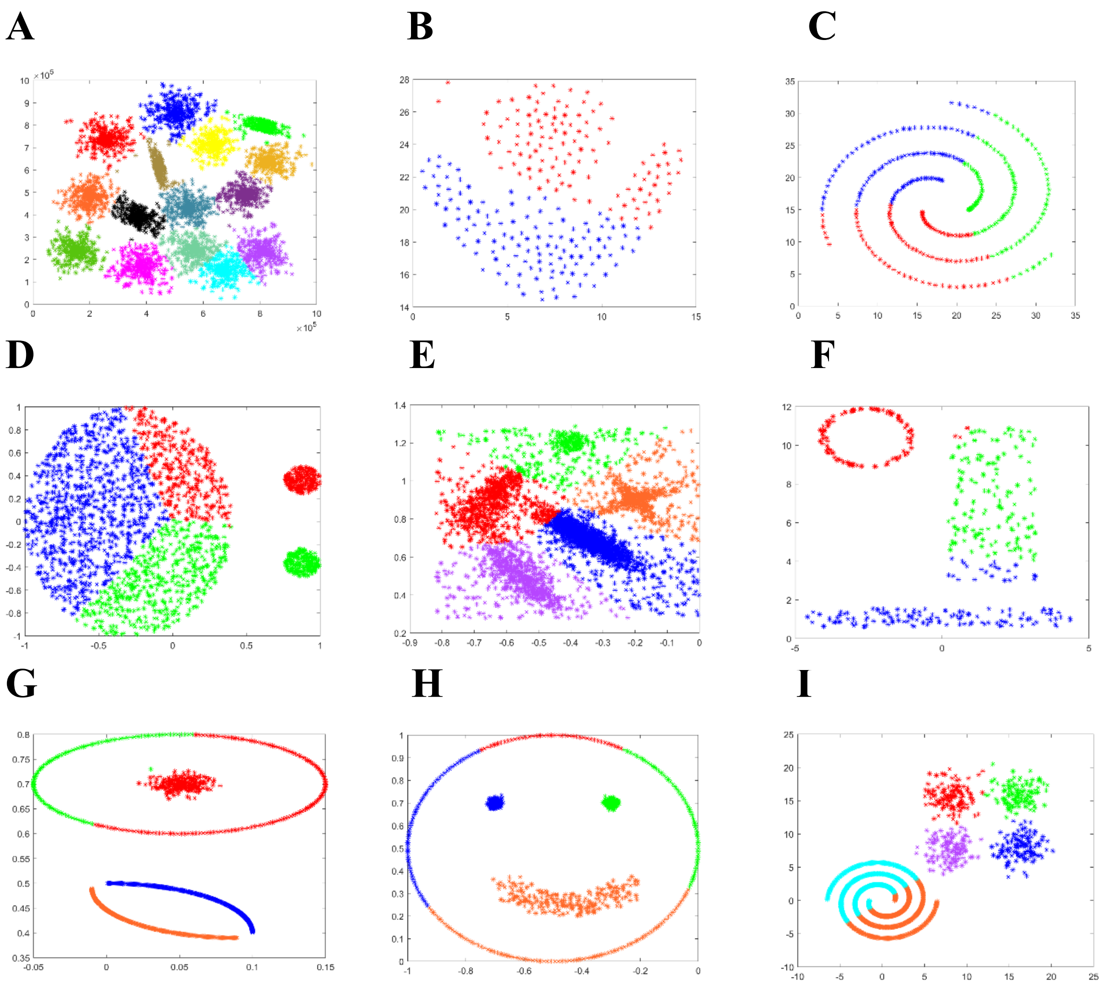


Fig. S3.1. K-means comparisons to the nine experiments in Fig. 3.5. The results reported are the best solution from 100 runs according to the ground-truth labels or, in the case of data set E, the objective function, as this data set does not contain ground-truth labels. The initialization method was K-means++. The value of K was set to the ground-truth number of clusters.

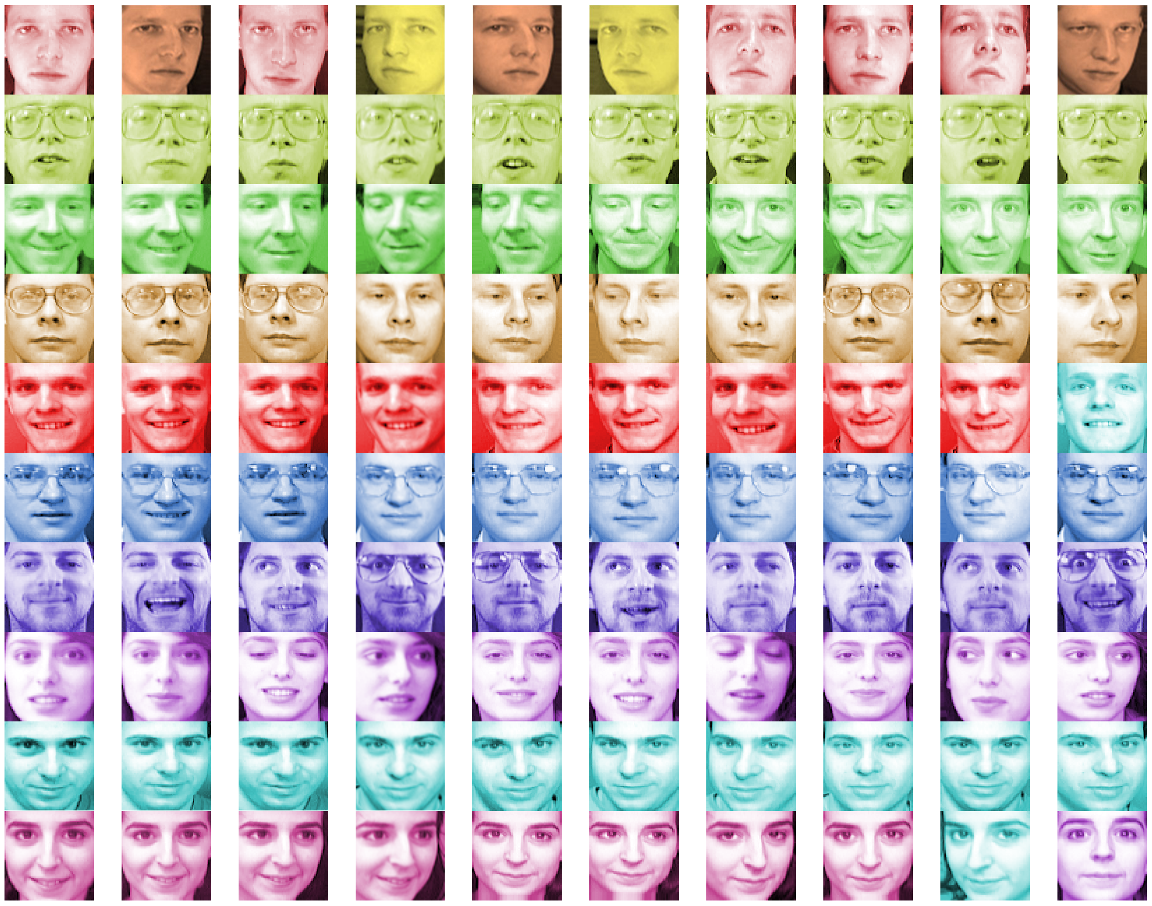


Fig. S3.2. Cluster analysis of the first 100 images of the Olivetti Face Database. Faces with the same color wash belong to the same cluster, as identified by TC. TC recognized 12 faces with a ground truth of 10, giving a 92% accuracy rate. The two extra clusters returned were as a result of the method we used for calculating similarity (i.e., Ref. [146]). This method is biased toward rotations, which, in this case, made the intra-cluster distance of Class 1 much larger than the other classes. In other words, it made the samples in Class 1 too sparse, which caused the TC algorithm to identify two extra abnormal connections (see Table. S3.3).

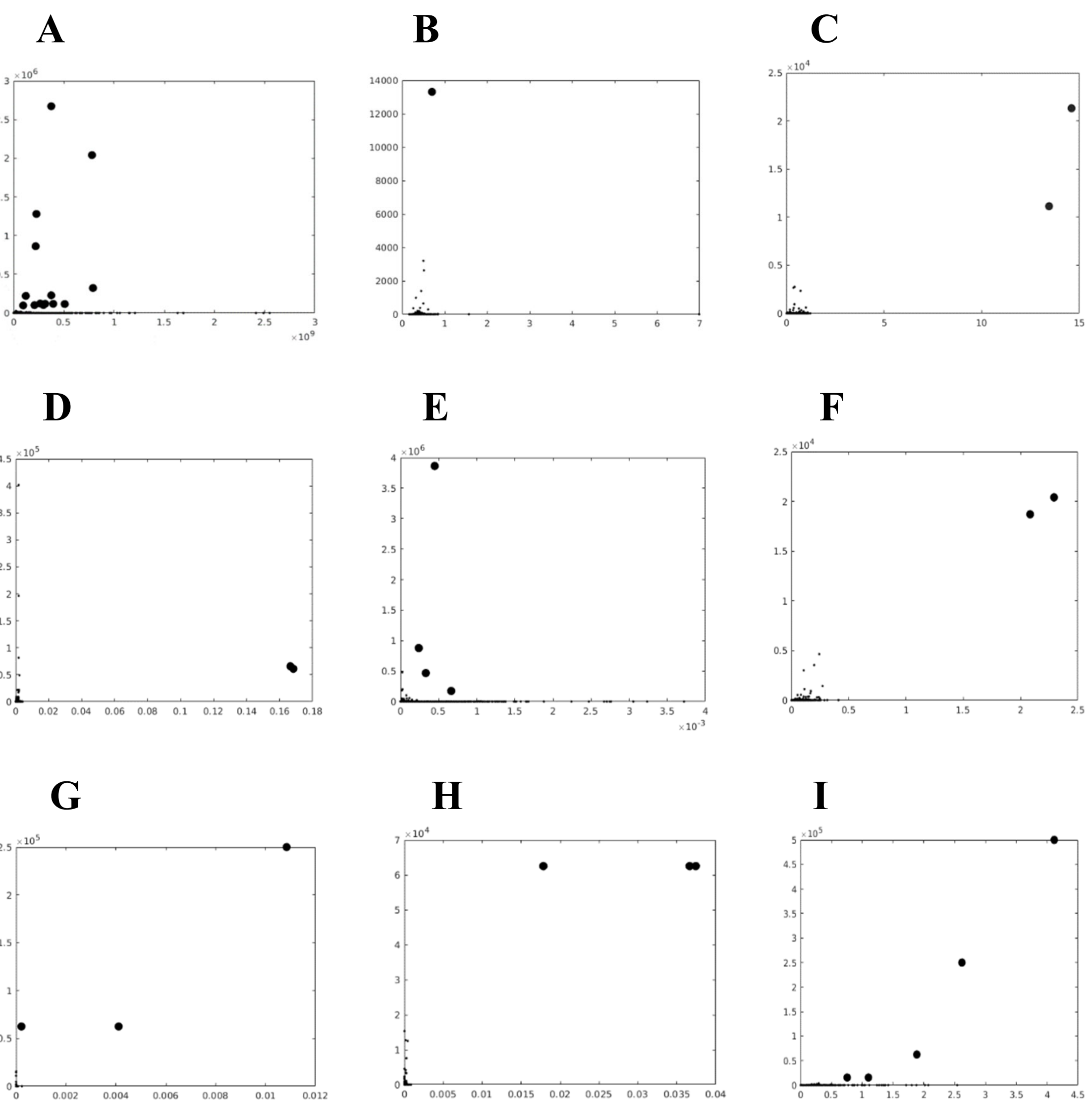


Fig. S3.3. TC decision graphs for the data point distributions in Fig. 3.5. D_i is on the horizontal axis, and M_i is on the vertical axis. The abnormal connections determined by TC algorithm in each data set appear in bold. Removing these connections leaves the final cluster partitions for each data set.

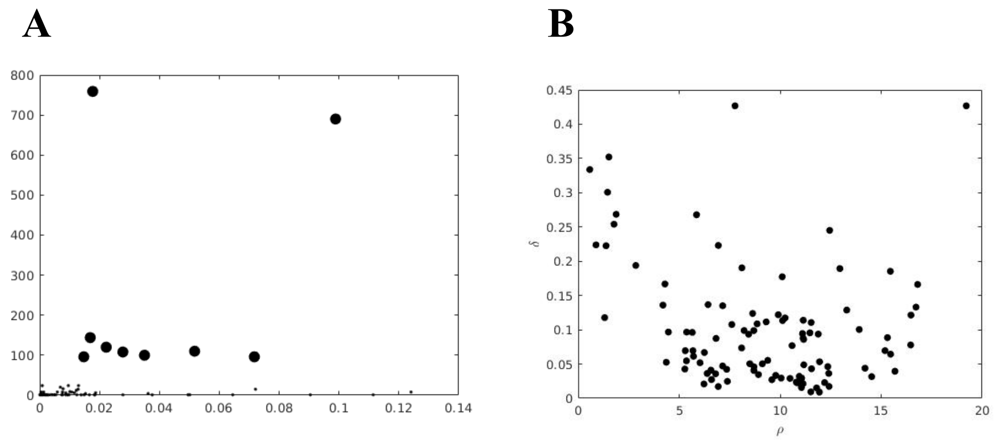


Fig. S3.4. TC and DPC decision graphs for the Olivetti Face Database tests. TC's decision graph is helpful for correcting excessive sensitivity to abnormal connections. Here, **(A)** shows TC with D_i on the horizontal axis and M_i on the vertical axis. TC automatically determined the number of abnormal connections to be 11, inconsistent with the ground-truth of 9. However, it is easy to see the nine abnormal connections in the decision, which appear in bold. Removing them leaves the 10 ground-truth clusters. By comparison, **(B)** shows the decision graph for DPC with data density on the horizontal axis and density-relative distance on the vertical axis. Identifying the correct 10 clusters here by eye would be extremely difficult, if not impossible.

3.6.1 Experimental details

In this study, we benchmarked TC against 19 representative clustering algorithms on 20 data sets (nine synthetic + 11 real-world). Following common practice, Euclidean metric was used as the similarity measure for most data sets. However, considering the advantage of cosine metric in processing functional data sets [183] and capturing semantic relations [184], [185], cosine metric was used as the similarity measure for these data sets: MNIST, COIL-100, Shuttle, RNA-seq, Haberman, and CMU-PIE. Additionally, Jaccard metric was used for the categorical data sets with non-numeric features, Zoo and Soybean, following Ref. [186], and the image similarity measure outlined in Ref. [146] was used for the Olivetti face database (OFD-F100) in consideration of comparison fairness. Figure S3.8 gives an example to illustrate the robustness of TC with respect to changes in the metric.

Details of the 19 baselines chosen for comparison follow. The parameter settings for those algorithms that require them are given in Table S3.1.

K-M++: K-means++ [20]

We used the implementation provided in Matlab and, as recommended, report the best index values at termination from 100 runs.

GMM: Gaussian mixture model clustering

We used the implementation provided by Ref. [187] and, again, report the best index values at termination from 100 runs as recommended.

Fuzzy: Fuzzy clustering [68]

We used the Matlab implementation and present the results averaged over 100 random initializations.

SC: Spectral clustering [72], [129]

As with Fuzzy, we used the Matlab implementation and present the results averaged over 100 random initializations.

AC-S: Hierarchical agglomerative clustering single-linkage

The implementation was provided by Matlab.

AC-C: Hierarchical agglomerative clustering complete-linkage

As above, the implementation was provided by Matlab.

AC-A: Hierarchical agglomerative clustering average-linkage

As above, Matlab.

AC-W: Hierarchical agglomerative clustering ward-linkage

As above, Matlab.

AC-CR: Hierarchical agglomerative clustering centroid-linkage [127]

As above, Matlab.

DBSCAN (DB) [74]

Another implementation from Matlab. DB has two parameters: Minpts and Eps. The paper [188] gives a rule of thumb, Minpts = 2*no. of dimensions, which we also followed on the low-dimensional data sets, i.e., Highly overlapping, FLAME, Spectral-path, Unbalanced, Noisy, Heterogeneous geometric, Multi-objective 1, Multi-objective 2, Multi-objective 3, Shuttle, Haberman, and Atom. However, if we had followed this rule for the high-dimensional data sets (OFD-F100, MNIST, COIL-100, RNA-seq, Zoo, Soybean, Cell-track and CMU-PIE), Minpts would be close to or greater than the total number of samples, which means most or all samples would be considered noise. Therefore, for the high-dimensional data sets, we tested $Minpts = 10, 20, 30, 40, 50$, then chose the best setting for each data set.

The Eps settings for all data sets were determined by the method in Ref. [189], i.e., $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d(x_i, \bar{x})$, where $\bar{x} = \sum_{j=1}^{n} \frac{x_j}{n}$, $d(,)$ denotes the distance, and x_i or x_j means the samples. We tested $Eps = \bar{d}, \bar{d}/2, \bar{d}/3, \ldots, \bar{d}/10$, and chose the best setting for each data set.

MS: Mean Shift [130]

We used the implementation provided by Ref. [190]. MS has a bandwidth parameter h. We borrowed the idea from paper [191] to set $h = \frac{1}{n}\sum_{i=1}^{n} d\left(x_i, x_{i,k}\right)$, where $x_{i,k}$ is the kth distant neighbor from x_i. In addition, we tested k = [5% *n*], [10% *n*], [15% *n*], …, [30% *n*], then chose the best setting for each data set, where [ ] means rounding.

AP: Affinity Propagation [131]

We used the codes provided by the authors [192] with their recommended parameter settings.

DPC: Clustering by fast search and find of density peaks [75]

We followed the implementation recommended in the original paper [193]. DPC has a parameter, dc, which is used to calculate the density of samples. To maximize the clustering quality, we followed the guidelines in the paper and tested $dc = 1.0\%$ through 2.0% in steps of .1%, then we chose the best setting for each data set. Further, in some data sets, it was very difficult to determine the optimal number of clusters from the decision graphs alone (see Fig. S3.4B). Hence, to maximize DPC's clustering quality with these data sets, we used the ground-truth number of clusters instead of referring to the decision graphs.

FINCH: Efficient parameter-free clustering using first neighbor relations [13]

With this algorithm, we also followed the implementation recommended in the original paper [194]. FINCH is a parameter-free algorithm that provides several options cluster partitions and asks the user to make a subjective decision as to which is the "ideal" scheme. As an example, with the MNIST image set, FINCH generated five partitioning schemes ranging from 1,699 clusters right down to 10 clusters. Therefore, to maximize clustering quality, we chose the scheme closest to the ground-truth number of clusters, which accords with the authors' approach [13].

RCC: Robust continuous clustering [1]

We used the Python implementation provided by Ref. [195] with the default recommended parameter settings.

BP: Border-Peeling Clustering [132]

We used the codes provided by the authors [196] with their recommended parameter settings.

DPCLP: Dynamic graph-based label propagation for density peaks clustering [27]

We followed the implementation from the authors [197]. This implementation has two parameters that need to be input: p (fraction of instances) and C (number of clusters). According to the original paper, we set p to [0.0002, 0.001, 0.01, 0.16] and chose the best setting for each data set. We set C to the ground-truth number of clusters.

DPA: Automatic topography of high-dimensional data sets by non-parametric Density Peak clustering [77]

We used the Python implementation provided by Ref. [198] with the default recommended parameter settings.

SNNDPC: Shared Nearest Neighbor-based Clustering by Fast Search and Find of Density Peaks [76]

We used the Matlab implementation provided by the authors [199], and set the algorithm to assign the hyper-parameter *K* automatically. Besides, we set the target number of clusters to the ground-truth number of clusters.

Table S3.1. Parameter settings for baselines

Algorithm	Parameter settings
K-means++	EmptyAction= 'singleton', MaxIter= 100, Replicates= 1, K= ground-truth number of clusters
GMM	tol = 10^{-10}, maxiter = 500, required number of clusters = ground-truth number of clusters
Fuzzy	exponent for the matrix U= 2.0, MaxIter= 100, threshold= 10^{-5}, required number of clusters = ground-truth number of clusters
SC	LaplacianNormalization= 'randomwalk', SimilarityGraph= 'knn', NumNeighbors= 10, KNNGraphType= 'complete', ClusterMethod= 'kmeans'
AC-S	'cutoff'= ground-truth number of clusters
AC-C	'cutoff'= ground-truth number of clusters
AC-A	'cutoff'= ground-truth number of clusters
AC-W	'cutoff'= ground-truth number of clusters
AC-CR	'cutoff'= ground-truth number of clusters
DB	– *Minpts* = 2*the no. of dimensions for the low-dimensional data sets (Highly overlapping, FLAME, Spectral-path, Unbalanced, Noisy, Heterogeneous geometric, Multi-objective 1, Multi-objective 2, Multi-objective 3, Shuttle, Haberman, and Atom). – $Minpts \in \{10, 20, 30, 40, 50\}$ for the high-dimensional data sets (OFD-F100, MNIST, COIL-100, RNA-seq, Zoo, Soybean, Cell-track and CMU-PIE) – $Eps \in \{\bar{d}, \bar{d}/2, \bar{d}/3, \dots, \bar{d}/10\}$, where $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d(x_i, \bar{x})$, $\bar{x} = \sum_{j=1}^{n} \frac{x_j}{n}$ for all data sets
MS	$h = \frac{1}{n}\sum_{i=1}^{n} d(x_i, x_{i,k})$, where $k \in \{[5\%\, n], [10\%\, n], [15\%\, n], \dots, [30\%\, n]\}$
AP	'maxits'=1000, 'convits'=100, 'dampfact'=0.9, Preference = median of similarities
DPC	$dc \in \{1.0\%, 1.1\%, 1.2\%, 1.3\%, 1.4\%, 1.5\%, 1.6\%, 1.7\%, 1.8\%, 1.9\%, 2\%\}$, ideal number of clusters= ground-truth number of clusters
DPCLP	$p \in \{0.0002, 0.001, 0.01, 0.16\}$, *C*= ground-truth number of clusters
DPA	Z=1, k_max=1000 or k_max=n/2 when the number of samples is less than 1000
SNNDPC	'AutoPick'= ground-truth number of clusters, K=0
RCC	k=10, verbose='True', preprocessing='none', clustering_threshold=1.0
BP	pca='none', spectral='none'

All experiments were performed in Matlab2019b or Python 3.6 (2.7 for BP algorithm as the authors recommended), and performance was evaluated against the two well-known external metrics, NMI [200] and ACC [201]. The results for all baselines on all data sets are reported in Tables 3.3A and 3.3B, with the maximum values highlighted in bold. We also ranked each algorithm according to its average performance across all data sets. If an algorithm could not be scaled to a data set, we set its ranking to last for that data set. We also evaluated the ability of the automatic clustering algorithms to estimate the ground-truth number of clusters. The number of clusters determined by each automatic algorithm is provided in Table 3.4. Exact estimates are marked in bold. When an algorithm needed to be run many times under different parameters to determine the optimal settings, i.e., DB and MS, we took the number of clusters corresponding to their maximum metric values. The Noisy data set does not contain ground-truth labels, so we could not calculate values for the external metrics. However, the ground-truth number of clusters is known, so, for DB and MS, we took the number of clusters closest to the ground-truth from many runs.

Table S3.2. Hierarchical tree of TC for each data set.
Table S3.2 shows the number of clusters in each layer of the hierarchical tree for each data set produced by the TC algorithm. In Step 0, the number of clusters equals the number of samples but, by the end of the process, each data set will have been merged into one giant cluster. Notably, TC's hierarchies averaged 7.5 mergers (layers) across the 20 data sets used in this study. A standard hierarchical clustering algorithm would require an average of 4784.1 mergers because it needs to merge each sample *n*-1 times, where *n* is the number of samples in the data set.

Step	Step**0**	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8	Step9	Step10	Step11	Step12
Highly overlapping	5000	1530	618	267	99	38	19	9	5	4	1	-	-
FLAME	240	63	22	9	4	2	1	-	-	-	-	-	-
Spectral-path	312	113	43	21	10	6	4	2	1	-	-	-	-
Unbalanced	2000	626	259	118	54	25	10	4	1	-	-	-	-
Noisy	4000	1217	490	209	98	47	22	8	4	2	1	-	-
Heterogeneous geometric	400	126	51	28	14	9	5	3	1	-	-	-	-
Multi-objective 1	1000	342	146	65	32	14	6	4	2	1	-	-	-
Multi-objective 2	1000	332	153	69	33	16	12	9	6	4	1	-	-
Multi-objective 3	1500	164	70	31	13	5	3	2	1	-	-	-	-
OFD-100	100	31	11	4	1	-	-	-	-	-	-	-	-
MNIST	10000	1699	528	155	54	19	9	4	3	1	-	-	-
COIL-100	7200	2211	975	473	241	119	53	18	8	2	1	-	-
Shuttle	58000	15884	5963	2362	928	375	163	78	36	12	5	2	1
RNA-seq	801	80	18	7	3	2	1	-	-	-	-	-	-
Haberman	306	96	40	19	9	5	2	1	-	-	-	-	-
Zoo	101	23	7	3	1	-	-	-	-	-	-	-	-
Atom	800	240	104	38	16	6	2	1	-	-	-	-	-
Soybean	47	12	5	2	1	-	-	-	-	-	-	-	-
Cell-track	40	4	2	1	-	-	-	-	-	-	-	-	-
CMU-PIE	2856	799	213	108	54	11	2	1	-	-	-	-	-

Table S3.3. The intra-cluster distance of each class on OFD-F100 data set.
We took the average of the distances between any two samples in each class as the intra-cluster distance and found that the samples of Class 1 are too sparse because of the bias toward rotations of the similarity calculation of Ref. [146]. As a result, the TC algorithm automatically identified two more abnormal connections in Class 1, making 11 abnormal connections in total.

Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	Class 10
.3846	.1566	.2621	.2266	.2376	.1898	.0684	.1404	.2000	.2304

3.6.2 Further evaluation on additional 56 data sets with peculiar characteristics

In this set of tests, we examined how the eight clustering algorithms performed on the data sets from four different sources. These included: (1) data sets with noise, outliers, overlaps, or other peculiar distributions from a benchmark database; (2) synthetic data sets with different degrees of unbalance; (3) synthetic data sets with different degrees of uniform noise; and (4) poorly-separated, high-dimensional data sets.

The seven comparators included the two well-known algorithms of the 19 mentioned above, DB and SC. Moreover, since TC is hierarchy-based, five standard hierarchical clustering algorithms, AC-S, AC-C, AC-A, AC-W, AC-CR, were also included. For a fair comparison, considering the bias toward data unbalance of accuracy [202], the following results were measured only by NMI, which is the most widely used metric for evaluation in the clustering community [203].

A. Test on 27 data sets with noise, outliers, overlaps, or other peculiar distributions from the benchmark database

Table S3.4 provides the details of the data sets [134], [135], save to say each contains various numbers of clusters, degrees of noise, overlaps and outliers or other distribution peculiarities. Fig. S3.5 plots the results for the 27 data sets. TC achieved the highest NMI on 18 of them and was highest overall in mean rankings, outperforming the next-best algorithm, SC, by more than two times (1.59 vs. 3.59). TC returned the exact number of ground-truth clusters on 25 of the 27 data sets compared to the next best automatic algorithm, DB, with perfect accuracy on only 10.

B. Test on nine synthetic data sets with unbalanced clusters

In this experiment, to evaluate the robustness of TC to data set unbalance, we used the Highly overlapping data set in Fig. 3.5 and the parameter *s* was adjusted between 0.1 and 0.9 to manage the level of unbalance. Class 14 received a probability of 1, class 0 received a probability of *s*, and all other classes were linearly varied between *s*+0.1 and 1. We evaluated the performance of TC and the seven algorithms for each value of *s* in terms of NMI. The results appear in Fig. S3.6 and Table S3.5. TC still retained the performance advantage on the data sets with unbalanced clusters and identified the correct numbers of clusters on all the nine data sets.

C. Test on five synthetic data sets with different degrees of uniform noise

In this experiment, to evaluate the robustness of TC to noise, we used the original synthetic data set in Fig. 3.3 and controlled the degree of noise by adding 5%-30% uniform noise respectively. We compared TC and two density-based algorithms, DB and DPC, which have the noise detection capability. TC achieved the highest NMI and identified the correct numbers of clusters on all the five data sets. Compared with the other two algorithms, TC is more robust to noise. Table S3.6 and Fig. S3.7 give the quantitative and visualized results, respectively.

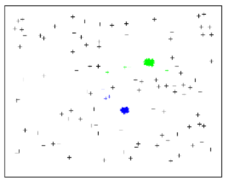
D. Test on 15 poorly-separated, high-dimensional gene expression data sets

These poorly-separated, high-dimensional real-world data sets of cancer gene expressions [204] come from various tissues of the human body. They also contain outliers. The results are shown in Table S3.7. TC achieved the highest NMI on 11 of the 15 data sets, and SC was highest on only three but was given knowledge of the ground-truth number of clusters in advance. TC achieved the highest mean NMI across the data sets, and interestingly, it outperformed the next-best algorithm, SC, by about 9%. Additionally, TC returned the exact number of ground-truth clusters on seven of the 15 data sets compared to the next best automatic algorithm, DB, with perfect accuracy on only three.

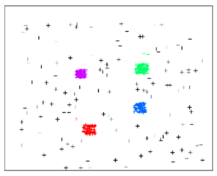
Table S3.4. Performance comparison on 27 data sets with noise, outliers, overlaps, or other peculiar distribution, measured by NMI.
#C means the ground-truth cluster number, and NC stands for the cluster number identified by the algorithms. NGC indicates the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters. "NA" means not applicable; here, SC broke within 100 runs on the Insect data set.

		NC set manually						NC determined automatically			
Algorithm		**AC-S**	**AC-C**	**AC-A**	**AC-W**	**AC-CR**	**SC**	**DB**		**TC**	
Data Set	**#C**	**NMI**						**NC**	**NMI**	**NC**	**NMI**
Zelnik2	2	.0776	.5164	.1063	.5165	.1275	.5178	**2**	**1**	**2**	.9084
Zelnik4	4	.0845	.7255	.6176	.7224	.6185	.7220	**4**	**.9897**	**4**	.9609
Cluto-t5-8k	6	.0173	.6817	.8183	.8229	.8198	.0210	1	0	**6**	**.8838**
S1	15	.8090	.9770	.9835	.9848	.9830	.0127	8	.8409	**15**	**.9880**
S2	15	.0334	.8786	.9308	.9268	.9259	.0121	2	.2795	**15**	**.9341**
S3	15	.0376	.7071	.7516	.7705	.7437	.0123	11	.0508	**15**	**.7800**
S4	15	.0373	.6200	.6690	.6942	.6585	.0119	6	.0463	**15**	**.7164**
2d-20c-no0	20	.9164	.9230	.9600	.9908	.9481	.9919	**20**	.9786	**20**	**.9943**
2d-4c-no4	4	.7029	.6880	.8698	.9930	.8698	.7029	10	.8668	**4**	**.9930**
DS-850	5	.7206	.8052	.9470	.9750	.9316	.9862	**5**	.9715	**5**	**.9918**
D31	31	.6784	.9519	.9517	.9508	.9518	**.9600**	5	.5803	**31**	.9573
R15	15	.8822	.9844	.9922	.9864	.9913	**.9942**	12	.9203	**15**	.9893
2d-10c	9	.9538	.9257	.9963	1	1	.9516	8	.9545	**9**	**1**
Insect	3	.1859	.5399	.5031	.5654	.5031	NA	2	.4614	2	**.5887**
Longsquare	6	.6860	.8525	.9088	.8719	.9001	**.9897**	10	.9128	**6**	.9834
Square4	4	.0230	.6012	.6915	.6859	.7139	**.7253**	9	.5525	**4**	.6940
Tetra	4	.0386	.9823	.9899	.9640	1	1	2	.6369	**4**	**1**
Triangle2	4	.4955	.7985	.9014	.9550	.9060	**.9829**	6	.9038	**4**	.9763
Chainlink	2	1	.3899	.3615	.3673	.4519	1	**2**	1	**2**	**1**
Compound	6	.8109	.8100	.8371	.7338	.8394	.8335	5	**.9300**	3	.8217
Diamond9	9	.8022	.9990	.9967	.9961	.9967	.9971	1	0	**9**	**1**
Ds4c2sc8	8	.1917	.7213	.8458	.8783	.7907	**.8866**	9	.7163	**8**	.8586
Lsun	3	1	.5317	.4988	.5150	.4988	1	**3**	1	**3**	**1**
Wingnut	2	1	1	1	.4935	1	.9795	**2**	1	**2**	**1**
Zelnik3	3	1	.5156	.5595	.5723	.5595	1	**3**	1	**3**	**1**
Zelnik5	4	1	.5797	.6481	.6695	.5501	1	**4**	1	**4**	**1**
Banana	2	1	.5067	.3763	.3934	.6020	1	**2**	1	**2**	**1**
Rank		5.93	5.48	4.52	4.22	4.19	3.59	4.33		**1.59**	
NGC		-						10		**25**	

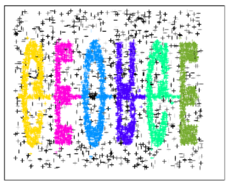
Zelnik2



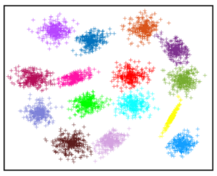
Zelnik4



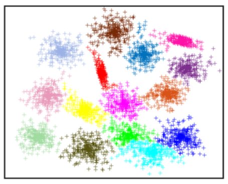
Cluto-t5-8k



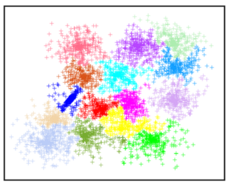
S1



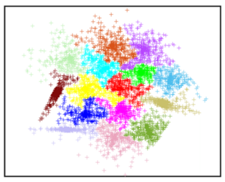
S2



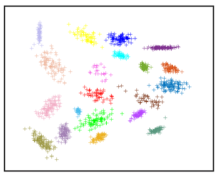
S3



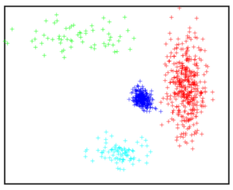
S4



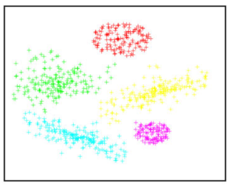
2d-20c-no0



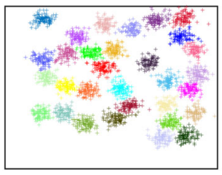
2d-4c-no4



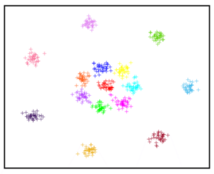
DS-850



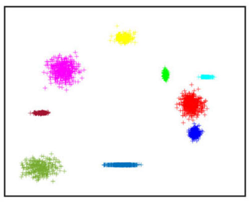
D31



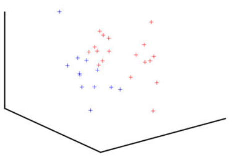
R15



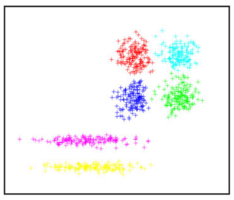
2d-10c



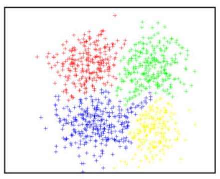
Insect



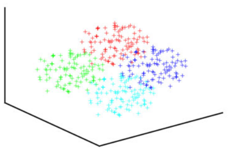
Longsquare



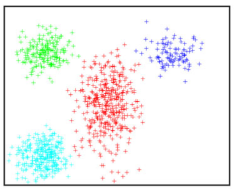
Square4



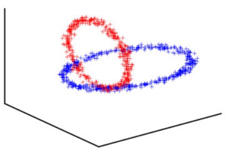
Tetra



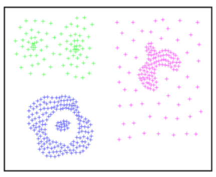
Triangle2



Chainlink



Compound



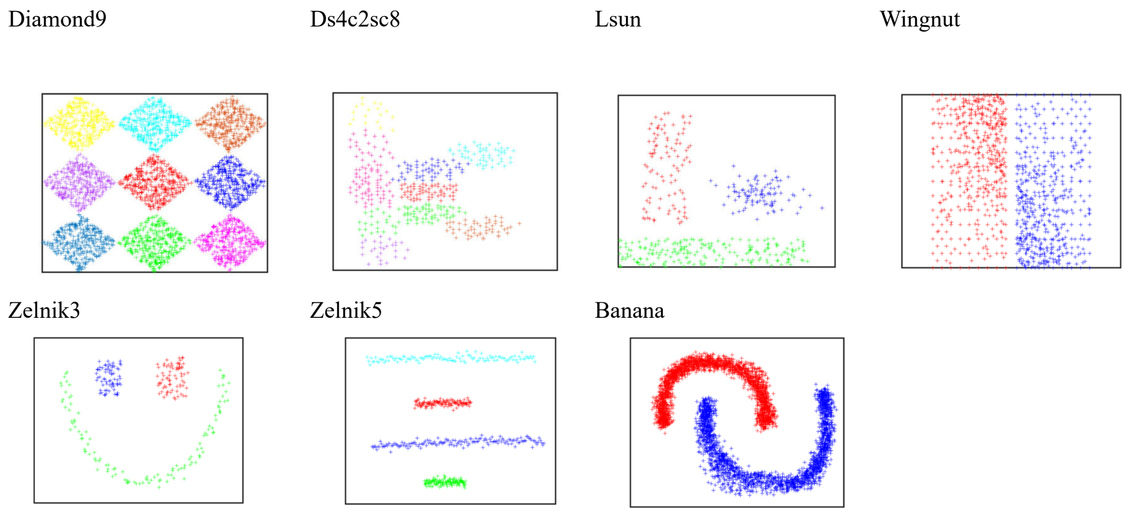


Fig. S3.5. TC's results on 27 data sets with noise, outliers, overlaps, or other peculiar distribution.

Table S3.5. Performance comparison on nine synthetic data sets with unbalanced clusters, measured by NMI.

We used the Highly overlapping data set (see Fig. 3.5) and the parameter s was adjusted between 0.1 and 0.9 to manage the level of unbalance. Class 14 received a probability of 1, class 0 received a probability of s, and all other classes were linearly varied between s+0.1 and 1. #C means the ground-truth cluster number, and NC means the cluster number identified by the algorithms. NGC indicates the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters. TC still retained the performance advantage on the data sets with unbalanced clusters.

		NC set manually						NC determined automatically			
Algorithm		**AC-S**	**AC-C**	**AC-A**	**AC-W**	**AC-CR**	**SC**	**DB**		**TC**	
Data Set	**#C**	**NMI**						**NC**	**NMI**	**NC**	**NMI**
s=0.1	15	.6090	.9241	.9507	.9446	.9417	.0412	17	.8805	15	**.9512**
s=0.2	15	.6127	.9365	.9556	.9459	.9308	.0300	14	.8788	15	**.9560**
s=0.3	15	.5243	.9146	.9518	.9354	.9409	.0260	8	.5964	15	**.9526**
s=0.4	15	.2542	.8527	.9555	.9459	**.9566**	.0216	6	.6068	15	.9449
s=0.5	15	.2619	.8494	.9395	.9331	.9382	.0185	5	.5988	15	**.9599**
s=0.6	15	.2629	.8840	.9418	.9379	.9335	.0163	7	.5569	15	**.9538**
s=0.7	15	.0388	.8760	.9460	.9496	.9481	.0161	2	.2622	15	**.9596**
s=0.8	15	.2701	.8548	.9392	.9444	.9378	.0138	2	.2748	15	**.9621**
s=0.9	15	.0348	.8939	**.9593**	.9391	.9551	.0131	2	.2723	15	.9565
Rank		7.0	4.89	2.22	3.11	3.33	8.0	6.0		**1.44**	
NGC		-						0		**9**	

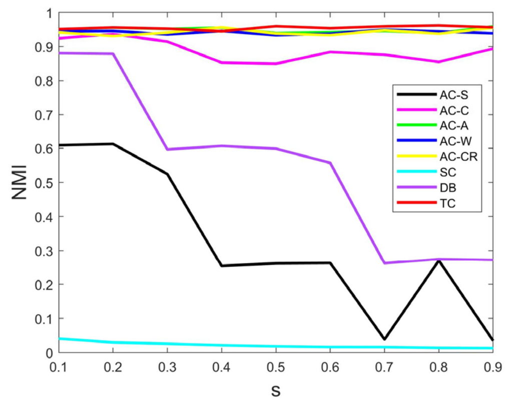
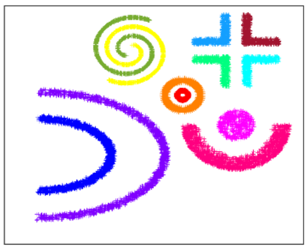


Fig. S3.6. The performance comparison of eight methods on the data sets with unbalanced clusters.

Table S3.6. Performance comparison on five synthetic data sets with different degrees of uniform noise, measured by NMI.
We used the original synthetic data sets in Fig. 3.3 and controlled the degree of noise by adding 5%-30% uniform noise respectively. #C stands for the ground-truth cluster number, and NC denotes the cluster number identified by the automatic algorithms. NGC indicates the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters. Compared with the other two algorithms, TC is more robust to noise.

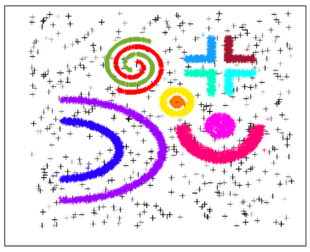
	Data set		0% noise	5% noise	10% noise	20% noise	30% noise	***Rank***	***NGC***
	#C		12	12	12	12	12		
NC determined automatically	TC	NC	**12**	**12**	**12**	**12**	**12**	**1.0**	**5**
		NMI	**1**	**.9765**	**.9614**	**.9418**	**.9163**		
	DB	NC	**12**	22	52	20	1	2.4	1
		NMI	**1**	.8508	.6929	.0755	.0097		
NC set manually	DPC	NMI	.8266	.6924	.7017	.6065	.4882	2.4	-

A



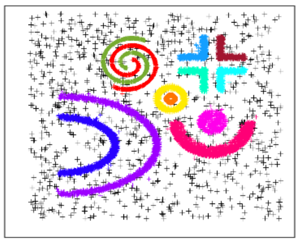
0% noise

B



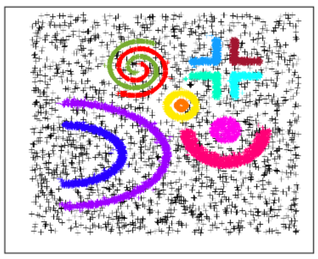
5% noise

C



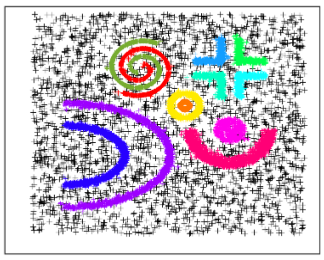
10% noise

D



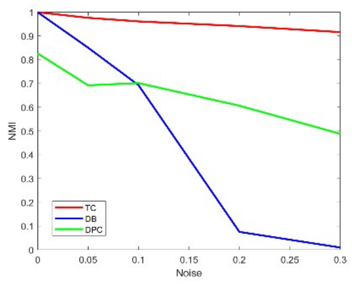
20% noise

E



30% noise

F



Robustness comparison

Fig. S3.7. Results on the synthetic data sets with different degrees of uniform noise. (A)- (E) visualize the results of TC on the data sets with different degrees of uniform noise. (F) gives the comparison results of TC and other two density-based clustering algorithms on the data sets with different degrees of uniform noise, measured by NMI. Compared with the other two algorithms, TC is more robust to noise.

Table S3.7. Performance comparison on 15 poorly-separated, high-dimensional gene expression data sets, measured by NMI.

These are data sets of cancer gene expressions from Ref. [204]. They are poorly-separated, high-dimensional, and contain outliers. #C stands for the ground-truth cluster number, and NC indicates the cluster number identified by the algorithms. NGC indicates the number of data sets each automatic clustering algorithm returns the exact ground-truth number of clusters. Note that DB returned an NC of 0 on the Nutt-2003-v3 and Pomeroy-2002-v1 data set, which means it classified all the data samples as noise.

		NC set manually						**NC determined automatically**			
Algorithm		**AC-S**	**AC-C**	**AC-A**	**AC-W**	**AC-CR**	**SC**	**DB**		**TC**	
Data Set	**#C**	**NMI**						**NC**	**NMI**	**NC**	**NMI**
Alizadeh-2000-v2	3	.0342	.5608	.7226	.5490	.7226	.5838	1	.6325	2	**.7642**
Alizadeh-2000-v3	4	.5847	.4528	.5987	.4602	.5847	.5934	1	.5077	2	**.6258**
Armstrong-2002-v2	3	.0673	.7812	.5849	.7580	.5714	**.8944**	2	.4496	**3**	.8074
Bittner-2000	2	.0640	.0323	.0026	.0588	.0640	.0183	**2**	.1120	7	**.4223**
Golub-1999-v1	2	.0684	.0660	.1376	**.8328**	.1376	.6916	**2**	.5494	**2**	.6190
Gordon-2002	2	.0083	.6482	.0219	.6709	.0083	**.7469**	1	.5994	**2**	.6659
Laiho-2007	2	.0263	.0085	.0340	.0177	.0263	.0177	1	.1453	7	**.1651**
Lapointe-2004-v1	3	.0466	.0684	.1826	.1826	.0285	.0314	1	.1057	14	**.3086**
Liang-2005	3	**.3545**	**.3545**	**.3545**	**.3545**	**.3545**	**.3545**	2	**.3545**	**3**	**.3545**
Nutt-2003-v3	2	.0527	.1395	.0527	.0420	.0527	.0196	0	0	6	**.2520**
Pomeroy-2002-v1	2	.1457	.1337	.0079	.0079	.1457	.0688	0	0	7	**.1715**
Shipp-2002-v1	2	.0937	.0602	.0937	.1477	.0937	.1770	**2**	.1265	**2**	**.1811**
Tomlins-2006	5	.1196	.3192	**.4933**	.4848	.0957	.4463	1	.0556	**5**	.4503
Tomlins-2006-v2	4	.1120	.1850	.3491	.3311	.0928	.3536	1	.0482	21	**.4807**
West-2001	2	.0563	.4508	.4965	.4965	.0563	.4965	1	.3578	**2**	**.5231**
Rank		5.13	5.2	3.6	3.87	4.73	3.67	5.13		**1.47**	
Mean		.1223	.2841	.2755	.3596	.2023	.3663	.2696		**.4528**	
NGC		-						3		**7**	

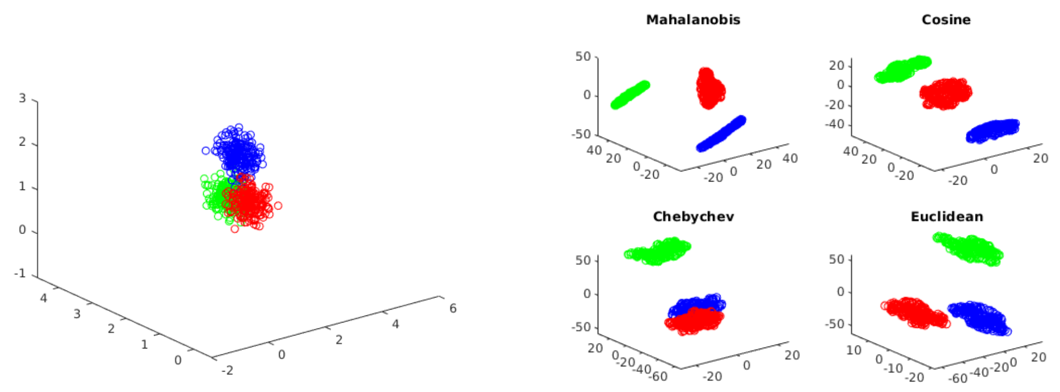


Fig. S3.8. The robustness of TC with respect to changes in the metric. (A) shows the original data distribution. **(B)** shows the clustering results of TC under four different distance metrics, visualized by t-SNE. TC achieved a perfect accuracy among these four cases.

3.6.3 Robustness guarantees for the proposed TC

In this section, we showed that if we add some samples (possibly adversarially) to the original data set, then the number of clusters will not change and the clustering assignments are preserved (i.e. no new clusters generated, no old clusters merged, etc.) when using TC. The related proof is based on two previous definitions of data distribution, one is $(\alpha, K) - strictly\ separable$ and the other is $(\varepsilon, K) - strictly\ \ additive\ separable$ [205].

<u>*Guarantees on Results when manually setting number of clusters*</u>.

The original data set is denoted as X, and the data set after adding adversarial samples is denoted as X'. The adversarial samples are denoted as $W = \{w|w \in X', w \notin X\}$.

Theorem 1. Suppose a data set X is $(\varepsilon, K) -\ strictly\ additive\ separable$, adding some adversarial samples W to $X\ to\ get\ X'$. If $\max_{w\in W}\min_{x\in X} d(w,x) \leq \varepsilon$ and X' is $\left(\sqrt{n_i}\,/\,2\sqrt{n_j}, K\right) - strictly\ separable$, the clustering assignments are preserved when setting the number of clusters as K.

Proof: Since *X* is $(\varepsilon, K) - \mathit{strictly\ additive\ separable}$, so for all $i \neq j$ and all $x, y \in \zeta_i, z \in \zeta_j$, $d(x,y) + \varepsilon \leq d(x,z)$. And because the adversarial sample W satisfies $\max_{w \in W} \min_{x \in X} d(w,x) \leq \varepsilon$, according to Eq. (3.1) of TC, all adversarial samples will be merged by some or all of the unique clusters in $\{\zeta_1, \ldots, \zeta_K\}$. On the other hand, since X' is $\left(\sqrt{n_i} \,/\, 2\sqrt{n_j}, K\right) - \mathit{strictly\ separable}$, then for all $i \neq j$ and all $x, y \in \zeta_i, z \in \zeta_j$,

$$\frac{1}{2}\sqrt{\frac{n_i}{n_j}}\, d(x,y) \leq d(x,z) \tag{3.11}$$

where n_i and n_j are the number of samples in ζ_i and ζ_j. Because $d(x,y), d(x,z) \geq 0$, then

$$\frac{1}{4}\frac{n_i}{n_j} d^2(x,y) \leq d^2(x,z) \tag{3.12}$$

We can further derive

$$\frac{n_i^2}{4} d^2(x,y) \leq n_i n_j d^2(x,z) \tag{3.13}$$

Suppose there are any two sub-clusters in ζ_i, the numbers of samples they contain are $sn_o$and sn_p respectively, so

$$sn_o + sn_p \leq n_i \tag{3.14}$$

Since $sn_o + sn_p \geq 2\sqrt{sn_o sn_p}$, then

$$n_i^2 \geq 4 sn_o sn_p \tag{3.15}$$

So according to (13), we have

$$sn_o sn_p d^2(x,y) \leq \frac{n_i^2}{4} d^2(x,y) \leq n_i n_j d^2(x,z) \tag{3.16}$$

that is,

$$sn_o sn_p d^2(x,y) \leq n_i n_j d^2(x,z) \tag{3.17}$$

According to the definition of τ_i (i.e., Eq. (3.5)), the Eq. (3.17) above, and the arbitrariness of x, y and z, if we perform TC on X′, the torque values of connections formed between ground truth clusters must be greater than the torque values of connections formed between sub-clusters in the ground truth clusters. Therefore, if setting the cluster number as *K*, the *K*-1 abnormal connections must be the connections formed between the ground truth clusters, the results will be preserved.

Guarantees on Results when automatically setting number of clusters.

Theorem 2. Suppose that the conditions of Theorem 1 hold. Let $TGap_i$ and $TGap_i{}'$ be the torque gap when performing TC on X and X', respectively. If additionally, the following holds:

$$arg\max_i TGap_i = arg\max_i TGap_i{}' = K - 1 \tag{3.18}$$

then the clustering assignments are preserved when using the mechanism for automatically determining abnormal connections (i.e., Eqs (3.6)-(3.9)).

Proof: According to the proof for Theorem 1 and the mechanism for automatically determining abnormal connections (i.e., Eqs (3.6)-(3.9)), the proof for Theorem 2 is obvious.

As an illustration, we added some adversarial samples to the three synthetic data sets introduced in the main text. As the Fig S3.9 shown, TC still achieved good results.

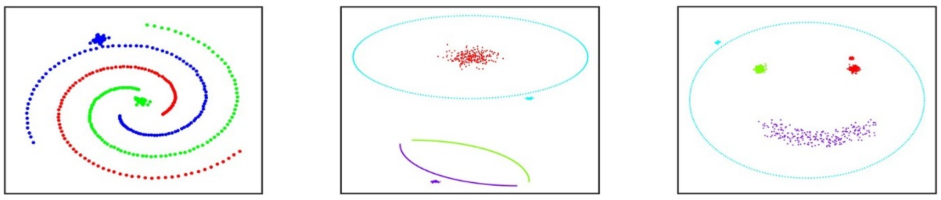


Figure S3.9. Results of TC on the data sets with adversarial samples.

Chapter 4. Multi-view adjacency-constrained hierarchical clustering

4.1 Introduction

Recently, multi-view clustering has been a research hotspot as an important learning paradigm in machine learning. Different from traditional clustering methods, multi-view clustering is exploited to process multi-view data. Multi-view data means the data is collected from different sources in diverse domains, or obtained from various feature collectors [38]. For example, multiple heterogeneous features can be used to characterize an image, such as scale-invariant feature transform (SIFT) descriptors [39], GIST descriptors [40], local binary patterns (LBP) [41], etc. Multiple compatible and complementary features are combined in multi-view clustering algorithms to improve clustering performance.

Yet despite the importance of multi-view clustering and the plethora of existing algorithms in past decades, most contemporary approaches in multi-view clustering have problems with the following two issues: a) parameter tuning and b) significant computational expense. For most multi-view clustering, e.g., multi-view spectral clustering [43]–[46] and multi-view subspace clustering [47]–[50], the final performance of the models is heavily dependent on parameter tweaking. For example, Zong et al. proposed a multi-view spectral clustering algorithm based on distinct view weights, which has two parameters that need to be set in order to assign an optimal weight to each view [46]. Zheng et al. proposed a constrained bilinear factorization multi-view subspace clustering algorithm, which also has two prior information-related parameters to tune in order to obtain competitive performance [49]. For most current multi-view clustering methods, prior knowledge, such as noise level and label information, is required to guide the specific parameter choice steps, which is troublesome. Furthermore, the computational complexity of most existing multi-view clustering algorithms is also high; multi-view clustering based on subspace learning and spectral representation learning, for example, both have time complexities of $O(n^3)$. Additionally, in some multi-view clustering algorithms [86], the iterative optimization of the objective function will also greatly increase the computational overhead. These two limitations significantly impede the practical use of multi-view clustering.

On the other hand, from the perspective of basic clustering principles, many previous multi-view clustering algorithms are based on spectral clustering or subspace clustering, which have some inherent limitations. For example, spectral clustering [72], [129] suffers from the following three problems: a) the instability of results caused by different initializations; b) the K value required to construct adjacency matrix needs to be adjusted; and c) it can only provide clustering results with a single granularity. For subspace clustering [206], a) establishing the global density threshold causes the method to perform poorly in detecting clusters with varying densities; and b) setting regularization parameters for the number of subspaces is time-consuming. Few multi-view clustering algorithms are based on hierarchical clustering [42]. Compared with spectral clustering and subspace clustering, hierarchical clustering does not need extra hyper-parameters, and a dendrogram can be generated to provide clustering results with different granularity levels.

We propose a Multi-view adjacency-Constrained Hierarchical Clustering algorithm (MCHC) in this chapter to overcome the issues above. MCHC consists of three main parts: including the Fusion Distance matrices with Extreme Weights (FDEW); adjacency-Constrained Nearest Neighbor Clustering (CNNC); and the internal evaluation Index based on Rawls' Max-Min criterion [207] (MMI). FDEW attempts to learn a fusion distance matrix set, which only uses complementary and consensus information among multiple views, but exploits the information from each single view. CNNC obeys an intuitive rule that one cluster and its nearest neighbor with higher mass (size) should be merged into one bigger cluster during the clustering procedure. CNNC generates multiple partitions based on FDEW. MMI is exploited to choose the best one from the multiple partitions. MCHC just needs to be assigned a desired number of clusters, which can be estimated based on the decision graph of CNNC. In addition, we propose a parameter-free version of MCHC (MCHC-PF). Without any parameter selection, MCHC-PF can give partitions at different granularity levels. MCHC-PF has lower time complexity, which is $O(nlogn)$.

The following are the main contributions of this chapter:

- 1) Proposing a multi-view adjacency-constrained hierarchical clustering (MCHC) algorithm that can obtain promising clustering results.
- 2) Proposing a parameter-free MCHC algorithm with low computational complexity.
- 3) Proposing the fusion distance matrices with extreme weights, which only uses complementary information among multiple views, but exploits the information from each single view.
- 4) Proposing the internal evaluation index based on Rawls' Max-Min criterion for selecting best partition.
- 5) The proposed methods' superiority is demonstrated by experimental results on eight real-world data sets.

4.2 Proposed method

4.2.1 Fusion distance matrices with extreme weights (FDEW)

When dealing with multi-view clustering problems, there are two intuitive methods to fuse multi-view data. One is concatenating all the features of multiple views, and then performing single-view algorithms directly on the concatenation [208]. Obviously, this method increases the dimensionality of fusion data, thereby increasing the computational complexity and possibly reducing the clustering accuracy because of the curse of dimensionality. The other method is to calculate an average similarity matrix $\bar{S} = \frac{1}{v}\sum_{i=1}^{v} S^{(i)}$, and then input it to a single-view clustering algorithm [209]. The above two methods give each view the same weight. In fact, the importance of each view may be different. Treating the data of each view equally may reduce the final clustering accuracy. There are many studies devoted to assigning different weights to each view according to the discriminative power of the view [209], but this also brings some problems. Firstly, this may introduce more parameters (i.e., weights-related parameters), thereby reducing the ease of use of the model; secondly, when using optimization algorithms to iteratively update the weights, it will greatly increase the computational cost and operation time. On the other hand, previous research has pointed out that sometimes the utilization of multiple views may even deteriorate the final performance, which is even worse than the performance of best single-view [210]. Therefore, only extracting information from each single view is also important.

Based on the above analysis, we propose fusion distance matrices with extreme weights (FDEW). FDEW gives the fixed weights to the distance matrix of each view, which does not need to be optimized. Besides, FDEW not only uses the complementary information among multiple views, but also exploits the information from each single view.

Given multi-view data $\{X^{(i)}\}_{i=1}^{v}$ collected from *v* views, for *i*-th view, $X^{(i)} \in R^{n\times dim_i}$, where *n* and dim_i denote the number of data points and the dimensions of the *i*-th view respectively.

On the one hand, we regard the distance matrix $D^{(i)}$ ($D^{(i)} \in R^{n\times n}$) of each view as a fusion distance matrix with extreme weights, that is

$$D^{(i)} = 1 \times D^{(i)} + \sum_{j=1,j\neq i}^{v} 0 \times D^{(j)} \tag{4.1}$$

On the other hand, we define a fusion distance matrix with equal weights:

$$D^{*} = \frac{1}{v}\sum_{i=1}^{v} D^{(i)} \tag{4.2}$$

$D^{(i)}$ assigns the weight of $D^{(i)}$ to 1, and assigns the weight of the distance matrix of other views to 0. D^{*} treats the distance matrix of each view equally, and assigns the same weight to the distance matrix of each view. $D^{(i)}$ only uses the information from each single view, but D^{*} exploits complementary information among multiple views. Combine $D^{(i)}$ and D^{*} to form fusion distance matrices with extreme weights (FDEW), where $\{FDEW^{(r)}\}_{r=1}^{v+1} = \{D^{(1)}, D^{(2)}, \ldots, D^{(v)}, and\ D^{*}\}$.

When calculating the distance matrix $D^{(i)}$ of each view, we exploit cosine distance. For the cosine distance between any two samples $x_a^{(i)}, x_b^{(i)}$ in the *i*-th view, it is defined as

$$d\left(x_a^{(i)}, x_b^{(i)}\right) = 1 - \frac{{x_a^{(i)}}^T x_b^{(i)}}{\sqrt{{x_a^{(i)}}^T x_a^{(i)}}\sqrt{{x_b^{(i)}}^T x_b^{(i)}}} \tag{4.3}$$

where $d\left(x_a^{(i)}, x_b^{(i)}\right) \in [0,2]$.

Theorem 1. The cosine distance between $x_a^{(i)} and\ x_b^{(i)}$ in the i-th view is equivalent to the cosine distance between them on the latent representation [42].

Next, we use a single-view clustering algorithm to get cluster partitions based on each distance matrix in FDEW.

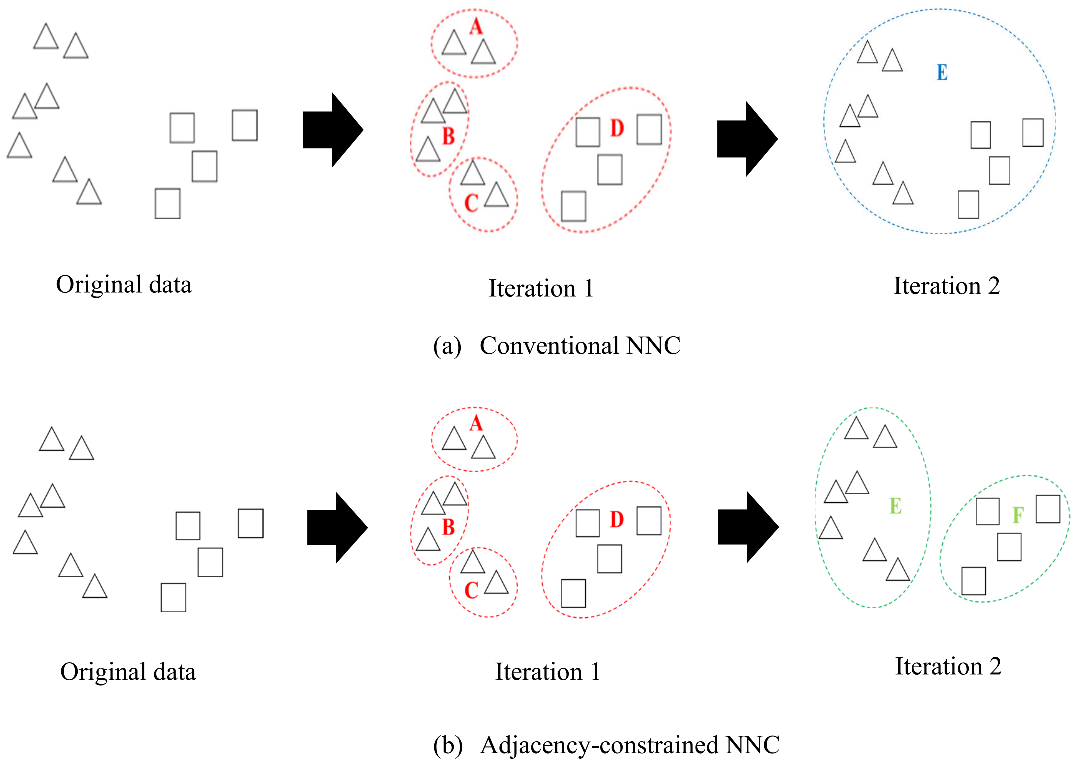


Figure 4.1. a simple example of the traditional NNC merging process (a), and the CNNC procedure (b). We use dotted lines to denote clusters (i.e., A, B, C, etc.), and rectangles or triangles to represent data samples. We can see that, in iteration1, the generated clusters from conventional NNC and adjacency-constrained NNC are the same. Initially each sample is regarded as a cluster, and they all have the same mass (i.e., 1); i.e., they all satisfy the mass requirement in Eq. (4.4), so if the neighbor relationship is satisfied, a connection can be generated between clusters. In iteration 2, for conventional NNC, cluster A, cluster B, cluster C and cluster D are chosen to merge into one big cluster, E. Because cluster A and cluster B are the nearest neighbor of each other, the nearest neighbor of cluster C is cluster B, and the nearest neighbor of cluster D is cluster C. However, in the CNNC procedure, cluster C and cluster D are not chosen to merge because mass(D)>mass(C). Cluster A and cluster B, and cluster B and cluster C are both chosen to merge because mass(A)≤mass(B) and mass(C)≤mass(B).

4.2.2 Adjacency-constrained nearest neighbor clustering (CNNC)*

Recently, nearest neighbor clustering (NNC) has become a research focal point [13], [59], [211]. NNC is a kind of hierarchy-based clustering. Compared with traditional

* Since this thesis is organized by the compilation of papers (Chapters) and each paper (Chapter) has a different focus, the partial clustering mechanism of TC in Chapter 3 is expressed as CNNC here.

hierarchical clustering, such as average-link or ward-link, NNC has lower computational complexity (i.e.,$O(nlogn)$) and can achieve better clustering performance. In addition, NNC can provide natural partitions at different granularity levels to meet the requirements for different clustering resolutions in application scenarios. Existing NNC approaches, on the other hand, are entirely based on the statistic of nearest neighbor, i.e., the merging is done as long as the neighbor relationship is satisfied. Data samples from different classes may be also merged in this fashion, lowering clustering accuracy. In this study, we introduce a parameter-free adjacency-constrained nearest neighbor clustering (CNNC) algorithm, which exploits the clusters with relatively large mass to guide the merging process, preventing trivial wrong merging in conventional NNC approaches. The difference between the traditional NNC method and the CNNC method is shown in Fig. 4.1.

Given a single-view data $X^{(i)}$, initially, each sample is its own cluster. Given the number of samples contained in a cluster as the mass of the cluster, therefore, in the beginning, the mass of each cluster equals 1. Then, the following rule is applied to form connections between clusters:

$$\zeta_j \to \zeta_j^N, if\ mass(\zeta_j) \leq mass(\zeta_j^N) \tag{4.4}$$

where ζ_j denotes the j-th cluster, ζ_j^N denotes the 1-nearest cluster of ζ_j. Much research has been conducted to define the distance between two clusters. Here, we simply leverage the minimum distance from any sample in one cluster to any sample in the other cluster as the distance between the two clusters, which is similar to the single-linkage method. $mass(\zeta_j)$ represents the mass of ζ_j (i.e, the number of samples ζ_j contains). Similarly, $mass(\zeta_j^N)$ is the mass of ζ_j^N. The symbol $" \to "$ denotes a connection (i.e, merger) C_j between ζ_j and ζ_j^N.

This process can be also defined in a graph *G*,

$$A(\zeta_j, \zeta_j^N) = \begin{cases} 1,\ if\ mass(\zeta_j) \leq mass(\zeta_j^N) \\ 0,\ otherwise \end{cases} \tag{4.5}$$

where *A* stands for the adjacency matrix of *G*. Then, new clusters can be obtained by calculating the connected components of the adjacency matrix *A*. At this point, one iteration has been completed. By repeating this merger process according to Eq. (4.4), all clusters will eventually merge into one cluster and form a hierarchical tree. Each layer of the hierarchical tree can be regarded as a partition under a specific granularity.

Each connection (i.e., merger) C_j has two intuitive properties. One of the properties is the product of the mass of the two clusters it connects

$$M_j = mass(\zeta_j) \times mass(\zeta_j^N) \tag{4.6}$$

The other is the square of the distance between the two clusters it connects

$$S_j = d^2(\zeta_j, \zeta_j^N) \tag{4.7}$$

Plotting all the connections on a two-dimensional graph of the two properties, called the decision graph. By observing the decision graph and finding the connections with relatively large M_j and S_j, remove these connections to get the final reasonable partition.

CNNC is parameter-free. A reasonable partition can be obtained through a certain layer (granularity) of the clustering tree, or it can be obtained by observing the decision graph and removing the connections with relatively large M_j and S_j. However, CNNC can also be assigned the desired number of clusters *K*. After simply removing *K*-1 connections with relatively large $M_j \times S_j$, then we can get a partition containing *K* clusters. On the other hand, in each iteration, CNNC only needs to find the nearest neighbor of each cluster. An efficient method for obtaining nearest neighbors is through the use of approximate nearest neighbor search techniques, such as locality-sensitive hashing. Therefore, the complexity of the algorithm can be reduced to $O(nlogn)$. Compared with traditional hierarchical clustering algorithms, CNNC has a lower computational overhead.

Exploiting CNNC to perform clustering based on each fusion distance matrix in FDEW, then *v*+1 partitions $P^{(r)}$ can be obtained, where $\{P^{(r)}\}_{r=1}^{v+1} = \{P^{(1)}, P^{(2)}, \dots, P^{(v)}, P^{(v+1)}\}$. So which partition is the best? This requires an evaluation index to evaluate the clustering quality of each partition.

4.2.3 Internal evaluation index based on Rawls' max-min criterion (MMI)

In practice, the ground-truth labels are often not known in advance. Therefore, we cannot objectively judge which partition is the best. A simple idea is to use internal evaluation indices to evaluate each partition to find the best one. Most of the past internal evaluation indices need to know cluster centers of partition [212]. However, CNNC does not output specific cluster centers. Here we propose a new internal clustering evaluation index based on distance matrix to select the best partition based on Rawls' max-min criterion [207], which is called Max-Min Index (MMI).

For a partition $P^{(r)}$, 1) we arbitrarily select two clusters, and then arbitrarily select a sample from each cluster, and use the distance between the two samples as the inter-class distance; 2) We again, choose a cluster arbitrarily, and use the average of the distance between any two samples in this cluster as the intra-class distance. Based on 1) and 2), we first define an initial evaluation index:$\forall x_a^{(r)} \in \forall\zeta_k^{(r)}, \forall x_b^{(r)} \in \forall\zeta_l^{(r)};\ \forall x_c^{(r)}, x_d^{(r)} \in \forall\zeta_m^{(r)}$,

$$I^{(r)} = \frac{d\left(x_a^{(r)},x_b^{(r)}\right)}{\frac{2}{|\zeta_m^{(r)}|\left(|\zeta_m^{(r)}|-1\right)}\sum d\left(x_c^{(r)},x_d^{(r)}\right)} \tag{4.8}$$

The larger $I^{(r)}$, $P^{(r)}$ may have a larger inter-class distance and a smaller intra-class distance, but it is not certain. This is because we randomly select clusters and samples when calculating $I^{(r)}$, which may not be representative. According to Rawls' max-min criterion, the right decision is that which maximizes the minimum outcome. Inspired by the max-min criterion, we first calculate the minimum value of $I^{(r)}$:

$$min\left(I^{(r)}\right) = min\left\{\frac{d\left(x_a^{(r)},x_b^{(r)}\right)}{\frac{2}{\left|\zeta_m^{(r)}\right|\left(\left|\zeta_m^{(r)}\right|-1\right)}\sum d\left(x_c^{(r)},x_d^{(r)}\right)}\right\} = \frac{\min\limits_{k}\min\limits_{l}\min\limits_{x_a^{(r)}\in\zeta_k^{(r)},x_b^{(r)}\in\zeta_l^{(r)}} d\left(x_a^{(r)},x_b^{(r)}\right)}{\max\limits_{m}\frac{2}{\left|\zeta_m^{(r)}\right|\left(\left|\zeta_m^{(r)}\right|-1\right)}\sum_{x_c^{(r)},x_d^{(r)}\in\zeta_m^{(r)}} d\left(x_c^{(r)},x_d^{(r)}\right)} \tag{4.9}$$

Furthermore, we believe that $P^{(r)}$ that maximizes $min\left(I^{(r)}\right)$ is the best, that is

$$s = \underset{r}{\operatorname{argmax}}\, min\left(I^{(r)}\right) = \ arg\max_{r}\frac{\min\limits_{k}\min\limits_{l}\min\limits_{x_a^{(r)}\in\zeta_k^{(r)},x_b^{(r)}\in\zeta_l^{(r)}} d\left(x_a^{(r)},x_b^{(r)}\right)}{\max\limits_{m}\frac{2}{\left|\zeta_m^{(r)}\right|\left(\left|\zeta_m^{(r)}\right|-1\right)}\sum_{x_c^{(r)},x_d^{(r)}\in\zeta_m^{(r)}} d\left(x_c^{(r)},x_d^{(r)}\right)} \tag{4.10}$$

Therefore, we can finally determine that $P^{(s)}$ is the best partition by Eq. (4.10). Compared with other distance matrix-based internal indices, the proposed MMI is more accurate in selecting the best partition (see Table 4.10). The method proposed in [213] can be utilized to reduce the computational complexity of MMI.

4.2.4 Algorithm of MCHC and MCHC-PF

For Multi-view data $\{X^{(i)}\}_{i=1}^{v}$, we first calculate $\{FDEW^{(r)}\}_{r=1}^{v+1}$ according to Eqs. (4.1)-(4.3). Then we use CNNC to perform clustering based on each $FDEW^{(r)}$, and get $\{P^{(r)}\}_{r=1}^{v+1}$. Finally, we select the best partition $P^{(s)}$ in $\{P^{(r)}\}_{r=1}^{v+1}$ according to Eqs. (4.8)-(4.10). Fig. 4.2 shows the simple flowchart of MCHC, and Algorithm 4.1 shows the pseudo code of MCHC.

In real life, the correct number of clusters is often not known in advance. Therefore, we provide a parameter-free version of MCHC (MCHC-PF). Algorithm 4.2 gives the pseudo code of MCHC-PF. MCHC-PF does not require any parameters, it can provide several partitions at different granularity levels, and draw a decision graph according to Eqs. (4.6)-(4.7) for users to estimate a reasonable number of clusters. On the other hand, MCHC-PF only uses CNNC to perform clustering based on the fusion distance matrix with equal weights D^* from FDEW, so it has a shorter runtime than MCHC. Fig. 4.3 shows the decision graph of MCHC-PF on the data set UCI-digits (this data set will be introduced in the experimental

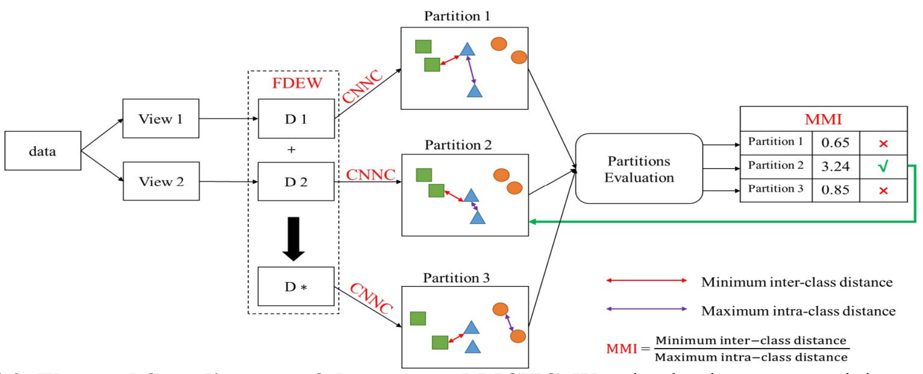


Figure 4.2. The workflow diagram of the proposed MCHC. We take the data set containing two views as an example. First, we calculate the distance matrix of each view by Eq. (4.3) to get D1 and D2. Then we calculate $D*$ by Eq. (4.2). D1, D2, and $D*$ together form Fusion Distance matrices with Extreme Weights (FDEW). Next, based on each distance matrix in FDEW, we exploit adjacency-Constrained Nearest Neighbor Clustering (CNNC) to obtain three partitions (i.e., Partition 1, Partition 2, and Partition 3). Finally, we choose the most reasonable partition (i.e., Partition 2) based on MMI (i.e., Eqs. (4.8)-(4.10)).

part). It can be clearly seen that there are 9 connections (mergers) with larger M_j and S_j. Remove them in the adjacency matrix, and we can get the correct 10 clusters, which matches the ground truth.

There are three main differences between MCHC and MCHC-PF. First, MCHC not only uses complementary information among multiple views, but exploits the information from each single view. However, MCHC-PF only exploits the complementary information among multiple views. Second, MCHC uses naive way to calculate distance matrix for each view, while MCHC-PF uses k-d tree to approximate the calculation to obtain a sparse distance matrix. Third, like most existing multi-view clustering methods, MCHC needs to be set target number of clusters. However, MCHC-PF does not need to set this parameter. It provides multiple clustering results at different granularity levels for users to choose according to specific scenarios.

Now we analyze the complexity of MCHC and MCHC-PF. We first analyze the complexity of MCHC. According to Algorithm 4.1, Steps 3-7 costs $O(vn^2)$, where v is the number of views. When the distance matrix is known, the cost of CNNC is $O(n)$. Therefore, the cost of Steps 8-19 is $O((v+1)n)$. Steps 20-23 costs $O((v+1)K^2)$ for MMI calculation and best partition finding, where K is the target number of clusters. In summary, the total cost of MCHC is $O(vn^2)+O\big((v+1)n\big)+O((v+1)K^2)$, approximately $O(n^2)$. Compared to $O(n^3)$ of most multi-view spectral clustering or subspace clustering methods, the complexity of MCHC is acceptable. For MCHC-PF, we leverage the k-d tree to compute the sparse distance matrix for each view, so Steps 3-6 costs $O(vnlogn)$. Because MCHC-PF only runs CNNC on the fusion distance matrix with equal weights D^*, the cost of Steps 7-16 is approximately $O(n)$. Therefore, the total cost of MCHC-PF is $O(nlogn)$.

Algorithm 4.1: Algorithm of the proposed MCHC

```
1  Input: Multi-view data $\{X^{(i)}\}_{i=1}^{v}$ and the target number of clusters $K$.
2  Output: Best partition $P^{(s)}$.
3  for $i$=1:$v$ do
4      Calculating distance matrix $D^{(i)}$ by Eq. (4.3).
5  end
6  Calculating $D^*$ by Eq. (4.2).
7  Combine $D^{(i)}$ and $D^*$ to get FDEW.
8  for $r$=1:$v$+1 do
9  Initializing adjacency matrix $A$.
10 Constructing cluster sets $\{\zeta_j\}$ (Initially, regard each sample as a cluster).
11 while cluster sets $\{\zeta_j\}$ have more than two clusters do
12  Searching the nearest cluster of $\zeta_j$ with higher mass according to $FDEW^{(r)}$.
13 Updating $A$ by Eqs. (4.4)-(4.5) (Using two nearest samples respectively from two clusters to represent these two clusters).
14 Calculating $M_j$ and $S_j$ of $C_j$ by Eqs. (4.6)-(4.7).
15 Updating cluster sets $\{\zeta_j\}$ based on $A$.
16 end
17 Updating $A$ by removing $K$-1 $C_j$ with largest $M_j \times S_j$.
18 Getting partition $P^{(r)}$ based on $A$.
19 end
20 for $r$=1:$v$+1 do
21     Calculating $min(I^{(r)})$ by Eqs. (4.8)-(4.9).
22 end
23 Finding best partition $P^{(s)}$ by Eq. (4.10).
```

Algorithm 4.2: Algorithm of the proposed MCHC-PF

```
1  **Input:** Multi-view data $\{X^{(i)}\}_{i=1}^{v}$.
2  **Output:** partitions at different granularity levels $\{R_t\}$.
3  **for** $i$=1:$v$ **do**
4      Calculating sparse distance matrix $D^{(i)}$ by Eq. (4.3).
5  **end**
6  Calculating $D^*$ by Eq. (4.2).
7  Initializing adjacency matrix $A$.
8  Constructing cluster sets $\{\zeta_j\}$ (Initially, regard each sample as a cluster).
9  **while** cluster sets $\{\zeta_j\}$ have more than two clusters **do**
10     Searching the nearest cluster of $\zeta_j$ with higher mass according to $D^*$.
11     Updating $A$ by Eqs. (4.4)-(4.5) (Using two nearest samples respectively from two clusters to represent these two clusters).
12     Getting partition $R_t$ at current granularity level based on $A$.
13     Calculating $M_j$ and $S_j$ of $C_j$ by Eqs. (4.6)-(4.7).
14     Updating cluster sets $\{\zeta_j\}$ based on $A$.
15 **end**
16 Plotting decision graph by $M_j$ and $S_j$.
```

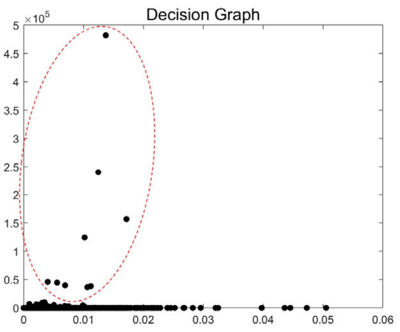


Figure 4.3. Decision graph of MCHC-PF on the UCI-digits data set. The horizontal axis represents the property M_j of each connection in the CNNC, and the vertical axis represents the property S_j of each connection. The nine connections in the red circle have relatively large M_j and S_j. Remove them to leave 10 clusters, which exactly matches the ground-truth.

Table 4.1. Statistics of multi-view data sets.

Data sets	#Views	#Samples	#Clusters
100-leaves	3	1600	100
UCI-digits	3	2000	10
COIL20	3	1440	20
Handwritten	2	2000	10
ORL	3	400	40
UMIST	3	575	20
CMU-PIE	3	2856	68
COIL100	3	7200	100

4.3 Experiments and results

In this part, we conducted several experiments to show the superiority of MCHC and MCHC-PF.

4.3.1 Data sets description

(1) 100-leaves: There are 1600 samples in the 100-leaves data set, divided into 100 categories. The original 100-leaves photos are also different in size. There are three views which display samples from several angles using shape descriptors, fine scale margins, and texture histogram characteristics [229].

(2) UCI-digits: The UCI-digit data set can be found in the UCI repository (https://archive.ics.uci.edu/ml/index.php). The digits (0–9) in this collection were extracted from Dutch utility maps in 2000 samples. Each class contains 200 samples, each of which is represented by six feature sets. We employed three feature sets following [214]: 76 character shape Fourier coefficients, 216 profile correlations, and 64 Karhunen-Loève coefficients.

(3) COIL20: This data set has 1440 grayscale photos of 20 different objects [160]. Each image is downscaled to 32 by 32 pixels for the original features scenario. Three types of features are extracted in the case of several hand-crafted features: Intensity, LBP, and Gabor. The sizes of their features are 1024, 3304 and 6750, respectively [217].

(4) Handwritten: The data set comprises 2000 instances of handwritten digits ranging from 0 to 9, each of which is represented by two distinct views. The first view is a feature vector with 240 elements, calculated as the average of pixels in 2×3 windows, while the second view is a Fourier coefficient vector with 76 elements [215].

(5) ORL: This is made up of 400 photos of 40 people's faces. Following [47], each image is down-sampled to 32 by 32 pixels for the original features scenario. Each image in the handcrafted features scenario is represented by three types of features.

(6) UMIST: This collection [158] contains 564 photos of 20 people (mixed race, gender, and appearance). Each person is depicted in a variety of poses, from profile to frontal perspectives. Each image has a resolution of about 220×220 pixels and a 256-bit greyscale. Following [216], each image is represented by three heterogeneous feature sets:30 isometric projection

(ISO), 30 principal component analysis (PCA), and 30 neighborhood preserving embedding (NPE).

(7) CMU-PIE: This data set [148] contains 2856 frontal-face photos of 68 persons, with 42 distinct illuminations for each object. Each photograph was cropped to a size of 32×32 pixels. Three feature sets are used to express each image: 30 ISO, 30 PCA, and 30 NPE. Fig. 4.4 shows some sample face images from the CMU-PIE database.

(8) COIL-100: This data set [147] is a library of 7200 color images representing 100 different types of objects. Each image is 128×128 pixels in size. Each object has 72 distinct photos in various positions. Each image is expressed using three feature sets: 30 ISO, 30 PCA, and 30 NPE. The full statistics of these data sets are shown in Table 4.1.

4.3.2 Compared algorithms

We compared MCHC and MCHC-PF with 10 state-of-the-art multi-view clustering algorithms. They include: K-means; Graph-based multi-view clustering (GMC) [92]; Unified graph learning for multi-view clustering (UGLMC) [89]; View variation and view heredity clustering (V3H) [217]; Affinity aggregation for spectral clustering (AASC) [43]; Multi-view clustering via adaptively weighted Procrustes (AWP) [91]; Co-regularized multi-view spectral clustering (CoReg) [44]; Multi-view consensus graph clustering (MCGC) [218]; Robust multi-view spectral clustering (RMSC) [219]; and Weighted multi-view spectral clustering (WMSC) [46]. We employed three widely used external clustering validation indices to evaluate the efficacy of clustering algorithms: Accuracy (ACC), Normalized mutual information (NMI) [133], and F-score [220]. The best and second-best clustering results were highlighted and underlined respectively. We presented the optimal clustering outcomes of multiple views for K-means, which is a clustering algorithm that operates on a single view of the data. For other multi-view clustering algorithms, the parameters were tuned according to the suggestions of the original papers to obtain the best results. All experiments were conducted on a workstation with two 14-core Intel Xeon 6132 CPUs clocked at 2.6

GHz and 3.7GHz and 96GB memory.

4.3.3 Results and analysis

Tables 4.2, 4.3, and 4.4 show the clustering results and Fig. 4.5 give the average rankings for all multi-view clustering methods on all data sets. In essence, the proposed MCHC outperforms all other clustering methods. Whether compared with single-view clustering algorithms or multi-view clustering algorithms, MCHC shows unparalleled performance advantages. In particular, for the metric ACC, the results of our MCHC were about 4.8%, 6.3%, 26.4%, 22.5%, and 12.5% better than the second-best (except for MCHC-PF) clustering results on Handwritten, ORL, UMIST, CMU-PIE and COIL100 data set, respectively. For the metric NMI, the results of our MCHC were about 5.1%, 4.5%, 8.9%, 8.6%, and 4.5% better than the second-best clustering results on Handwritten, ORL, UMIST, CMU-PIE and COIL100 data set, respectively. Finally, for the metric F-score, the results of our MCHC were about 6.9%, 7.2%, 7.7%, 28.1%, 24.5%, and 15.9% better than the second-best clustering results on COIL20, Handwritten, ORL, UMIST, CMU-PIE and COIL100 data set, respectively. Besides, in terms of the rank in Fig. 4.5, the next-best algorithm, CoReg, was three times plus higher than MCHC in its ranking scores among ACC, NMI, and F-score.

Table 4.2. Clustering results of MCHC and other methods in the metric of ACC.

Sources	Methods	100-leaves	UCI-digits	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
-	K-means	.5780	.6814	.6410	.6921	.5703	.4617	.5377	.5737
TKDE-20	GMC	.8238	.8495	.7910	.8300	.6325	.5217	.7048	.7692
ICDM-19	UGLMC	.8001	.8825	.9014	.7425	.6900	.6043	.1863	.7267
TAI-21	V3H	.8219	.9078	.6005	.8670	.7478	.5245	.7283	.6514
CVPR-12	AASC	.8779	.8505	.7806	.8334	.7352	.4428	.5396	.6564
KDD-18	AWP	.7800	.8670	.7708	.9315	.6975	.5461	.7749	.7029
NeurIPS-11	CoReg	.8421	.9570	.8280	.9111	.7880	.5294	.7382	.7839
TIP-18	MCGC	.6075	.4920	.3882	.1005	.5950	.4487	.7006	.5194
AAAI-14	RMSC	.7313	.2474	.4092	.4098	.7977	.4753	.7465	.2216
AAAI-18	WMSC	.8789	.8410	.8463	.8335	.8068	.4897	.6633	.7142
-	**MCHC-PF**	.6931	.6820	.8236	.8805	.8100	.6557	.8883	.8385
-	**MCHC**	**.8888**	**.9655**	**.9250**	**.9795**	**.8700**	**.8678**	**1**	**.9087**

Table 4.3. Clustering results of MCHC and other methods in the metric of NMI.

Sources	Methods	100-leaves	UCI-digits	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
-	K-means	.7996	.7025	.8004	.7071	.7784	.6771	.7990	.8239
TKDE-20	GMC	.9296	.9013	.9410	.8767	.8590	.7373	.8892	.9371
ICDM-19	UGLMC	.9196	.9231	.9705	.8505	.8630	.8373	.3919	.9309
TAI-21	V3H	.9099	.8145	.7663	.7425	.8632	.6833	.8666	.8656
CVPR-12	AASC	**.9590**	.9025	.8958	.8827	.8538	.6619	.7921	.8676

KDD-18	AWP	.9013	.8949	.9264	.9026	.8584	.7203	.9140	.9163
NeurIPS-11	CoReg	.9325	.9197	.9425	.8811	.8905	.7412	.8829	.9264
TIP-18	MCGC	.7606	.7395	.6515	.0350	.8035	.6716	.8070	.7841
AAAI-14	RMSC	.8828	.3425	.7155	.4814	.8896	.6673	.8432	.5092
AAAI-18	WMSC	.9527	.8839	.9484	.8772	.8950	.7047	.8610	.9026
-	**MCHC-PF**	.9146	.8407	.9452	.9214	.9228	.8486	.9666	.9663
-	**MCHC**	.9466	**.9276**	**.9826**	**.9535**	**.9404**	**.9262**	**1**	**.9816**

Table 4.4. Clustering results of MCHC and other methods in the metric of F-score.

Sources	Methods	100-leaves	UCI-digits	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
-	K-means	.4662	.6331	.6125	.6356	.4547	.4078	.4744	.5164
TKDE-20	GMC	.5042	.8426	.7943	.8113	.3599	.4620	.6171	.7195
ICDM-19	UGLMC	.7501	.8709	.8696	.7547	.5951	.5701	.0332	.5252
TAI-21	V3H	.7469	.8276	.5622	.7613	.6591	.4403	.6178	.5902
CVPR-12	AASC	.7783	.8440	.7758	.8167	.5948	.3656	.3766	.4795
KDD-18	AWP	.7129	.8455	.7187	.8875	.6163	.4891	.7555	.6985
NeurIPS-11	CoReg	.7907	.9171	.8088	.8588	.7091	.4778	.6790	.7583
TIP-18	MCGC	.0991	.5294	.1992	.1810	.2588	.3189	.2694	.1390
AAAI-14	RMSC	.5129	.2158	.3194	.2881	.7178	.2846	.3751	.0465
AAAI-18	WMSC	**.8467**	.8315	.8331	.8187	.7283	.4177	.6079	.6916
-	**MCHC-PF**	.6228	.6718	.8311	.8715	.7560	.6461	.8829	.8605
-	**MCHC**	.8212	**.9323**	**.9389**	**.9591**	**.8051**	**.8509**	**1**	**.9174**

Figure 4.4. Sample face images from the CMU-PIE database. MCHC achieved 100% accuracy on this data set.

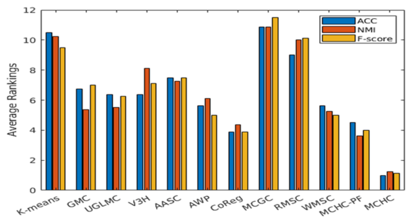


Figure 4.5. Average rankings for all multi-view clustering methods on all data sets.

Unlike the other 11 multi-view clustering methods (including MCHC) that require at least the ground-truth number of clusters to be set, MCHC-PF can give natural partitions at different granularity levels without any parameters. Table 4.5 shows the number of clusters obtained at different granularity levels. For most data sets, the clustering results obtained by

MCHC-PF can yield a relatively accurate number of clusters. The clustering results of MCHC-PF in Tables 4.2, 4.3, and 4.4 are based on the number of clusters that are closest to the ground truth. The clustering performance of MCHC-PF was worse than that of MCHC, because MCHC-PF only considers the complementary information (i.e., D^*) from multiple views, not the information from each single view. However, as Fig. 4.5 shows, compared with all other methods, MCHC-PF still achieved competitive results. Particularly, on the five data sets (i.e., Handwritten, ORL, UMIST, CMU-PIE and COIL100), regarding the NMI index, the results of MCHC-PF was better than that of other 10 state-of-the-art multi-view clustering algorithms.

From a theoretical point of view, the reasons why the performance of most multi-view spectral or subspace clustering methods is not competitive to MCHC are mainly due to the following two aspects. First, the backbones of these methods are spectral clustering or subspace clustering, which have inherent limitations. For example, it is hard for spectral clustering to accurately capture the intrinsic manifold structure in data when constructing the k-nearest neighbor similarity graph. However, CNNC in MCHC can catch it more accurately due to its constrained way of merging. Second, when conflicting views exist in multi-view data, performing clustering on the information from a specific single view may achieve better results than that on the complementary information from multiple views. Most compared methods only consider the consensus information from multi-view data. However, MCHC not only uses complementary information among multiple views but exploits the information from each single view, achieving better results.

4.3.4 Runtime

To demonstrate the efficiency of the proposed methods, we compared the runtime of the proposed MCHC and MCHC-PF to three algorithms, including AWP, CoReg, and V3H, from the 10 state-of-the-art multi-view clustering algorithms. Among them, AWP and CoReg are the two best-compared ones according to the ACC rankings, and V3H is the latest one.

According to Table 4.6, the proposed MCHC-PF can produce clustering results in roughly 20 seconds for all data sets. The total runtime of MCHC-PF on all data sets was the lowest. The runtime of MCHC was significantly more than that of MCHC-PF, because MCHC needs to run the CNNC algorithm multiple times and once needed for MCHC-PF. However, the clustering performance of MCHC was better than that of all other clustering algorithms (see Tables 4.2-4.4).

Table 4.5. Clustering results of MCHC-PF at different granularity levels. #C means the ground-truth number of clusters, and NC means the number of clusters.

Data sets	#C	NC at different granularity levels	Closest NC
100-leaves	100	{391, 155, 76, 33, 14, 5, 1}	76
UCI-digits	10	{429, 134, 45, 17, 7, 3, 1}	7
COIL20	20	{414, 181, 88, 47, 24, 13, 5, 3, 1}	24
Handwritten	10	{397, 125, 41, 17, 9, 3, 2, 1}	9
ORL	40	{113, 44, 17, 6, 3, 2, 1}	44
UMIST	20	{184, 84, 42, 21, 11, 7, 4, 1}	21
CMU-PIE	68	{972, 397, 154, 78, 33, 12, 4, 3, 1}	78
COIL100	100	{2148, 924, 469, 251, 127, 52, 14, 6, 2, 1}	127

Table 4.6. Runtime (in seconds) comparison with three representative compared algorithms. The lowest runtimes were marked in bold.

Data sets	AWP	CoReg	V3H	**MCHC**	**MCHC-PF**
100-leaves	0.87	14.60	687.10	2.19	**0.60**
UCI-digits	**0.59**	11.20	876.83	2.99	0.92
COIL20	**2.47**	10.26	412.52	4.16	2.79
Handwritten	**0.40**	7.79	617.86	2.51	0.75
ORL	0.18	2.69	155.72	0.38	**0.12**
UMIST	0.27	3.72	22.67	0.53	**0.15**
CMU-PIE	**2.63**	92.72	1568.32	12.14	3.07
COIL100	26.77	1276.28	14584.9	84.74	**20.35**

4.4 Ablation study

4.4.1 Impact of fusion distance matrices with extreme weights (FDEW)

According to Eqs. (4.1)-(4.3), FDEW not only uses the fusion distance matrix with equal weights (i.e., D^*) containing complementary information from multiple views, but exploits the distance matrix $D^{(i)}$ of each view, which includes the information from each single view. In this section, we explored the results of CNNC on each distance matrix in FDEW to show the necessity of including these two pieces of information. As Table 4.7 shows, partitions 1-

3 denotes the results of CNNC on the distance matrix $D^{(i)}$ of each single view, and partition$*$ means the results of CNNC on the fusion distance matrix with equal weights (i.e., D^{*}). Here, we exploited NMI to evaluate the performance of each partition. The best and MMI-selected clustering results were highlighted and underlined respectively. On the one hand, the best clustering results were from partitions 1-3 on some data sets (e.g., COIL20 and ORL), or from partition$*$ on some data sets (e.g., 100-leaves and UCI-digits). Therefore, it is necessary to consider both the information from each single view and the complementary information from multiple views. On the other hand, MMI can accurately select the best one from several partitions.

Table 4.7. the results of CNNC on each distance matrix in FDEW using the metric of NMI. The best and MMI-selected clustering results are highlighted and underlined respectively.

Data sets	Partition1	Partition2	Partition3	Partition$*$
100-leaves	.8020	.6550	.7577	**.9466**
UCI-digits	.7711	.7697	.9163	**.9276**
COIL20	.9555	**.9826**	.9769	.9662
Handwritten	.9186	.7697	-	**.9535**
ORL	.8175	**.9404**	.8912	.9309
UMIST	.7677	**.9262**	.8102	.8854
CMU-PIE	.8946	.9432	**1**	.9801
COIL100	.8913	**.9816**	.9159	.9782

4.4.2 Impact of adjacency-constrained nearest neighbor clustering (CNNC)

To show the superiority of CNNC in the proposed model, on the one hand, we explored CNNC's clustering performance advantage by comparing it with other state-of-the-art single-view hierarchical clustering methods or nearest neighbor clustering (NNC) methods. Seven well-known hierarchical clustering methods were used including single-linkage, complete-linkage, average-linkage, ward-linkage, centroid-linkage, median-linkage, weighted-linkage. Three recent NNC methods, GDL [62], SNNDPC [76] and Finch [13], were used for comparison. For each of the eight multi-view data sets described above, we concatenated all the features of multiple views, and then performed CNNC and other baselines directly on the concatenation. As Table 4.8 shows, compared with other state-of-the-art single-view clustering methods, CNNC achieved the best results on all data sets.

On the other hand, in the MCHC framework, we replaced CNNC with other hierarchical clustering or NNC methods and kept other components in the framework unchanged. The

Table 4.8. Performance comparison for CNNC and other hierarchical clustering or NNC methods, in the metric of NMI.

Methods	100-leaves	UCI-digits	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
Single-linkage	.5343	.0348	.9708	.0370	.4951	.6456	.7586	.8847
Complete-linkage	.8159	.4727	.7693	.5383	.7221	.5836	.4708	.7753
Average-linkage	.8477	.4958	.7745	.6046	.7221	.6461	.4634	.7899
Ward-linkage	.8634	.6204	.8664	.7022	.7889	.6228	.5563	.8242
Centroid-linkage	.8234	.5387	.6454	.4013	.5784	.5788	.3790	.7727
Median-linkage	.8054	.4286	.6713	.1106	.6446	.5998	.4595	.7918
Weighted-linkage	.8318	.5071	.7784	.5510	.7324	.6276	.5324	.8033
GDL	.8890	.7623	.9318	.8871	.7769	.7488	.7916	.9442
SNNDPC	.6875	.6733	.8035	.8200	.6363	.6570	.4243	.6986
Finch	.8811	.6002	.8377	.8111	.5729	.6252	.3959	.8435
CNNC	**.9072**	**.7715**	**.9715**	**.9186**	**.8181**	**.9221**	**.9455**	**.9816**

Table 4.9. Performance comparison with other multi-view hierarchical clustering methods, in the metric of NMI. The COIL100 data set causes errors in the author's GDL code.

Methods	100-leaves	UCI-digits	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
MHC-single	.6215	.0348	.9708	.0348	.6742	.6693	.9383	.8847
MHC-complete	.8731	.5666	.7504	.6289	.8245	.5842	.6827	.7724
MHC-average	.8885	.5878	.7854	.5878	.8611	.6524	.7171	.8506
MHC-ward	.9219	.7041	.8954	.7041	.8937	.6276	.7335	.8319
MHC-centroid	.8645	.5549	.5019	.0347	.3486	.5148	.7093	.7727
MHC-median	.8618	.3585	.6980	.3015	.4279	.5633	.6977	.7905
MHC-weighted	.8774	.6596	.8009	.6693	.8306	.6663	.7118	.8067
MHC-GDL	.9325	.9010	.9374	.9141	.8983	.7573	.8518	-
MCHC	**.9466**	**.9276**	**.9826**	**.9535**	**.9404**	**.9262**	**1**	**.9816**

Table 4.10. Performance comparison with other distance matrix-based internal indices, in the metric of NMI.

Data sets	Best partition	DI	Sil	CVNN	CVDD	MMI
100-leaves	0.9466	**0.9466**	**0.9466**	.8020	**.9466**	**0.9466**
UCI-digits	0.9276	**0.9276**	0.7711	.7711	.7711	**0.9276**
COIL20	0.9826	**0.9826**	0.9769	.9769	.9662	**0.9826**
Handwritten	0.9535	**0.9535**	**0.9535**	.7697	**.9535**	**0.9535**
ORL	0.9404	**0.9404**	0.8912	.8175	.8175	**0.9404**
UMIST	0.9262	0.7677	0.8102	**.9262**	**.9262**	**0.9262**
CMU-PIE	1	0.8946	**1**	**1**	**1**	**1**
COIL100	0.9816	**0.9816**	0.8913	.8913	**.9816**	**0.9816**

generated new multi-view hierarchical clustering (MHC) methods were named as MHC-single, MHC-complete, MHC-average, MHC-ward, MHC-centroid, MHC-median, MHC-weighted and MHC-GDL, respectively. MHC-SNNDPC and MHC-Finch were removed, because SNNDPC and Finch need to know the coordinates of data points, and the coordinates of data points corresponding to D^* are unknown. We also performed these alternative ones on the eight multi-view data sets mentioned above. According to Table 4.9, MCHC still achieved the best performance on all data sets.

From the above two experiments, CNNC has better performance than previous NNC methods or hierarchical clustering methods, whether processing single-view data or multi-view data. This is because the constrained merging way of CNNC can more accurately capture the manifold structure in the data.

4.4.3 Impact of internal evaluation index based on Rawls' max-min criterion (MMI)

To show the validity of the Internal evaluation index based on Rawls' max-min criterion (MMI), we exploited four other distance matrix-based internal indices, including Dunn Index (DI) [221], Silhouette index (Sil) [222], Clustering Validation index based on Nearest Neighbors (CVNN) [223] and Clustering Validity index based on Density-involved Distance (CVDD) [224] to select the best partition in the MCHC framework, and kept other components in the framework unchanged. DI and Sil are two classic internal validity indices, and CVNN and CVDD are the recent ones. After that, the metric NMI was used to objectively evaluate the selected partition based on the ground-truth labels. According to Table 4.10, the proposed MMI can select the best partition on all data sets, whereas the other four distance matrix-based internal indices cannot.

From a theoretical point of view, the other four distance matrix-based internal indices all have some inherent flaws. For example, DI exploits the distance between the two farthest points in cluster as the intra-class distance, which is obviously susceptible to outliers. Additionally, Sil does not adapt well to non-spherical data sets, and CVDD is susceptible to changing densities in clusters.

4.5 Discussion

The proposed MCHC consists of three main components: including FDEW, CNNC, and MMI. Here, we will explain why FDEW, CNNC, and MMI are combined into MCHC, that is, the theoretical significance of the combination of these three components. First, this is determined by the generalized paradigm of multi-view clustering. Almost all multi-view clustering methods first learn complementary (or consensus) information from multi-view data, and then use a single-view clustering method for post-processing of the complementary information. The proposed MCHC framework follows this paradigm. Second, another significance for combining these three components is to inherit their respective advantages. For example, FDEW alleviates the poor impact of conflicting views; CNNC can capture the manifolds in data and improve clustering accuracy compared to traditional NNC methods; MMI can select the best one from several partitions in an unsupervised manner, no additional manual intervention is required.

On the other hand, most previous multi-view clustering methods focus on exploring the different importance of each view to learn an optimal clustering-friendly representation. However, they ignore the optimization for the clustering mechanism and are only based on the existing backbone of spectral clustering or subspace clustering. Instead, this study focuses on the optimization of the clustering mechanism. Even based on representations with extreme weights (i.e., FDEW), the proposed frameworks still achieved state-of-the-art performance, which provides new thinking for multi-view clustering. Besides, according to Table 4.7, we can see that for five out of the eight data sets, the best clustering results were obtained by one single view. This confirms the viewpoint in [210]: sometimes the utilization of multiple views may even deteriorate the final performance, which is even worse than the performance of the best single view.

We further discussed the potential limitations of MCHC and MCHC-PF. On the one hand, both MCHC and MCHC-PF are based on the single-view clustering method of CNNC. However, the merging process of CNNC relies on nearest-neighbor statistics and leverages a method like single-linkage to measure the distance between clusters. Therefore, the performance of CNNC on some high-dimensional and sparse data sets may be particularly satisfactory, which also further leads to the non-competitive performance of MCHC and MCHC-PF. On the other hand, unlike MCHC, MCHC-PF does not consider the information from each single view. When there are conflicting views in the multi-view data, the performance of MCHC-PF may decrease.

4.6 Conclusion

This chapter proposes a Multi-view adjacency-Constrained Hierarchical Clustering (MCHC) and its parameter-free version (MCHC-PF). By introducing the fusion distance matrices with extreme weights, adjacency-constrained nearest neighbor clustering and the internal evaluation index based on Rawls' Max-Min criterion, the promising clustering performance can be obtained by MCHC. Furthermore, without any parameter selection, MCHC-PF can provide partitions at different granularity levels with a low time complexity. Extensive tests on eight real-world data sets demonstrate that the proposed MCHC (-PF) method outperforms the 10 current state-of-the-art methods.

Chapter 5. PSO-based multi-view nearest neighbor clustering

5.1 Introduction

This chapter will mainly address another issue in prior multi-view clustering algorithms, difficulty in finding globally optimal view weights.

On the one hand, most previous multi-view clustering algorithms are based on spectral clustering [88], [225] or subspace clustering [47], [101]. However, from the perspective of basic clustering principles, both spectral clustering and subspace clustering have some inherent limitations. For example, spectral clustering [72], [129] suffers from the following three problems: a) the instability of results caused by different initializations; b) the K value required to construct adjacency matrix needs to be adjusted, and c) it can only provide clustering results with a single granularity. For subspace clustering [206], a) setting the global density threshold causes the algorithm to have poor performance in identifying clusters with varying densities; b) regularization parameters regarding the number of the subspaces are cumbersome to be set.

On the other hand, most multi-view clustering algorithms employ gradient-based optimization algorithms in finding optimal view weights. However, these algorithms may become trapped in a local minimum, resulting in poor performance. Moreover, these optimization algorithms generally require that objective (fitness) functions must be derivable and continuous, thereby greatly reducing the diversity of objective functions. In contrast, evolutionary optimization algorithms, such as particle swarm optimization (PSO) [236], are more likely to reach the global optimum [54], [55], and are much less restrictive on the properties of the objective functions. Furthermore, evolutionary optimization algorithms are widely used to improve single-view clustering algorithms [83], [227], but there are very few works that explore PSO to optimize multi-view clustering.

Based on the above analysis, in this chapter, we propose a particle swarm optimization (PSO)-based Multi-view Nearest Neighbor Clustering (PMNNC) algorithm. Different from most previous multi-view clustering based on spectral clustering or subspace clustering, we introduce an adjacency-constrained nearest neighbor clustering (CNNC) to enhance the clustering performance on fusion data from multiple views. Further, we propose a new fitness function based on the clustering internal validity index to help learn parameters more accurately. Finally, we combine PSO and CNNC to learn a fusion distance matrix from multiple views to improve the clustering results.

The following are the main contributions of this chapter:

- 1) Proposing a particle swarm optimization (PSO)-based Multi-view Nearest Neighbor Clustering (PMNNC) algorithm that can obtain promising clustering results.
- 2) Proposing a new fitness function based on the clustering internal validity index to help learn parameters more accurately.
- 3) The proposed method's superiority is demonstrated by experimental results on seven real-world data sets.

5.2 Proposed Method

5.2.1 Adjacency-constrained nearest neighbor clustering (CNNC)[*]

Nearest neighbor clustering (NNC) has recently become a research focus [13], [59], [211]. Compared with conventional hierarchical clustering algorithms, such as average-linkage and ward-linkage, NNC has a reduced computing complexity (i.e., *O(nlogn)*),

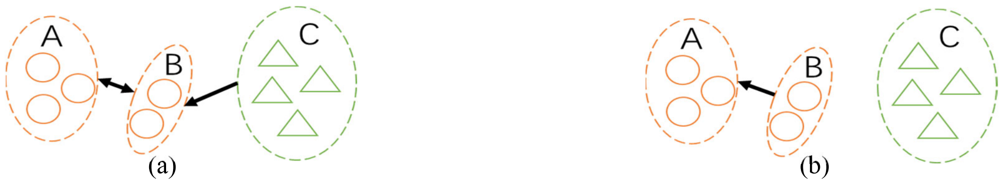


Figure 5.1. a simple example of the traditional NNC merging process (a), and the CNNC procedure (b). We employ dotted lines to denote clusters (i.e., A, B, and C), and circles or triangles to represent data samples. In traditional NNC, cluster A and cluster B are chosen to merge, cluster C and cluster B are chosen to merge, because cluster A and cluster B are the nearest neighbor of each other, and the nearest neighbor of cluster C is cluster B. However, in the CNNC, cluster C and cluster B are not chosen to merge because mass(C)>mass(B).

[*] Since this thesis is organized by the compilation of papers (Chapters) and each paper (Chapter) has a different focus, the partial clustering mechanism of TC in Chapter 3 is expressed as CNNC here.

because, in each iteration, NNC merges several clusters that are close to each other, instead of merging only one nearest cluster like traditional hierarchical clustering. However, whether traditional hierarchical clustering or NNC, it is based on the statistic of nearest neighbor, that is, as long as the neighbor relationship is satisfied, the merger is performed. In this way, samples from different classes may also be merged, thereby impairing clustering accuracy. Here, we introduce an adjacency-constrained nearest neighbor clustering (CNNC) algorithm, which leverages clusters with larger masses to guide the merging process, thereby preventing trivial wrong merging in conventional NNC methods. Fig. 5.1 illustrates the difference between the conventional nearest neighbor clustering and the adjacency-constrained nearest neighbor clustering.

Given a data set X, initially, each sample is its own cluster. Given the number of samples contained in a cluster as the mass of the cluster, therefore, in the beginning, the mass of each cluster equals 1. The following rule is then applied to form connections between clusters:

$$\zeta_j \rightarrow \zeta_j^N, if\ mass(\zeta_j) \leq mass(\zeta_j^N) \tag{5.1}$$

where ζ_j denotes the *j*-th cluster, ζ_j^N denotes the 1-nearest cluster of ζ_j. $mass(\zeta_j)$ represents the mass of ζ_j (i.e, the number of samples ζ_j contained). Similarly, $mass(\zeta_j^N)$ is the mass of ζ_j^N. The symbol " $\rightarrow$ " denotes a connection (i.e, merger) C_j between ζ_j and ζ_j^N. This process can be also defined in a graph *G*,

$$A(\zeta_j, \zeta_j^N) = \begin{cases} 1,\ if\ mass(\zeta_j) \leq mass(\zeta_j^N) \\ 0,\ otherwise \end{cases} \tag{5.2}$$

where *A* stands for the adjacency matrix of *G*. Then, new clusters can be obtained by calculating the connected components of the adjacency matrix *A*. At this point, one iteration has been completed. By repeating this merger process according to Eq. (5.2), all clusters will eventually merge into one cluster and form a hierarchical tree. Each layer of the hierarchical tree then can be regarded as a partition under a specific granularity.

Each connection (i.e., merger) C_j has two intuitive properties. One is the product of the mass of the two clusters it connects

$$M_j = mass(\zeta_j) \times mass(\zeta_j^N) \tag{5.3}$$

The other is the square of the distance between the two clusters it connects

$$D_j = d^2\left(\zeta_j, \zeta_j^N\right) \tag{5.4}$$

A reasonable partition can be obtained through a certain layer (granularity) of the clustering tree, Besides, CNNC can also be assigned the desired number of clusters *K*. After simply removing *K*-1 connections with relatively large $M_j \times D_j$, then we can get a partition containing *K* clusters.

5.2.2 Particle swarm optimization (PSO)

Particle Swarm Optimization (PSO) is an effective method for optimizing continuous nonlinear functions that models so-called social behaviors [83], [226]. A swarm in the context of PSO is a collection of potential solutions to an optimization problem, each of which is referred to as a particle. The primary objective of PSO algorithm is to find the particle's position that results in the best assessment of the fitness (objective) function. Each particle in the search space is represented by a position in N_d dimensional space, and the algorithm moves the particles across this multi-dimensional space, adjusting their position towards the best position they have discovered so far, as well as the best position in their respective neighborhoods [83], [226].

In addition, each particle retains the following values:

- x_i, its current position
- v_i, its current velocity
- y_i, its best position, found so far [83], [226].

Based on the above notation, the position of a particle is modified as per the following equation:

$$v_{i,k}(t+1) = wv_{i,k}(t) + c_1 r_{1,k}(t)\left(y_{i,k}(t) - x_{i,k}(t)\right) + c_2 r_{2,k}(t)\left(y(t) - x_{i,k}(t)\right) \tag{5.5}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{5.6}$$

In Eq. (5.5), w stands for the inertia weight, c_1 and c_2 for acceleration constants, while $r_{1,k}(t)$ and $r_{2,k}(t)$ are taken from a uniform distribution $U(0,1)$. The particle's velocity is then determined using 1) its prior velocity, 2) a cognitive component relating to its best-achieved distance, and 3) a social component that considers the best-achieved distance over all the particles in the swarm. The best position of a particle is calculated using Eq. (5.7), which simply updates the best position if the current i-timestep's fitness value is less than the particle's prior fitness value.

$$y_i(t+1) = \begin{cases} y_i(t) \;\; if \;\; f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) \;\; if \;\; f(x_i(t+1)) \geq f(y_i(t)) \end{cases} \tag{5.7}$$

The PSO is commonly run by using Eqs. (5.5)-(5.6) repeatedly until a certain number of iterations is reached. It is crucial to note that, whereas [83], [226] presents two types of PSO techniques, g*best,* and l*best*, where the social components are essentially bound to the particle's current neighborhood rather than the entire swarm, we refer only to the fundamental g*best* proposal in this study.

5.2.3 Fitness function based on a novel internal validity index: minimum spanning tree-based Dunn's index (MSTDI)

To evaluate the performance of PSO after each iteration, we need to propose a fitness function. Here we consider using internal validity index [221] – Dunn's index (DI),

$$DI_\pi = \min_l \left\{ \min_m \left(\frac{\min_{a \in \zeta_l, b \in \zeta_m} d(a,b)}{\max_j \left\{ \max_{a,b \in \zeta_j} d(a,b) \right\}} \right) \right\} \tag{5.8}$$

where ζ_j is the j-th cluster, $d(a,b)$ stands for the distance between the samples a and b. π denotes a partition generated from a clustering algorithm. DI is a distance matrix-based validity index, which doesn't need to know specific coordinates of data samples. As Eq. (5.8) shows, DI is calculated by dividing the minimum inter-cluster distance by the maximum intra-cluster distance. However, DI uses the distance between the two samples furthest apart in the cluster as the intra-class distance. This method of calculating intra-class distance is extremely susceptible to noise or outliers. Therefore, we alleviate this shortcoming based on the minimum spanning tree (MST) to propose an MST-based DI. We transform each cluster ζ_j to a connected, weighted, undirected graph $G_j(V,E)$, where each node corresponds to a data sample, and each weight corresponds to the distance between samples. Then we exploit Prim's algorithm [228] to generate an MST, $T_j(V,E')$ for ζ_j. For each edge $(q,s) \in T_j(V,E')$, there is a weight $d(q,s)$ associated with it. Finally, we exploit the weighted sum of the edges in $T_j(V,E')$ as the intra-class distance ζ_j. MSTDI can be formulated as

$$MSTDI_\pi = \min_l \left\{ \min_m \left(\frac{\min_{a\in\zeta_l, b\in\zeta_m} d(a,b)}{\max_j \left\{ \sum_{(q,s)\in T_j(V,E')} d(q,s) \right\}} \right) \right\} \tag{5.9}$$

Since both MSTDI and DI are positively correlated with clustering performance, we take the reciprocal of MSTDI as the fitness function (F), that is

$$F_\pi = \frac{\max_j \left\{ \sum_{(q,s)\in T_j(V,E')} d(q,s) \right\}}{\min_l \left\{ \min_m \left(\min_{a\in\zeta_l, b\in\zeta_m} d(a,b) \right) \right\}} \tag{5.10}$$

5.2.4 PSO-based multi-view nearest neighbor clustering

Given multi-view data $\{X^{(o)}\}_{o=1}^{h}$ collected from h views, for o-th view, $X^{(o)} \in R^{n\times dim_o}$, $D^{(o)}$ is the distance matrix of $X^{(o)}$, where n and dim_o denote the number of instances and the dimensions of the o-th view respectively. Our goal is to learn a fusion distance matrix (FDM) from these h views. Therefore, in the context of the PSO-based clustering, a single particle x_i represents an h-dimension weight vector.

$$x_i = (p_{i1}, \dots, p_{io}, \dots p_{ih}) \tag{5.11}$$

$$FDM_i = \sum_{o=1}^{h} p_{io} D^o = x_i \cdot \{D^{(o)}\}_{o=1}^{h} \tag{5.12}$$

where p_{io} is the weighting for the o-th view distance matrix in the i-th particle. That is, each particle x_i corresponds to a fusion distance matrix FDM_i. We perform the CNNC algorithm on FDM_i and get the partition π_i, then leverage the fitness function F_π to evaluate the loss. Therefore, the induced optimization model of this strategy is as follows

$$\mathbf{min}\, F_{\pi_i}\left(x_i, \{D^{(o)}\}_{o=1}^{h}\right) = \frac{\max_j \left\{ \sum_{(q,s)\in T_j(V,E')} d_i(q,s) \right\}}{\min_l \left\{ \min_m \left(\min_{a\in\zeta_l, b\in\zeta_m} d_i(a,b) \right) \right\}} \tag{5.13}$$

$$s.t. \sum_{o=1}^{h} p_{io} = 1, p_{io} \in [0,1] \tag{5.14}$$

where $d_i(\ )$ denotes the distance between samples in i-th fusion distance matrix FDM_i. A simple flowchart of the proposed PMNNC is shown in Fig. 5.2, and the pseudocode of

PMNNC is presented in Algorithm 5.1.

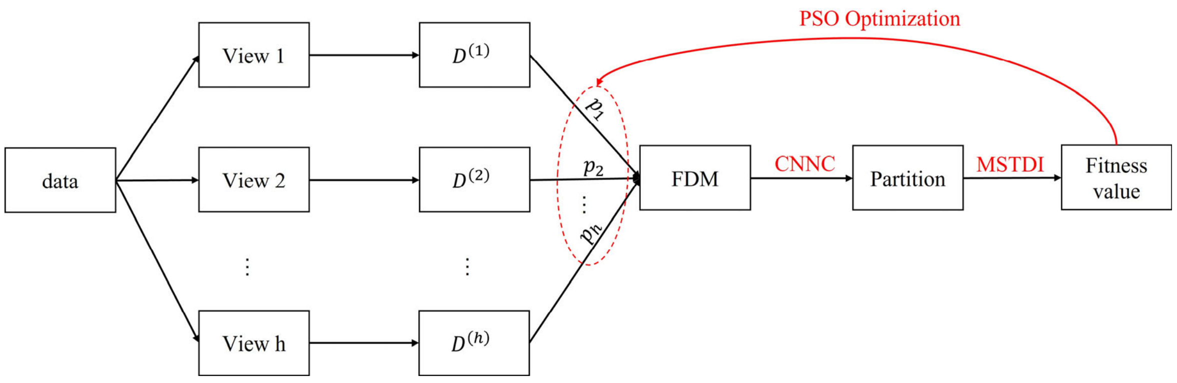


Figure 5.2. A simple flowchart of the proposed PMNNC.

Algorithm 5.1: Algorithm of the proposed PMNNC

```
1 **Input:** Multi-view data $\{X^{(o)}\}_{o=1}^{h}$, the desired cluster number $K$.
2 **Output:** Best partition $\pi_{best}$.
3 Calculating the distance matrix $D^{(o)}$ of each view.
4 **for** $t$=1: $t_{max}$ **do**
5 **for** z=$x_i$: $x_{max}$ **do**
6 Calculating $FDM_i$ by Eq. (5.12).
7 Initializing adjacency matrix $A$.
8 Constructing cluster sets $\{\zeta_j\}$ (Initially, regard each sample as a cluster).
9 **while** cluster sets $\{\zeta_j\}$ have more than two clusters **do**
10 Searching the nearest cluster of $\zeta_j$ with higher mass according to $FDM_i$.
11 Updating $A$ by Eqs. (5.1)-(5.2) (Using two nearest samples respectively from two clusters to represent
       these two clusters).
12 Calculating $M_j$ and $D_j$ of $C_j$ by Eqs. (5.3)-(5.4).
13 Updating cluster sets $\{\zeta_j\}$ based on $A$.
14 **end**
15 Updating $A$ by removing $K$-1 $C_j$ with largest $M_j \times D_j$.
16 Getting partition $\pi_i$ based on $A$.
17 Calculating $\mathrm{F}_{\pi_i}$ by Eqs. (5.13)-(5.14).
18 **end**
19 Updating the global best and local best positions.
20 Updating the weight vectors by Eqs. (5.5)-(5.6).
21 **end**
22 **Return** the optimal weight vector $x_{best}$ and its corresponding partition $\pi_{best}$.
```

5.3 Experiment and results

5.3.1 Data sets description

Table 5.1. Statistics of multi-view data sets.

Datasets	#Views	#Samples	#Clusters
100-leaves	3	1600	100
COIL20	3	1440	20
Handwritten	2	2000	10
ORL	3	400	40
UMIST	3	575	20
CMU-PIE	3	2856	68
COIL100	3	7200	100

(1) 100-leaves: The 100-leaves data set [229]. The size of the original 100-leaves images varies as well. Shape descriptors, fine-scale margins, and texture histogram features are used to display samples from various perspectives in three different views.

(2) COIL20: There are 1440 grayscale images of 20 different things in this data set [160]. For the original features scenario, each image is downscaled to 32 by 32 pixels. In the case of numerous hand-crafted features, three sorts of features are extracted. Their features are 1024, 3304, and 6750 pixels in size, respectively.

(3) Handwritten: This is made up of 2000 samples spanning from 0 to 9 digits. Each sample is represented by two views: the first is a 240-feature vector formed from the average of pixels in 2x3 windows, and the second is a 76-feature Fourier coefficient vector [229] [215].

(4) ORL: This is made up of 400 images of the faces of 40 different people. For the original features scenario, each image is down sampled to 32 by 32 pixels [47] [145]. Three different types of features are used to represent each image in the handcrafted features scenario.

(5) UMIST: There are 564 photographs of 20 persons in this collection [158] (mixed race, gender, and appearance). Each individual is shown in several positions, ranging from profile to frontal views. Each image has a 256-bit greyscale with a resolution of around 220 by 220 pixels. Each image is represented by three heterogeneous feature sets, as described in [216]: 30 isometric projection (ISO), 30 principal component analysis (PCA), and 30 neighborhood preserving embedding (NPE).

(6) CMU-PIE: This data set [148] includes 2856 frontal-face images of 68 people, each with 42 different illuminations. Each photograph was reduced to a 32×32-pixel size. Each image is expressed using three feature sets: 30 ISO, 30 PCA, and 30 NPE.

(7) COIL-100: This data set [147] is a collection of 7200 color photographs that represent 100 different objects. Each image has a resolution of 128×128 pixels. There are 72 different images in varied positions for each object. Three feature sets are used to express each image: 30 ISO, 30 PCA, and 30 NPE. Table 5.1 contains the complete statistics for these data sets.

5.3.2 Compared algorithms

We put PMNNC up against 10 state-of-the-art multi-view clustering techniques. They include: K-means; Graph-based multi-view clustering (GMC) [92]; Unified graph learning for multi-view clustering (UGLMC) [89]; View variation and view heredity clustering (V3H) [217]; Affinity aggregation for spectral clustering (AASC) [43]; Multi-view clustering via adaptively weighted Procrustes (AWP) [91]; Co-regularized multi-view spectral clustering (CoReg) [44]; Multi-view consensus graph clustering (MCGC) [218]; Robust multi-view spectral clustering (RMSC) [219]; and Weighted multi-view spectral clustering (WMSC) [46]. To evaluate the performance of clustering algorithms, we utilized three widely used external clustering validation indices: Accuracy (ACC), Normalized mutual information (NMI) [133], and F-score [220]. The best clustering results were highlighted. We also presented the best clustering results of multiple views for K-means, a single-view clustering algorithm. To get the best results from various multi-view clustering algorithms, the settings were tweaked as specified in the original papers. For PMNNC, we set the swarm sizes and maximum iterations equal to 5 and 40, respectively (i.e., set $x_{max} = x_5$ and $t_{max} = 40$ in Algorithm 5.1) and took the average results of the three runs. All the tests were run on a workstation with two 14-core Intel Xeon 6132 CPUs running at 2.6 GHz and 3.7 GHz, as well as 96GB of RAM.

Table 5.2. Clustering results of PMNNC and other algorithms in the metric of ACC.

Sources	Methods	100-leaves	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
-	K-means	.5780	.6410	.6921	.5703	.4617	.5377	.5737
TKDE-20	GMC	.8238	.7910	.8300	.6325	.5217	.7048	.7692
ICDM-19	UGLMC	.8001	.9014	.7425	.6900	.6043	.1863	.7267
TAI-21	V3H	.8219	.6005	.8670	.7478	.5245	.7283	.6514
CVPR-12	AASC	.8779	.7806	.8334	.7352	.4428	.5396	.6564
KDD-18	AWP	.7800	.7708	.9315	.6975	.5461	.7749	.7029
NeurIPS-11	CoReg	.8421	.8280	.9111	.7880	.5294	.7382	.7839
TIP-18	MCGC	.6075	.3882	.1005	.5950	.4487	.7006	.5194
AAAI-14	RMSC	.7313	.4092	.4098	.7977	.4753	.7465	.2216
AAAI-18	WMSC	.8789	.8463	.8335	.8068	.4897	.6633	.7142
-	**PMNNC**	**.9273**	**.9771**	**.9775**	**.8642**	**.8145**	**1.0000**	**.9261**

Table 5.3. Clustering results of PMNNC and other algorithms in the metric of NMI.

Sources	Methods	100-leaves	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
-	K-means	.7996	.8004	.7071	.7784	.6771	.7990	.8239
TKDE-20	GMC	.9296	.9410	.8767	.8590	.7373	.8892	.9371
ICDM-19	UGLMC	.9196	.9705	.8505	.8630	.8373	.3919	.9309
TAI-21	V3H	.9099	.7663	.7425	.8632	.6833	.8666	.8656
CVPR-12	AASC	.9590	.8958	.8827	.8538	.6619	.7921	.8676
KDD-18	AWP	.9013	.9264	.9026	.8584	.7203	.9140	.9163
NeurIPS-11	CoReg	.9325	.9425	.8811	.8905	.7412	.8829	.9264
TIP-18	MCGC	.7606	.6515	.0350	.8035	.6716	.8070	.7841
AAAI-14	RMSC	.8828	.7155	.4814	.8896	.6673	.8432	.5092
AAAI-18	WMSC	.9527	.9484	.8772	.8950	.7047	.8610	.9026
-	**PMNNC**	**.9643**	**.9866**	**.9493**	**.9385**	**.9056**	**1.0000**	**.9841**

Table 5.4. Clustering results of PMNNC and other algorithms in the metric of F-score.

Sources	Methods	100-leaves	COIL20	Handwritten	ORL	UMIST	CMU-PIE	COIL100
-	K-means	.4662	.6125	.6356	.4547	.4078	.4744	.5164
TKDE-20	GMC	.5042	.7943	.8113	.3599	.4620	.6171	.7195
ICDM-19	UGLMC	.7501	.8696	.7547	.5951	.5701	.0332	.5252
TAI-21	V3H	.7469	.5622	.7613	.6591	.4403	.6178	.5902
CVPR-12	AASC	.7783	.7758	.8167	.5948	.3656	.3766	.4795
KDD-18	AWP	.7129	.7187	.8875	.6163	.4891	.7555	.6985
NeurIPS-11	CoReg	.7907	.8088	.8588	.7091	.4778	.6790	.7583
TIP-18	MCGC	.0991	.1992	.1810	.2588	.3189	.2694	.1390
AAAI-14	RMSC	.5129	.3194	.2881	.7178	.2846	.3751	.0465
AAAI-18	WMSC	.8467	.8331	.8187	.7283	.4177	.6079	.6916
-	**PMNNC**	**.8817**	**.9646**	**.9552**	**.8149**	**.8243**	**1.0000**	**.9280**

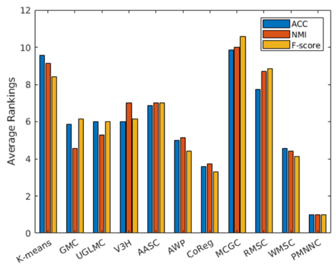


Figure 5.3. Average rankings for all multi-view clustering methods on all data sets.

5.3.3 Results and analysis

The clustering results are shown in Tables 5.2-5.4. On all data sets, the proposed

PMNNC beat all other clustering algorithms. Whether compared with single-view clustering algorithms or multi-view clustering algorithms, PMNNC shows considerate performance advantages. In particular, the results of our PMNNC were roughly 21%, 22.5%, and 14.2% better than the second-best clustering results on UMIST, CMU-PIE, and COIL100 data sets, respectively, using the metric ACC. For the metric NMI, the results of our PMNNC were about 6.8%, 8.6%, and 4.7% better than the second-best clustering results on UMIST, CMU-PIE, and COIL100 data sets, respectively. Finally, our PMNNC results were roughly 9.5%, 25.4%, 24.5%, and 17% better than the second-best clustering results on COIL20, UMIST, CMU-PIE, and COIL100 data sets, respectively, using the metric F-score. Besides, in terms of the rank in Fig. 5.3, the next-best algorithm, CoReg, was three times plus higher than PMNNC in its ranking scores among ACC, NMI, and F-score. Finally, Fig. 5.4. shows the trend of best fitness function value with PSO iterations on the seven real-world data sets, where we can conclude that PMNNC converges quickly on most data sets.

From a theoretical perspective, the inferior performance of numerous multi-view spectral or subspace clustering methods in comparison to PMNNC can primarily be ascribed to two key aspects. Firstly, these techniques are based on spectral clustering or subspace clustering, both of which exhibit inherent limitations. For instance, spectral clustering faces difficulties in accurately discerning the intrinsic manifold structure within data when generating the k-nearest neighbor similarity graph. This challenge arises due to the fact that spectral clustering is not adept at capturing complex structures. Conversely, CNNC, which is a component of PMNNC, can more effectively capture this structure owing to its constrained merging manner, providing more accurate connections between data samples. Secondly, the majority of previous algorithms rely on gradient-based optimization strategies to learn the weight of each view, with the aim of discovering a globally optimal joint representation. However, this objective is impeded by the very nature of gradient-based optimization algorithms, as they have a tendency to become ensnared in local optima. Consequently, the learned view weights fail to contribute positively to the final clustering phase. In stark contrast, PMNNC employs PSO to determine the most appropriate view weights. By utilizing PSO, the algorithm is better equipped to find the optimal global solution, thereby more effectively serving the subsequent clustering step. This advantage ultimately results in a more robust and efficient clustering process, overcoming the limitations faced by traditional multi-view spectral or subspace clustering techniques.

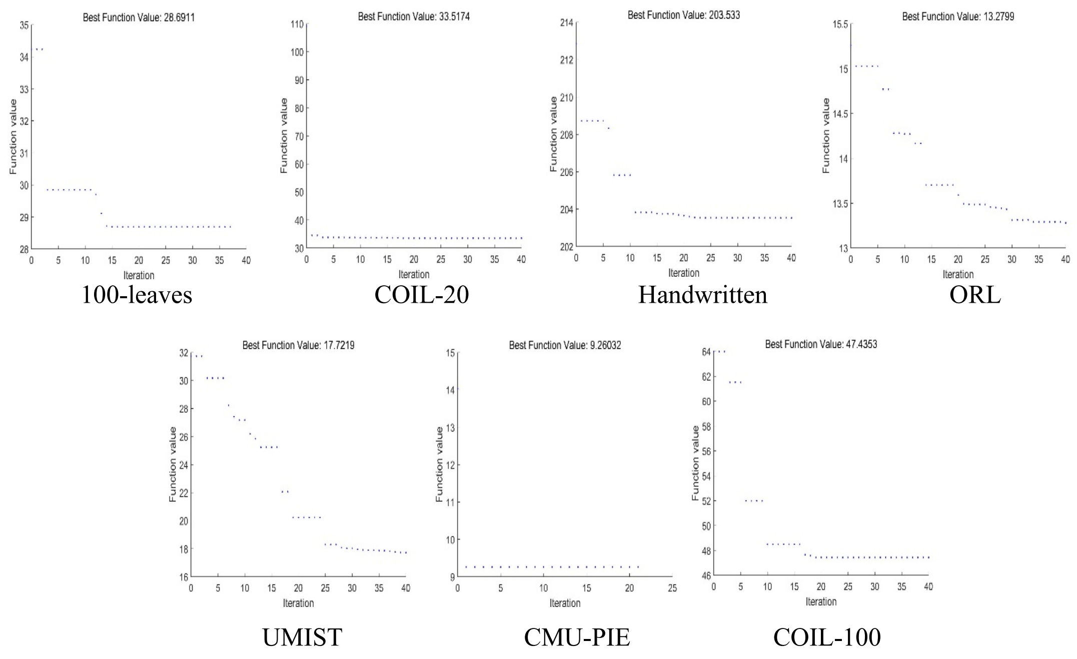


Figure 5.4. The trend of best fitness function value with PSO iterations on the seven real-world data sets respectively.

Table 5.5. Performance comparison with other PSO-based multi-view clustering methods, in the metric of ACC. The COIL100 data set causes errors in the GDL and SC codes.

Data sets	PMC-avg	PMC-ward	PMC-GDL	PMC-SC	PMNNC
100-leaves	.6677	.8635	.8048	.4008	**.9273**
COIL20	.4516	.7338	.8620	.6623	**.9771**
Handwritten	.5208	.9393	.8785	.7408	**.9775**
ORL	.6575	.7958	.7225	.7258	**.8642**
UMIST	.3548	.4284	.4913	.4470	**.8145**
CMU-PIE	.3910	.4230	.9895	.9076	**1**
COIL100	.6038	6703	-	-	**.9261**

Table 5.6. Performance comparison with other distance matrix-based internal indices, in the metric of ACC.

Data sets	DI	CVNN	CVDD	MSTDI
100-leaves	.8977	.6415	.9140	**.9273**
COIL20	.8525	.9431	.9155	**.9771**
Handwritten	.9775	.9067	.9670	**.9775**

ORL	.8625	.6783	.7692	**.8642**
UMIST	.6064	.6701	**.8209**	.8145
CMU-PIE	.8964	.7588	.9834	**1**
COIL100	.9194	.8428	.9261	**.9261**

5.4 Ablation study

5.4.1 Impact of adjacency-constrained nearest neighbor clustering (CNNC)

To show the superiority of CNNC in the proposed model. We replaced CNNC of PMNNC with other well-known clustering methods that can be performed on distance (or similarity) matrices, including average-linkage, ward-linkage, GDL [62], and spectral clustering (SC). GDL is a recent NNC-based hierarchical clustering algorithm. The generated new PSO-based multi-view clustering (PMC) methods were PMC-avg, PMC-ward, PMC-GDL, and PMC-SC. Similarly, we performed them on the seven data sets and used ACC to evaluate the results. The metric ACC was used to evaluate the results. According to Table 5.5, PMNNC still achieved the best performance on all data sets. Essentially, CNNC of PMNNC outperforms prior NNC methods or hierarchical clustering techniques when handling the fusion distance matrix. The superior performance can be attributed to CNNC's constrained merging manner, which enables it to more precisely identify the manifold structure present within the data.

5.4.2 Impact of minimum spanning tree-based Dunn's index (MSTDI)

To show the validity of the proposed MSTDI, we replaced it with the original validity index Dunn Index (DI) [221] and other two recent distance matrix-based internal validity indices, Clustering Validation index based on Nearest Neighbors (CVNN) [223] and Clustering Validity index based on Density-involved Distance (CVDD) [224] in the PMNNC framework. The metric ACC was used to evaluate the results. According to Table 5.6, compared with other three distance matrix-based internal validity indices, the proposed MSTDI is more suitable as the fitness function of PSO. Theoretically speaking, the other three distance matrix-based internal indices each possess some intrinsic shortcomings. For instance, DI calculates the intra-class distance using the distance between the two most distant points within a cluster, making it vulnerable to outliers. Moreover, CVNN struggles with adapting to sparse data sets, while CVDD is sensitive to variations in cluster densities.

5.4.3 Impact of the two hyperparameters

The PSO algorithm used in PMNNC typically involves two hyperparameters, namely the swarm size and maximum iterations. While PMNNC inherits these hyperparameters, we fixed the swarm size and maximum iterations at 5 and 40, respectively, for all the experiments reported in this study. We made this decision based on the observation that increasing these hyperparameters' values can lead to easier convergence to smaller fitness function values but also entails greater computational overhead. Thus, we arrived at a compromise setting for these hyperparameters.

To further investigate the impact of these hyperparameters on PMNNC's performance, we conducted additional experiments on the CMU-PIE data set, where we varied the swarm sizes within the range of {3, 4, 5, 6, 7} and the maximum iterations within the range of {20, 30, 40, 50, 60}. As illustrated in Figure 5.5, the ACC scores of PMNNC remained at 1 for all hyperparameter settings, indicating that the algorithm's performance is relatively insensitive to these hyperparameters. However, it is worth noting that the optimal hyperparameter values for PMNNC may vary across different data sets and applications, and further research is necessary to explore their impact fully.

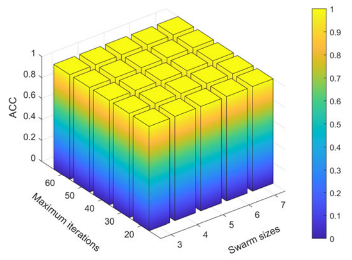


Figure 5.5. The ACC scores of PMNNC under varying hyperparameter settings on the CMU-PIE data set.

5.5 Conclusion

This chapter proposes a particle swarm optimization (PSO)-based Multi-view Nearest Neighbor Clustering (PMNNC). By introducing an adjacency-constrained nearest neighbor clustering (CNNC) algorithm and a new fitness function based on the clustering internal validity index, the promising clustering performance can be obtained by PMNNC. Extensive tests using seven real-world data sets demonstrate the supremacy of the proposed PMNNC over the 10 current state-of-the-art approaches.

Chapter 6. Almost ultrametric learning using pseudo labels from clustering

6.1 Introduction

In machine learning, measuring similarity between data points is a fundamental operation that plays a crucial role in many tasks. Similarity measures are used in various applications, such as clustering, classification, and retrieval systems, to identify patterns and similarities in the data. Under the right similarity measures, an unknown pattern can be accurately identified. Distance is a typical measure of similarity that is used in traditional machine learning algorithms such as KNN [105] and K-means [16]. The performance of KNN and K-means is strongly reliant on distance, with Euclidean distance being used in the majority of applications. Conversely, the Euclidean distance measure considers all the components of a feature vector as equal, without taking into account their relative importance when calculating the vector's output [106]. Therefore, in many real-world scenarios, using Euclidean distance as a metric will degrade the performance of machine learning algorithms. The emergence of distance metric learning solves this problem [230], [231]. Distance metric learning has gained significant attention in recent years for improving the performance of distance-based methods like KNN [105] and K-means [16], after being first introduced in 2003. The primary objective of metric learning is to reduce intra-class distance while increasing inter-class distance, resulting in each point being closer to other points with the same label and farther away from those with different labels [106].

On the other hand, past studies have shown that if a distance metric space is closer to an ultrametric space, it tends to have better clusterability, that is, it is more friendly to clustering [232]. An ultrametric space is a metric space where the triangle inequality is reinforced to $d(x,y) \leq max\{d(x,z), d(z,y)\}$ in mathematics [232]. The associated metric, ultrametric, is used in a variety of fields, such as condensed matter physics, geography, and landscape ecology.

In this chapter, we first introduce the difference between metric space and ultrametric space. Then, we propose a new metric called Almost UltraMetric (AUM) and prove that under weak conditions, it will be a true ultrametric. Since the learning of the proposed AUM requires the guidance of ground truth labels, we further propose using pseudo labels to approximate ground truth labels, thus making the learning process completely unsupervised. Since the pseudo labels are obtained by Chapter 3 (torque clustering), we call this whole metric learning framework, Almost UltraMetric Learning using Torque Clustering's pseudo labels (AUMLTC). It is worth mentioning that, unlike most previous methods, the proposed AUMLTC is unsupervised and parameter-free. The comparison and ablation experiments tested on several data sets validate the superiority of the proposed framework.

The following are the main contributions of this chapter:

• 1) Proposing a new metric called Almost UltraMetric (AUM).

• 2) Proving that under weak conditions, the proposed AUM will be a true ultrametric. Several additional properties of AUM are also extended.

• 3) Proposing a new parameter-free unsupervised metric learning framework, Almost UltraMetric Learning using Torque Clustering's pseudo labels (AUMLTC).

• 4) The proposed framework's superiority is demonstrated by experimental results on six data sets.

6.2 Proposed Method

6.2.1 Metric space and ultrametric space

A dissimilarity on a set of distinct points P is a mapping $d: P \times P \to \mathbb{R}$ such that

i. $d(x,y) \geq 0$;

ii. $d(x,y) = d(y,x)$;

iii. $d(x,y) = 0$ if and only if $x = y$;

A dissimilarity on P that adheres to the triangular inequality

iv. $d(x,y) \leq d(x,z) + d(z,y)$

for every $x, y, z \in P$ is a metric [232]. Furthermore, if the following condition is also satisfied

v. $d(x,y) \leq max\{d(x,z), d(z,y)\}$

in this case, d becomes an ultrametric, and the pair (P, d) is an ultrametric space [232]. A ultrametric must be a metric, but not vice versa. It is worth noting that in an ultrametric space, any r-spheric clustering is a perfect clustering [233].

6.2.2 Convert metric space to the proposed almost ultrametric (AUM) space

Consider a data set X including some distinct data points, and suppose their labels are known as $\mathcal{L}$; points with the same label are represented as a class ζ_i (or cluster). For $x, y \in X$ and labels $\mathcal{L}$ of X, write $x \sim_{\mathcal{L}} y$ if x and y belong to the same class in $\mathcal{L}$ and $x \nsim_{\mathcal{L}} y$ otherwise. First, we map X to a widely used metric space (e.g., Euclidean), denoted as (X, d). We regard (X, d) as the original metric space. Then, the proposed AUM space (X, d') can be obtained from the original space as follows.

For $x \sim_{\mathcal{L}} y$, from any class, the distance between them on the AUM space can be defined as:

$$d'(x,y) = \min_{i} \min_{x,y \in \zeta_i} d(x,y) \tag{6.1}$$

For $x \nsim_{\mathcal{L}} z$, from any two different classes, the distance between them on the AUM space can be defined as:

$$d'(x,z) = \max_{i} \max_{j} \max_{x \in \zeta_i, z \in \zeta_j} d(x,z) \tag{6.2}$$

Essentially, for Eq. (6.1), guided by labels $\mathcal{L}$, we take the minimum value of the distances between any two points from the same class in the original space as the distance between any two points from the same class in the proposed AUM space; for Eq. (6.2), we take the maximum value of the distances between any two points from different classes in the original space as the distance between any two points from different classes in the proposed AUM space. According to Eqs. (6.1)-(6.2), in the AUM space, the distances between points belonging to the same class are reduced, and the distances between points belonging to different classes are increased. That is, the proposed metric space shrinks the intra-class (or cluster) distance and enlarges the inter-class (or cluster) distance.

Property 1. The proposed AUM space (X, d') is an ultrametric space if $\min_{i} \min_{x,y \in \zeta_i} d(x,y) \le \max_{i} \max_{j} \max_{x \in \zeta_i, z \in \zeta_j} d(x,z)$ *in the original metric space.*

Proof: We first consider the condition $\min_i \min_{x,y\in\zeta_i} d(x,y) < \max_i \max_j \max_{x\in\zeta_i,z\in\zeta_j} d(x,z)$. For $x,y,z \in X$, there are three possibilities:

a) x,y,z belong to the same class, that is, $x\sim_{\mathcal{L}} y$, $y\sim_{\mathcal{L}} z$, and $z\sim_{\mathcal{L}} x$. According to Eq. (6.1), $d'(x,y) = d'(y,z) = d'(z,x) = \min_i \min_{x,y\in\zeta_i} d(x,y)$, so Eq. v in section 6.2.1 is satisfied.

b) x,y belong to the same class, and x,z and y,z do not belong to the same class, that is, $x\sim_{\mathcal{L}} y$, $y \nsim_{\mathcal{L}} z$, and $z \nsim_{\mathcal{L}} x$. According to Eqs. (6.1)-(6.2), $d'(x,y) = \min_i \min_{x,y\in\zeta_i} d(x,y) < \max_i \max_j \max_{x\in\zeta_i,z\in\zeta_j} d(x,z) = d'(y,z) = d'(z,x)$, so Eq. v in section 6.2.1 is satisfied.

c) Any two of x,y,z do not belong to the same class, that is, $x \nsim_{\mathcal{L}} y$, $y \nsim_{\mathcal{L}} z$, and $z \nsim_{\mathcal{L}} x$. According to Eq. (6.2), $d'(x,y) = d'(y,z) = d'(z,x) = \max_i \max_j \max_{x\in\zeta_i,z\in\zeta_j} d(x,z)$, so Eq. v in section 6.2.1 is also satisfied.

We can visualize the three cases a), b), and c) as shown in Fig. 6.1.

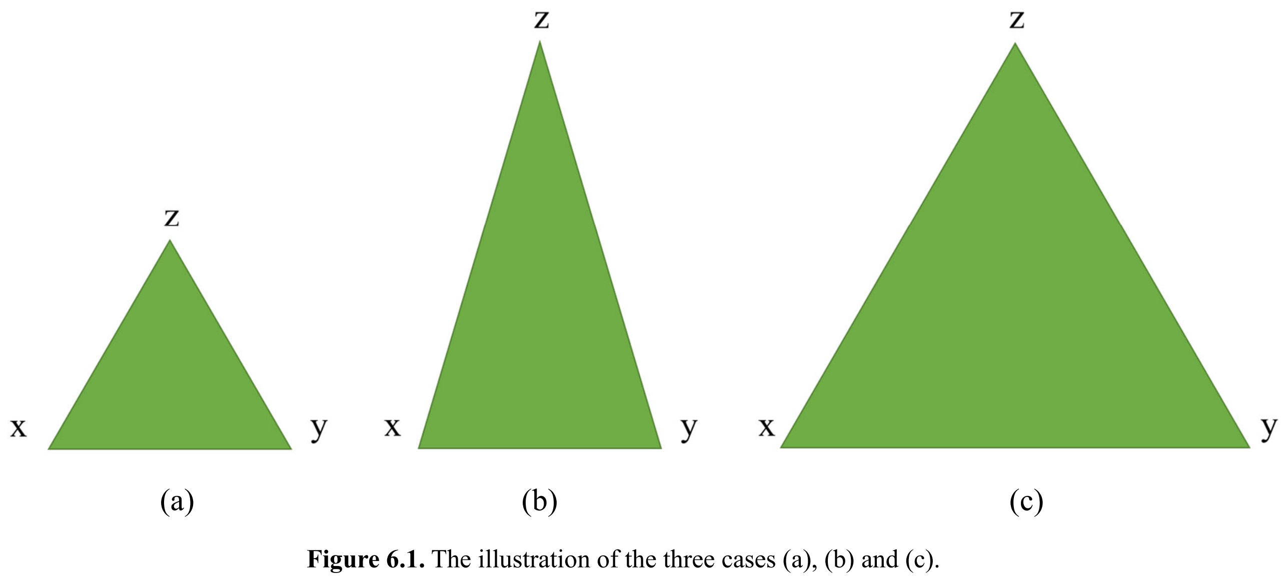


Figure 6.1. The illustration of the three cases (a), (b) and (c).

When $\min_i \min_{x,y\in\zeta_i} d(x,y) = \max_i \max_j \max_{x\in\zeta_i,z\in\zeta_j} d(x,z)$, no matter which of a), b), and c) holds, x,y,z will form an equilateral triangle, so Eq. v in section 6.2.1 is also satisfied.

$\min_i \min_{x,y\in\zeta_i} d(x,y) \le \max_i \max_j \max_{x\in\zeta_i,z\in\zeta_j} d(x,z)$ is a weak condition, which means that the proposed AUM is always an ultrametric.

In the past, many studies explained the characteristics of data sets from the perspective of metric space [157], [232]. The distribution characteristic of the data set is related to its clusterability. For example, if a data set is strictly separable, many clustering algorithms are robust on it [205]. Here, we first introduce two definitions [234], [205].

Definition 1. ($(\varepsilon, k) - strictly\ additive\ separable$). A data set with$(X, d)$ is $(\varepsilon, K) -$ strictly additive separable *if there exists a unique clustering* $\Gamma = \{\zeta_1, \ldots, \zeta_K\}$ *of X so that for all* $i \neq j$ *and all* $x, y \in \zeta_i, z \in \zeta_j$, $d(x, y) + \varepsilon \leq d(x, z)$[234], [205].

Definition 2. (Nice clustering). The clustering result $\mathbb{L}$ of a data set with (X, d) is nice *if for all* $x, y, z \in X$, $d(y, x) < d(z, x)$ *whenever* $x \sim_{\mathbb{L}} y$, $y \nsim_{\mathbb{L}} z$[234], [205].

According to definitions 1 and 2, we can easily deduce that the proposed AUM space has the following two properties.

Property 2. A data set with the proposed AUM space (X, d') is $(\varepsilon, K) -$ strictly additive separable *if* $\min_i \min_{x,y \in \zeta_i} d(x, y) \leq \max_i \max_j \max_{x \in \zeta_i, z \in \zeta_j} d(x, z)$, *where* $\varepsilon \geq 0$.

Property 3. The clustering result $\mathbb{L}$ of a data set with the proposed AUM(X, d') is nice *if* $\min_i \min_{x,y \in \zeta_i} d(x, y) < \max_i \max_j \max_{x \in \zeta_i, z \in \zeta_j} d(x, z)$.

The proofs for properties 2 and 3 are obvious, so we omit them here. From properties 2 and 3, we can see that the AUM space is very friendly to clustering.

6.2.3 Exploit pseudo labels of torque clustering to approximate ground truth labels

Even if the AUM space has some good properties, it still needs to be guided by labels $\mathcal{L}$. However, in the real world, the acquisition of ground truth labels is expensive. Therefore, we consider using pseudo labels $\mathbb{L}$ to approximate it, that is,

$$\mathbb{L} \approx \mathcal{L} \tag{6.3}$$

Pseudo labels are easily obtained by some clustering algorithms. To make the pseudo labels closer to the ground truth labels, we need to choose a relatively general clustering algorithm. In Chapter 3, we proposed a parameter-free autonomous clustering algorithm called torque clustering (TC). The algorithm has shown good performance on many data sets with different characteristics, such as overlapping, unbalanced, shaped, multi-objective, etc. Therefore, here we leverage TC to generate pseudo labels $\mathbb{L}$.

Given a data set X, initially, each sample is its own cluster. Given the number of samples contained in a cluster as the mass of the cluster, therefore, in the beginning, the mass of each cluster equals 1. The following rule is then applied to form connections between clusters:

$$\zeta_j \rightarrow \zeta_j^N, if\ mass(\zeta_j) \leq mass(\zeta_j^N) \tag{6.4}$$

where ζ_j denotes the j-th cluster, ζ_j^N denotes the 1-nearest cluster of ζ_j. $mass(\zeta_j)$ represents the mass of ζ_j (i.e, the number of samples ζ_j contained). Similarly, $mass(\zeta_j^N)$ is the mass of ζ_j^N. The symbol " $\rightarrow$ " denotes a connection (i.e, merger) C_j between ζ_j and ζ_j^N. This process can be also defined in a graph *G*,

$$A(\zeta_j, \zeta_j^N) = \begin{cases} 1, & if\ mass(\zeta_j) \leq mass(\zeta_j^N) \\ 0, & otherwise \end{cases} \tag{6.5}$$

where *A* stands for the adjacency matrix of *G*. Then, new clusters can be obtained by calculating the connected components of the adjacency matrix *A*. At this point, one iteration has been completed. By repeating this merger process according to Eq. (6.5), all clusters will eventually merge into one cluster and form a hierarchical tree. Each layer of the hierarchical tree then can be regarded as a partition under a specific granularity.

Each connection (i.e., merger) C_j has two intuitive properties. One is the product of the mass of the two clusters it connects

$$M_j = mass(\zeta_j) \times mass(\zeta_j^N) \tag{6.6}$$

The other is the square of the distance between the two clusters it connects

$$D_j = d^2(\zeta_j, \zeta_j^N) \tag{6.7}$$

Plotting all the connections on a two-dimensional graph of the two properties, called the decision graph. The decision graph helps to determine the abnormal connections, which have relatively large M_j and D_j. Then, removing these abnormal connections to get the final reasonable partition.

On the other hand, TC provides an automatic mechanism, named Torque Gap (*TGap*), to determine abnormal connections, eliminating the need to manually observe abnormal connections from the decision graph. The *TGap* is calculated by first calculating the torque τ_j of all connections, where

$$\tau_j = M_j \times D_j \tag{6.8}$$

Then sorting all connections in descending order according to their corresponding torque values, and call it the torque sorted connections list (TSCL). The connection in TSCL and its torque are denoted as $\grave{C}_j$ and $\grave{\tau}_j$, respectively. Abnormal connections must be the top several connections in the TSCL, because they have the largest torque values among all the connections.

The $TGap_j$ between each connection along with its following connection in the TSCL is calculated next. The formula for computing $TGap_j$ is

$$TGap_j = \omega_j \frac{\grave{\tau}_j}{\grave{\tau}_{j+1}},\ \grave{\tau}_{j+1} \neq 0 \tag{6.9}$$

where ω_j is a weighted value that indicates the proportion of connections among the top *j* connections of TSCL that have relatively large M_j, D_j, and τ_j values.

The process for defining ω_j is as follows: Eq. (6.4) will reveal many connections C_j throughout the entire clustering process and, as we know, each C_j has two properties, M_j and D_j. Therefore, the set of connections that have relatively large M_j, D_j, and τ_j values among all the connections (denoted as $Large\_C$) can be defined as:

$$Large\_C = \{C_k | (\tau_k \geq mean\_\tau) \cap (M_k \geq mean\_M) \cap (D_k \geq mean\_D)\} \tag{6.10}$$

where $mean\_\tau$ is the mean value of all τ_j, $mean\_M$ is the mean value of all M_j, and $mean\_D$ is the mean value of all D_j.

Top_C_j is the set of the top *j* connections of TSCL, and can be defined as

$$Top\_C_j = \{\grave{C}_1, \grave{C}_2, \dots, \grave{C}_j\} \tag{6.11}$$

Based on $Large\_C$ and Top_C_j, ω_j is defined as:

$$\omega_j = \frac{|Large\_C \cap Top\_C_j|}{|Large\_C|} \tag{6.12}$$

The largest $TGap_j$ is denoted as $TGap_L$, and the L connections at the top of the TSCL (i.e., $\{\grave{C}_1, \grave{C}_2, \ldots, \grave{C}_L\}$) are regarded as abnormal connections to be removed. The pseudo code of the torque clustering is shown as Algorithm 6.1.

Algorithm 6.1: Algorithm of Torque Clustering (TC)

```
1 Input: Data set $X$.
2 Output: Clustering labels $\mathbb{L}$.
3 Initializing adjacency matrix $A$.
4 Constructing cluster sets $\{\zeta_j\}$ (Initially, regard each sample as a cluster).
5 while cluster sets $\{\zeta_j\}$ have more than two clusters do
6 Searching the nearest cluster of $\zeta_j$ with higher mass according to the distance matrix of $X$ or via
      k-d tree.
7 Updating $A$ by Eqs. (6.4)-(6.5) (Using two nearest samples respectively from two clusters to represent these
      two clusters).
8 Calculating $M_j$ and $D_j$ of $C_j$ by Eqs. (6.6)-(6.7).
9 Updating cluster sets $\{\zeta_j\}$ by calculating the connected components of $A$.
10 end
11 Computing abnormal connections by Eqs. (6.8)-(6.12).
12 Updating $A$ by removing abnormal connections.
13 Getting clustering labels $\mathbb{L}$ by calculating the connected components of $A$.
```

6.2.4 Almost ultrametric learning using TC pseudo labels (AUMLTC)

Combining AUM and TC, we can obtain a pseudo label-based AUM space learning method, which we call AUMLTC. Given a data set X, we first use torque clustering (TC) to obtain the pseudo labels $\mathbb{L}$ on X and then leverage $\mathbb{L}$ to convert the original metric space to the proposed AUM space. Since TC is an unsupervised and parameter-free clustering algorithm, the proposed AUMLTC framework remains parameter-free and unsupervised. The pseudocode of AUMLTC is shown in Algorithm 6.2.

We further analyze the time complexity of AUMLTC in Algorithm 6.2. According to Chapter 3, the time complexity of TC is $O(n^2)$, so the complexity of step 3 is also $O(n^2)$. The complexity of steps 4-6 is $O(n)$, $O(n^2)$, and $O(n)$ respectively. The sum of the complexity of steps 7-8 is $O(n^2)$. To sum up, the time complexity of AUMLTC is approximately $O(n^2)$. In addition, if the nearest neighbor estimation algorithm such as the k-d tree is used when calculating the distance matrix, the sum of the complexity of step 3, step 5, and steps 7-8 will be reduced to $O(nlogn)$. At this point, the time complexity of AUMLTC is approximately $O(nlogn)$.

```
Algorithm 6.2: Algorithm of the proposed AUMLTC.
1 Input: Data set $X$.
2 Output: New distance matrix $DM'$.
3 Getting pseudo labels $\mathbb{L}$ using Torque Clustering (i.e., Algorithm 6.1).
4 Getting class (or cluster) sets $\Gamma = \{\zeta_1, \dots, \zeta_K\}$ by $\mathbb{L}$.
5 Computing the original distance matrix of $X$, denoted as $DM$, where $d(x,y), d(x,z) \in DM$.
6 Initializing a new distance matrix $DM'$, where $d'(x,y), d'(x,z) \in DM'$.
7 For $x \sim_{\mathcal{L}} y$, from any class of $\Gamma$, computing $d'(x,y)$ by Eq. (6.1).
8 For $x \nsim_{\mathcal{L}} z$, from any two different classes of $\Gamma$ respectively, computing $d'(x,z)$ by Eq. (6.2).
9 Return $DM'$.
```

6.3 Experiments and results

6.3.1 Data sets description

Table 6.1. Statistics of the test data sets.

Data sets	Instances	Dimensions	Clusters
FLAME	240	2	2
Spectral-path	312	2	3
Multi-objective	1500	2	6
COIL-20	1440	16384	20
UMIST	575	10304	20
CMU-PIE	2856	1024	68

In the experiment, we leveraged three synthetic data sets and three real-world data sets, including FLAME, Spectral-path, Multi-objective, UMIST, COIL-20, and CMU-PIE.

1) **FLAME** [136]: This data set was designed to test fuzzy clustering by local approximation of membership.
2) **Spectral-path** [137]: This data set was used to illustrate the performance of a path-based spectral clustering algorithm.
3) **Multi-objective** [143]: This data set was exploited to test an improved multi-objective clustering algorithm.
4) **COIL-20** [160]: There are 1440 greyscale pictures of 20 different objects in this data set. For the original feature scenario, each image is downscaled to 32 by 32 pixels.
5) **UMIST** [158]: There are 564 photographs of 20 people in this collection (mixed race, gender, and appearance). Each person is shown in a number of positions, ranging from

profile to frontal views. Each image has a 256-bit greyscale and a resolution of approximately 220×220 pixels.

6) **CMU-PIE** [148]: This data set includes 2856 frontal-face images of 68 people, each with 42 different illuminations. Each photograph was reduced to 32×32 pixels in size. Table 6.1 contains the complete statistics for these data sets.

6.3.2 Compared algorithms

We processed the six data sets mentioned above with the proposed AUMLTC and several classic or recent unsupervised metric learning methods and then ran the K-means algorithm on these processed data sets. Note that K-means can be run directly on the distance matrix due to its duality [235]. The compared metric learning methods include PCA [116], t-SNE [117], and UMAP [118]. UMAP is the latest method and can be used as an effective preprocessing step to enhance the performance of clustering. In addition, we included two metric learning-based clustering algorithms for comparison, i.e., spectral clustering (SC) [72] and subspace clustering (SSC) [84]. All experiments were evaluated in terms of the two commonly used external indices: normalized mutual information (NMI) [133] and accuracy (ACC). Due to the randomness of the K-means results, we took the average of 10 times as the final results. The experimental results are shown in Table 6.2 and Table 6.3.

6.3.3 Results and analysis

As Tables 6.2-6.3 show, the proposed AUMLTC outperformed all other metric learning methods. In particular, for the NMI score, the results of our AUMLTC were approximately 11.2%, 20.3%, 7.1%, and 18.8% better than the second-best metric learning results on the FLAME, Multi-objective, COIL-20, and UMIST data sets, respectively. For the ACC score, the results of our AUMLTC were approximately 44.1% and 31% better than the second-best metric learning results on the Multi-objective and UMIST data sets, respectively. Compared to the original K-means results, AUMLTC greatly boosts its clustering performance. Besides, in terms of the rank in Fig. 6.2, the next-best algorithm, SC, was two times plus higher than AUTCML in its ranking scores both on NMI and ACC.

To more intuitively reflect the performance of AUMLTC, for the three synthetic data sets FLAME, Spectral-path, and Multi-objective, we showed the heatmaps of the original distance matrix and the new distance matrix learned by AUMLTC (see Fig. 6.3). In addition, we applied t-SNE to reduce the new distance matrix to 2D, which was compared with the original data distribution, as shown in Fig. 6.4.

Table 6.2. Comparison of AUMLTC with other unsupervised metric learning methods, measured by NMI.

Data sets	Kmeans	Kmeans+PCA	Kmeans+t-SNE	Kmeans+UMAP	SC	SSC	Kmeans+AUTCML
FLAME	.4261	.4441	.4278	.8502	.8883	.1786	**1**
Spectral-path	.0001	.0001	.0057	.5512	1	.0023	**1**
Multi-objective	.5973	.5535	.6580	.7152	.7895	.3508	**.9925**
COIL20	.7596	.7680	.8796	.8838	.8870	.8734	**.9583**
UMIST	.6434	.6538	.7434	.6989	.7025	.7321	**.9310**
CMU-PIE	.4131	.3995	.6313	.8512	.8518	.9964	**1**

Table 6.3. Comparison of AUMLTC with other unsupervised metric learning methods, measured by ACC.

Data sets	Kmeans	Kmeans+PCA	Kmeans+t-SNE	Kmeans+UMAP	SC	SSC	Kmeans+AUTCML
FLAME	.8425	.8454	.8154	.9750	.9833	.6292	**1**
Spectral-path	.3446	.3458	.3776	.7513	1	.3622	**1**
Multi-objective	.5574	.4899	.5529	.5030	.5053	.2727	**.9980**
COIL20	.5706	.5977	.7558	.7592	.8240	.7306	**.8489**
UMIST	.4167	.4250	.5233	.4927	.4755	.5530	**.8626**
CMU-PIE	.1932	.1765	.3471	.6501	.7249	.9790	**1**

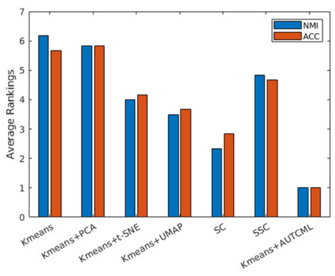


Figure 6.2. Average rankings for unsupervised metric learning methods on all data sets.

6.4 Ablation study

TC is an agglomerative clustering algorithm based on nearest neighbor statistics, and the performance of AUMLTC relies on the pseudo labels of TC. To illustrate this, in the AUMLTC framework, we replaced TC with other clustering algorithms based on nearest neighbor statistics and kept other components in the framework unchanged. These clustering algorithms include agglomerative clustering single-linkage (AC-S) [127], ward-linkage (AC-W), graph degree-linkage (GDL) [62], FINCH [13], and SNNDPC [76]. Among them, FINCH and SNNDPC are the two latest ones. The generated new metric learning frameworks were named MLACS, MLACW, MLGDL, MLFINCH, and MLSNNDPC. We compared these frameworks with the proposed AUMLTC on the six data sets and still exploited NMI and ACC to evaluate the results. As Tables 6.4-6.5 show, AUMLTC still maintained the advantages. Therefore, we conclude that the pseudo labels from TC are more suitable for learning the proposed AUM space. In addition, we can also see the impact of the accuracy of the pseudo labels on the performance of the AUM space learning from the experimental results. In Chapter 3, we have demonstrated that the accuracy of the above clustering algorithms is not as good as TC. Therefore, if their relatively inaccurate pseudo labels are adopted, the performance of the AUM space learning will decrease accordingly.

Table 6.4. Replace the pseudo labels of TC with those of other clustering algorithms, measured by NMI.

Data sets	MLACS	MLACW	MLFINCH	MLGDL	MLSNNDPC	AUMLTC
FLAME	.0479	.3297	.1216	1	.8288	**1**
Spectral-path	1	.0068	.1377	.4740	1	**1**
Multi-objective	.8341	.7229	.7812	.9906	.7304	**.9925**
COIL20	.7415	.7601	.6702	.9418	.5576	**.9583**
UMIST	.6685	.6092	.6963	.7410	.6146	**.9310**
CMU-PIE	.9897	.5982	.5291	.9047	.3930	**1**

Table 6.5. Replace the pseudo labels of TC with those of other clustering algorithms, measured by ACC.

Data sets	MLACS	MLACW	MLFINCH	MLGDL	MLSNNDPC	AUMLTC
FLAME	.6458	.7208	.6038	1	.9708	**1**
Spectral-path	1	.3750	.4019	.7340	1	**1**
Multi-objective	.7520	.7840	.6133	.9973	.7213	**.9980**
COIL20	.3993	.5444	.4280	.8556	.3354	**.8489**
UMIST	.4261	.3826	.4452	.5600	.4226	**.8626**
CMU-PIE	.9496	.2272	.2738	.7850	.1499	**1**

6.5 Conclusion

Compared with the previous metric learning algorithms, the proposed AUMLTC has the following advantages. First, unlike most existing metric learning algorithms, the AUMLTC is parameter-free; Second, the time complexity of most existing algorithms is $O(n^2)$, however, the cost of AUMLTC can be reduced to $O(nlogn)$; Finally, compared with existing algorithms, the AUMLTC generates a feature space that is more friendly to clustering.

Overall, there is a certain positive correlation between data ultrametricity and data clusterability [232]. This study is inspired by this viewpoint. First, we propose a new metric called the almost ultrametric (AUM) and prove that under certain weak conditions, the proposed AUM will be a true ultrametric. Then, based on a parameter-free clustering algorithm, torque clustering (TC), we propose using the pseudo labels of TC to approximate ground truth labels to learn the AUM. This learning framework is called AUMLTC. Both comparison and ablation experiments demonstrate the superiority of the proposed AUMLTC.

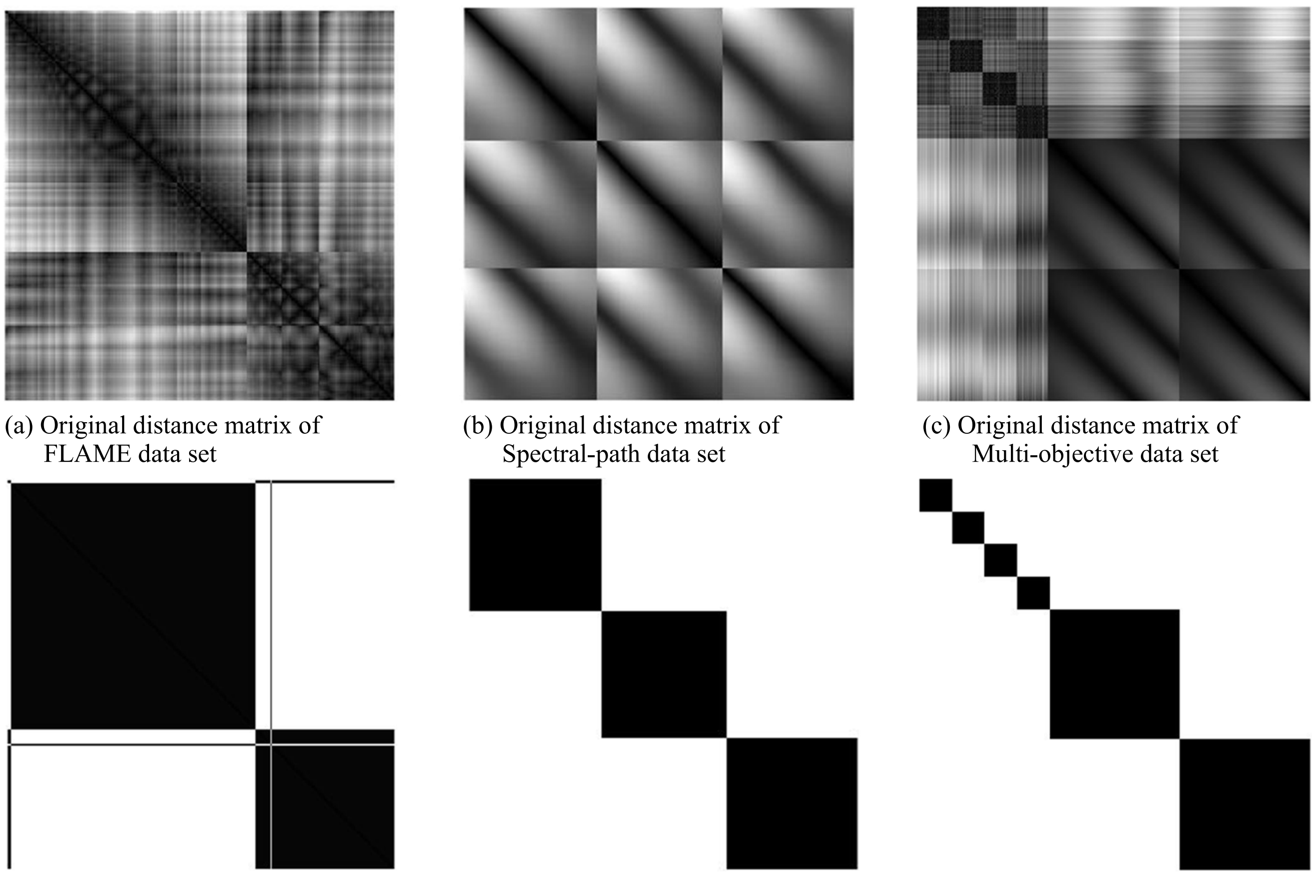
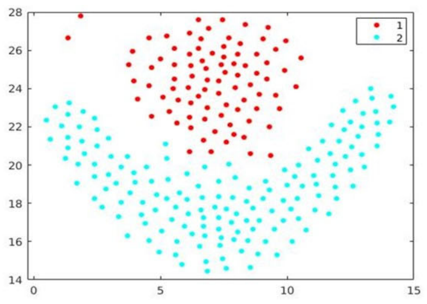
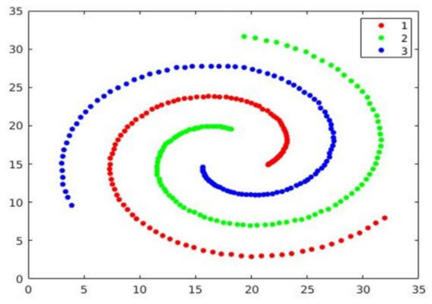
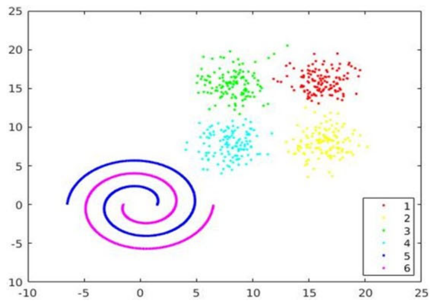
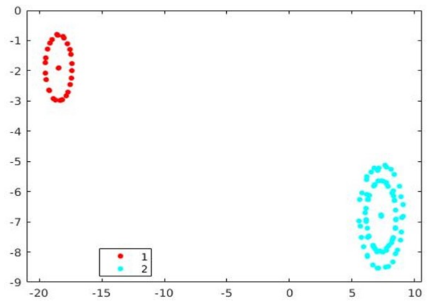

(a) Original distance matrix of FLAME data set
(b) Original distance matrix of Spectral-path data set
(c) Original distance matrix of Multi-objective data set
(d) New distance matrix of FLAME data set
(e) New distance matrix of Spectral-path data set
(f) New distance matrix of Multi-objective data set

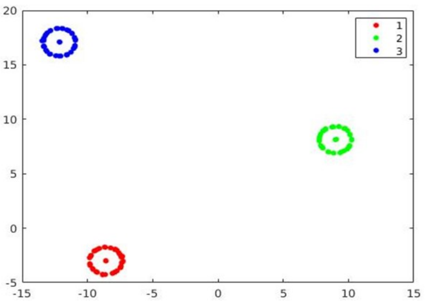
Figure 6.3. The visualization of the original distance matrix and the new distance matrix, on FLAME, Spectral-path, and Multi-objective data set, respectively.


(a) Original data distribution of FLAME data set


(b) Original data distribution of Spectral-path data set


(c) Original data distribution of Multi-objective data set


(d) New data distribution of FLAME data set


(e) New data distribution of Spectral-path data set

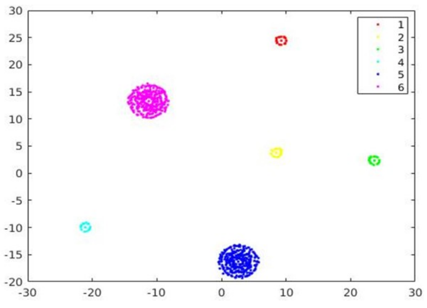

(f) New data distribution of Multi-objective data set

Figure 6.4. The visualization of the original data distribution and the new data distribution, on FLAME, Spectral-path, and Multi-objective data set, respectively.

Chapter 7. Conclusion and future work

This chapter presents the conclusions of the entire research and discusses future research directions.

7.1 Conclusion

This research can be divided into four main stages:

In stage one, to overcome various shortcomings in the previous clustering algorithms, we propose a brand-new autonomous clustering algorithm (Torque Clustering, TC) based on classical physics theory. The TC algorithm is parameter-free and can recognize various kinds of clusters and find the proper number of clusters and noise autonomously. Experiments on 76 synthetic and real-world data sets show the enormous versatility of the proposed TC algorithm, which remarkably outperforms the best compared algorithm. In particular, when compared with 19 state-of-the-art algorithms, the average ranking of these algorithms is at least four multiplicative factors higher than that of TC over all data sets. Additionally, we also compare TC with the latest state-of-the-art deep clustering algorithms on several challenging image data sets. The proposed TC algorithm without any deep representation achieves better or similar performance compared to deep clustering algorithms on image clustering.

In stage two, due to the good performance of the proposed TC, we exploit the clustering mechanism of TC as a backbone to propose a simple but efficient multi-view clustering framework: multi-view adjacency-constrained hierarchical clustering (MCHC). MCHC tries to solve two problems in existing multi-view clustering methods: a) parameter tuning and b) significant computational expense. Extensive tests on eight real-world data sets demonstrate that the proposed method outperforms the 10 current state-of-the-art methods. The average ranking of these algorithms is at least three multiplicative factors higher than that of MCHC over all data sets.

In stage three, we also exploit the clustering mechanism of TC as a backbone to propose a particle swarm optimization (PSO)-based Multi-view Nearest Neighbor Clustering

(PMNNC) algorithm. Different from the gradient-based optimization methods used in most prior multi-view clustering algorithms, PMNNC leverages the PSO optimization algorithm to calculate the optimal view weights, which solves the problem in current methods: difficulty in finding globally optimal view weights. Extensive tests using seven real-world data sets demonstrate the supremacy of the proposed PMNNC over the 10 current state-of-the-art approaches. The average ranking of these algorithms is at least three multiplicative factors higher than that of PMNNC over all data sets.

In stage four, we further apply the pseudo labels generated by TC to learn a new distance metric to help other algorithms improve performance in a parameter-free and unsupervised manner. We call this whole metric learning framework, Almost UltraMetric Learning using Torque Clustering's pseudo labels (AUMLTC). The comparison and ablation experiments tested on several data sets validate the supremacy of the proposed AUMLTC framework. The average ranking of prior algorithms is at least two multiplicative factors higher than that of AUMLTC over all data sets.

First, this research promotes the development of clustering theory and proposes a brand-new clustering algorithm (i.e., TC) that does not require any human intervention. Second, this research also overcomes the three problems in current multi-view clustering from different perspectives. Finally, this study also combines ultrametric space theory with pseudo labels from clustering to propose a new unsupervised and parameter-free metric learning method.

7.2 Future work

For Chapter 3, first, the proposed TC leverages a method such as single-linkage to measure the distance between clusters, which only considers the clusters' local structure. In future work, we will use other methods to compute the distance between clusters. Second, when determining the cluster halo, we will try to set different thresholds for different clusters, rather than relying on the global mean variable values, to make TC more robust to non-uniform noise. Thirdly, considering that TC's performance on certain high-dimensional and sparse data sets is not entirely satisfactory, future efforts will involve combining TC's pseudo labels with a deep neural network to develop representations better suited for TC. This procedure remains unsupervised, as the neural network is trained under the guidance of TC's pseudo labels. Fourth, in certain scenarios, data may be dispersed across multiple nodes, prompting us to consider extending TC into a distributed TC method in the future. Specifically, we could run TC independently and concurrently on each node, followed by further data compression for each node. After inter-node communication, we would perform the final clustering stage using TC on the compressed data at a hub once more. Lastly, taking into account that in certain situations, users might have specific preferences regarding clustering resolution, the necessity for two data samples to be in the same or different clusters, and the prior semantics of clusters, we plan to integrate certain decision rules and semi-supervised learning techniques to evolve TC into an interactive, self-governing clustering approach in the future.

For Chapters 4 and 5, the proposed MCHC and PMNNC, respectively, overcome some shortcomings in current multi-view clustering methods. For example, MCHC attempts to solve two problems in current multi-view clustering methods: a) parameter tuning and b) significant computational expense. PMNNC focuses on solving the third problem: c) difficulty in finding globally optimal view weights. Firstly, although both MCHC and PMNNC utilize TC's partial clustering mechanism, they focus on addressing different issues. This raises a new question: can we merge the advantages of MCHC and PMNNC to simultaneously tackle the three aforementioned problems? This will be a future research direction. Secondly, multi-view data may be distributed across various nodes, and adapting MCHC or PMNNC to this situation is another potential avenue for future research. One possible solution involves running MCHC or PMNNC in parallel on each node to further compress multi-view data. Once inter-node communication is complete, MCHC or PMNNC can be performed on a hub for the final clustering step. Lastly, in some cases, multi-view data may be presented as data streams. Extending MCHC and PMNNC into real-time multi-view clustering methods is another potential area of exploration. One possible approach includes integrating specific decision rules to determine whether current clusters should be further merged or divided, and if necessary, updating the current clusters by continuously executing MCHC or PMNNC.

For Chapter 6, the performance of the proposed AUMLTC relies on pseudo labels provided by TC. However, the performance of TC on some high-dimensional and sparse data sets is not particularly satisfactory. Therefore, in future work, we will try to obtain consensus pseudo labels from multiple clustering algorithms based on ensemble learning to further improve the performance of AUMLTC.

Bibliography

[1] S. A. Shah and V. Koltun, "Robust continuous clustering," *Proc. Natl. Acad. Sci.*, vol. 114, no. 37, pp. 9814–9819, Sep. 2017, doi: 10.1073/pnas.1700770114.

[2] H.-J. Chang, L.-P. Hung, and C.-L. Ho, "An anticipation model of potential customers' purchasing behavior based on clustering analysis and association rules analysis," *Expert Syst. Appl.*, vol. 32, no. 3, pp. 753–764, Apr. 2007, doi: 10.1016/j.eswa.2006.01.049.

[3] V. Y. Kiselev, T. S. Andrews, and M. Hemberg, "Challenges in unsupervised clustering of single-cell RNA-seq data," *Nat. Rev. Genet.*, vol. 20, no. 5, pp. 273–282, May 2019, doi: 10.1038/s41576-018-0088-9.

[4] C. Dorai and A. K. Jain, "Shape spectra based view grouping for free-form objects," in *Proceedings., International Conference on Image Processing*, Oct. 1995, vol. 3, pp. 340–343 vol.3. doi: 10.1109/ICIP.1995.538548.

[5] S. D. Connell and A. K. Jain, "Learning prototypes for online handwritten digits," in *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, Aug. 1998, vol. 1, pp. 182–184 vol.1. doi: 10.1109/ICPR.1998.711110.

[6] Z. Huang, "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining," *DMKD*, vol. 3(8), pp. 34–39, 1997.

[7] D. Jiang, G. Chen, B. C. Ooi, K.-L. Tan, and S. Wu, "epiC: An Extensible and Scalable System for Processing Big Data," *Proc VLDB Endow*, vol. 7, no. 7, pp. 541–552, Mar. 2014, doi: 10.14778/2732286.2732291.

[8] L. Mahon and T. Lukasiewicz, "Selective Pseudo-Label Clustering," in *KI 2021: Advances in Artificial Intelligence*, Cham, 2021, pp. 158–178. doi: 10.1007/978-3-030-87626-5_12.

[9] A. Saxena *et al.*, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, Dec. 2017, doi: 10.1016/j.neucom.2017.06.053.

[10] L. McInnes and J. Healy, "Accelerated Hierarchical Density Based Clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 33–42. doi: 10.1109/ICDMW.2017.12.

[11] M. Dash, H. Liu, P. Scheuermann, and K. L. Tan, "Fast hierarchical clustering and its validation," *Data Knowl. Eng.*, vol. 44, no. 1, pp. 109–138, Jan. 2003, doi: 10.1016/S0169-023X(02)00138-6.

[12] H. Koga, T. Ishibashi, and T. Watanabe, "Fast agglomerative hierarchical clustering algorithm using Locality-Sensitive Hashing," *Knowl. Inf. Syst.*, vol. 12, no. 1, pp. 25–53, May 2007, doi: 10.1007/s10115-006-0027-5.

[13] S. Sarfraz, V. Sharma, and R. Stiefelhagen, "Efficient Parameter-Free Clustering Using First Neighbor Relations," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 8926–8935. doi: 10.1109/CVPR.2019.00914.

[14] S. Kpotufe and U. von Luxburg, "Pruning nearest neighbor cluster trees," *ArXiv11050540 Cs Stat*, May 2011, Accessed: Nov. 23, 2020. [Online]. Available: http://arxiv.org/abs/1105.0540

[15] K. Chaudhuri, S. Dasgupta, S. Kpotufe, and U. von Luxburg, "Consistent Procedures for Cluster Tree Estimation and Pruning," *IEEE Trans. Inf. Theory*, vol. 60, no. 12, pp. 7900–7912, Dec. 2014, doi: 10.1109/TIT.2014.2361055.

[16] J. MacQueen, "SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, vol. 1(14), pp. 281–297.

[17] D. Pelleg and A. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *In Proceedings of the 17th International Conf. on Machine Learning*, 2000, pp. 727–734.

[18] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, Aug. 2004, pp. 551–556. doi: 10.1145/1014052.1014118.

[19] Rong Zhang and A. I. Rudnicky, "A large scale clustering scheme for kernel K-Means," in *Object recognition supported by user interaction for service robots*, 2002 International Conference on Pattern Recognition, Aug. 2002, vol. 4, pp. 289–292 vol.4. doi: 10.1109/ICPR.2002.1047453.

[20] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," *Proc. Eighteenth Annu. ACM-SIAM Symp. Discrete Algorithms Soc. Ind. Appl. Math.*, pp. 1027–1035, 2007.

[21] J. Yang, Y. Ma, X. Zhang, S. Li, and Y. Zhang, "An Initialization Method Based on Hybrid Distance for *k* -Means Algorithm," *Neural Comput.*, vol. 29, no. 11, pp. 3094–3117, Nov. 2017, doi: 10.1162/neco_a_01014.

[22] M. Wang, W. Zuo, and Y. Wang, "An improved density peaks-based clustering method for social circle discovery in social networks," *Neurocomputing*, vol. 179, pp. 219–227, Feb. 2016, doi: 10.1016/j.neucom.2015.11.091.

[23] T. Liu, H. Li, and X. Zhao, "Clustering by Search in Descending Order and Automatic Find of Density Peaks," *IEEE Access*, vol. 7, pp. 133772–133780, 2019, doi: 10.1109/ACCESS.2019.2939437.

[24] Z. Liang and P. Chen, "Delta-density based clustering with a divide-and-conquer strategy: 3DC clustering," *Pattern Recognit. Lett.*, vol. 73, pp. 52–59, Apr. 2016, doi:

10.1016/j.patrec.2016.01.009.

[25] A. Lotfi, P. Moradi, and H. Beigy, “Density peaks clustering based on density backbone and fuzzy neighborhood,” *Pattern Recognit.*, vol. 107, p. 107449, Nov. 2020, doi: 10.1016/j.patcog.2020.107449.

[26] R. J. G. B. Campello, D. Moulavi, and J. Sander, “Density-Based Clustering Based on Hierarchical Density Estimates,” in *Advances in Knowledge Discovery and Data Mining*, Berlin, Heidelberg, 2013, pp. 160–172. doi: 10.1007/978-3-642-37456-2_14.

[27] S. A. Seyedi, A. Lotfi, P. Moradi, and N. N. Qader, “Dynamic graph-based label propagation for density peaks clustering,” *Expert Syst. Appl.*, vol. 115, pp. 314–328, Jan. 2019, doi: 10.1016/j.eswa.2018.07.075.

[28] W. Zhang and J. Li, “Extended fast search clustering algorithm: widely density clusters, no density peaks,” *Comput. Sci. Inf. Technol. CS IT*, pp. 01–17, Apr. 2015, doi: 10.5121/csit.2015.50701.

[29] J. Xie, H. Gao, W. Xie, X. Liu, and P. W. Grant, “Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors,” *Inf. Sci.*, vol. 354, pp. 19–40, Aug. 2016, doi: 10.1016/j.ins.2016.03.011.

[30] M. Du, S. Ding, and H. Jia, “Study on density peaks clustering based on k-nearest neighbors and principal component analysis,” *Knowl.-Based Syst.*, vol. 99, pp. 135–145, May 2016, doi: 10.1016/j.knosys.2016.02.001.

[31] L. Scrucca and A. Raftery, “Improved initialisation of model-based clustering using Gaussian hierarchical partitions,” *Adv. Data Anal. Classif.*, vol. 9, no. 4, pp. 447–460, 2015, Accessed: Nov. 23, 2020. [Online]. Available: https://ideas.repec.org/a/spr/advdac/v9y2015i4p447-460.html

[32] A. O’Hagan and A. White, “Improved model-based clustering performance using Bayesian initialization averaging,” *Comput. Stat.*, vol. 34, no. 1, pp. 201–231, Mar. 2019, doi: 10.1007/s00180-018-0855-2.

[33] E. Kebriaei, K. Bijari, and H. Zare, “Improved model-based clustering using evolutionary optimization,” in *2017 Artificial Intelligence and Robotics (IRANOPEN)*,

Apr. 2017, pp. 182–187. doi: 10.1109/RIOS.2017.7956464.

[34] J. Chen, X. Lin, Q. Xuan, and Y. Xiang, “FGCH: a fast and grid based clustering algorithm for hybrid data stream,” *Appl. Intell.*, vol. 49, no. 4, pp. 1228–1244, Apr. 2019, doi: 10.1007/s10489-018-1324-x.

[35] B. Wu and B. M. Wilamowski, “A Fast Density and Grid Based Clustering Method for Data With Arbitrary Shapes and Noise,” *IEEE Trans. Ind. Inform.*, vol. 13, no. 4, pp. 1620–1628, Aug. 2017, doi: 10.1109/TII.2016.2628747.

[36] Q. Liu, K. Zhang, J. Shen, Z. Fu, and N. Linge, “GLRM: An improved grid-based load-balanced routing method for WSN with single controlled mobile sink,” in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, Jan. 2016, pp. 34–38. doi: 10.1109/ICACT.2016.7423264.

[37] L. Fu, P. Lin, A. V. Vasilakos, and S. Wang, “An overview of recent multi-view clustering,” *Neurocomputing*, vol. 402, pp. 148–161, Aug. 2020, doi: 10.1016/j.neucom.2020.02.104.

[38] Y. Yang and H. Wang, “Multi-view clustering: A survey,” *Big Data Min. Anal.*, vol. 1, no. 2, pp. 83–107, Jun. 2018, doi: 10.26599/BDMA.2018.9020003.

[39] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Sep. 1999, vol. 2, pp. 1150–1157 vol.2. doi: 10.1109/ICCV.1999.790410.

[40] A. Oliva and A. Torralba, “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope,” *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, May 2001, doi: 10.1023/A:1011139631724.

[41] L. Wang and D.-C. He, “Texture classification using texture spectrum,” *Pattern Recognit.*, vol. 23, no. 8, pp. 905–910, Jan. 1990, doi: 10.1016/0031-3203(90)90135-8.

[42] Q. Zheng, J. Zhu, and S. Ma, “Multi-view Hierarchical Clustering,” *ArXiv201007573 Cs*, Oct. 2020, Accessed: Aug. 25, 2021. [Online]. Available: http://arxiv.org/abs/2010.07573

[43] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, “Affinity aggregation for spectral clustering,” *2012 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 773-780, 2012, doi: 10.1109/CVPR.2012.6247748.

[44] A. Kumar, P. Rai, and H. Daume, “Co-regularized Multi-view Spectral Clusteri ng,” in *Advances in Neural Information Processing Systems*, 2011, vol. 24. Ac cessed: Aug. 19, 2021. [Online]. Available: https://papers.nips.cc/paper/2011/has h/31839b036f63806cba3f47b93af8ccb5-Abstract.html

[45] J. Wu, Z. Lin, and H. Zha, “Essential Tensor Learning for Multi-View Spectral Clustering,” *IEEE Trans. Image Process.*, vol. 28, no. 12, pp. 5910–5922, Dec. 2019, doi: 10.1109/TIP.2019.2916740.

[46] L. Zong, X. Zhang, X. Liu, and H. Yu, “Weighted Multi-View Spectral Clustering Based on Spectral Perturbation,” in *AAAI*, Vol. 32, No. 1, 2018.

[47] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, “Deep Subspace Clustering Networks,” in *Advances in Neural Information Processing Systems*, 2017, vol. 30. Accessed: Aug. 18, 2021. [Online]. Available: https://papers.nips.cc/paper/20 17/hash/e369853df766fa44e1ed0ff613f563bd-Abstract.html

[48] Z. Kang, W. Zhou, Z. Zhao, J. Shao, M. Han, and Z. Xu, “Large-scale Multi-view Subspace Clustering in Linear Time,” in *AAAI*, Vol. 34, No. 04, pp. 4412-4419, 2020. doi: 10.1609/AAAI.V34I04.5867.

[49] Q. Zheng, J. Zhu, Z. Tian, Z. Li, S. Pang, and X. Jia, “Constrained bilinear factorization multi-view subspace clustering,” *Knowl.-Based Syst.*, vol. 194, p. 105514, Apr. 2020, doi: 10.1016/j.knosys.2020.105514.

[50] Q. Zheng, J. Zhu, Z. Li, S. Pang, J. Wang, and Y. Li, “Feature concatenation multi-view subspace clustering,” *Neurocomputing*, vol. 379, pp. 89–102, Feb. 2020, doi: 10.1016/j.neucom.2019.10.074.

[51] O. Dababneh, T. Kipouros, and J. F. Whidborne, “Application of an Efficient Gradient-Based Optimization Strategy for Aircraft Wing Structures,” *Aerospace*, vol. 5, no. 1, Art. no. 1, Mar. 2018, doi: 10.3390/aerospace5010003.

[52] E. Pan and Z. Kang, "Multi-view Contrastive Graph Clustering," in *Advances in Neural Information Processing Systems*, 2021, vol. 34, pp. 2148–2159. Accessed: May 09, 2022. [Online]. Available: https://papers.nips.cc/paper/2021/hash/10c66082c124f8afe3df4886f5e516e0-Abstract.html

[53] S. Ouadfel and M. Abd Elaziz, "A multi-objective gradient optimizer approach-based weighted multi-view clustering," *Eng. Appl. Artif. Intell.*, vol. 106, p. 104480, Nov. 2021, doi: 10.1016/j.engappai.2021.104480.

[54] M. Nasser Al-Andoli, S. Chiang Tan, and W. Ping Cheah, "Distributed parallel deep learning with a hybrid backpropagation-particle swarm optimization for community detection in large complex networks," *Inf. Sci.*, vol. 600, pp. 94–117, Jul. 2022, doi: 10.1016/j.ins.2022.03.053.

[55] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 211–224, Jun. 2004, doi: 10.1109/TEVC.2004.826076.

[56] B. X. Nguyen, B. D. Nguyen, G. Carneiro, E. Tjiputra, Q. D. Tran, and T.-T. Do, "Deep Metric Learning Meets Deep Clustering: An Novel Unsupervised Approach for Feature Embedding," *ArXiv200904091 Cs*, Sep. 2020, Accessed: May 12, 2022. [Online]. Available: http://arxiv.org/abs/2009.04091

[57] P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. W. H. Freeman and Co., 1973.

[58] B. King, "Step-Wise Clustering Procedures," *J. Am. Stat. Assoc.*, vol. 62, no. 317, pp. 86–101, Mar. 1967, doi: 10.1080/01621459.1967.10482890.

[59] W.-B. Xie, Y.-L. Lee, C. Wang, D.-B. Chen, and T. Zhou, "Hierarchical clustering supported by reciprocal nearest neighbors," *Inf. Sci.*, vol. 527, pp. 279–292, Jul. 2020, doi: 10.1016/j.ins.2020.04.016.

[60] J. H. Ward, "Hierarchical Grouping to Optimize an Objective Function," *J. Am. Stat. Assoc.*, vol. 58, no. 301, pp. 236–244, Mar. 1963, doi: 10.1080/01621459.1963.10500845.

[61] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, “Cluster analysis and display of genome-wide expression patterns,” *Proc. Natl. Acad. Sci.*, vol. 95, no. 25, pp. 14863–14868, Dec. 1998, doi: 10.1073/pnas.95.25.14863.

[62] W. Zhang, X. Wang, D. Zhao, and X. Tang, “Graph Degree Linkage: Agglomerative Clustering on a Directed Graph,” in *Computer Vision – ECCV 2012*, Berlin, Heidelberg, 2012, pp. 428–441. doi: 10.1007/978-3-642-33718-5_31.

[63] M.-F. Balcan, Y. Liang, and P. Gupta, “Robust Hierarchical Clustering,” *J. Mach. Learn. Res.*, vol. 15, no. 118, pp. 4011–4051, 2014, Accessed: Feb. 19, 2022. [Online]. Available: http://jmlr.org/papers/v15/balcan14a.html

[64] P. D’Urso and V. Vitale, “A robust hierarchical clustering for georeferenced data,” *Spat. Stat.*, vol. 35, p. 100407, Mar. 2020, doi: 10.1016/j.spasta.2020.100407.

[65] D. Lam and D. C. Wunsch, “Chapter 20 - Clustering,” in *Academic Press Library in Signal Processing*, vol. 1, P. S. R. Diniz, J. A. K. Suykens, R. Chellappa, and S. Theodoridis, Eds. Elsevier, 2014, pp. 1115–1149. doi: 10.1016/B978-0-12-396502-8.00020-6.

[66] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 2009.

[67] R. T. Ng and J. Han, “CLARANS: a method for clustering objects for spatial data mining,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, Sep. 2002, doi: 10.1109/TKDE.2002.1033770.

[68] J. C. Bezdek, R. Ehrlich, and W. Full, “FCM: The fuzzy c-means clustering algorithm,” *Comput. Geosci.*, vol. 10, no. 2–3, pp. 191–203, Jan. 1984, doi: 10.1016/0098-3004(84)90020-7.

[69] A. Nagpal, A. Jatain, and D. Gaur, “Review based on data clustering algorithms,” in *2013 IEEE Conference on Information Communication Technologies*, Apr. 2013, pp. 298–303. doi: 10.1109/CICT.2013.6558109.

[70] A. Kalogeratos and A. Likas, “Dip-means: an incremental clustering method fo r estimating the number of clusters,” in *Advances in Neural Information Proce*

ssing Systems, 2012, vol. 25. Accessed: May 15, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/a8240cb8235e9c493a0c30607586166c-Abstract.html

[71] G. Tzortzis and A. Likas, “The global kernel k-means clustering algorithm,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Jun. 2008, pp. 1977–1984. doi: 10.1109/IJCNN.2008.4634069.

[72] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On Spectral Clustering: Analysis and an algorithm,” in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 849–856. Accessed: Mar. 05, 2020. [Online]. Available: http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf

[73] P. Bhattacharjee and P. Mitra, “A survey of density based clustering algorithms,” *Front. Comput. Sci.*, vol. 15, no. 1, p. 151308, Sep. 2020, doi: 10.1007/s11704-019-9059-3.

[74] M. Ester, H.-P. Kriegel, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Kdd*, 1996, vol. 96(34), pp. 226–231.

[75] A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks,” *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014, doi: 10.1126/science.1242072.

[76] R. Liu, H. Wang, and X. Yu, “Shared-nearest-neighbor-based clustering by fast search and find of density peaks,” *Inf. Sci.*, vol. 450, pp. 200–226, Jun. 2018, doi: 10.1016/j.ins.2018.03.031.

[77] M. d’Errico, E. Facco, A. Laio, and A. Rodriguez, “Automatic topography of high-dimensional data sets by non-parametric density peak clustering,” *Inf. Sci.*, vol. 560, pp. 476–492, Jun. 2021, doi: 10.1016/j.ins.2021.01.010.

[78] L. McInnes, J. Healy, and S. Astels, “hdbscan: Hierarchical density based clustering,” *J. Open Source Softw.*, vol. 2, p. 205, Mar. 2017, doi: 10.21105/joss.00205.

[79] Z. Akbari and R. Unland, “Automated Determination of the Input Parameter of

DBSCAN Based on Outlier Detection,” in *Artificial Intelligence Applications and Innovations*, Cham, 2016, pp. 280–291. doi: 10.1007/978-3-319-44944-9_24.

[80] J. Zhao *et al.*, “An Automatic Density Clustering Segmentation Method for Laser Scanning Point Cloud Data of Buildings,” *Math. Probl. Eng.*, vol. 2019, Jul. 2019, doi: 10.1155/2019/3026758.

[81] J. D. Banfield and A. E. Raftery, “Model-Based Gaussian and Non-Gaussian Clustering,” *Biometrics*, vol. 49, no. 3, pp. 803–821, 1993, doi: 10.2307/2532201.

[82] W. Wang, J. Yang, and R. Muntz, “STING : A Statistical Information Grid Approach to Spatial Data Mining,”, *Vldb*, Vol. 97, pp. 186-195, 1997.

[83] D. W. van der Merwe and A. P. Engelbrecht, “Data clustering using particle swarm optimization,” in *The 2003 Congress on Evolutionary Computation, 2003. CEC ’03.*, 2003, vol. 1, pp. 215-220 Vol.1. doi: 10.1109/CEC.2003.1299577.

[84] E. Elhamifar and R. Vidal, “Sparse Subspace Clustering: Algorithm, Theory, and Applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, 2013, doi: 10.1109/TPAMI.2013.57.

[85] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, “A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture,” *IEEE Access*, vol. 6, pp. 39501–39514, 2018, doi: 10.1109/ACCESS.2018.2855437.

[86] M. Brbić and I. Kopriva, “Multi-view low-rank sparse subspace clustering,” *Pattern Recognit.*, vol. 73, pp. 247–258, Jan. 2018, doi: 10.1016/j.patcog.2017.08.024.

[87] S. Huang, Z. Kang, and Z. Xu, “Auto-weighted multi-view clustering via deep matrix decomposition,” *Pattern Recognit.*, vol. 97, p. 107015, Jan. 2020, doi: 10.1016/j.patcog.2019.107015.

[88] Y. Jia, H. Liu, J. Hou, S. Kwong, and Q. Zhang, “Multi-View Spectral Clustering Tailored Tensor Low-Rank Representation,” *IEEE Trans. Circuits Syst. Video Technol.*, pp. 1–1, 2021, doi: 10.1109/TCSVT.2021.3055039.

[89] Y. Liang, D. Huang, and C.-D. Wang, “Consistency Meets Inconsistency: A Unified Graph Learning Framework for Multi-view Clustering,” in *2019 IEEE International*

Conference on Data Mining (ICDM), Nov. 2019, pp. 1204–1209. doi: 10.1109/ICDM.2019.00148.

[90] J. Lv, Z. Kang, B. Wang, L. Ji, and Z. Xu, "Multi-view subspace clustering via partition fusion," *Inf. Sci.*, vol. 560, pp. 410–423, Jun. 2021, doi: 10.1016/j.ins.2021.01.033.

[91] F. Nie, L. Tian, and X. Li, "Multiview Clustering via Adaptively Weighted Procrustes," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, NY, USA, Jul. 2018, pp. 2022–2030. doi: 10.1145/3219819.3220049.

[92] H. Wang, Y. Yang, and B. Liu, "GMC: Graph-Based Multi-View Clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1116–1129, Jun. 2020, doi: 10.1109/TKDE.2019.2903810.

[93] W. Xia, Q. Wang, Q. Gao, X. Zhang, and X. Gao, "Self-supervised Graph Convolutional Network for Multi-view Clustering," *IEEE Trans. Multimed.*, pp. 1–1, 2021, doi: 10.1109/TMM.2021.3094296.

[94] J. Xu, Y. Ren, G. Li, L. Pan, C. Zhu, and Z. Xu, "Deep embedded multi-view clustering with collaborative training," *Inf. Sci.*, vol. 573, pp. 279–290, Sep. 2021, doi: 10.1016/j.ins.2020.12.073.

[95] X. Yu, H. Liu, Y. Wu, and C. Zhang, "Fine-grained similarity fusion for Multi-view Spectral Clustering," *Inf. Sci.*, vol. 568, pp. 350–368, Aug. 2021, doi: 10.1016/j.ins.2021.03.059.

[96] G.-Y. Zhang, Y.-R. Zhou, C.-D. Wang, D. Huang, and X.-Y. He, "Joint representation learning for multi-view subspace clustering," *Expert Syst. Appl.*, vol. 166, p. 113913, Mar. 2021, doi: 10.1016/j.eswa.2020.113913.

[97] F. Ye, Z. Chen, H. Qian, R. Li, C. Chen, and Z. Zheng, "New Approaches i n Multi-View Clustering," *Recent applications in data clustering*, Vol. 195, 20 18, doi: 10.5772/intechopen.75598.

[98] C. Tang, Z. Li, J. Wang, X. Liu, W. Zhang, and E. Zhu, "Unified One-step Multi-view Spectral Clustering," *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2022, doi:

10.1109/TKDE.2022.3172687.

[99] S. Shi, F. Nie, R. Wang, and X. Li, “Self-weighting multi-view spectral clustering based on nuclear norm,” *Pattern Recognit.*, vol. 124, p. 108429, Apr. 2022, doi: 10.1016/j.patcog.2021.108429.

[100] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, “Robust Recovery of Subspace Structures by Low-Rank Representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013, doi: 10.1109/TPAMI.2012.88.

[101] C. Zhang, Q. Hu, H. Fu, P. Zhu, and X. Cao, “Latent Multi-view Subspace Clustering,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 4333–4341. doi: 10.1109/CVPR.2017.461.

[102] G.-Y. Zhang, X.-W. Chen, Y.-R. Zhou, C.-D. Wang, D. Huang, and X.-Y. He, “Kernelized multi-view subspace clustering via auto-weighted graph learning,” *Appl. Intell.*, vol. 52, no. 1, pp. 716–731, Jan. 2022, doi: 10.1007/s10489-021-02365-8.

[103] H. Zhao, Z. Ding, and Y. Fu, “Multi-view clustering via deep matrix factorization,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, Feb. 2017, pp. 2921–2927.

[104] X. Peng, Z. Huang, J. Lv, H. Zhu, and J. T. Zhou, “COMIC: Multi-view Clustering Without Parameter Selection,” in *International Conference on Machine Learning*, May 2019, pp. 5092–5101. Accessed: Aug. 25, 2021. [Online]. Available: https://proceedings.mlr.press/v97/peng19a.html

[105] T. M. Cover, “Rates of convergence for nearest neighbor procedures,” presented at *Proceedings of the Hawaii international conference on system sciences*, Vol. 415, 1968.

[106] D. Li and Y. Tian, “Survey and experimental study on metric learning methods,” *Neural Netw.*, vol. 105, pp. 447–462, Sep. 2018, doi: 10.1016/j.neunet.2018.06.003.

[107] Fukunaga, K., “Introduction to Statistical Pattern Recognition,” Elsevier, 2013.

[108] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, “Adjustment Learning and Relevant Component Analysis,” in *Proceedings of the 7th European Conference on Computer Vision-Part IV*, Berlin, Heidelberg, May 2002, pp. 776–792.

[109] F. Wang and C. Zhang, “Feature Extraction by Maximizing the Average Neighborhood Margin,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–8. doi: 10.1109/CVPR.2007.383124.

[110] E. Xing, M. Jordan, S. J. Russell, and A. Ng, “Distance Metric Learning with Application to Clustering with Side-Information,” in *Advances in Neural Information Processing Systems*, 2002, vol. 15. Accessed: Apr. 30, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2002/hash/c3e4035af2a1cde9f21e1ae1951ac80b-Abstract.html

[111] M. Kaya and H. Ş. Bilge, “Deep Metric Learning: A Survey,” *Symmetry*, vol. 11, no. 9, Art. no. 9, Sep. 2019, doi: 10.3390/sym11091066.

[112] E. Hoffer and N. Ailon, “Deep Metric Learning Using Triplet Network,” in *Similarity-Based Pattern Recognition*: *Third International Workshop, SIMBAD 2015, Copenhagen, Denmark*, pp. 84–92, 2015, doi: 10.1007/978-3-319-24261-3_7.

[113] M. Bilenko, S. Basu, and R. J. Mooney, “Integrating constraints and metric learning in semi-supervised clustering,” in *Proceedings of the twenty-first international conference on Machine learning*, New York, NY, USA, Jul. 2004, p. 11. doi: 10.1145/1015330.1015360.

[114] M. S. Baghshah and S. B. Shouraki, “Semi-supervised metric learning using pairwise constraints,” in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, Jul. 2009, pp. 1217–1222.

[115] Y. Liang, K. Maeda, T. Ogawa, and M. Haseyama, “Cross-Domain Semi-Supervised Deep Metric Learning for Image Sentiment Analysis,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 4150–4154. doi: 10.1109/ICASSP39728.2021.9414150.

[116] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemom. Intell. Lab. Syst.*, vol. 2, no. 1, pp. 37–52, Aug. 1987, doi: 10.1016/0169-7439(87)80084-9.

[117] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008, Accessed: Apr. 29, 2022. [Online]. Available:

http://jmlr.org/papers/v9/vandermaaten08a.html

[118] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform Manifold Approximation and Projection," *J. Open Source Softw.*, vol. 3, no. 29, p. 861, Sep. 2018, doi: 10.21105/joss.00861.

[119] X. Cao, B.-C. Chen, and S.-N. Lim, "Unsupervised Deep Metric Learning via Auxiliary Rotation Loss," arXiv, arXiv:1911.07072, Nov. 2019. doi: 10.48550/arXiv.1911.07072.

[120] U. K. Dutta, M. Harandi, and C. C. Sekhar, "Unsupervised Deep Metric Learning via Orthogonality Based Probabilistic Loss," *IEEE Trans. Artif. Intell.*, vol. 1, no. 1, pp. 74–84, 2020, doi: 10.1109/TAI.2020.3026982.

[121] G. De Lucia and J. Blaizot, "The hierarchical formation of the brightest cluster galaxies," *Mon. Not. R. Astron. Soc.*, vol. 375, no. 1, pp. 2–14, Feb. 2007, doi: 10.1111/j.1365-2966.2006.11287.x.

[122] M. S. Petersen, M. D. Weinberg, and N. Katz, "Using torque to understand barred galaxy models," *Mon. Not. R. Astron. Soc.*, vol. 490, no. 3, pp. 3616–3632, Dec. 2019, doi: 10.1093/mnras/stz2824.

[123] E. R. Stanway *et al.*, "Exploring the cosmic evolution of habitability with galaxy merger trees," *Mon. Not. R. Astron. Soc.*, vol. 475, no. 2, pp. 1829–1842, Apr. 2018, doi: 10.1093/mnras/stx3305.

[124] S. Cole, C. G. Lacey, C. M. Baugh, and C. S. Frenk, "Hierarchical galaxy formation," *Mon. Not. R. Astron. Soc.*, vol. 319, no. 1, pp. 168–204, Nov. 2000, doi: 10.1046/j.1365-8711.2000.03879.x.

[125] K. R. V. Casteels *et al.*, "Galaxy And Mass Assembly (GAMA): refining the local galaxy merger rate using morphological information," *Mon. Not. R. Astron. Soc.*, vol. 445, no. 2, pp. 1157–1169, Dec. 2014, doi: 10.1093/mnras/stu1799.

[126] O. Bertolami, F. Gil Pedro, and M. Le Delliou, "Dark energy–dark matter interaction and putative violation of the equivalence principle from the Abell cluster A586," *Phys. Lett. B*, vol. 654, no. 5, pp. 165–169, Oct. 2007, doi: 10.1016/j.physletb.2007.08.046.

[127] S. C. Johnson, “Hierarchical clustering schemes,” *Psychometrika*, vol. 32, no. 3, pp. 241–254, Sep. 1967, doi: 10.1007/BF02289588.

[128] Y. Jeon, J. Yoo, J. Lee, and S. Yoon, “NC-Link: A New Linkage Method for Efficient Hierarchical Clustering of Large-Scale Data,” *IEEE Access*, vol. 5, pp. 5594–5608, 2017, doi: 10.1109/ACCESS.2017.2690987.

[129] Jianbo Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000, doi: 10.1109/34.868688.

[130] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975, doi: 10.1109/TIT.1975.1055330.

[131] B. J. Frey and D. Dueck, “Clustering by Passing Messages Between Data Points,” *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007, doi: 10.1126/science.1136800.

[132] H. Averbuch-Elor, N. Bar, and D. Cohen-Or, “Border-Peeling Clustering,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1791–1797, Jul. 2020, doi: 10.1109/TPAMI.2019.2924953.

[133] A. Strehl and J. Ghosh, “Cluster Ensembles --- A Knowledge Reuse Framewo rk for Combining Multiple Partitions,” *J. Mach. Learn. Res.*, vol. 3, no. Dec, pp. 583–617, 2002, Accessed: Mar. 20, 2020. [Online]. Available: http://www.j mlr.org/papers/v3/strehl02a.html

[134] B. Tomas, “Clustering benchmarks,” 2019. https://github.com/deric/clustering-benchmark

[135] P. Fränti and O. Virmajoki, “Iterative shrinking method for clustering problems,” *Pattern Recognit.*, vol. 39, no. 5, pp. 761–775, May 2006, doi: 10.1016/j.patcog.2005.09.012.

[136] L. Fu and E. Medico, “FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data,” *BMC Bioinformatics*, vol. 8, no. 1, p. 3, Jan. 2007, doi: 10.1186/1471-2105-8-3.

[137] H. Chang and D.-Y. Yeung, “Robust path-based spectral clustering,” *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, Jan. 2008, doi: 10.1016/j.patcog.2007.04.010.

[138] S. Guha, R. Rastogi, and K. Shim, “Cure: an efficient clustering algorithm for large databases,” *Inf. Syst.*, vol. 26, no. 1, pp. 35–58, Mar. 2001, doi: 10.1016/S0306-4379(01)00008-4.

[139] M.-C. Su, C.-H. Chou, and C.-C. Hsieh, “Fuzzy C-Means Algorithm with a Point Symmetry Distance,” *Int. J. Fuzzy Syst.*, no. 7(4), pp. 175–181, 2005.

[140] M. H. C. Law, A. P. Topchy, and A. K. Jain, “Multiobjective data clustering,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Jun. 2004, vol. 2, p. II–II. doi: 10.1109/CVPR.2004.1315194.

[141] A. Garcia-Piquer, A. Sancho-Asensio, A. Fornells, E. Golobardes, G. Corral, and F. Teixidó-Navarro, “Toward high performance solution retrieval in multiobjective clustering,” *Inf. Sci.*, vol. 320, pp. 12–25, Nov. 2015, doi: 10.1016/j.ins.2015.04.041.

[142] H. Julia and K. Joshua, “Multiobjective clustering with automatic determination of the number of clusters,” *Tech. Rep.*, 2004.

[143] K. Faceli, T. C. Sakata, M. C. P. de Souto, and A. C. P. L. F. de Carvalho, “Partitions selection strategy for set of clustering solutions,” *Neurocomputing*, vol. 73, no. 16, pp. 2809–2819, Oct. 2010, doi: 10.1016/j.neucom.2010.03.028.

[144] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, no. 86(11), pp. 2278–2324, 1998.

[145] F. S. Samaria and A. C. Harter, “Parameterisation of a stochastic model for human face identification,” in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, Dec. 1994, pp. 138–142. doi: 10.1109/ACV.1994.341300.

[146] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, “Complex Wavelet Structural Similarity: A New Image Similarity Index,” *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2385–2401, Nov. 2009, doi: 10.1109/TIP.2009.2025923.

[147] S. A. Nene, S. K. Nayar, and H. Murase, “Columbia Object Image Library (COIL-

100)," *Tech. Rep.*, no. CUCS-006-96, 1996.

[148] T. Sim, S. Baker, and M. Bsat, "The CMU Pose, Illumination, and Expression (PIE) database," in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, May 2002, pp. 53–58. doi: 10.1109/AFGR.2002.1004130.

[149] The Cancer Genome Atlas Research Network *et al.*, "The Cancer Genome Atlas Pan-Cancer analysis project," *Nat. Genet.*, vol. 45, no. 10, pp. 1113–1120, Oct. 2013, doi: 10.1038/ng.2764.

[150] A. Dogan, "Cell-Tracking-Analysis-with-K-Means-Clustering-Method." https://github.com/ddaskan/Cell-Tracking-Analysis-with-K-Means-Clustering-Method

[151] "Zoo Data Set - UCI Machine Learning Repository." https://archive.ics.uci.edu/ml/datasets/zoo

[152] Tan, M., and Eshelman, L. "Using Weighted Networks to Represent Classification Knowledge in Noisy Domains," *Mach. Learn. Proc. 1988*, pp. 121–134, Jan. 1988, doi: 10.1016/B978-0-934613-64-4.50018-9.

[153] S. J. Haberman, "Generalized residuals for log-linear models," presented at the *In Proceedings of the 9th international biometrics conference*, 1976, pp. 104–122.

[154] A. Ultsch, "Strategies for an artificial life system to cluster high dimensional data," *Abstr. Synth. Princ. Living Syst.*, vol. GWAL-6, pp. 128–137, 2004.

[155] M. C. Thrun and A. Ultsch, "Clustering benchmark datasets exploiting the fundamental clustering problems," *Data Brief*, vol. 30, p. 105501, Jun. 2020, doi: 10.1016/j.dib.2020.105501.

[156] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," *ACM SIGKDD Explor. Newsl.*, vol. 4, no. 1, pp. 65–75, Jun. 2002, doi: 10.1145/568574.568575.

[157] M.-F. Balcan, A. Blum, and S. Vempala, "A discriminative framework for clustering via similarity functions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, Victoria, British Columbia, Canada, May 2008, pp. 671–680. doi: 10.1145/1374376.1374474.

[158] D. B. Graham and N. M. Allinson, “Characterising Virtual Eigensignatures for General Purpose Face Recognition,” in *Face Recognition: From Theory to Applications*, H. Wechsler, P. J. Phillips, V. Bruce, F. F. Soulié, and T. S. Huang, Eds. Berlin, Heidelberg: Springer, 1998, pp. 446–456. doi: 10.1007/978-3-642-72201-1_25.

[159] “Face Recognition Grand Challenge (FRGC v.2.0) data collection.” [Online]. Available: https://cvrl.nd.edu/projects/data/#face-recognition-grand-challenge-frgc-v20-data-collection

[160] S. A. Nene, S. K. Nayar, and H. Murase, “Columbia Object Image Library (COIL-20),” *Tech. Rep.*, vol. Technical Report CUCS-005-96, 1996.

[161] F. Alimoglu and E. Alpaydin, “Combining multiple representations and classifiers for pen-based handwritten digit recognition,” in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Aug. 1997, vol. 2, pp. 637–640 vol.2. doi: 10.1109/ICDAR.1997.620583.

[162] J. Yang, D. Parikh, and D. Batra, “Joint Unsupervised Learning of Deep Representations and Image Clusters,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5147–5156. doi: 10.1109/CVPR.2016.556.

[163] D. Lim, R. Vidal, and B. Haeffele, “Doubly Stochastic Subspace Clustering,” *ArXiv*, arXiv:2011.14859, 2020.

[164] Z. Wang, Y. Ni, B. Jing, D. Wang, H. Zhang, and E. Xing, “DNB: A Joint Learning Framework for Deep Bayesian Nonparametric Clustering,” *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, 2021, doi: 10.1109/TNNLS.2021.3085891.

[165] Q. Huang, Y. Zhang, H. Peng, T. Dan, W. Weng, and H. Cai, “Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning,” *Neurocomputing*, vol. 404, pp. 340–350, Sep. 2020, doi: 10.1016/j.neucom.2020.04.120.

[166] X. Huang, Z. Hu, and L. Lin, “Deep clustering based on embedded auto-encoder,” *Soft Comput.*, vol. 27, no. 2, pp. 1075-1090, 2023, doi: 10.1007/s00500-021-05934-8.

[167] W. Wang, F. Chen, Y. Ge, S. Huang, X. Zhang, and D. Yang, “Discriminative deep semi-nonnegative matrix factorization network with similarity maximization for unsupervised feature learning,” *Pattern Recognit. Lett.*, vol. 149, pp. 157–163, Sep. 2021, doi: 10.1016/j.patrec.2021.06.013.

[168] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, “Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization,” presented at the 2017 IEEE International Conference on Computer Vision (ICCV), Oct. 2017, pp. 5747–5756. doi: 10.1109/ICCV.2017.612.

[169] Z. Yu, Z. Zhang, W. Cao, C. Liu, J. Philip Chen, and H. S. Wong, “GAN-based Enhanced Deep Subspace Clustering Networks,” *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2020, doi: 10.1109/TKDE.2020.3025301.

[170] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock, “N2D: (Not Too) Deep Clustering via Clustering the Local Manifold of an Autoenco ded Embedding,” *ArXiv E-Prints*, vol. 1908, p. arXiv:1908.05968, Aug. 2019, Accessed: Feb. 11, 2021. [Online]. Available: http://adsabs.harvard.edu/abs/2019 arXiv190805968M

[171] M. Yang and S. Xu, “Orthogonal Nonnegative Matrix Factorization using a novel deep Autoencoder Network,” *Knowl.-Based Syst.*, vol. 227, p. 107236, Sep. 2021, doi: 10.1016/j.knosys.2021.107236.

[172] Z. Chen, S. Ding, and H. Hou, “A novel self-attention deep subspace clustering,” *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 8, pp. 2377–2387, 2021, doi: 10.1007/s13042-021-01318-4.

[173] M. Jabi, M. Pedersoli, A. Mitiche, and I. B. Ayed, “Deep Clustering: On the Link Between Discriminative Models and K-Means,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 06, pp. 1887–1896, Jun. 2021, doi: 10.1109/TPAMI.2019.2962683.

[174] H. Li, X. Ye, A. Imakura, and T. Sakurai, “Divide-and-conquer based Large-Scale Spectral Clustering,” *Neurocomputing*, vol. 501, pp. 664-678, 2022.

[175] J. Wang and J. Jiang, “Unsupervised deep clustering via adaptive GMM modeling and

optimization," *Neurocomputing*, vol. 433, pp. 199–211, Apr. 2021, doi: 10.1016/j.neucom.2020.12.082.

[176] Z. Kang, X. Lu, J. Liang, K. Bai, and Z. Xu, "Relation-Guided Representation Learning," *Neural Netw.*, vol. 131, pp. 93–102, Nov. 2020, doi: 10.1016/j.neunet.2020.07.014.

[177] C. Leiber, L. G. M. Bauer, B. Schelling, C. Böhm, and C. Plant, "Dip-based Deep Embedded Clustering with k-Estimation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 903–913. Accessed: Feb. 14, 2022. [Online]. Available: https://doi.org/10.1145/3447548.3467316

[178] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognit.*, vol. 83, pp. 161–173, Nov. 2018, doi: 10.1016/j.patcog.2018.05.019.

[179] P. Zhou, Y. Hou, and J. Feng, "Deep Adversarial Subspace Clustering," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 1596–1604. doi: 10.1109/CVPR.2018.00172.

[180] W. Hu, C. Chen, F. Ye, Z. Zheng, and Y. Du, "Learning deep discriminative representations with pseudo supervision for image clustering," *Inf. Sci.*, vol. 568, pp. 199–215, Aug. 2021, doi: 10.1016/j.ins.2021.03.066.

[181] L. Wu, Z. Liu, J. Xia, Z. Zang, S. Li, and S. Z. Li, "Generalized Clustering and Multi-Manifold Learning with Geometric Structure Preservation," presented at the *Proceedings of the IEEE/CVF Winter Conference on Applications of C omputer Vision*, 2022, pp. 139–147. Accessed: Feb. 15, 2022. [Online]. Availa ble: https://openaccess.thecvf.com/content/WACV2022/html/Wu_Generalized_Clust ering_and_Multi-Manifold_Learning_With_Geometric_Structure_Preservation_WA CV_2022_paper.html

[182] X. Li, X. Zhao, D. Chu, and Z. Zhou, "An autoencoder-based spectral clustering algorithm," *Soft Comput.*, vol. 24, no. 3, pp. 1661–1671, 2020, doi: 10.1007/s00500-

019-03994-5.

[183] T. Zhu and J.-T. Zhang, "Cosine Similarity-Based Classifiers for Functional Data," in *Contemporary Experimental Design, Multivariate Analysis and Data Mining: Festschrift in Honour of Professor Kai-Tai Fang*, J. Fan and J. Pan, Eds. Cham: Springer International Publishing, 2020, pp. 277–292. doi: 10.1007/978-3-030-46161-4_18.

[184] B. Li and L. Han, "Distance Weighted Cosine Similarity Measure for Text Classification," in *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, Berlin, Heidelberg, 2013, pp. 611–618. doi: 10.1007/978-3-642-41278-3_74.

[185] H. V. Nguyen and L. Bai, "Cosine Similarity Metric Learning for Face Verification," in *Computer Vision – ACCV 2010*, Berlin, Heidelberg, 2011, pp. 709–720. doi: 10.1007/978-3-642-19309-5_55.

[186] de França, F. O., "A hash-based co-clustering algorithm for categorical data," *Expert Syst. Appl.*, vol. 64, pp. 24–35, Dec. 2016, doi: 10.1016/j.eswa.2016.07.024.

[187] C. Michael, "Perform EM algorithm for fitting the Gaussian mixture model." http://www.ccs.neu.edu/home/vip/teach/MLcourse/3_generative_models/HW3/emgm.m.pdf

[188] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, Jun. 1998, doi: 10.1023/A:1009745219419.

[189] L. Bai, X. Cheng, J. Liang, H. Shen, and Y. Guo, "Fast density clustering strategies based on the k-means algorithm," *Pattern Recognit.*, vol. 71, pp. 375–386, Nov. 2017, doi: 10.1016/j.patcog.2017.06.023.

[190] F. Bryan, "Perform MeanShift Clustering of data using a flat kernel." https://ww2.mathworks.cn/matlabcentral/fileexchange/52698-k-means-mean-shift-and-normalized-cut-segmentation

[191] A. Mayer and H. Greenspan, "An Adaptive Mean-Shift Framework for MRI Brain Segmentation," *IEEE Trans. Med. Imaging*, vol. 28, no. 8, pp. 1238–1250, Aug. 2009,

doi: 10.1109/TMI.2009.2013850.

[192] "APCLUSTER Affinity Propagation Clustering (Frey/Dueck, Science 2007)." [Online]. Available: https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/52053/versions/1/previews/apcluster.m/index.html

[193] R. A. and L. A., "Clustering by fast search-and-find of density peaks." https://people.sissa.it/~laio/Research/Res_clustering.php (accessed Nov. 27, 2020).

[194] S. Sarfraz, "Efficient Parameter-free Clustering Using First Neighbor Relations, https://github.com/ssarfraz/FINCH-Clustering." Nov. 26, 2020. Accessed: Nov. 27, 2020. [Online]. Available: https://github.com/ssarfraz/FINCH-Clustering

[195] "A python implementation of Robust Continuous Clustering." [Online]. Available: https://github.com/yhenon/pyrcc

[196] "Border-Peeling Clustering." [Online]. Available: https://github.com/nadavbar/BorderPeelingClustering

[197] S. A. Seyedi, "Dynamic Graph-Based Label Propagation for Density Peaks Clustering, https://github.com/amjadseyedi/DPC-DLP." Oct. 05, 2020. Accessed: Nov. 27, 2020. [Online]. Available: https://github.com/amjadseyedi/DPC-DLP

[198] "Density Peaks Advanced clustering." [Online]. Available: https://github.com/mariaderrico/DPA

[199] "Shared Nearest Neighbor-based Clustering by Fast Search and Find of Density Peaks." [Online]. Available: https://github.com/liurui39660/SNNDPC/tree/MatlabImplementation

[200] C. Mo, "Normalized mutual information." https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/35625/versions/2/previews/InfoTheory/nmi.m/index.html

[201] W. Lingfei, "Accuracy." https://github.com/IBM/SpectralClustering_RandomBinning/blob/master/accuracy.m

[202] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based

Approaches," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012, doi: 10.1109/TSMCC.2011.2161285.

[203] X. Liu, H.-M. Cheng, and Z.-Y. Zhang, "Evaluation of Community Detection Methods," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 9, pp. 1736–1746, Sep. 2020, doi: 10.1109/TKDE.2019.2911943.

[204] M. C. de Souto, I. G. Costa, D. S. de Araujo, T. B. Ludermir, and A. Schliep, "Clustering cancer gene expression data: a comparative study," *BMC Bioinformatics*, vol. 9, no. 1, p. 497, Nov. 2008, doi: 10.1186/1471-2105-9-497.

[205] J. Moore and M. Ackerman, "Foundations of Perturbation Robust Clustering," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 1089–1094. doi: 10.1109/ICDM.2016.0141.

[206] V. Menon, G. Muthukrishnan, and S. Kalyani, "Subspace Clustering Without Knowing the Number of Clusters: A Parameter Free Approach," *IEEE Trans. Signal Process.*, vol. 68, pp. 5047–5062, 2020, doi: 10.1109/TSP.2020.3018665.

[207] T. Kameda *et al.*, "Rawlsian maximin rule operates as a common cognitive anchor in distributive justice and risky decisions," *Proc. Natl. Acad. Sci.*, vol. 113, no. 42, pp. 11817–11822, Oct. 2016, doi: 10.1073/pnas.1602641113.

[208] P. Zhao, Y. Jiang, and Z.-H. Zhou, "Multi-View Matrix Completion for Clustering with Side Information," in *Advances in Knowledge Discovery and Data Mining*, Cham, 2017, pp. 403–415. doi: 10.1007/978-3-319-57529-2_32.

[209] F. Nie, J. Li, and X. Li, "Self-weighted Multiview Clustering with Multiple Graphs," In *IJCAI*, pp. 2564–2570, 2017, Accessed: Aug. 15, 2021. [Online]. Available: https://www.ijcai.org/proceedings/2017/357

[210] H. Tao, C. Hou, X. Liu, T. Liu, D. Yi, and J. Zhu, "Reliable Multi-View Clustering," In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[211] L. Ertoz, M. Steinbach, and V. Kumar, "A New Shared Nearest Neighbor Clu stering Algorithm and its Applications," *2nd SIAM Int. Conf. Data Min.*, Vol. 8, 2002, Accessed: Aug. 25, 2021. [Online]. Available: https://www.semanticsc

holar.org/paper/A-New-Shared-Nearest-Neighbor-Clustering-Algorithm-Ertoz-Steinbach/a46d877a1b7948e7b62ad1ea0bab28269bfcae6b

[212] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, “Understanding of Internal Clustering Validation Measures,” in *2010 IEEE International Conference on Data Mining*, Dec. 2010, pp. 911–916. doi: 10.1109/ICDM.2010.35.

[213] A. Bakshi and D. P. Woodruff, “Sublinear time low-rank approximation of distance matrices,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2018, pp. 3786–3796.

[214] C. Lu, S. Yan, and Z. Lin, “Convex Sparse Spectral Clustering: Single-View to Multi-View,” *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2833–2843, Jun. 2016, doi: 10.1109/TIP.2016.2553459.

[215] X. Cai, F. Nie, and H. Huang, “Multi-View K-Means Clustering on Big Data,” in *IJCAI*, pp. 2598–2604, 2013.

[216] J. Han, J. Xu, F. Nie, and X. Li, “Multi-view K-Means Clustering with Adaptive Sparse Memberships and Weight Allocation,” *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2020, doi: 10.1109/TKDE.2020.2986201.

[217] X. Fang, Y. Hu, P. Zhou, and D. O. Wu, “V3H: View Variation and View Heredity for Incomplete Multiview Clustering,” *IEEE Trans. Artif. Intell.*, vol. 1, no. 03, pp. 233–247, Jul. 2020, doi: 10.1109/TAI.2021.3052425.

[218] K. Zhan, F. Nie, J. Wang, and Y. Yang, “Multiview Consensus Graph Clustering,” *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1261–1270, Mar. 2019, doi: 10.1109/TIP.2018.2877335.

[219] R. Xia, Y. Pan, L. Du, and J. Yin, “Robust Multi-View Spectral Clustering via Low-Rank and Sparse Decomposition,” *Proc. AAAI Conf. Artif. Intell.*, vol. 28, no. 1, Art. no. 1, Jun. 2014, Accessed: Aug. 19, 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8950

[220] D. M. Powers, “Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation,” *Journal of Machine Learning*

Technologies, vol. 2, no. 1, pp. 37–63, Dec. 2011, Accessed: Oct. 20, 2019. [Online]. Available: https://dspace.flinders.edu.au/xmlui/handle/2328/27165

[221] J. C. Dunn, “A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters,” *J. Cybern.*, vol. 3, no. 3, pp. 32–57, Jan. 1973, doi: 10.1080/01969727308546046.

[222] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987, doi: 10.1016/0377-0427(87)90125-7.

[223] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, and S. Wu, “Understanding and Enhancement of Internal Clustering Validation Measures,” *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 982–994, Jun. 2013, doi: 10.1109/TSMCB.2012.2220543.

[224]L. Hu and C. Zhong, “An Internal Validity Index Based on Density-Involved Distance,” *IEEE Access*, vol. 7, pp. 40038–40051, 2019, doi: 10.1109/ACCESS.2019.2906949.

[225] W. Hao, S. Pang, and Z. Chen, “Multi-view spectral clustering via common structure maximization of local and global representations,” *Neural Netw.*, vol. 143, pp. 595–606, Nov. 2021, doi: 10.1016/j.neunet.2021.07.020.

[226] A. L. Ballardini, “A tutorial on Particle Swarm Optimization Clustering,” *ArXiv180901942 Cs*, Sep. 2018, Accessed: Jan. 11, 2022. [Online]. Available: http://arxiv.org/abs/1809.01942

[227] X. Zhu, J. Shang, Y. Sun, F. Li, J.-X. Liu, and S. Yuan, “PSO-CFDP: A Particle Swarm Optimization-Based Automatic Density Peaks Clustering Method for Cancer Subtyping,” *Hum. Hered.*, vol. 84, no. 1, pp. 9–20, 2019, doi: 10.1159/000501481.

[228] R. C. Prim, “Shortest connection networks and some generalizations,” *Bell Syst. Tech. J.*, vol. 36, no. 6, pp. 1389–1401, 1957, doi: 10.1002/j.1538-7305.1957.tb01515.x.

[229] D. Dua and C. Graff, “UCI Machine Learning Repository.” 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[230] F. Wang and J. Sun, “Survey on distance metric learning and dimensionality reduction in data mining,” *Data Min. Knowl. Discov.*, vol. 29, no. 2, pp. 534–564, Mar. 2015,

doi: 10.1007/s10618-014-0356-z.

[231] A. Gupta, D. Foster, and L. Ungar, “Unsupervised Distance Metric Learning Using Predictability,” *Tech. Rep. CIS*, vol. 885, Jun. 2008, [Online]. Available: https://repository.upenn.edu/cis_reports/885

[232] D. Simovici and K. Hua, “Data ultrametricity and clusterability,” *J. Phys. Conf. Ser.*, vol. 1334, no. 1, p. 012002, 2019, doi: 10.1088/1742-6596/1334/1/012002.

[233] M. Ackerman, S. Ben-David, S. Brânzei, and D. Loker, “Weighted Clustering,” *Proc. AAAI Conf. Artif. Intell.*, vol. 26, no. 1, pp. 858-863, 2012, Accessed: Apr. 25, 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8282

[234] M. Ackerman and S. Dasgupta, “Incremental clustering: the case for extra clusters,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, Cambridge, MA, USA, Dec. 2014, pp. 307–315.

[235] Z. Yu *et al.*, “Transitive Distance Clustering with K-Means Duality,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 987–994. doi: 10.1109/CVPR.2014.131.

[236] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, Nov. 1995, vol. 4, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.