Big Data Cluster Analysis and its Applications

Punit Rathore

Submitted in partial fulfilment of the requirements of the degree of

Doctor of Philosophy

Department of Electrical and Electronic Engineering
THE UNIVERSITY OF MELBOURNE

August 2018

Abstract

The increasing prevalence of Internet of things (IoT) technologies, smartphones, and social media services generates a huge amount of data, popularly known as 'big data'. Extracting useful information from big data is essential for many businesses and applications for providing better services and increasing their profits. For example, smart city solutions aim to use this wealth of data for formulating effective policies to solve the problems faced by citizens. These voluminous data are usually unlabeled, therefore, scalable and efficient unsupervised algorithms are required to manage and extract actionable information from big data.

Cluster analysis is a useful unsupervised approach to discover the underlying groups and useful patterns in the data. Cluster Analysis for any data consists of three problems, (P1) cluster assessment, which asks "Do the data have clusters? If yes, how many?"; (P2) Clustering i.e., partitioning the data into clusters, and (P3) cluster validity, which asks "Are the clusters found useful? Is there a better one we did not find?" Traditional cluster analysis algorithms are not suitable for big data owing to its volume, variety, and velocity property.

This thesis developed a suite of novel scalable algorithms to solve each of the three problems of cluster analysis, namely, cluster assessment, clustering, and cluster validity, for big data, that may be high-dimensional, anomalous and streaming. For demonstration, a novel scalable framework for predicting large-scale taxi trajectories is presented as a real application of big data clustering.

Our first contribution addresses the high-dimensionality and scalability issues for soft clustering methods. Specifically, we developed a simple and computationally efficient framework for high-dimensional data clustering: CAFCM, which employs fuzzy c-means clustering on an ensemble of random projections to obtain multiple fuzzy clustering partitions, and then cumulatively aggregates them based on their quality to get a final output partition. The CAFCM framework scales linearly in the number of samples in the data and does not require any prior knowledge of

the number of clusters, which makes it an attractive clustering approach for big datasets.

Our second contribution solves the cluster tendency assessment and clustering problem for voluminous, high-dimensional datasets. We developed a fast cluster tendency assessment and subsequent clustering algorithm: FensiVAT, which integrates an intelligent sampling scheme, called *Maximin Random Sampling* (MMRS), and a new random projection (RP)-based ensemble method with a *visual assessment of cluster tendency* (VAT) method, in an efficient manner. The *reordered dissimilarity image* (RDI) (aka cluster heat map) obtained in FensiVAT suggests the number of clusters in data. The FensiVAT is more effective than the existing big data clustering techniques, both in terms of CPU-time and cluster quality.

Our third contribution deals with the cluster validity problem for big data. Notably, we presented six novel approximation algorithms including two incremental methods to compute Dunn's cluster validity index for big data. Four methods used variations of the MMRS sampling and two are based on unsupervised training of one class support vector machines. All six methods for estimation of Dunn's index (DI) are linear in the number of samples. Computing approximations to DI with MMRS methods is both tractable and accurate.

After dealing with big static data, our next contribution focused on detecting evolving structure in high-velocity, streaming data. Existing VAT-based algorithms for streaming data, inc-VAT/ inc-iVAT and dec-VAT/dec-iVAT, are impractical for high-velocity data streams. We developed a novel algorithm, inc-siVAT, for incremental and time efficient visualization of evolving cluster structures in high-velocity, data streams. The inc-siVAT extracts an initial smart (MMRS) sample and its RDI image, then it incrementally updates them on the fly to track changes in cluster structure after each chunk. The new algorithm is demonstrated for visualizing evolving cluster structures and detecting anomalies in dynamic streams of four big datasets, including a real IoT data.

Finally, we demonstrate our big data clustering framework for a real-life smart city application. Based on a big data clustering method and Markov models, we developed a scalable framework for vehicle trajectory prediction which is suitable for a large number of overlapping trajectories in a dense road network, typically for major cities around the world. The short-term and long-term prediction performance of our framework on two real-life, large-scale taxi trajectory data from the Beijing and Singapore Road networks is found to be better than two current methods, in terms of prediction accuracy and distance error.

This page intentionally left blank.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Punit Rathore, August 2018

This page intentionally left blank.

Preface

All the work in this thesis was conducted by the author of this thesis including theoretical analysis, algorithm development, experiments, and manuscript writing. However, the author benefited from his supervisors through group meeting sessions in which they provided technical comments and guidance.

The thesis has not been submitted for other qualifications. All the work towards the thesis was carried out after the enrolment in the degree. No third party editorial assistance was provided in the preparation of the thesis. We acknowledge the support from the Australian Research Council (ARC) Linkage Project grant (LP120100529), the ARC Linkage Infrastructure, Equipment and Facilities scheme (LIEF) grant (LF120100129).

The author's total publications during the Ph.D. candidature are listed in two sections. The first section lists the publications directly related to this thesis i.e., these publications contribute towards this thesis.The second section lists the other publications produced during the Ph.D. candidature. However, they are not included in this thesis contributions in order to maintain focus on big data. Overall, I have a total of 10 publications which include 7 (1 under review and 1 to be submitted soon) journal papers and 3 conference papers. I got one *best paper award* at the *IEEE* 4*th World Forum on Internet of Things* (WF-IoT).

Publications directly related to thesis

Chapters 3 to 7 are the contributory chapters of the thesis and are based on the following publications. All the papers produced from this thesis are my original work. In all the publications, my contributions are: literature survey, theoretical analysis, problem formulation, algorithm development, programming in MATLAB and Python, data preparation and pre-processing, per-

forming experiments, results analysis, and manuscript writing. The contributions of co-authors Dr. Dheeraj Kumar, Dr. Sutharshan Rajasegarar and Prof. James C. Bezdek are: Providing technical comments and guidance, discussion on result analysis, manuscript proofreading, and providing critical feedbacks on manuscript writing. Other co-authors, Dr. Sarah M. Erfani, Zahra Ghafoori, Prof. Christopher Leckie, and my supervisor Prof. Marimuthu Palaniswami provided helpful guidance, suggestions, and proofreading to improve manuscript quality.

Journal Papers

1. **Rathore P.**, Bezdek J. C., Erfani S. M., Rajasegarar S., Palaniswami M. "Ensemble Fuzzy Clustering using Cumulative Aggregation on Random Projections" in *IEEE Transactions on Fuzzy Systems (IEEE T-FS)*, 26(3): 1510-1524, 2018

   This article develops a simple and computationally efficient framework, *cumulative agreement based fuzzy c-mean* (CAFCM), for high-dimensional data clustering. The CAFCM framework employs fuzzy *c*-means clustering on an ensemble of random projections to obtain multiple fuzzy clustering partitions, and then cumulatively aggregates them based on their quality to get a consensus bsest final partition. Experimental results on various big, high-dimensional datasets demonstrate that CAFCM outperforms three state-of the-art methods in terms of accuracy and space-time complexity. CAFCM runs one to two orders of magnitude faster than other state-of-the-arts algorithms. **Chapter 3 is based on this paper.**

2. **Rathore P.**, Kumar D., Bezdek J. C., Rajasegarar S., Palaniswami M. "A Rapid Hybrid Clustering Algorithm for Large Volumes of High-dimensional Data" in *IEEE Transactions on Knowledge and Data Engineering (IEEE T-KDE)*, 2018

   Existing clustering algorithms encounter serious problems related to computational complexities and/or cluster quality for datasets that are jointly large in the number of samples and the number of dimensions. This article presents a new relative of the visual assessment of tendency (VAT), which is popular method for cluster tendency assessment and subsequent clustering. To simultaneously overcome both the 'curse of dimensionality' problem due to high dimensions and scalability problems due to large sample size, this article develops a novel, hybrid ensemble-based clustering framework, FensiVAT, by leveraging a fast data-space reduction and an intelligent sampling strategy. FensiVAT also provides visual

evidence that is used to estimate the number of clusters (cluster tendency assessment) in the data. FensiVAT was compared with nine state-of-the-art approaches which are popular for large sample size and/or high-dimensional data clustering. Experimental results suggest that FensiVAT, which can cluster large volumes of high-dimensional datasets in a few seconds, is the fastest and most accurate method of the ones tested. **Chapter 4 is linked with this paper.**

3. **Rathore P.**, Ghafoori Z., Bezdek J. C., Palaniswami M., Leckie C."Approximating Dunn's Cluster Validity Indices for Partitions of Big Data" in *IEEE Transactions on Cybernetics (IEEE T-CYB)*, 2017 (DOI: 10.1109/TCYB.2018.2806886)

   Dunn's internal cluster validity index assesses partition quality, however, it is infeasible for big data due to $O(N^2)$ complexity. This article presents six novel methods, including two incremental methods, for approximating Dunn's index (DI) for big data. Four methods are based on Maximin sampling, which identifies a skeleton of the full partition that contains some boundary points in each cluster. Two additional methods are presented that estimate boundary points associated with unsupervised training of one class support vector machines. Numerical examples compare approximations to DI based on all six methods. Experiments on several big datasets show that a MMRS based incremental method offered an average speedup of about $1000:1$, and produced average values that matched DI values up to ± 0.01 when computed on the full dataset. **Chapter 5 is based on this paper.**

4. **Rathore P.**, Kumar D., Bezdek J. C., Rajasegarar S., Palaniswami M. "A Scalable Framework of Trajectory Prediction for Connected Vehicles". *IEEE Transactions on Intelligent Transport System (IEEE T-ITS)* (under review)

   This article develops a novel scalable framework, based on a new big data clustering method and Markov models, for both short-term and long-term trajectory prediction (TP) which can handle a large number of overlapping trajectories in a dense road network. The proposed framework can also determine the number of clusters, which represent different movement behaviours in trajectory data. We compare the proposed framework with a mixed Markov model (MMM)-based scheme and NETSCAN-based TP method on two real-life, large-scale taxi trajectory datasets from the Beijing and Singapore road networks. Experimental results

show that our proposed approach outperforms the existing approaches in terms of both short- and long-term prediction performances, based on prediction accuracy and distance error . **Chapter 7 is linked with this paper.**

5. **Rathore P.**, Kumar D., Bezdek J. C., Rajasegarar S., Palaniswami M. "Detecting Evolving Cluster Structures and Anomalies in High-Velocity, Big Streaming Data", to be submitted in *ACM Transactions on Knowledge Discovery from Data (ACM TKDD)* (status: final draft in preparation)

   The widespread use of Internet of Things (IoT) technologies, smartphones, and social media services generate huge amounts of data streams at high velocity. Automatic interpretation of high-velocity streams is required for timely detection of interesting events that usually emerge in the form of clusters. This article proposes a new relative of the improved VAT (iVAT) model for high velocity streaming data. Existing incremental VAT algorithms, inc-VAT/inc-iVAT and dec-VAT/dec-iVAT, are not suitable for high-velocity streams. To address this problem, this article proposes an incremental method, inc-siVAT, which deals with the large streaming data in chunks. It first extracts a small size smart sample using the MMRS sampling scheme, then incrementally updates the smart sample points on the fly, using a new incremental MMRS algorithm, to reflect changes in data streams after each chunk, and finally, produces an incrementally built iVAT image of the updated smart sample, using inc-VAT/inc-iVAT and dec-VAT/dec-iVAT algorithms. The sequence of these images can be used to detect evolving cluster structure and anomalies in streaming data. Our evaluation on dynamic streams of several big datasерts demonstrates the algorithm's ability to successfully identify anomalies and visualize changing cluster structure in high-velocity big, streaming data. **Chapter 6 is based on this paper.**

Conference Papers

1. **Rathore P.**, Kumar D., Bezdek J. C., Rajasegarar S., Palaniswami M. "Approximate Cluster Heat maps for Big Data". *24th IEEE International Conference on Pattern Recognition (ICPR)* at Beijing, China, 2018.

   A scalable version of iVAT called siVAT approximates iVAT images, but siVAT can be computationally expensive for big datasets. This article develops a new intelligent sampling

scheme, MMRS+, that in turn, introduces a modified version of siVAT, siVAT+, which approximates cluster heat maps for large volumes of high dimensional data much more rapidly than siVAT. We show that the samples obtained using MMRS+ retain almost the same geometry as the MMRS does. Experimental results suggest that images obtained using siVAT+ provide visual evidence about potential cluster structure in all datasets, including two unlabeled datasets, in significantly less time ($8-55$ times faster) than siVAT with no loss of accuracy or visual acuity.

2. **Rathore P.**, Ghafoori Z., Bezdek J. C., Palaniswami M., Leckie C. "Estimating Generalized Dunn's Cluster Validity Indices for Big Data", *IEEE Conference on System, Man, and Cybernetics (IEEE-SMC)* at Miyazaki, Japan, 2018. (**Best student paper award - Finalist**)

   Original Dunn's index (DI) is sensitive to anomalies due to the way distances are used in its computation. Generalized Dunn's indices (GDIs) overcome this drawback using various different distance measures. However, similar to DI, GDIs also have quadratic time complexity making them infeasible for big data. This article extended our previous work on Dunn's index for approximating GDIs. This article also illustrates that how our incremental approach from previous work approximates DI values with an optimal number of points, and shows that DI value monotonically decreases with the addition of new data point. The proposed algorithms are compared with a support vector machine based boundary extraction method and a random-sampling based estimation method. Experiments on several big datasets show that computing approximations to (three) GDIs with the MM skeleton is both computationally feasible and reliably accurate.

Other Publications

Following publications were also produced during the Ph.D. candidature. However, these publications do not contribute towards this thesis.

Journal Papers

1. **Rathore P.**, Rao A., Rajasegarar S., Vanz E., Gubbi J., Palaniswami M. "Real-time Urban Micro-climate analysis using Internet of Things", in *IEEE Internet of Things Journal (IEEE*

IoT), 5(2): 500-511, 2018

2. **Rathore P.**, Kumar D., Rajasegarar S., Palaniswami M. “Maximum Entropy based Auto Drift Correction using High and Low Precision Sensors", *ACM Transactions of Sensors and Networks (ACM TOSN)*, 13(3), 2017

Conference Papers

1. **Rathore P.**, Kumar D., Rajasegarar S., Palaniswami M. “Bayesian Maximum Entropy and Interacting Multiple Model based Auto Drift Correction in an IoT environment" in *IEEE* 4*th World Forum on Internet of Things (WF-IoT)* at Singapore 2018. (**Best Paper Award**)

Acknowledgements

As I am about to complete this thesis, I am already getting mixed feelings. Although I am happy about finishing my Ph.D. and entering a new stage of my life, I have already started missing the people, the campus, this lovely city Melbourne and everything here. There were hurdles and difficulties during these four years of my Ph.D., but I have been lucky that helping hands were always around. Each day of these four years has been a great opportunity for learning, thanks to my advisors, colleagues, friends, and family.

Firstly, I would like to express my sincere gratitude to my advisor Prof. Marimuthu Palaniswami for his patience, guidance, motivation, support, and immense knowledge. He is not only an excellent mentor, teacher, and researcher, but most importantly, he is a kind person in all spheres of life. I enjoyed freedom both in thoughts and research direction while working under his aegis. It was always enjoyable to chat with him in the kitchen or corridor. Indeed, it has been a rewarding experience to work with him, which I would cherish forever.

I also want to thank Prof. Jim Bezdek, who gave me many helpful suggestions for work and life. I am not only amazed by his strong academic expertise and life experience, but also his passion for research and his professional research attitude. I would also like to thank Dr. Sutharshan Rajasegarar, Dr. Dheeraj Kumar, and Dr. Aravinda Rao, not only for their insightful comments and helpful guidance but also for the hard question which incented me to widen my research from various perspectives. I am privileged to have worked with them. Dr. Jayavardhana Gubbi was very helpful when I was working with him during the first year of my Ph.D.

I would like to thank all my ISSNIP labmates and lovely friends: Nandakishor, Shitanshu, Bigi, Motin, William, Radha, Emerson, Bapin, Sharmistha, Ronit, and Anthony. You all made my Ph.D. journey memorable, colorful and delightful. There are many others that I am grateful to but cannot thank in this limited space. Last but not the least, I would like to thank my family. I am

deeply indebted to my wife Nitisha for her love, motivation, patience, and support that have been an enormous factor in successful completion of my Ph.D. My parents, my brother, my sister, and my parents-in-law have been very supportive throughout the past four years. I thank them for their constant encouragement and unconditional support. I would like to specially mention my father who always encouraged me to pursue higher studies. He has been my role model and inspiration. I can always count on him to inspire me and uplift my spirits.

Contents

This page intentionally left blank.

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

The ubiquity of the Internet and personal computing technologies, especially mobile computing and social media, has resulted in *digital data explosion*. Everyday an abundant amount of data is generated in the form of text, image, audio, video, time-series, and GPS logs, from various sources such as *Internet of things* (IoT) devices, smartphones, social networks activities, emails, and video-hosting services. Facebook alone logs over 25 terabytes (TB) of data per day [1]. It is estimated that more than 200 million emails are exchanged every minute [1], and 300 hours of videos are uploaded to Youtube [2] every minute [2]. Such data are termed as 'big data' [3].

Big data analytics can extract meaningful information from the oceans of the data produced by various sources [4]. Virtually every large business is interested in collecting large amounts of data from its customers or underlying infrastructure, and mining it to generate useful information in timely manner. This information helps businesses to provide better customer services and increase their profitability. The New York Exchange gathers about 1 TB of trade information during each trading session [2]. The real-time processing of this data can assist traders in making important trade decisions. About 23% of available digital data are believed to contain meaningful information that can be utilized by companies, government institutions, policy makers, and individual users [3].

Big data with IoT technologies have also played an essential role in the feasibility of smart city initiatives [5, 6]. Big data collected from smart city infrastructure and citizens through IoT

[1] http://mashable.com/2014/04/23/data-online-every-minute
[2] https://www.youtube.com/yt/press/statistics.html
[3] http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation

technologies and social media services offer the potential for the city to obtain valuable insights, that assist the city councils and administrators to plan and manage the city in a better way. Big data analytics has also lead to the development of new applications and services like Microsoft's HealthVault [4] - a platform to gather, store, utilize and share the health information online for health management.

Effective analysis of big data is a key factor for the success in many applications, businesses, and services, including several smart city application domains such as health care, transportation, finance, and energy sectors. The big data can be effectively utilized using suitable learning approaches, which can be supervised, semi-supervised, or unsupervised. Supervised approaches, such as classification techniques, require *labeled* training data for learning. Supervised learning techniques [7] finds its utility in many applications such as intrusion detection, spam detection, machine translation, sentiment analysis, and object recognition. Unfortunately, a significant amount of big data available to us are unlabeled. Only about 3% of the potentially useful data on the web is labeled. Moreover, obtaining the labels for massive amounts of data is extremely expensive and time-consuming, making supervised learning difficult for most big data applications. Semi-supervised approaches [8] alleviate the labeling problem by using a large pool of *unlabeled* data with a small set of labeled data. However, it is expensive to obtain supervision in many applications, such as medical and stock market analysis, where high-level of expertise is required for labeling. Unlike supervised and semi-supervised approaches, unsupervised methods do not require labeled data, thereby, avoiding the labeling cost and allowing one to leverage big data with minimum prior knowledge. Therefore, many applications demand sophisticated unsupervised tools to analyze big data for their better understanding.

Cluster analysis or clustering is the most common unsupervised approach to discover the underlying groups and patterns in the data. In clustering, data are partitioned into several subsets of similar objects without any prior knowledge. Clustering provides a summarized view of data in the form of patterns, and with the domain-specific information, these patterns may provide a better understanding of big data. Cluster analysis is an essential tool for knowledge discovery [9], outlier/anomaly detection [10–12], indexing [13], and compression [14]. It has also been used in many applications such as web search [15], social network analysis [16], information re-

[4] https://www.healthvault.com/us/en/overview

trieval [17], bioinformatics [18], gene expression analysis [19], image processing [20], market analysis [21, 22] and recommendation systems [23]. Clustering also finds it utility in several smart city applications to gain valuable insights from raw data obtained through various sensing devices. For example, the work in [24, 25] employ k-means clustering on weather station data to analyze *urban heat island* (UHI) effect and understand the characteristics of different surrounding environments, which assist city councils in urban planning. For smart parking applications, the studies in [26–28] apply clustering to urban car parking data to automatically obtain useful trends for better utilization of available parking facilities. Clustering methods have also been employed in *intelligent transportation systems* (ITS) to extract urban mobility patterns [29, 30] from pedestrians and vehicles movement data for better traffic management. For smart grid applications [31–33], clustering has been applied to time-series energy data from smart meters to identify energy usage profiles of residential, commercial, and industrial consumers. There are also several applications of clustering in other smart city contexts, such as in smart health care [34–36], smart agriculture [37, 38], and smart waste management [39, 40].

While cluster analysis is useful for many applications and services, it is a challenging task to analyze big data that are mostly noisy, streaming, high-dimensional, and heterogeneous. Therefore, sophisticated and efficient algorithms are required to manage and extract actionable information from big data. There are several fundamental questions to be answered when performing cluster analysis. We explain each of these questions of cluster analysis, followed by their challenges and limitations of existing algorithms for big data, in the next section.

1.2 Problem Statement and Challenges

Consider a dataset X consisting of N objects is partitioned into $k \in \{1, 2, .., n\}$ subsets (clusters), and each object in X is defined by a p-dimensional feature vector. Cluster analysis in X consists of three problems [41]: **(P1) pre-clustering *assessment of tendency*, which asks the question "Do the data have clusters? If yes, how many (k)?"; (P2) *partitioning the data* – finding the k clusters; and (P3) post-clustering *cluster validity*, which asks "Are the k clusters found useful?"**. Below, we explain each of the three problems of cluster analysis.

1. **Clustering tendency assessment:** A natural question that comes before applying any clus-

tering method on the dataset is- "Does the data contain any inherent grouping structure or cluster?" A major problem in unsupervised machine learning, specifically in cluster analysis, is that all clustering methods will partition the data into groups even if no "actual" cluster exists in the data. Therefore, prior to clustering or as a pre-clustering step, it is important to determine whether the data contains meaningful clusters (i.e., non-random structure) or not. If yes, then how many clusters, k? This problem is defined as an *assessment of clustering tendency or clusterability.*

There are mainly two types of approaches to evaluate the clustering tendency : (i) statistical methods, that measure the probability that a given dataset is generated by a uniform data distribution to test the spatial randomness of the data. A popular statistical approach is Hopkins statistic [42, 43]; and (ii) visual methods, which visually inspect the clustering tendency of the dataset. These methods reorder the dissimilarity matrix of the input data, and visually estimate the number of clusters that appear as the dark blocks along the diagonal of *reordered dissimilarity image* (RDI). Some popular methods in this category are *visual assessment of cluster tendency* (VAT) [44], *improved VAT* (iVAT) [45, 46] and their scalable versions scalable VAT (sVAT) and scalable iVAT (siVAT) [47].

2. **Clustering:** Once the number of clusters (k) in the data is known, the next task is to partition the data into k clusters using a suitable clustering algorithm. Numerous clustering methods have been proposed in the literature, which can be broadly classified into three categories: partitional, hierarchical, distribution, and density-based methods. The partitional algorithms such as k-means, fuzzy c-means (FCM) attempt to determine partitions that optimize a given objective function. In hierarchical clustering algorithms such as single-linkage (SL), complete-link (CL), and minimum variance, data are organized into hierarchical clusters based on the proximity between pairs of objects. Distribution-based approaches such as *Expectation Maximization* (EM) with *Gaussian Mixture models* (GMMs) partition the data based on the distribution of the data points. Density-based clustering such as DBSCAN, OPTICS detects regions (neighborhoods) of high density that are separated from one another by regions of low density.

3. **Cluster validity:** Clustering algorithm with different cost functions provide different clus-

tering results, and there is no single best choice of clustering algorithm and cost function for all possible datasets. The last important problem in cluster analysis is the evaluation of clustering results to find the partitioning that best represents the structure of the dataset. It also asks the question whether the k clusters found useful or not? Is there any better partition that clustering algorithm did not find? This process of evaluating the clustering results in a post-clustering step is commonly known as cluster validity. One approach of finding the best partition is through the use of scaler measures of partition quality. These measures are known as *cluster validity indices* (CVIs). The most important distinction for such measures is whether the index is internal or external. Internal CVIs use only data and/or algorithmic outputs, whereas, external CVIs require additional "outside" information such as a ground truth partition that labels subsets in the data. Some of internal CVI examples are *Dunn's index* (DI) [48], *David-Bouldin index* (DBI) [49], *Xie-Beni* (XB) index [50] and Silhouette coefficient [51] which use the output partition from clustering and the input dataset itself. Examples of external CVIs include *Rand index*(RI) [52], *Adjusted Rand index* (ARI) [53], and Purity [54].

Challenges

Despite its usefulness for many applications, big data cluster analysis is a challenging task due to the following properties of big data:

1. **Volume:** The two most important ways a dataset can be big are: (1) it has a very large number (N) of instances, and (2) each instance has many (p) features, i.e. it is high-dimensional data. In this era of big data, we witness tremendous growth of data, not only in the number of observations but also in the number of features, collected for each data object. In many applications such as biomedical imaging, sequencing, and time series matching, the dataset may consist of millions of instances in hundreds to thousands of dimensions [55]. The volume property of big data refers to the ability of a clustering algorithm to deal with large volumes of high-dimensional data.

   To deal with voluminous data, clustering algorithms should be scalable and efficient. This means that their complexity should be nearly linear or sub-linear with respect to the sample

size. Although the time complexity of clustering algorithms is related to the number of instances in the dataset, the dimensionality of the dataset is another critical aspect. Real-world datasets in the higher dimensional feature space are usually highly sparse, which makes it difficult to find statistically meaningful structures from such redundant and sparse data [56] through traditional clustering algorithms. Also, noisy and irrelevant attributes in the data can worsen the performance of a clustering algorithm.

A variety of clustering algorithms [41, 56–60] have been developed (discussed in Chapter 2) for a dataset that has either (1) large N but small p, or (2) small N but large p, but most clustering algorithms are impractical for handling datasets that are large jointly in N and p [61]. Most existing clustering algorithms encounter serious problems related to computational and space complexities and/or cluster quality for big datasets.

Similar to clustering, the implementation of clustering tendency assessment algorithms and *cluster validity indices* (CVIs) is often very computationally expensive for big datasets [62]. The scalable versions of VAT/iVAT algorithm such as sVAT [47], siVAT, and clusiVAT [63] are adequate for large sample size datasets, however, they still suffer from substantial computation time when the data is large in the number of dimensions. For cluster validity, internal CVIs that require both the input data and the output partition such as DI [48], XB [50], and Silhouette [51] have quadratic or higher computational complexity, which restricts their use for small to medium size datasets. There has been a considerable amount of work done to address the problem of clustering for big data. However, there is very little work available in the literature to address cluster tendency assessment and cluster validity problem for big data.

2. **Velocity:** Several processes generate large amounts of data which grow at an unlimited rate. These data processes are referred to as data streams. New data points are added to an already voluminous dataset at a fast rate, so it has to be dealt within a reasonable time. The velocity property refers to the ability of a clustering algorithm to handle high-velocity data streams in a timely manner.

   Extracting knowledge as a set of patterns in a continuous stream of data is a challenging task, due to the constraints imposed by the nature of data streams. First, due to memory

constraints, it is not feasible to store big data streams for a longer period. Second, patterns continuously appear and/or disappear in streaming data. Therefore, data must be processed faster before a new data stream is generated otherwise trends may change. These constraints on memory and computational complexity make cluster analysis a challenging task for data streams.

A suitable clustering approach should be able to update the existing result by accommodating new information without running the experiment on entire data again. Several clustering algorithms [64–68] have been developed to handle data streams. These approaches are usually incremental, which continuously update existing result either with each new data point or with a small chunk (of fix size) of data points. The inability of these approaches to determine the number of evolving clusters (patterns might change with new data points) dynamically makes them ineffective for unlabeled data (for which k is mostly unknown).

The incremental and decremental VAT algorithms for streaming data, inc-VAT/dec-VAT and inc-iVAT/dec-iVAT [69], provide a point by point visualization of evolving cluster structures in streaming data using a sliding window based approach. However, hardware and software constraints limit them to a maximum window size of about $N \sim 5,000$ inputs, due to system limitations to storing and visualize reordered dissimilarity matrix. When this limit is reached, point by point deletion and insertion maintains this fixed window size. Thus, If $N = 100,000$, the user will have a cluster heat map (RDI) of only the last window at the end of the process. The salient point is that the history of cluster evolution is not available. At present, to our knowledge, there is no technique on offer for visualization of evolving cluster structures in high-velocity, data streams.

3. **Variety:** The variety property refers to an ability of a clustering algorithm to handle heterogeneous and unstructured data. As technology moves into more realms of human lives, big data is taking on a larger variety of forms. The massive spread out of smart devices, sensors, and social collaboration technologies has made it challenging to deal with big data, as data collected from various sources with different specializations not only includes the traditional data but also raw, unstructured, and semi-structured data in the form of audio, text, emails, videos etc [70].

Consider a transport application in smart city perspective for real-time traffic monitoring. Assume that various noise and pollution sensors are deployed across the road segment to estimate the crowd-density or road traffic. Besides, people available in that locality share their GPS information, geo-tagged images of traffic, text, audio, video using a participatory sensing platform. They are also able to send nominal information (yes/no for "Is your area crowded") by using a dedicated platform or an app. These data may also have mixed attributes such as numerical, categorical, nominal, and ordinal. The heterogeneity and noise make processing and clustering of big data a challenging task. Specialized techniques may be needed to handle different formats of the data. Most clustering algorithms handle heterogeneous data either by employing feature transformation to unify the format of data or using heterogeneous data directly with customized distance measures to compute pairwise similarities among data points.

Although there are two other characteristics (*veracity* and *value*) of big data, these (above) three core characteristics must be taken into account when developing a clustering algorithm for big data. Next, we discuss our research contributions to address each of these challenges for big data cluster analysis.

1.3 Research Contributions

Based on the above discussion on challenges, it is evident that sophisticated and efficient algorithms are required for cluster analysis of big data which (i) should have linear or sub-linear running time complexity, (ii) need a minimum amount of memory, and (iii) do not compromise with the output quality. The new algorithms should be adaptable for various applications which means they should (i) not require lots of tunable and sensitive input parameters, (ii) not require the number of clusters to be known in advance, and (iii) be able to handle high-velocity data for streaming data applications.

The objective of this thesis is **to design algorithms to solve each of the three problems of cluster analysis viz., cluster tendency assessment, clustering, cluster validity, for large volumes of high-dimensional data, including streaming data**. The main contributions of this thesis are outlined as follows:

1. **Clustering high-dimensional data**: The first contribution of this thesis addresses the high-dimensionality and scalability issues for soft clustering methods. Specifically, a novel, simple and computationally efficient framework, *cumulative agreement based fuzzy c-means* (CAFCM), has been developed for high-dimensional data clustering which employs fuzzy c-means clustering on an ensemble of random projections. The proposed ensemble approach combines multiple fuzzy (or soft) partitions sequentially based on their quality, as measured using cluster validity indices (CVIs).

   (a) The performance of CAFCM was compared with three other ensemble-based clustering methods, called EFCM [71], RPFCM-A [72], and RPFCM-B [73], on two synthetic and six real large, high-dimensional datasets. Experimental results show that CAFCM outperforms the other three approaches based on accuracy, stability (standard deviation), space, and time complexity.

   (b) CAFCM scales linearly in the number of data points and the number of repetitions, making CAFCM approach feasible for large and high-dimensional datasets.

   (c) CAFCM does not require any prior knowledge of the number of clusters that might be present in the dataset, which makes it attractive for real-world clustering applications.

2. **Cluster tendency assessment and subsequent clustering on big, high-dimensional data**: The second contribution of this thesis solves the cluster tendency assessment and clustering problem for large-scale, high-dimensional datasets. We proposed a fast, hybrid clustering algorithm called FensiVAT, which effectively integrates a *visual assessment of cluster tendency* (VAT) approach with a new random projection based ensemble technique and a smart sampling strategy, called *Maximin and Random sampling* (MMRS), to deal with large amounts of high-dimensional data.

   (a) FensiVAT provides reliable visual evidence about the number of clusters that may be present in big, high-dimensional data, in a few seconds.

   (b) Experiments were performed on two synthetic and seven real datasets including one unlabeled dataset, that are large in sample size (N) and dimensions (p). The performance of FensiVAT FensiVAT was compared with nine other methods, which include six big data clustering methods, viz., *single pass k-means* (spkm) [74, 75],

mini-batch k-means (MBKM) [76], CLARA [77], CURE [78], clusiVAT [63], GARDENkm [79], and FastSpec [80] and two high-dimensional data clustering approaches, PROCLUS [81], and *random projection based ensemble clustering* (RP-EN) [72, 82].

(c) Experimental results suggest that FensiVAT is up to several order of magnitudes faster than the other nine approaches (except MBKM), without compromising accuracy.

3. **Cluster validity for big data**: The third contribution deals with the cluster validity problem for big data. Six approximation methods are proposed to address the high computational complexity problem of Dunn's internal cluster validity indices for big data.

   (a) The four methods viz., αMMRS, αnMMRS, *i*MMRS, *in*MMRS are based on a variant of Maximin random sampling (MMRS) [83] which identifies a skeleton of the full partition that contains some boundary points (required to compute Dunn's indices) in each cluster. The *i*MMSR and *in*MMSR schemes are incremental methods, which produce a specified number of (boundary) points to compute the approximate Dunn's index.

   (b) The other two methods are based on the unsupervised training of *one class support vector machines* (OCSVM) [84, 85].

   (c) All six methods presented have linear complexity in the number of samples (N).

   (d) Experiments were performed on three synthetic and four real labeled datasets that are large in sample size (n) and dimension (p). Our experiments show that computing approximations to DI with Maximin skeleton based methods are both tractable and accurate.

4. **Cluster tendency assessment and anomaly detection in high-velocity, streaming data**: This contribution of the thesis focused on detecting evolving structure and anomalies in high-velocity, streaming data. We developed an incremental version of siVAT, inc-siVAT, for visualization of evolving cluster structures in high-velocity, big streaming data.

   (a) inc-siVAT deals with the large streaming data in chunks. First, it extracts a small size smart sample using the MMRS sampling scheme, then it incrementally updates the smart sample points on the fly, using a new incremental MMRS algorithm, to reflect

changes in data streams after each chunk, and finally, produces an incrementally built RDI image of the updated smart sample, using inc-VAT/inc-iVAT and dec-VAT/dec-iVAT algorithms. The image of the updated RDI provides the visualization of evolving cluster structure after each chunk of data streams.

(b) Experiments were performed on dynamic streams of a two-dimensional Gaussian mixture data and KDD Cup'99 data to show time effectiveness of inc-siVAT over existing incremental VAT/iVAT methods.

(c) We demonstrated the applicability of inc-siVAT for cluster assessment and subsequent anomaly detection in evolving data streams of three real datasets including a smart city IoT data, collected from the Heron Island weather station deployed on the Great Barrier Reef, Australia [86].

5. **Big data clustering for a real-world application**: The thesis also presents a utility of big data clustering for a real-life smart city application. We proposed a novel scalable framework for vehicle *trajectory prediction* (TP), based on a big data clustering algorithm and Markov chain models, which can utilize a huge number of trajectories in a dense road network, typical for major cities around the world.

   (a) A modified version of clusiVAT, Traj-clusiVAT, was developed to cluster a large number of trajectories accurately and efficiently, for better trajectory prediction performance.

   (b) The proposed TP framework was compared with two existing TP algorithms: a *mixed Markov model* (MMM)-based [87] and a trajectory clustering model NETSCAN [88]-based algorithm, for both short and long-term trajectory prediction.

   (c) Experiments used two real, large-scale taxi trajectory datasets: (i) T-Drive taxi trajectory dataset [89, 90] consisting of $43,405$ trajectories on a road network in the center of Beijing, and (ii) Singapore taxi dataset consisting of 370 million GPS traces and 3.28 million passenger trips from $15,061$ taxis during one month period in Singapore. This was the first time any TP approach used such a large number of real-life road network trajectories for trajectory prediction.

   (d) Experimental results show that the proposed TP framework outperforms the existing

two approaches for both short- and long-term prediction performances, based on prediction accuracy and distance error (in km).

1.4 Thesis Outline

In this thesis, we introduce a suite of novel algorithms to solve each of the three problems of cluster analysis for big data (including streaming data). These algorithms address most of the challenges of big data. This thesis is structured into eight chapters presenting five main contributions, as shown in Fig. 1.1. Chapter 2 provides the detailed review of traditional algorithms for each of the three problems of cluster analysis viz., cluster tendency assessment, clustering, and cluster validity, with a specific focus on existing visual assessment of tendency family algorithms, e.g., VAT, iVAT, sVAT, siVAT, and incremental VAT methods that provide foundation to this thesis contributions. Chapters 3-7 presents the main contributions of the thesis introducing a novel algorithm for each of the three problems of cluster analysis for big data in each chapter. Chapter 3 describes a novel clustering algorithm, CAFCM, for high-dimensional data clustering, which employs FCM clustering on an ensemble of random projections, and provides a final output partition using a new aggregation scheme. Chapter 4 address the cluster tendency assessment and clustering problem for large-volume, high-dimensional datasets. Particularly, it presents a fast cluster tendency assessment and subsequent clustering algorithm, based on an intelligent sampling scheme and a new random projection-based ensemble method, for large volumes of high-dimensional data. Chapter 5 address the problem of cluster validity for big data. Specifically, it introduces six approximation algorithms for Dunn's cluster validity indices for big data. Chapter 6 proposes a novel algorithm, inc-siVAT, for visualizing evolving cluster structures and detecting anomalies in high-velocity data streams. Chapter 7 presents a novel scalable framework for vehicle trajectory prediction as a real-world application of big data clustering. Finally, Chapter 8 concludes this thesis and discusses possible future work.

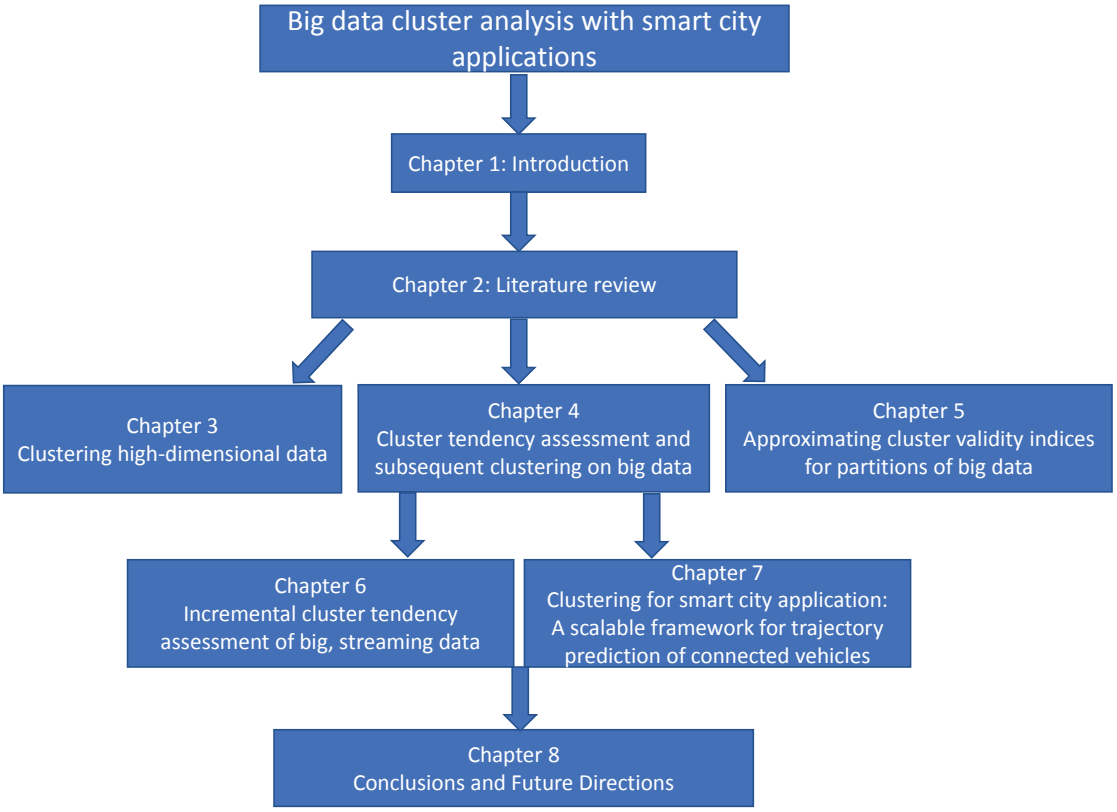


Figure 1.1: Thesis outline

This page intentionally left blank.

Chapter 2
Background and Literature Review

This chapter gives a detailed review of the major cluster analysis techniques proposed in the literature. In particular, clustering tendency assessment techniques are discussed in Section 2.2. Then, traditional clustering algorithms and big data clustering techniques are reviewed in Section 2.3. Cluster validity indices are discussed in Section 2.4. Significant work in the area of a big data clustering application for trajectory prediction are discussed in Section 2.5. Fig. 2.1 outlines popular cluster analysis techniques, and Table 2.1 provides a summary of the existing clustering algorithms for big data.

2.1 Cluster analysis

Cluster analysis is an important unsupervised technique in exploratory data analysis. It aims to divide data objects into groups (clusters), so that data objects within the same group are more similar than those in different groups. It is often used at the initial stage of data analysis, when there is little knowledge available about the data. Next, we introduce some basic notations.

Consider a set of N objects $O = \{o_1, o_2, ..., o_N\}$, partitioned into $k \in \{2, .., N-1\}$ subsets, where each object o_i is defined by a p-dimensional feature vector, $\mathbf{x}_i \in \mathbb{R}^p$ in a set of $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. Alternatively, data may be presented in the form of $N \times N$ dissimilarity matrix $D_N = [d_{ij}]$, where d_{ij} represents dissimilarity between o_i and o_j. We denote the set of all non-degenerate (no zero rows corresponding to empty clusters) soft (fuzzy/probabilistic) k-partitions

of N objects as:

$$M_{fkN} = \{U \in \mathbb{R}^{k \times N} : u_{ij} \in [0,1] \forall \ 1 \leq i \leq k, \ 1 \leq j \leq N; \sum_{i=1}^{k} u_{ij} = 1 \ \forall j; 0 < \sum_{j=1}^{N} u_{ij} < N \ \forall i\}, \tag{2.1}$$

where U is the membership matrix (partition), and its entry u_{ij} denotes the membership of point j in the cluster i, for fuzzy clustering. If the clustering is probabilistic, the value $u_{ij} = p_{ij}$ of data point j is the posterior probability that, given point j, it came from class i. The crisp partition can be viewed as a special case of soft partition, where membership u_{ij} is 1 if point j belongs to cluster i, else u_{ij} is 0. The crisp k-partition can be denoted as:

$$M_{hkN} = \{U \in M_{fkN} | u_{ij} \in \{0,1\} \forall i, j\}, \tag{2.2}$$

Alternatively, a crisp partition U of X is a set of disjoint clusters that partitions X into k groups: $C = \{C_1, C_2, ..., C_k\}$; $U \leftrightarrow X = \bigcup_{i=1}^{k} C_i$; and $C_i \cap C_j = \emptyset \ \forall \ i \neq j$. The centroid of cluster C_i is its mean vector $\mathbf{v}_i \leftrightarrow \overline{C}_i = \frac{1}{|C_i|} \sum_{\mathbf{x}_i \in C_i} \mathbf{x}_i$, where $|C_i|$ represents the the number of data points in cluster C_i.

Cluster analysis consists of three problems viz., (P1) cluster tendency assessment; (P2) clustering; and (P3) cluster validity. The problem of estimating the number of clusters k prior to actual clustering is known as cluster tendency assessment (P1). Once the k is known, the next problem (P2) is to partition the data into k subsets of similar objects. The last problem (P3) comprises computational models and algorithms that identify a "best" member amongst a set of *candidate partitions* $CP = \{U \in M_{fkN}$ or $M_{hkN}\}$ of the objects in O, obtained using either different clustering algorithms or using different configurations of the same clustering algorithm. Below, we discuss important techniques and algorithms available in the literature to address each of the three problems of cluster analysis. The classification of these techniques in the form of the graph along with example algorithms for each class is shown in Fig. 2.1.

2.2 Cluster tendency assessment

Clustering is used in many different scientific domains and applications as a practical tool to identify structure in complex data. There is renewed interest in clustering because of new areas

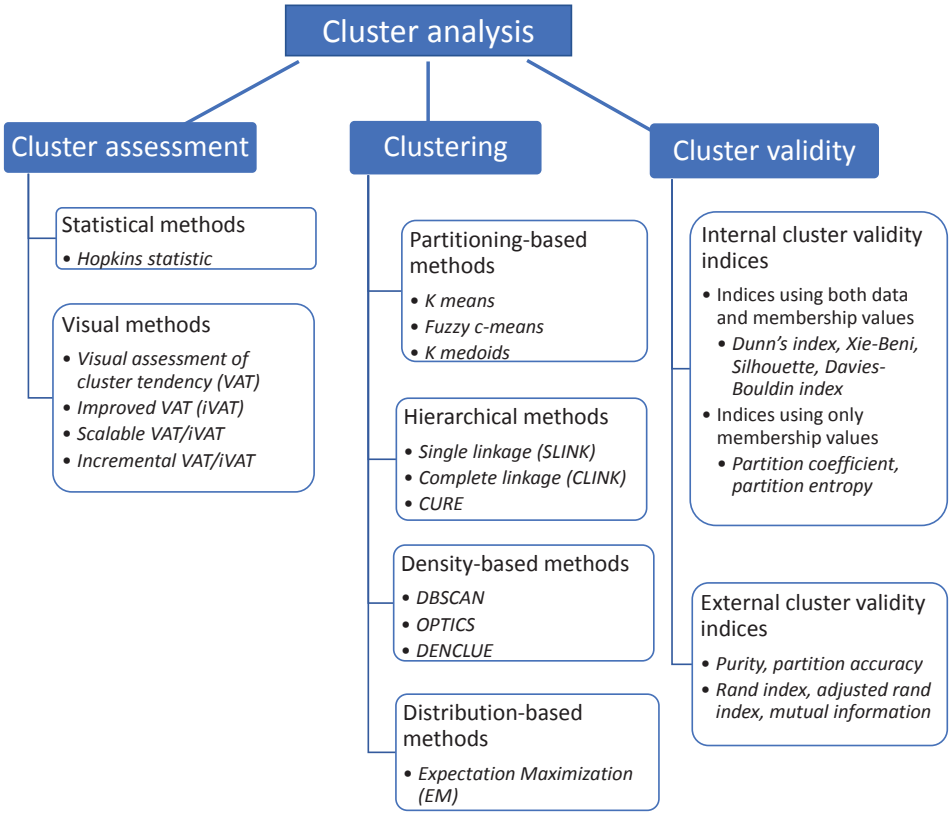


Figure 2.1: Cluster analysis techniques

of application, such as image and speech processing, bioinformatics, social network and IoT data analysis. Most clustering algorithms require the number of clusters, k, as an input, which is usually unknown for real-life data [91]. The estimation of k for real datasets has been identified as "one of the most difficult problems in cluster analysis" by Bock [92].

Traditional approach to this problem is based on either (i) a framework in which clusters of a particular shape are assumed as a model or (ii) on a two-step procedure using clustering and separate criterion in which clustering criterion determines an optimal partition for a given k and a separate criterion measures the goodness of the classification to determine k [93]. These two steps can be combined in a single principle in the former approach, e.g., this is achieved in the probabilistic mixture model assuming that data can be described by a mixture of multivariate distributions with some parameters that determine their shape. However, now the problem of finding the number of clusters turns to statistical model selection problem.

Density-based clustering approaches such as DBSCAN and OPTICS do not require the specification of k, however, they require the choice of other two parameters which are sensitive to the clustering algorithm. Hierarchical clustering avoids this problem by providing a hierarchical structure of all the data points. However, it also requires the choice of similarity threshold that results in a different number of clusters. There is no general technique to estimate k for centroid and distribution based clustering algorithms. The most practiced technique is to run clustering for different values of k, then find the best partition and the corresponding number of clusters in it, using a cluster validity index. Other methods for the estimation of k and model selections include elbow technique [83] and information theoretic criterion like *Bayesian information criteria* (BIC) [94] and *Akaike information criterion* (AIC) [95]. Clustering tendency approaches can be divided into the two groups (i) statistical methods, and (ii) visual methods.

2.2.1 Statistical methods

Statistical methods assume that data is generated by a particular distribution on the hypothesis that random data should not have clusters. They compare the input data against random data to measure to what degree clusters exist in the data to be clustered. The Hopkins statistic [42, 43] is one of the most popular statistical methods for cluster tendency assessment. It measures the probability that a given dataset is generated by a uniform data distribution. There are several formulations of Hopkins statistic. A typical formulation is as follows:

- Let X be the set of N data points in p-dimensional space
- Consider a random sample (without replacement) of $n << N$ data points with members $\mathbf{x}_i$.
- Generate a set Y of n uniformly randomly distributed data points.
- Define two distance measures, u_i to be the distance of $\mathbf{y}_i \in Y$ from its nearest neighbor in X and w_i to be the distance of $\mathbf{x}_i \in X$ from its nearest neighbor in X
- Compute Hopkins statistic using the following formula

$$H = \frac{\sum_{i=1}^{n} u_i^p}{\sum_{i=1}^{n} u_i^p + \sum_{i=1}^{n} w_i^p} \tag{2.3}$$

A value close to 0 indicates uniformly distributed data, a value around 0.5 indicates the data is random, and a value close to 1 indicates that the data is highly clustered. However, data containing just a single Gaussian will also score close to 1, as Hopkins statistic measures deviation from a uniform distribution, not multimodality. This problem makes it largely useless in application.

2.2.2 Visual methods

Visual methods for various data analysis problems have been extensively studied in [96]. Particularly, the representation of data structures in an image format has a long and continuous history [44, 97–99]. The earliest published work that discusses visual display of clusters is the Shade approach in [98]. SHADE approximates a digital image representation of clusters using a crude 15 level halftone scheme created by overstriking standard printed characters. It displays the lower triangulation part of a complete square display. SHADE is used *after* application of a hierarchical clustering scheme, as an alternative to visual displays of hierarchically nested clusters via the standard dendrogram. Visual identification of (triangular) patterns in SHADE is more difficult than when a full, square display is used.

Visual representation of structure in unlabeled dissimilarity data using *reordered dissimilarity image* (RDI) started in 1909 [97]. The visual representation of pairwise dissimilarity between a set of N objects is depicted by a $N \times N$ image, where objects are reordered such that resulting image (RDI) is able to highlight the potential cluster structure in the data. The intensity of each pixel in an RDI reflects the dissimilarity between the corresponding row and column objects. In a grayscale image of RDI, white pixels represent high dissimilarity, while black represents low dissimilarity. A "useful" RDI highlights potential clusters as a set of "dark blocks" along the diagonal of the image. Several schemes [44, 97–99] have been presented to generate RDI. Among them, *visual assessment of cluster tendency* (VAT) and its relatives are most popular. Below, we discuss some of them.

2.2.2.1 Visual assessment of cluster tendency (VAT)

The VAT [44] algorithm is based on (but not identical to) Prim's algorithm [100] for finding the *minimum spanning tree* (MST) of a weighted undirected graph. It is a single-linkage (SL)

based approach which proceeds by connecting the next nearest vertex to the current edge until the complete MST is formed. It reorders the dissimilarity matrix D_N to D_N^* using edge insertion ordering of the vertices added to the MST and specifies either end of the longest edge as the initial vertex for MST formation. When the dark blocks appear along the diagonal of the image $I(D_N^*)$ of the reordered distance matrix D_N^*, they potentially represent different (ideally, k) clusters. Since single-linkage clusters are always diagonally aligned in the VAT ordered images, so, having the estimate of k from $I(D_N^*)$, k-aligned clusters can be obtained by cutting the largest $k-1$ edges (given by the MST cut magnitude order) in the MST. SL performs best if data has long, chain-like clouds, well-separated clusters. As overlap among clusters increases, SL becomes unreliable. Nonetheless, SL has been successfully used in many data clustering applications. Pseudocode for VAT is given in Algorithm 1.

Algorithm 1 VAT

Input: D_N- $N \times N$ dissimilarity matrix
Output: D_N^* - $N \times N$ VAT reordered dissimilarity matrix of D_N
P- VAT reordering indices of D_N
h- Ordering of MST cut magnitudes
F- MST connection indices
Initialize the MST with the first element
Set $\mathbf{K} = \{1, 2, .., N\}$;
$\mathbf{I} = \mathbf{J} = \emptyset$;
Select $(i, j) \in \arg\max_{a \in \mathbf{K}, b \in \mathbf{K}} D_{Nab}$
Set $P(1) = i$;
$\mathbf{I} = \{i\}$,
$\mathbf{J} = \mathbf{K} - \{i\}$
$F_1 = 1$
Keep on adding the nearest of the remaining points to the current MST
for $r = 2$ to N **do**
Select $(i, j) \in \arg\min_{a \in \mathbf{I}, b \in \mathbf{J}} D_{Nab}$
$h_{r-1} = D_{Na_i b_j}$
$P_r = j$
$\mathbf{I} \leftarrow \mathbf{I} \cup j$
$\mathbf{J} \leftarrow \mathbf{J} - i$
$F_r = i$
end for
Rearrange the distance matrix D_N as per the VAT reordering indices P to obtain D_N^*
$D_{Nab}^* = D_{NP_aP_b} \quad 1 \leq a, b \leq N$

2.2.2.2 improved VAT (iVAT)

Though VAT often provides a useful estimate of k in a dataset, a much sharper reordered diagonal matrix image can be obtained using improved VAT (iVAT) [45, 46]. iVAT provides better reordered diagonal matrix image by replacing input distance d_{ij} in distance matrix D_N by distances $D_N^{'} = [d_{ij}^{'}]$,

$$d_{ij}^{'} = \min_{r \in P_{ij}} \max_{1<h<|r|} D_{N_{r[h]r[h+1]}}, \tag{2.4}$$

where $r \in P_{ij}$ is an acyclic path in the set of all acyclic paths from object (o_i) and (o_j) (vertices i and j) in O.

The recursive version of iVAT [45] has a time complexity of $O(N^2)$ as compared to $O(N^3)$ for iterative version of iVAT [46]. Importantly, the theory that connects SL to VAT also holds for recursive iVAT, which preserves VAT order. Pseudocode for recursive iVAT is given in Algorithm 2.

Algorithm 2 iVAT

Input: D_N^* - $N \times N$ VAT reordered dissimilarity matrix
Output: $D_N^{'*}$ - $N \times N$ iVAT dissimilarity matrix

```
for r = 2 to N do
    j = arg min_{1≤a≤r-1} D*_{Nra}
    D'*_{Nrj} = D*_{Nrj}
    b = {1,2,...,r-1}/ j
    D'*_{Nrb} = max{D*_{Nrj}, D*_{Njb}}
end for
D'*_{Nrb} = D'*_{Nbr}
```

2.2.3 Cluster tendency assessment for big data

Existing cluster tendency assessment algorithms for big data are based on an intelligent sampling technique that combines Maximin and Random Sampling, called *Maximin Random Sampling* (MMRS) [47, 83]. *The Maximin sampling rule* was introduced in 1953 by Thorndike [83] in this way:

Our procedure is to assume that the two jobs [objects underlying the data] which are at the greatest distance from one another will axiomatically fall in different families. The third cluster

starts with the job which is least near to the other two. Each cluster is built up by adding on that specimen which is nearest to the one which initially defined the cluster.

Thorndike illustrated his idea by initializing a sequential 3-means clustering algorithm with cluster centers obtained by this approach. Casey and Nagy [101] described the same procedure in exact detail this way:

[MM sampling] *The first sample in the batch to be processed is designated cluster center number one. The distances of the remaining samples from this one are calculated, and the farthest sample is called center number two. The smaller of the two distances from each sample to these two centers are listed, and the sample having the greatest minimum distance is selected. The remaining centers are chosen in turn to have maximum separation from the existing centers. These initial cluster centers are well-scattered over the sample space, an intuitively desirable property.*

Kennard and Stone [102] used the MM sampling to select initial prototypes. Gonzalez [103] describes an algorithm that at first glance looks different than the MM sampling, but upon closer examination, his algorithm is identical to the MM sampling. A formal definition [104] of Maximin sampling is given as:

Maximin (MM) sampling is a distance-based sampling method, which selects a few samples far from each other so that they represent diverse regions of the input space. The rationale for Maximin sampling is to select data points from the input data such that the minimal pairwise distance between sampled points is maximized. This means that a Maximin sample of size $n << N$ contains sample whose pairwise distances are maximum compared to any other n-sized sample of the same data.

Several variations of *random sampling* (RS) are used to enrich the MM samples. MMRS sampling is the basis of the success of *scalable visual assessment of tendency* (sVAT) [47] and scalable improved VAT (siVAT) for building approximate cluster heat maps in big data.

MMRS sampling starts with the selection of $k^{'}$ *distinguished objects* (the selected MM samples) in X, which are furthest from each other. Then, each object in O is grouped with its nearest *distinguished object*. This stage divides the entire dataset X into k' groups, $\{S_t\}_{t=1}^{k'}$, by associating $|S_t|$ objects to the t-th *distinguished object*. This grouping task requires the computation of a $k' \times N$ distance matrix. Then, the Maximin sample $\tilde{S}$ of size n (just a small fraction of N), is built by selecting a specified number of random data points (Random sampling (RS)) from each of the

$k^{'}$ groups. The number of points, n_t extracted from subset S_t is proportional to the number of data points in S_t, namely, $n_t = \lceil n \times |S_t|/N \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function. The term MMRS is used for overall process. Psuedocode for MMRS sampling is given in Algorithm 3.

Algorithm 3 Maximin Random Sampling (MMRS)

Input: Dataset $X = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\} \subset \mathbb{R}^{N \times p}$ or a pairwise dissimilarity matrix D_N;
k': desired number of MM samples (*distinguished objects*) in X
n: an approximate sample size.
Output: M: A vector containing indices of k' MM points of D_N;
$Dmax$: A vector containing maximum distance of each MM point from previous MM points;
$R \subset \mathbb{R}^{N \times k'}$: A matrix containing distance of each MM point from objects in X;
nt_{all}: a set of number of local (neighbour) samples for each MM point;
$\tilde{S}$: indices of MMRS sample (of size n) of D_N.
Step 1: Select [the indices $M = \{m_1, ..., m_{k'}\}$ of] MM points
Start point, m_1 = any random point in X;
$\mathfrak{D} = \{dist\{\mathbf{x_{m_1}}, \mathbf{x_{m_1}}\}, .., dist\{\mathbf{x_{m_1}}, \mathbf{x_N}\}\} = \{\mathfrak{D}_1, ..., \mathfrak{D}_N\} \leftrightarrow \{r_{11}, r_{21} ..., r_{N1}\} = R_{\bullet 1}$
for t $\leftarrow$ 2 to k′ do
$\mathfrak{D} = \underbrace{(min\{\mathfrak{D}_1, r_{1m_{t-1}}\}, ..., min\{\mathfrak{D}_N, r_{Nm_{t-1}}\})}_{min(\mathfrak{D}, R_{\bullet(t-1)})}$
$m_t = \underset{1 \le j \le N}{\arg\max} \{\mathfrak{D}_j\}$
$dmax_t = \mathfrak{D}_{m_t}$
$R_{\bullet t} = \{dist\{\mathbf{x_{m_t}}, \mathbf{x_{m_1}}\}, .., dist\{\mathbf{x_{m_t}}, \mathbf{x_{m_t}}\}, .., dist\{\mathbf{x_{m_t}}, \mathbf{x_N}\}\}$
end for
$M = \bigcup_{t=1}^{k'} m_t; \quad Dmax = \{dmax_1, dmax_2, ..., dmax_{k'}\}$
Step 2: Group each object in X with its nearest MM point
$S_1 = S_2 = ... = S_{k'} = \emptyset$
for t $\leftarrow$ 1 to N do
$l = \underset{1 \le j \le k'}{\arg\min} \{dist\{\mathbf{x_{m_j}}, \mathbf{x_t}\}\}$
$S_l = S_l \cup \{t\}$
end for
Step 3: Randomly select data near each MM points to obtain the n number of samples
$n_t = \lceil n * |S_t|/N \rceil \quad t = 1, 2, ..., k'$
Draw n_t unique random indices $\tilde{S}_t$ from S_t
$\tilde{S} = \bigcup_{t=1}^{k'} \tilde{S}_t; \quad \mathscr{N} = \{n_1, n_2, ..., n_{k'}\}$

MMRS sampling picks distinguished objects from the dataset. Hence, it requires relatively very few samples compared to random sampling to yield a diverse subset of the big data, which represents the cluster structure in the original (big) dataset. Hathaway [47] provided two propositions about the MMRS procedure, which form the basis of cluster tendency assessment algorithms

for big datasets.

Proposition 2.1. *Let O be a finite set of distinct objects that can be partitioned into k compact-separated (CS) [48] clusters and let $k' \geq k$ (i.e. k' is an overestimate of the true number of clusters k), then*

***A**. Step 1 of the MMRS algorithm selects at least one distinguished object (MM sample) from each cluster.*

***B.** In addition, if $n_t = n \times |S_t|/N$ is an integer for $i = 1,2,...,k'$ (Step 3 of MMRS) then the proportion of the objects in the MMRS sample from cluster $O^{(j)}$ equals the proportion of objects from same cluster $O^{(j)}$ in the original data, for $j = 1,2,...,k$.*

Proof. See [47] for proof. ■

2.2.3.1 sVAT/siVAT

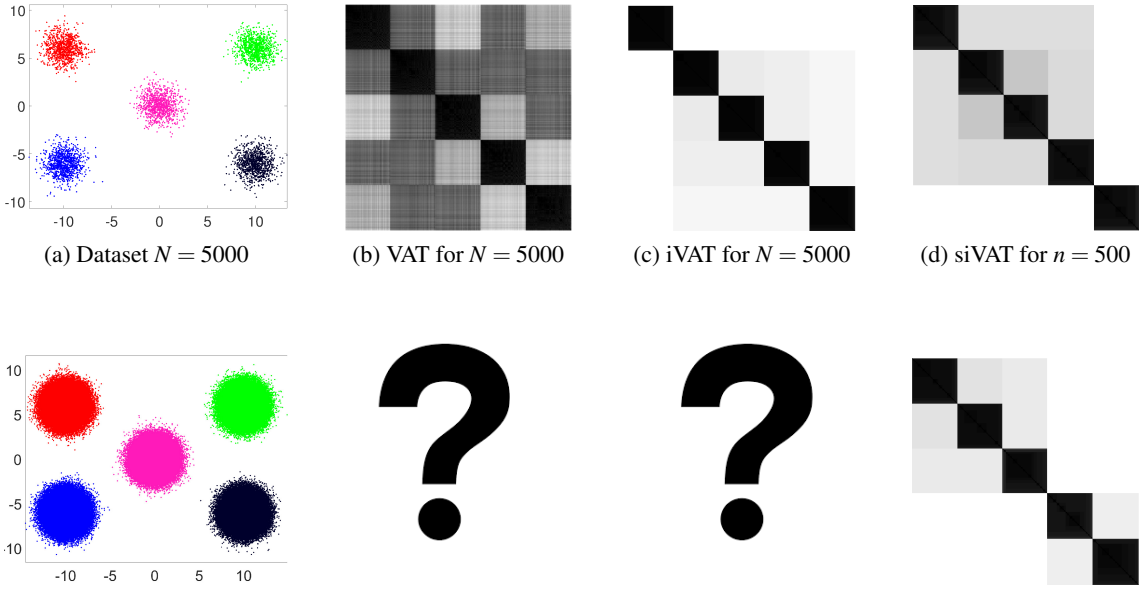
While VAT and iVAT algorithms work fine on small datasets, they suffer from resolution and memory constraints that limit their usefulness for input matrix sizes of order of 10^5 and so. To overcome these limitations, *scalable single linkage algorithms* sVAT/siVAT [45, 47] were proposed, which first find $n << N$ samples using MMRS sampling, and then apply VAT (or iVAT) to the small distance matrix D_n (computed from n samples) to obtain its reordered distance matrix D_n^* (or $D_n'^*$). The image $I(D_n'^*)$ usually provides a useful visual estimate of k without the need to compute the very large distance matrix, D_N of the big dataset, and circumvents the problem that $I(D_N^*)$ is not computable. sVAT is just like siVAT, except it uses only VAT after the sampling step. Pseudocodes for siVAT algorithm is given in Algorithm 4. The siVAT scheme does not involve any sensitive threshold parameter, and requires the user to supply only two parameters: n the desired sample size, and $k^{'}$, an overestimate of k, the assumed number of clusters, to obtain $k^{'}$ *distinguished objects* (or MM points) in the sample.

Fig. 2.2 illustrates VAT, iVAT, and siVAT for a 2D synthetic dataset. View (a) is the scatterplot of 5000 data points randomly drawn from five *Gaussian mixture* (GM) components with equal prior probabilities. Its VAT and iVAT images are shown in Views (b) and (c). While both VAT and iVAT images show five dark blocks along the diagonal corresponding to the five clusters in the dataset, dark blocks in the iVAT image are much clearer than the VAT image. View (d) shows the

Algorithm 4 siVAT

Input: Dataset X or $N \times N$ dissimilarity matrix, D_N
k': Overestimate of the true number of clusers, k, in X
n: an approximate sample size.
Output: D'^{*}_{n} - $n \times n$ iVAT dissimilarity matrix of D_n
$\tilde{S}$- indices of samples in D_n
P- VAT reordering indices of D_n
h- Ordering of MST cut magnitudes

Apply Maximin Random sampling on X (or D_N) returning a MMRS sample $\tilde{S}$ of size n (Algorithm 3)
Compute $D_n = dist\{\mathbf{x}_{\tilde{S}}, \mathbf{x}_{\tilde{S}}\}$
Apply VAT on D_n, returning D^{*}_{n}, P, h. (Algorithm 1)
Apply iVAT on D^{*}_{n}, returning D'^{*}_{n} (Algorithm 2)



(a) Dataset $N = 5000$ (b) VAT for $N = 5000$ (c) iVAT for $N = 5000$ (d) siVAT for $n = 500$

(e) Dataset $N = 1{,}000{,}000$ (f) VAT for $N = 1{,}000{,}000$ (g) iVAT for $N = 1{,}000{,}000$ (h) siVAT for $n = 500$

Figure 2.2: Data scatterplot, VAT, iVAT, and siVAT images for a small (top) and a big dataset (bottom).

siVAT image of $n = 500$ samples (10% of the total dataset) which was made in $1/1000$ fraction of the time taken to compute the full iVAT image.

Figs. 2.2 (e-h) illustrate VAT, iVAT, and siVAT for a big data ($N = 1,000,000$) extracted from same five GM components with equal probabilities. In this case, the VAT and iVAT images cannot be generated due to their high computational complexity and memory constraints, indicated by question marks (?) in Views (f) and (g). However, the siVAT extracts a small size ($n = 500$) MMRS sample from this big data and produces its iVAT image which suggests five clusters present in the big dataset. The sizes of the diagonal blocks in siVAT images show the relative size of each cluster accurately, which supports the Proposition 2.1 (B) that the number of objects selected from each partition in the MMRS sample is proportional to the number of data points in that partition in big data.

The sVAT and siVAT algorithms suggest the number of clusters (k) to seek in the big data. However, these algorithms do not partition the data into k subsets. sVAT-SL and clustering using iVAT (clusiVAT) produce the actual clusters of the big data from the sVAT/siVAT samples. Both algorithms are discussed in clustering (next) section.

Despite its computational efficiency in lower dimensions, Maximin sampling is time-consuming for big, high-dimensional data, and consequently, sVAT/siVAT take much time when the data is large jointly in the number of samples (N) and the number of dimensions (p). Therefore, there is a need of cluster tendency assessment for handling datasets that are jointly large in N and p. Chapter 4 proposes a new algorithm, FensiVAT, to deal with large amounts of high-dimensional datasets. FensiVAT not only provides a reliable visual assessment about the number of clusters that may be present in big, high-dimensional data, but it also produces the actual cluster of the big data in a shorter time than sVAT-sL and clusiVAT, without compromising clustering accuracy.

2.2.4 Cluster tendency assessment for streaming data

For streaming data, VAT/iVAT needs to be (re)executed at each arrival of a new data point, which is time-consuming and very inefficient. Kumar *et al.* [69] proposed incremental methods for VAT and iVAT, called inc-VAT/dec-VAT and inc-iVAT/dec-iVAT, respectively, for visualizing evolving cluster structures in streaming data using a sliding window approach. The inc-VAT/inc-iVAT updates the current minimum spanning tree (MST) used by VAT with an efficient edge

insertion scheme. Similarly, dec-VAT/dec-iVAT efficiently removes a node from the current VAT MST. A sequence of inc-iVAT/dec-iVAT images can be used for (visual) anomaly detection in evolving data streams and for sliding window based cluster assessment for time series data. The pseudocodes of all four algorithms and their procedure subroutines are well-documented in [69]. Since, these incremental methods of VAT and iVAT are used in our proposed approach (presented in Chapter 6) for visualizing evolving cluster structures in high-velocity streaming data, we explain them here in detail.

Consider a time series dataset $X_N = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, having N p-dimensional data points arriving sequentially. Let $N \geq 2$ so that an initial VAT image of X_N can be constructed. When $\mathbf{x}_{N+1}$ arrives, the augmented dataset is denoted with $X_{N+1} = X_N \cup \{\mathbf{x}_{N+1}\}$. VAT on D_N (dissimilarity matrix of X_N) results an $N \times N$ reordered dissimilarity matrix D_N^*, VAT reordering indices P_N, the MST cut magnitude order h_N, and the MST connection indices F_N as outputs (refer to Algorithm 1). VAT reorders the N data points (or objects) in such a way that each point in the reordered matrix is closer than any data point after it to any data point before it (before and after here refer to positions in the reordered list of indices, not to times of data arrival). This property of VAT reordering is the basis for incremental methods of VAT algorithms.

2.2.4.1 inc-VAT

The inc-VAT algorithm comprises three main steps:

1. **Finding the insertion position of the new data point $\mathbf{x}_{N+1}$ in the VAT ordering of X_N:** When the new data point $\mathbf{x}_{N+1}$ arrives, its distances to all (previous) points in X_N, denoted by $V = \{v_1, v_2, ..., v_N\}$, are computed, where v_j $(1 \geq j \geq N)$ is the distance between $\mathbf{x}_j$ and $\mathbf{x}_{N+1}$. Reordering the distances in V using the indices in P_N gives L, the distances of the $\mathbf{x}_{N+1}$ from the VAT reordered data points of $\mathbf{x}_N$, so $L = V_{P_N} = \{v_{P_1}, v_{P_2}, ..., v_{P_N}\}$. First, the insertion position (say i) of the new data point is identified in the VAT reordering of the augmented set X_{N+1}. The condition for determining the new position is satisfied at i when $\min\{L_1, L_2, ..., L_{i-1}\} \leq d_{N_i}$ for $1 \leq i \leq N$. For the augmented dataset X_{N+1}, the new index array P_{N+1} is initialized with the first $i-1$ elements of P_N and the new index, $N+1$, is appended at the end, hence $P_{N+1} = \{P_{N_1}, P_{N_2}, ...P_{N_{i-1}}, N+1\}$. Similarly, the new MST cut magnitude order is initialized as $h_{N+1} = \{h_{N_1}, h_{N_2}, ...h_{N_{i-2}}, \min\{L_1, L_2, ...L_{i-1}\}\}$. and the new

MST connection indices are initialized as $F_{N+1} = \{F_{N_1}, F_{N_2}, ...F_{N_{i-1}}, arg\min(\{L_1, L_2, ...L_{i-1}\})\}$.

2. **Reordering the remaining data points after the insertion position:** Next, the remaining indices in P_N, denoted by $A = \{P_{N_i}, P_{N_{i+1}}, ...P_{N_N}\}$, which are not in P_{N+1}, are reordered. As the indices from A are added to P_{N+1} one by one, the data points corresponding to indices in P_{N+1} can be divided into three groups.

   - Group $G1$ represents the data points, whose indices form the longest subsequence C of P_N i.e., $C = LongestSubsequence(P_N, P_{N+1})$. The remaining indices in P_N, which are not in C are represented by B, i.e., $B = P_N / C$. C is initialized to $\{P_{N_1}, P_{N_2}, ...P_{N_{i-1}}\}$, and B is initialized to $\{P_{N_i}, P_{N_{i+1}}, ...P_{N_N}\}$. The MST connection indices of the data points in B is given by H, which is initialized to $\{F_{N_i}, F_{N_{i+1}}, ...F_{N_N}\}$.
   - Group $G2$ represents the new data point $\mathbf{x}_{N+1}$.
   - Group $G3$ represents the data points corresponding to the remaining indices in P_{N+1}, which are not in $G1$ or $G2$. Let E represent the indices of the data points in $G3$, i.e., $E = P_{N+1} / \{N+1\} / C$. E is initialized to $\emptyset$.

   The next index to be added to P_{N+1} corresponds to the data point represented by indices in A that is nearest to any data point associated with the indices in P_{N+1}. Since the indices in P_{N+1} are divided into three groups $G1$, $G2$, and $G3$, the index of the closest data point is found from each of the three groups. Let w_j and $z_j : 1 \leq j \leq 3$, respectively, represent the index and the distance of the closest data point from G_j group. The closest data point from $G1$ is found using the VAT property of P_{N+1}, from $G2$ using L and A, and from $G3$ using a sub-matrix of the VAT reordered dissimilarity matrix D_N^* of X_N, whose rows and columns are given by indices in A and E, respectively.

   Let $b_j : 1 \leq j \leq 3$ represents the positions in P_{N+1} of the data points in G_j which are closest to w_j. If $z_i = \min(z_j) : 1 \leq j \leq 3$, then z_i is added to h_{N+1}, w_i is added to P_{N+1}, and b_i is added to F_{N+1}. Based on which of the z_1, z_2, and z_3 is minimum, three separate procedures [69] are used to update new reordering indices P_{N+1}, MST cut magnitude order h_{N+1}, MST connection indices F_{N+1}, A, B, E, G, and H. This procedure is repeated until all the remaining data points (indices in A) are added to P_{N+1}. The vector G represents the reordering of the indices of P_N to obtain P_{N+1}.

3. The last step is to compute the new reordered dissimilarity matrix D^*_{N+1}. First, the rows and columns of D^*_N are reordered by the indices of G. Then, the distances of new data point $\mathbf{x}_{N+1}$ to the previous points in L are reordered using the indices of G to obtain L^*. The distance value of 0 is inserted at ith position of L^* to account for the distance of $\mathbf{x}_{N+1}$ to itself. Then, the vector L^* is inserted after $(i-1)$th row and $(i-1)$th column of D^*_N to obtain D^*_{N+1}.

2.2.4.2 inc-iVAT

The inc-iVAT produces the iVAT dissimilarity matrix D'^*_{N+1} for X_{N+1}. The sub-matrix consisting of the first $i-1$ rows and the first $i-1$ columns of D'^*_{N+1} is same as that of D'^*_N, where i is the insertion position of the new data point $\mathbf{x}_{N+1}$. The remaining elements of D'^*_{N+1} are computed using the same procedure as iVAT.

2.2.4.3 dec-VAT

For sliding window based clustering applications for streaming data, it is not sufficient to just add the latest data point to the VAT-generated MST using inc-VAT/inc-iVAT because of the memory requirement to store the $O(N^2)$ size reordered distance matrix. A suitable strategy is to keep deleting oldest data points from the MST to keep the number of data points manageable (assuming a sliding window of fixed size). The decremental version of the VAT, dec-VAT, achieves this using the VAT reordering information of X_N, similar to inc-VAT.

If a data point (say) $\mathbf{x}_l$ is to be removed from X_N, the modified dataset is denoted as $X_{N-1} = X_N / \{\mathbf{x}_l\}$. The dec-VAT comprises two main steps:

1. **Finding the position of the to-be-removed data point $\mathbf{x}_l$ in the VAT ordering of X_N:** The first step in dec-VAT is to determine the position i of $\mathbf{x}_l$ in P_N using $i = arg(P_N = l)$. Then, the set of data points after position i in P_N, that are connected to $\mathbf{x}_l$ in the current MST of X_N, are determined. This is obtained using F_N, the MST connection indices of P_N. Let these data points be represented by J. The initializations of P_{N-1}, F_{N-1}, and h_{N-1} differs in two following cases:

    - If $\mathbf{x}_l$ is a leaf node (the nodes that have only one branch edge) i.e., $J = \emptyset$, then, only the node $\mathbf{x}_l$ is to be removed from the MST of X_N to obtain the VAT reordering of X_{N-1}.

This is achieved by removing the i-th element (position of $\mathbf{x}_l$ in P_N) of P_N and F_N, and the $(i-1)$-th element of h_N to obtain P_{N-1}, F_{N-1}, and h_{N-1}, respectively. Since, the i-th element $\mathbf{x}_l$ is deleted, the values of F_{N-1}, which are greater than i, are decreased by 1. The i-th row and i-th column of D_N^* is also deleted to obtain D_{N-1}^*.

- If $\mathbf{x}_l$ is not a leaf node i.e., $J \neq \emptyset$, then the P_{N-1}, F_{N-1}, and h_{N-1} are initialized based on whether the data point to be removed $\mathbf{x}_l$ is the first element of P_N or not. If $\mathbf{x}_l = P_{N_1}$, the VAT reordering starts from the second element of P_N, P_{N_2}, else, the P_{N-1} ad F_{N-1} are initialized with the first $i-1$ elements of P_N and F_N, respectively, and the MST cut magnitude order is initialized as $h_{N-1} = \{h_{N_1}, h_{N_2}, ...h_{N_{i-2}}\}$.

Similar to inc-VAT, $A = \{P_{N_{i+1}}, P_{N_{i+2}}, ...P_{N_N}\}$ represents the remaining indices to be reordered and $H = \{F_{N_{i+1}}, F_{N_{i+2}}, ...F_{N_N}\}$ represents their MST connection indices. $C = LongestSubsequence(P_N, \{P_{N-1}, i\})$ represents data points belonging to $G1$. The remaining indices in P_N, which are not in C, are given by $B = P_N / C$. Since no new data point is added here, $G2 = \emptyset$. $G3$ represents the data points in P_{N-1}, which are not in $G1$. Let E represents the indices of the data points in $G2$, so that $E = \{P_{N-1}, i\} / C$. The vector G represents the reordering of the indices of P_N to obtain P_{N-1}.

2. **Reordering the remaining data points after the deletion position:** As $\mathbf{x}_l$ is removed from the MST of D_N, the edges joining $\mathbf{x}_l$ to the data points whose indices are given by J are cut. The next nearest point to the current MST of X_{N-1} is provided by B_1. Based on whether $B_1 = J_1$ or not, two different procedures [69] are used to insert the remaining elements having indices A.

   - If $B_1 = J_1$, the distance of data points in A (indices of the remaining points) to the current points in P_{N-1} (whose rearranged order is given by G) is computed. In this case, the minimum distance (z), nearest data point (w), and MST connection index (l) are appended to the appropriate matrices P_{N-1}, h_{N-1}, F_{N-1}, and G. If $w \in J$, then we delete w from A and J, otherwise add it to E. Additionally, if $w = J_1$, then C, B, and H are updated as the data point attached to the deleted node of the MST of X_N is reconnected to the MST of X_{N-1}.
   - If $B_1 \neq J_1$, the remaining data points whose indices are given by A are inserted into the

MST of X_{N-1} using the same procedure that was used in inc-VAT (using $G1$, $G2$, and $G3$), with $G2 = \emptyset$. Subsequently, based on which of the z_1 and z_3 is minimum, two different procedures (similar to inc-VAT) are used to update P_{N-1}, h_{N-1}, and F_{N-1}.

2.2.4.4 dec-iVAT

The dec-iVAT provides the iVAT dissimilarity matrix D'^{*}_{N-1} of X_{N-1}. The sub-matrix consisting of the first $i-1$ rows and the first $i-1$ columns of D'^{*}_{N-1} is same as that of D'^{*}_{N}. The remaining elements of D'^{*}_{N-1} are computed using the same procedure as iVAT Algorithm.

Although, the inc-VAT/dec-VAT and inc-iVAT/dec-iVAT significantly lower the time complexity of VAT and iVAT, the output of these algorithms is a reordered distance matrix that has N^2 elements, so storing and visualizing them could be problematic due to software and hardware constraints as N becomes large. Both sVAT and siVAT are suitable for cluster tendency assessment of big data. However, to handle streaming data, they also need to be re(applied) each time a new data point or a chunk of new data points arrive, which is not feasible due to computational complexities associated with retraining at each instance of the new data point or new chunk arrival.

To address this problem, Chapter 6 proposes an incremental version of the siVAT algorithm, inc-siVAT, for online visual assessment of evolving cluster structures in high-velocity, streaming data.

2.3 Clustering

Once the number of clusters k is known, the next task is clustering, i.e., partitioning the data into k subsets. Clustering is an essential method of exploratory data analysis in which data are partitioned into several (k) subsets such that objects in each subset are similar to each other and dissimilar to members of other subsets. Clustering can be roughly distinguished into two types: hard and soft. In hard (crisp) clustering, each object either belongs to only a cluster or not, whereas, in soft (fuzzy/probabilistic) clustering, each object belongs to each cluster to a certain degree of membership. Several paper and books [105–108] discuss different clustering algorithms, which can be broadly classified into four main categories: partitioning, hierarchical, distribution-based, and density-based methods. Below, we first discuss traditional clustering algorithms, followed by

different strategies and algorithms for big data clustering. The classification and summary of these methods are provided in Table 2.1.

2.3.1 Partitioning-based methods

The partitioning-based clustering methods attempt to determine partitions to optimize a certain objective function defined in advance. Generally, these algorithms are combinatorial optimization algorithms which means they minimize a given objective criterion by iteratively relocating data points between clusters until a (locally) optimal partition is obtained. The most intuitive and commonly used objective function of partitioning methods is the squared error function. Some well-known techniques in this category are k-means, k-medoids, and fuzzy c-means.

2.3.1.1 k-means

The k-means algorithm is one of the most popular, and computationally efficient partitional clustering algorithms. Llyod [109] proposed standard k-means algorithm in 1957 as a pulse code modulation technique which was published as a Balls Laboratory paper [110] (published as a journal [109] in 1982). Therefore, k means is also refereed to Lloyd's algorithm. The k-means algorithm consists of two steps. It starts with selecting k initial cluster centroids from the input data, where each centroid is a representative of a cluster. In the first (assignment) step, each data point is assigned to the closest center using some dissimilarity measure function (usually the Euclidean or L2 distance for numerical data). Then, in the second (update) step, each centroid is updated using the points assigned to its cluster. The assignment and update steps are repeated until the objective function converges to an optimum solution. k-means is intuitive and easy to implement, however, it has some drawbacks. The major limitation is that it requires the number of clusters, k, to be known prior to clustering, which is usually unknown for real-world data. It is very sensitive to the selection of initial centroids which may result in an suboptimal solution. Moreover, it works assuming that the variance of distribution of each attribute is spherical, and each cluster has roughly equal number of observations.

2.3.1.2 Fuzzy c-means (FCM)

Fuzzy c means [48, 111, 112] is a soft clustering method which is based on k-means concept of partitioning data into clusters i.e., it iteratively searches the cluster centers and update the memberships of objects in each cluster until it converges to an optimal solution. The FCM clustering is obtained by minimizing the following objective function (within membership matrix constraints (2.1)):

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{c} u_{ij}^m ||\mathbf{x}_i - \mathbf{v}_j||^2, \tag{2.5}$$

where m is a fuzziness factor (≥ 1), u_{ij} is the degree of membership of object $\mathbf{x}_i$ in the cluster j, c[1] is the number of clusters, $\mathbf{v}_j$ is the p-dimensional center of cluster j, and $||*||$ is a distance norm. The memberships u_{ij} and centroid $\mathbf{v}_j$ are updated using following equations:

$$u_{ij} = \frac{1}{\sum_{l=1}^{c} \left(\frac{||\mathbf{x}_i - \mathbf{v}_j||}{||\mathbf{x}_i - \mathbf{v}_l||} \right)^{\frac{2}{m-1}}}, \quad 1 \leq i \leq c, \quad 1 \leq j \leq N \tag{2.6}$$

$$\mathbf{v}_j = \frac{\sum_{i=1}^{N} u_{ij}^m \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}^m}, \quad 1 \leq j \leq c. \tag{2.7}$$

FCM also suffers from most of the problems suffered by k-means. However, the fuzzy partition smoothes the search space, thus making optimization easier and therefore, providing better results, especially in recovering from bad initialization of centroids.

2.3.1.3 k-medoids

The k-medoid [113, 114] algorithm is related to k-means that seeks a subset of points, called medoids, such that average dissimilarity between them and their closest points (all the objects in the cluster) is minimal. A medoid is a most centrally located point in the cluster, and hence, can be considered as a representative point of the cluster. Since the k-medoids algorithm attempts to minimize a sum of pairwise dissimilarity instead of a sum of squared Euclidean distances, it is robust to noise and outliers as compared to the k-means algorithm. The most common realization

[1] In fuzzy clustering, c is used to define the number of clusters. Therefore, we use k and c interchangeably to represent the number of clusters for crisp and soft (fuzzy) clustering, respectively, in this thesis.

of *k*-medoid clustering is the *Partitioning around Medoid* (PAM) algorithm. PAM uses a greedy search which may not find the optimum solution, but it is faster than exhaustive search.

2.3.2 Hierarchical methods

In hierarchical clustering, data are organized in a hierarchical manner based on the proximity between pairs of data points. Hierarchical methods can be agglomerative (bottom-up) and divisive (top-down). An agglomerative clustering starts with one object for each cluster and recursively merges pairs of clusters as it moves up the hierarchy. A divisive clustering starts with the dataset as one cluster and splits them in a top-down fashion as it moves down in the hierarchy. The results of hierarchical clustering can be represented as a dendrogram [115]. A dendrogram is a type of tree diagram which represents the nested grouping of data points and similarity levels where groupings change. The dendrogram can be broken at certain levels based on the given criterion such as the requested number of clusters or desired similarity to form clusters.

Measures for proximity among clusters are single linkage (minimum distance between any two points from the cluster), complete linkage (maximum distance between any two points from the cluster), and their variations such as mean (average linkage [116, 117]) or median of distances among all data points between clusters. Three prominent examples of hierarchical clustering are SLINK [118], CLINK [119], and Ward's method [120] which are variants of single linkage [121], complete linkage [122], and the minimum variance [123] algorithms. The advantage of hierarchical clustering includes the easy handling of any similarity measure and the flexibility regarding the level of granularity. However, it has a major drawback that once a merge or split process is performed, this cannot be undone.

2.3.3 Density-based methods

In density-based clustering [124, 125], data points are separated based on their regions of density, boundary, and connectivity to form clusters. The notion of a cluster is identified by dense regions with nearest neighbour concept. The notion of dense region results in discovering clusters of arbitrary shapes. This also provides natural protection against outliers [58]. DBSCAN, OPTICS, and DENCLUE are well-known density based algorithms.

2.3.3.1 DBSCAN

The most popular density-based algorithm is *Density Based Clustering of Applications with Noise* (DBSCAN) [124]. DBSCAN searches for the regions that have at least a certain number of data points (*MinPts*), within a certain distance threshold called *reachability distance* (ε- neighbourhood). Based on these user-defined parameters (*MinPts* and ε), each data point is labeled as either core, border, or noise point. DBSCAN can discover arbitrary shape clusters, and it is effective even for a spatial database. However, it is susceptible to the choice of its parameter. Moreover, it performs poor in identifying meaningful clusters in data of varying density.

2.3.3.2 OPTICS

Ordering Points To Identify the Cluster Structure (OPTICS) [126] is similar to DBSCAN but it overcomes DBSCAN's weakness in detecting meaningful clusters when they vary widely in their densities. To address this problem, the data points are (linearly) ordered such that points which are closest become neighbours in the ordering, similar to single linkage clustering. The ordering works on the principle that sparsely populated clusters have a higher value of ε-neighbourhood and vice versa. OPTICS produces a hierarchical result for different values of the *reachability distance* parameter. Unlike DBSCAN, it ensures good quality clustering by maintaining the order in which data points are processed, i.e., high-density clusters are preferred over lower-density clusters.

2.3.3.3 DENCLUE

DENsity based CLUstEring (DENCLUE) [127] models the cluster distribution as the sum of influence functions of all data points. The influence function describes the impact of a data point within its neighbourhood. The cluster can be determined mathematically as density attractors, which are local maxima of the overall density function. Density attractors are determined using a hill-climbing method guided by the gradient of the overall density function. Although DENCLUE can detect the clusters of arbitrary shapes and sizes and can handle noise, it shares same limitations of DBSCAN.

2.3.4 Distribution-based methods

Distribution-based methods are based on the assumption that data is generated from a mixture of underlying probability distributions. Such methods optimize the fit between the given data and a predefined mathematical model. These methods aim to identify the number of distributions and their parameters based on the standard statistics, taking noise into account, thus yielding a robust clustering result. Most of the distributed-based methods assume that the individual component of the mixture is Gaussian, and in such case, its parameters are estimated by *expectation maximization* (EM) procedure. Distribution-based methods can produce complex models for clusters, however, they may suffer from over-fitting if the number of mixtures is not fixed. Moreover, there may be no concisely defined mathematical model for many real datasets.

2.3.4.1 Expectation maximization (EM)

Expectation maximization [128] is a well-known method to estimate the maximum likelihood parameters of a statistical model such as a Gaussian mixture model. A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. EM algorithm provides an iterative method to find the maximum likelihood estimators of distributions. First, it assumes random components and computes for each point a probability of being generated by each component of the model. Then, it tweaks (updates) the parameters to maximize the likelihood of the data given those assignments of data points to mixture components. Repeating this process is guaranteed to find a locally optimal solution for the model parameters estimate. However, it is sensitive to the selection of initial parameters, similar to k-means algorithm. Moreover, it has a slow convergence rate which results in a decreased precision of output within a finite number of steps [129].

2.3.5 Clustering big data

Traditional clustering algorithms cannot cope with big data because of their high complexity and computational cost. The main objective is to speed up the clustering algorithms without compromising clustering quality. There are many ways to classify clustering algorithms that scale up to big static data, based on their style of processing the inputs: sampling, streaming, incremental,

distributed (parallel) [41, 57]. Fig. 2.3 illustrates three most common approaches [41] to address the problem of clustering for big static datasets. For convenience, let BD_N represent big static data, and let SD_n denote small subsets of BD_N.

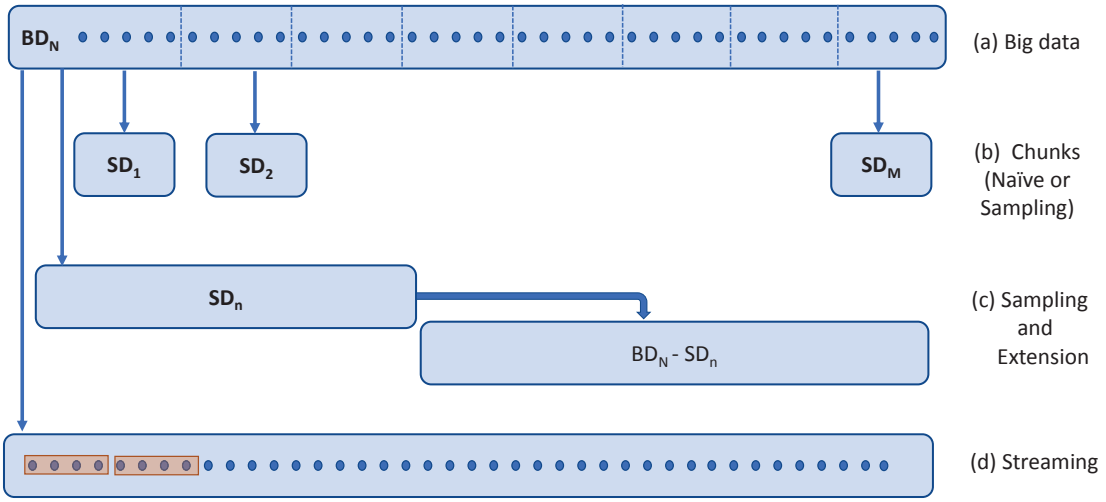


Figure 2.3: Three ways to make big data look small [41]

1. Naive or Sample Chunks: Fig. 2.3(b) depicts clustering in loadable chunks of big static data in which BD_N is partitioned into M loadable subsets (or chunks) $\{SD_i\}_{i=1}^{M}$. There are two ways to generate and handle these small size chunks:

   - Naive Chunking: The partitioning of BD_N is done by cutting successive chunks (shown with the dashed vertical bar in Fig. 2.3 (a)) out of the big data.
   - Sample Chunking: Another way is to build each chunk with a sampling function (random or intelligent sampling) that hopefully collects representative samples for each chunk from each of the k clusters assumed to be in the big data.

   Once the chunks are obtained, each chunk is clustered literally by any suitable clustering algorithm which produces M literal partition $U_1, U_2, ...U_M$. Fig. 2.4 shows two different way of processing chunks to obtain the final clustering result.

   - The first way of processing naive chunks is to process each chunk in *parallel* (or *distributed*) fashion, which amounts to the independent processing of each chunk. Chunks can be processed using either a parallel or distributed clustering algorithm

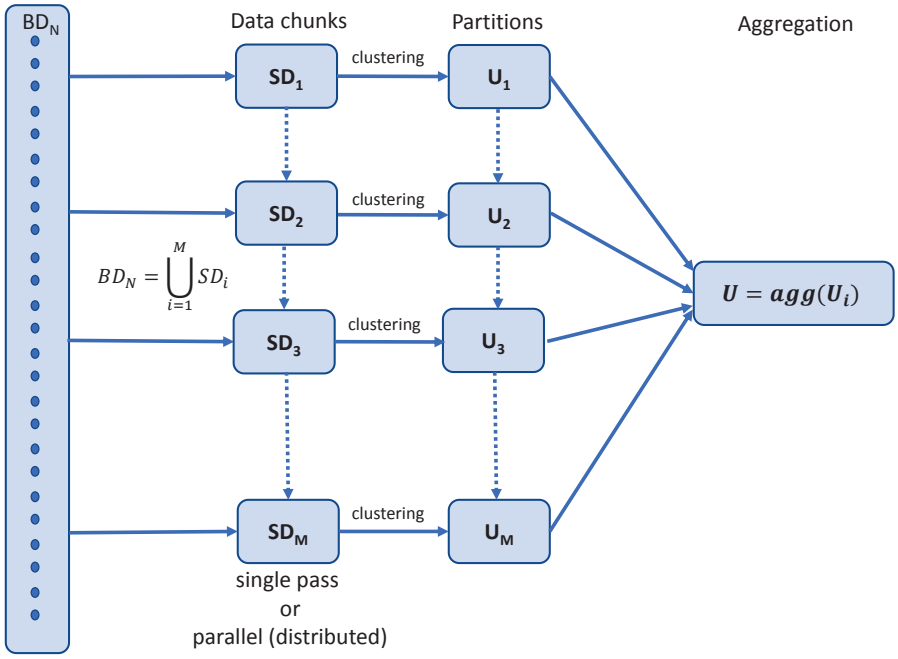


Figure 2.4: Processing naive or sample chunks of big data [41]

or parallel processing architecture (e.g., MapReduce) on multi-machine platform [57]. After having a partition from each of the M chunks, they need to be combined into a partition for big data. The problem of aggregating all literal partitions $U_1, U_2, ...U_M$ to obtain final partition is often known as *cluster ensembles* or *ensemble clustering*. The aim of the cluster ensemble approach is to obtain a final partition that provides a better idea of cluster structure than any of the individual partitions that contribute to the aggregation.

The usual assumption in ensemble clustering is that the M partitions are all of the same dataset, made by different clustering algorithms or by different initializations or different parameter settings of a clustering algorithm. Therefore, ensemble clustering methods are not usually applicable to naive chunking approach for clustering in big data, because literal chunks or partitions do not share common data. However, they can be useful in the distributed case depending on the objective of the model. They are also useful for high-dimensional data clustering [130], where all literal subsets have the same data points in different dimensions or each subset is processed by different clustering algorithms.

- Second way of processing naive chunks is to cluster the first chunk (usually the biggest) with a literal algorithm, summarize the results, and then move along to the next chunk (the vertical path is shown by dashed lines in Fig. 2.4). This way of handling naive chunks is sometimes called *partial data access*, and if only one pass is made through BD_N, the method is called is *single pass* (or *single scan*) algorithm [41].

2. Sampling and Extension: Fig. 2.3(c) illustrates a method called sampling followed by non-iterative extension. In this approach, a loadable sample (of size n) $SD_n \subset BD_N$ is built by sampling, and a literal partition is obtained by applying a clustering algorithm on it. Then, the sample partition is extended to the rest of the samples non-iteratively using nearest neighbour approach, also called *nearest prototyping rule* (NPR)

3. Streaming (Sequential): Fig. 2.3(d) shows one alternative to the collection, storage, sampling, and batch processing. In this approach, the points are regarded as streaming or sequential data, so BD_N is never stored. It is replaced by a data stream, say $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ that arrive as a time-ordered sequence or small size chunks in a sequential manner. In this approach, the idea is to build clusters using the first few points or chunks and then incrementally incorporate new inputs to update clustering results. This way it avoids the storage of big data and time complexities associated with batch processing [41].

Below we discuss some well-known algorithms developed for big data clustering in the following categories.

- Sampling based techniques
- Streaming (sequential) techniques
- Parallel processing (MapReduce)
- Dimensionality reduction based technique

Sampling based techniques Sampling based techniques speeds up the clustering task because computation is performed on small subsets. Consequently, the complexity and memory space needed for clustering decreases.

2.3.5.1 Clustering Large Applications (CLARA)

PAM worked well on very small datasets, but its computational complexity is $O(k(n-k)^2)$. CLARA [77] extends the PAM approach for a large number of objects using sampling and extension approach. CLARA begins by randomly drawing a small sample ($40+2k$ data points) from the big data using uniform random sampling. *Partitioning Around Medoids* (PAM) then operates on these data points to find an optimal set of k-medoids for the sample. The remaining objects are labeled with the *nearest prototype rule* (NPR), and the average dissimilarity between crisp clusters is computed. If this improves the objective, retain these k medoids and continue. To reduce sampling bias, CLARA repeats the sampling and clustering process multiple numbers of times and subsequently selects the set of medoids with the minimal cost as the final clustering result. However, the best k-medoids may not be selected in any of the samples, giving poor clustering result.

2.3.5.2 Clustering Large Applications based on Randomized Search (CLARANS)

CLARANS [131] improves the quality and scalability of CLARA using dynamic sampling based on the randomized search. CLARANS views the desired medoids as special nodes in a graph representation of the data. The best medoids are selected using serial randomized search. While CLARA draws a random sample of data points at the beginning of a search, CLARANS draws a random sample of neighbors of the data points in each step of its search, and hence, is more efficient than the sampling method used in CLARA.

2.3.5.3 Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)

If the dataset is too large to fit in memory, it causes a lot of overhead for a clustering algorithm in maintaining high clustering quality while minimizing the cost of additional *input-output* (I/O) operation. BIRCH [132] is perhaps the first algorithm explicitly designed to solve this problem using a compressed representation (summarization) of part of the input data. In the initial phase of BIRCH, as much data as possible is scanned, clustered using a hierarchical clustering algorithm, and then each cluster is summarized by a special data structure called *clustering feature* (CF). Once the CFs are calculated, the data underlying them are deleted, and only the set which summarizes

the process is retained. The CFs are stored in the highly balanced cluster feature tree. As new data points arrive, the nodes of the tree are dynamically built and inserted into the tree. BIRCH is shown to perform well on several large datasets, and it is better to CLARANS in terms of run-time, space, and in handling outliers.

2.3.5.4 Clustering Using REpresentative (CURE)

A shortcoming of the previously described clustering algorithms (CLARANS, BIRCH) is that they consider only one point (centroid/medoid) to represent a cluster, which means they work well in identifying cluster of spherical shapes but do not perform well in identifying clusters of non-spherical or arbitrary shapes. To deal with this challenge, CURE [78] represents clusters by well-scattered points. CURE is a sampling and extension based algorithm that adopts a middle ground between a type of hierarchical clustering and centroid-based algorithm. It randomly samples a constant number of points from the large dataset so that the selected (representative) points (hopefully) retain the geometry of the entire dataset. In CURE, each cluster is represented by a fixed number, g, of well-scattered points, which are shrunk towards the centroid of a cluster by a fraction, α. These scattered points after shrinking are defined as representatives of the cluster. Then, the clusters with the nearest representative points are merged at each step until the desired value of k is attained, akin to SL. Once CURE has labeled the sample, the extension function that assigns cluster labels to the remaining data points employs a fraction of randomly selected representative points for each of the final k clusters. Two main data structures are used in CURE for efficient search: heap and k-d tree. Heap is used for tracking the distance of each existing cluster to its closest cluster, and the k-d tree is used to store all the representative points for each cluster. As the number of clusters in the data increases, the probability of CURE samples retaining the data geometry decreases, and hence, the accuracy of CURE decreases.

2.3.5.5 Single pass fuzzy c-means

A single pass fuzzy c-means algorithm was presented in [74] for large datasets, which produces a final clustering in a single pass through the data with limited memory allocation. It first divides the N points into M chunks and requires only a portion of the data (one of the M chunks) to

be stored in the memory for k-means clustering. Then, it updates the current model (k weighted centroids) over the contents of the buffer (other chunks) iteratively until the whole dataset is loaded and processed. After obtaining the final k centroids, the big data are labeled based on the label of the nearest object, i.e., the standard k-means *nearest prototype rule* (NPR).

2.3.5.6 Mini-batch k-means

Mini-batch k-means (MBKM) [76], also known as web-scale k-means, is a k-means variation using "mini-batch" samples for datasets that do not fit into memory. This scheme processes small random batches (mini-batches) of the dataset to reduce the computation time while attempting to optimize the same objective function. The algorithm iterates between two major steps. In the first step, samples are drawn randomly from the dataset, to form a mini batch. Then, each data point in the batch is assigned to a cluster (nearest centroid). In the second step, a new random sample is obtained and used to update the centroid until termination is achieved. For each sample in the batch, the assigned centroid is updated using a convex combination of the samples of mini batch and previous samples assigned to that centroid, applying a learning rate. The learning rate is the inverse of the number of samples in each batch assigned to a centroid during the process. The effect of new samples decreases as the number of iterations increases. These steps are performed until termination or until a pre-determined number of iterations is obtained. MBKM reduces the computation time by not using all the dataset but a subsample of a fixed size in each iteration, however, at the cost of lower cluster quality.

2.3.5.7 clustering using iVAT (clusiVAT)

sVAT and siVAT algorithms (discussed in Section 2.2.3) answer the tendency assessment problem by suggesting the number of clusters to seek in the big data, however, they do not produce the actual clusters of the data. clusiVAT produces the actual cluster of the big data X from the siVAT samples, as described below:

Single-linkage clusters are always diagonally aligned in the VAT/iVAT ordered images [133, 134]. Having an estimate of k from the iVAT image of Maximin random sample (MMRS) produced by siVAT, the longest $(k-1)$ edges of the MST are cut to form the corresponding k aligned

partition. For big data clustering, clusiVAT [63, 135] use this idea to extend the k partition of D_n (or D_n^*) non-iteratively to the $(N-n)$ unlabeled objects in X using the NPR. sVAT-SL is similar to clusiVAT except that sVAT-SL applies sVAT instead of siVAT, after the sampling step.

Both sVAT-SL and clusiVAT are adequate for large sample size datasets, however, they still suffer from large computation time when the dataset is large in the number of dimensions. The two main time-consuming steps in both algorithms for large, high-dimensional data clustering are (i) the Maximin step of MMRS sampling; and (ii) Extension. The computational complexities in the first and second steps of MMRS sampling in siVAT are $O(pk'N)$, and the last stage requires $O(pn^2)$ operations to build sample $\tilde{S}$ (refer to Section 2.2.3). Despite its computational efficiency in lower dimensions, MMRS sampling is computationally expensive in high-dimensions. The computational complexity of *extension* step is $O(pn(N-n))$. Because, N and $(N-n)$ can be very large for big datasets, and the distance computations are performed in the original (high) p-dimension, siVAT-SL and clusiVAT take a large amount of time to cluster large volumes of high-dimensional data. Chapter 4 presents a fast cluster tendency assessment and subsequent clustering algorithm, FensiVAT, which can cluster large volumes of high-dimensional data in significantly less time, without compromising clustering accuracy.

Algorithm 5 clusiVAT

Input: Dataset X or $N \times N$ dissimilarity matrix, D_N
k': an overestimate of the true number of clusers, k, in X
n: an approximate sample size.
Output: D'^*_n - $n \times n$ iVAT dissimilarity matrix of D_n
U - cluster membership vector of data points in X.

siVAT

Apply Maximin Random Sampling on X returning $\tilde{S}$ — Algorithm 3
Compute $D_n = dist\{\mathbf{x}_{\tilde{S}}, \mathbf{x}_{\tilde{S}}\}$
Apply VAT/iVAT on D_n returning D'^*_n, P, h — Algorithms 1 and 2
Choose the number of clusters k using image of D'^*_n.

Clustering

Find indices t of k largest values in MST cut magnitudes h.
Form the aligned partition: $U^* = \{t_1 : t_2 - t_1 : \ldots : t_k - t_{k-1}\}$
$u_{\tilde{S}} = u^*_{P_i},\ 1 \le i \le k$.

Extension

for $\hat{\mathbf{x}} \in \hat{X} = X - X_{\tilde{S}}$ **do**
  $j = \arg\min_{i \in \tilde{S}} \{dist\{\hat{\mathbf{x}}, \mathbf{x}_i\}\}$
  $u^{(i)}_{\hat{\mathbf{x}}} = u_l$ — **Nearest prototype rule (NPR)**
end for

Sequential clustering techniques High-velocity data streams form a significant part of big data. For example, most wireless sensor networks continuously transmit reports of sensor data, e.g., temperature, humidity and barometric pressure for monitoring applications. The amount of data generated by streaming processes is usually too large for collection and batch storage, and hence, such type of data is interpreted as a different instance of big data. In a streaming environment, clustering is often used to summarize or compress the data stream into a representative model of the data. Most of the streaming clustering algorithms usually have a change detection algorithm, which identifies a change in the data streams as potential times for creating/deleting/updating cluster(s) in the data.

Streaming clustering algorithms can be grouped into two groups based on the assumptions made by each method: (i) *one-pass*, and (ii) *evolving* methods. One-pass (or single-pass) methods assume that data streams follow only one model throughout its lifetime, and hence, they cannot handle evolving data distributions. The one-pass family of clustering algorithms include mainly the first generation of *scalable* clustering algorithms e.g., BIRCH [132], Randomized k-means [136], clustering with fractals [137], STREAM [65].

The evolving methods view the data streams evolving over time, where clusters may appear and/or disappear in streaming data as time progresses. These clusters may contain timely information about anomalies, switching between events, or evolving drift, and so on, that requires immediate action. Cluster partitions on evolving data streams are often computed based on certain time intervals (or windows). There are three well-known window-based methods: landmark window, damped window, and sliding window [68]. Landmark window models consider the data stream from the beginning until now. An example of the landmark window model is the CluStream algorithm [64]. Damped window models associate weights, also called a forgetting or decay factor, with the data in the stream such that higher weights are given to recent data than those in the past. An example of the damped window model is the Den-Stream algorithm [68]. The sliding-window based approach considers the data from now up to a certain range in the past. It is the most common approach to visualize evolving cluster structure in streaming data. An example of the sliding window based model is the SWClustering algorithm [138].

Some other examples of evolving methods are TRAC-Streams [139], STREAM fuzzy k-means [140], and Sequential leader algorithm [141]. Aggrawal [142, 143] provides an extensive

survey of stream clustering algorithms. Below, we discuss some popular clustering techniques for streaming data.

2.3.5.8 STREAM

STREAM [65] algorithm is a divide and conquer based approach that divides the data into chunks, each of them is of manageable size and fits into main memory. Then, it applies a *k*-medians algorithm to each chunk to find its *k* optimal medoid representatives. After processing the entire data stream, all the representative medoids of different chunks are clustered together using a final application of *k*-medoid algorithm. The quality of the final output depends on the manner in which data is partitioned into chunks.

2.3.5.9 CluStream

CluStream [64] framework consists of online and offline components. Online component computes statistical information about the data locality in terms of micro-clusters. Each micro-cluster is represented by a set of *clustering feature* (CF) that incorporates temporal feature in CF's used by BIRCH. The micro-clusters are stored in main memory in the form a pyramidal tree structure which offers an efficient means for recovering historical information about clusters across time. When a new data point arrives, the micro-clusters are updated in order to reflect changes. Offline analytical component applies a variant of *k*-means algorithm on the micro-clusters to obtain final clusters from the stream. CluStream offers a straight-forward approach to summarize data streams. However, it maintains a constant number of micro-clusters, hence it might consider some outliers as real clusters, or even split some of the good clusters.

2.3.5.10 DenStream

Inspired by DBSCAN and CluStream, Cao *et al.* [68] presented DenStream that makes no assumption on the number of clusters, discover arbitrary shaped clusters, and can handle outliers. In DenStream, each data point is given a weight using a function $f(t) = 2^{-\lambda.t}$, where t is the current time, and $\lambda > 0$ is a constant that reflects more importance to most recent data. DenStream also consists of two components: (i) online component incrementally maintains a set of

potential-micro-clusters (p-micro-clusters), and outlier-micro-clusters (o-micro-clusters), and (ii) offline component generates final clusters by applying DBSCAN on the set of p-micro-clusters maintained by the online component.

2.3.5.11 D-Stream

D-Stream [67] is a grid-based and density-based clustering algorithm similar to DenStream, hence it shares the same advantages as DenStream. In D-Stream, each data point is assigned a density coefficient which decays over time: $D(x,t) = \lambda^{t-t_c}$, where t is the current time and t_c is the arrival time of point **x**. Online component maps each data point into a grid, and the offline component computes the grid density and clusters the grids based on the density. The overall density of a grid is defined as the sum of the density coefficients for all the points belonging to that grid. This density is used to determine if the grid is dense or not. This technique makes high-speed data stream clustering feasible without degrading the clustering quality.

Parallel data clustering techniques Parallel data clustering methods are usually implemented on multi-machine platforms. These techniques break the big data into smaller chunks which can be loaded on different machines and then uses their processing power to cluster big data. Parallel clustering can be performed using either clustering algorithm which adapts parallel processing of chunks or using automated distributing architecture (MapReduce) on multiple machines. In parallel clustering, developers need to configure the data distribution and networking process among multiple machines which makes it complicated and time-consuming. Whereas, MapReduce relieves programmers from unnecessary networking problems and concepts such as fault tolerance, load balancing, and data distribution, by handling them automatically.

In parallel clustering, data are divided into chunks that are distributed over machines. Then, each machine performs clustering individually on the assigned chunk of the data. Two main challenges with these methods are minimizing data traffic and their lower accuracy in comparison to their serial implementation. Lower accuracy can be due to the two main reasons: first, different clustering algorithms are used in different machines, and secondly, each chunk may have different data distribution resulting in poor accuracy. Examples of parallel/distributed clustering algorithms are DBDC [144] (distributed and density-based clustering), ParMETIS [145] (a parallel version of

METIS [146]), and G-DBSCAN [147] (GPU accelerated DBSCAN).

MapReduce is a programming model for processing big data with a parallel or distributed algorithm on a cluster. It contains two important task, namely *map* and *reduce*. *Map* function maps a set of data to another set of data, where individual elements are represented as tuples (key/value pairs). *Reduce* function summarizes the *map* output. Specifically, it takes the *map* output as an input and combines those data tuples into a smaller set of tuples. The key contributions of the MapReduce framework are scalability, load scheduling, and fault-tolerance achieved for a variety of applications. MapReduce libraries have been written in many languages like Java, C#, and C++. A popular open-source implementation that has support for distributed scheduling is part of Apache Hadoop. A few examples of parallel implementation of various clustering algorithms using MapReduce include [148–152].

PKMeans [148] is a distributed version of k-means clustering algorithm that distributed the computation between multiple machines using MapReduce framework to scale up the process. An individual clustering is performed in the *mapper* and then general clustering is performed in the *reducer*. Ene *et al.* [149] introduced the first approximation algorithms for the k-center and k-median on MapReduce. They adopt an iterative sampling strategy to reduce the data size and run a (time-consuming) clustering algorithm, such as local search or Llyod's algorithm on resulting subset. These algorithms run in a constant number of MapReduce rounds and achieve a constant factor approximation.

Ferreira *et al.* [150] examine MapReduce for clustering the datasets that do not fit even on a single disk. To minimize the I/O and network cost, they propose the *Best of both Worlds* (BOW) and derive its cost function that dynamically determines the best strategy to balance the cost for disk accesses and network accesses. They showed that it could work with most of the serial clustering algorithms as a plugged-in clustering subroutine. It matches the clustering quality of the serial algorithm with near-linear scale-up.

Jin *et al.* [151] presented a *Distributed Single Linkage hierarchical Clustering* (DiSC) algorithm using a MapReduce framework. The key idea is to divide the original problem into a set of overlapped subproblems, to solve each subproblem, and to merge the sub-solutions to obtain an overall solution. DiSC is a memory efficient algorithm, and it scales linearly. MR-DBSCAN [152] is a scalable MapReduce-based DBSCAN clustering algorithm in which all the

Table 2.1: Clustering algorithms for big data

Categories	Algorithm name	Dataset size (N)	Handling high-dimensionality	Number of (main) input parameters	Algorithm complexity	Underlying strategy to handle big data
Partitioning	k-means	Large	No	1	$O(Npkt)$	linear in N
	FCM	Large	No	1	$O(Npc^2)$	linear in N
	k-medoids	Small	Yes	1	$O(k(N-k)^2)$	not suitable for big data
	single pass k-means	Large	No	2	$O(Npk)$	Sequential (one-scan)
	mini-batch k-means	Large	No	3	$O(npkt)$	Sampling (mini-batches)
	CLARA	Large	No	1	$O(kn^2+k(N-k))$	Sampling and extension
	CLARANS	Large	No	2	$O(N^2)$	Sampling and randomized search
	STREAM	Large	No	4	$O(Npkt)$	Sequential (chunks processing)
	Parallel k-means	Large	No	1	$O(Npkt)$	Distributed (MapReduce)
	CluStream	Large	No	5	$O(Npkt)$	Data summarization, sequential (data stream processing)
Hierarchical	CURE	Large	Yes	2	$O(n^2\log(n)$	Sampling and extension
	BIRCH	Large	Yes	2	$O(Np)$	Data summarization
	sVAT-SL/clusiVAT	Large	Yes	2	$max(O(pk'N), pn^2)$	Sampling and extension
	DisC	Large	No	1	$O(N\log(N))$	Distributed (MapReduce)
Density-based	DBSCAN	Large	No	2	$O(N\log(N))$	Process each point once spatial indexing
	OPTICS	Large	No	2	$O(N\log(N))$	Process each point once, spatial indexing
	DENCLUE	Large	Yes	2	$O(\log(Np))$	Uses tree-based access structure
	DenStream	Large	No	5	$O(N\log(N))$	Sequential (data stream processing)
	D-Stream	Large	No	5	$O(N\log(N))$	Sequential (data stream processing)
Distribution-based	EM	Large	No	3	$O(Nkp)$	linear in N

t:number of iterations, N: data size, n: sample size, p: dimension, k, c: number of crisp (or fuzzy) clusters

critical sub-procedures are fully parallelized using a MapReduce framework. Their novel data partitioning scheme is based on computation cost estimation that aims to achieve load-balancing, even for a heavily skewed data.

Dimensionality Reduction techniques Although the algorithm complexity is related to the number of samples in the dataset, dimensionality is another important aspect that contributes to the algorithm complexity. Higher dimension adds more complexity to any clustering algorithm that results in the larger computation time. High-dimensional data also causes the problem of the curse of dimensionality that means the data becomes sparse in high-dimensional space which makes it difficult for final meaningful structures (clusters) in the data [56]. Sampling techniques reduce the number of instances in the dataset, but they do not offer a solution for high-dimensional datasets. One common solution to improve the clustering solution and to reduce the computational time for high-dimensional data is to perform dimensionality reduction before clustering [153]. The goal is to obtain a new dataset with a reduced set of features that preserves, up to a level, the original structure (geometry) of the data. *Feature selection* and *feature extraction* (or *transformation*) are

two popular approaches for dimensionality reduction [153].

The data may contain many features that are either redundant or irrelevant, and thus can be removed without loosing much information. *Feature selection* methods try to find a subset of original features which are relevant to model construction. Most feature selections methods for unsupervised learning can be divided on *filters* that use the characteristics of the features, such as the correlation among them to select features regardless of the model, and *wrappers* that explore the subset of features with a clustering algorithm to evaluate the quality of output partition using internal or external quality criteria.

Feature extraction or *feature transormation* involves transforming the original features into a new set of features (usually, of reduced size) in order to reduce the computation cost and increase the accuracy of a clustering/classifier algorithm. Feature extraction methods generate new features that are generally linear or non-linear combinations of the original features. Two popular methods that obtain a linear transformation of the data are *principal component analysis* and *random projection*. *Principal component analysis* (PCA) [154] uses an orthogonal transformation to obtain a set of observations of linearly uncorrelated variables that account for variance, called *principal components*. *Random projection* (RP) [155–157] is a linear transformation from high-dimensional space to a lower-dimensional space. RP is a simple and efficient way to reduce the data dimension by trading a controlled amount of distance error for faster processing times. Popular non-linear feature extraction methods include kernelized variants [158] of PCA, and manifold learning methods such as ISOMAP [159], *locality linear embedding* [160], and *multidimensional scaling* (MDS) [161]. Non-linear feature extraction methods can uncover more complex patterns in the data.

PCA and other non-linear methods use a well-defined criterion to optimize the projection in a lower dimension. However, these methods have high computational complexity, especially for large sample size datasets. Unlike these algorithms, random projection is a simple and computationally efficient technique which does not use any special criteria to find optimal lower dimensional projections. Lower computational complexity, (approximate) distance preservation in lower dimension subspaces and its easy implementation make it [156] an attractive choice for dimensionality reduction. Our contributions presented in Chapters 3 and 4 use random projection to deal with high-dimensional data. Below, we briefly explain random projection methods.

2.3.5.12 Random Projection

A *random projection* (RP) is a linear transformation from $\mathbb{R}^p$ to $\mathbb{R}^q$, represented by a matrix T. Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \subset \mathbb{R}^p$ be a set of N points in p dimensions, denoted as the "upspace". X can be mapped to a reduced dimension dataset $Y = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\} \subset \mathbb{R}^q, q \ll p$, denoted as the "downspace", by the linear transformation of X with T. Most RP methods are based on *Johnson-Lindenstrauss* (JL) Lemma [162] which states that if data points in a vector space are of sufficient high-dimension, then they may be projected into a suitable lower-dimensional space in a way that approximate distances among them are preserved. Several RP methods [155–157] have been proposed in the literature for different applications. A variant of the JL lemma proposed by Achlioptas in [156] is one of the most popular RP methods for clustering. The theorem proved by Achlioptas is as follows:

Theorem 2.1. *Let matrix $X \subset \mathbb{R}^{N \times p}$ be a dataset of N points and p attributes. Given $\varepsilon > 0$, and $\beta > 0$, for any integer q*

$$q \geq q_0 = \frac{(4+2\beta)log(N)}{\varepsilon^2/2 - \varepsilon^3/3}. \tag{2.8}$$

The parameter ε controls the accuracy in distance preservation, while β controls the probability that distance preservation to within $1 \pm \varepsilon$ is achieved. Let T be a $p \times q$ random matrix, in which each element $t_{i,j}$ is drawn from one of the following independently identically distributed distributions:

$$t_{i,j} = \begin{cases} +1 & \text{with probability 1/2} \\ -1 & \text{with probability 1/2} \end{cases} \tag{2.9}$$

$$t_{i,j} = \begin{cases} +\sqrt{3} & \text{with probability 1/6} \\ 0 & \text{with probability 2/3} \\ -\sqrt{3} & \text{with probability 1/6.} \end{cases} \tag{2.10}$$

Let $Y = \frac{1}{\sqrt{q}}XT$ be the projection matrix of the N points in $\mathbb{R}^q$. Let $f : \mathbb{R}^p \to \mathbb{R}^q$ map the i^{th} row of X to the i^{th} row of Y. Then for any $\mathbf{u}, \mathbf{v} \in X$ with probability at least $1 - N^{-\beta}$, we have

$$(1-\varepsilon)||\mathbf{u}-\mathbf{v}||^2 \leq |f(\mathbf{u}) - f(\mathbf{v})||^2 \leq (1+\varepsilon)||\mathbf{u}-\mathbf{v}||^2.$$

According to Theorem 2.1, if the reduced (downspace) dimension q is equal or bigger than the JL lower bound q_0, then pairwise Euclidean distance squares are preserved within a multiplicative factor of $1 \pm \varepsilon$, and it is said that Y has *JL certificate*. An older version of this projection operator is based on randomly choosing each element of T from a Gaussian distribution with zero mean and unit variance which carries a similar guarantee [156, 163]. The authors in [164] assert that the JL bound often holds for $q \ll q_0$. They called such projections "rogue random projections".

Since RP methods provide a probabilistic guarantee for distance preservation in lower-dimensional space, clustering results using a single random projection might be unstable. Therefore they are more often used in ensemble-based frameworks [71–73, 82] that aggregate the results of various lower dimensional clustering results to obtain a reliable final solution. Chapter 3 proposes a novel RP-based ensemble framework, CAFCM, for high-dimensional data clustering which employs FCM clustering on an ensemble of random projections. Also, the FensiVAT algorithm presented in Chapter 4 uses a new RP-based ensemble technique with MMRS sampling scheme for fast clustering of large-scale, high-dimensional datasets.

2.4 Cluster Validation

Once a clustering algorithm has processed the dataset and obtained a partition of the input data, the last question arises: Does the output partition represent the underlying structure of the input data? In other words, how well does the output partition fit the input data? This is a genuine question as an optimal clustering algorithm does not exist. Different clustering algorithms or even the same clustering algorithm with different configurations produce different partitions, and none of them have to be proved best in all situation. Therefore, the best strategy is to compute different partitions and choose the one which best fits the data. The process of evaluating clustering results is known as cluster validation. Cluster validity can also be used to estimate the number of clusters in the data. The usual approach is to apply a clustering algorithm with a different value of the

number of clusters, evaluate all partitions, and the chosen one that best fits the data.

One approach to measure the quality of the output partition is through the use of scalers measures, which are known as *cluster validity indices* (CVIs). There are more than hundreds CVIs [62, 165–169] are available in the literature, and most of them can be classified into two groups: *internal* and *external*. *Internal* CVIs use only data and/or algorithmic outputs, whereas, *external* CVIs require additional "outside" information such as a ground truth partition that labels subsets in the data.

Various CVIs are available in the literature for evaluating soft (fuzzy/probabilistic) [62, 165, 170] and hard (crisp) clustering partitions [167, 169]. Both soft and crisp partitions are mathematically defined (at the beginning of this chapter) using Eqs. (2.1) and (2.2). We call a CVI whose minimum value over a set of *candidate partitions* (CP) points to the "best one" as min-optimal [171], indicated by the symbol ($\downarrow$), and CVIs that indicate a choice of U in CP by their maximum value are max-optimal, indicated by ($\uparrow$).

2.4.1 Internal CVIs

Most internal CVIs consider two criteria to find an optimal partition: (i) *Compactness or Cohesion*- It measures how similar (or close) the data objects in the same cluster are, and (ii) Separation - it measures how well a cluster is separated from other clusters. Ideally, points of the same clusters should be as close to each other as possible, and clusters should be well separated from each other.

CVIs for soft clustering (fuzzy/probablistic) The internal CVIs available in the literature for fuzzy clustering can be classified mainly in two categories [62, 165]: the first category uses only the membership values U; and the second type involves both the U matrix and the dataset X itself. Some of the internal CVIs are discussed below:

Indices using both data (X) and membership values (U)

2.4.1.1 Xie Beni Index (↑)

Xie-Beni (XB) index [50], also called the compactness and separation validity function, was originally devised for use with FCM clustering. It is defined as:

$$XB(U) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{N}[u_{ij}^{m}||\mathbf{x}_j - \mathbf{v}_i||^2]}{N \ \min_{i\neq j}(||\mathbf{v}_i - \mathbf{v}_j||)} \tag{2.11}$$

2.4.1.2 Partition Index (↑)

The *partition index*, abbreviated as SC, is based on a fuzzy CVI proposed by Gath and Geva [172] which uses the concept of hypervolume and density, and it is defined as:

$$SC(U) = \sum_{i=1}^{k}\left(\frac{\sum_{j=1}^{N}(u_{ij})||x_j - V_i||^{m/2}}{\sum_{j=1}^{n}(u_{ij})^2}\right) \tag{2.12}$$

Small values of SC indicates the existence of compact clusters.

Indices using only membership values (U)

2.4.1.3 Partition Coefficient (↑)

Bezdek [173] proposed the *partition coefficient* (PC) index that indicates the average relative amount of membership sharing between pairs of fuzzy subsets in U. The PC index is max-optimal and, its value ranges in $[1/k, 1]$, where k is the number of clusters. PC is

$$PC(U) = \sum_{i=1}^{k}\sum_{j=1}^{N}(u_{ij})^2; \tag{2.13}$$

The PC index possess monotonic evolution tendency with k. Dependence on k makes interpretation of its relative values difficult to evaluate. To alleviate this, Roubens [174] defined a normalized version of PC,

$$PCR(U) = \frac{(kPC(U) - 1)}{(c-1)}. \tag{2.14}$$

The PCR value ranges in $[0, 1]$ for all values of k.

2.4.1.4 Normalized Partition Entropy (↓)

Bezdek [173] proposed another fuzzy CVI, *partition entropy* (PE), which is defined as:

$$PE(U) = -\frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{N} u_{ij} log(u_{ij}) \tag{2.15}$$

The PE index is a scalar measure of the amount of fuzziness in a given U. The PE is min-optimal and its value ranges in $[0, ln_a k]$. Similar to PC, PE index also shows monotonicity with k. Bezdek [175] defined the normalization of PE as:

$$PEB(U) = PE(U)/ln_a k \tag{2.16}$$

CVIs for crisp clustering

2.4.1.5 Davies-Bouldin Index (↓)

Davies-Bouldin Index (DBI) [49] is the ratio of the sum of within cluster scatter or cohesion (the distance from the points in a cluster to its centroid) to the between cluster separation (distance between cluster centroids). Hence, a lower value of DBI is desirable.

$$DBI(U) = \frac{1}{k}\sum_{i=1}^{k}\max_{i\neq j}\frac{S(C_i)+S(C_j)}{dist(C_i,C_j)}, \quad \text{where} \quad S(C_i) = \frac{1}{|C_i|}\sum_{\mathbf{x}_i\in C_i} dist(\mathbf{x}_i,\mathbf{v}_i) \tag{2.17}$$

2.4.1.6 Silhouette Index (↑)

The silhouette [51] index is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The cohesion is measured based on the distance between all the points in the same cluster and separation is based on the nearest neighbour distance. The Silhouette value ranges in $[-1, 1]$

$$Sil(U) = \frac{1}{N}\sum_{i=1}^{k}\sum_{\mathbf{x}_i\in C_i}\frac{b(\mathbf{x}_i,C_i)-a(\mathbf{x}_i,C_i)}{max\{a(\mathbf{x}_i,C_i),b(\mathbf{x}_i,C_i)\}}, \tag{2.18}$$

where

$$a(\mathbf{x}_i, C_i) = \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} dist(\mathbf{x}_i, \mathbf{x}_j) \; ; \quad b(\mathbf{x}_i, C_i) = \min_{j \neq i} \left\{ \frac{1}{|C_j|} \sum_{\mathbf{x}_j \in C_j} dist(\mathbf{x}_i, \mathbf{x}_j) \right\} \tag{2.19}$$

2.4.1.7 Dunn's Index (↑)

Dunn's index (DI) [48] is a metric of how well a set of clusters represent *compact separated* (CS) clusters. It is defined as:

$$DI(U) = \frac{min_{1 \leq i,j \leq k \; i \neq j} \; \text{dist}(C_i, C_j)}{max_{1 \leq l \leq k} \; \text{diam}(C_l)}, \tag{2.20}$$

where C_i is the i-th cluster, $\text{dist}(C_i, C_j)$ is the distance between two clusters, and $\text{diam}(C_l)$ is the cluster diameter. The distance between two clusters can be *single linkage*, *complete linkage*, *average linkage*, or *centroid-based distance*. The diameter of a cluster can be its maximum or average diameter. Bezdek and Pal [169] proposed 18 variants of DI, which they called *Generalized Dunn's indices* (GDIs), based on various combinations of distance function in the numerator and denominator of Eq. (2.20).

The computation of DI is very computationally expensive due to its quadratic growth in the number of samples. To solve this problem, this thesis presents six approximation methods to compute DI for big data, in Chapter 5. These methods may also be useful for computation of other CVIs that require both X and U, such as DBI, XB, Silhouette index and the GDIs. Chapter 5 discusses DI and GDIs in more detail, and presents their approximation methods for big data.

2.4.2 External CVIs

External CVIs validate the clustering model against external information (ground truth subsets). The aim of these indices is to evaluate the extent to which a true (ground truth) partition can be discovered using clustering. Many well-known external CVIs are derived from the values of contingency table [176, 177]. Given the partitions U and V, the contingency table is defined as:

The partition U and V need not possess the same number of clusters. The classical approach to compare U and V begins by considering the four possible combinations for pairs of objects from

Table 2.2: The Contingency Table A to compare partition U and V

		Partition V v_j = row j of V	
	Class	v_1 v_2 ... v_r	Sums
Partition U u_i = row i of U	u_1 u_2 u_3 . u_k	$A = \begin{bmatrix} N_{11} & N_{12} & \dots & N_{1r} \\ N_{21} & N_{22} & \dots & N_{2r} \\ N_{31} & N_{32} & \dots & N_{3r} \\ \dots & \dots & \dots & \dots \\ N_{k1} & N_{k2} & \dots & N_{kr} \end{bmatrix} = UV^T$	$N_{1\bullet}$ $N_{2\bullet}$ $N_{3\bullet}$. $N_{k\bullet}$
	Sums	$N_{\bullet 1}$ $N_{\bullet 2}$ ... $N_{\bullet r}$	$N_{\bullet\bullet} = N$

the set O in clusters of U and V. These four quantities are defined as:

- f_{00}- the number of pairs of data points that are in the different subset in V, and are in different subset in U
- f_{01}- the number of pairs of data points that are in the different subset in V, and are in same subset in U
- f_{10}- the number of pairs of data points that are in the same subset in V, and are in different subset in U
- f_{11}- the number of pairs of data points that are in the same subset in V, and are in same subset in U

Below, we discuss some clustering oriented external validity indices.

2.4.2.1 Partition Accuracy (↑)

Partition accuracy (PA) of a clustering algorithm is the ratio of the number of samples with matching ground truth and algorithmic labels to the total number of samples in the dataset. The assignment of class labels to the clusters is done on the basis of the majority value of the class attribute within each cluster. The value of PA ranges from 0 to 1, and a higher value implies a better match to the ground truth partition.

2.4.2.2 Purity (↑)

Purity [54] is a measure of the extent to which clusters contain a single class. It is defined as:

$$Purity = \sum_{i=1}^{k} \frac{N_{i\bullet}}{N} \max_{j} \frac{N_{ij}}{N_{i\bullet}} \tag{2.21}$$

A high purity means that the cluster is pure (i.e. contains objects from the same class). Its values range in $[0,1]$.

2.4.2.3 Rand Index (↑)

Rand index (RI) [52] measures the similarity between two clustering partitions. It takes into account the numbers of point pairs that are in the same and different clusters, and is defined as,

$$RI(U,V) = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}} \tag{2.22}$$

Intuitively, the sums $f_{00} + f_{11}$ and $f_{10} + f_{01}$ are interpreted, respectively, as (the total number of) *agreements* and *disagreements* between U and V. RI values lie between 0 and 1, and a value close to 1 indicates high agreement between partition U and V.

2.4.2.4 Adjusted Rand Index (↑)

A problem with RI is that its expected value between two random partitions is not a constant, and on certain datasets it can achieve a high score otherwise. This problem is corrected by the *adjusted Rand index* (ARI) [53] that assumes the generalized hyper-geometric distribution as the model of randomness. It is defined as:

$$ARI = \frac{\text{Rand Index} - \text{Expected index}}{\text{Maximum index} - \text{Expected index}}, \tag{2.23}$$

The ARI has the maximum value 1 for perfect partition, and 0 when the index equals its expected value. The adjusted RI takes following form [53]:

$$ARI = \frac{a - \frac{(a+c)(a+b)}{(a+b+c+d)}}{\frac{(a+c)+(a+b)}{2} - \frac{(a+c)(a+b)}{(a+b+c+d)}}, \tag{2.24}$$

where the values of a, b, c, and d [53] are derived from the contingency table (Table 2.2), and are given as:

$$a = \frac{1}{2}\sum_{i=1}^{r}\sum_{j=1}^{k} N_{ij}(N_{ij}-1) \tag{2.25a}$$

$$b = \frac{1}{2}\left(\sum_{j=1}^{k} N_{\bullet j}^2 - \sum_{i=1}^{r}\sum_{j=1}^{k} N_{ij}^2\right) \tag{2.25b}$$

$$c = \frac{1}{2}\left(\sum_{i=1}^{r} N_{i\bullet}^2 - \sum_{i=1}^{r}\sum_{j=1}^{k} N_{ij}^2\right) \tag{2.25c}$$

$$d = \frac{1}{2}\left(N^2 + \sum_{i=1}^{r}\sum_{j=1}^{k} N_{ij}^2 - \left(\sum_{i=1}^{r} N_{i\bullet}^2 + \sum_{j=1}^{k} N_{\bullet j}^2\right)\right) \tag{2.25d}$$

2.4.2.5 Normalized Mutual Information (↑)

The *Normalized mutual information* (NMI) [130] measures the information shared between two clustering partitions. It is defined as:

$$NMI(U,V) = MI(U,V)/\sqrt{H(U)H(V)}, \tag{2.26}$$

where MI is the *Mutual information* (MI) between U and V, and $H(U)$ is the entropy associated with partition U which are defined as:

$$MI = \sum_{i=1}^{c}\sum_{j=1}^{r}(N_{ij}/N)log(\frac{N_{ij}/N}{N_{i\bullet}N_{j\bullet}/N^2}) \tag{2.27}$$

$$H(U) = -\sum_{i=1}^{c}(N_{i\bullet}/N)log(N_{i\bullet}/N) \tag{2.28}$$

If two classes are completely independent, then NMI is 0. If both are identical, then their NMI is 1.

The soft RI [178], soft ARI [177, 178], and soft NMI [170] can be computed using entries (in their formulas) from generalized contingency matrix $A^* = \phi UV^T$ (Table 2.2), where $\phi = \frac{N}{\sum_{i=1}^{c} N_{i\bullet}}$.

2.4.3 Cluster validity for big data

There has been a considerable amount of work done to address clustering tendency assessment and clustering problem for big data. However, very little work is available on cluster validity for big data. Early validity indices considered only membership values due to the advantage of being easy to compute. These CVIs can handle big data as they use only the output partition (U), which is not usually big as compared to the input (big) dataset X. However, these indices may have some drawbacks [165] such as their monotonous dependency on the number of clusters and the lack of direct connection to the geometry of the data. Now, it is widely accepted that a better definition of a validity index always considers both partition matrix U and the dataset itself. The implementation of most of such indices is very computationally expensive, especially when the number of clusters and number of objects in the dataset grows very large [62].

There is very limited work available in the literature for cluster validity index for big data. Tlili *et al.* [179] proposed a fuzzy version of Davies-Bouldin index [49] for big data clustering. Since DBI uses both the output partition and the data itself, it still has the higher computational complexity for big data. Another work [180] introduced Spark implementation of DI and Silhouette index to deal with big data. Chapter 5 of the thesis proposes six novel approximation methods to compute DI (and GDIs) for big data.

2.5 Big data clustering applications

Cluster analysis techniques have been used in many applications such as social network analysis [16, 23, 181], bioinformatics [18], market analysis [21, 22], gene expression analysis [19], and image processing [20]. Big data image segmentation is an image analysis technique, which has emerged especially in the medical area. Due to the complexity of image segmentation and with limited prior knowledge, unsupervised methods are a better choice for image segmentation. Partitioning-based clustering methods are popular for big data image segmentation due to their easy implementation. The scalable fuzzy c-means clustering algorithms were applied to large-scale *magnetic resonance images* (MRI) of the brain for image segmentation [1]. A distributed c-means algorithm was proposed for big data image segmentation and was applied to MRIs in [182]. Fuzzy clustering methods have also been applied to high-dimensional DNA mi-

croarray data for gene expression analysis [71, 183]. Clustering also finds its utility to organize and retrieve databases in an efficient way [15, 17].

2.5.1 Big data clustering for smart city applications

Clustering has also been used in several smart city applications to gain valuable insights from raw data obtained through various sensing devices and crowdsourcing. For example, the work in [31–33] applied clustering to time-series energy data from smart meters to identify energy usage profiles of residential, commercial, and industrial consumers, and for load-forecasting. There are also several applications of clustering in other smart city contexts, such as in smart parking [26–28], smart environment [24, 25], smart health care [34–36], smart agriculture [37, 38], and smart waste management [39, 40], and intelligent transportation systems [29, 30].

This thesis also presents a big data clustering application for smart city generated big data, viz., prediction of vehicle trajectories using a clustering-based approach on large-scale GPS data. Below, we first discuss trajectory prediction and its important for smart city applications, and then we provide a brief review of the existing trajectory prediction approaches and their limitations.

Trajectory prediction The widespread use of *global positioning system* (GPS) navigation systems and wireless communication technology enabled vehicles have resulted in huge volumes of spatio-temporal data, especially in the form of trajectories. These data often contain a great deal of information [184], which give rise to many *location-based services* (LBSs) and applications such as vehicle navigation, traffic management, and location-based recommendations. One key operation in such applications is the route prediction of moving objects. Vehicle route prediction allows certain services to improve their quality, e.g., if the route of vehicles is known in advance, *intelligent transportation systems* (ITSs) can provide route-specific traffic information to drivers such as forecasting traffic conditions and routing the driver to avoid traffic jams. Route prediction also enables location-based advertising, which can advertise certain products/services and special offers to the target commuters most likely to pass through business outlets and stores based on their travel trajectory.

The first step in the trajectory prediction task is the trajectory representation. Won *et al.* [185] represented a trajectory as a list of segments, each of which has its identifier and length. How-

ever, the method to divide the road segment is not explicitly described and is driven by intuition. Guo *et al.* [186] used topological information from graph-based structures to represent trajectory data. A road network is represented as an undirected graph, where each node represents a junction/intersection and each edge represents a road (path) segment of the road network. The trajectory of a moving object appears as a sequence of symbols in [88, 187–189], where each symbol represents a road section. For vehicles moving along the road segments, their trajectory can be best represented as a 1-dimensional array of nodes or edges of the road network [88, 187–190]. The (Latitude, Longitude) coordinate pair obtained from the GPS sensor can be associated with a corresponding road network edge using one of the many popular map-matching techniques [191, 192].

Several studies have been carried out on trajectory prediction, particularly after Song *et al.* [193] demonstrated a 93% potential for predictability in user mobility, which supplied the theoretical basis for location prediction methods. These methods mainly focus on two kinds of prediction models. The first type is the short-term trajectory prediction model, which aims to predict the next-location or a few locations in the near future. These models usually rely on current location and one or two previous locations of an object to predict its next location. The second type is the long-term trajectory prediction model which focuses on location prediction at a more distant future time or on complete route prediction. These models generally rely on an available partial trajectory of a moving object to predict the complete trajectory.

Trajectory prediction approaches Existing TP methods are hybrid in nature and can be broadly classified into three categories: (i) Rule-based learning based approaches (ii) Markov model-based approaches (iii) Clustering-based approaches.

2.5.1.1 Rule-based learning based approaches:

Morzy [194] implemented a modified version of the PrefixSpan algorithm to extract association rules from a moving object database. Then, the trajectory of a moving object was matched with the database of movement rules using matching functions to select the best association rule and then used it for prediction. The matching function in [194] is based on the notion of support and confidence and does not consider any notion of spatial and/or temporal distance. Jeung *et*

al. [195] proposed a hybrid prediction approach, which combines association rules in the form of trajectory patterns with the motion functions of an object's recent movements, to estimate future locations. Given an object's recent movement and predictive queries, the best association rule is chosen for prediction. The query processing approaches presented in [195] can only support near and distant-time predictive queries, unsuitable for long-term trajectory prediction. Moreover, with the huge number of trajectories, the number of association rules is also huge, which makes association-rule based algorithms impractical for large-scale mobility data.

2.5.1.2 Markov model-based approaches:

A *Markov model* (MM) is a stochastic model used to model randomly changing systems. It assumes that future states depend only on the current state, not on the past events before it (that is, it assumes the Markov property). MMs [196–198] have been widely used to mine frequent patterns for route prediction problems. Ishikawa *et al.* [196] proposed a model to extract mobility statistics, called the Markov transition probability, which is based on a cell-based organization of target space and a Markov chain model, and employed R-tree spatial indices to compute Markov transition probabilities. Simmon *et al.* [197] presented a *Hidden Markov Model* (HMM) based probabilistic approach to predict a driver's intended route and destination through observations of the driver's habits. Asahara *et al.* [87] suggested that standard MM and HMM are not generic enough to encompass all types of movement behaviour. They proposed a variant of Markov model, called the *mixed Markov-chain model* (MMM), as an intermediate model between individual and generic models, for pedestrian movement prediction. This approach clusters individuals into groups based on similar movement behaviour, and generates a Markov model for each group. The next location is predicted by first identifying the group a particular individual belongs to and then inferring next location using corresponding Markov model. Gambs *et al.* [198] extended a previously proposed mobility model, named *w-Mobility Markov Chain* (w-MMC), to incorporate the w previous visited locations. They showed that prediction accuracy increases with w, but increasing w beyond two ($w > 2$) does not compensate for the significant overhead in terms of computation and space for learning and storing the mobility model. They only considered the sequence of the significant locations, instead of all locations, to build higher order MM.

2.5.1.3 Clustering based approaches:

Some researchers have proposed trajectory clustering based route prediction methods [199, 200], which partition the trajectories into several clusters representing different motion patterns based on the trajectory similarity. Various clustering approaches [201] using different methods and distance measures between trajectories have been proposed in the literature. Road network constrained trajectory clustering approaches can be classified into two broad categories. The first type uses the traditional clustering approaches such *k*-means and DBSCAN with specially designed distance measures [185, 188, 202, 203] for trajectories. For example, Ashbrook *et al.* [199] presented a system that first extracts the significant locations, called *Point of Interests* (POIs), using *k*-means clustering on GPS data, and then calculates the probability of transitions between these significant location using various orders of Markov models, to find next most likely significant location based on those recently visited.

The second category of algorithms [88, 190] cluster road segment vehicle frequencies based on density and flow. Flow and density based trajectory clustering schemes such as NETSCAN [88] and *network aware trajectory* (NEAT) [190] first summarize the trajectory data into an edge density or edge transition matrix based on the frequency of the trajectories passing through a road segment (density) or two consecutive road segments (flow). Clustering is then performed on road segments based on their traffic flow to create a set of consecutive road segments having continuity of traffic density and flow.

Traditional clustering-based TP algorithms [199, 200, 204] are not scalable to large numbers of trajectories in a city environment as computation of the distance matrix is time intensive. Most of them require the number of clusters to be known in advance, but in practice, it is often unknown, making it difficult for the user to choose the optimal number of clusters for location prediction. Furthermore, the clusters are determined by fixed rules. Although some of the road network based clustering approaches [88, 190] are scalable, they produce loose clusters which span a large space of the road network and may not represent actual traffic flow. Hence, they are not suitable for vehicle trajectory analysis.

Most TP methods demonstrated in the literature [87, 194, 195, 205] use synthetic or small to medium size real trajectory datasets. Most of them cannot handle big trajectory datasets. There have been several attempts to demonstrate trajectory prediction on real data having a large number

of samples. For example, [206] uses a real dataset consisting of 4.9 million trajectories (790 million GPS points) as a population, but only small subsets having a maximum $30,000$ trajectories are used in their experiments.

As a real-world application of big data clustering, Chapter 7 of this thesis presents a novel, scalable, hybrid framework for vehicle trajectory prediction, which can handle a large number of trajectories in a dense road network, typically for major cities around the world. The proposed framework is based on a scalable clustering approach, Traj-clusiVAT, a modified version of clusiVAT implemented for trajectory prediction. Traj-clusiVAT can also determine the number of clusters, which represent different movement behaviours in input trajectory data. After clustering trajectories, Markov chain models are constructed from the trajectories in each cluster. These models quantify the movement patterns within clusters, and subsequently, are used for trajectory prediction. The proposed TP framework used two large, city-scale taxi trajectory datasets: T-Drive dataset consisting of $43,405$ GPS trajectories from $10,357$ taxis in Beijing over a period of one week and Singapore taxi dataset consisting of 3.28 million passenger trajectories from $15,061$ taxis over a period of one month. This was the first time trajectory prediction task had been performed on such a large number of real-world road network trajectories.

2.6 Summary

In this chapter, we have reviewed major techniques to solve each of the three problems of cluster analysis, viz., clustering tendency assessment, clustering, and cluster validity. We reviewed and compared popular methods for cluster analysis for big data, including streaming data. Several methods in the area of big data clustering applications for trajectory prediction are also reviewed.

We discussed that most cluster analysis algorithms encounter serious problems related to computational and space complexities and/or cluster quality for large-scale, high-dimensional datasets. To address these challenges, we developed a suite of novel algorithms to solve each of the three problems of cluster analysis for knowledge discovery from large-scale, high-dimensional data, by leveraging intelligent sampling and dimensionality reduction strategies. These contributions are presented in Chapters 3-5. In this chapter, we also explored that existing cluster assessment techniques are not suitable for high-velocity, massive streaming data. To address this problem, we

presented an incremental algorithm for online assessment of evolving cluster structures in high-velocity data streams, in Chapter 6. Finally, as a real application of big data clustering, we present a scalable framework for vehicle trajectory prediction in Chapter 7.

This page intentionally left blank.

Chapter 3

Clustering High-Dimensional Data using Cumulative Aggregation on Random Projections

This chapter proposes a new, simple and computationally efficient framework called CAFCM for high-dimensional data clustering, which employs FCM clustering an ensemble of random projections. CAFCM outperforms the three other state-of-the-art approaches, EFCM [71], RPFCM-A [72], and RPFCM-B [73], in terms of accuracy, stability, space, and time complexity, based on our numerical comparisons on two synthetic and six real large, high-dimensional datasets.

3.1 Introduction

Many applications such as biomedical imaging, physiological monitoring, sequencing [207], and time-series matching produce large amounts of high-dimensional data [55]. High-dimensional feature vector data, i.e., data described by a large number of attributes, poses two challenges for clustering. First, the so-called “curse of dimensionality", which is caused by the lack of a sufficient number of samples in most high-dimensional data, makes it difficult to find statistically meaningful structures in the data [56]. Second, noisy and irrelevant attributes in the data can worsen the performance of a clustering algorithm. One possible solution to improve the utility of clustering algorithms for high-dimensional data is to perform dimensionality reduction [208].

Popular algorithms for dimensionality reduction, such as *Principal Component Analysis* (PCA) and *Singular Value Decomposition* (SVD), use well-defined criteria to optimize the projection in lower dimensional space. Unlike these algorithms, random projection [155–157] is a relatively simple, computationally efficient linear transformation method which does not use any special

criteria to find "optimal" lower dimensional projections. Two key properties, namely low computational complexity and (approximate) distance preservation in lower dimension subspaces, make random projection [156] an attractive choice for dimensionality reduction.

Over the past few years, ensemble clustering [130, 209–214] has drawn significant attention in addressing the clustering problem. Random projection based ensemble frameworks [71–73, 82] have been proposed for high-dimensional clustering using fuzzy or probabilistic clustering algorithms. These approaches use random projection to generate multiple subsets into a lower dimension from the original dataset, and then some method of integration is used across the soft clustering results obtained on all projected datasets. Among these random projection based fuzzy clustering approaches, the most recent approaches [72, 73] require less memory and run faster than earlier approaches [71, 82]. However, the ensemble algorithms developed in [72, 73] still require very large amounts of space for storing a big affinity matrix; moreover, they take a lot of time to cluster the affinity matrix.

This chapter proposes a new, simple, and efficient random projection based ensemble framework using a cumulative agreement scheme to aggregate multiple fuzzy membership matrices based on their quality. *Cluster Validity Indices* (CVIs) are used to determine the quality of consensus partitions. This framework eliminates the need of a final time-consuming clustering step such as the ones reported in [71–73, 82] to obtain output partitions. The ensemble approach in our framework combines fuzzy partitions in a sequential manner, thus avoiding the complexity required by simultaneous aggregation of the suite of fuzzy partitions produced by clustering many random projections of the high-dimensional data. Our method, which we called *Cumulative Agreement* FCM (CAFCM), scales linearly in the number of data points and the number of repetitions, making our random projection based ensemble approach feasible for large, high-dimensional datasets.

3.2 Related Work

This section reviews existing random projection based cluster ensemble methods for high-dimensional data clustering and agreement based combination schemes.

3.2.1 Random Projection Based Ensemble Approaches

Several ensemble approaches have been proposed for high-dimensional data clustering which are based on RP and FCM. The main idea of the existing approaches is as follows; First, multiple downspace datasets $\{Y_r\}_{r=1}^{Q}$ are generated in a fixed lower dimension $\mathbb{R}^q$ using RP, where Q is the number of RPs. Then, FCM clustering is performed on each downspace copy to obtain Q fuzzy partitions, e.g., $U_r = \text{FCM}\,(Y_r)$, where $U_r \in M_{fcN}$. These output partitions $\{U_r\}_{r=1}^{Q}$ are aggregated using an ensemble scheme. The final output partition is typically obtained by performing soft clustering on the rows of an aggregated matrix.

Apparently, the first cluster ensemble approach that used random projection was proposed in [82], in which GMM/EM clustering was used to obtain probabilistic partitions $P \in M_{fcN}$, where $p(c|i,\theta)$ is the probability of point i being in cluster c under a model θ. Subsequently, a similarity matrix M_i was computed between two joint probability distributions for each downspace dataset. The final similarity matrix M was obtained by averaging the M_is, and then the final clustering output was obtained by applying a hierarchical clustering algorithm, called *complete linkage* (CL), on the aggregated similarity matrix M.

A similar approach using FCM for fuzzy clustering (EFCM) was used in [71] to find the significant genes in DNA microarray data. Random projection was used to reduce the data dimensionality. Then, the FCM clustering algorithm was employed on each downspace dataset to generate membership matrices $U_r \in M_{fcN}$. Then for each r, a similarity matrix M_r was computed as $M_r = U_r^T U_r \subset \mathbb{R}^{N\times N}$. Then, an aggregated similarity matrix (M) was calculated by averaging the Q M_rs across multiple projection runs. The distance matrix $D = 1 - M$ was computed, and then FCM was performed on the rows of $D \subset \mathbb{R}^{N\times N}$ to obtain a final membership matrix.

Both of the above approaches have space complexity $O(N^2)$ for storing the similarity matrix (M). There is a time complexity of $O(N^2 log(N))$ in applying complete linkage (GMM/EM-based approach) and $O(dlNc^2)$ in applying FCM (the EFCM approach) on $D \subset \mathbb{R}^{N\times N}$, where N is number of data points, d is the dimensions of the matrix on which clustering is applied (for EFCM approach, $d = N$), c is the number of clusters, and l is the number of iterations used by FCM. There is additional time complexity of $O(cQN^2)$ in the EFCM approach due to computing the product of the Q partition matrices and their transposes. Therefore, both of these algorithms are limited to applications for which the number of objects N is small (e.g., some thousands of samples), and

the original dimension p of the upspace data is large (e.g., more than tens of thousands). As N increases, the EFCM approach becomes intractable for big data.

To address the limitations of these two approaches for big data clustering, Popescu *et al.* [72] proposed a new method, RPFCM-A, that began with the FCM clustering of random projections of the data. The resultant membership matrices $\{U_r\}_{r=1}^{Q}$ were concatenated as $U_{con} = [U_1^T|U_2^T|....|U_Q^T]$, and the final membership partition was obtained by applying FCM to the rows of the aggregated matrix $U_{con} \subset \mathbb{R}^{N\times cQ}$. Concatenating Q partitions of $N \times c$ dimension by stacking them along the element dimension results in an $N \times cQ$ matrix which is significantly smaller than M_r (used in EFCM). This approach eliminates the time complexity spent computing products of the membership matrices and their transposes. Thus, it seems more suitable than the EFCM based approach. However, it still requires the multiplication of the concatenated matrix with its transpose when a crisp output partition is desired. Moreover, this scheme has time complexity of $O(dlNc^2)$ when applying FCM to the concatenated matrix $U_{con} \subset \mathbb{R}^{N\times cQ}$, where $d = cQ$. If the number of clusters c in the data and the number of downspace datasets Q are such that $cQ > p$; it means the dimension of the agreement matrix becomes higher than the original dimension of the dataset, which makes this approach unsuitable for high-dimensional data clustering.

Mao *et al.* [73] proposed a modified approach, RPFCM-B, based on spectral graph partitioning. Instead of considering the full agreement matrix U_{con}, they performed the clustering on the first c left singular vectors of $\hat{U}_{con}$, where $\hat{U}_{con} = SVD(U_{con}) \subset \mathbb{R}^{N\times c}$, which reduces the computational time as compared to RPFCM-A approach. However, there is space complexity of $O(cNQ)$, and computational complexity of $O(N(cQ)^2)$ for SVD and $O(dlNc^2)$ for the FCM clustering, where $d = c$.

3.2.2 Agreement Based Combination Schemes

Among existing ensemble approaches, agreement based merging algorithms are popular due to their simplicity and computational efficiency. The idea of the agreement based combination scheme for fuzzy clustering was first introduced by Dimitriadou *et al.* [213], which is based on minimizing the average squared distance between ensemble membership partitions and an optimal output partition. This algorithm computes an approximate solution sequentially, in which, the best cluster label permutation is obtained for each ensemble partition with respect to a reference parti-

tion, followed by updating the reference partition through averaging. However, the determination of the best cluster label for each cluster in a partition for large values of c is a time-consuming task due to the computation of squared distances between partitions across each possible permutation of cluster labels. The labelling correspondence problem is solved in [212] using a maximum-likelihood estimate found with the Hungarian method [215], and then plurality voting is applied to obtain an optimal partition. The Hungarian algorithm can be costly because it is $O(c^3)$. The most recent work on consensus clustering employs a voting based mechanism [214], where the cluster label assignment problem is addressed using a contingency matrix which requires less computation time than that required by previous methods. The study in [214] was limited to crisp partitions. This scheme may not enjoy the same performance for soft partitions, which are obtained from projected datasets using random projection. This is because random projection produces highly unstable and radically different outputs [82, 163].

Although a fair amount of work has been done on agreement based aggregation schemes, only a few schemes are applicable to soft clustering. In our work, the use of FCM clustering on the aggregated matrix to get a final output partition is eliminated using an agreement based aggregation scheme which is computationally efficient and easy to implement. Fig. 3.1 compares the three FCM based schemes in [71], [72] and [73] to our proposed CAFCM method.

In the next section, we discuss our agreement based scheme for aggregating the fuzzy partitions $\{U_r\}_{r=1}^{Q}$, obtained from FCM clustering on Q randomly projected datasets.

3.3 Agreement based Aggregation Model

The objective of an aggregation model is to find a partition U_f, which represents a set of Q fuzzy partitions $\{U_r\}_{r=1}^{Q}$, the representation being optimal in some well-defined sense. We assume that U_f and the U_r are all the same size ($c \times N$). Let $\mathbf{u}_i^{(r)}$ and $\mathbf{u}_i^{(f)}$ be the label vectors of data point $\mathbf{x}_i$ for the partitions U_r and U_f, respectively. That is, $\mathbf{u}_i^{(r)}$ is the i-th column of U_r, and similarly for $\mathbf{u}\_i^{(f)}$. The average dissimilarity function $h(U_r, U_f)$ is chosen as an optimality criteria, and can be expressed as the average squared distance between the Q columns of U_r and U_f, as [213]

$$h(U_r, U_f) = \frac{1}{N} \sum_{i=1}^{N} ||(\mathbf{u}_i^{(r)} - \mathbf{u}_i^{(f)})||^2. \tag{3.1}$$

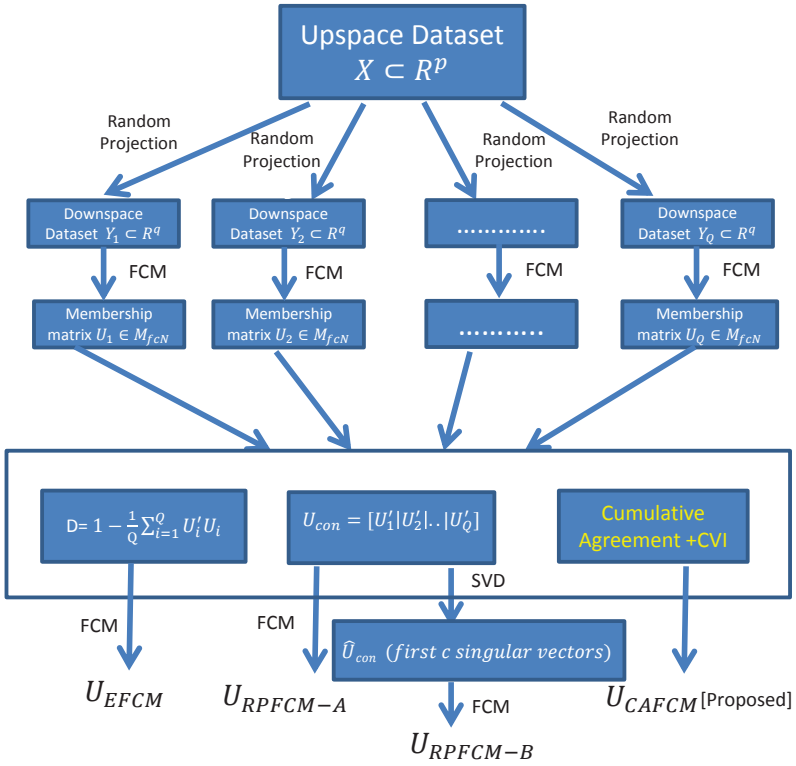


Figure 3.1: Four methods (including CAFCM (proposed)) of ensemble FCM clustering using random projection

The computation in equation (3.1) measures the similarity between U_r and (the unknown solution) U_f on the assumption that the c clusters in U_r and U_f are "aligned", i.e., the rows of U_r and U_f represent the clusters in the same order. This is the so-called "registration problem" in clustering, and care must be taken to ensure that all of the partitions being aggregated are aligned in this sense. This problem is exacerbated when the partitions are fuzzy. We want to relabel the Q U_r's so that they are aligned. This ensures that they will be aligned with the unknown U_f.

One way to approach this problem is to let $\Pi_b(U_r)$ represent the mapping of partition U_r to an optimally relabelled partition $U_{r,b}$ with respect to some base (or core) partition U_b. Then, an optimal partition can be obtained as the solution to [213],

$$U_f = \underset{U_b \in M_{fcN}}{\arg\min} \left(\frac{1}{Q} \sum_{r=1}^{Q} h(\Pi_b(U_r), U_b) \right). \tag{3.2}$$

The solution of this minimization problem in [213] gives $\mathbf{u}_i^{(f)}$ as the arithmetic mean of $\mathbf{u}_i^{(r)}$ over all partitions. In order to obtain the best cluster label permutation for each ensemble partition, the squared distance (minimization) between the ensemble and base partitions was chosen as mapping $\Pi_b(U_r)$. A contingency weight matrix based mapping scheme was proposed in [214] as a solution of (3.2). These solutions are not effective in combining multiple fuzzy partitions which are obtained using random projections. Our experiments with this method did not show very promising results. So, we turned to another approach, which effectively combines fuzzy partitions, obtained using RPs, based on their quality, as measured by cluster validity indices.

The concept behind agreement based ensemble approach is that pairs of points that stick together (appear in the same cluster) in most or all of the individual partitions should also stick together in the final ensemble partition. Suppose the number of clusters c_r for individual partitions U_r is randomly selected within some range $\{c_{min}, c_{max}\}$. The intuition underlying our approach is that the pairs of points that are members of a cluster for higher values of c should be considered to be more strongly associated to each other than pairs of points which are together in a cluster at a smaller value of c.

The Q partitions obtained by applying FCM clustering to Q random projections will have different information content (quality). The best quality partition, which has maximum information content about the cluster labels distribution, is chosen as the base partition, U_b, in the first step of the aggregation. Assuming that no prior knowledge for the selection of the base partition and the "true" number of clusters is available, an internal CVI is used to choose the base partition (discussed in the next section).

The remaining $Q-1$ partitions are ranked in decreasing order of quality based on their relationship to the base partition, and are combined sequentially based on their rank. The objective of this scheme is to secure the strongest agreement between the highest ranked partitions in the queue with the base partition. In this way, low-quality partitions will have minimal effect on the quality of the overall output partition. Minor variations in ranking are not expected to impact the performance of this scheme, because using an ordered sequence based on decreasing quality effectively integrates the good and bad fuzzy partitions, and decreases the effects of bad partitions on the overall output. If the base partition is of poor quality or there is major variation in ranking (for example, a few poor-quality partitions are in the top five partitions in the CVI queue), then

the performance may deteriorate. At the other extreme, if all Q partitions are of roughly the same quality, then the selection of the base partition and ranking of the remaining partitions will not have a significant effect on the output partition.

In the next section, we discuss the use of CVIs to achieve the best performance for CAFCM.

3.4 Quality of Consensus Partitions

The projected datasets can be drastically different from each other due to the random mapping from upspace to downspace. Consequently, clustering on these different downspace datasets with any algorithm may result in output partitions of different quality. In our work, a CVI is used to determine the quality of partitions. A CVI can be used to identify the "best" member amongst a set of multiple partitions (where best means, with respect to the CVI in use). A detailed analysis and discussion on various internal and external CVIs are provided in Chapter 2 (Section 2.4).

The quality of the output partition U_f constructed by CAFCM depends on the quality of the base partition U_b, which is chosen in the initialization phase. The fuzzy partition from the set $\{U_r\}_{r=1}^{Q}$, which best preserves the structure of the ground truth partition of labeled data will be taken as the base partition (we use internal CVI to find the best partition (explained in subsequent paragraphs)). The intuition behind using the best member from the set of ensemble partitions as the base partition is that the output partition U_f should contain the maximum amount of information about structure in the data that is present in the best quality partition amongst all ensemble partitions. Most importantly, this will eventually lead us to a method for identifying U_b for the unlabeled data case.

The quality of individual fuzzy partitions compared to a ground truth (labeled data) partition can be determined using a soft external CVI. Let the quality of any partition U_r with respect to the ground truth partition U_{gt}, using an external soft CVI $\mathscr{V}_{ext_s}$, be denoted as $\mathscr{V}_{ext_s}(U_r|U_{gt})$, where subsubscript "$s$" means soft. Based on the optimality of $\mathscr{V}_{ext_s}(U_r|U_{gt})$, the Q ensemble partitions can be ranked in descending order of quality such that

$$\mathscr{V}_{ext_s}(U_{(1)}|U_{gt}) \geq \mathscr{V}_{ext_s}(U_{(2)}|U_{gt}) \geq ... \geq \mathscr{V}_{ext_s}(U_{(Q)}|U_{gt}), \tag{3.3}$$

where parenthetical subscripts indicate the permutation of the original indices that results in the

ordering shown in (3.3), and we assume without loss of generality that the CVI is max-optimal (best is maximum). This gives a set of sorted partitions $\mathbb{U}_{sorted}^{(ext_s)}$ based on their quality with respect to the external CVI $\mathscr{V}_{ext_s}$. In real-world applications, the data is unlabeled so the ground truth information, which is required to evaluate partition quality based on (3.3), is not available. In this case, a question that must be answered is: can internal CVIs ($\mathscr{V}_{int_s}$) be used to achieve similar rankings for a set of partitions $\mathbb{U}_{sorted}^{(int_s)}$? Internal/external (I/E) matching analysis is discussed in Section 3.6 to determine whether the same base partition and similar ranking of the sorted partitions, suggested by an external CVI, can be obtained using internal CVIs.

Assuming that similar sets of partitions $\mathbb{U}_{sorted}^{(int_s)} = \mathbb{U}_{sorted}^{(ext_s)}$ can be obtained using an internal CVI, the best quality partition for unlabeled data, $U_{(1)}$ from $\mathbb{U}_{sorted}^{(int_s)}$, can be chosen as the base partition U_b. Using the base partition in Algorithm 6, chosen by this criterion, results in an output partition U_f, which is an aggregation of the ensemble of inputs that is optimal with respect to the chosen CVI. This minimizes the average dissimilarity between ensemble matrices and the best quality partition, which best preserves apparent cluster structure or information about X. Next, we discuss the proposed framework, CAFCM.

3.5 Cumulative Agreement FCM (CAFCM) Algorithm

Suppose we have a set of ensemble partitions $\mathbb{U}_{sorted} = \{U_{(r)}\}_{r=1}^{Q}$, each partition having c_r clusters, ranked according to (3.3) in decreasing order of their quality with respect to a specified CVI. Let the best (first) partition $U_{(1)}$ in $\mathbb{U}_{sorted}$ have c clusters and take $U_{(1)} = U_b$. The partitions $\{U_{(r)}\}_{r=2}^{Q}$ are designated as voting partitions with respect to U_b. The entries of each column vector of membership matrix $U_{(r)} \in M_{fc_rN}$ represent the degree of membership of that object in each cluster (rows), and sum to 1, whereas, in the Moore-Penrose pseudoinverse $U_{(r)}^{-1} \in M_{fc_rN}$, each column vector turns into the row (cluster) vector $\{c_i\}_{i=1}^{c_r}$ whose entries sum to 1 [216]. These values can be interpreted as the weight of each data point (rows) in cluster (columns) vector c_i. Multiplying the pseudoinverse of $U_{(r)}$ with base partition U_b gives the weight matrix $W_{r,b} \subset \mathbb{R}^{c \times c_r}$,

$$W_{r,b} = U_b U_{(r)}^{-1}. \tag{3.4}$$

Due to the pseudoinverse $U_{(r)}^{-1}$ in the weight matrix calculation, the entries in $W_{r,b}$ do not lie in the range [0,1]. The relabelling of partition $U_{(r)}$ against the base partition U_b is achieved by multiplying $U_{(r)}$ with this weight matrix $W_{r,b}$, which gives the transformed partition $U_{r,b}$ as

$$U_{r,b} = W_{r,b}U_{(r)}. \tag{3.5}$$

The degrees of membership in the transformed partition $U_{r,b}$ correspond to degrees of memberships in $U_{(r)}$, which are scaled by the entries of $W_{r,b}$. This accomplishes the vote by $U_{(r)}$ to the base partition U_b. The ensemble approach in [214], that computes the weight matrix W [1] as

$$W = U_b U_{(r)}^T, \tag{3.6}$$

is a special case of approach (3.4) (suitable for fuzzy partitions).

Both approaches are demonstrated in Example 1 with a base partition U_b and an ensemble partition $U_{(r)}$. The mutual information between the transformed and the base partition is measured using the soft *Normalized mutual information* index (NMI) $\mathscr{V}_{NMI_s}$ [170] . It can be inferred from the NMI values in Example 1 that $U_{r,b}$ contains more mutual information with respect to the base partition U_b, than U (obtained using (3.6) and (3.5)).

Example 1. *Consider a fuzzy base partition U_b of size 3×4 and an ensemble fuzzy partition $U_{(r)}$ of size 2×4, as given below:*

$$U_b = \begin{bmatrix} 0.8 & 0.9 & 0.0 & 0.1 \\ 0.1 & 0.1 & 0.9 & 0.1 \\ 0.1 & 0.0 & 0.1 & 0.8 \end{bmatrix}, U_{(r)} = \begin{bmatrix} 0.6 & 0.7 & 0.1 & 0.1 \\ 0.4 & 0.3 & 0.9 & 0.9 \end{bmatrix}$$

The weight matrix $W_{r,b}$, computed using (3.4), and the matrix W, computed with (3.6), are as

[1] The columns of weight matrix, W, are normalized in [214] such that $w_{ij} \in [0,1]$, and $\sum_{j=1}^{c_r} w_{ij} = 1$.

follows:

$$W_{r,b} = \begin{bmatrix} 1.35 & -0.09 \\ -0.15 & 0.57 \\ -0.20 & 0.52 \end{bmatrix}, W = \begin{bmatrix} 0.74 & 0.27 \\ 0.15 & 0.39 \\ 0.11 & 0.34 \end{bmatrix},$$

which gives the corresponding transformed partitions $U_{r,b}$ and U, using (3.5), as:

$$U_{r,b} = \begin{bmatrix} .78 & .92 & .05 & .05 \\ .14 & .06 & .50 & .50 \\ .08 & .02 & .45 & .45 \end{bmatrix}, U = \begin{bmatrix} .56 & .60 & .32 & .32 \\ .24 & .22 & .36 & .36 \\ .20 & .18 & .32 & .32 \end{bmatrix},$$

$\mathscr{V}_{NMI_s}(U_{r,b}|U_b)$=*0.2178,* $\mathscr{V}_{NMI_s}(U|U_b)$=*0.0217.*

When multiplying the partition $U_{(r)}$ with weight matrix $W_{r,b}$, each row vector $\{c_i\}_{i=1}^{c_r}$ of $U_{(r)}$ votes for each of the clusters $\{c_j\}_{j=1}^{c}$ of U_b, with weights w_{ij} from the cumulative vote weight matrix $W_{r,b}$. In the general case, each partition $U_{(r)}$ from $\mathbb{U}_{sorted}$, casts its vote with U_b this way in decreasing order of their quality in a sequential manner. Following [213], the base partition $U_b^{(i)}$ at iteration i is calculated by averaging the last base partition $U_b^{(i-1)}$ with transformed partition $U_{r,b}^{(i)}$.

It is evident from (3.4) and (3.5) that $U_{r,b}$, and in turn U_f, will have the same number of clusters as the base partition U_b. If the number of clusters c_r for each ensemble partition is chosen randomly from c_{min} to c_{max}, the criterion of selecting the base partition based on the CVI ranking (refer to Section 3.4) does not always capture the most 'meaningful' information i.e., true number of clusters in the base partition. The problem of finding the true or best number of clusters, using CVIs, is well addressed in the literature. In our work, each ensemble partition having the best number of clusters c_r is obtained using a chosen CVI. For each downspace dataset, FCM clustering is performed with the number of clusters varying from c_{min} to c_{max}. Depending on the evaluation of the CVI, the ensemble partition U_r having the CVI-best number of clusters, c_r is obtained for each downspace dataset.

Our CAFCM algorithm for high-dimensional data clustering using random projection and cumulative agreement based aggregation with FCM clustering is presented in Algorithm 6. In Step

Algorithm 6 CAFCM: Cluster Ensemble for FCM Clustering with Random Projection

Input: Dataset $X \subset \mathbb{R}^{N \times p}$
$\{c_{min}, c_{max}\}$- cluster range (an underestimated and an overestimated value of the number of clusters)
q- downspace dimension, Q- number of random projections
Output: Fuzzy partition U_f.
Step 1: Dataset generation in downspace.
for $r = 1$ to Q **do**
Generate downspace datasets $Y_r \subset \mathbb{R}^{N \times q}$ using $Y = \frac{1}{\sqrt{q}} XT$, where $T \subset \mathbb{R}^{p \times q}$ is the random matrix built using (2.9).
end for
Step 2: Run FCM on each Y_r, obtaining $U_r \in M_{fcN}$: $c = c_{min}$ to c_{max}.
Step 3: Get partitions $\{U_r\}_{r=1}^{Q} \in M_{fc_rN}$, each partition having a CVI-best c_r number of clusters, choosing each c_r with an internal cluster validity index, $\mathscr{V}_{int_s}$.
Step 4: Get a set $\mathbb{U}$ of sorted partitions $\{U_{(r)}\}_{r=1}^{Q} \in M_{fc_rN}$, as given in (3.3), using the cluster validity index, $\mathscr{V}_{int_s}$.
Step 5: Assign the best partition $U_{(1)}$ (from Step 4) as the base partition, i.e., $U_b^{(1)} = U_{(1)}$.
for $i = 2$ to Q **do**
$W_{i,b} = U_b^{(i-1)} U_{(i)}^{-1}$
$U_{i,b} = W_{i,b} U_{(i)}$
$U_b^{(i)} = \frac{i-1}{i} U_b^{(i-1)} + \frac{1}{i} U_{i,b}$
end for
$U_f = U_b$.

1 of the Algorithm 6, multiple downspace datasets $\{Y_r\}$ are generated in fixed lower dimensions; downspace $\mathbb{R}^q$ using random projection, as discussed in Chapter 2 (Section 2.3.5.12). In Step 2, FCM clustering is applied to each downspace dataset Y_r, with the number of clusters varying from c_{min} to c_{max}. In Step 3, the partition U_r with the best number of clusters c_r is obtained for each downspace dataset, using a chosen CVI. This step gives Q fuzzy partitions, each having a CVI-best number of clusters c_r. In Step 4, these Q fuzzy partitions are ranked based on their quality as in (3.3). In our experiments, the *Normalized Partition Entropy* (PEB) $\mathscr{V}_{PEB_s}$ [175] (see Eq. 2.16) was chosen as an internal index in Steps 3 and 4. Step 5 corresponds to the cumulative agreement based aggregation approach, as discussed in this Section. While FCM is part of the title of our algorithm, this scheme applies without change when the ensemble of soft partitions is generated by ANY fuzzy or probabilistic clustering algorithm.

The time and space complexity of the proposed aggregation approach and the three state-of-the-art ensemble approaches that are used for comparison is shown in Table 3.1. Our aggregation

Table 3.1: Time and space complexity of four FCM-based ensemble approaches

Ensemble Methods	Time Complexity	Space Complexity
EFCM [71]	$O(dlNc^2)+O(cQN^2), d=N$	$O(N^2)$
RPFCM-A [72]	$O(dlNc^2)+O(cQN^2), d=cQ$	$O(N^2)$
RPFCM-B [73]	$O(dlNc^2)+O(N(cQ)^2), d=c$	$O(cNQ)$
CAFCM (Proposed)	$O(NQc^2)$	$O(cN)$

l is the number of iterations to termination, d is the dimensions of the matrix on which clustering is applied, c is the number of clusters, N is the number of data points, and Q is the number of random projections.

approach has time complexity of $O(NQc^2)$ for matrix multiplication and computation of pseudoinverse of the rectangular matrix [217]. The fast Moore-Penrose inverse method [217] was used to compute the pseudo inverse of ensemble partition $U_{(r)}$. Therefore, the proposed aggregation method has linear computational complexity in the number (N) of input samples. The CAFCM approach has the minimal space complexity, $O(cN)$, which is required to store the base partition that is updated sequentially in each iteration.

3.6 Experiments

Five sets of experiments were performed. In the first experiment, the effect of using downspace datasets generated by different RP distributions (2.9) and (2.10) on the output partition is discussed. In the second experiment, an internal CVI validation test was performed among all internal CVIs to choose the best 'c_r' corresponding to each RP, and subsequently, a best internal CVI is chosen. In the third experiment, an Internal/External (I/E) agreement test was performed to determine whether the partitions ranking, achieved by a soft external CVI, can also be obtained using a soft internal CVI. Based on the agreement performance of each internal CVI against the soft external CVI, one best internal CVI is chosen to get sorted partitions for each dataset in our ensemble approach. In the fourth experiment, the effect of altering the ordering sequence of ensemble partitions on the output partition for CAFCM is studied. In the last experiment, different cluster ensemble approaches for high-dimensional data clustering are compared. The experiments were performed in the MATLAB environment on a normal PC with the following configurations; OS: Windows 7 (64 bit); processor: Intel(R) Core(TM) i7-4770 @3.40GHz; RAM: 16GB.

3.6.1 Datasets and Parameter Settings

The experiments were performed on the following datasets.

3.6.1.1 Synthetic datasets

Two synthetic datasets, each having $N = 10000$ data points in $p = 1000$ dimensions, were constructed by drawing labeled samples from a mixture of three Gaussian distributions. GM1 is a well separated Gaussian mixture, while GM2 presumably has overlapping Gaussian clusters because its means are closer than those in GM1. The properties of these synthetic datasets are given in Table 3.2.

Table 3.2: Properties of two synthetic datasets GM1 and GM2

Component	**1**	**2**	**3**
Means			
GM1	$(-6,-6,...,-6)_{1000}$	$(0,0,...,0)_{1000}$	$(6,6,...,6)_{1000}$
GM2	$(-2,-2,...,-2)_{1000}$	$(0,0,...,0)_{1000}$	$(2,2,...,2)_{1000}$
Standard deviations in all directions			
GM1	$(1,1,...,1)_{1000}$	$(2,2,...,2)_{1000}$	$(3,3,...,3)_{1000}$
GM2	$(1,1,...,1)_{1000}$	$(2,2,...,2)_{1000}$	$(3,3,...,3)_{1000}$

3.6.1.2 Real datasets

Six publicly available real high-dimensional labeled datasets were chosen to demonstrate the applicability of our approach. The details are as follows:

KDD CUP 99 [218]

We used a sample of KDD CUP 99, which contains a wide variety of internet attacks simulated in a military environment. It consists of 494021 instances of 41 dimensional vectors, and each vector is labeled to specify the attack type. All 41 features were normalized to the interval $[0,1]$ by subtracting the minimum and then dividing by the subsequent maximum so that they all had same scale. This dataset contains 22 types of simulated attacks which fall into one of four main categories [218].

ACT [219]

This is a time-series dataset which contains data representing 19 activities such as sitting, walking, jumping etc., captured by 45 motion sensors over a 5 minute window sampled at 25Hz. Each activity is performed by 8 different subjects. The 5-min signals are divided into 5-sec segments so that 480 $(= 60 \times 8)$ signal segments are obtained for each activity. In each segment, there are a total of 125 $(=$ 5sec $\times$25Hz) rows and 45 columns. We concatenated each segment data to obtain 9120 $(= 480 \times 19)$ instances in 5625 dimensions. All features were normalized to [0,1] using the method discussed earlier.

Forest Covertype [220]

These data consist of 54 cartographic features obtained by the U.S. Geological Survey and U.S. Forest Services, collected from a total of 581012 $(30m \times 30m)$ cells, which were then categorized into 7 forest cover types. This is a challenging dataset for any clustering algorithm as it contains ten continuous features, and 44 binary features (four wilderness types and 40 soil types). Because of the different nature of 54 features, we started developing our own distance metric using Euclidean and Hamming distance with normalized continuous feature (within [0,1]) that accounts for these differences to give similar weight to all the features. But the clustering results were slightly worse than using Euclidean distance alone. After several experiments, we discovered that the binary features do not add too much value in discriminating the forest Cover type. Using the Euclidean distance with scaled continuous features, with all binary features, yielded the best results in our experiments, therefore, Euclidean distance model was used for Forest dataset. The continuous features were normalized to the interval [0,1].

MNIST [221]

This dataset is a subset of a large set of handwritten images from the *National Institute of Standards and Technology* (NIST). It contains a total of 70000 784 $(= 28 \times 28)$ dimensional binary images of the digits 0 to 9. The main problem with handwritten images is that a single character can be written in many often quite different ways. This causes overlapping clusters in the data and makes it challenging for clustering.

HAR [222]

This time-series dataset contains 10299 instances of 6 daily activities performed by 30 subjects, while carrying a waist-mounted smartphone with embedded inertial sensors. It is a preprocessed dataset which has 561 features with time and frequency domain variables.

CIFAR 10 [223]

This dataset contains 60000 $32x32$ color images in 10 classes, with 6000 images per class. The classes are mutually exclusive. We concatenated each image into a $3072 = (32 \times 32 \times 3)$ dimensional feature vector.

3.6.1.3 Parameters

The model and error norms were both Euclidean for FCM except for the two time-series datasets. The Cosine distance was used as the model norm for HAR and ACT, based on its performances in previous studies [72]. This was done by replacing the Euclidean norm by the Cosine distance in the FCM function. In this case, the resultant algorithm is not an alternating optimization since the FCM objective function has been abandoned. So this is an instance of alternating cluster estimation. The number of random projection (RPs), Q is chosen as 30, unless stated otherwise. The weighting exponent $m = 2$, termination threshold $\varepsilon = 0.000001$, and the number of maximum iterations l is chosen as 100 for the MATLAB implementation of FCM. Termination occurs when the absolute value of the difference between successive values of the FCM objective function using either distance is less than ε.

3.6.2 Evaluation Criteria

Adjusted Rand Index

The soft version [177] of the *adjusted rand index*, ARI_s (Hubert and Arabie [53]) is used as an external soft CVI. This index $\mathscr{V}_{ARI_s}(U|U_{gt})$ measures the degree to which a fuzzy partition U matches a crisp U_{gt}. Higher values indicate a better match, so $\mathscr{V}_{ARI_s}$ is a max-optimal CVI. This

index maximizes at 1 when $U = U_{gt}$, and its minimum may be negative when its expected value is not zero.

The *Normalized Partition Entropy* (PEB) $\mathscr{V}_{PEB_s}$ [175], *Partition Index* (SC) $\mathscr{V}_{SC_s}$ [224], *Normalized Partition Coefficient* (PCR) $\mathscr{V}_{PCR_s}$ [174], and *Xie-Beni index* (XB) $\mathscr{V}_{XB_s}$ [50], are used for internal CVI comparisons. More details about these CVIs are given in Chapter 2 (Section 2.4). Based on the min or max-optimality of internal CVIs, a set $\mathbb{U}$ of partitions, ordered in decreasing quality as in (3.3), is obtained for each internal CVI $\mathscr{V}_{int_s}$. The performance of each internal CVI $\mathscr{V}_{int_s}$ against the external CVI $\mathscr{V}_{ARI_s}$ is evaluated using two metrics:

Kendall's rank correlation coefficient [225]

Let E_{ext_s} and E_{int_s} be position vectors of $\mathscr{V}_{ext_s}$ and $\mathscr{V}_{int_s}$ respectively, which contain the ranking of sorted (descending order of quality) partitions. Kendall's coefficient τ measures the similarity between orderings in E_{ext_s} and E_{int_s}, which is given as [225]:

$$\tau = \frac{\begin{array}{c}\text{Number of}\\ \text{concordant pairs}\end{array} - \begin{array}{c}\text{Number of}\\ \text{discordant pairs}\end{array}}{Q(Q-1)/2}. \tag{3.7}$$

Kendall's τ is valued in $[-1,1]$, where 1 is for perfect agreement between two rankings, and -1, for perfect disagreement.

Position of the base partition

The selection of the best quality partition to be the base partition is important in our approach. Let the position of the best partition $U_{(1)}$ (first in E_{ext_s}) in E_{int_s} be denoted as $e_{U_{(1)}}$, then a position metric V_{U_b} is used to evaluate how accurately an internal CVI determines the position of the base partition in E_{int_s}, thus

$$V_{U_b} = 1 - \frac{e_{U_{(1)}} - 1}{Q-1}. \in [0,1] \tag{3.8}$$

The integer $e_{U_{(1)}}$ is the position of the partition in the internal ranking E_{int_s} whose partition matches

$U_{(1)} = U_b$, so $e_{U_{(1)}}$ can take any value from 1 to Q. Suppose $e_{U_{(1)}} = 1$, so that $U_{(1)}$ is the best partition in both rankings E_{ext_s} and E_{int_s}, then $V_{U_b} = 1$. On the other hand, suppose $e_{U_{(1)}} = Q$, then $V_{U_b} = 0$, So the range of V_{U_b} is $[0,1]$, maximum at 1 when the best external and best internal partition are the same; and minimum at 0 when the best external partition is the worst internal partition. The higher the value of V_{U_b}, the higher the ranking of the best partition $U_{(1)}$ in E_{int_s}.

The evaluation criteria to compare the performances of different ensemble approaches are:

Accuracy

The similarity of the final clustering solution U_f with respect to ground truth partition U_{gt} is measured using $\mathscr{V}_{ARI_s}(U_f|U_{gt})$, for all four fuzzy ensemble approaches.

Run-Time

Running time is also an important criterion for comparison, which is related to the scalability of an algorithm. For each dataset, downspace datasets were pre-generated using random projection, and the same projection matrices were used for all algorithms. The number of RPs Q and other parameters were kept fixed for all approaches. We also compare the four fuzzy ensemble approaches based on the aggregation time T_{agg}, required to get a final output partition U_f from the Q ensemble partitions.

3.6.3 Selection of Random Matrix T for Downspace Data (Y) Generation

An experiment was conducted to demonstrate that either of equations (2.9) or (2.10) can be used as the basis for random projection. Using datasets GM1 and GM2 with distributions (2.9) and (2.10), downspace datasets $\{Y_r\}$ ($q = 100$) were generated and used in CAFCM framework for ensemble clustering. The average (10 trials) execution times for downspace data generation and the corresponding soft adjusted rand indices $\mathscr{V}_{ARI_s}$ for output partitions are shown in Table 3.3. These values confirm that there is very little difference between the projections based on equations (2.9) and (2.10). As also shown in [156], both (2.9) and (2.10) are very simple probability distributions and all mathematical operations required to compute $Y = \frac{1}{\sqrt{q}}XT$ are very efficient

Table 3.3: The average $\mathscr{V}_{ARI_s}$ and downspace data generation time for distribution (2.9) and (2.10)

Random Matrix\Datasets	GM1		GM2	
	$\mathscr{V}_{ARI_s}$	Time (s)	$\mathscr{V}_{ARI_s}$	Time (s)
Distribution (2.9)	1.00	0.0266	0.90	0.0267
Distribution (2.10)	1.00	0.0265	0.90	0.0265

and easy to implement. Subsequently, distribution (2.9) was used to generate downspace datasets in all the remaining experiments.

3.6.4 Internal CVIs Validation for Best 'c_r'

The base partition should ideally contain the nominally "true" target value for the number of clusters c_{gt}, that are identified by U_{gt}. In this regard, the best-c validation test [171] was performed using the four soft internal CVIs to estimate c_{gt} in all datasets. The downspace dimension q was chosen as 20. For the choices of $\varepsilon = \beta = 0.25$, and $N = 10000$, $q_o = 1591$, so q is well below the JL bound q_o. In this experiment, FCM was performed on each downspace dataset by partitioning the data at each value of c between $\{c_{min}, c_{max}\}$. The lower (c_{min}) and the upper (c_{max}) limits were chosen such that they under- and over-estimated the possible number of clusters in the data. The best quality partition, U_r, having c_r clusters, was chosen using each CVI based on its min/max optimality. This procedure was performed for each downspace projection, and *the (round) average of the 'best c's was used as an estimate of the true number of clusters in the upspace data*. In this test, randomly chosen subsets of each upspace dataset were used for the big datasets.

Table 3.4 shows the estimated number of clusters in each dataset for each of the internal CVIs. The value of the apparent[2] true number of clusters c_{gt} is shown in the second column of Table 3.4. The values in the last row of Table 3.4 show the square root of the sum of squared errors (RMSE) between c_{gt} and the estimated values for each internal CVI. In this exercise, $\mathscr{V}_{SC_s}$ produces slightly more reliable estimates of c_{gt} than the other three CVIs, whilst $\mathscr{V}_{PEB_s}$ produces the second best estimates of c_{gt}. We remark that these conclusions are not generally applicable. You could test many different CVIs and get different best results. Or you could change datasets and discover that $\mathscr{V}_{SC_s}$ and $\mathscr{V}_{PEB_s}$ performed badly. And so on, ad infinitum. It can also be observed from Table 3.4

[2]We say apparent because it is well known that labeled data which contain $c1$ physically labeled subsets often possess $c2 \neq c1$ "best clusters" with respect to a given model and algorithm [41].

Table 3.4: The average (20 trials) of the best 'c's from all internal CVIs ($\mathscr{V}_{int_s}$)

<Internal CVI>	$\mathbf{c_{gt}}$	$<\mathscr{V}_{PEB_s}>$	$<\mathscr{V}_{SC_s}>$	$<\mathscr{V}_{XB_s}>$	$<\mathscr{V}_{PCR_s}>$
Synthetic Datasets					
GM1	3	3.0	3.0	2.1	3.0
GM2	3	3.0	3.0	2	2.9
Real Datasets					
MNIST	10	10.83	11.98	6	10.12
CIFAR	10	7.1	9.6	6.2	6.4
HAR	6	5.3	6.5	3	4.9
FOREST	7	4.8	6.7	4.2	4.4
ACT	19	18.8	21.5	17.1	18.2
KDD CUP	23	19.3	20.7	19.5	18.8
Root Mean Square Error		5.30	**4.00**	8.04	6.26

that $\mathscr{V}_{PCR_s}$ works best for MNIST, $\mathscr{V}_{PEB_s}$ for ACT, while $\mathscr{V}_{SC_s}$ is best for rest of the datasets. The performance of the CAFCM algorithm was tested using both $\mathscr{V}_{SC_s}$ and $\mathscr{V}_{PEB_s}$ in Step 3, and the final results were very similar. Therefore, $\mathscr{V}_{PEB_s}$ was chosen as the best internal CVI based on this and the I/E agreement test (next) for use in Steps 3 and 4 of CAFCM Algorithm.

3.6.5 The Internal/External (I/E) Agreement Test

In this experiment, we performed the Internal/External (I/E) agreement test, in which the performance of an internal CVI is compared with the performance of an external CVI to assess whether they both yield similar base partition and similar partition rankings or not [41, 167]. We compared the partition rankings and the base partition obtained using the external CVI ($\mathscr{V}_{ARI_s}$), with the partition rankings and base partition obtained using each of the four internal CVIs. Among these four internal CVIs, the CVI which determines the most similar partition ranking and base partition obtained using the external CVI is chosen for use in our framework. Using this best internal CVI, we hope to achieve the desired partition rankings and base partition in the best possible way when ground truth data are not available (the unlabeled case).

Partition rankings comparison

Step 3 of the CAFCM algorithm produces the Q ensemble partitions having best 'c_r' number of clusters. The ranking of each ensemble of fuzzy partitions is established using the external

CVI $\mathscr{V}_{ARI_s}$, and the four soft internal CVIs $\mathscr{V}_{PEB_s}$, $\mathscr{V}_{SC_s}$, $\mathscr{V}_{XB_s}$, and $\mathscr{V}_{PCR_s}$, based on the partition quality. The partitions ranking, E_{int_s}, of each of the four soft internal CVIs, was compared with the partitions ranking, E_{ext_s}, of soft external CVI, $\mathscr{V}_{ARI_s}$, for each dataset using the Kendall rank correlation coefficient.

Base partition comparison

Besides the partition rankings, the selection of the base partition, U_b, is also important in our framework. In this experiment, the position $e_{U_{(1)}}$ of the base partition U_b, the best external CVI partition (first in E_{ext_s}), in each internal CVI partition ranking E_{int_s} was used to compute the position metric V_{U_b} for each internal CVI and for each dataset.

The values of τ and V_{U_b} were computed between rankings $E_{ext_s} = \{E_{ARI_s}\}$ and each ranking of $E_{int_s} = \{E_{PEB_s}, E_{SC_s}, E_{XB_s}, E_{PCR_s}\}$, using (3.7) and (3.8). This procedure was repeated 5 times for each dataset.

Table 3.5: Average Values (5 trials) of Kendall's τ and (V_{U_b}) of internal CVIs against $\mathscr{V}_{ARI_s}$.

<Internal CVI>	$<\mathscr{V}_{PEB_s}>$	$<\mathscr{V}_{SC_s}>$	$<\mathscr{V}_{XB_s}>$	$<\mathscr{V}_{PCR_s}>$
Synthetic Datasets				
GM1	1.00 (1.00)	0.99 (1.00)	0.05 (0.96)	1.00 (1.00)
GM2	0.89 (1.00)	0.99 (1.00)	0.01 (0.41)	0.89 (1.00)
Real Datasets				
MNIST	0.36 (0.98)	0.11 (0.95)	0.01 (0.66)	0.23 (0.97)
CIFAR 10	0.25 (0.98)	0.42 (0.98)	-0.06(0.55)	0.28 (0.98)
HAR	0.68 (1.00)	0.26 (0.98)	0.06 (0.96)	0.58 (0.99)
FOREST	0.17 (0.98)	0.11 (0.98)	0.10 (0.86)	0.15 (0.96)
ACT	0.65 (1.00)	0.36 (1.00)	0.17 (0.94)	0.64 (1.00)
KDD CUP	0.19 (0.93)	0.09 (0.28)	0.10 (0.96)	0.18 (0.93)
Column Average	**0.52 (0.98)**	0.41 (0.89)	0.04 (0.78)	0.49 (0.98)

Table 3.5 shows the averaged values of τ and V_{U_b} (in parentheses) corresponding to the order of the Q fuzzy partitions established by each internal CVI for each dataset. The notation <CVI> in the first row of the table indicates the basis of the τ and V_{U_b} values that are displayed in each column, not to be confused with the value of the CVIs, which are NOT shown. The values in each column are formatted with just enough resolution so that the optimal values can be seen.

Apparently, all of the CVIs except $\mathscr{V}_{XB_s}$ perform well for the two synthetic datasets, which

Table 3.6: The effects of ordered versus random aggregation of ensemble partitions (tabulated values are the 10 trial average of $\mathscr{V}_{ARI_s}$).

Sequence Ordering of Partitions	**GM1** $(q = 30)$	**GM2** $(q = 100)$
Decreasing order of quality	1.00	0.90
Arbitrary order	0.98	0.85

means three internal CVIs are able to achieve almost the same ranking of partitions as obtained by the external CVI $\mathscr{V}_{ARI_s}$. The τ value of all four CVIs degrades for the real datasets. However, the (V_{U_b}) values of $\mathscr{V}_{PCR_s}$ and $\mathscr{V}_{PEB_s}$ are high for all real datasets, which means they reliably choose the best quality partition from the Q ensemble partitions. The last row of Table 3.5 contains column averages, and it shows that overall $\mathscr{V}_{PCR_s}$ and $\mathscr{V}_{PEB_s}$ perform well (with a very slight advantage to $\mathscr{V}_{PEB_s}$), while $\mathscr{V}_{XB_s}$ performs worst.

Based on this overall performance of four internal CVIs in determining partition rankings and the base partition, the performance of $\mathscr{V}_{PEB_s}$ (internal CVI) agrees best with the performance of the soft external index $\mathscr{V}_{ARI_s}$. Therefore, $\mathscr{V}_{PEB_s}$ is chosen to determine the base partition and a set of sorted partitions, required in Step 4 of CAFCM Algorithm. The CVI $\mathscr{V}_{PEB_s}$ is also used in Step 3 of Algorithm 6 to obtain the ensemble partitions, having the best 'c_r' number of clusters.

3.6.6 Effect of Ordering Sequence of Partitions on Output Partition

To demonstrate the effect of altering the ordering of the ranked queue, as shown in (3.3), on the output partition, an experiment was performed using datasets GM1 and GM2 considering two cases viz., where the sequence of ensemble partitions is (i) ordered and (ii) arbitrary. First, we obtained a base partition for each dataset in the manner described. Table 3.6 compares the $\mathscr{V}_{ARI_s}$ values of the output partition obtained when the ensemble partitions are combined in a sequential manner based on their CVI quality as in (3.3) to the $\mathscr{V}_{ARI_s}$ values of the output partition obtained when the $Q-1$ remaining partitions are combined with the base partition in an arbitrary order. The average $\mathscr{V}_{ARI_s}$ values (10 trials) in Table 3.6 make it clear that combining the remainder partitions according to their CVI rank yields better $\mathscr{V}_{ARI_s}$ values (and hence, a better output partition) than an arbitrary combination.

3.6.7 Comparison of Different Cluster Ensemble Methods

In this experiment, we compare the performance of our approach with three existing ensemble approaches for high-dimensional data clustering using random projection with FCM. The performance of all four cluster ensemble approaches is discussed in 5 data groups (**G1-G5**), based on the different attributes of datasets.

Synthetic datasets of different downspace dimensions q (G1)

For synthetic datasets GM1, GM2, experiments were performed for downspace dimension $q = 10, 20, 30, 50, 100$. These q values are corresponding to rogue random projections, which are chosen irrespective of ε and β (below the JL bound) as mentioned in Section 3.6.4. The average $\mathscr{V}_{ARI_s}$ values and ensemble time T_{agg} of all approaches over 5 trials for GM1 and GM2 are shown in Table 3.7. The best performance approach for each downspace dimension is highlighted in bold. It is evident from the values in Table 3.7 that even with $q = 10$, all the ensemble approaches achieve very good clustering results ($\mathscr{V}_{ARI_s} > 0.9$) for the GM1 dataset. This is because the clusters in this dataset are (probably) well separated from each other. EFCM and RPFCM-B get perfect results ($\mathscr{V}_{ARI_s} = 1$) for $q = 10$ and 20. The CAFCM approach performs reasonably well ($\mathscr{V}_{ARI_s} > 0.9$) in significantly less computation time, and achieves perfect results for $q = 30$. It can be concluded from Table 3.7 that the CAFCM approach is $10 - 100$ times faster than the other three approaches. All four approaches get perfect results for $q = 30$ and above, so they are not compared for higher downspace dimensions.

For the GM2 dataset, CAFCM performs significantly better than the other three approaches for all downspace dimensions except $q = 10$. The weak performance of CAFCM for $q = 10$ may be because of the overlapping clusters present in the GM2 dataset. Due to this, the distribution of points among clusters changes in each consensus partition, which in turn, causes the weak agreements of points for any cluster across all consensus partitions. Whereas for $q > 10$, more features make a stronger agreement of each data point for any cluster. The CAFCM algorithm performs aggregation in negligible time compared to the other three approaches, for both synthetic datasets. This is because, unlike other ensemble approaches, CAFCM does not use FCM on a final aggregation matrix to get the final membership matrix.

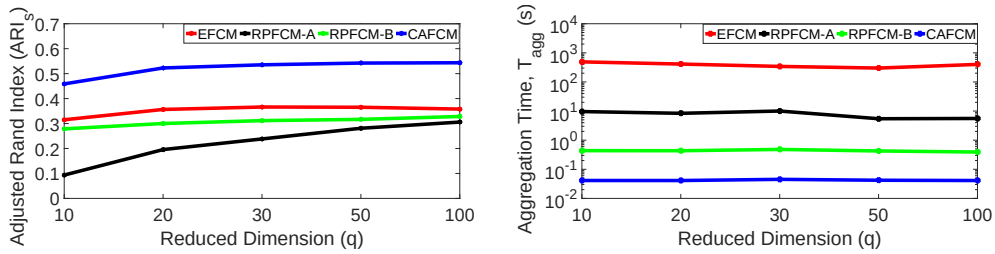
Table 3.7: Average $\mathscr{V}_{ARI_s}$ values and ensemble time T_{agg} (in s) for all approaches on the GM1 and GM2 datasets.

	EFCM		***RPFCM-A***		***RPFCM-B***		***CAFCM***	
q	$\mathscr{V}_{ARI_s}$	T_{agg}	$\mathscr{V}_{ARI_s}$	T_{agg}	$\mathscr{V}_{ARI_s}$	T_{agg}	$\mathscr{V}_{ARI_s}$	T_{agg}
GM1 Dataset $c_r \in \{2,8\}$								
10	1.00± 0.0	68.9	0.97± 0.0	0.36	**1.00± 0.0**	0.15	0.94± 0.0	**0.01**
20	1.00± 0.0	70.9	0.99± 0.0	0.39	**1.00± 0.0**	0.13	0.99± 0.0	**0.01**
30	1.00± 0.0	71.9	1.00± 0.0	0.40	1.00± 0.0	0.16	**1.00± 0.0**	**0.02**
GM2 Dataset $c_r \in \{2,8\}$								
10	**0.76± 0.02**	89.6	0.40± 0.02	0.18	0.75± 0.12	0.15	0.61± 0.01	**0.00**
20	0.60± 0.10	83.2	0.43± 0.15	0.54	0.45± 0.26	11.7	**0.68± 0.02**	**0.01**
30	0.79± 0.18	82.4	0.47± 0.03	0.52	0.30± 0.01	11.03	**0.83± 0.01**	**0.02**
50	0.90± 0.02	71.1	0.55± 0.16	0.54	0.70± 0.22	0.12	**0.90± 0.01**	**0.02**
100	0.85± 0.19	73.1	0.75± 0.23	0.47	0.63± 0.29	0.11	**0.90± 0.02**	**0.02**

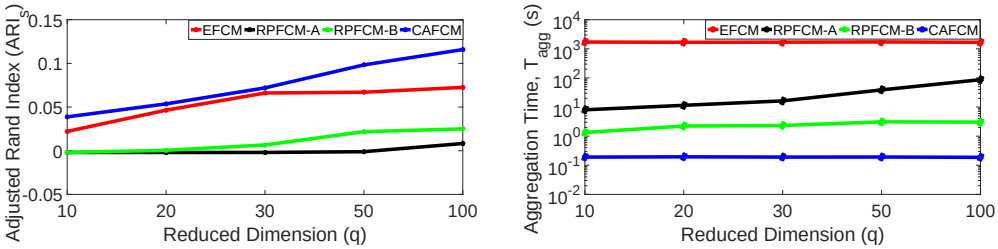
In order to compare the performance of all four ensemble methods with respect to stability, the standard deviation (rounded off) of $\mathscr{V}_{ARI_s}$ values with average values are shown in Table 3.7. CAFCM seems to be the least variable among all the approaches. This might be due to the smoothing effect from sequential averaging of the transformed partitions and base partition (refer to Algorithm 6). The EFCM algorithm seems to be the most stable of the other three approaches.

Synthetic dataset GM2 for different number of RPs, Q (G2)

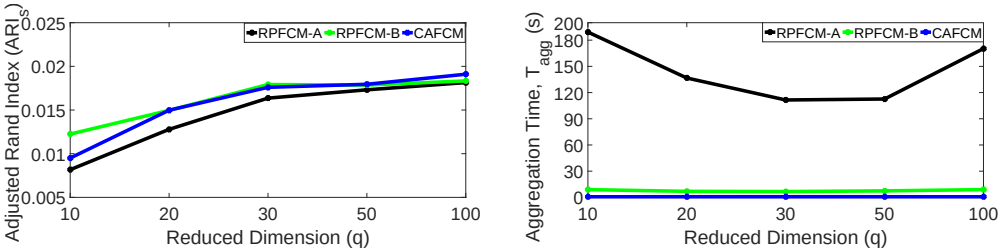
We conducted another experiment for the GM2 dataset for different numbers of RPs, Q (ensemble size). For datasets having high diversity (overlapping clusters) like GM2, increasing Q may be beneficial because there will probably be much more diversity in the random projections due to the mixed clusters in the upspace. Table 3.8 shows the average $\mathscr{V}_{ARI_s}$ values and ensemble time (5 trials) of all approaches for a fixed value of $q(=40)$. It can be noted that CAFCM gives the best performance for all Qs except $Q = 5$ and 10. As expected, the adjusted Rand index ($\mathscr{V}_{ARI_s}$) increases for all approaches as Q increases. Unlike existing approaches, increasing the ensemble size has a negligible effect on the computational time of CAFCM. The maximum speedup is CAFCM:EFCM is 4200 : 1 at $Q = 50$, and the minimum speedup is CAFCM:RPFCM-B 11 : 1 at $Q = 20$.



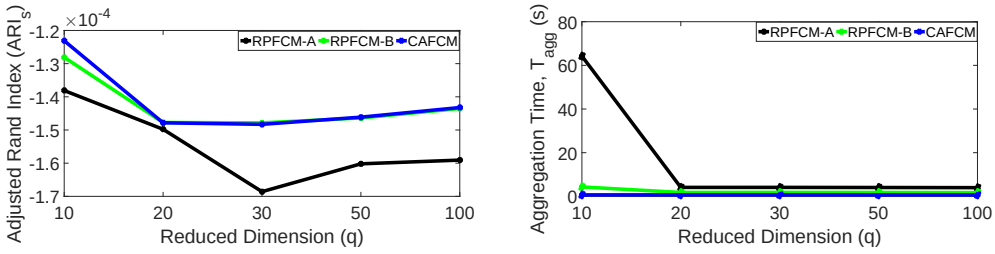
(a) HAR Dataset, $c_r \in \{3, 10\}$



(b) ACT Dataset, $c_r \in \{15, 25\}$



(c) CIFAR Dataset, $c_r \in \{4, 16\}$

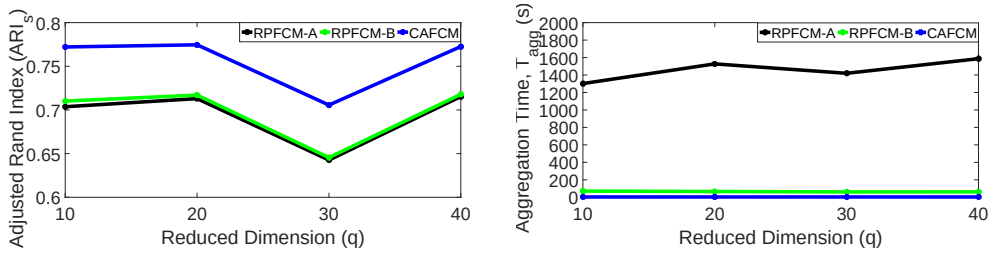


(d) MNIST Dataset,, $c_r \in \{4, 16\}$

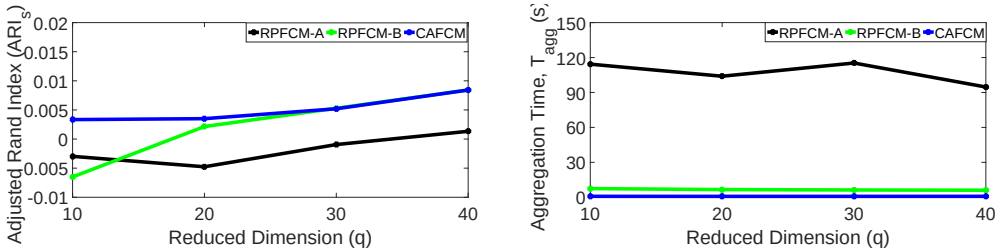
Figure 3.2: $\mathscr{V}_{ARI_s}$ values (in left column) and Aggregation time T_{agg} (in right column) for different downspace dimensions

Table 3.8: Average $\mathscr{V}_{ARI_s}$ values and ensemble time T_{agg} (s) for different number of RPs (Q) on the GM2 dataset.

	EFCM		***RPFCM-A***		***RPFCM-B***		***CAFCM***	
Q	ARI	T_{agg}	ARI	T_{agg}	ARI	T_{agg}	ARI	T_{agg}
5	**0.56**	75	0.43	0.12	0.45	0.12	0.52	**0.00**
10	**0.69**	70	0.44	0.17	0.60	0.12	0.65	**0.00**
20	0.66	88	0.43	0.40	0.70	0.11	**0.74**	**0.01**
30	0.62	98	0.58	0.66	0.71	0.13	**0.79**	**0.02**
40	0.63	97	0.41	0.85	0.74	0.16	**0.85**	**0.03**
50	0.80	126	0.62	1.08	0.82	0.18	**0.89**	**0.03**



(a) KDD Dataset, $c_r \in \{15, 25\}$



(b) FOREST Dataset, $c_r \in \{3, 15\}$

Figure 3.3: $\mathscr{V}_{ARI_s}$ values (in left column) and Aggregation time T_{agg} (in right column) for different downspace dimensions

High-dimensional real datasets (ACT, HAR, MNIST and CIFAR) for different q (G3)

In this group, we discuss the performance on the real datasets ACT, HAR, MNIST and CIFAR, which have relatively high-dimensions (in hundreds and thousands) as compared to the KDD CUP and FOREST datasets, which have smaller upspace dimensions. For G3 datasets, the downspace dimensions $q = 10, 20, 30, 50, 100$ were chosen. Line-plots are used to present the $\mathscr{V}_{ARI_s}$ values of all ensemble approaches for different downspace dimensions, which are shown in the left columns of Figs. 3.2 and 3.3, whereas, the right columns in Figs. 3.2 and 3.3 shows the time performance (on logarithmic scale) of all ensemble approaches for different numbers of downspace dimensions. We did not apply EFCM to MNIST, CIFAR (as $N > 50000$) to avoid an out of memory error, and its associated computational load. Therefore, the time performance for these datasets is shown on a non-logarithmic scale. The minimum and maximum number of clusters in consensus partitions is shown in the title of the figure for each dataset.

Figs. 3.2(a) and (b) show that CAFCM outperforms all other ensemble methods for the two time-series datasets (HAR and ACT). For the image datasets (MNIST and CIFAR), the performance of CAFCM is comparable to RPFCM-B, and outperforms RPFCM-A. The aggregation time for CAFCM is quite small compared to the other three approaches, which agrees with our time complexity analysis as discussed in Section 3.2.

KDD CUP and FOREST Covertype (G4)

The upspace dimensions for FOREST and KDD CUP are 41 and 54, respectively, so we chose the downspace dimensions to be $q = 10, 20, 30, 40$. For each of these datasets, the experiments were performed on a subset of $N = 100,000$ instances. Consequently, the EFCM algorithm was not applied on these datasets to avoid the associated computational load. The performance of all ensemble approaches for these two datasets, is shown in Figs. 3.3 (a) and (b) respectively. The CAFCM approach performs better than the other three ensemble methods for almost all of the downspace dimensions. The CAFCM algorithm achieves near to best accuracy even with $q = 10$ (25%) dimensions for these two datasets. The time performance in Fig. 3.3 (b) shows that even for the large datasets, CAFCM takes negligible time for aggregation compared to the other approaches.

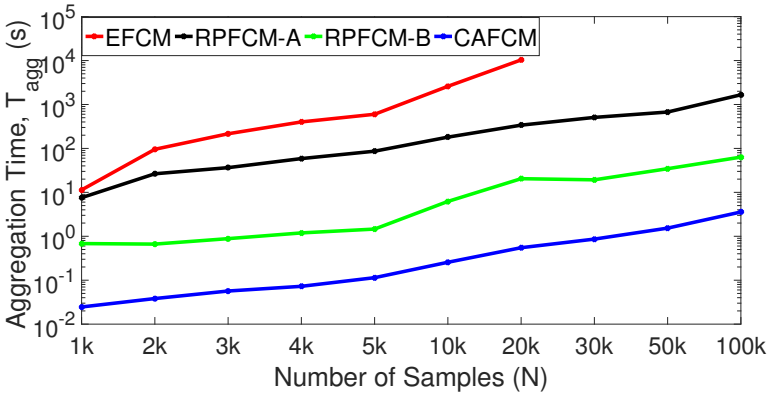


Figure 3.4: KDD CUP Dataset: Aggregation time T_{agg} for different number of samples

Performance of all ensemble approaches for different number of samples (N) (G5)

In order to demonstrate the applicability of our algorithm for big data, the time performance of each ensemble approach for different number of samples of the KDD CUP dataset is presented in Fig. 3.4 (on logarithmic scale). EFCM tests were limited to $N = 20,000$ input samples to avoid the large computational burden. CAFCM takes just a few seconds for even $N = 100,000$ samples. The maximum computational time (for $100,000$ samples) of CAFCM is no more than the minimum time (for $10,000$ samples) taken by the other approaches.

3.7 Summary

This chapter introduces a simple and computationally efficient framework called CAFCM for high-dimensional data clustering, which employs FCM clustering on an ensemble of random projections. Three other state-of-the-art ensemble approaches that also use FCM clustering are compared with CAFCM in this chapter. These approaches require large amounts of space for storing a big affinity matrix. In addition, they also require FCM clustering on a large affinity matrix to get the final partition, so they incur much larger computation time than CAFCM does.

The CAFCM algorithm eliminates the complexity involved in dealing with a final affinity matrix using a cumulative agreement based fuzzy partition aggregation approach. The final CAFCM partition is achieved with a cumulative agreement based relabelling and averaging of the ensemble of fuzzy partitions. Each partition is taken sequentially from a ranked queue established per

equation (3.3). The ranks are computed with a CVI. The highest ranking partition becomes the core partition U_b, and this partition drives the agreement procedure.

Different internal CVIs were used to assess the quality of ensemble partitions having known target (true) numbers of labeled subsets. The performance of four internal CVIs was correlated with the assessments made by the soft external ARI, $\mathscr{V}_{ARI_s}$. The normalized soft partition entropy ($\mathscr{V}_{PEB_s}$) index led to the best final partitions in the experiments presented here. Once the CVIs for steps 3 and 4 in Algorithm 6 are chosen, CAFCM does not require any prior knowledge of the number of clusters that might be present in the dataset, which makes it attractive for real clustering problems.

The superiority of the CAFCM approach was demonstrated by comparing it with three existing approaches on two Gaussian mixture datasets and six real datasets. Experimental results show that CAFCM outperforms the other three approaches in terms of accuracy, stability, space, and time complexity. Experimental results reveal that on average our algorithm runs one to two orders of magnitude ($10-100$ times) faster than other state-of-the-art algorithms, and at best, can achieve speedups in on the order of $4000:1$.

We showed that CAFCM can produce reasonable performance even for downspace dimensions well below the JL bound (rogue random projections). This is very important when the dataset has many features. For example, even with $q = 10$, the CAFCM approach produced good results on the ACT data. The proposed CAFCM algorithm has linear $O(N)$ time complexity in the number (N) of data points. It was also showed empirically that CAFCM algorithm scales linearly in the number of samples (N) for a big dataset (KDD CUP). The CAFCM ensemble time for $N = 100,000$ samples was less than the minimum ensemble time for the other approaches for any number of samples.

This page intentionally left blank.

Chapter 4

Cluster Tendency Assessment and Subsequent Clustering on Big, High-Dimensional Data

This chapter introduces FensiVAT which uses fast data-space reduction and an intelligent sampling strategy to deal with large volumes of high-dimensional data. FensiVAT also provides visual evidence that is used to estimate k (cluster tendency assessment) in the data. Experimental results report that FensiVAT can cluster large volumes of high-dimensional data in a few seconds time without sacrificing accuracy, and it is orders of magnitude faster than other state-of-the-art algorithms.

4.1 Introduction

The scalable VAT (sVAT) algorithm and its extension, siVAT, presented in Chapter 2 answer the clustering tendency assessment problem by suggesting the number of clusters to seek for in the big data X. However, they do not produce actual cluster in X. Two *single linkage* (SL) type clustering algorithms, sVAT-SL and clusiVAT [41, 63] extend sVAT and siVAT, respectively, to cluster the data X into k aligned partitions. Both sVAT-SL and clusiVAT are adequate for large sample size datasets, however, they still suffer from large computation time when the dataset is large in the number of dimensions.

To deal with large amounts of high-dimensional data, this chapter presents a rapid, hybrid clustering algorithm , which efficiently integrates (i) a new *random projection* (RP) based-ensemble technique; (ii) an iVAT algorithm [45], and (iii) a smart sampling strategy, called *Maximin Random Sampling* (MMRS) [47, 83]. The proposed method achieves fast clustering by combining

ensembles of random projections with scalable version of iVAT, hence we call it FensiVAT. FensiVAT aggregates multiple distance matrices, computed in a lower-dimensional space, to obtain the iVAT image in a fast and efficient manner, which provides visual evidence about the number of clusters (k) to seek in the original dataset.

4.2 Related Work

Many papers and surveys [57, 58] discuss different clustering approaches for big datasets. The most popular algorithms for big data clustering are based on partitioning and hierarchical techniques. Among them, *single pass k-means* (spkm) [74, 75], *mini-batch k-means* (MBKM) [76], CLARA (*CLustering LARge Applications*) [77] and CURE (*Clustering Using REpresentatives*) [78], are the most widely known for big datasets. A recently developed algorithm clusiVAT [63] has also shown promising results for big datasets. These algorithms have been discussed in detail in Chapter 2 (Section 2.3.5).

These methods depend on nearest neighbor(s) information, so they are ineffective when clustering high-dimensional data, due to diminishing differences in distance in high-dimensional upspaces [60]. Most of these clustering algorithms use sampling-based strategies to reduce computational time. Therefore, they are fast for large N; however, they are inefficient for datasets jointly large in N and p. At the other extreme, there are methods that excel for large p and small N. There are a number of surveys [56, 59, 60] of high-dimensional data clustering techniques available in the literature. Several widely known clustering algorithms for high-dimensional data [60] are based on subspace clustering [208] or dimensionality reduction [226] (from 'upspace' to 'downspace').

Subspace clustering [81, 227] is an extension of traditional clustering that seeks to find clusters in different subspaces within a dataset. These methods [208] do not suffer from nearest neighbor problems in high-dimensional space. PROCLUS [81] is a subspace clustering approach, which first samples the data, then selects a set of k medoids, and iteratively improves the clustering. PROCLUS is capable of discovering arbitrary shaped clusters in high-dimensional datasets. However, it is very sensitive to input parameters. In practice, most of the subspace clustering approaches suffer from long run-times and/or low accuracies for large volumes of high-dimensional data. Dimensionality reduction based approaches such as global projection (e.g., *singular value*

decomposition (SVD)) and *random projection* (RP) based ensemble approaches [72, 82] reduce computational time by clustering the projected data in a lower dimensional space. However, they too suffer from space and/or time complexity problems for big datasets, and clusters in the projected space do not necessarily correspond to clusters in the original space.

There has been a limited amount of work on clustering algorithms that work efficiently on datasets that are jointly large in (N) and (p). These algorithms are hybrid in nature. They use random sampling or dimensionality reduction techniques either together [80] or with some other approach such as axis-parallel partitioning [228] or indexing [229], to reduce computation time. *O-cluster* [228] (*Orthogonal partitioning CLUSTERing*) combines active random sampling with an axis-parallel partitioning strategy to identify continuous areas of high density in the input space. O-cluster works well for high-dimensions, but it does not function optimally when the dimensionality is low. The low number of dimensions makes the use of axis-parallel partitioning algorithm problematic. O-cluster uses a parameter *sensitivity*, ρ, which require careful tuning when it is applied to a dataset where the number of clusters is unknown, and also, it requires a large buffer size to correctly identify all original clusters in the dataset. *GARDEN k-means* (GARDENkm) [79] begins with a Gamma region density partitioning scheme for data summarization. Using this partitioning technique, it eliminates the empty regions in the data space so that only tight, high-dense regions are retained. Then, it utilizes k-means to cluster the summarized information. Like k-means, this algorithm also requires the number of clusters (k) as an input prior to clustering. Another hybrid approach, *fast spectral clustering* (FastSpec) [230], combines random sampling and FastMap projection with spectral clustering to identify clusters. In an intermediate step, it computes an affinity matrix of $N \times r$ size in the downspace (r is the number of random samples, $r = 300 \times k$ [230]), and a diagonal matrix of $N \times N$ size, which can be very big for big datasets. Therefore, FastSpec has very high space complexity.

Almost all these approaches use sampling and/or dimensionality reduction for clustering high-dimensional massively large datasets. However, random sampling may fail [41] to provide a faithful representation of cluster structure in the input data, which may degrade clustering accuracy. The authors of [78] give a theorem that (in probability) insures representative random samples, but this result often leads to samples that are roughly half the size of the original data. This is still too large when N is big, say $N > 10^7$. Therefore, all the methods reviewed either take hours for

large size datasets having hundreds to thousands of dimensions and/or sacrifice accuracy for faster computation time.

In the next section, we discuss our FensiVAT algorithm.

4.3 FensiVAT algorithm

The FensiVAT algorithm finds its root in the VAT/iVAT algorithm. Scalable VAT-based SL clustering algorithms, sVAT-SL and clusiVAT, first find $n << N$ MMRS sample points that are representative of full data, and then construct an image of this sample using distance matrix D'^{*}_{n}. Supposedly one of these images suggests that the best guess for the number of clusters in X is k. Having this estimate, the longest $k-1$ edges of the MST are cut resulting in k connected subtrees (the clusters). For big data clustering, sVAT-SL and clusiVAT [63, 135] extend this partition of X_n non-iteratively to the $(N-n)$ unlabeled objects in X using the nearest (object) prototype rule (NPR).

The two main time-consuming steps in both algorithms for large, high-dimensional data clustering are (i) the Maximin step of MMRS sampling; and (ii) Extension. In the Maximin step, k' distinguished objects are chosen which are furthest from each other in the dataset. This requires the computation of a $k' \times N$ distance matrix, (say) $\hat{D} \subset D_N$, in p dimensions. In the extension step, the labels of n samples are used to label the remaining $(N-n)$ objects in the data using the NOPR. This requires the computation of an $n \times (N-n)$ distance matrix (say) $\hat{\hat{D}} \subset D_N$, the distances again being in p dimensions. In an intermediate step of clusiVAT, SL clustering is applied to an $n \times n$ matrix, D_n. Because, N and $(N-n)$ can be very large for big datasets, and the distance computations ($\hat{D}$ and $\hat{\hat{D}}$) are performed in the original (high) p-dimension, siVAT-SL and clusiVAT take a large amount of time to cluster large volumes of high-dimensional dataset.

To address the above challenges, FensiVAT integrates random projection with MMRS sampling and VAT method. In FensiVAT, MMRS is performed in randomly projected downspace, so we call MMRS in downspace as Near-MMRS and MM in downspace as Near-MM. The essential steps in FensiVAT are: (i) **Near-MMRS Sampling:** MMRS sampling is done in Y, the downspace (subscript d), to obtain a small and diverse subset $\tilde{S}_d \subset Y$ from the full dataset, which is then lifted by using the same indices to the upspace (subscript u), $\tilde{S}_u \subset X$; and (ii) **Ensemble:** Aggregation

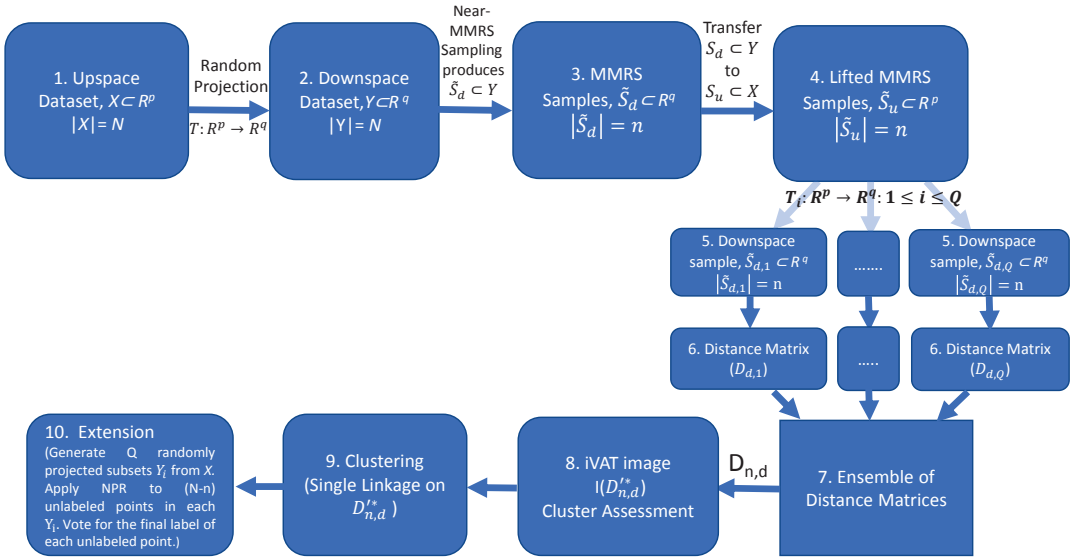


Figure 4.1: The FensiVAT architecture.

of Q $n \times n$ distance matrices, $\{D_i\}_{i=1}^{Q}$, computed from multiple random projections of $\tilde{S}_u$ to obtain Q sets of Near-MMRS samples $\{\tilde{S}_{d,i}\}_{i=1}^{Q}$ in the downspace. This is done to obtain a reliable output iVAT image, $I(D'^*_{n,d})$, which visually suggests the number of clusters, k, in the dataset, (iii) **Clustering**: SL partitioning on the $D'^*_{n,d}$ to obtain k clusters, and (iv) **Extension in downspace** to label the remaining data points in the dataset Y by giving them the label of their nearest object from sample $\tilde{S}_d$. Pseudocode of FensiVAT algorithm is given in Algorithm 7. Below, we explain each step of the FensiVAT algorithm, whose architecture is shown in Fig. 4.1.

Near-MMRS Sampling

The input data to FensiVAT is a set of objects $O = \{o_1, o_2, ..., o_N\}$ in the form of a set of feature vectors $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\} \subset \mathbb{R}^p$; N and p are large. In the second step, random projection is applied to $X \subset \mathbb{R}^p$ to obtain downspace data $Y \subset \mathbb{R}^q$. Unlike ensemble-based approaches, random projection is applied only once to the large dataset to obtain a downspace dataset, which is subsequently used for the sampling step. It is possible that the clusters in a sample from the downspace dataset Y are drastically different from the points that MMRS sampling would produce when applied to X. This point is discussed below with the Near-MMRS sampling (third) step of the FensiVAT algorithm.

Near-MMRS sampling begins by finding the k' Maximin (MM) samples (*distinguished objects*) in Y, which are furthest from each other. MM sampling starts at a random point and then chooses as the second MM sample the point which is furthest from the initial point with respect to a chosen measure of distance on the set being sampled. The third object selected maximizes the distance from both of the first two points. This process continues until k' MM samples are chosen. Then, each object in O is grouped with its nearest *distinguished object*. This stage divides the entire dataset O into k' groups, $\{Z_i\}_{i=1}^{k'}$ by associating $|Z_i|$ objects to the ith *distinguished object*, which provides a representation of each of the k' clusters. This grouping task requires the computation of a $k' \times N$ matrix $\hat{D}$ now done in downspace ($\mathbb{R}^q$), which reduces the computational time that would be needed for the calculations of a $k' \times N$ distance matrix of $p-$dimensional feature vectors. Finally, the sample $\tilde{S}_d$ of size n (just a small fraction of N), is built by selecting random data points (Random sampling (RS)) from each of the k' clusters $\{Z_i\}_{i=1}^{k'}$. The number of points, n_i extracted from cluster Z_i is proportional to the number of data points in Z_i, namely, $n_i = \lceil n \times |Z_i|/N \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function.

The approximate distance preservation (within $1 \pm \varepsilon$) property of randomly projected pairs from X asserted by Theorem 2.1 (Chapter 2) supports a belief that if the Near-MM distinguished objects in Y are generated by applying MM to it, beginning with the same initial point, that the MM samples in Y should be the same or close (due to approximation distance error) to the k' MM points in X (upspace) that would be produced by MM sampling in the upspace. Two Propositions from [47] discussed in Chapter 2 (Section 2.2.3) about MMRS procedure provide some justification for believing this.

In Near-MMRS sampling, MMRS sampling (Algorithm 3) is performed in the randomly projected lower dimensional space Y (downspace). Therefore, if dataset X has k *compact separated* (CS) clusters and $k' \geq k$, and if downspace data Y has k CS clusters, and carries a JL certificate ($q \geq q_0$) as in Theorem 2.1, then Proposition 1A guarantees that Near-MMRS sampling will select at least one distinguished object from each of the k clusters, and Proposition 1B assures us that the proportion of the objects in each cluster in the Near-MMRS sample will be similar to the proportion of objects in each subset in the original data.

Algorithm 7 FensiVAT

Step 1. Input: Dataset $X = \{\mathbf{x}_1, .., \mathbf{x}_N\} \subset \mathbb{R}^p$
q- downspace dimension
k': an overestimate of the true number of clusters, k, in X
n: an approximate sample size
Q- number of RPs
Output: $D'^{*}_{n,d}$ - iVAT reordered dissimilarity matrix of $D_{n,d}$.
U - cluster membership vector of data points in O.
Step 2. Dataset generation in downspace.
Generate downspace datasets $Y \subset \mathbb{R}^{N\times q}$ using $Y = \frac{1}{\sqrt{q}}XT$, where $T \in \mathbb{R}^{p\times q}$ is the random matrix as discussed in Section 2.3.5.12.
Step 3: Near-MMRS Sampling: MMRS on Y
Apply MMRS on Y returning a MMRS sample $\tilde{S}_d$ of size n (Algorithm 3)
Step 4-7: Ensemble method to obtain a reliable iVAT image.
Generate Q, downspace datasets $\{\tilde{S}_{d,i}\}_{i=1}^{Q} \subset \mathbb{R}^q$ from $\tilde{S}_u \subset \mathbb{R}^p$ ($\tilde{S}_d \to \tilde{S}_u$), using random matrices $\{T_i\}_{i=1}^{Q} \in \mathbb{R}^{q\times q}, |\tilde{S}_u| = |\tilde{S}_d| = n$.
Compute distance matrices $\{D_{d,i}\}_{i=1}^{Q}$ from $\{\tilde{S}_{d,i}\}_{i=1}^{Q}$.
$D_{n,d} \leftarrow 0$ (Initialize a $n \times n$ distance matrix).
for $i = 1$ to Q **do**
$W_i = NormalizeRows(D_{d,i})$
$V_i = \frac{1}{2}(W_i + W_i^{\mathsf{T}})$
$D_{n,d} = D_{n,d} + V_i$
end for
Step 8: Apply VAT/iVAT on $D_{n,d}$, returning $D'^{*}_{n,d}$, P, h. (Algorithms 1 and 2)
Choose the number of clusters k using image of $D'^{*}_{n,d}$.
Step 9: Clustering:
Find indices u of k largest values in MST cut magnitudes h.
Form the aligned partition, $U^* = \{u_1 : u_2 - u_1 : ... : u_k - u_{k-1}\}$
$U_{\tilde{S}_u} = U^*_{P_i},\ 1 \le i \le k$.
Step 10: Extension in Downspace:
Generate downspace datasets $\{Y_i\}_{i=1}^{Q} \subset \mathbb{R}^q$ using RP, $|Y_i| = N$.
for each Y_i **do**
Consider sample $Y^{(i)}_{\tilde{S}_d} \subset \mathbb{R}^q$ and $Y_i - Y^{(i)}_{\tilde{S}_d} \subset \mathbb{R}^q$, where $|Y^{(i)}_{\tilde{S}_d}| = n$, and $|Y_i - Y^{(i)}_{\tilde{S}_d}| = N - n$.
for each data point, $\hat{\mathbf{y}} \in Y_i - Y^{(i)}_{\tilde{S}_d}$ **do**
$l = \arg\min_{i\in\tilde{S}_d} \{dist\{\hat{\mathbf{y}}, \mathbf{y}_i\}\}$
$U^{(i)}_{\hat{\mathbf{y}}} = U_l$
end for
end for
U = Mode of labels for each data points $U^{(i)}_{\hat{\mathbf{y}}}$.

Distance Matrix using Ensemble Method

The third (previous) step provides n samples in the downspace, $\tilde{S}_d \subset \mathbb{R}^q$, which can be used to build an $n \times n$ distance matrix $D_{n,d}$. A reliable iVAT image is needed in order to select the number

of clusters obtained by SL in penultimate steps of FensiVAT. The VAT/iVAT image provides a subjective visual assessment of potential cluster substructure based on how distinctive the dark blocks (clusters) appear in the image. However, the quality of the image of the reordered distance matrix $D'^{*}_{n,d}$, obtained by applying VAT/iVAT to $D_{n,d}$, often turns out to be very poor due to the unstable nature of random projection. Hence, we turned to an ensemble-based approach to obtain a good quality iVAT image from multiple reordered distance matrices ($\{D'^{*}_{d,i}\}_{i=1}^{Q}$) in the downspace. Since the ordering of the data in every reordered matrix $D'^{*}_{d,i}$ may be different, it is not feasible to directly aggregate multiple reordered distance matrices ($\{D'^{*}_{d,i}\}_{i=1}^{Q}$). Therefore, a new method is devised to aggregate the Q $n \times n$ ensemble of distance matrices to obtain a better quality iVAT image.

The new ensemble-based approach to build the aggregate $n \times n$ distance matrix, $D_{n,d}$ is shown in Steps 4-7 of Algorithm 7. First (in the fourth step), the Near-MMRS samples $\tilde{S}_d$ are back-projected to the upspace by using the sample indices in $\tilde{S}_d$ to identify the corresponding samples $\tilde{S}_u$ in X. Then random projection is applied to $\tilde{S}_u$ Q times, resulting in the downspace sample sets $\{\tilde{S}_{d,i}\}_{i=1}^{Q}$ (Step 5 in Fig. 4.1).

Next, the Q downspace samples, $\{\tilde{S}_{d,i}\}_{i=1}^{Q}$ are used to compute Q distance matrices, $\{D_{d,i}\}_{i=1}^{Q}$ in the sixth step. Since the downspace samples can be drastically different from each other due to the random nature of the mapping from upspace to downspace, the distance matrices will be diverse. Therefore, the Q $n \times n$ distance matrices are aggregated to obtain a more reliable distance matrix, which in turn yields a better iVAT image than the Q individual iVAT images. The aggregation (Step 7) is performed in three sub-steps: Normalization, Symmetrization, and Summation.

Normalization: Since each distance matrix is computed from randomly projected samples, the distance of each data point from the remaining data points may have a different range in different distance matrices. Therefore, the distance of each data point from the remaining data points is normalized to a unit scale in each distance matrix. The rows (or columns) of each $D_{d,i}$ are normalized such that the ij-th entry of $D_{d,i}$ is in $[0,1]$, and the row sum of each row is 1.

Symmetrization: The normalized distance matrices, W_i (in Algorithm 7), may be asymmetric. The input distance matrix to VAT/iVAT must be symmetric, so all normalized distance matrices are replaced by symmetric matrices using $V_i = \frac{1}{2}(W_i + W_i^{\mathsf{T}})$.

Summation: After symmetrization, the output distance matrix $D_{n,d}$ is obtained using element-

wise summation of the Q distance matrices $\{V_i\}_{i=1}^{Q}$.

Cluster Assessment

In the eighth step, the VAT/iVAT algorithm is applied to distance matrix $D_{n,d}$, which returns a reordered matrix, $D'^{*}_{n,d}$ and the cut magnitudes of the MST links, h. The visualization of $D'^{*}_{n,d}$ using $I(D'^{*}_{n,d})$ suggests the number of clusters k present in the dataset. The comparison of iVAT images obtained using the Q single distance matrices $\{D_{d,i}\}_{i=1}^{Q}$ to the image based on $D_{n,d}$ is discussed in Section 4.5. Although human interpretation is used to estimate the number of clusters by viewing the output iVAT image, there are also methods [231–233] to automatically determine the number of clusters from VAT/iVAT images or $D'^{*}_{n,d}$.

Clustering

All single linkage partitions are *aligned* partitions [234] in the VAT/iVAT ordered matrices, so SL is an obvious choice for the clustering algorithm in Step 9. Having the estimate of the number of clusters, k from the previous step, the $k-1$ longest edges are cut in the iVAT-built MST, resulting in k single linkage clusters.

If the dataset is complex and clusters are intermixed, cutting the $k-1$ longest edges may not always be a good strategy as the data points (outliers), which are typically furthest from normal clusters, might comprise most of the $k-1$ longest edges of the MST, leading to misleading partitions. Such data points need to be partitioned (usually in their own cluster) before a reliable partition can be found via the SL criterion. However, the iVAT image provides visual evidence as to how large the clusters should be. Thus, if the size of SL-clusters does not match well the visual evidence, then the partition can be discarded (perhaps choosing a different clustering algorithm to partition the sample of feature vectors in $\mathbb{R}^p$ or throwing out data from small clusters).

Next, the aligned partition $\{U^{*}_{P_i}\}_{i=1}^{k}$ is calculated using the indices of the $k-1$ longest edges. Since the objects in $U^{*}_{P_i}$ are arranged according to VAT reordering indices P, the cluster labels in vector $U^{*}_{P_i}$ are reordered to match the index-ordering of samples $\tilde{S}_u$ in the original objets, resulting in the partition $U_{\tilde{S}_u}$ of $\tilde{S}_u$.

Extension

In the extension step (Step 10) of FensiVAT, the remaining $\tilde{N} = (N - n)$ data points in O are labeled by giving them the label of their nearest object in $\tilde{S}_d$. This requires the computation of an $n \times \tilde{N}$ size matrix, $\hat{\tilde{D}}$, with computational complexity $O(qn\tilde{N})$. In this step, the sample $\tilde{S}_d$ and feature vectors Y in $\mathbb{R}^q$ (obtained in Step 2) are used to compute the distance matrix $\hat{\tilde{D}}$. This further reduces the computation time which would be needed for the equivalent operation in $\mathbb{R}^p$.

Next, the remaining $\tilde{N}$ data points in O are labeled using this distance matrix, based on the label of the nearest object in $\tilde{S}_d$. Although a single random projection (RP) might be sufficient to achieve comparable accuracy in the NPR labeling step, several [235] RPs are used to best ensure robust nearest neighbour search in NPR. First, multiple RPs are applied on the full dataset to get multiple Ys. Then, the sample labels are extended to each of these Ys using NPR, which would give multiple sets of labels $\{U_{\hat{\mathbf{y}}}^{(i)}\}_{i=1}^{Q}$ for full dataset. The final labels (U) are selected using voting, based on the labels cast by each voter from each RP, for each remaining data point in O.

4.4 Time Complexity

For dataset $Y \subset \mathbb{R}^q$, the computational complexity in the first and second stages of Near-MMRS (Algorithm 3) sampling are $O(qk'N)$, and the last stage requires $O(qn^2)$ operations to build sample $\tilde{S}_d$. The complexity in computing multiple distance matrices in ensemble step is $O(qn^2Q)$, and the complexity of iVAT is $O(qn^2)$. The computational complexity to compute the aligned partition and reordering in the clustering step is $O(N)$. The computational complexity of extension step is $O(qn\tilde{N}Q)$. So, the overall complexity of FensiVAT is $O(max\{qk'N, qn^2, qn^2Q, N, qn\tilde{N}Q\})$. In other words, FensiVAT is linear in N, i.e., it is scalable with respect to the number of samples while simultaneously reducing the dimensional complexity from p to q.

4.5 Experiments

Six set of experiments were performed on two synthetic and six real datasets, that are relatively big in sample size (N) as well as in dimension (p). In the first experiment, the cluster distribution obtained using three sampling schemes are compared. In the second experiment, the

quality of iVAT images, obtained using Q distance matrices built with Q RPs, is compared to the quality of the iVAT image obtained using ensemble distance matrix. In the third experiment, the capability of FensiVAT is explored to visually suggest the number of clusters in big datasets in the downspace dimension. In the fourth and fifth experiments, the performance of FensiVAT for different numbers (Q) of RPs in the ensemble step and for different downspace dimensions $q = 5, 10, 20, 30, 50$, and 100, respectively, is investigated. In the last experiment, the performance of FensiVAT is compared with nine state-of-the-art methods, discussed in Section 4.2. These nine approaches are clusiVAT [63], MBKM [76], CLARA [77], spkm [74] (a crisp adaptation of the single pass fuzzy k-means [74]), CURE [78], a RP based ensemble technique, called RP-EN [72, 82], PROCLUS [81], GARDENkm [79], and FastSpec [80]. While the comparison of FensiVAT with O-Cluster [228] would have been desirable, a publicly available code does not exist, and [228] does not offer sufficient implementation details to develop a reliable in-house version. The experiments were performed using MATLAB, WEKA and ELKI software on a Windows 7 (64 bit) PC with 16 GB RAM and Intel i7 @3.40 GHz processor.

4.5.1 Datasets and Parameter Settings

We performed experiments on the following datasets.

Synthetic datasets:

Two synthetic datasets, each having $N = 100,000$ data points in $p = 1000$ dimensions, were constructed by drawing labeled samples from a mixture of $k = 3$ Gaussian distributions. GM1 is a well separated Gaussian mixture, while GM2 has overlapping Gaussian clusters. The properties of these synthetic datasets are provided in Table 3.2 (Chapter 3).

Real datasets

Six publicly available real, high-dimensional (large volumes) datasets were chosen to demonstrate the applicability of FensiVAT. The details of all real datasets[1] are given in Table 4.1. All

[1]These datasets can be found at the UCI machine learning data repository [236] and [221]. The features are normalized to the interval [0,1] by subtracting the minimum and then dividing by the subsequent maximum so that they all had the same scale.

Table 4.1: Properties of real datasets

Dataset	**N**	**p**	**k**	**Dunn's Index**, $DI(k, U_{gt})$
US Census 1990	2458285	68	Unknown	Unknown
KDD CUP'99	4898431	41	23	0 (Non-CS)
FOREST	581012	54	7	0.002 (Non-CS)
MiniBooNE	130064	50	2	0 (Non-CS)
MNIST	70000	784	10	0.15 (Non-CS)
ACT	9162	5625	19	0.01 (Non-CS)

datasets are labeled except the US Census 1990 dataset. We point out that the labeled subsets in these datasets may or may not correspond to computationally identifiable sets of clusters.

Parameter settings

In all the experiments, FensiVAT and clusiVAT parameters, k' and n are randomly chosen between $2k$ and $4k$, and $10k$ and $30k$ respectively (unless stated otherwise), where $k', n \in \mathbb{Z}$, and k is the number of labeled subsets in the ground truth data. The number of random projections Q in the ensemble step of FensiVAT algorithm is chosen as 5, unless stated otherwise. For MBKM, the parameter batch size = 50, the iteration limit = 100, and the termination threshold = 0.001. The initial centroids for MBKM were built using 'kmeans++' method to speed-up convergence. For CLARA, the number of samples was 5, and sample size was $40 + 2k$ [77]. For spkm, n is 10% of N. The k-means++ seeding technique was used to choose k initial centroids in CLARA. For CURE, the number of representative (well-scattered) points in clusters is 5, shrink factor is 0.7, and the number of (random) samples is kept the same as n in FensiVAT. For PROCLUS, the average dimensionality of clusters, x_d, were chosen based on the grid search for best clustering performance. For the ensemble clustering method, RP-EN, we chose the number of random projections as 20, the weighting exponent as 2, termination threshold as 0.000001, and the iteration limit as 100. For FastSpec, we used $r = 300k$ for all datasets except KDD Cup, US Census, and FOREST. FastSpec [80] has very high space complexity, so we could not run it on our PC for datasets using $r = 300k$ [80] with very big N and k such as KDD Cup, US Census, and Forest dataset, so we ran it with $r = 100k$ for FOREST, and $r = 10k$ for the KDD and US datasets, and using sparse MATLAB function to store diagonal matrices. The downspace dimensions for RP-EN and FastSpec were the same as those chosen for FensiVAT. The authors of [79] kindly provided us

the GARDEN k-means code, written in C++. The density threshold in GARDENkm was chosen based on the best performance, for each dataset. All the experiments were performed 20 times on each dataset except KDD (5 times) and the average results are reported.

4.5.2 Evaluation Criteria

Partition Accuracy

For all datasets, except US Census 1990, the quality of the output crisp partition obtained by various clustering algorithms is assessed using ground truth information, U_{gt}. The similarity of computed partitions with respect to ground truth labels is measured using the *partition accuracy* (PA).

Dunn's Index

Since the ground truth information is not available for US Census 1990 dataset, an internal CVI, *Dunn's Index* (DI) [48], is used to evaluate the quality of output partitions for all clustering algorithms for this dataset.

Dunn defined CS clusters in X with a distance criterion, and showed that X contains CS clusters if and only there is a partition U^* of X for which $DI(k,U^*) > 1$. Havens *et al.* [133] related the effectiveness of VAT in showing cluster tendency to DI. The sVAT-SL [134] partition is equivalent to the SL partition for CS datasets. Since the recursive version of iVAT [45] is used in FensiVAT algorithm, the same rule applies to FensiVAT. If a dataset does not contain k-CS clusters, then FensiVAT is not guaranteed to find the same partition as SL. However, we show in our comparison experiments that FensiVAT produces a good approximation for large datasets whether they are CS or not. The DI of the ground truth partition for all real datasets (except US Census) is shown in Table 4.1.

Chi-square distance

The similarity between two cluster distributions (histograms) A and B can be compared using the chi-square distance [237], $\chi^2(A,B)$, as follows

$$\chi^2(A,B) = \frac{1}{2}\sum_{i=1}^{f}\frac{(A_i - B_i)^2}{A_i + B_i}, \tag{4.1}$$

where f is the number of bins in histograms of the data. We take $f = k \in \mathbb{Z}$ as the number of bins. The value of χ^2 is in $[0,\infty]$, and a lower value implies higher similarity between two distributions.

Run-time

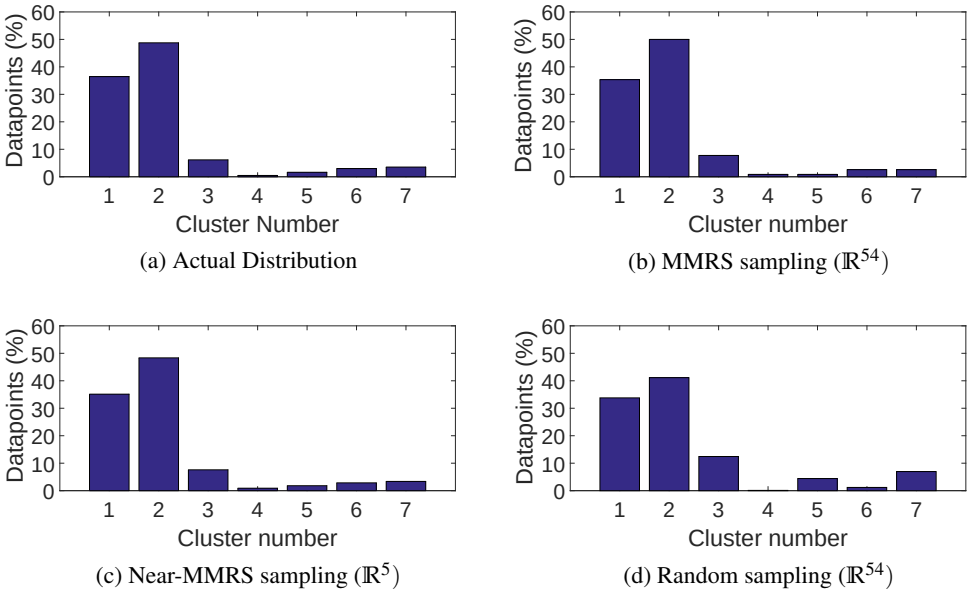
We also report the run-time (in seconds), another important criteria for comparison, which is related to the scalability of an algorithm.

4.5.3 Cluster Distribution using Various Sampling Schemes

In this experiment, we compare the cluster distribution in samples, obtained using three sampling schemes viz., random, MMRS, and Near-MMRS sampling for four datasets GM2, MNIST, ACT, and FOREST, which have different numbers of labeled subsets (k) and cluster distributions. First, for each sample, obtained from a sampling scheme, a histogram is computed using the label distribution in that sample in $\{1,2,..,k\}$ integer bins. Then, the similarity of each cluster distribution is computed with respect to the actual (ground truth) distribution in the full dataset using chi-square distance.

The average distribution of data points in samples obtained using the three sampling schemes for FOREST are shown in Fig. 4.2. The distribution of data points using MMRS and Near-MMRS sampling are very similar to each other, and to the actual distribution in the data, whereas, with random sampling alone, subsets 3, 5 and 7 are oversampled, while subsets 4 and 6 are undersampled. MMRS and Near-MMRS sampling both acquired at least one data point from each subset in every trial. On the other hand, random sampling did not select any data points from subset 4 on $4/10$ trials (not shown in Fig. 4.2).

Table 4.2 shows the run-time and average (20 trials) chi-square values between cluster distri-



(a) Actual Distribution

(b) MMRS sampling ($\mathbb{R}^{54}$)

(c) Near-MMRS sampling ($\mathbb{R}^{5}$)

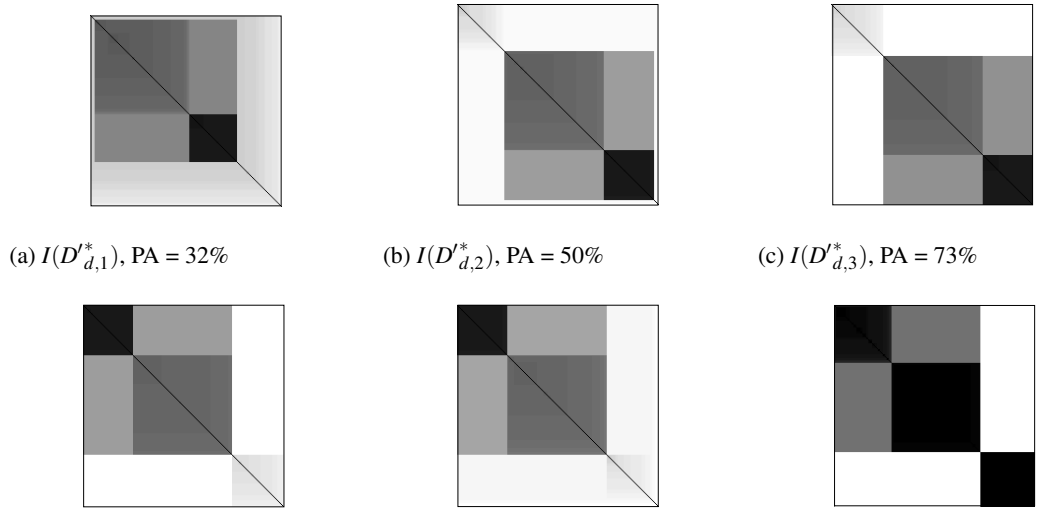
(d) Random sampling ($\mathbb{R}^{54}$)

Figure 4.2: Histogram of data in the Forest Dataset. The MMRS and Near-MMRS parameters are $k^{'} = 30$, and $n = 100$ samples, and $q = 5$ (for Near-MMRS).

butions for full and sampled datasets for each sampling scheme. The number of samples n for each sampling scheme, the number of distinguished objects k' for MMRS and Near-MMRS sampling scheme, and the downspace dimension q for Near-MMRS sampling scheme are also shown. The values in Table 4.2 show that the chi-square values for MMRS and Near-MMRS sampling differ from each other by either 0.01 or 0.02, so these two methods yield essentially the same samples, which match the distribution quite well. The random sampling scheme has much higher χ^2 values, indicating a poorer match to the full distribution. Experimental results indicate that Near-MMRS sampling accurately portrays the distribution of the original data in randomly projected lower dimensions, and takes significantly less time (around a second) than the MMRS sampling scheme.

Table 4.2: Average (20 trials) chi-square values and run-time (seconds) for each sampling scheme

Dataset	Random		MMRS		Near-MMRS	
	χ^2	Time	χ^2	Time	χ^2	Time
GM2 ($n = 200, k' = 10, q = 50$)	0.3	0.00	0.06	20.5	0.05	0.8
MNIST ($n = 300, k' = 30, q = 100$)	1.25	0.00	0.59	25.2	0.60	1.2
ACT ($n = 100, k' = 30, q = 50$)	6.71	0.01	4.98	115	4.50	0.2
FOREST ($n = 100, k' = 30, q = 5$)	1.86	0.01	1.18	4.4	1.19	0.9



(a) $I(D'^{*}_{d,1})$, PA = 32% (b) $I(D'^{*}_{d,2})$, PA = 50% (c) $I(D'^{*}_{d,3})$, PA = 73%

(d) $I(D'^{*}_{d,4})$, PA = 74% (e) $I(D'^{*}_{d,5})$, PA = 51% (f) $I(D'^{*}_{n,d})$, PA = 99.6%

Figure 4.3: iVAT images obtained using single distance matrices (a-e) and ensemble distance matrix (f).

4.5.4 Single Random Projection vs. Ensemble RP for iVAT Image

In this experiment, we compare the quality of iVAT images obtained by applying VAT/iVAT to distance matrices $\{D_{d,i}\}_{i=1}^{Q}$, for $Q = 5$, computed from single random projections to the iVAT image of the distance matrix $D_{n,d}$, computed from an ensemble of multiple random projections in our FensiVAT scheme. We also compare the PA values of NPR partitions U, obtained by SL partitioning based on the VAT reordered distance matrices $\{D_{d,i}\}_{i=1}^{Q}$ and $D_{n,d}$.

Figs. 4.3 (a)-(e) show the iVAT images, $\{I(D'^{*}_{d,i})\}_{i=1}^{Q}$, obtained using single random distance matrices $\{D_{d,i}\}_{i=1}^{Q}$, for the GM2 dataset, which has three (true) clusters. It is clear from these five iVAT images and corresponding PA values that the qualities of these images vary due to random nature of RP, and none of them strongly suggests that actual number of clusters in GM2 is $k = 3$. Fig. 4.3 (f) shows the iVAT image $I(D'^{*}_{n,d})$ based on the ensemble distance matrix $D_{n,d}$, which is obtained by aggregating the five distance matrices, $\{D_{d,i}\}_{i=1}^{Q}$ using FensiVAT ensemble scheme. View 4.3 (f) contains three dark blocks that are clearly visible along the diagonal in this image, and the PA value corresponding to this image is nearly perfect (99.6%).

Table 4.3 shows the PA values for the GM1 and GM2 datasets. The PA values corresponding to individual distance matrix for GM1 dataset show better accuracy than those obtained for the

Table 4.3: Average PA (%) values (20 trials) using single distance matrices, $\{D_{d,i}\}_{i=1}^{Q=5}$ and ensemble distance matrix $D_{n,d}$ for VAT/iVAT in FensiVAT.

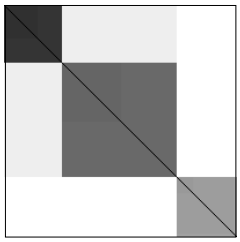
Distance Matrix	$D_{d,1}$	$D_{d,2}$	$D_{d,3}$	$D_{d,4}$	$D_{d,5}$	$D_{n,d}$
GM1 ($q = 20, k' = 10, n = 200$)	86	99	100	100	89	100
GM2 ($q = 50, k' = 10, n = 200$)	32	50	73	74	51	99.6

GM2 dataset. This is because the clusters in GM1 are much more separated than in the GM2 data, which has overlapping clusters. In contrast, the PA value corresponding to ensemble distance matrix is almost perfect for both datasets, which demonstrate the effectiveness of FensiVAT to obtain accurate NPR partitions with the ensemble approach.

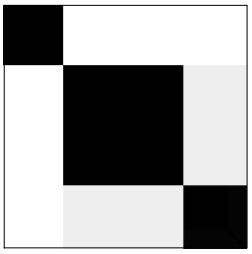
4.5.5 Cluster Assessment

The FensiVAT algorithm can be used to assess the potential number of clusters present in large, high dimensional data in significantly less time (discussed in Section 4.5.3) than clusiVAT, and with similar iVAT image quality. In this experiment, we compare iVAT images obtained using clusiVAT and FensiVAT for GM1 and GM2, and then we will showcase the ability of FensiVAT to correctly estimate the number of labeled subsets for the real datasets.

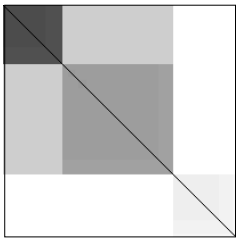
The iVAT images obtained using clusiVAT and FensiVAT for GM1 and GM2 are shown in Fig. 4.4 with corresponding algorithm parameter values. The ground truth partition of GM1 dataset has CS clusters because its DI $= 1.26\ (> 1)$. Figs. 4.4 (a) and (b) show that both clusiVAT and FensiVAT exhibit three (well-separated) dark blocks along the diagonal, suggesting that $k = 3$ for GM1. The ground truth partition for GM2 is non-CS because its DI is $0.66\ (< 1)$. Figs. 4.4 (c) and (d) show that FensiVAT produces three dark blocks along the diagonal for GM2, whereas clusiVAT shows three light blocks including many tiny blocks (data points) along the diagonal. Both views show the two darker blocks superimposed on a lighter dark block, which indicates that this data has a high degree of overlap. While clusiVAT and FensiVAT both show three blocks for GM1 and GM2, FensiVAT provides the more convincing assessment because of the sharper contrast between diagonal blocks and the background. Moreover, FensiVAT takes only a fraction of second for both datasets, whereas, clusiVAT takes around $20s$ to obtain poorer quality iVAT images. The sizes of the diagonal blocks in all four images show the relative size of each cluster accurately, which supports our claim that Near-MMRS sampling replicates (approximately) the



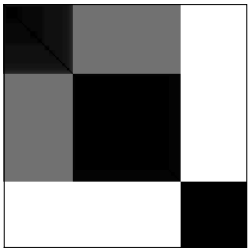
(a) GM1, ClusiVAT (Time=20.1s)



(b) GM1. FensiVAT (Time=0.35s)



(c) GM2, ClusiVAT (Time=21.3s)



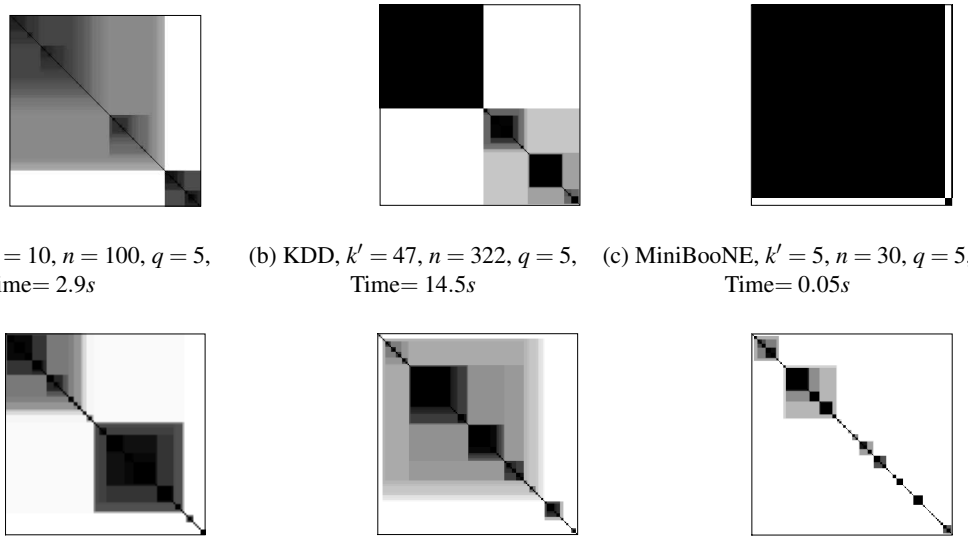
(d) GM2, FensiVAT (Time=0.84s)

Figure 4.4: ClusiVAT (a) and (c), and FensiVAT images (b) and (d) for GM1 and GM2. The parameters are $k' = 9$, $n = 205$ for GM1 and $k' = 12$, $n = 206$ for GM2 dataset. The downspace dimensions for FensiVAT are $q = 20$ for GM1 and $q = 50$ GM2.

same cluster distribution in the sample as the MMRS sampling used by clusiVAT.

The iVAT images of $D'^{*}_{n,d}$ for six real datasets are shown in Fig. 4.5 with corresponding FensiVAT algorithm parameter values. A (large) zoom is required to see all tiny dark blocks. The US Census 1990 data is an example of a real-world unlabeled, big data that is very large in both the number of records (N) and the number of attributes (p). Fig. 4.5 (a) shows the FensiVAT image for the US Census 1990 data. It can be seen that it shows two distinguished dark blocks along the diagonal in which the lower dark block comprises two small dark blocks. This suggests that there are two or three clusters in this data. Several previous researches [238, 239] also suggest $k = 2$ or 3 as the best estimate of the number of clusters for this dataset. FensiVAT just takes approximately 3 seconds to make this estimate.

The KDD CUP'99 is a big, labeled dataset that specifies attack types (normal or attack). It has 23 labeled subsets (22 simulated attacks and a normal subset), that fall into four main categories: DOS, R2L, U2R, and probing. The FensiVAT image of KDD-99 in View 4.5 (b) suggests 4 primary dark blocks and $18-20$ tiny dark blocks. The top left big dark block represents the 'smurf'



(a) Census, $k' = 10$, $n = 100$, $q = 5$, Time= $2.9s$

(b) KDD, $k' = 47$, $n = 322$, $q = 5$, Time= $14.5s$

(c) MiniBooNE, $k' = 5$, $n = 30$, $q = 5$, Time= $0.05s$

(d) FOREST, $k' = 20$, $n = 108$, $q = 5$, Time= $0.7s$

(e) MNIST, $k' = 28$, $n = 313$, $q = 100$, Time= $1.5s$

(f) ACT, $k' = 38$, $n = 314$, $q = 50$, Time= $0.5s$

Figure 4.5: iVAT images of $D'^{*}_{n,d}$ for each of the datasets obtained by FensiVAT algorithm.

attack (60% of the total dataset) in the DOS category. The right bottom dark block represents 'normal' data, which comprises approximately 18% of the data, and the middle dark block represents the 'neptune' attack in the DOS category, which comprises approximately 20% of the data. The remaining attacks are represented by $18-20$ tiny (hard to see) dark blocks along the diagonal.

The MiniBooNE data consists of $N = 130064$ instances divided into $36,499$ signal events of electron neutrinos and $93,565$ background events of muon neutrinos. The FensiVAT image (Fig. 4.5 (c)) for the MiniBooNE dataset shows two dark blocks along the diagonal. Although it shows two blocks, their sizes are not relative to the actual number of points in both classes. This is because some of the signal events are grouped with the background events due to similar attribute values. The FensiVAT image (Fig. 4.5 (d)) for FOREST dataset shows 2 big dark blocks on low resolution, $6-7$ dark blocks (of moderate size) in medium resolution, and $14-15$ tiny dark blocks on high-resolutions. The FOREST dataset has overlapping clusters due to heterogeneous features, so it has inter-mixed dark blocks along the diagonal. This is a case where the physically labeled subsets do NOT form well-defined clusters, at least not in the sense of SL distance, the basis of the iVAT image.

The MNIST dataset is a big, fairly dimensional ($p = 784$) dataset. This is also a challenging

Table 4.4: Average (20 trials) PA (%) values (with standard deviation) and run-time (in seconds) of FensiVAT for different RPs Q in ensemble step

	$Q=2$		$Q=3$		$Q=5$	
	PA	Time	PA	Time	PA	Time
GM1	99.9 ± 0.04	1.01	100 ± 0	1.11	100 ± 0	1.14
GM2	97 ± 5.8	1.68	99 ± 0.74	1.72	99.99 ± 0.1	1.89
	$Q=10$		$Q=20$		$Q=30$	
GM1	100 ± 0	1.38	100 ± 0	1.75	100 ± 0	2.11
GM2	100 ± 0	1.95	100 ± 0	2.33	100 ± 0	2.62

dataset for clustering because handwritten images of a single character can be executed in many often quite different ways, which causes overlapping clusters in the data. Fig. 4.5 (e) shows the FensiVAT image for the MNIST dataset, which indicates $10-12$ dark blocks. The ACT dataset is a high-dimensional ($p = 5625$), time-series dataset, which contains 19 activity types such as sitting, walking, jumping etc. The FensiVAT image (Fig. 4.5 (f)) shows $18-24$ tiny and middle size dark blocks along the diagonal. Thus, the FensiVAT recommendation for this dataset is to cluster it at every k from 18 to 24, and use a post-clustering validation method to select the "best" partition of the data. The MNIST and ACT datasets contain inter-mixed clusters, so we removed outliers using the strategy mentioned in Section 4.3 to improve the quality of the FensiVAT images. In summary, FensiVAT takes only about a second for most of the datasets ($14.5s$ maximum for KDD) to provide visual evidence about potential cluster structure, which makes it one of the best cluster assessment tools for big, high-dimensional dataset.

4.5.6 Synthetic Dataset for Different Numbers of RPs in Ensemble Step

In this experiment, we compare the PA of FensiVAT algorithm for different number of RPs (or distance matrices) used in the ensemble step in FensiVAT algorithm. For datasets having high diversity (overlapping clusters) like GM2, increasing Q in the ensemble method may be beneficial because there will probably be much more diversity in the random projections due to the mixed clusters in the upspace. Table 4.4 shows the average (20 trials) PA values with standard deviation and run-time of FensiVAT for $Q = 2, 3, 5, 10, 20$, and 30, for a fixed value of q for GM1 ($q = 20$) and GM2 ($q = 50$). FensiVAT achieves 100% accuracy for all $Q \geq 3$ on GM1, and for all $Q > 5$ on GM2, respectively. As expected, the accuracy of FensiVAT for GM2 increases as Q increases.

Table 4.5: Average (20 trials) PA (%) values (with standard deviation) and run-time (in seconds) of FensiVAT for different downspace dimensions, q.

	$q=5$		$q=10$		$q=20$	
	PA	Time	PA	Time	PA	Time
GM1	99.8 ± 0.2	0.80	99.9 ± 0.1	0.94	100 ± 0	1.11
GM2	92.4 ± 8.9	0.85	98.1 ± 1.2	0.96	99.2 ± 0.4	1.14
	$q=30$		$q=50$		$q=100$	
GM1	100 ± 0	1.33	100 ± 0	1.61	100 ± 0	2.51
GM2	99.9 ± 0.1	1.21	100 ± 0.0	1.53	100 ± 0.0	2.52

Furthermore, increasing the ensemble size has very little effect on FensiVAT CPU time.

4.5.7 Effect of Different Downspace Dimesions, q

In this experiment, we compare the performance of FensiVAT for different downspace dimensions $q = 5, 10, 20, 30, 50, 100$ for synthetic datasets GM1 and GM2. For the choices of $\varepsilon = \beta = 0.25$, and $n = 9162$, $q_0 = 1576$ (using (2.8)) and the probability of distance preservation = 0.9, so the chosen q values are well below the JL bound. These q values correspond to rogue random projections, which are chosen irrespective of ε and β. Table 4.5 shows the average (20 trials) PA values with standard deviation and run-time of FensiVAT for a fixed value of $Q\ (= 5)$. As expected, the accuracy of FensiVAT increases with increasing q and the performance becomes more stable (as standard deviation decreases). This is because higher q's correspond to more dimensions, so there is a better chance to preserve distances and lose less information under the projection.

The values in Table 4.5 show that even at $q = 5$ downspace dimensions, FensiVAT achieves very good clustering results (PA$>$ 99%) and achieves perfect (PA$=$ 100%) results with $q \geq 30$ for GM1. This is because the clusters in this dataset are (probably) well separated (recall that the ground truth partition of GM1 has CS clusters in the sense of Dunn). Thus, FensiVAT takes a fraction of a second to achieve perfect accuracy for the big, high-dimensional data GM1. For GM2, FensiVAT achieves near perfect results with $q \geq 50$. Unlike GM1, the performance of FensiVAT for GM2 is unstable for $q < 30$, most likely due to overlapping clusters in GM2 in the input space. Overall, FensiVAT achieves very good and stable clustering solutions even with rogue random projections.

4.5.8 Comparison of Different Clustering Methods

In this last experiment, we compare the performance of FensiVAT with nine existing approaches, which are best known for big and/or high-dimensional data clustering. The downspace dimensions for FensiVAT are chosen based on its best performance for each dataset. These values are shown in Figs. 4.4 and 4.5 for each dataset. Table 5.1 shows the comparison of FensiVAT to nine other algorithm based on the accuracy (PA) and run-time. Since US Census 1990 dataset is not labeled, we use DI as a measure of accuracy for different algorithms on the census data. The highest accuracy and smallest CPU time are shown in bold for each dataset.

For GM1, GM2, and ACT, FensiVAT outperforms the other approaches in terms of accuracy and CPU time. For GM2, FensiVAT, CLARA, CURE and RP-EN achieve perfect results (ave. PA= 100%), however, CLARA, CURE, and RP-EN take 16.6s, 511.9s and 183.9s respectively, whereas FensiVAT just takes 1.7s. For the FOREST, FensiVAT and clusiVAT achieve the highest PA (48.9%), however, FensiVAT takes the least time (2.17*s*). For MNIST, clusiVAT achieves the highest PA (50.1%) in 25.3s, whereas FensiVAT just takes 2.6*s* to achieve (nearly) similar accuracy (50.0%). For KDD, FensiVAT and clusiVAT achieve the highest accuracy, but FensiVAT is about 10 times faster than clusiVAT.

Table 4.6: Average PA (%) values (DI for US Census) and run-time (in seconds) for all the approaches on all the datasets.

Dataset /	GM1		GM2		KDD		FOREST		MiniBooNE		US Census		MNIST		ACT	
Methods	PA	Time	PA	Time	PA	Time	PA	Time	PA	Time	DI	Time	PA	Time	PA	Time
FensiVAT	**100**	**1.12**	**100**	**1.72**	**96.1**	88.4	**48.9**	2.17	71.9	**0.12**	0.10	6.9	50.0	**2.6**	**49.5**	**1.2**
clusiVAT	100	22.3	75.6	24.9	96.1	798.8	48.9	5.39	71.9	0.38	0.08	21	**50.1**	25.3	49.2	123.7
MBKM	90.1	2.07	89.9	2.03	74.3	**33.2**	34.2	**1.75**	71.9	0.12	0.08	**3.12**	44.3	3.38	47.8	7.8
CLARA	100	16.7	100	16.6	73.9	223.4	37.2	10.30	71.9	0.94	0.07	31.8	37.5	19.94	45.7	44.1
spkm	100	39.5	95.8	37.4	78.6	147.6	45.3	53.3	64.6	1.97	**0.12**	33	19.8	2296.8	12.5	3315.6
CURE	100	505.2	**100**	511.9	95.2	828.8	44.7	17.6	**76.8**	11.36	0.12	270	18.5	78.2	19.8	1871.3
RP-EN	100	31.8	100	183.9	95.7	52584	45.4	1596.3	76.8	30.3	0.12	475	26.5	95.8	26.5	205.9
PROCLUS	79.3	19.8	75.5	23.3	94.5	14346	45.1	2901.3	71.9	36.7	0.02	9162355	17.9	1185.7	17.8	12479.5
GARDENkm	65.8	1564	51.3	1652	94.1	326	38.2	44.5	71.9	340	0.01	3617	17.8	1133	18.5	3458
FastSpec	100	30.3	100	31.2	71.8	66.38	42.4	86	65.7	14.5	0.06	420	33.5	68.4	45.2	21.7

For GM1, GM2, MNIST, and ACT data, which have relatively high dimensions, FensiVAT outperforms the other clustering methods. It achieves the highest PA values in less than 2.6s (maximum 2.6s for MNIST), whereas, clusiVAT takes more than 20s for GM1 and MNIST, and hundreds of seconds for ACT. For all datasets except GM2, clusiVAT and FensiVAT achieve approximately equal PA values, but clusiVAT is $5-100$ times slower than FensiVAT. Surprisingly, FensiVAT achieves perfect results for GM2 dataset, whereas clusiVAT achieves 75.6%. This is

probably because of the robust distance matrix obtained using ensemble scheme in FensiVAT.

For the US Census data, spkm, CURE, and RP-EN achieve the highest DI value (0.12), which implies that their partitions are very slightly superior with regard to Dunn's validity measure. FensiVAT achieves the second highest DI value (0.1) and takes only 6.9s. ClusiVAT and MBKM achieve similar DI value, with MBKM the fastest algorithm. For KDD, which has millions of samples, FensiVAT achieves the best accuracy (96.1%) in just 88.4s, whereas the other approaches (except MBKM) take up to about $52,000$s average CPU time.

The MBKM approach is the second fastest method for all datasets except US Census, KDD, and FOREST (fastest), but, at the cost of lower clustering accuracy. CLARA is able to achieve the best PA for GM1 and GM2 dataset, but it is $10-60$ times slower than FensiVAT. The spkm algorithm achieves good accuracy for GM1, GM2, US Census, and FOREST, but performs poorly on MiniBoone, MNIST, and ACT. It is approximately $30-50$ times slower than FensiVAT for GM1, GM2, KDD, and FOREST dataset, and $1500-6000$ times slower for the high-dimensional datasets MNIST and ACT. CURE achieves comparable accuracy on all datasets except MNIST and ACT. It is likely that with many clusters, the randomly drawn samples used by CURE do not adequately capture the geometry of the big data. Therefore it suffers for the ACT and MNIST data, which appear to have many clusters. CURE is approximately $20-500$ times slower than FensiVAT for GM1, GM2, MNIST, and FOREST, and 3500 times slower for ACT dataset. PROCLUS is inaccurate for all datasets except KDD. The ensemble clustering method, RP-EN, achieves its highest PA values for GM1, GM2, and MiniBooNE and highest DI for US Census dataset, but at a time cost that is about $100-1500$ times higher than FensiVAT. RP-EN becomes intractable for KDD, and takes 52584s. Among high-dimensional clustering algorithms, RP-EN outperforms PROCLUS for all datasets except KDD. FensiVAT is about $4-3000$ times faster than GARDENkm. GARDENkm is relatively inaccurate for all the datasets except for KDD and MiniBooNE. FastSpec performs better than GARDENkm based on the clustering accuracy and CPU time. FastSpec achieves perfect results for GM1 and GM2, and exhibits comparable accuracy for all other datasets except KDD and MNIST. FensiVAT is faster ($20-120$ times) than FastSpec for all datasets except KDD. FastSpec is little faster than FensiVAT on KDD, but at the cost of clustering accuracy. To summarize, FensiVAT seems superior to the nine comparison algorithms for the datasets used in this paper.

4.6 Summary

This chapter introduced a new, fast clustering algorithm, called FensiVAT, which can be used to cluster large volumes of high-dimensional data. FensiVAT integrates a new random projection-based distance matrix ensemble method with Maximin and Random sampling (MMRS) and a visual assessment of cluster tendency method. We showed that the samples obtained using MMRS sampling in the downspace dimension (Near-MMRS sampling) retain the same geometry in the downspace as samples in the upspace. This enables us to use random projection effectively with MMRS sampling and in our ensemble method to reduce the computation time.

We demonstrated the superiority of our FensiVAT approach by comparing it with nine state-of-the-art approaches on two Gaussian mixture datasets and six real datasets which have both large sample size and high dimensions. Experimental results on eight large, high-dimensional datasets show that FensiVAT almost always outperforms the other nine approaches. FensiVAT is an order of magnitude faster than clusiVAT, and several order of magnitudes faster than the other nine approaches (except MBKM), without compromising accuracy.

Chapter 5

Approximating Dunn's Cluster Validity Indices for Big Data

This chapter presents six approximation algorithms including two incremental approaches to compute Dunn's cluster validity index for big data. Four methods are based on MMRS sampling, and two are based on unsupervised training of one class support vector machines. Numerical experiments on seven real and synthetic datasets assert that MMRS methods provide accurate DI estimates, and represent a speedup on the order of 1000:1.

5.1 Introduction

As discussed in Chapter 2 (Section 2.4.1), there has been a considerable amount of work done to address clustering tendency assessment and clustering problem for big data. However, there is very little work done on cluster validity for big data. CVIs that use only membership values (U) are easy to compute for big data because the output partition (U) is not usually big as compared to the input (big) dataset X. However, most of these CVIs have a monotonic dependency on the number of clusters. They also lack the direct connection with the geometry of the data (X) because they do not use data itself. It is a well-known fact that a better definition of validity index should always consider the geometry of data. However, the implementation of such CVIs, that consider both partition U and dataset X, is often very computationally expensive, especially when the number of clusters and number of objects in the dataset grows very large [62].

This chapter focuses on Dunn's internal CVI which uses both the data and the partition resulting from any hard clustering algorithm. *Dunn's index* (DI) [48] is one of the most popular internal CVIs finding its way into many cluster validity studies [62, 165, 167]. DI has been related

to visual assessment methods such as VAT and iVAT algorithms in [133]. DI provides a measure of contrast between the blocks on the VAT/iVAT image diagonal and the background regions. A recent study using neuron spike data relates DI to both iVAT and SL [240]. Since DI has quadratic time complexity $O(pN^2)$, their computation is infeasible for big datasets (X with big N).

To address this issue, this chapter presents six methods for approximating DI for big data. The first four proposed methods viz., αMMRS, αnMMRS, *i*MMRS, and *i*nMMRS are based on the *Maximin Random Sampling* (MMRS) rule [83], which identifies *approximate boundary points* in each cluster. The *i*MMRS and *in*MMRS schemes are incremental methods, which produce an optimal number of (boundary) points to compute the approximate Dunn's index. Two additional methods are presented here to compute approximate DI, that are based on the unsupervised training of *one class support vector machines* (OCSVM) [84, 85]. Our experiments show that computing approximations to DI with Maximin skeleton based methods are both tractable and accurate.

5.2 Related Work

Many books on cluster analysis contain at least one chapter on cluster validity [41, 105, 241, 242]. Surveys on crisp CVIs that compare various validation schemes in one way or another began to appear in the 1980s [166]. Halkidi *et al.* [62] present a review of many popular clustering validity measures along with numerical examples of experimental evaluation. Milligan and Cooper [243] compared 30 validity tests (which they called "stopping rules") using partitions generated by four hierarchical clustering methods, and their paper is considered the classic reference on "best-k" studies of internal CVIs. Gurrutxaga *et al.* [244] present a very thorough critique of Milligan and Cooper's "best-k" methodology.

Dimitriadou *et al.* [245] presented a nicely written survey of 15 internal CVIs in 2002. Arbelaitz *et al.* [167] published an extensive comparison of 30 internal CVIs for crisp c-partitions that channels the Milligan-Cooper style. Three crisp clustering algorithms were used to populate candidate partitions in their study. Seventeen goodness of fit functions which can be regarded as internal CVIs for probabilistic (Gaussian) clusters generated by the *Expectation-Maximization* (EM) algorithm [41] for *Gaussian Mixture Decomposition* (GMD) are compared to both crisp and fuzzy CVIs in [246]. Nguyen *et al.* [168] presented a "best-k" study of similarity measures and

distance based functions that compare pairs of crisp partitions using external information-theoretic CVIs. They identify a total of 26 measures that are subdivided into 10 similarity measures and 16 distance measures.

Very limited research literature studies CVIs for big data. Tlili *et al.* [179] proposed a fuzzy version of the *Davies Bouldin Index* (DBI) for big data clustering. Since, the DBI uses both the output partition and the data itself, it still has the higher computational complexity for big data. Although this method was proposed for big data, the datasets used to illustrate it (largest one $N = 10,000$ and $p = 85$) are not considered large in today's computing environment. Moreover, they did not report the computation time in their experiments. Another work [180] implemented DI and Silhouette on a Spark platform to deal with big data.

Chapter 2 (Section 2.4.2) briefly discussed DI and its limitation for big data. Below, we explain DI and its generalized indices, called GDIs, in detail before presenting their approximation methods.

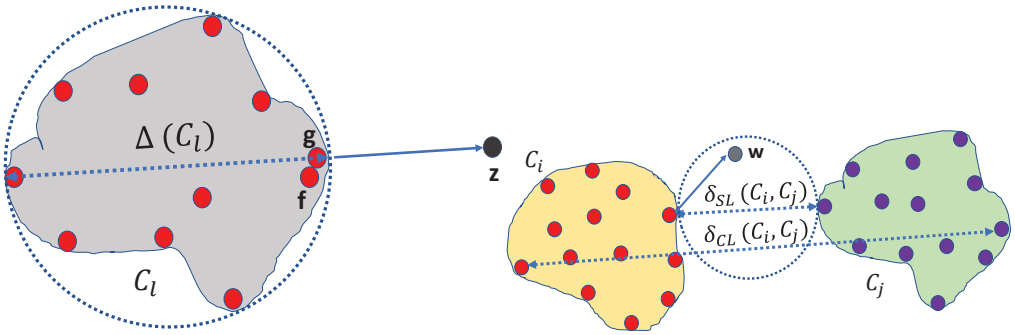
5.3 Dunn's Index (DI)

Let $X = \{\mathbf{x}_1, .., \mathbf{x}_N\} \subset \mathbb{R}^p$ be a set of N feature vectors in p-dimensional space. A crisp partition U of X can also be represented in terms of the k disjoint subsets $\{C_i\}$ written as $U \leftrightarrow X = \bigcup_{i=1}^{k} C_i;\ C_i \cap C_j = \emptyset$ for $i \neq j$. The cardinalities (or sizes) of the ith clusters is given as $|C_i|$ so that $\sum_{i=1}^{k} |C_i| = N$.

Dunn's index is based on the geometrical premise that "good" sets of clusters are compact (dense about their means) and well separated from each other. To quantify this index, Dunn let C_i and C_j be non-empty subsets of $\mathbb{R}^p$, and let $d : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}^+$ be any metric on $\mathbb{R}^p \times \mathbb{R}^p$. Dunn based his index on the standard definitions of the diameter Δ of C_l and the set distance δ between C_i and C_j.

$$\Delta(C_l|d) = \underbrace{\max}_{\mathbf{x},\mathbf{y} \in C_l} \{d(\mathbf{x}, \mathbf{y})\}, \tag{5.1}$$

$$\delta(C_i, C_j|d) = \delta_{SL}(C_i, C_j|d) = \underbrace{\min}_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_j}} \{d(\mathbf{x}, \mathbf{y})\}. \tag{5.2}$$



(a) The classical diameter $\Delta(C_l)$ of C_l

(b) The classical (SL) distance $\delta_{SL}(C_i,C_j)$ and complete linkage (CL) distance $\delta_{CL}(C_i,C_j)$ between C_i and C_j

Figure 5.1: Set distance and diameter with respect to $d = d_E$.

For any partition, $U \leftrightarrow X = \{C_1 \cup ...C_i \cup ...C_k\}$, Dunn defined the separation index of U as follows:

$$\mathscr{V}_{DI}(U|d,\delta,\Delta) = \frac{\underbrace{\min}_{1\le i\le k}\left\{\underbrace{\min}_{1\le j\ne i\le k}\{\delta(C_i,C_j|d)\}\right\}}{\underbrace{\max}_{1\le l\le k}\{\Delta(C_l|d)\}}. \tag{5.3}$$

The notation in (5.3) indicates that Dunn's index requires choices for three functions, $\{d,\delta,\Delta\}$; the pair wise distance metric d, the set distance δ, and the diameter function Δ. Dunn's definition at (5.3) and hence, in (5.1) or (5.2) as well, are based on an arbitrary metric d on any real vector space. The notation $(*|d)$ for δ and Δ indicate that they both depend only on d.

The set distance shown in equation (5.2) is used by the hierarchical *single linkage* (SL) clustering algorithm [41, 105, 241, 242], and is often called the single linkage set distance for this reason. Fig. 5.1 depicts the classical meaning of equation (5.1) for Δ and (5.2) for δ when d is chosen as *Euclidean distance*, $d = d_E$. The outlier point **z** in Fig. 5.1 (a) and inlier point **w** in Fig. 5.1 (b) will be discussed shortly.

Following many subsequent authors, we refer to $\mathscr{V}_{DI}$ at (5.3) as *Dunn's index* (DI). The most common metric for d in the numerator and denominator of $\mathscr{V}_{DI}$ i.e., in (5.1) and (5.2), is d_E but there are many other choices. The diameter $\Delta(C_l|d)$ in (5.1) which appears in the denominator of (5.3) is a measure of scatter volume for cluster C_l. Compact clusters will have smaller diameters than ones that are more dispersed about their mean vectors. A set of clusters is relatively compact when the largest of its k diameters and, hence the denominator in (5.3), is small.

The quantity $\delta(C_i, C_j|d)$ that appears in the numerator of $\mathscr{V}_{DI}$ is the SL set distance with respect to d at equation (5.2) between pairs of crisp clusters in U. Hence, the larger $\delta(C_i, C_j|d)$ is, the better separated are C_i and C_j. Taking the double minimum in the numerator identifies the pair of clusters that are *least* well separated. As the k clusters become better separated, the numerator in (5.3) grows.

Thus, the geometric objective of DI is to maximize inter-cluster distances (big numerators) while minimizing intra-cluster distances (small denominators). Large values of $\mathscr{V}_{DI}$ intuitively correspond to better clusters in the sense of DI. The partition U^* that maximizes $\mathscr{V}_{DI}$ over a set of candidate partitions is taken as the (DI) optimal set of clusters. Consequently, $\mathscr{V}_{DI}$ is called a max-optimal internal CVI.

The range of $\mathscr{V}_{DI}$ is $(0, \infty)$, and $\mathscr{V}_{DI}$ is undefined when $k = 1$ ($U_{1\times N} = \mathbf{1}_N$) and $k = N$ ($U_{N\times N} = I_N$). Dunn called a partition $U \in M_{hkN}$ *compact and separated* (CS) relative to d if and only if the following property is satisfied: for all s, q and r with $q \neq r$, any pair of points $\mathbf{x}, \mathbf{y} \in C_s$ are closer together (with respect to d) than any pair $(\mathbf{u}, \mathbf{v})$ with $\mathbf{u} \in C_q$ and $\mathbf{v} \in C_r$. Dunn [48] proved that X can be clustered into a CS k-partition with respect to d if and only if there is a $U \in M_{hkN}$ for which the index is greater than 1.

Theorem 5.1. *$X \subset \mathbb{R}^p$ has a CS k-partition U^* with respect to $d \Longleftrightarrow U^* = \underbrace{\max}_{U \in M_{hkN}} \{\mathscr{V}_{DI}(U|d)\} > 1$ [48].*

This is a nice theoretical result, but in practice, it is quite difficult to verify that a *given input dataset* can be partitioned into CS clusters, because M_{hkN} is finite, but very, very large. The exact cardinality of M_{hkN} is $|M_{hkN}| = \left(\frac{1}{k!}\right) \sum_{j=1}^{k} \binom{k}{r}(-1)^{k-j} j^N$. For $k << N$, the last term dominates this sum, which yields the approximation $|M_{hkN}| \approx k^N/k!$. Consequently, computing DI over all of M_{hkN} is impractical for all but trivial values of k and N. The value of $\mathscr{V}_{DI}(U|d)$ for a given U, however, is easily computed with (5.3) when N is not too large, and if it happens to be greater than 1, its clusters are said to be compact and separated in the sense of Dunn.

DI at (5.3) has a well known flaw, viz., sensitivity to anomalies, which can render it ineffective for partitions of data that have clusters with outliers and/or inliers. This is easy to see. Return to Fig. 5.1 (a) and imagine adding the single outlier point $\mathbf{z}$ to C_l . As shown, this one point can double the diameter of C_l, $\Delta(C_l \cup \{\mathbf{z}\}) = 2\Delta(C_l)$. Similarly, adding the inlier point $\mathbf{w}$ to C_j

in Fig. 5.1 (b) will significantly decrease the distance between C_i and C_j. Thus, a single anomaly can alter the numerator and/or denominator of Dunn's index by orders of magnitude.

To address this issue, a family of 18 *generalized Dunn's indices* (GDIs) were defined and analyzed in [169]. Under the same conditions as in (5.3), these indices take the general form

$$\mathscr{V}_{GDI}(U|d,\delta_a,\Delta_b)) = \underbrace{\min}_{1\le i\le k}\left\{\underbrace{\min}_{\substack{1\le j\le k\\ j\neq i}}\left\{\frac{\delta_a(C_i,C_j)}{\underbrace{\max}_{1\le l\le k}\{\Delta_b(C_l)\}}\right\}\right\}, \tag{5.4}$$

where $a \in \{1,...,6\}, b \in \{1,\ 2,\ 3\}$. For brevity we write (5.4) as $\mathscr{V}_{ab}$, where a and b define the choice of set distance (numerator) and diameter (denominator), respectively, from GDIs (5.4). Equation (5.4) reduces to the original DI at (5.3) when $a = b = 1$, i.e., $\mathscr{V}_{11} = \mathscr{V}_{DI}$. Below, we list the equations that result in the 17 GDIs.

Equations (5.5)-(5.11) define the six numerators alluded to by equation (5.4). The numerator for $a = 2$ is the *complete linkage* (CL) distance between C_i and C_j, and the choice $a = 3$ corresponds to the *average linkage* (AL) distance.

$$\delta_2(C_i,C_j) = \delta_{CL}(C_i,C_j) = \underbrace{\max}_{\substack{\mathbf{x}\in C_i\\ \mathbf{y}\in C_j}}\{d(\mathbf{x},\mathbf{y})\} \tag{5.5}$$

$$\delta_3(C_i,C_j) = \delta_{AL}(C_i,C_j) = \frac{1}{|C_i||C_j|}\sum_{\substack{\mathbf{x}\in C_i\\ \mathbf{y}\in C_j}} d(\mathbf{x},\mathbf{y}). \tag{5.6}$$

The choice of $a = 4$ and 5 corresponding to δ_4 and δ_5, respectively, are the set distances that incorporate the averaging concept of δ_3. The sixth set distance ($a = 6$) is based on the Hausdorff metric.

$$\delta_4(C_i,C_j) = d(\mathbf{v}_i,\mathbf{v}_j), \tag{5.7}$$

$$\text{where} \quad \mathbf{v}_i = \frac{1}{|C_i|}\sum_{\mathbf{x}\in C_i}\mathbf{x} \ \text{ and } \ \mathbf{v}_j = \frac{1}{|C_j|}\sum_{\mathbf{y}\in C_j}\mathbf{y}$$

$$\delta_5(C_i,C_j) = \frac{1}{|C_i|+|C_j|}\left(\sum_{\mathbf{x}\in C_i} d(\mathbf{x},\mathbf{v}_i) + \sum_{\mathbf{y}\in C_j} d(\mathbf{y},\mathbf{v}_j)\right) \tag{5.8}$$

$$\begin{aligned}\delta_6(C_i,C_j) &= \delta_{Hausdorff}(C_i,C_j)\\ &= max\{\delta(C_i,C_j),\delta(C_j,C_i)\}\end{aligned} \tag{5.9}$$

where

$$\delta(C_i,C_j) = \underbrace{\max}_{\mathbf{x}\in C_i}\{\underbrace{\min}_{\mathbf{y}\in C_j}\{d(\mathbf{x},\mathbf{y})\}\} \tag{5.10}$$

$$\delta(C_j,C_i) = \underbrace{\max}_{\mathbf{y}\in C_j}\{\underbrace{\min}_{\mathbf{x}\in C_i}\{d(\mathbf{x},\mathbf{y})\}\} \tag{5.11}$$

If $b = 1$, $\mathscr{V}_{21}(U|d)$ and $\mathscr{V}_{31}(U|d)$ will also be sensitive to outliers and inliers, so the authors of [169] proposed two other diameters for the denominator of (5.4). The choice $b = 2$ corresponds to the average distance between the data points in cluster C_l, and the choice $b = 3$ is the average distance between the points in C_l and its cluster center, $\bar{\mathbf{v}}$.

$$\Delta_2(C_l) = \frac{1}{|C_l|\cdot(|C_l|-1)}\sum_{\substack{\mathbf{x},\mathbf{y}\in C_l\\ \mathbf{x}\neq\mathbf{y}}} d(\mathbf{x},\mathbf{y}) \tag{5.12}$$

$$\Delta_3(C_l) = 2\left(\frac{\sum_{\mathbf{x}\in C_l} d(\mathbf{x},\bar{\mathbf{v}})}{|C_l|}\right), \text{where} \quad \bar{\mathbf{v}} = \frac{1}{|C_l|}\sum_{\mathbf{x}\in C_l}\mathbf{x} \tag{5.13}$$

The generalized Dunn's index $\mathscr{V}_{33}(U|d)$ has done well in several comparative studies [167, 169, 247]. For example, the study in [247] ranked $\mathscr{V}_{33}$ 8-th among 40 competing internal CVIs, scoring 389 hits in 432 clustering scenarios, whereas $\mathscr{V}_{11}$ was ranked 29-th in this same study. The datasets used in [247] were relatively small.

The terms "good CVIs" and "bad CVIs" are oxymorons in cluster validity and it can not be asserted that DI ($\mathscr{V}_{11}$) is either a good one or bad one. Despite its sensitivity to noisy points, DI provides a rich and very general structure for defining cluster validity indices for different types of clusters. DI finds its way into many cluster validity studies [62, 165, 167]. And, various commercial software packages such as the spike extraction and sorting software (Offline Sorter, Plexon Inc, Dallas, TX) report DI as part of their default cluster validity statistics [248].

Our approximations of Dunn's index are based on Maximin Random Sampling (MMRS), the topic we turn to next.

5.4 The Maximin Random Sampling (MMRS)

Although MMRS sampling is well explained in Chapter 2 (Section 2.2.3), we briefly discuss it here for completeness. MMRS is a combination of Maximin and Random sampling. Maximin (MM) sampling selects a few samples far from each other so that they represent diverse regions of the input space. These samples are called *distinguished* objects or MM samples. Pseudocode for the MMRS algorithm is shown in Algorithm 3. Hathaway *et al.* [47] proved a proposition (Proposition 2.1) that relates the quality of the objects selected by MMRS to *compact and separated* (CS) clusters as defined by Dunn. According to this proposition, if dataset X has k CS clusters and $k' \geq k$, then Maximin step of MMRS sampling will always select at least one object (MM sample) from each cluster, where k' is the number of MM samples.

The practical implication stemming from this result is that the MMRS procedure *probably* finds points that are fairly well distributed across even non-CS clusters when there are many points in each subset. The efficacy of this assumption has been well tested in a variety of big data application domains. For example, MMRS underlies the success of *scalable visual assessment of tendency* (sVAT) [47] for building approximate cluster heat maps in big data. And, clusiVAT [135] and FensiVAT (presented in Chapter 4) use MMRS to scale a generalization of the single linkage clustering algorithm up to big data of arbitrary size in N and p. However, our interest in this chapter lies in another direction.

Fig 5.1 shows that $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$ both depend on the extremal (or "boundary") points in each cluster. We do not define the term boundary point exactly, but it is clear that for $\Delta_1(C_l)$ we want the points in each C_l that are furthest from each other. And for $\delta_1(C_i, C_j)$ (or $\delta_2(C_i, C_j)$), we want the points in the two sets that are closest (or furthest) away from each other. We loosely call such points boundary points.

So, we want to find a set of *approximate boundary points*, $\delta B(C_i)$, in each cluster C_i of the partition $U \leftrightarrow X = \bigcup_{i=1}^{k} C_i$. If the extremal points in C_i are contained in $\delta B(C_i)$, computing Dunn's index on the sub-partition (of reduced size) $\delta B(X) = \bigcup_{i=1}^{k} \delta B(C_i)$ should provide a good estimate

of the literal Dunn's index that we would get by computing it on all of the points in each cluster. This affords a way to estimate Dunn's index and its generalizations on intractably large partitions of big data.

Since the GDIs in [169] are designed to overcome the sensitivity of Dunn's index to extremal points in the data, we do not expect the methods in this chapter to provide as good approximations to their literal values as we will get for $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$.

5.5 Approximating Dunn's index

This section describes six algorithms that build a reduced partition $\delta B(X) = \bigcup_{i=1}^{k} \delta B(C_i)$ of $U \leftrightarrow X = \bigcup_{i=1}^{k} C_i \in M_{hkN}$. The first four algorithms are based on the MMRS and the last two are based on boundary (support) vector algorithms. All six algorithms share these *common inputs*:

1. $X = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\} \in \mathbb{R}^p$,
2. a $k \times N$ partition $U \leftrightarrow X = \bigcup_{i=1}^{k} C_i \in M_{hkN}$,
3. a distance metric d for equations (5.3) to (5.13), and for nearest neighbour rules.

5.5.1 The MMRS algorithms

MM (Step 1 of MMRS (Algorithm 3)) finds the distinguished points which are furthest from each other. This is the basic motivation for using the MMRS to approximate the boundary points of each cluster. These furthest points, roughly called "boundary points", from each cluster can be used to estimate $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$. We also expect the MMRS to provide a good skeleton because of the last sentence in Casey and Nagy's MMRS description [101] (Chapter 2): "*These initial cluster centers are well scattered over the sample space.*". Hathaway *et al.* [47] proved a theorem (Theorem 2.1) which says that, for CS datasets, the proportion of objects in each cluster C_i in the MMRS sample equals the proportion of the objects from the same cluster in the original data for $i = 1, 2, .., k$. This means that the MMRS algorithm not only finds the boundary points, but it also selects well-distributed points within each cluster, which can be used to estimate $\mathscr{V}_{31}$ (using AL distance).

Instead of applying MMRS to the entire input data as in [47], we apply MMRS to each cluster in a k-partition U of data X. Applying MMRS to each C_i should find most of the points that are far apart from each other within the cluster C_i in the sense of the metric d used in equations (5.1) to (5.6). Algorithms 8 and 9 can take inputs $a \in \{1,...,6\}, b \in \{1,\ 2,\ 3\}$ as described at (5.4). Algorithms 10 and 11 are restricted to $a = 1,2; b = 1$.

5.5.1.1 αMMRS

The pseudocode for the αMMRS algorithm is shown in Algorithm 8. The parameter α dictates the choice of the first $\lceil \alpha N \rceil$ MM points from each cluster. When $\alpha = 1$, all the points in X are used, and algorithm αMMRS computes literal Dunn's indices. Can unequal cluster sizes $\{N_i\}$ bias the approximation of X by X_α? To address this question, we conducted an experiment, which studies whether a fixed fraction α of $|X| = N$ (αMMRS) yields a different approximation to the DI than applying the fraction α to each C_i individually. We call latter approach $\alpha\{i\}$MMRS. Experiment 1 in Section 5.6.3.3 shows that estimates of $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$ are not biased by unbalanced cluster sizes, but $\mathscr{V}_{31}$ is affected. This is probably because $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$ only relies on border points, whereas, in $\mathscr{V}_{31}$, average distance changes with each extracted data point.

Algorithm 8 The αMMRS algorithm

Input: α- a fraction of N, $0 < \alpha \leq 1$;
a, b- inputs to compute GDI $\mathscr{V}_{ab}$
1: $\forall_{\mathbf{i}}$: Extract $k' = \lceil \alpha N \rceil$ MM points $\delta B_\alpha(C_i)$ from cluster C_i with Step 1 of Algorithm 3.
2: Form the reduced partition $U_\alpha \leftrightarrow X_\alpha = \bigcup_{i=1}^{k} \delta B_\alpha(C_i)$
3: $\forall_{\mathbf{a,b}}$: Compute $\mathscr{V}_{ab}(U_\alpha|d)$
Output: $\mathscr{V}_{ab}(U_\alpha|d)$

5.5.1.2 αnMMRS

Steponavičė *et al.* [104] showed that *Maximin sampling tends to select decision vectors that are located near the boundary of the decision space.*. In such cases, MM may require a relatively large number of samples to extract exact boundary points. This is probably because once the MM sampling extracts a point, say point **f** (see Fig. 5.1 (a)), which is near to the border but not exactly on the borderline of a cluster, it may take quite a few more samples before MM extracts

the borderline point **g**, which is near **f**, due to the MM property of picking points furthest from all previous points.

To address this issue, the neighbourhood part of the MMRS algorithm is utilized. αnMMRS (Algorithm 9) adds some points to X_α in the neighborhood of each αMMRS sample (using Step 3 of Algorithm 3), resulting in $X_{\alpha n}$. The rationale is that local neighbors of "nearly extremal" points in X_α should contain points essential for a better approximation of $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$. Evidently $X_\alpha \subset X_{\alpha n}$.

Algorithm 9 The αnMMRS algorithm

Input: α- a fraction of N, $0 < \alpha \leq 1$;
n_{id}- the number of local neighbourhood samples of MM points for each cluster, and $\bigcup_{i=1}^{k} n_{id} = n$;
a, b- inputs to compute GDI $\mathscr{V}_{ab}$

1: **for i** $\leftarrow$ **1 to k do**
2: Extract $k' = \lceil \alpha N \rceil$ MM points $\delta B_\alpha(C_i)$ from C_i with Step 1 of Algorithm 3
3: Extract n_t neighbors of each MM point with Step 3 of Algorithm 3, resulting a total of $n_{id} = \bigcup_{t=1}^{k'} n_t$ neighbour points
4: $\delta B_{\alpha n}(C_i) = \delta B_\alpha(C_i) \cup n_{id}$
5: **end for**
6: Form the reduced partition $U_{\alpha n} \leftrightarrow X_{\alpha n} = \bigcup_{i=1}^{k} \delta B_{\alpha n}(C_i)$
7: $\forall_{\mathbf{a},\mathbf{b}}$: Compute $\mathscr{V}_{ab}(U_{\alpha n}|d)$

Output: $\mathscr{V}_{ab}(U_{\alpha n}|d)$

5.5.1.3 *i*MMRS

Instead of generating all k' MM points with αMMRS or αnMMRS and then computing approximate Dunn's indices, the *i*MMRS algorithm tries to estimate an optimal number to points to approximate DI. The intuition behind this algorithm is that inclusion of a new MM data point to any cluster C_i will not increase the value of Dunn's original index for X. We explain this idea for $\mathscr{V}_{11}$ based on the changes to the diameter (denominator) and the SL set distance between two clusters (numerator) using Fig. 5.1, as follows:

First, we discuss the effect of a new data point on cluster diameter (Eq (5.1)). The maximum intra-cluster distance is used as cluster diameter in the denominator of DI (Eq (5.3)). As can be seen in Fig. 5.1 (a), a new data point can lie either within, on, or outside the hypersphere with diameter $\Delta(C_l)$ of an existing set of data points (shown in red). If a new data point lies within or

on the boundary of the hypersphere, the maximum within cluster distance is unchanged, so the diameter in Eq (5.1) will be unchanged. If the new data point (say **z**) lies outside the hypersphere, the maximum distance within the cluster will increase, so the diameter will also increase. In all three cases, the denominator of DI will not decrease.

How will a new data point affect the SL set distance between two clusters in Eq (5.2)? The minimum of inter-cluster distances is used to compute the numerator in DI. Consider the hypersphere shown in Fig. 5.1 (b) with diameter $\delta_{SL}(C_i, C_j)$ which is in between ("tangent" to) the current pair of minimal points. A new data point can lie either within, on or outside this hypersphere. So, when a new data point falls on or outside the hypersphere, the minimum distance between two clusters is unchanged. If the new data point falls within the hypersphere, the inter-cluster distance will decrease. Therefore, in all cases, the numerator of DI will not increase. In summary, when a new data point is added to a cluster, DI ($\mathscr{V}_{11}$) will either be the same or it will decrease.

Algorithm 10 The *i*MMRS algorithm

Input: T-Loop Limit;
ε- termination threshold, $\varepsilon > 0$;
a, b- inputs to compute GDI $\mathscr{V}_{ab}$

for i $\leftarrow$ **1 to k do**
  Draw $\mathbf{x_{m_{0,i}}} \in C_i$, ($m_0$ = a random point in C_i)
  Find first MM point $\mathbf{x_{m_{1,i}}}$
  $\delta B_{1,i}(C_i) = \mathbf{x_{m_{1,i}}}$; delete $\mathbf{x_{m_{0,i}}}$;
  Find second MM point $\mathbf{x_{m_{2,i}}}$;
  $\delta B_{2,i}(C_i) = \{\mathbf{x_{m_{1,i}}}\} \cup \{\mathbf{x_{m_{2,i}}}\}$
end for
$\forall_{\mathbf{a},\mathbf{b}} : \mathscr{V}_{ab}(U_2|d)$
for j $\leftarrow$ **3 to T do**
  for i $\leftarrow$ **1 to c do**
    Find MM point $\mathbf{x_{m_{j,i}}} \in C_i$
    $\delta B_{j,i}(C_i) = \delta B_{j-1,i}(C_i) \cup \{\mathbf{x_{m_{j,i}}}\}$
  end for
  Form $U_j = \bigcup_{i=1}^{k} \delta B_{j,i}(C_i)$
  Compute: $\mathscr{V}_{ab}(U_j|d)$; $\mathscr{V}_{ab}(U_{j-1}|d)$
  if $\left|\mathscr{V}_{ab}(U_j|d) - \mathscr{V}_{ab}(U_{j-1}|d)\right| \leq \varepsilon$ **then**
    $\mathscr{V}_{ab}(U_j|d)$
    break;
  else
    next j
  end if
end for

Output: $\forall_{\mathbf{a},\mathbf{b}} : \mathscr{V}_{ab}(U_j|d) : j$

Now, we discuss the *i*MMRS algorithm to approximate DI. Initially, the approximate boundary points, $\partial B(C_i)$ for each cluster C_i is empty, so $\partial B(X) = \bigcup_{i=1}^{k} \partial B(C_i) = \emptyset$. This method starts with k random initial points $\{\mathbf{x}_{m_{0,i}}\}$, one from each cluster C_i. Then, based on these k initial points, the first MM, $\{\mathbf{x}_{m_{1,i}}\}$, is extracted from each cluster and added to $\partial B(C_i)$. Since at least two points per cluster are required to compute the diameter of each cluster, so $\{\mathbf{x}_{m_{2,i}}\}$, the second MM points (furthest from $\{\mathbf{x}_{m_{1,i}}\}$) are extracted from each cluster and added to $\partial B(C_i)$. Then, we can compute the two MM points estimate of $\mathscr{V}_{ab}$ using $\partial B(X)$. Then, $\{\mathbf{x}_{m_{3,i}}\}$, the next (third) MM points are added to each subset resulting in a 3-point estimate. This procedure continues until successive estimates of $\mathscr{V}_{11}$ are close. This amounts to progressive MM sampling with termination criteria as shown at line 16 of Algorithm 10. The objective is to obtain a reasonable estimate of $\mathscr{V}_{11}$ with a minimal number of MM points per cluster. Since this method is *incremental*, we call it the *i*MMRS approximation (Algorithm 10).

5.5.1.4 *i*nMMRS

Our initial experiments with *i*MMRS showed that successive local estimates of Dunn's index were often small, but the overall approximation descended in staircase fashion, so *i*MMRS often terminated before reaching a good approximation to the literal value of Dunn's index. This led us to the modification (Algorithm 11), which combines ideas from the neighborhood part of αnMMRS with the incremental part of *i*MMRS algorithm. Similar to αnMMRS, *i*nMMRS algorithm add some points in the neighborhood ($\{n_{t,i}\}$) of each MM point for fast convergence. We will discuss the termination issue further in Experiment 4. In contrast with αMMRS and αnMMRS, *i*MMRS and *i*nMMRS do not require the parameter α.

Algorithms 10 and 11 are incremental methods, in which approximate boundary points are extracted incrementally one by one from each cluster using Maximin algorithm. Once the extracted approximate boundary points contain the closest (for SL distance) and furthest elements (for CL distance) from each cluster, the SL ($a = 1, b = 1$) and CL distance ($a = 2$) does not change, and consequently, DI value also does not change. Once the optimal DI value is achieved, the algorithm terminates. Therefore, Algorithms 10 and 11 are restricted to $a = 1, 2$ and $b = 1$. Unlike $\mathscr{V}_{11}$, $\mathscr{V}_{21}$ does not decrease or increase monotonically with an addition of an MMRS point, however, it achieves a steady value after adding enough MMRS points. The termination may not be achieved

Algorithm 11 The *i*nMMRS algorithm

Input: T-Loop Limit;
ε- termination threshold, $\varepsilon > 0$;
a, b- inputs to compute GDI $\mathscr{V}_{ab}$

for i $\leftarrow$ **1 to k do**
  Draw $\mathbf{x_{m_{0,i}}} \in C_i$, ($m_0$ = a random point in C_i)
  Find first MM point $\mathbf{x_{m_{1,i}}}$
  $\delta B_{1,i}(C_i) = \mathbf{x_{m_{1,i}}}$; delete $\mathbf{x_{m_{0,i}}}$;
  get $n_i = 1$ local neighbors, $n_{1,id}$, of $\mathbf{x_{m_{1,i}}}$
end for
$U_1 = \bigcup_{i=1}^{k} \delta B_1(C_i) = \bigcup_{i=1}^{k} [\mathbf{x_{m_{1,i}}} \cup n_{1,id}]$
$\forall_{\mathbf{a},\mathbf{b}} : \mathscr{V}_{ab}(U_1|d)$
for i $\leftarrow$ **1 to k do**
  Find second MM point $\mathbf{x_{m_{2,i}}}$
  get $n_i = 2$ local neighbors $n_{2,id}$ of $\{\mathbf{x_{m_{1,i}}}\} \cup \{\mathbf{x_{m_{2,i}}}\}$
end for
$U_2 = \bigcup_{i=1}^{k} \delta B_2(C_i) = \bigcup_{i=1}^{k} [\{\mathbf{x_{m_{1,i}}}\} \cup \{\mathbf{x_{m_{2,i}}}\} \cup n_{2,id}]$
$\forall_{\mathbf{a},\mathbf{b}} : \mathscr{V}_{ab}(U_2|d)$
for j $\leftarrow$ **3 to T do**
  for i $\leftarrow$ **1 to k do**
    Find MM point $\mathbf{x_{m_{j,i}}}$
    get $n_i = j$ local neighbors $n_{j,id}$ of $\bigcup_{r=1}^{j} \{\mathbf{x_{m_{r,i}}}\}$
    $\delta B_j(C_i) = \{\bigcup_{r=1}^{j} \{\mathbf{x_{m_{r,i}}}\}\} \cup n_{j,id}$
  end for
  Form the reduced partition $U_j \leftrightarrow \bigcup_{i=1}^{k} \delta B_j(C_i)$
  Compute: $\mathscr{V}_{ab}(U_j)$
  Sort $\{\mathscr{V}_{ab}(U_r) : r = 1, ..., j\}$ in descending order
  $\{\mathscr{V}_{ab}(U_{(r)}) : r = 1, ..., j\}$ ▷ $U_{(k)}$ is the k-th partition after sorting
  if $std(\mathscr{V}_{ab}(U_j), \mathscr{V}_{ab}(U_{j-1}), \mathscr{V}_{ab}(U_{j-2})) \leq \varepsilon$ **then**
    $\mathscr{V}_{ab}(U_j|d)$ ▷ std= standard deviation
    break;
  else
    next j
  end if
end for

Output: $\forall_{\mathbf{a},\mathbf{b}} : \mathscr{V}_{ab}(U_j|d) : j$

for average linkage distance ($a = 3$) because the average distance changes with each extracted data point.

5.5.2 Boundary Vector algorithms

The support vector machine (SVM) [242] relies on support vectors at the boundaries of labeled subsets to define an optimal separating hyperplane. The boundary points of each data class that are "away" from the supporting hyperplanes play no role in SVM designs. However, several recent papers that estimate hyper-parameters for the one class support vector machine [84, 85] do estimate and use all of the boundary points in datasets assumed to contain normal and anomalous (but unlabeled) samples. Algorithms 12 [84] and 13 [85] are based on this type of boundary estimation.

5.5.2.1 QMS+

Algorithm 12 presents a brief summary of the part of the *Quick Model Selection* (QMS) [84] algorithm that estimates the boundary points of X using the K-nearest-neighbor (K-NN) rule. The parameter η is a shrinking factor which divides the sample into three groups viz., normal, outlier, and border-line [84]. Since, we used QMS to generate boundary point estimates in Algorithm 12, we name it QMS+.

Algorithm 12 The QMS+ algorithm

Input: K- the number of nearest neighbors,
η- shrinking factor,
a, b- inputs to compute GDI $\mathscr{V}_{ab}$

1: **for i** $\leftarrow$ **1 to k do**
2: Find $n_{iK,d} = \{K\text{-NNs of each } \mathbf{x} \in C_i \text{ wrt. } d\}$.
3: $\bar{d}_i = \sum_{\mathbf{y} \in n_{iK,d}} d(\mathbf{x}, \mathbf{y})/K$
4: Sort $\{\bar{d}_i\} \rightarrow \{\bar{d}_{(i)}\}$ in ascending order
5: Find last sudden change point (index) m in $\{\bar{d}_i\}$. ▷ see [249] for details of this step
6: $\omega_L = \lceil \eta m \rceil$; $\omega_U = \lceil (2-\eta) m \rceil$
7: $X_{Q_i} = \{\mathbf{x}_{(i)} : \omega_L \leq (i) \leq \omega_U\}$
8: **end for**
9: $U_{QMS+} \leftrightarrow X_{QMS+} = \bigcup_{i=1}^{k} X_{Q_i}$
10: Compute $\mathscr{V}_{ab}(U_{QMS+}|d)$

Output: $\forall_{\mathbf{a},\mathbf{b}}$: $\mathscr{V}_{ab}(U_{QMS+}|d)$

5.5.2.2 BEPS+

Li *et al.* [85] describe a method they call *Border-Edge Pattern Selection* (BEPS) that is also based on the K-NN rule. Since, we used the portion of the BEPS boundary estimation algorithm to generate boundary (border) points, and subsequently, used them to compute approximate DI, we call modified algorithm BEPS+. The pseudocode of BEPS+ is shown in Algorithm 13.

The rationale given in [85] for the BEPS algorithm is that a boundary point has all or most of its nearest neighbors sitting on one side of the tangent plane which has **e** as an approximate normal vector. The vector **x** is an edge pattern when the number of neighbors $y_j \in n^{\mathbf{x}}_{iK,d}$ of **x** with $\theta_{y_j} \geq 0$ exceeds the threshold $1-\gamma$. Since $g(\mathbf{x})/K$ ranks points in ascending order of being an edge pattern, a predefined fraction of edge patterns with top ranks can be selected to compute $\mathscr{V}_{ab}(U_{BEPS+}|d)$ without finding a best γ. We modified the algorithm in [85] to select a fraction of the top ranked points (given the ranking $g(\mathbf{x})/K$) as $X_{BEPS+,i}$.

Algorithm 13 The BEPS+ algorithm

Input: $\gamma \in (0,1]$- a threshold
$K = \lceil 5 \ln N \rceil$- nearest neighbors
a, b- inputs to compute GDI $\mathscr{V}_{ab}$

for $\mathbf{i} \leftarrow$ **1 to k do**
  for $\mathbf{t} \leftarrow$ **1 to** $|\mathbf{C_i}|$ **do**
    Find $n^{\mathbf{x}_t}_{iK,d} = \{K\text{-NNs of } \mathbf{x}_t \in C_i \text{ wrt. } d\}$.
    for $\mathbf{j} \leftarrow$ **1 to K do**
      $\mathbf{y}_j \in n^{\mathbf{x_t}}_{iK,d} : \mathbf{e_j}(\mathbf{x_t}) = (\mathbf{x_t} - \mathbf{y_j})/||\mathbf{x_t} - \mathbf{y_j}||_\mathbf{d}$, ▷ $\mathbf{y}_j$ is the j-th NN from $\mathbf{x_t}$; and $\mathbf{e_j}(\mathbf{x_t})$ is the norm vector for each K-NN
    end for
    $\mathbf{z_t} = \sum_{j=1}^{K} \mathbf{e}_j(\mathbf{x_t})$
    $g(\mathbf{x}_t) = 0$
    for $\mathbf{j} \leftarrow$ **1 to K do**
      $\theta_{\mathbf{y}_j} = \langle (\mathbf{x_t} - \mathbf{y}_j), \mathbf{z_t} \rangle$
      if $\theta_{\mathbf{y}_j} \geq 0$ **then**
        $g(\mathbf{x}_t) = g(\mathbf{x}_t) + 1$
      end if
    end for
    if $(g(\mathbf{x}_t)/K) > (1-\gamma)$ **then**
      $\mathbf{x}_t \in X_{BEPS+,i}$
    end if
  end for
end for
$U_{BEPS+} \leftrightarrow X_{BEPS+} = \bigcup_{i=1}^{k} X_{BEPS+,i}$
Compute $\mathscr{V}_{ab}(U_{BEPS+}|d)$

Output: $\forall_{\mathbf{a},\mathbf{b}}: \ \mathscr{V}_{ab}(U_{BEPS+}|d)$

5.6 Experiments

We performed six sets of experiments. In the first experiment, we study the impact of α on the approximation of DI by the αMMRS algorithm. In the second experiment, we discuss the impact of K on the approximation of DI for the QMS+ algorithm. In the third experiment, we study the effect of unequal cluster sizes N_i on the approximation of X by X_α. In the fourth experiment, we compare the boundaries and MMRS skeletons of all six methods on the 2*D Banana* dataset. In the fifth experiment, we compare all six methods on all datasets based on their approximated DI value and computation time. In the last experiment, we discuss the termination of the *i*MMRS and *i*nMMRS algorithms.

5.6.1 Computation Protocols

All algorithms were coded in MATLAB on a PC with the following configuration; OS: Windows 7 (64 bit); processor: Intel Core $i7-4770$ @3.40GHz; RAM: 16GB. The parameters for each method were chosen as follows, unless stated otherwise:

- metric $d = d_E$ (Euclidean),
- $K = 5$ for QMS+ and $K = \lceil 5 \ln N \rceil$ as in [85] for BEPS+
- $\alpha = 0.005$ for XG, BANANA, HAR and MNIST dataset,
- $\alpha = 0.0005$ for FOREST dataset,
- $\alpha = 0.00005$ for BigX and ACTR dataset,
- $k' = n = \lceil \alpha N \rceil$ for αMMRS and αnMMRS.

The input parameters η and γ ordinarily determine the boundary points extracted by QMS+ and BEPS+, respectively. However, to make the comparisons as fair as possible, we computed $n^* = \lceil \alpha N \rceil$ for αMMRS (Algorithm 8), and used the n^* highest ranked boundary points from each method in the experiments. Our motivation for choosing the values of α and K are discussed in Experiments 1 and 2. The value of K should be selected such that it reflects the local neighborhood of the data points. A small K will reflect a very local neighborhood. A large K will make nearest

Table 5.1: Seven datasets used for our experiments

Dataset	$\|\mathbf{X}\| = \mathbf{N}$	p	k	$\mathbf{pN^2}$
XG	55,500	2	3	6.05E9
BANANA	50,000	2	2	5E9
ACTR	1,140,000	45	19	6E13
HAR	10,299	561	6	6E10
FOREST	581,082	54	7	2E13
MNIST	60,000	784	10	3E12
BigX	1,000,000	100	4	1E14

neighbors span a large space around **x**. Thus, a reasonable value of K should be used in order to reflect the curvature of the class surface. For BEPS+, Li *et al.* [85] suggest making the value of K a function of the size of the dataset, viz., $K = \lceil 5 \ln N \rceil$. The experimental study in [85] shows that the pattern selection rate is relatively insensitive to the value of K after a certain value of K.

5.6.2 Datasets

Table 5.1 lists the seven datasets used in the experiments. XG, Banana and BigX are synthetic, the ACTR, HAR and FOREST datasets are available from the UCI repository [236]. The MNIST data is available at [221]. The last column of Table 5.1 shows pN^2. The complexity of Dunn's index on the full data is $O(pN^2)$ [247]. ACTR is a resized version of the ACT data at the UCI website to $1,140,000 \times 45$ which we made by splitting a large time window signal to multiple small time windows. The BigX dataset is created by drawing labeled samples from a mixture of $k = 4$ circular Gaussian distributions, having the mean components $(-12,..,-12)_{100}$, $(-6,..,-6)_{100}$, $(6,..,6)_{100}$ and $(12,..,12)_{100}$, and the standard deviations $(1,..,1)_{100}$, $(2,..,2)_{100}$, $(1,..,1)_{100}$ and $(2,..,2)_{100}$. The values of k shown in Table 5.1 are the numbers of labeled subsets (assumed here to form clusters) in each dataset.

5.6.3 Experiments

5.6.3.1 Effect of α on DI approximation

In this experiment, we investigate the effect of α on the approximations of Dunn's original index $\mathscr{V}_{11}$, by varying α from 0.05 to 0.00005 in multiples of 10. Although both αMMRS and

Table 5.2: $\mathscr{V}_{11}$ values of αMMRS for different values of α.

Dataset	$\alpha = 1$ (literal $\mathscr{V}_{11}$)	$\alpha = 0.05$	$\alpha = 0.005$	$\alpha = 0.0005$	$\alpha = 0.00005$
XG	***0.27*** (61s)	0.27 (1.38s)	0.27 (0.23s)	0.28 (0.15s)	0.32 (0.14s)
Banana	***0.07*** (39s)	0.07 (0.11s)	0.07 (0.07s)	0.08 (0.10s)	0.11 (0.09s)
ACTR	***0*** (56656s)	0 (37220s)	0 (3704s)	0 (363s)	0 (22s)
BigX	***1.26*** (820788s)	1.27 (41484)	1.28 (4152)	1.30 (419s)	1.31 (44s)
FOREST	***0.002*** (208382s)	0.003 (92312s)	0.005 (893s)	0.007 (113s)	0.02 (8.76s)
HAR	***0.09*** (614s)	0.11 (64.3s)	0.17 (4.8s)	0.32 (0.8s)	0.54 (0.4s)
MNIST	***0.15*** (24320)	0.16 (2879s)	0.18 (273s)	0.33 (36.9s)	0.43 (4.8s)

αnMMRS require the user to input α, we present this study for αMMRS, as $X_\alpha \subset X_{\alpha n}$. Table 5.2 shows the $\mathscr{V}_{11}$ values and computation times (in parentheses) for different values of α for all seven datasets. When $\alpha = 1$ (second column of Table 5.2), all of the input data are used to compute $\mathscr{V}_{11}$ without applying the MMRS algorithm to each cluster. The values in Table 5.2 show that the approximations are close to literal values for higher values of α. However, the computation time also increases significantly for each multiplication of 10 to α. For small datasets, e.g., XG and Banana, a very small value of $\alpha = 0.00005$ means that only two[1] to three MMRS points per cluster are extracted, which are insufficient to compute a good approximate DI. And, for a big dataset, e.g., FOREST, a large value of $\alpha = 0.05$ means almost all the points are extracted from a cluster if $k' = \lceil \alpha N \rceil \geq |C_i|$. The values in Table 5.2 suggest that $\alpha \approx inverse(\text{order of N})$ is a good choice for any dataset. Hence, the values of α for subsequent experiments are chosen using this criterion, and as discussed in Section 5.6.1.

The parameter α of αMMRS specifies the number of points extracted by MMRS from X from each labeled subset. When $\alpha = 0.05$, the approximations are based on using $1/20$ of the input data. Table 5.2 shows that for this choice, the worst approximation to DI by αMMRS is in error by 0.02 (the HAR data), and it is obtained in about $1/10$ of the time required to secure the literal value. Three of the seven approximations at $\alpha = 0.05$ are exact.

5.6.3.2 Effect of K in QMS+ on DI approximation

In this experiment, we investigate the effect of the number of nearest neighbors K on boundary extraction by QMS+, by varying K from 1 to 9 with an interval of 2. Table 5.3 shows the $\mathscr{V}_{11}$ values

[1]For each cluster in each data, we choose $k' = (\max(2, \lceil \alpha N \rceil))$ MMRS points, as at least two points per cluster are required to compute the DI. If $k' = \lceil \alpha N \rceil \geq |C_i|$, then we choose $k' = \min(\lceil \alpha N \rceil, |C_i|)$

Table 5.3: $\mathscr{V}_{11}$ values based on the QMS+ algorithm for different values of K.

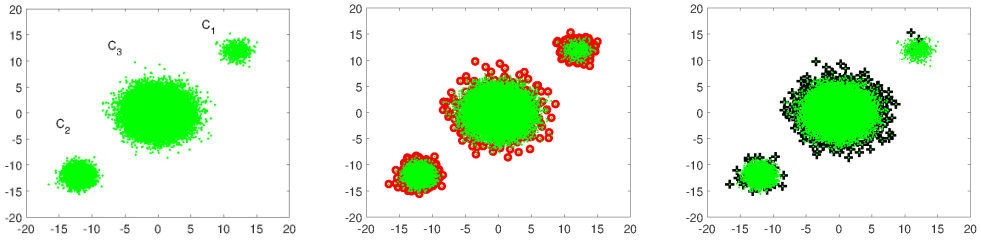
Dataset	$\mathscr{V}_{11}$	$K=1$	$K=3$	$K=5$	$K=7$	$K=9$
XG	***0.27***	0.27 (0.24s)	0.27 (0.19s)	0.27 (0.19s)	0.27 (0.19s)	0.27 (0.19s)
Banana	***0.07***	0.10 (0.1038s)	0.07 (0.1895s)	0.07 (0.1818s)	0.07 (0.1817s)	0.07 (0.1960s)
ACTR	***0***	0.13 (3677s)	0.10 (3665)	0.10 (3689s)	0.10 (3695s)	0.10 (3686s)
BigX	***1.26***	1.36 (19845s)	1.34 (19891)	1.30 (19895)	1.30 (19847)	1.30 (19899)
FOREST	***0.002***	0.006 (5686s)	0.006 (5693s)	0.006 (5678s)	0.006 (5696s)	0.006 (5648s)
HAR	***0.09***	0.16 (16.42s)	0.14 (16.10s)	0.14 (16.34s)	0.14 (16.24s)	0.14 (16.46s)
MNIST	***0.15***	0.21 (357s)	0.20 (356s)	0.20 (359s)	0.20 (361s)	0.20 (360s)

and computations times (in parentheses) for different values of K for all the seven datasets. The values are not too sensitive to K. In particular, the approximations are all identical for $K = 3, 5$, 7 and 9, excepting the difference of 0.04 for BigX on passing from $K = 3$ to $K = 5$. Moreover, the computation time is fairly stable to small changes in K and is random for some datasets as the complexity of the k-d tree, to compute nearest neighbors, may slightly vary in different runs. For $K = 1$, the $\mathscr{V}_{11}$ approximations are affected as $K = 1$ does not reflect the local proximity of the data points. As discussed in Section 5.6.1, the approximations do not change after a certain value of K ($= 5$).

5.6.3.3 Effect of unequal cluster sizes (αMMRS vs $\alpha\{i\}$MMRS)

This experiment studies whether a fixed fraction α of $|X| = N$ yields different approximations to Dunn's index than applying the fraction α to each C_i individually, i.e., to $|C_i|$. This study is performed on the XG and FOREST datasets. These datasets have highly unbalanced cluster distributions, which makes them suitable for this study. The other datasets in our experiments have (almost) balanced cluster distributions, so αMMRS and $\alpha\{i\}$MMRS will probably yield similar DI values for these datasets.

The input data XG are shown in Fig. 5.2 (a). XG has $k = 3$ labeled, $p = 2$ dimensional Gaussian clusters with $N = 55,500$ points, composed of $|C_1| = 500$, $|C_2| = 5000$ and $|C_3| = 50000$ points. Choosing $\alpha = 0.005$ for $N = 55,500$ points instructs αMMRS to extract 275 points from each of the three clusters, resulting in a total of 875 points. These points are X_α, the round circles in Fig. 5.2 (b). This scheme extracts 55% of the points in C_1, but only 0.55% of the points in C_3. The crosses in Fig. 5.2 (c) show the 278 points retrieved when the fraction of points extracted from C_i is α of $|C_i|$ instead of α of N. For example, only $0.005 \times 500 = 2.5$ (rounded up to 3) points



(a) Unbalanced dataset XG (b) $U_\alpha \leftrightarrow X_\alpha = \bigcup_{i=1}^{k} \delta B_\alpha(C_l)$ for $\alpha = 0.005$ (c) $U_{\alpha\{i\}} \leftrightarrow X_{\alpha\{i\}}$ for $\alpha\{i\} = 0.005$

Figure 5.2: αMMRS and $\alpha\{i\}$MMRS for the 2D XG dataset.

Table 5.4: $\mathscr{V}_{*1}$ values (times) for $\alpha = \alpha\{i\} = 0.005$, for XG dataset

	$\alpha = 1$, $\lvert X\rvert = 55{,}000$	$\alpha = 0.005$, $\lvert X_\alpha\rvert = 875$	$\alpha\{i\} = 0.005$, $\lvert X_{\alpha\{i\}}\rvert = 278$
$\mathscr{V}_{11}(U\lvert d_E)$	***0.27*** $(61s)$	**0.27** $(0.35s)$	**0.27** $(0.26s)$
$\mathscr{V}_{21}(U\lvert d_E)$	***1.57*** $(61s)$	1.58 $(0.35s)$	**1.57** $(0.26s)$
$\mathscr{V}_{31}(U\lvert d_E)$	***3.40*** $(46s)$	**3.40** $(0.35s)$	2.99 $(0.26s)$

are extracted from C_1 using this method.

Table 5.4 shows Dunn's index for X, X_α, and $X_{\alpha\{i\}}$ using the three set distances corresponding to single, complete and average linkage; $a = 1, 2$ and 3 in the numerator of (5.4). The approximations of $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$ are not affected by unbalanced cluster fractions. The estimate of $\mathscr{V}_{31}$ using X_α agrees with its value on X, but drops when using $X_{\alpha\{i\}}$, suggesting that $\mathscr{V}_{31}$ is affected by unbalanced cluster sizes. Apparently XG has CS clusters when the set distance is $\delta = \delta_{CL}$ or $\delta = \delta_{AL}$, indicated by DI values greater than 1, but does not contain CS clusters for the choice $\delta = \delta_{SL}$.

Table 5.5 shows $\mathscr{V}_{*1}$ values for X, X_α, and $X_{\alpha\{i\}}$, for the FOREST dataset. All the approximations for X_α agree with their value on X; however, $\mathscr{V}_{11}$ and $\mathscr{V}_{31}$ differ for $X_{\alpha\{i\}}$, suggesting that they are affected by unbalanced cluster fractions. Seemingly, FOREST has non-CS clusters for all three set distances.

Table 5.5: $\mathscr{V}_{*1}$ values (times) for $\alpha = \alpha\{i\} = 0.005$, for FOREST dataset

	$\alpha = 1$, $\lvert X\rvert = 581{,}012$	$\alpha = 0.005$, $\lvert X_\alpha\rvert = 2030$	$\alpha\{i\} = 0.0005$, $\lvert X_{\alpha\{i\}}\rvert = 295$
$\mathscr{V}_{11}(U\lvert d_E)$	***0.00*** $(208382s)$	**0.00** $(115s)$	0.02 $(0.29s)$
$\mathscr{V}_{21}(U\lvert d_E)$	***0.93*** $(208349s)$	0.94 $(119s)$	0.94 $(0.27s)$
$\mathscr{V}_{31}(U\lvert d_E)$	***0.64*** $(172460s)$	**0.65** $(112s)$	0.70 $(0.25s)$

Computation times for XG and FOREST in this experiment are shown in parentheses in Table 5.4 and Table 5.5, respectively. For XG , using X_α instead of X affords a speedup of about 175 times for $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$; and about 130 times for $\mathscr{V}_{31}$, with almost no loss in accuracy. For FOREST, a speedup of about 1750 times is achieved for all approximations ($\mathscr{V}_{*1}$) using X_α instead of X, and a speed up of 750,000 is achieved using $X_{\alpha\{i\}}$. However, the approximation accuracy drops for $X_{\alpha\{i\}}$, especially for $\mathscr{V}_{11}$ and $\mathscr{V}_{31}$.

The results of this experiment suggest that unbalanced clusters do not bias the αMMRS algorithm for some Dunn's indices, but not all of them. Approximated values of $\mathscr{V}_{11}$ and $\mathscr{V}_{31}$ based on X_α are equal to their literal values (bold and italic) based on all of X; $\mathscr{V}_{21}$ differs by 0.01 for the XG and FOREST datasets. The estimations of Dunn's original index for all six algorithms for the XG dataset are shown in the second row of Table 5.6. Three of the six estimates produce the exact value on 10 trials (0.27, the literal value) for the XG data; the other three are within ± 0.01 or ± 0.04 on all 10 trials.

5.6.3.4 Boundary points and MMRS skeleton on Banana data

This experiment compares estimates of Dunn's index based on extracted partitions found by the six algorithms presented in Section 5.5. The input dataset *Banana* ($X_{\mathscr{B}}$) is the pair of semi-ellipsoidal clusters shown in Fig. 5.3, which we call the Banana data. There are $|C_1| = |C_2| = 25,000$ points in each of the clusters, so, $N = |X_{\mathscr{B}}| = 50,000$.

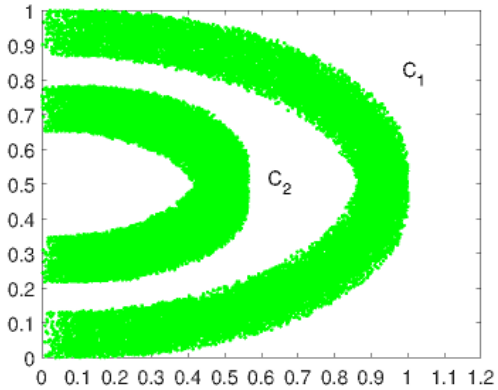
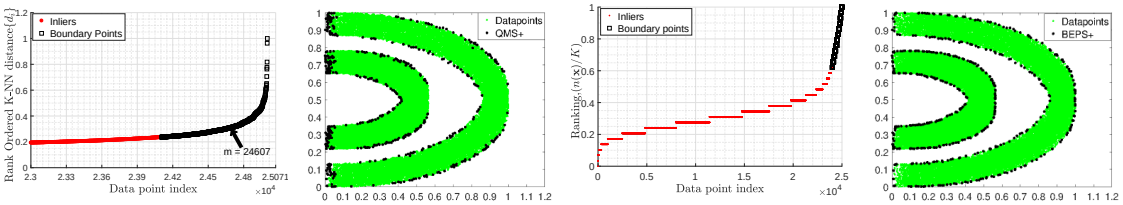
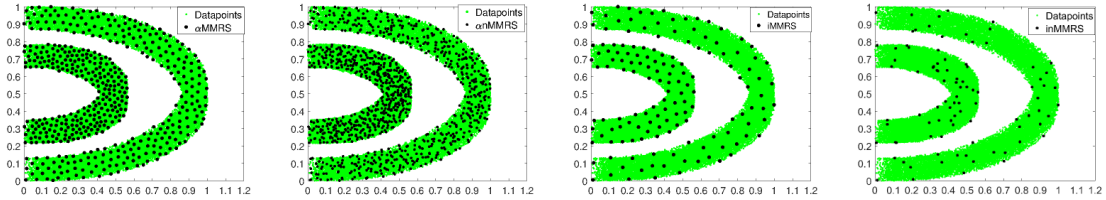


Figure 5.3: The Banana (two-dimensional) data: $|X_{\mathscr{B}}| = 50,000$

The cardinalities of the αMMRS (500) and αnMMRS (1000), QMS+ (500) and BEPS+ (500)



(a) $\{\bar{d}_i\} \Rightarrow$ QMS+ on X_2 (b) $|X_{QMS+}| = 500$ (c) $\{g(i)/K\} \Rightarrow$ BEPS+ on X_2 (d) $|X_{BEPS+}| = 500$



(e) $|X_{\alpha MMRS}| = 500$ (f) $|X_{\alpha nMMRS}| = 500$ (g) $|X_{iMMRS}| = 141$ (h) $|X_{inMMRS}| = 121$

Figure 5.4: Boundaries and MMRS Skeletons for the 2*D* Banana data.

subsets are a direct consequence of choosing $\alpha = 0.005$. We selected only the highest ranked $n^* = \lceil \alpha N \rceil = 500$ boundary points from each of QMS+ and BEPS+. The cardinalities for *i*MMRS (141) and *in*MMRS (121) are determined by the termination criteria for these two MMRS methods.

Fig. 5.4 (a) graphs the rank-ordered K-NN distances $\{\bar{d}_i\}$ obtained by QMS+ for cluster C_2. The last sudden change point in this cluster occurs at index $m = 24,607$, and QMS+ obtained a similar graph for cluster C_1, the outer cluster in Fig. 5.3. The overall result was the selection of 2% of 50,000, of which the highest ranked 500 were selected as shown for X_{QMS+} in Fig. 5.4 (b). Fig. 5.4 (c) and (d) are the analogous displays for the ranking function $\text{rank}(\mathbf{x}) = g(\mathbf{x})/K$ of the BEPS+ algorithm, for which we also prespecified a total of 2% of the boundary points; the highest ranked 500 points are shown in Fig. 5.4 (d).

QMS+ (Algorithm 12) and BEPS+ (Algorithm 13) both produce visually appealing subsets of extreme points. The BEPS+ points appear to be slightly better visually since they capture a few points missed by QMS+ at the extreme right side of each cluster, whereas the QMS+ points tend to pool a bit at the left sides of the two banana clusters.

Figs. 5.4 (e)-(h) show the MMRS points extracted from the Banana data with Algorithms 8-11, respectively. All four MMRS methods extract *some* boundary points, but they also retrieve some points in the "interior" of each cluster. Hence, the term *MMRS skeleton* seems more descriptive than calling them boundary approximations. Nonetheless, we will see that the MMRS skeletons

contain the points we need to make pretty good estimates of Dunn's index.

Estimates of Dunn's original index ($\mathscr{V}_{11}$) for the Banana data are shown in the third row of Table 5.6. All six estimates are exact, producing the same value (0.07) as the literal computation on all $50,000$ points, so the Banana data illustrates differences in the extraction methods, but it does not provide us with a comparison for the six methods that points to one of them as being superior to the others in terms of the best approximation to Dunn's literal index. However, one of the MMRS methods will emerge as "empirically best in class" when we consider the remaining five datasets.

Table 5.6: Average (10 trials) approximate values of Dunn's index $\mathscr{V}_{11}$ for six algorithms on seven datasets.

Dataset	$\mathscr{V}_{11}$ ($\alpha = 1$)	αMMRS	αnMMRS	*i*MMRS	*i*nMMRS	QMS+	BEPS+
XG	***0.27***	$\mathbf{0.27 \pm 0.04}$	$\mathbf{0.27 \pm 0.01}$	$\mathbf{0.27 \pm 0.01}$	$\mathbf{0.27 \pm 0.00}$	**0.27**	**0.27**
Banana	***0.07***	$\mathbf{0.07 \pm 0.01}$	$\mathbf{0.07 \pm 0.00}$	$\mathbf{0.07 \pm 0.01}$	$\mathbf{0.07 \pm 0.01}$	**0.07**	**0.07**
ACTR	***0***	$\mathbf{0 \pm 0}$	$\mathbf{0 \pm 0}$	$\mathbf{0 \pm 0}$	$\mathbf{0 \pm 0}$	0.10	0.10
BigX	***1.26***	1.32 ± 0.09	1.28 ± 0.04	1.32 ± 0.03	$\mathbf{1.26 \pm 0.01}$	1.30	1.43
FOREST	***0.002***	0.005 ± 0.001	0.003 ± 0.000	0.005 ± 0.001	$\mathbf{0.002 \pm 0.000}$	0.006	0.004
HAR	***0.09***	0.18 ± 0.06	0.11 ± 0.03	0.13 ± 0.06	$\mathbf{0.09 \pm 0.00}$	0.14	0.13
MNIST	***0.15***	0.18 ± 0.08	0.17 ± 0.06	0.20 ± 0.06	$\mathbf{0.15 \pm 0.01}$	0.20	0.21

5.6.3.5 Comparison of six algorithms on all datasets

MMRS algorithm (Algorithm 3) is initialized at a random object, so 10 trials with different initializations may result in different MMRS skeletons. This experiment will determine how sensitive the four MMRS sampling methods are to this parameter. The BEPS+ and QMS+ methods will produce the same boundary points in all 10 trials, so there is no variance in the estimates of Dunn's index based on these two methods. Table 5.6 lists the values of Dunn's original index and average (10 trials) estimates of it made by the six algorithms for all seven datasets in our experimental study. All of the estimates are pretty good, but *i*nMMRS is exact (± 0.01 for three of the datasets) for all six datasets, as shown by the bolded entries in the table. The largest variation in estimates of DI is 0.09 for BigX with αMMRS, which also shows the largest variations for XG, HAR and MNIST. The values in Table 5.6 confirm that MMRS is not very sensitive to changes in its initial index (m_0, in the first step of the Algorithm 3). So, on the basis of quality of approximation, *i*nMMRS algorithm is the clear leader.

An interesting and somewhat unexpected result seen in Table 5.6 is that Dunn's index and its four MMRS estimates are all 0 for the ACTR dataset, whereas the QMS+ and BEPS+ estimates of it are greater than 0. This confirms our suspicion that *removing points* from a crisp partition of a dataset might increase the estimated value of Dunn's index. We believe that this cannot happen when points are *added* to a crisp partition (at least for the $\mathscr{V}_{11}$ and $\mathscr{V}_{21}$ cases), but we do not pursue this conjecture in this chapter.

Table 5.7 lists the average (10 trials) CPU times needed by our six approximation methods (shortest times bolded) and Dunn's index. Fig. 5.5 is a graphical representation of the column values in Table 5.7. Dunn's index is the leftmost bar graph of the seven for each dataset, with an average computation time of $1.85 \times 10^5 = 51.4$ hours, as seen in the last row of Table 5.7. How much time do we save using the six approximation algorithms to estimate this value?

The minimum *average* time, 63 seconds, is achieved by the αMMRS algorithm, followed by αnMMRS at 77 seconds, *i*nMMRS at 140 seconds and *i*MMRS at 203 seconds. So, the four MMRS methods all represent a speedup on the order of $1000:1$. The QMS+ method averaged 1.17 hours, while the BEPS+ method averaged 1.87 hours, so the boundary point algorithms (Algorithms 12 and 13) take quite a bit more (several orders more) time and do not provide better estimates than the MMRS methods.

Table 5.7: Average (10 trials) CPU times (seconds) for six algorithms on seven datasets.

Dataset	$\mathscr{V}_{11}$ ($\alpha = 1$)	αMMRS	αnMMRS	*i*MMRS	*i*nMMRS	QMS+	BEPS+
XG	60	0.27	0.65	1.09	1.12	**0.19**	1.85
Banana	*38.8*	0.34	0.35	0.16	**0.13**	0.19	1.587
ACTR	*56656*	19	24	**4.0**	4.3	3690	4372
BigX	*820788*	**40**	45	276	343	19895	33687
FOREST	*208382*	101	109	**65**	207	5679	8630
HAR	*614*	**4.8**	5.2	132	28	17	27
MNIST	*24320*	**278**	354	946	395	359	539
Average	$1.85(10^5)$	**63**	77	203	140	4234	6751

Graphs of the CPU times in Fig. 5.5 also show that there are two pairs of estimates that are essentially equal in CPU time, viz., (αMMRS, αnMMRS) and (QMS+, BEPS+), so from the standpoint of CPU time, either choice from the two pairs will run in about the same amount of time.

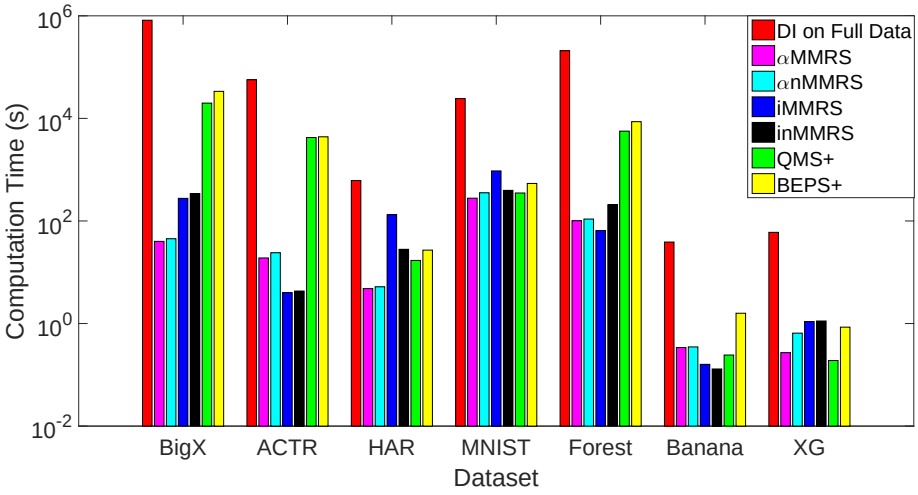


Figure 5.5: CPU times (log scale on y-axis) for six methods and seven datasets.

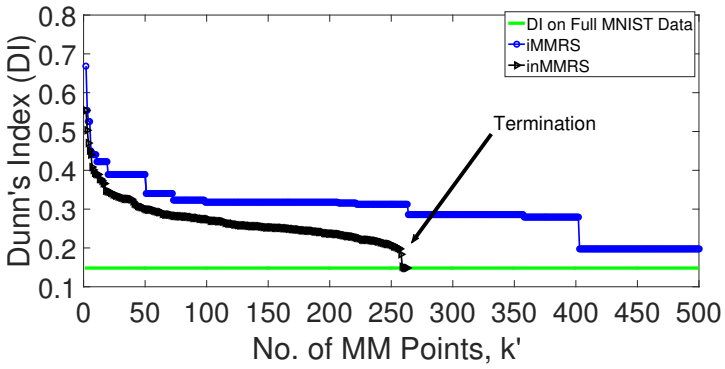


Figure 5.6: Termination: *i*MMRS and *i*nMMRS

5.6.3.6 Termination of *i*MMRS and *i*nMMRS

Recall that the *i*nMMRS algorithm was introduced to overcome the tendency for premature termination of *i*MMRS. Fig. 5.6 compares the estimates of Dunn's index shown in Table 5.7 made by these two algorithms at successive values of k' on the MNIST dataset ($\mathscr{V}_{11} = 0.1482 \approx 0.15$). The *i*MMRS estimate does not approach the target value using the termination criterion (say $T1$), as shown at line 15 of Algorithm 10, even when 500 additional MMRS points are used. On the other hand, *iN*MMRS terminates for $k' = 262$ at the exact value of 0.1482 using the standard deviation (std) based termination criterion (say $T2$) shown at line 25 of Algorithm 11. This shows both the necessity for and accuracy of termination criterion ($T2$) of Algorithm 11.

The next section provides a brief analysis of the asymptotic time complexity for Dunn's indices and our six approximation algorithms for computing it when N is very large.

5.7 Computational Complexity

The computational complexity of all 18 generalized Dunn's indices is derived in [247]. Sixteen of the GDIs are $O(pN^2)$: $\mathscr{V}_{43} = O(pN + pk^2)$; $\mathscr{V}_{53} = O(pkN)$. Here are the complexities for the six approximation algorithms:

$$MMRS = \max\{O(c'pN), O(k'N)\} = O(k'pN) \tag{5.14a}$$

$$\alpha MMRS = \max\{O(k'pN), O((kp(kk')^2)\} \tag{5.14b}$$

$$\alpha \text{n}MMRS = \max\{O(k'pN), O(kN), O(p(k(k'+n))^2)\} \tag{5.14c}$$

$$iMMRS = \max\{O(k'pN), O((p(kk')^2)\} \tag{5.14d}$$

$$i\text{n}MMRS = \max\{O(k'pN), O(k'N), O(p(k(k'+n))^2)\} \tag{5.14e}$$

$$QMS+, BEPS+ = \max\{O(N(K+p)), O(N\ln(N))\} \tag{5.14f}$$

The estimate $O(k'pN)$ for MMRS was established in [47]. The four MMRS algorithms have different complexities, but in all practical cases, they are dominated by the complexity of the MMRS algorithm. Specifically, for big datasets N is very large, so k, k', kk', $k'+n$ should all be small relative to N. The value of k' in (5.14d) is known at termination of iMMRS. The values of k' and n in (5.14e) are known at termination of inMMRS. It is pretty safe to assume that $(K+p) > \ln N$, for which (5.14f) becomes $O(N(K+p))$ for QMS+ and BEPS+. Thus, all six methods for estimation of Dunn's index are linear in N, the number of samples in the data.

5.8 Summary

This chapter presented six methods for approximating values of Dunn's index that all have linear complexity in N. Four methods used variations of the Maximin Random Sampling (MMRS) to extract skeletal subsets from crisp partitions of the big data that presumably contain the extremal

points in the data needed for determination of Dunn's index. We compared the four MMRS methods to estimates of Dunn's index based on two boundary point estimation methods (QMS+ and BEPS+) borrowed from the field of support vector machine research.

We conducted experiments on seven labeled (hence, partitioned) datasets of varying sizes to compare approximation accuracy and savings in CPU time using the algorithms developed in this article. The *i*nMMRS algorithm offered an average speedup of about $1000:1$, and produced average values that matched values of Dunn's index up to ± 0.01 on all seven datasets when computed on the full dataset (cf. Table 5.7). Experiment 6 also demonstrated that randomly initializing Algorithm 3 was not detrimental to the stability of *i*nMMRS approximations. This method was, on average, 50 seconds slower than the best method for minimum CPU time (cf. Table 5.7), so the added average expense of less than a minute in CPU time makes the *i*nMMRS algorithm a clear winner in the competition for approximating Dunn's index on these seven datasets.

Chapter 6

Cluster Tendency Assessment and Anomaly Detection in High-Velocity Streaming Data

This chapter develops an incremental method of scalable iVAT, inc-siVAT, to (visually) detect evolving structure in high-velocity, streaming data. The inc-siVAT updates the MMRS sample points and reordered dissimilarity image (RDI) on the fly to track the changes in the data stream after each chunk. Numerical experiments demonstrate the applicability of the inc-siVAT algorithm for successfully detecting anomalies and visualizing evolving cluster structure in dynamic streams of four big datasets, including a real IoT data.

6.1 Introduction

The widespread realization of the IoT infrastructure in smart city networks generates huge streams of data from various sources, at a high rate. Automatic interpretation of high-velocity, massive data streams is required for timely detection of interesting or abnormal events, that usually emerge in the form of clusters. This problem necessitates the needs of visualization and event detection techniques for high-velocity streaming data.

At present, there is no technique on offer for visual assessment of *evolving cluster structure in high-velocity massive data streams*. The inc-VAT/dec-VAT and inc-iVAT/dec-iVAT algorithms [69] (discussed in Chapter 2) provide a fast method for visualizing a point by point cluster evolution in streaming data. But hardware and software constraints limit incremental VAT algorithms to a maximum window size of about $N \sim 5,000$ inputs. When this limit is reached, point by

point deletion and insertion maintains this fixed window size. Thus, incremental VAT presents the user with a view of possible structure for only the last N points in the input stream. If $N = 100,000$, the user will have an image of only the last window at the end of the process. The salient point is that the *history of cluster evolution is not available.*

Both sVAT and siVAT are suitable for cluster tendency assessment of big data. However, to handle streaming data, they also need to be re(applied) each time a new data point or a chunk of new data points arrives, which is not feasible due to computational complexities associated with retraining at each instance of the new data point or new chunk arrival.

To address this problem, this chapter proposes an *incremental version of the scalable iVAT* algorithm, called inc-siVAT, which deals with the high-velocity, massive streaming data in chunks. It first extracts a small size smart sample using Maximin Random Sampling (MMRS), then it incrementally updates the smart sample points on the fly, using our novel incremental MMRS (inc-MMRS) algorithm, to reflect changes in data streams after each chunk, and finally, produces an incrementally built iVAT image of the updated smart sample after each chunk, using inc-VAT/inc-iVAT and dec-VAT/dec-iVAT algorithms. These images (*aka* cluster heat maps) can be used to visualize evolving cluster structure and for anomaly detection in high-velocity streaming data. The inc-MMRS model summarizes the data seen by a fixed (small) size of intelligent samples. Therefore, when N exceeds 5000, the intelligent sample size will be fixed which also constrains the size of the iVAT image.

6.2 Related Work

Besides cluster heat-maps, scatterplotting is the main approach for visualizing evolving cluster structure in streaming data. Scatter plots visualize streaming data instances across time, as points in 2D or 3D spaces. For visualizing high-dimensional data, a typical approach is to project the data into lower-dimensions (2D or 3D) using a dimensionality reduction method, e.g., multi-dimensional scaling, principal component analysis [250]. A more sophisticated approach is to project the data into many 2D scatterplots to get a sense of the relationship between the features [251]. For data streams, most of these approaches visualize a sliding window of the data. However, these methods can fail for clusters having high overlap or shapes that are difficult to dis-

tinguish on a scatter plot. When the stream presents a large number of instances, it can be difficult to distinguish the clusters and data points themselves, as well as the clustering trends. For high-dimensional data, these methods suffer from all of the difficulties inherent with dimensionality reduction, such as which projection method to use, loss of structural information, computational complexity etc.

Cluster partitions on evolving data streams are often computed based on certain time intervals (or windows). There are three well-known window-based methods: landmark window, damped window, and sliding window [68]. Landmark window models consider the data stream from the beginning until now. An example of the landmark window model is the CluStream algorithm [64] which clusters the data stream over different time horizons in an evolving environment. CluStream uses a modified k-means algorithm for clustering which requires k as input. Damped window models associate weights, also called a forgetting or decay factor, with the data in the stream such that higher weights are given to recent data than those in the past. An example of the damped window model is the Den-Stream algorithm [68]. Den-Stream uses a density-based algorithm for clustering with a similar concept as of CluStream algorithm. The sliding-window based approach considers the data from now up to a certain range in the past. It is the most common approach to visualize evolving cluster structure in streaming data. Zhou *et al.* proposed SWClustering algorithm [138] which introduces a new data structure called *Exponential Histogram of Cluster Features* (EHCF), a combination of exponential histograms with temporal cluster features, to record the evolution of each cluster and to capture the distribution of recent records. In contrast to CluStream, which uses batch updating and stores the whole snapshot, SWClustering updates EHCF only when new records are collected.

Most of the existing methods for streaming data clustering require the number of clusters (k) to be known in advance, but in practice, this is usually unknown. Moreover, one of the main aspects of streaming data is that the number of clusters changes with time. In some methods, clusters are determined using some rules or using other parameters which are sensitive to changes in cluster structure. Moreover, these approaches were originally developed for clustering streaming data. Once clusters are obtained, the clusters can be visualized across time windows. However, they do not offer online visualization of evolving cluster structures in streaming data.

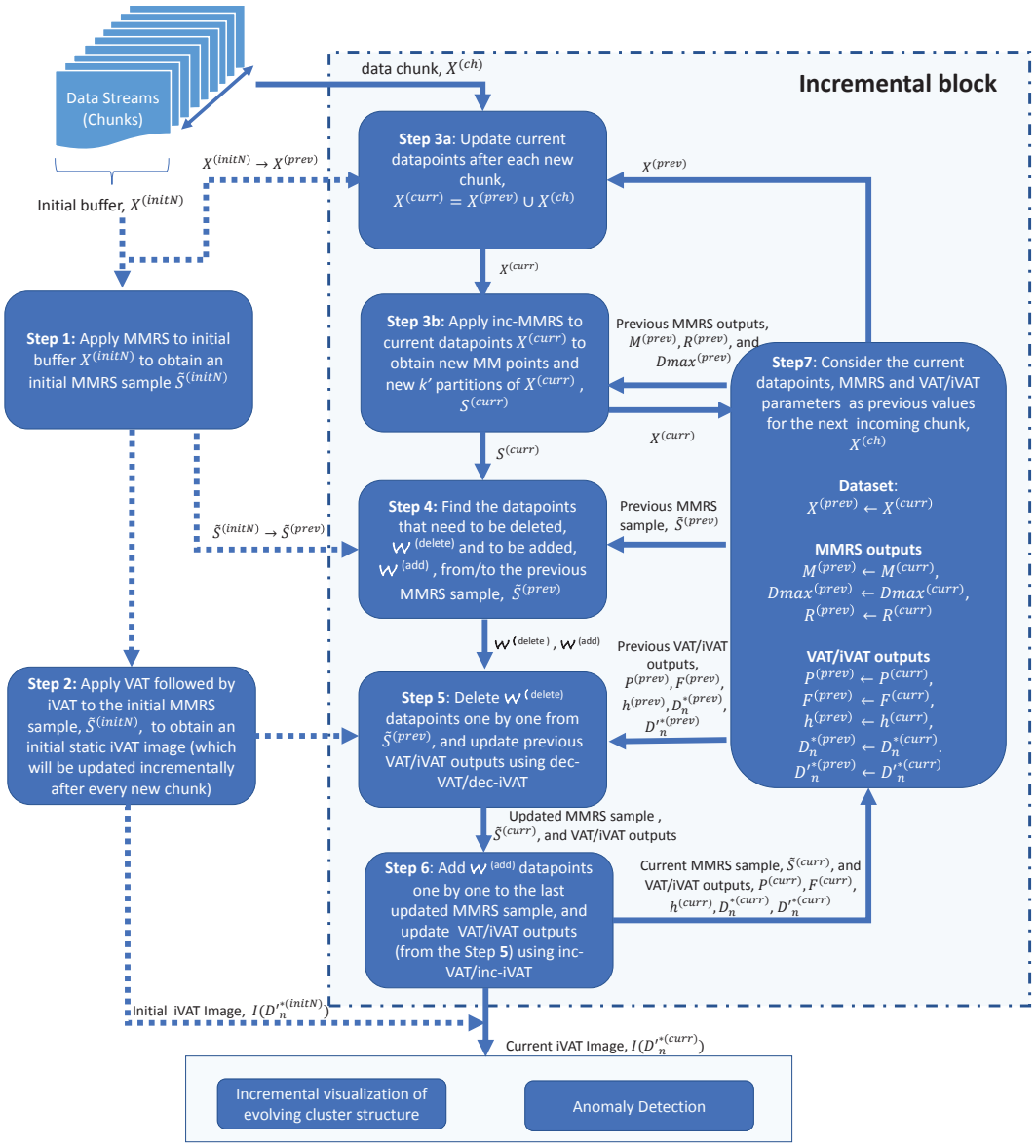


Figure 6.1: The architecture of our proposed framework.

6.3 Proposed Algorithm

The architecture of our inc-siVAT algorithm is shown in Fig. 6.1. The inc-siVAT algorithm deals with the high-velocity, big streaming data in chunks (of configurable size). (Step 1) First, an initial MMRS sample is obtained by applying the MMRS (Algorithm 3) on an initial buffer (or first chunk). A static *reordered dissimilarity image* (RDI) is obtained by applying VAT/iVAT on the initial MMRS sample (Step 2). At arrival of every new chunk, first, an augmented (current) dataset is obtained by appending the data points of the new chunk to the previous dataset (Step 3a), and then, the updated MM points and k' partitions are obtained by applying our novel inc-MMRS algorithm (Algorithm 15) to the current dataset (Step 3b). The inc-MMRS algorithm summarizes the data seen (after each chunk) by a fix (small) size ($n << N$) of intelligent MMRS sample. Next, the k' partitions of the current dataset are compared to the k' partitions of the previous MMRS sample, to estimate the number of data points that need to be deleted from or to be added to each of the k' partitions of previous MMRS sample to obtain the current (updated) MMRS sample (Step 4). Then, (the estimated number of) data points are deleted from the previous MMRS sample, and subsequently, the previous VAT/iVAT outputs are updated using dec-VAT/dec-iVAT algorithms (Step 5). Similarly, (the estimated number of) the new data points are added to the last updated MMRS sample, and subsequently, the previously computed VAT/iVAT outputs are updated using inc-VAT/inc-iVAT algorithms (Step 6), to obtain the updated RDI corresponding to the current dataset. The current dataset, MMRS outputs, and the VAT/iVAT parameters act as the previous values for the next chunk (Step 7). Steps 3-7 are performed for each new chunk to obtain an updated RDI. A display of updated RDI provides the visualization of evolving cluster structure after each new incoming chunk. Next, we discuss each step of our inc-siVAT algorithm in detail.

Let an initial dataset containing *initN* number of data points be denoted as $X^{(initN)}$ and an incoming chunk containing N_{ch} number of data points be denoted as $X^{(ch)}$. The inc-MMRS algorithm outputs M, $Dmax$, R, $\mathscr{N}$, and $\tilde{S}$ are updated everytime a new chunk arrives. For an input data X_N (having N data points) to MMRS algorithm, these outputs are defined as:

- a set of k' MM points in X_N, $M = \{m_1, m_2, ...m_{k'}\}$.
- $Dmax = \{dmax_1, dmax_2, ..., dmax_{k'}\}$, where $dmax_j$ represents the maximum distance of the j-th MM point in X_N, m_j, to the previous $(j-1)$ MM points in X_N, denoted by $M_{1:j-1}$.

Algorithm 14 inc-siVAT

Input: *initN*- The initial number of points; N_{ch}- Chunk size; k'- desired number of MM (*distinguished*) points; and n: an approximate (desired) size of MMRS sample

---

Step 1: Apply MMRS sampling (Algorithm 3) on the first *initN* points, $X^{(initN)}$
$(\tilde{S}^{(initN)}, M^{(initN)}, R^{(initN)}, \mathscr{N}^{(initN)}, Dmax^{(initN)})$ = MMRS $(X^{(initN)}, k', n)$

Step 2: Apply VAT and iVAT on D_n (a square distance matrix indexed by $\tilde{S}^{(initN)}$ in both rows and columns)
$(D'^{*(initN)}_n, P^{(initN)}, F^{(initN)}, h^{(initN)})$ = VAT $(D^{(initN)}_n)$
$D^{*(initN)}_n$ = iVAT $(D'^{*(initN)}_n)$
Initialization:
MMRS input: $M^{(prev)} = M^{(initN)}$, $Dmax^{(prev)} = Dmax^{(initN)}$, $R^{(prev)} = R^{(initN)}$, $\mathscr{N}^{(prev)} = \mathscr{N}^{(initN)}$, and $\tilde{S}^{(prev)} = \tilde{S}^{(initN)}$
inc-VAT/dec-VAT input: $P^{(prev)} = P^{(initN)}$; $F^{(prev)} = F^{(initN)}$; $h^{(prev)} = h^{(initN)}$; $D'^{*(prev)}_n = D'^{*(initN)}_n$; and $D^{*(prev)}_n = D^{*(initN)}_n$

%%Process each chunk of streaming data in incremental way%%
for each incoming chunk $X^{(ch)}$ or until termination **do**

Step 3: Apply inc-MMRS to $X^{(curr)}$ with previous MMRS outputs as input
3a. $X^{(curr)} = \{X^{(prev)} \cup X^{(ch)}\}$
3b. $(M^{(curr)}, R^{(curr)}, Dmax^{(curr)}, \mathscr{N}^{(curr)}, S^{(curr)})$ = Inc-MMRS$(X^{(prev)}, X^{(chunk)}, X^{(curr)}, M^{(prev)}, R^{prev)}, Dmax^{(prev)}, N_{ch}, k', n)$

Step 4: Find the data points $\in X^{(curr)}$ that need to be deleted from and to be added in $\tilde{S}^{(prev)}$ to obtain $\tilde{S}^{(curr)}$
for t $\leftarrow$ **1 to** k′ **do**
$Z_t = \{S^{(curr)}_t \cap \tilde{S}^{(prev)}_t\}$ ▷ data points in $S^{(curr)}_t$ which are already present in previous MMRS sample, $\tilde{S}^{(prev)}_t$
if $|Z_t| > n_t$ **then** ▷ $n_t = |\tilde{S}^{(curr)}_t|$ (or, the value at t-th index in $\mathscr{N}^{(curr)}$)
$\{w^{(delete)}_t\} \leftarrow |Z_t| - n_t$ random data points to be removed from $\tilde{S}^{(prev)}_t$ to obtain $\tilde{S}^{(curr)}_t$
else if $|Z_t| < n_t$ **then**
$\{w^{(add)}_t\} \leftarrow n_t - |Z_t|$ random data points from $S^{(curr)}_t$ (not present in $\tilde{S}^{(prev)}_t$) to be added in $\tilde{S}^{(prev)}_t$ to obtain $\tilde{S}^{(curr)}_t$.
end if
end for
$\mathscr{W}^{(delete)} = \{\{w^{(delete)}_1\}, \{w^{(delete)}_2\}, ..., \{w^{(delete)}_{k'}\}\}$; $\mathscr{W}^{(add)} = \{\{w^{(add)}_1\}, \{w^{(add)}_2\}, ..., \{w^{(add)}_{k'}\}\}$

Step 5: Delete $\mathscr{W}^{(delete)}$ elements and update previous VAT and iVAT outputs using dec-VAT/dec-iVAT algorithms
for i $\leftarrow$ **1 to** $|\mathscr{W}^{(\mathbf{delete})}|$ **do**
$j = arg(P^{(prev)} = \tilde{S}^{(prev)}_{w^{(delete)}_i})$ ▷ Find the position of $w^{(delete)}_i$ in $P^{(prev)}$. where $w^{(delete)}_i$ is the i-th element of $\mathscr{W}^{(delete)}$
$(P^{(curr)}, F^{(curr)}, h^{(curr)}, D^{*(curr)}_n)$ = dec-VAT $(P^{(prev)}, F^{(prev)}, h^{(prev)}, D^{*(prev)}_n, j)$
$D'^{*(curr)}_n$ = dec-iVAT $(D'^{*(prev)}_n, D^{*(curr)}_n, j)$
$\tilde{S}^{(curr)} \leftarrow \tilde{S}^{(prev)} / w^{(delete)}_i$
$K = arg(P^{(curr)} > j)$
$P^{(curr)}_K \leftarrow P^{(curr)}_K - 1$
end for

Step 6: Add $\mathscr{W}^{(add)}$ elements and update current VAT and iVAT outputs using inc-VAT/inc-iVAT algorithms
for i $\leftarrow$ **1 to** $|\mathscr{W}^{(\mathbf{add})}|$ **do**
$\{\mathbf{j}\} = \{\tilde{S}^{(curr)}_{P^{(curr)}_1}, \tilde{S}^{(curr)}_{P^{(curr)}_2}, ..., \tilde{S}^{(curr)}_{P^{(curr)}_n}\}$
$V = \{dist(\mathbf{x}^{(curr)}_{w^{(add)}_i}, \mathbf{x}^{(curr)}_{j_1}), ..., dist(\mathbf{x}^{(curr)}_{w^{(add)}_i}, \mathbf{x}^{(curr)}_{j_n})\}$ ▷ distance of $\mathbf{x}^{(curr)}_{w^{(add)}_i}$ to $\tilde{S}^{(curr)}$, ordered by $P^{(curr)}$
$(P^{(curr)}, F^{(curr)}, h^{(curr)}, D^{*(curr)}_n)$ = inc-VAT $(P^{(curr)}, F^{(curr)}, h^{(curr)}, D^{*(curr)}_n, V)$
$D'^{*(curr)}_n$ = inc-iVAT $(D'^{*(curr)}_n, D^{*(curr)}_n)$
$\tilde{S}^{(curr)} \leftarrow \tilde{S}^{(curr)} \cup w^{(add)}_i$
end for

Step 7: Update the previous dataset, MMRS and VAT/iVAT parameters assigning with the current dataset, MMRS and VAT/iVAT parameters, respectively, for the next chunk (or iteration)
Input data: $X^{(prev)} = X^{(curr)}$
MMRS input: $M^{(prev)} = M^{(curr)}$; $Dmax^{(prev)} = Dmax^{(curr)}$; $R^{(prev)} = R^{(curr)}$; $\mathscr{N}^{(prev)} = \mathscr{N}^{(curr)}$; and $\tilde{S}^{(prev)} = \tilde{S}^{(curr)}$
inc-VAT/dec-VAT input: $P^{(prev)} = P^{(curr)}$; $F^{(prev)} = F^{(curr)}$; $h^{(prev)} = h^{(curr)}$; $D'^{*(prev)}_n = D'^{*(curr)}_n$; and $D^{*(prev)}_n = D^{*(curr)}_n$
end for

Step 8 (optional): Obtain k-aligned partition of sample $\tilde{S}^{(curr)}$ by cutting the MST using cut threshold magnitudes ordered by $h^{(curr)}$ in MST, as given Eq. 6.1.

Output: iVAT image $I(D^{*(curr)}_n)$ to estimate k, and (optional) k-aligned partition of sample $\tilde{S}^{(curr)}$.

- an $N \times k'$ (distance) matrix R, whose entry r_{ij} denotes the distance of the j-th MM point in X_N, m_j, to the i-th data point in X_N, $\mathbf{x}_i$.
- $\mathscr{N} = \{n_1, n_2, ..., n_{k'}\}$ whose entry n_j represents the number of local (random) samples for m_j.
- a set of MMRS data points, $\tilde{S}$, whose entry $\tilde{S}_j$ contains the indices of the n_j local samples of m_j.

The pseudocode of inc-siVAT algorithm is given in Algorithm 14. Below, we explain each step of inc-siVAT algorithm:

Step 1: First, MMRS algorithm is applied to the initial data $X^{(initN)}$ to obtain an initial set of k' MM points, $M^{(initN)}$, and a MMRS sample, $\tilde{S}^{(initN)}$ (of size n). MMRS algorithm on $X^{(initN)}$ also returns $Dmax^{(initN)}$ and $R^{(initN)}$.

Step 2: Then, a static image of reordered dissimilarity matrix $D'^{*(initN)}_n$ is obtained by applying VAT followed by iVAT on a square distance matrix $D^{(initN)}_n$, which is computed using initial MMRS sample $\tilde{S}^{(initN)}$.

Let $X^{(prev)}$ denotes the previous data points arrived until the last time instant (or last iteration). After Step 2, $X^{(prev)}$ is initialized with $X^{(initN)}$, so $X^{(prev)} = X^{(initN)}$. Similarly, $M^{(prev)} = M^{(initN)}$, $Dmax^{(prev)} = Dmax^{(initN)}$, $R^{(prev)} = R^{(initN)}$, and $\tilde{S}^{(prev)} = \tilde{S}^{(initN)}$. Then, for each incoming chunk $X^{(ch)}$, following steps are performed to incrementally update the MMRS samples and the RDI.

Step 3: When a new chunk $X^{(ch)}$ arrives, an augmented current dataset $X^{(curr)}$ is obtained such that $X^{(curr)} = \{X^{(prev)} \cup X^{(ch)}\}$. Then, our novel incremental MMRS algorithm, inc-MMRS, is applied to $X^{(curr)}$ with k', n, $X^{(ch)}$, $X^{(prev)}$ and previous MMRS outputs as input to the inc-MMRS algorithm. The pseudocode of the inc-MMRS algorithm is given in Algorithm 15. Below, we explain the inc-MMRS algorithm.

1. *Initialization*: First, we initialize the current MMRS output (to be computed for $X^{(curr)}$), $M^{(curr)}$ and $Dmax^{(curr)}$, with the previous MMRS output i.e., $M^{(curr)} \leftarrow M^{(prev)}$, and $Dmax^{(curr)} \leftarrow Dmax^{(prev)}$. In order to store the distance of each MM point $\in M^{(currr)}$ to each of the data points in current dataset $X^{(curr)}$, the distance matrix $R^{(curr)}$ is first initialized with $R^{(prev)}$ which already contains the distance of each MM point $\in M^{(prev)}$ (now, $M^{(curr)}$) to $X^{(prev)}$, and then, appended with a zero matrix, Q^{ch} (of size $N_{ch} \times k'$), to store the distance of each

MM point $\in M^{(curr)}$ to each of the data points in new chunk $X^{(ch)}$, so $R^{(curr)} \leftarrow R^{(prev)}|Q^{ch}$. Let N_{prev} denotes the number of data points in $X^{(prev)}$ (also, the number of rows in $R^{(prev)}$), then the size of $R^{(curr)}$ is $N_{curr} \times k'$, where N_{curr} is the number of data points in $X^{(curr)}$, so $N_{curr} = N_{prev} + N_{ch}$.

2. In step 1 of the inc-MMRS algorithm, we identify if there is a new MM point in $X^{(curr)}$, after a new chunk $X^{(ch)}$ arrives. The change in MM points is identified using a boolean variable, *ChangeMM*, which is 1 if we get a new furthest (distinguished) point in $X^{(curr)}$, else it is 0. Recall that, a (new) MM point is the point which is furthest to the closest element of the existing MM points. To identify the change in MM points, first, we compute the distance of the first MM point $m_1^{(prev)}$ to each of the new data points $\in X^{(ch)}$, and store them at corresponding rows and column of $R^{(curr)}$. Then, for each MM point $m_t^{(prev)} \in M^{(prev)}$, the minimum Euclidean distance of each new data point $\in X^{(ch)}$ to the existing $(t-1)$ MM points, $M_{1:t-1}^{(prev)}$, is computed, which is denoted by $\mathfrak{D}$. If, corresponding to a MM point $m_t^{(prev)}$, the maximum distance in $\mathfrak{D}$ is greater than the previous maximum distance $dmax_t^{(prev)}$, then it indicates the presence of a new furthest or MM point in $X^{(curr)}$ i.e., $ChangeMM = 1$.

3. If a change in MM points is identified (say) at t-th MM point of $M^{(prev)}$, then the first $(t-1)$ MM points of $M^{(prev)}$ are retained in $M^{(curr)}$, but the next $k'-t+1$ MM points, denoted as $M_{t:k'}^{(curr)}$, are recomputed using Step 2 of the inc-MMRS algorithm. The remaining MM points in $X^{(curr)}$ are computed following the similar steps as mentioned in Step 1 of the MMRS algorithm.

4. It is possible that the some of the data points in X_{prev} that were closest to a MM point (say) $m_t^{(prev)} \in M^{(prev)}$ may not be closest to the same MM point $m_t^{(curr)} \in M^{(curr)}$ in current data (after a chunk arrives), even if $m_t^{(curr)} = m_t^{(prev)}$. In other words, those data points may now belong to a different group in X_{curr}. Therefore, we again divide the entire dataset $X^{(curr)}$ into the k' groups $\{S_t\}_{t=1}^{k'}$ using NPR, similar to the Step 2 of the MMRS algorithm (Algorithm 3). Some of these k' groups in $X^{(curr)}$ may correspond to the new MM points identified in the previous step.

Step 4: In this step, we identify the data points that were associated with the t-th group of the

Algorithm 15 inc-MMRS

Input: $(X^{(prev)}, X^{(ch)}, X^{(curr)}, M^{(prev)}, R^{prev)}, Dmax^{(prev)}, N_{ch}, k', n$

Initialization: Boolean variable, $ChangeMM = 0$ (1, if there is a new (furthest) MM point after adding $X^{(ch)}$ to $(X^{(prev)})$, Initialize $M^{(curr)} \leftarrow M^{(prev)}$; $Dmax^{(curr)} \leftarrow Dmax^{(prev)}$; and $R^{(curr)} \leftarrow \{R^{prev)}|Q^{ch}\}$ ▷ A zero matrix Q^{ch} (of size $N_{ch} \times k'$) is appended to $R^{(prev)}$ to store the distances of existing MM points, $M^{(prev)}$, to the data points of new chunk, $X^{(ch)}$, hence, the size of $R^{(curr)}$ is $N_{curr} \times k'$, where $N_{curr} = N_{prev} + N_{ch}$, and N_{prev} is the number of data points in $X^{(prev)}$ (or the number the rows in $R^{(prev)}$).

Step 1: Check if there is a change in previous MM points

$\mathfrak{D} = \{dist\{\mathbf{x}^{(prev)}_{m_1^{(prev)}}, \mathbf{x}^{(ch)}_1\}, .., dist\{\mathbf{x}^{(prev)}_{m_1^{(prev)}}, \mathbf{x}^{(ch)}_{N_{ch}}\}\}$

$R^{(curr)}_{((N_{prev}+1):N_{curr})1} = \mathfrak{D}$ ▷ Store the distances of the first MM point, $m_1^{(prev)}$, to the new data points, $X^{(ch)}$, in $R^{(curr)}$

for t ← 2 to k′ do

  $\mathfrak{D} = min(\mathfrak{D}, R^{(inc)}_{((N_{prev}+1):N_{curr})(t-1)})$ ▷ See Step 1 of Algorithm 3.

  if $\max(\mathfrak{D}) > dmax_t^{(prev)}$ **then**

    $ChangeMM = 1$; ▷ if there is a new furthest point

    break

  else

    $R^{(curr)}_{((N_{prev}+1):N_{curr})t} = \{dist\{\mathbf{x}^{(prev)}_{m_t^{(prev)}}, \mathbf{x}^{(ch)}_1\}, .., dist\{\mathbf{x}^{(prev)}_{m_t^{(prev)}}, \mathbf{x}^{(ch)}_{N_{ch}}\}\}$

  end if

end for

Step 2: If $ChangeMM = 1$ at t^{th} MM point, $m_t^{(prev)}$, then recompute next $M^{(curr)}_{t:k'}$, $Dmax^{(curr)}_{t:k'}$, $R^{(curr)}_{\bullet(t:k')}$

if $ChangeMM = 1$ **then**

  $M^{(curr)}_{t:k'} \leftarrow 0; \quad Dmax^{(curr)}_{t:k'} \leftarrow 0; \quad R^{(curr)}_{\bullet(t:k')} \leftarrow 0$

  $\mathfrak{D} = min(R^{(curr)}_{\bullet(1:t-1)})$

  while $t \leq k'$ **do**

    $\mathfrak{D} = min(\mathfrak{D}, R^{(curr)}_{\bullet(t-1)})$

    $m_t^{(curr)} = \underset{1 \leq j \leq N_{curr}}{\arg\max} \{\mathfrak{D}_j\}$

    $dmax_t^{(curr)} = d_{m_t^{(curr)}}$

    $R^{(curr)}_{\bullet t} = \{dist\{\mathbf{x}^{(curr)}_{m_t^{(curr)}}, \mathbf{x}^{(curr)}\}, .., dist\{\mathbf{x}^{(curr)}_{m_t^{(curr)}}, \mathbf{x}^{(curr)}\}\}$

    $t \leftarrow t + 1$

  end while

end if

Step 3: Group each object in $X^{(curr)}$ with its nearest MM point in $M^{(curr)}$

$S_1^{(curr)} = S_2^{(curr)} = ... = S_{k'}^{(curr)} = \emptyset$

for j ← 1 to N_{curr} do

  $t = \underset{1 \leq j \leq k'}{\arg\min} \{dist\{\mathbf{x}^{(curr)}_{m_j^{(curr)}}, \mathbf{x}_t^{(curr)}\}\}$

  $S_t^{(curr)} = S_t^{(curr)} \cup \{l\}$

end for

$n_t = \left\lceil n * |S_t^{(curr)}|/N_{curr} \right\rceil; \quad \mathscr{N}^{(curr)} = \{n_1, , ..., n_{k'}\}$

Output: $M^{(curr)}, R^{(curr)}, Dmax^{(curr)}$, $\mathscr{N}^{(curr)}$, and $S^{(curr)}$

previous MMRS sample, $\tilde{S}_t^{(prev)}$, and are still associated with the same (t-th) group of the current dataset $X^{(curr)}$, $S_t^{(curr)}$ (not to be confused with MMRS sample $\tilde{S}_t^{(curr)}$, which is yet to be computed). Let Z_t denotes the data points which are present in both $\tilde{S}_t^{(prev)}$ and $S_t^{(curr)}$. If the number of data points in $S_t^{(curr)}$ that are already present in $\tilde{S}_t^{(prev)}$, denoted by $|Z_t|$, are greater than the number of data points required to build the t-group of the current MMRS sample $\tilde{S}_t^{(curr)}$, n_t, i.e., if $|Z_t| > n_t$ then we require $|Z_t| - n_t$ random data points to be deleted from $\tilde{S}_t^{(prev)}$ to obtain $\tilde{S}_t^{(curr)}$. If $|Z_t| < n_t$, then we require $n_t - |Z_t|$ random data points from $S^{(curr)}$, which are not present in $\tilde{S}_t^{(prev)}$, to be added in $\tilde{S}_t^{(prev)}$ to obtain $\tilde{S}_t^{(curr)}$. Let $\mathscr{W}^{(delete)}$ and $\mathscr{W}^{(add)}$ contain the indices of these (random) elements in $X^{(curr)}$ which are required to be deleted from or added to $\tilde{S}^{(prev)}$, respectively, to obtain $\tilde{S}^{(curr)}$.

Step 5: In this step, we delete the $\mathscr{W}^{(delete)}$ elements one by one and update the previous VAT/iVAT outputs using dec/VAT and dec-iVAT algorithms [69]. For each element $w_i^{(delete)} \in \mathscr{W}^{(delete)}$, we first find the $w_i^{(delete)}$-th element of $X^{(curr)}$, in the current sample $\tilde{S}^{(curr)}$, and subsequently, find its position, j, in the previous VAT reordering indices, $P^{(prev)}$. Then, the dec-VAT algorithm is used to delete the j-th element in $P^{(prev)}$, and subsequently, to obtain current (updated) VAT outputs, $P^{(curr)}$, $F^{(curr)}$, $h^{(curr)}$, and $D_n^{*(curr)}$. Then, the current iVAT reordered dissimilarity matrix $D'^{*(curr)}_n$ is obtained using dec-iVAT algorithm. Also, the $w_i^{(delete)}$-th element is deleted from $\tilde{S}^{(prev)}$ to obtain $\tilde{S}^{(curr)}$. Since, the j-th element is deleted, the values of $P^{(curr)}$, which are greater than j, are decreased by 1.

Step 6: In this step, we insert the new elements $\mathscr{W}^{(add)}$ one by one, and update the current (from the last step) VAT/iVAT outputs using inc-VAT and inc-iVAT algorithms [69]. For each element $w_i^{(add)} \in \mathscr{W}^{(add)}$, its distances to all the data points in $\tilde{S}^{(curr)}$ are computed and reordered using the indices in $P^{(curr)}$, to obtain L (refer to inc-VAT description in Chapter 2). Then, using L and VAT outputs from the last step, the inc-VAT algorithm is used to insert $w_i^{(add)}$-th element of $X^{(curr)}$ in $P^{(curr)}$ to obtain updated $P^{(curr)}$, $F^{(curr)}$, $h^{(curr)}$, and $D_n^{*(curr)}$. Subsequently, the updated iVAT reordered dissimilarity matrix $D'^{*(curr)}_n$ is obtained using inc-iVAT algorithm. The current MMRS sample $\tilde{S}^{(curr)}$ is updated by adding $w_i^{(add)}$-th element of $X^{(curr)}$.

Step 7: For the next incoming chunk, the current data $X^{(curr)}$ acts as the previous data $X^{(curr)}$, so $X^{(curr)} \leftarrow X^{(prev)}$. Similarly, the current MMRS and VAT outputs computed for $X^{(curr)}$ act as previous outputs for processing the next incoming chunk.

The image of the current iVAT reordered dissimilarity matrix $D'^{*(curr)}_n$, $I(D'^{*(curr)}_n)$, provides the visualization of evolving cluster structure in $X^{(curr)}$. For each incoming chunk, Steps $3-7$ are performed to delete and/or insert the MMRS data points from/to the previous MMRS samples, and subsequently, to obtain the iVAT image $I(D'^{*(curr)}_n)$ for the current data $X^{(curr)}$.

The inc-siVAT algorithm is a landmark window based approach which considers the data from the beginning until the current time instant. So, the number of data points in current data, N_{curr}, increases with the arrival of each new chunk which may slow down the computation process over the time due to memory constraints. This problem can be solved by adopting a sliding window approach with the existing landmark window approach of inc-siVAT. Let N_{mem} denotes the number of data points that can be accommodated by an allowable memory without significantly slowing down the computation process. When $N_{curr} > N_{lm}$, the oldest N_{ch} (or as per user selection) data points and corresponding MMRS points are deleted using dec-VAT/dec-iVAT algorithm to accommodate new N_{ch} number of data points of $X^{(ch)}$ in the inc-MMRS algorithm.

Anomaly Detection

In data clustering, clusters that are too far from the main clusters, or have only a few data points, are considered as anomalies or outliers. In inc-siVAT, we use the clustering based concept as used in [69] for anomaly detection in streaming data. In (optional) Step 8 of the inc-siVAT algorithm, clusters in $\tilde{S}^{(curr)}$ are obtained by cutting the MST using cut threshold magnitudes [252] ordered by edge distances $h^{(curr)}$ in the MST. The cluster boundaries are defined by those indices t, which satisfy

$$h_{n_t} > \alpha \times mean(h^{(curr)}), \tag{6.1}$$

where $\alpha \geq 1$ is a user-defined parameter that controls how far two groups of data points should be from each other to be considered as separate clusters. Smaller values of α represent tighter cluster boundaries, while large values of α create loose cluster boundaries. So, we cut those edges of the MST, given by $h^{(curr)}$ that satisfy (6.1), to obtain k-aligned partitions of $\tilde{S}^{(curr)}$. The k-partitions of the MMRS $\tilde{S}^{(curr)}$ are non-iteratively extended to the remaining (non-sampled) $N-n$ objects in $X^{(curr)}$ using the NPR.

A partition in MMRS sample $\tilde{S}_t^{(curr)}$ is considered as an anomalous set of points if it satisfies the following condition:

$$|\tilde{S}_t^{(curr)}| < \beta \times n, \tag{6.2}$$

where $|\tilde{S}_t^{(curr)}|$ is the number of data points in partition $\tilde{S}_t^{(curr)}$, $0 \leq \beta \leq 1$ is a user-defined parameter, and n is the number of data points in $\tilde{S}^{(curr)}$. Large values of β cause even large groups of data points that are far from other partitions to be regarded as anomalous. As β decreases, the same anomalous partitions eventually becomes part of the normal partitions, and only isolated data points or a partition of few data points remain anomalous. The procedure to choose an optimal value of α and β is described in [69, 252].

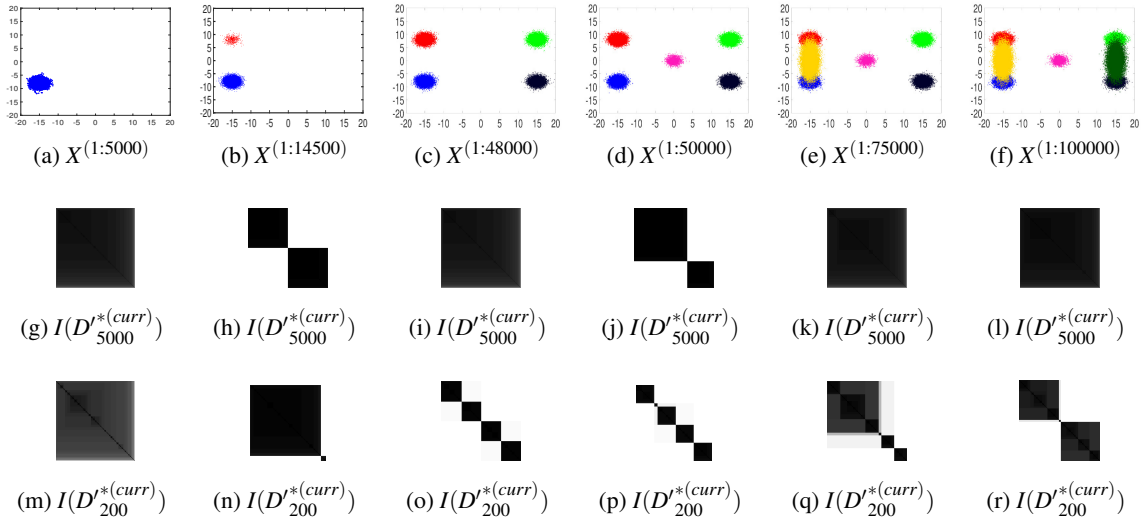
6.4 Experiments

We performed three sets of experiments. In the first experiment, we illustrate the efficacy of the inc-siVAT algorithm to facilitate cluster evolution analysis in high-velocity, big data streams. In the second experiment, we compare siVAT and inc-siVAT algorithm based on their run-time performance on the large streams of big data. In the last experiment, we demonstrate the applicability of the inc-siVAT algorithm for anomaly detection in big streaming data. The experiments were performed using MATLAB on a Windows 7 (64 bit) PC with 16 GB RAM and Intel i7 @3.40 GHz processor.

Unless otherwise mentioned, the inc-siVAT parameters are, $k' = 30$, $n = 200$, and $initN = N_{ch} = 500$. Since, the inc-siVAT displays an incrementally built iVAT image every time after a new chunk arrives, it is not possible to show all the images here.

6.4.1 Cluster Evolution Analysis in Big, Streaming Data

In this experiment, we illustrate the effectiveness of the inc-siVAT algorithm to demonstrate the evolving cluster structures in streams of the two big datasets that evolve significantly over time.



(a) $X^{(1:5000)}$ (b) $X^{(1:14500)}$ (c) $X^{(1:48000)}$ (d) $X^{(1:50000)}$ (e) $X^{(1:75000)}$ (f) $X^{(1:100000)}$

(g) $I(D'^{*(curr)}_{5000})$ (h) $I(D'^{*(curr)}_{5000})$ (i) $I(D'^{*(curr)}_{5000})$ (j) $I(D'^{*(curr)}_{5000})$ (k) $I(D'^{*(curr)}_{5000})$ (l) $I(D'^{*(curr)}_{5000})$

(m) $I(D'^{*(curr)}_{200})$ (n) $I(D'^{*(curr)}_{200})$ (o) $I(D'^{*(curr)}_{200})$ (p) $I(D'^{*(curr)}_{200})$ (q) $I(D'^{*(curr)}_{200})$ (r) $I(D'^{*(curr)}_{200})$

Figure 6.2: 2D data scatterplots (first row) and (incrementally built) inc-iVAT (second row) and inc-siVAT (last row) images of a big, streaming data X at N_{curr} = 5000, 12500, 48000, 50000, 75000, and 100000 data points.

6.4.1.1 2D Synthetic Data Experiment

In this experiment, we compare inc-siVAT images to inc-iVAT images to demonstrate the superiority of the inc-siVAT to provide visualizations of evolving cluster structures in large data streams of a 2D synthetic dataset. The inc-iVAT algorithm handles input data streams by considering one data point at a time, whereas, the inc-siVAT algorithm handles input data streams in chunks by considering one chunk at a time. Only the most recent 5000 data points are considered in inc-iVAT to obtain a current iVAT image, due to its limitation for $N > 5000$, whereas, the inc-siVAT uses the entire data from the beginning until now, and summarizes it to a small size (n) MMRS sample, and subsequently, constructs its current iVAT image.

The 2D synthetic dataset X, having $100,000$ data points, is constructed by drawing samples from a mixture of seven Gaussian distributions, as shown in Fig. 6.2 (in the first row). The first $50,000$ data points in X are extracted from the five different Gaussian distributions, four of them having $12,000$ points each and one having $2,000$ data points, i.e., $|X_1| = |X_2| = |X_3| = |X_4| = 12,000$, and $|X_5| = 2,000$. These five clusters are shown with different clusters in Fig. 6.2(d). The next $50,000$ data points are generated from two different Gaussian distributions, with equal number ($25,000$) of data points from each of them, such that they form the bridge between two pairs

of clusters, X_1 and X_2, and X_3 and X_4, respectively, as shown with yellow and dark green color in Fig. 6.2(f). For a demonstration of evolving clusters, the input data X is taken as data streams, in which data points are arranged according to the cluster they belong. So, the first $12,000$ points belong to the cluster X_1, the next $12,000$ points belong to the cluster X_2, and so on. Views (a-f) in Fig. 6.2 show scatterplots of 2D data X at six different instants (in the first row), and views (g-i) and views (m-r) show corresponding (incrementally built) inc-iVAT (in the second row) and inc-siVAT images (in the last row), respectively.

Fig. 6.2(a) shows the scatterplot of the first $5,000$ data points, all belonging to cluster X_1. The presence of a single dark block in corresponding inc-iVAT and inc-siVAT images in views (g) and (m) confirm the presence of a single cluster in the current data, $X^{(1:5000)}$. Fig. 6.2(b) shows the scatterplot of the first $14,500$ data points, in which first $12,000$ points belong to cluster X_1 and the next $2,500$ points belong to cluster X_2. Although, both inc-iVAT and inc-siVAT images (views (h) and (n)), present two dark blocks confirming the two clusters in current data $X^{(1:14500)}$, the size of the dark blocks in both the images is different. Since the inc-iVAT considers the most recent 5000 data points, $X^{(9500:14500)}$, its image shows two dark blocks of equal size, proportional to the (equal) number of points from cluster X_1 and X_2 i.e., $2,500$ points from each cluster in $X^{(9500:14500)}$. Whereas, the inc-siVAT summarizes the entire data $X^{(1:14500)}$ in its MMRS sample ($n = 200$), so its image shows one big dark block corresponding to $12,000$ data points of cluster X_1 and one small dark block corresponding to $2,500$ data points from cluster X_2, in current data $X^{(1:14500)}$.

Fig. 6.2(c) shows the scatterplot of the first $48,000$ points, having $12,000$ points from each of the four clusters, X_1, X_2, X_3 and X_4. A single dark block in its inc-iVAT image (view (i)) indicates only a single cluster corresponding to points belonging to cluster X_4 in $X^{(43000:48000)}$, without showing four clusters that evolved over time in data streams of $X^{(1:48000)}$. Whereas, the four dark blocks (of the same size) in its inc-siVAT image confirms the presence of the four equal size clusters in the current data.

Fig. 6.2(d) shows the scatterplot of the first $50,000$ points, in which 2000 data points of cluster X_5 are added to the previous $48,000$ data points. Unlike inc-iVAT image which shows only two dark blocks corresponding to 3000 data points from cluster X_4 and 2000 data points from cluster X_5 in $X^{(45000:50000)}$, the inc-siVAT image shows five dark blocks including a tiny dark block

corresponding to recently evolved (small size) cluster X_5.

Fig. 6.2(e) shows the scatterplot of the first $75,000$ points, in which $25,000$ new data points are added (shown in yellow) to $X^{(1:50000)}$ from a different Gaussian distribution such that they create an overlap between cluster X_1 and X_2 forming one bigger cluster. The inc-iVAT image in view (k) indicates only a single cluster corresponding to the recent 5000 data points from the same distribution. The inc-siVAT image shows one big, one tiny, and two small dark blocks. Two small dark blocks correspond to cluster X_3 and X_4, a tiny dark block represents cluster X_5, and the big dark block corresponds to the bigger cluster which formed after adding $25,000$ new data points to the previous dataset. The three small sub-blocks inside the big dark block correspond to the cluster X_1, X_2 and the points from the new Gaussian distribution, showing an overlap (similarity) among them.

Finally, Fig. 6.2(f) shows the scatterplot of all the $100,000$ data points, in which $25,000$ new data points are added to $X^{(1:75000)}$ from a different Gaussian distribution such that they form an overlap between cluster X_3 and X_4. Similar to the previous example, the inc-iVAT image indicates only a single cluster corresponding to the last 5000 data points from the same distribution. The inc-siVAT image shows two big dark blocks and a tiny dark block indicating a total of three clusters. The two big dark blocks (top left and bottom right in view (r)) correspond to the two big clusters (in left and right side of view (f)) that evolved after adding last $50,000$ data points from two different Gaussian distribution forming an overlap between two pair of clusters, X_1 and X_2, and X_3 and X_4, respectively. The tiny dark block represents a small cluster X_5 containing $2,000$ data points.

The overall conclusions that can be made from Fig. 6.2 are: (i) Due to the practical limitations of inc-VAT for large N, it can consider maximum 5000 most recent data points to form iVAT image for large data streams, whereas, the inc-siVAT summarizes the entire available data streams to a small size (typically, n is 100 to 500) MMRS sample to obtain a small size iVAT image; (ii) The inc-siVAT is superior to the inc-iVAT to provide the visualization of evolving cluster structures in large data streams. (iii) Another advantage of inc-siVAT over siVAT is that the ordering of dark blocks in output image may change when you apply siVAT after every chunk. Whereas, in, inc-siVAT, ordering of dark blocks remain same after every chunk.

Table 6.1: The number of data points in the four main clusters of KDD Cup'99 dataset.

Number of data points after	Normal	smurf attack	back attack	neptune attack
16th Chunk (8,000 pts)	**7,787**	207	0	2
25th Chunk (12,500 pts)	8,799	**3,695**	0	2
102th Chunk (51,000 pts)	38,174	11,258	**1,037**	2
108th Chunk (54,000 pts)	39,298	11,258	2,002	**419**
150th Chunk (75,000 pts)	41,188	11,258	2,002	**20.482**
680th Chunk (340,000 pts)	71,225	**220,561**	2,103	41,122

6.4.1.2 KDD Cup'99 Data Experiment

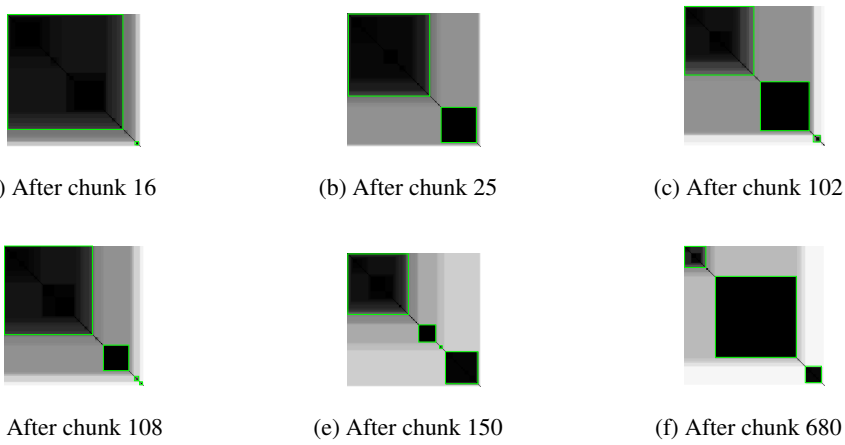
In this experiment, we illustrate inc-siVAT on the streams of KDD Cup'99 dataset, which has been used earlier in several studies [64, 67, 68, 138] for streaming data clustering. This dataset corresponds to an important problem of automatic and real-time detection of cyber attacks, that evolve significantly over time in the streams of this dataset. Therefore, it is challenging to track evolving clusters from the dynamic streams of this dataset.

KDD Cup'99 is a big, labeled dataset that specifies attack types (normal or attack). It consists of $494,021$ instances of 41 dimensional vectors[1], and each vector is labeled to specify the attack type. It has 23 labeled subsets, a normal subset and 22 simulated attacks that fall into four main categories: DOS, R2L, U2R, and probing. As a result, data contains a total of five clusters including normal connections. Some of these 22 attacks are neptune, back, smurf, pod, land, butter-overflow, rootkit, spy, imap, nmap, and so on. Most of the connections in the streams of this dataset are normal, but occasionally there could be a burst of attacks at certain times.

Fig. 6.3 shows the inc-siVAT images for KDD Cup'99 data streams at six different time instants viz. after 16th, 25th, 102th, 108th, 150th, and after 680th chunk ((each chunk having a size of 500 data points). We use the ground truth labels of this dataset to validate the cluster structures suggested by the inc-siVAT images. Table 6.1 shows the number of data points for four dominant categories of this dataset, namely, normal connections, smurf, back, and neptune attack, after six different time instants.

After 16 chunks, $7,787$ data points belong to the normal connections, 207 data points belong to the smurf attack, and the remaining 6 points belong to other attacks. The inc-siVAT in view

[1]We normalized all 41 features to the interval [0;1] by subtracting the minimum and then dividing by the subsequent maximum so that they all had same scale.



(a) After chunk 16 (b) After chunk 25 (c) After chunk 102

(d) After chunk 108 (e) After chunk 150 (f) After chunk 680

Figure 6.3: inc-siVAT images visualizing evolving clusters in KDD Cup' 99 datastreams at different time instant

(a) of Fig. 6.3 confirms this by showing a big dark block (shown at top left in green rectangle) corresponding to the data points belonging to the normal connections, a small dark block (at bottom right) corresponding to the smurf attack, and several (hard to see) singleton dark blocks corresponding to other attacks.

Between the 16th and 25th chunk, most of the data points belong to the smurf attack. The evolution of a new cluster corresponding to the smurf attack also appears in the inc-siVAT image in view (b) where the bottom right dark block, whose size has increased considerably after view (a), suggests an increased number of data points belonging to smurf attack.

After 102th chunk, 1037 new data points belong to *back* attack while the number of data points belonging to the normal connection and smurf attack increases to $38,714$ and $11,258$, respectively. This evolution can also be seen in the inc-siVAT image in view (c) where the size of the dark block corresponding to the smurf attack increases and a new small dark block emerges (at the bottom right) corresponding to the new 1037 data points belonging to *back* attack.

After 108th chunk, 419 new data points belong to neptune attack, 2002 data points belong to *back* attack, and the number of data points belonging to normal connection increases to $39,298$. The inc-siVAT image in view (d) reflects this evolution showing an additional small dark block corresponding to new 419 data points belonging to the neptune attack, and the increased size of the top left dark block indicates the increased number of data points belonging to the normal connection.

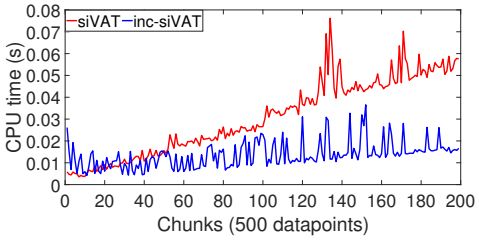
Almost all the data points between 109th and 150th chunk belong to the neptune attack resulting in total $20,482$ data points belonging to the neptune attack, $41,188$ data points belonging to the normal connection, and $11,258$ data points belonging to the smurf attack, after 150th chunk. This evolution also appears in the inc-siVAT image in view (e) where the biggest dark (at top left) block corresponds to the data points belonging to normal connections, the second biggest dark block (at bottom right) corresponds to the neptune attack, and a small dark block (in the middle) corresponds to the data point belonging to smurf attack.

Between 150th to 680th chunk, the number of data points belonging to smurf attack significantly increases to $220,561$, surpassing the data points belonging to normal connection ($71,225$) and neptune attack ($41,122$). The biggest dark block in the inc-siVAT image in view (f), which emerges in the middle, indicates that majority of the data points after 680 chunks belong to the smurf attack. The smallest dark block in view (e) (after 150th chunk) is now the biggest dark block in view (f), and the two biggest dark blocks in view (e) now appear as the two small dark blocks in view (f) suggesting that the cluster structure has drastically changed after 680 chunks. This illustrates the capability of inc-siVAT to provide the visualization of evolving cluster structures in dynamic, rapidly arriving data streams.

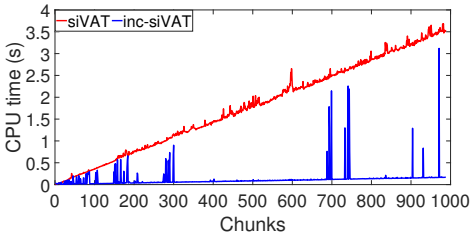
6.4.2 Time Comparison

To illustrate the time complexities of the siVAT and inc-siVAT algorithm, we perform an experiment on the same two datasets that we considered in the last experiment. We randomize the rows of the 2D synthetic dataset so that data points belonging to the same cluster are not adjacent anymore. At the arrival of every new chunk, the siVAT and inc-siVAT algorithm is applied on the current data $X^{(curr)}$, and their CPU time to compute the reordered dissimilarity matrices is recorded.

Fig. 6.4 shows the time comparison between siVAT and inc-siVAT algorithm for 2D synthetic and KDD Cup'99 datasets. The CPU time of the inc-siVAT increases most with every chunk, whereas, the CPU-time of the inc-siVAT algorithm is always less than that of siVAT. This is because siVAT needs to be retrained everytime a new chunk arrives, whereas, the inc-siVAT incrementally updates the MMRS sample after every chunk using the inc-MMRS algorithm. MM points are updated only when a change in the previous MM points is identified by the inc-MMRS algo-



(a) Synthetic Data: $N = 100,000$, $p = 2$



(b) KDD Data: $N = 494,021$, $p = 41$

Figure 6.4: Time comparison of siVAT and inc-siVAT for high-dimensional synthetic and KDD datasets.

rithm after a new chunk arrives. Subsequently, an updated MMRS sample and its (incrementally built) inc-siVAT image are produced using inc-VAT/dec-VAT and inc-iVAT/dec-iVAT.

The spikes in CPU-time plots for the inc-siVAT algorithm correspond to the changes in the MMRS sample distribution after a new chunk arrives. A higher number of spikes corresponds to the higher number of changes in the MMRS sample. And, a high magnitude spike in the CPU-time plot of inc-siVAT algorithm (usually) corresponds to a significant change in the MMRS sample distribution after a new chunk arrives. There are more spikes in CPU-time plot of the inc-siVAT algorithm for 2D synthetic dataset compared to KDD Cup'99, whereas there are high amplitudes spikes (but, low in numbers) in KDD Cup'99 dataset. This means that there were frequent changes in the MMRS sample after every chunk of the 2D synthetic dataset, and there were a few but significant changes in the MMRS sample distribution for the KDD Cup'99 dataset. This is probably because most of the chunks in KDD Cup'99 dataset belong to either normal connections, smurf, or neptune attacks that resulted in stable streams, but occasionally there were a burst of attacks at certain times (chunks), resulting in a sudden change in cluster distribution.

6.4.3 Anomaly Detection

In this experiment, we demonstrate the applicability of the inc-siVAT algorithm for anomaly detection in large, streaming data. This experiment is performed on three real datasets, including two unlabeled datasets that have been used in previous studies for anomaly detection [69, 253].

6.4.3.1 MiniBooNE particle identification data (MPID) experiment

This dataset consists of 130,064 records, and each record has 50 attributes. The samples of MPID dataset are divided into 36,499 signal events of electron neutrinos and 93,565 background events of muon neutrinos. For MPID, the parameters chosen for anomaly detection are: $\alpha = 50$ and $\beta = 0.02$.

The streams of this dataset are relatively stable compared to the KDD dataset. So, the inc-siVAT images do not change much across the chunks. Fig. 6.5 (a) shows the inc-siVAT (incrementally built) image for $\text{MPID}_{1:130500}$ (first 130500 data points). The two dark blocks (one big and one tiny) along the diagonal indicates the two clusters in MPID data. The MST cut magnitude, $h_n^{(curr)}$, plot for MPID dataset is shown in Fig. 6.5 (d) with a red horizontal line showing the cut threshold value of $\alpha \times mean(h_n^{(curr)})$. A group of the data point(s), for which $h_n^{(curr)}$ is greater than the cut threshold, are possible candidates for anomalies. Among them, the cluster whose size is less than $\lceil \beta \times n \rceil = \lceil 0.02 \times 200 \rceil = 2$ is declared as anomalous. Therefore, the big dark block in Fig. 6.5 (a) corresponds to a normal cluster (shown within a green rectangle) and the small dark block, containing only one data point, corresponds to the anomaly. It is hard to see the single red pixel at the bottom right corner of the image, so we have circled it for emphasis. Some previous researches [253, 254] confirmed an anomaly in MiniBooNE dataset correspond to a low neutrino energy signal in this dataset.

6.4.3.2 US Census 1990 data experiment

The US Census 1990 data is an example of a real-world unlabeled, big data which consists of 2458285 records. Each record has 68 demographic and employment-related attributes, such as age, gender, *place of birth* (POB), income, etc. For this experiment, we have set the parameters: $\alpha = 2$ and $\beta = 0.02$.

Figs. 6.5 (b) and (e), respectively, show the inc-siVAT image and $h_n^{(curr)}$ plot (with cut threshold) for $\text{Census}_{1:2,000,000}$. The inc-siVAT image suggests three dark blocks along its diagonal, among them two blocks (one big and one small) correspond to the normal clusters, and the tiny dark block, shown within the red circle, correspond to anomalies. Since it is an unlabeled data, it was harder to find what class or attribute these anomalies correspond to in the data. However,

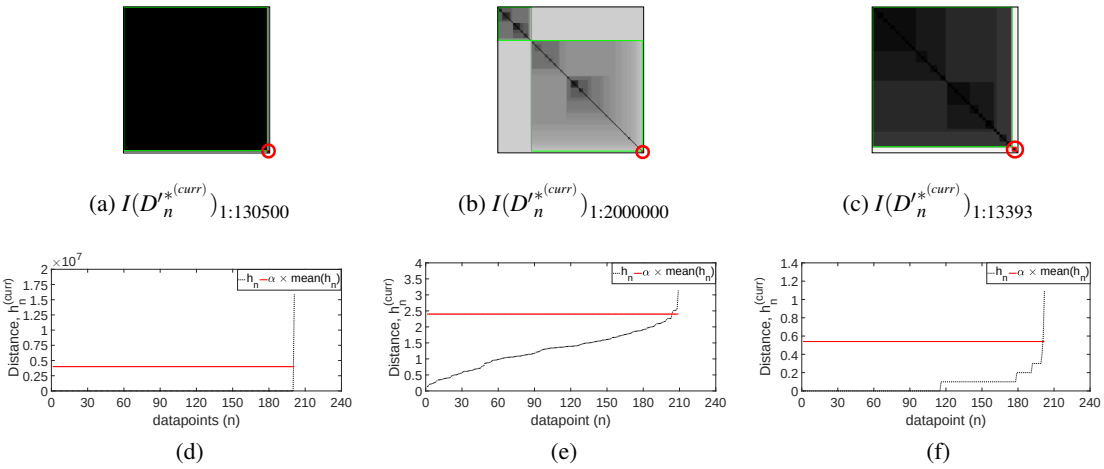


Figure 6.5: $h_n^{(curr)}$ plot and inc-siVAT images showing normal and anomalous data points for (a,d) MiniBoone; (b,e) US Census 1990l and (c,f) Heron Island Dataset

through eyeballing on the attributes of these anomalous data points, we could found out that these anomalous data points correspond to the place of birth attribute, as out of $2,458,285$ total census records (people), there are only (lowest) 3286 records (or people) whose place of birth is in one of the African countries, and they were living in the USA (in 1990).

6.4.3.3 Heron Island Data Experiment

This experiment is performed on a real-life dataset collected from the Heron Island weather station deployed on the Great Barrier Reef, Australia [86]. This is an environmental dataset that has three variables: air humidity, air pressure, and air temperature. The data were collected from 1st January 2009 to 2nd April 2009, every ten mins. For this experiment, we have set the parameters: $\alpha = 2.5$ and $\beta = 0.02$.

The inc-siVAT finds two anomalies in this data that appear as two tiny dark blocks at the bottom right corner of the image, as shown in Fig. 6.5 (c). A large zoom is required to see the smallest dark block in the image in view (c). These two anomalies correspond to the data points collected on 5th and 9th March 2009. Previous studies [69, 86] on Heron island data identifies these anomalies correspond to an unusual weather variation on 5th March and to Hamish Cyclone on 9th March. The big dark block (cluster) in view (c) corresponds to normal weather.

6.5 Summary

We proposed and developed an *incremental version of the scalable iVAT* algorithm, inc-siVAT, for online visual assessment of evolving cluster structures in high-velocity, massive data streams. The inc-siVAT algorithm uses Maximin Random Sampling (MMRS) scheme on initial data points to extract a small size smart sample. Then, it incrementally updates the smart sample points on the fly to reflect changes in data streams, using our novel incremental MMRS algorithm, inc-MMRS, and subsequently, it produces a reordered dissimilarity image (RDI) which is built incrementally using inc-VAT/inc-iVAT and dec-VAT/dec-iVAT algorithms.

We have demonstrated the effectiveness of the inc-siVAT to visualize evolving cluster structure in dynamic streams of a synthetic and KDD Cup'99 dataset. We have also shown that time-complexity of the inc-siVAT algorithm is less than the siVAT algorithm for each chunk of the input data streams. The inc-MMRS summarizes the available data points to a fixed (small) size of an intelligent sample, hence, it requires less memory to store and/or visualize the (small size) RDI image. We have also shown the applicability of the inc-siVAT algorithm for anomaly detection in large streams of the three real datasets including an IoT dataset.

Chapter 7

A Scalable Framework for Trajectory Prediction

This chapter demonstrates big data clustering for a real-world application. Specifically, a novel, hybrid framework, based on a scalable clustering, called Traj-clusiVAT, and Markov chain models, is presented for vehicle trajectory prediction, which is suitable for a large number of overlapping trajectories in a dense road network, typically for major cities around the world. The proposed framework is compared with a mixed Markov model (MMM)-based and a NETSCAN-based trajectory prediction model on two real-life, large-scale trajectory datasets. The short-term and long-term trajectory prediction performance of the proposed framework is found to be better in terms of prediction accuracy and distance error.

7.1 Introduction

With the widespread use of *Global Positioning System* (GPS) devices, smart-phones, sensor network, and wireless communication technology, it is possible to track all kinds of moving objects all over the world. The increasing prevalence of location-acquisition technologies has resulted in large volumes of spatio-temporal data, especially in the form of trajectories. These data often contain a great deal of information [184], which give rise to many location-based services (LBSs) and applications such as vehicle navigation, traffic management, and location-based recommendations. One key operation in such applications is the route prediction of moving objects.

Vehicle route prediction allows certain services to improve their quality, e.g., if the route of vehicles is known in advance, *intelligent transportation systems* (ITSs) can provide route-specific traffic information to drivers such as forecasting traffic conditions and routing the driver so as to

avoid traffic jams. Route prediction also enables location-based advertising, which can advertise certain products/services and special offers to the target commuters most likely to pass through business outlets and stores based on their travel trajectory.

Recently, several studies have been carried out on *trajectory prediction* (TP), particularly after Song *et al.* [193] demonstrated a 93% potential for predictability in user mobility, which supplied the theoretical basis for location prediction methods. These methods mainly focus on two kinds of prediction models. The first type is the short-term trajectory prediction model, which aims to predict the next-location or a few locations in the near future. These models usually rely on current location and one or two previous locations of an object to predict its next location. The second type is the long-term trajectory prediction model which focuses on location prediction at a more distant future time or on complete route prediction. These models generally rely on an available partial trajectory of a moving object to predict the complete trajectory.

In urban areas, vehicle trajectories are usually constrained to a complex road network with many parallel and perpendicular road segments and intersections, which makes their time progression very irregular. Due to the uncertainty of moving objects, most of the existing TP methods only focus on predicting short-term partial trajectories. They have poor prediction accuracy for long-term trajectory predictions, and they do not work well for estimating continuous and complete trajectories. Moreover, traditional distance-based TP methods can only be applied to predict possible routes within fixed constrained roadways, and they do not provide optimal routes for complex road networks.

The sheer amount of vehicle trajectory data, if analyzed effectively, can significantly improve route prediction performance. However, it is challenging to carry out trajectory prediction from a large amount of trajectory data. The huge volumes of data to be processed precludes using machine learning based TP methods. Existing TP methods are hybrid in nature and usually use classical frequent sequential pattern based algorithms, Markov model-based algorithms, or clustering based algorithms. Most of them cannot handle a large number of trajectories, especially when they span a large area of a road network. Therefore, most TP methods demonstrated in the literature use synthetic or small to medium size real trajectory datasets. Section 7.2 discusses existing TP methods and their limitations for large-scale trajectory data.

To address these challenges and overcome the drawbacks of existing TP methods, this chapter

presents a novel, scalable framework for both short-term and long-term TP, which is suitable for large numbers of overlapping trajectories in a dense road network, typical for major cities around the world. First, we cluster the large trajectory data using a modified version of two-stage clusiVAT (clustering using improved Visual Assessment of Tendency) algorithm [63], which we call *Traj-clusiVAT*, implemented for trajectory prediction task. The Traj-clusiVAT algorithm first extracts a smart sample using the *Maximin-Random sampling* (MMRS) scheme [41], which provides a good representation of input cluster structure (present in the original data). Then, it uses the iVAT algorithm to visually determine the number of clusters (k) in input data, and subsequently, it partitions the trajectory sample into k clusters which contain different frequent movement patterns in the trajectory data. Then, the remaining non-sampled trajectories are assigned to one of k clusters using the *nearest prototype rule* (NPR). Finally, Markov chain models are constructed from the trajectories in each cluster. These models quantify the movement patterns within clusters, and subsequently, can be used for TP.

7.2 Related Work

Several studies address the problem of trajectory prediction, which includes the problem of short-term prediction such as predicting the next location, and long-term prediction such as future locations or complete route prediction. These methods mainly focus on discovering frequent patterns using various data mining methods. Many of these methods are hybrid and can be broadly classified into three categories: (i) Rule-based learning based approaches (ii) Markov model-based approaches (iii) Clustering-based approaches.

7.2.1 Rule-based learning based approaches

Several rule-based methods have been used for location prediction. Morzy [194] implemented a modified version of the PrefixSpan algorithm to extract association rules from a moving object database, and used frequent pattern tree with a matching function to select the best association rule from the database of movement rules. Jeung *et al.* [195] proposed a hybrid prediction approach, which combines association rules in the form of trajectory patterns with the motion functions of an object's recent movements, to estimate future locations. Given an object's recent movement

and predictive queries, the best association rule is chosen for prediction. The query processing approaches presented in [195] can only support near and distant-time predictive queries, unsuitable for long-term trajectory prediction. Moreover, with the huge number of trajectories, the number of association rules is also huge, which makes association-rule based algorithms impractical for large-scale mobility data.

Monreale *et al.* [255] built a decision tree that they called a T-pattern Tree, based on the frequent movement patterns extracted using a *Trajectory Pattern* algorithm, and predicted the next location of a new trajectory based on the best matching functions. However, mining of frequent trajectory patterns is computationally expensive. The method in [255] is similar to the use of association rules as predictive rules in rule-based classifiers. Therefore, this method [255] may result in a large number of predictive rules for voluminous trajectories. Qiao *et al.* [206] proposed a TP algorithm, called PrefixTP, which examines only the prefix subsequences, and projects their corresponding postfix subsequences into projected sets. Then, for a partial trajectory, it recursively finds a postfix sequence based on the minimum support count requirement and then declares the most frequent sequential pattern as the most probable trajectory. Finding subsets of trajectory sequential patterns is a recursive mining process, which is also computationally extensive.

7.2.2 Markov model-based approaches

Markov models (MMs) have been widely used to mine frequent patterns for route prediction problems. Ishikawa *et al.* [196] proposed a model to extract mobility statistics, called the Markov transition probability, which is based on a cell-based organization of target space and a Markov chain model, and employed R-tree spatial indices to compute Markov transition probabilities. Simmon *et al.* [197] presented a *Hidden Markov Model* (HMM) based probabilistic approach to predict a driver's intended route and destination through observations of the driver's habits. Asahara *et al.* [87] suggested that standard MM and HMM are not generic enough to encompass all types of movement behaviour. They proposed a variant of Markov model, called the *mixed Markov-chain model* (MMM), as an intermediate model between individual and generic models, for pedestrian movement prediction. Gambs *et al.* [198] extended a previously proposed mobility model, named *v-Mobility Markov Chain* (v-MMC), to incorporate the v previous visited locations. They showed that prediction accuracy increases with v, but increasing v beyond two does not

compensate for the significant overhead in terms of computation and space for learning and storing the mobility model. They only considered the sequence of the significant locations, instead of all locations, to build higher order MM.

Most of the MMs do not consider the discontinuous chain of the hidden states, and therefore, the state retention problem can drastically degrade the accuracy of location prediction system [206]. For the irregular trajectory data, the movement rules cannot be easily represented by Markov models, which may cause loss of continuous location information [206]. Moreover, the HMM approaches use the Baum-Welch algorithm for parameter learning and the Viterbi algorithm to find the most likely sequences of hidden states. These algorithms impose a significant computation burden for large-scale trajectory datasets.

7.2.3 Clustering based approaches

Some researchers have proposed trajectory clustering based route prediction methods, which partition the trajectories into several clusters representing different motion patterns based on the trajectory similarity. Various clustering approaches [201] using different methods and distance measures between trajectories have been proposed in the literature. Road network constrained trajectory clustering approaches can be classified into two broad categories. The first type uses the traditional clustering approaches such k-means and DBSCAN with specially designed distance measures [185, 188, 202, 203] for trajectories. The second category of algorithms [88, 190] cluster road segment vehicle frequencies based on density and flow.

Ashbrook *et al.* [199] presented a system that automatically detected the significant locations from GPS data using k-means clustering, and then incorporated these locations into an MM to predict the next location. Mathew *et al.* [200] presented a hybrid method for human mobility prediction, which first clusters location histories according to their characteristics, and then trains an HMM for each cluster. A poor prediction accuracy of 13.85% was obtained on a real, large-scale trajectory dataset using this method. Chen *et al.* [256] proposed a next-location prediction approach combining two clustering models, which cluster the objects based on the spatial locations and trajectories using a similarity metric, respectively, and then it trains a series of MMs with trajectories in each cluster.

Ying *et al.* [257] proposed an approach for predicting the next location based on geographic

and semantic features of user trajectories. This method requires the calculation of a semantic score for each candidate path, which generally incurs additional overhead when compared with other methods. A probabilistic TP model was proposed in [204] based on two mixture models, a *Gaussian Mixture Model* (GMM) and a *Variational Gaussian Mixture Model* (VGMM), optimized using the *Expectation Maximization* (EM) algorithm. Their method requires the prior selection of the number of Gaussian components and other distribution parameters. They evaluated their method on a small dataset, which consists of only 69 trajectories. Qiujian *et al.* [258] proposed a spatio-temporal prediction and a next-place prediction model based on an entropy-based clustering approach and HMMs.

Traditional clustering [185, 188, 202, 203] based prediction methods are not scalable for a large number of trajectories as distance matrix computation is time and space prohibitive. Most of them require the number of clusters to be known in advance, but in practice, it is often unknown, making it difficult for the user to choose the optimal number of clusters for location prediction. Furthermore, the clusters are determined by fixed rules. Some of the road network based clustering approaches [88, 190], though scalable, produce clusters having high intra-cluster variance, which span a large area of a road network.

Most of the work done in the area of trajectory prediction either use synthetic datasets [87, 194, 195, 205] or real datasets with small to medium numbers of data points [255, 256, 259]. Most of them cannot handle big trajectory datasets. There have been several attempts to demonstrate trajectory prediction on real data having a large number of samples. For example, [206] uses a real dataset consisting of 4.9 million trajectories (790 million GPS points) as a population, but only small subsets having a maximum $30,000$ trajectories are used in their experiments. The largest real dataset used was in [258], consisting of 37 million GPS points. They utilized [258] the MapReduce model in their implementation to handle large datasets.

In this chapter, experiments were performed on two real-life, taxi trajectory datasets including a large-scale taxi trajectory dataset consisting of 370 million GPS traces and 3.28 million passenger trips from $15,061$ taxis during the period of one month in Singapore. This was the first time TP has been performed on such a large number of real-life road network trajectories.

7.3 Preliminaries

In this section, we introduce some basic terms and definitions, which are required in the sequel.

7.3.1 Road Network and Trajectories

The road network is represented as an undirected graph

$$G_{RN} = (V, E), \tag{7.1}$$

comprising a set V of *intersections* or *nodes* of the road network with a set E of road *segments* or *edges*, $R_i \in E$ such that $R_i = (r_{i_a}, r_{i_b})$, where $r_{i_a}, r_{i_b} \in V$ and there exists a road between r_{i_a} and r_{i_b}. The edge R_i is given a weight equal to the length of R_i. For such a road network, we define the following:

Definition 7.1. *(**Trajectory**): A **trajectory** T of length l is a time ordered sequence of road segments (RS), $T = \langle R_1, R_2, ..., R_l \rangle$, where $R_j \in E, 1 \leq j \leq l$, and R_j and R_{j+1} are connected.*

Definition 7.2. *(**Sub-Trajectory**): $T^s = \langle L_1, L_2, .., L_p \rangle$ is a **sub-trajectory** of sequence $T = \langle R_1, R_2, .., R_l \rangle$, $p \leq l$, if there are integers $\langle i_1, i_2, ..i_p \rangle$ $(1 \leq i_1 < i_2 < ... < i_p)$, $\langle j_1, j_2, ..j_p \rangle$ $(1 \leq j_1, j_2 = (j_1 + 1), ..., j_p = (j_{p-1} + 1) \leq l)$, and $i_1 \leq j_1$, $L_{i_1} = R_{j_1}$, $L_{i_2} = R_{j_2}, .., L_{i_p} = R_{j_p}$. Then T^s is called a sub-trajectory of T, denoted by $T^s \sqsubseteq T$.*

Definition 7.3. *(**Frequent Road Segment**): A **Frequent road segment** (FRS), R_{FRS}, in a trajectory set is a segment that contains at least $MinT$ percentage of trajectories of the set passing through the segment, otherwise, the segment is labeled as "non-FRS". The percentage $MinT$ is a tunable parameter, and we call it the FRS threshold.*

Definition 7.4. *(**Partial Trajectory**): A **partial trajectory** T^p is a sub-trajectory of a given trajectory T if and only if their sequences start from the same segment.*

Definition 7.5. *(**Source Segment**): The segment from which a trajectory T originates is called the **Source Segment** (SS), R_{SS}, and the start node of T is called the **Source Node** (SN) of that trajectory. For a trajectory $T = \langle R_1, R_2, ..., R_l \rangle$, the road segment R_1 is R_{SS}. Node r_{1_a} is the SN, if R_2 has node r_{1_b}, else r_{1_b} is SN, where $R_1 = (r_{1_a}, r_{1_b})$, and $r_{1_a}, r_{1_b} \in V$.*

Table 7.1: Notations

Symbol	Definition
$\mathscr{T}$	The set of trajectories
T_i	The i^{th} trajectory of set $\mathscr{T}$
l_i	The length (or number of segments) of trajectory T_i
R_i	The i^{th} segment of trajectory T_i
N, n	number of trajectories in $\mathscr{T}$ and MMSR sample S, respectively
k, K	number of non-directional and directional clusters in S
$\mathscr{T}^j$	Set of trajectories in cluster j
N_j	Number of trajectories in cluster j
$\mathscr{R}^j$	Set of points (segments) in cluster j
$\mathscr{C}(\mathscr{T})$	Set of cluster of trajectories
$R_{FRS}, \mathscr{R}_{FRS}$	Frequent road segment (FRS) and the set of FRSs, respectively
$R_{SS}, \mathscr{R}_{SS}$	Source segment and the set of SSs, respectively
$R_{FSS}, \mathscr{R}_{FSS}$	Frequent source segment (FSS) and the set of FSSs, respectively
M^j	Transition probability matrix for cluster j
W^j	Transition count matrix for cluster j

Definition 7.6. *(**Frequent Source Segment**): The SS which is FRS, is called Frequent source segment (FSS), R_{FSS}.*

Definition 7.7. *(**Problem Definition**): Assume that a historical trajectory database, containing N trajectories, denoted by $\mathscr{T} = \{T_1, T_2, ..., T_N\}$ is given. Then, for a given partial trajectory $T^p = \langle L_1, L_2, ..., L_m \rangle$, the goal is to predict the future road segments L_i, where $i, m \in \mathbb{Z}$ and $i \geq m+1$.*

7.3.2 Distance Measure (trajDTW)

Most of the existing distance measures for trajectory similarity are not suitable for a large number of overlapping trajectories in a dense road network due to the use of either the number of overlapping road segments or maximum/minimum distance between trajectories in their computation. In our work, we use the Dijkstra based *dynamic time warping* (DTW) distance measure, trajDTW [260] to compute trajectory similarities which is suitable for a large number of overlapping trajectories in a dense road network. The superiority of the trajDTW over the traditionally used *dissimilarity with length* (DSL) and Hausdorff distance measures is demonstrated in [260]. The trajDTW is a normal DTW algorithm with a Dijkstra distance matrix based cost function and a window parameter w, which is set to the half of the length of shorter of two trajectories, to avoid overestimation of the actual distance. As the road network is static, the distance matrix D_{all} (of

size ($|E|\times|E|$) of all the edges E in G_{RN} can be pre-computed and stored.

7.3.3 Non-directional trajDTW

The directionality of trajectories can result in misleading distances among them, which in turn may cause incorrect clustering results. For example, suppose there are two trajectories T_1 and T_2, which follow the same route but in opposite directions, then the distance between them considering their directions in computation will be higher than the distance computed without considering their directions. Therefore, if their movement direction is considered as part of the distance computation, T_1 and T_2 may not be grouped in the same cluster. The problem of incorrect distance measure due to the movement direction of trajectories is addressed by reversing one of them (reversing the sequence order so that the starting point becomes the ending point and vice versa), and taking the minimum distance between the first and second trajectory, and the first trajectory and second reverse trajectory. This distance is called non-directional trajDTW [260], and is given as:

$$\text{non-directional trajDTW}(T_1,T_2) = \min(trajDTW(T_1,T_2), trajDTW(T_1, Reverse(T_2)) \quad (7.2)$$

7.3.4 Markov Chain Model

A *Markov chain* (MC) is the simplest form of the Markov process in which only the current state determines the probability of transitioning to the next state. Specifically, a Markov chain model is defined by the transition matrix M, which contains the transition probabilities associated with various state changes. In a road network, an MC is constructed by assigning a state to each node or road segments in the given road network. For any two adjacent road segments R_i and R_j in road network G_{RN}, the transition probability of traveling from R_i to R_j in one step is given by

$$p_{ij} = \frac{\#(R_i,R_j)}{\#(R_i)}, \quad (7.3)$$

where $\#(R_i,R_j)$ is the number of trajectories that contain the sequence $\{R_i,R_j\}$ and $\#(R_i)$ is the total number of trajectories that passes through R_i. For each pair of adjacent road segments in the graph network, the transition probabilities can be computed using (7.3), and stored as entries M_{ij} of transition probability matrix M (of size $|E|\times|E|$). We also define a transition count matrix

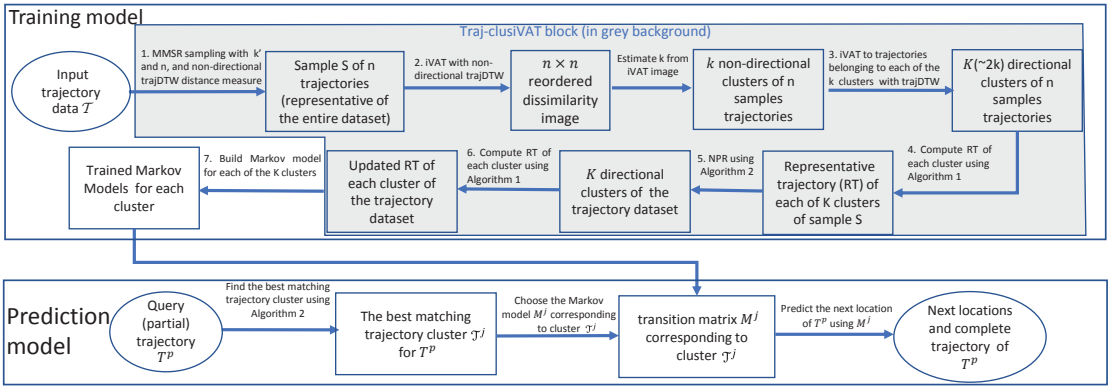


Figure 7.1: The architecture of our proposed framework.

W whose ij-th entry W_{ij} represents the number of trajectories that contain sequence $\{R_i, R_j\}$ i.e., $W_{ij} = \#(R_i, R_j)$. We utilize W in computing a representative trajectory for each cluster in our work.

7.4 Proposed Framework

This section presents our proposed framework for trajectory prediction. The frequent route patterns of moving objects can be discovered by clustering their historical trajectories. In our framework, we employ a modified version of the clusiVAT algorithm (Algorithm 5) that we called Traj-clusiVAT. In Traj-clusiVAT, we introduce a *representative trajectory* for each cluster to improve the performance of *nearest prototyping rule* (NPR) for trajectory clustering. We also modify the NPR technique in Traj-clusiVAT to improve its performance for trajectory prediction. The Traj-clusiVAT algorithm partitions the trajectories into different groups of similar trajectories, based on the trajDTW distance measure. After clustering trajectories, we train a first-order Markov chain model for each cluster using only the trajectories contained therein. Then, these trained Markov chain models are used for trajectory prediction. The architecture of our proposed framework consisting of both training and prediction models is illustrated in Fig. 7.1. Below, we explain the training and prediction model of our proposed TP framework in detail.

7.4.1 Training Model

The essential steps of our training model are: (i) MMRS sampling on input trajectory data, (ii) VAT/iVAT and clustering the trajectory sample using non-directional trajDTW to obtain k non-

directional clusters (iii) VAT/iVAT and clustering the trajectories of each of the k clusters using trajDTW resulting in K (approx. $2k$) directional clusters (iv) Compute *representative trajectory* (RT) of each cluster, (v) Assign remaining non-sampled trajectories to K clusters using NPR (vi) Re-compute the RT of each cluster, and (vii) Train a first-order Markov chain model for each cluster. The first six steps constitute the Traj-clusiVAT clustering algorithm. Below, we explain each step corresponding to the steps as shown in Fig. 7.1.

7.4.1.1 MMRS sampling on input trajectory data

The first step consists of extracting a small, representative sample from the large trajectory data using MMSR sampling with non-directional trajDTW distance measure on input trajectory data $\mathscr{T}$. The aim of this step is to find the most distinguished vehicle routes in a given road network. The use of non-directional trajDTW circumvents the selection of more than one trajectory from the same route. In this way, the Maximin (first) step of MMSR ensures that MMSR samples contain the k' MM trajectories of the most distinguished vehicle routes. This divides the trajectory data $\mathscr{T}$ into k' partitions. Then, additional trajectories are randomly chosen from each of the k' partitions to generate a sample S of n trajectories. The MMSR intelligently chooses n trajectories which are almost equally distributed among the different clusters as the N trajectories in the big trajectory data, i.e., it obtains a representative sample.

7.4.1.2 Clustering trajectory sample using non-directional trajDTW

The previous step provides a trajectory sample S containing n trajectories. In this step, VAT followed by iVAT is applied to the distance matrix D_n returning a reordered distance matrix D'^*_n, and the cut magnitudes of the MST links, h. The visualization of D'^*_n using $I(D'^*_n)$ suggests the number of clusters k present in the dataset. The k partitions can be obtained by cutting the $k-1$ longest edges in the iVAT-built MST of D_n.

If the dataset is complex and clusters are intermixed, cutting the $k-1$ longest edges may not always be a good strategy as the outliers, which are typically furthest from normal clusters, might comprise most of the $k-1$ longest edges of the MST, resulting in misleading partitions. A useful approach in such a scenario is to manually select the dark blocks, and find the sample

trajectories representing each dark block. Another useful approach [252] to obtain clusters is by cutting the MST using cut threshold magnitudes ordered by edge distances h in the MST. The cluster boundaries are defined by those indices z, which satisfy

$$h_z > \alpha \times mean(h), \tag{7.4}$$

where α is a parameter that controls how far two groups of data points should be from each other to be considered as separate clusters. Smaller values of α represent tighter cluster boundaries, while large values of α create loose cluster boundaries. The procedure to find an optimal value of α is described in [252].

The non-directional trajDTW distance measure is used in this step to cluster the n trajectories in order to avoid incorrect clustering due to the movement direction of trajectories, as mentioned in Section 7.3.3. From here on, in this chapter, we denote k as the number of non-directional clusters.

7.4.1.3 Clustering trajectories in each cluster using trajDTW (considers directions)

The previous step clusters the trajectories based on their path similarity computed using non-directional trajDTW, which ensures that the trajectories that are in opposite directions, but follow similar routes, are clustered together. Since Markov chain models are used in our framework to model the trajectories of each cluster, their transition probabilities may be misleading for trajectory prediction task for clusters in which the number of trajectories in opposite directions is approximately equal. To circumvent this problem, we use the trajDTW (directional) distance measure for the sample trajectories of each cluster obtained in the previous step to separate the trajectories going in opposite directions using a second application of the iVAT algorithm, which in turn, gives $K \sim 2k$ directional clusters.

7.4.1.4 Computing the RT of each cluster

In the NPR (next) step of clusiVAT, the non-sampled trajectories are assigned to one of the clusters (found in the previous step) based on their (nearest) distances from each cluster. For a fast implementation of NPR, we require a *representative trajectory* (RT) for each cluster that best

describes the cluster, much like centroid-based clustering methods identify a representative "center" for each cluster. However, it is not possible to compute the centroid of trajectory clusters in a conventional way due to different lengths of trajectories in each cluster. Existing methods of calculating RT [261–264] in the literature either compute the mean trajectory using the average of GPS coordinates [262, 264]; or select a trajectory from each cluster which minimizes the dissimilarity between all the trajectories within the cluster [261, 263]; or pick a random trajectory [263] from each cluster, and designates it as the RT. These methods incur a large computational cost to compute an RT that minimizes the dissimilarity among all the trajectories. Additionally, RTs computed using these methods do not show all the possible variability inside a cluster [265]. The mean trajectory computed from trajectories of different lengths may be inaccurate; thus, it may not be a good representative of the cluster.

Our scheme generates an *imaginary trajectory* (IT) (it may not belong to any of the trajectories in the cluster) as an RT for each cluster that describes the major movement patterns of the trajectories belonging to that cluster. The pseudocode of our proposed method to compute RT for each cluster is shown in Algorithm 16. Below, we explain our RT computing algorithm.

First, we compute the transition count matrix W^i for each cluster $\mathscr{T}^i$ using the trajectories in that cluster (line 2). Then, for each cluster $\mathscr{T}^i$, we compute the set of *frequent road segments* (FRSs) $\mathscr{R}^i_{FRS}$ using the $MinT$ threshold (line 3). The road segments in cluster $\mathscr{T}^i$ which contains at least $MinT\%$ of the total trajectories in that cluster are assigned to $\mathscr{R}^i_{FRS}$. Then, a set of *frequent source segments* (FSSs) $\mathscr{R}^i_{FSS}$ is identified (line 4). A source segment R^i_{SS} is a FSS, R^i_{FSS}, if at least $MinT\%$ of total trajectories in the cluster originate from R^i_{SS} i.e, $R^i_{FSS} \in \mathscr{R}^i_{SS},\ R^i_{FSS} \in \mathscr{R}^i_{FRS}$. Then for each FSS, $R^i_{FSS} \in \mathscr{R}^i_{FSS}$ (line 5), an imaginary trajectory $IT^i(R^i_{FSS})$ is initialized with R^i_{FSS} assigning it as current segment, $R_{current}$ (lines $6-7$). In lines $9-17$, we compute the next RS, R_{next} based on the highest transition count from current RS, $R_{current}$ using transition count matrix, W^i (refer to Section 7.3.4) . If $R_{next} \in \mathscr{R}^i_{FRS}$, then R_{next} is added to current $IT^i(R^i_{FSS})$, and assigned as $R_{current}$ to compute new R_{next}. The steps in lines $9-18$ are repeated until R_{next} is non-FRS, *which means an imaginary trajectory is an ordered sequence of only frequent road segments in that cluster*. A total of $|\mathscr{R}^i_{FSS}|$ imaginary trajectories will be generated for each cluster $\mathscr{T}^i$, corresponding to each $R^i_{FSS} \in \mathscr{R}^i_{FSS}$. We define a variable *Count_score* (line 8) for each imaginary trajectory $IT^i(R^i_{FSS}),\ R^i_{FSS} \in \mathscr{R}^i_{FSS}$, which is the sum of the total transition counts of each RS

$\in IT^i(R^i_{FSS})$ in cluster $\mathscr{T}^i$. Among all $|\mathscr{R}^i_{FSS}|$ ITs, the one which has the highest *Count_score* will be assigned as $RT(\mathscr{T}^i)$ of cluster $\mathscr{T}^i$ (line 20). As the $RT(\mathscr{T}^i)$ is the sequence of FRS with highest *Count_score*, it contains major movement behaviour or patterns of the trajectories belonging to the cluster $\mathscr{T}^i$.

Algorithm 16 does not require the computation of dissimilarity among all trajectories in that cluster to compute RT, which is computationally expensive for large size clusters. In contrast, Algorithm 16 is a novel algorithm to compute RT based on the transition count matrix of each cluster.

Algorithm 16 Computing the RT of each cluster

Input: $\mathscr{T}^j$- set of trajectories in cluster j, N_j- number of trajectories in cluster j, $\mathscr{R}^j$- set of road segments in cluster j, $\mathscr{C}(\mathscr{T}) = \{\mathscr{T}^1, ..., \mathscr{T}^K\}$- set of cluster of trajectories, $MinT$- FRS threshold

1: **for** each cluster $\mathscr{T}^i \in \mathscr{C}(\mathscr{T})$ **do**
2: Compute transition count matrix W^i for cluster $\mathscr{T}^i$
3: Compute FRSs, $\mathscr{R}^i_{FRS}$, from $\mathscr{R}^i$, $\mathscr{R}^i_{FRS} = \{R_j \in \mathscr{R}^i\}_{\#(R_j) \geq MinT \times N_i}$
4: Compute FSSs, $\mathscr{R}^i_{FSS}$ from $\mathscr{R}^i_{SS} = \{R_j\}_{R_j \in \mathscr{R}^i_{SS} \in \mathscr{R}^i_{FRS}}$
5: **for** each FSS $R^i_{FSS} \in \mathscr{R}^i_{FSS}$ **do**
6: Assign R^i_{FSS} as current road segment, $R_{current} = R^i_{FSS}$
7: Initialize an imaginary trajectory IT with $R_{current}$, $IT^i(R^i_{FSS}) = \{R_{current}\}$
8: $Count\_score(IT^i(R^i_{FSS})) = 0$
9: **while** each RS of $IT^i(R^i_{FSS}) \in \mathscr{R}^i_{FRS}$ **do**
10: Compute next RS, $R_{next} = \underset{R_j \in \mathscr{R}^i}{\arg\max} \{W^i_{current,j}\}$
11: **if** $R_{next} \in \mathscr{R}^i_{FRS}$ **then**
12: Append R_{next} to existing $IT^i(R^i_{FSS})$
13: $R_{current} = R_{next}$
14: $Count\_score(IT^i(R^i_{FSS}))$ += $W^i_{current,next}$
15: **else**
16: **break;**
17: **end if**
18: **end while**
19: **end for**
20: Select $IT^i(R^i_{FSS})$ with the highest $Count\_score(IT^i(R^i_{FSS}))$ from all $|\mathscr{R}^i_{FSS}|$ ITs of $\mathscr{T}^i$, and assign it as RT for cluster $\mathscr{T}^i$
21: **end for**

Output: $RT(\mathscr{T}^i)$- RT for each cluster $\mathscr{T}^i \in \mathscr{C}(\mathscr{T})$

7.4.1.5 Assigning non-sampled trajectories to identified K clusters using NPR

The previous step gives representative trajectory $RT(\mathscr{T}^i)$ for each cluster $\mathscr{T}^i$. In this step, $N-n$ non-sampled trajectories are assigned to one of the K directional clusters based on the

NPR. The NPR method in clusiVAT uses the trajDTW (directional) distance measure to assign non-sampled trajectories to one of the K clusters based on their nearest distance from (clustered) sample trajectories. However, trajDTW distance of a non-sampled trajectory to cluster RTs may not be an appropriate measure for the NPR step due to its dependency on the *length* of trajectories, as explained by the following example.

Suppose T_a is a non-sampled trajectory in $\mathscr{T}$, and $RT(\mathscr{T}^i)$ and $RT(\mathscr{T}^j)$ are the RT of cluster $\mathscr{T}^i$ and $\mathscr{T}^j$, respectively. Let T_a be a sub-trajectory of $RT(\mathscr{T}^i)$ i.e., T^a is fully contained in $RT(\mathscr{T}^i)$. Since the trajDTW distance relies on a warping window size parameter w, the $trajDTW(T_a, RT(\mathscr{T}^i))$ not only depends on the coordinates of RSs of both trajectories, but it also depends on the length of T_a and $RT(\mathscr{T}^i)$. Moreover, $trajDTW(T_a, RT(\mathscr{T}^i))$ also varies depending on the position of T_a in $RT(\mathscr{T}^i)$ due to window parameter. Therefore, even if $T_a \sqsubseteq RT(\mathscr{T}^i)$ and $T_a \not\sqsubseteq RT(\mathscr{T}^j)$, T_a may be incorrectly assigned to cluster $\mathscr{T}^j$ instead of $\mathscr{T}^i$ if $trajDTW(T_a, RT(\mathscr{T}^i)) \geq trajDTW(T_a, RT(\mathscr{T}^j))$. Here is such an example from T-Drive data. Suppose $T_1 = \langle 70, 75, 90, 89, 88\rangle$ is a non-sample trajectory, and $RT(\mathscr{T}^1) = \langle 16, 18, 68, 70, 75, 90, 89, 88\rangle$ and $RT(\mathscr{T}^2) = \langle 68, 70, 75, 91, 92\rangle$ are RTs of two clusters, where each trajectory is represented by a sequence of road segments' IDs of Beijing road network (refer to Section 7.6.1). The trajDTW distances are: $trajDTW(T_1, RT(\mathscr{T}^1)) = 0.3482$ and $trajDTW(T_1, RT(\mathscr{T}^2)) = 0.2767$. Therefore, although T_1 is a sub-trajectory of $RT(\mathscr{T}^1)$, it will be assigned to cluster $\mathscr{T}^2$ based on nearest trajDTW distance. Such assignments of non-sampled trajectories to (incorrect) cluster may include outlier trajectories or road segments in that cluster, which may adversely affect Markov chain modeling, and consequently, degrade the performance of trajectory prediction.

To address above issue, we propose a *hybrid* NPR strategy based on the path probability and trajDTW distance measure. *Hybrid* NPR is similar to clusiVAT NPR except for those non-sampled trajectories, which are sub-trajectory of any of the clusters' trajectories. The pseudocode of our hybrid NPR method is shown in Algorithm 17. For a query trajectory $T^q = \{R_1, R_2, ..., R_l\}$, we first compute the path probability $P^i(T^q)$ for each cluster $\mathscr{T}^i$, which is defined as

$$P^i(T^q) = \prod_{j=1}^{l} p_{j(j+1)} \Leftrightarrow \prod_{j=1}^{l} M^i_{j(j+1)}. \tag{7.5}$$

$P^i(T^q) > 0$ means that sequence T^q appears at least once in cluster $\mathscr{T}^i$, whereas $P^i(T^q) = 0$

means that sequence T^q is not present in cluster $\mathscr{T}^i$. If the sequence T^q is present in any cluster $\mathscr{T}^i$ i.e., any$(P^i(T^q)) > 0$, then T^q is assigned to the cluster with the highest path probability. If the sequence T^q is not present in all clusters $\mathscr{T}^i$ i.e., all$(P^i(T^q)) = 0$, then T^q is assigned to the cluster based on its (minimum) trajDTW distance from RTs. All non-sampled trajectories in $\mathscr{T}$ are assigned to one of the K clusters using Algorithm 17.

Algorithm 17 Hybrid NPR Method

Input: T_q - query trajectory, M^j - transition probability matrix for cluster j, $RT(\mathscr{T}^j)$- representative trajectory for cluster j

1: Compute the path probability $P^i(T_q)$ of query trajectory in each cluster $\mathscr{T}^i$ using M^i and Eq. 7.5.
2: **if** any$(P^i(T_q) > 0$ **then** ▷ if T_q is present in any cluster $\mathscr{T}^i$
3: Select the cluster c with the highest $P^i(T_q)$ i.e., $c = \underset{\mathscr{T}^i \in \mathscr{C}(\mathscr{T})}{\arg\max} \{P^i(T_q)\}$
4: **else**
5: Compute the trajDTW distance of T_q from $RT(\mathscr{T}^i)$, $y^i = trajDTW(T_q, RT(\mathscr{T}^i))$, for each cluster $\mathscr{T}^i \in \mathscr{C}(\mathscr{T})$
6: Select the cluster c with the minimum y^i i.e., $c = \underset{\mathscr{T}^i \in \mathscr{C}(\mathscr{T})}{\arg\min} \{y^i\}$
7: **end if**
8: Assign the T_q with cluster c (or $\mathscr{T}^c$).

Output: cluster label for T_q

7.4.1.6 Recompute the RT of each cluster after NPR

The assignment of all non-sampled trajectories to one of the K clusters in the NPR step updates each cluster with new trajectories. Therefore, the representative trajectory is recomputed for each updated cluster using Algorithm 16.

7.4.1.7 Train Markov chain model

For each of the K clusters, we build a first-order Markov chain model using the trajectories of that cluster. Specifically, we compute the transition probability matrix M^i for each cluster $\mathscr{T}^c$.

For a basic understanding of Traj-clusiVAT algorithm, we graphically explain its steps on a small trajectory data $\mathscr{T}$, as shown in Fig 7.2. An input trajectory data $\mathscr{T}$ containing $N = 9$ trajectories is shown in Fig 7.2 (a). The MMSR sampling on $\mathscr{T}$ with non-directional trajDTW in the first step returns a MMSR sample S containing $n = 6$ sample trajectories $\{1,4,5,6,7,9\}$, which are well-distributed in sample S, as shown in Fig 7.2 (b). In the next step, iVAT is applied

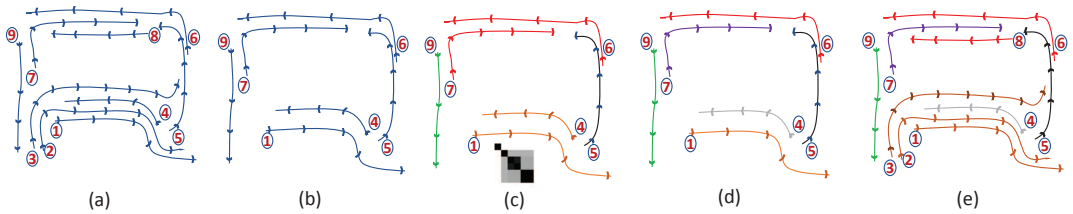


Figure 7.2: A simple illustration of Traj-clusiVAT for trajectory clustering

to S using the non-directional trajDTW distance measure, which clusters the trajectories based on the path similarity irrespective of their movement directions. The iVAT image in Fig 7.2 (c) shows four dark blocks along its diagonal, which indicates four clusters in sample S. Having an estimate of $k = 4$, sample S is partitioned into four (non-directional) clusters $\{\{1,4\},\{5\},\{6,7\},\{9\}\}$, as shown with four different colors in Fig 7.2 (c). Then, the trajectories in each cluster going in opposite directions are separated using the iVAT with the trajDTW distance measure, which gives $K = 6$ directional clusters $\{\{1\},\{4\},\{5\},\{6\},\{7\},\{9\}\}$, each cluster is shown with a different colour in Fig 7.2 (d). Since there is only one trajectory in each cluster in this case, they are the RTs for corresponding clusters. In the next step, non-sampled trajectories $\{2,3,8\}$ are assigned to one of the 6 clusters using NPR (Algorithm 17), which partitions the complete data into 6 clusters $\{\{1,2,3\},\{4\},\{5\},\{6,8\},\{7\},\{9\}\}$. Trajectory 4 is in different cluster than $\{1,2,3\}$ due to opposite direction. Then, a Markov chain model is trained for each cluster using the trajectories of that cluster.

7.4.2 Prediction Model

For a given partial trajectory $T^p = \langle L_1, L_2, ..., L_m \rangle$, we first estimate the best matching representative cluster $\mathscr{T}^c$ using our hybrid NPR approach, and then choose the corresponding Markov model of the cluster to predict the next locations L_i, $i \geq m+1$. Using the cluster $\mathscr{T}^c$, the location L_{m+1} that the object will arrive at next is given by

$$L_{m+1} = \underset{L_j \in \mathscr{R}^c}{\arg\max} \{p_{mj}\} \Leftrightarrow \underset{L_j \in \mathscr{R}^c}{\arg\max} \{M^c_{mj}\} \tag{7.6}$$

The T^p is updated with the next predicted location L_{m+1}. Then, the updated T^p is used to estimate the best matching cluster and the corresponding MM is used to predict the next location. The

complete trajectory is predicted by computing next locations in a sequential manner using these steps.

7.5 Time Complexity

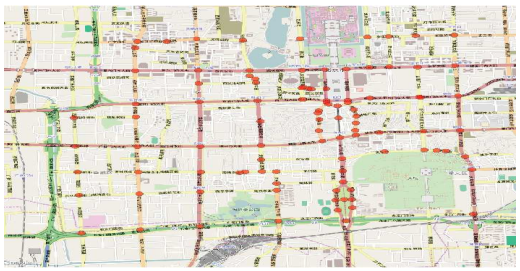
The first step in Traj-clusiVAT is the selection of k' distinguished trajectories which are at maximum distance from each other. This step has the time complexity of $O(k'N)$, where k' is a user-defined parameter for an overestimate of the number of clusters in the input trajectory data and is usually chosen to be (inessentially) large (usually 50 to 200). The next step is to randomly select n trajectories from k' NPR groups to get a sample S. The computation of distance matrix D_n and VAT on a sample S has a time complexity of $O(n^2)$. Usually $n << N$, so the computation of D_n and VAT on S is pretty fast and takes just a small fraction of the total run time of Traj-clusiVAT. The trajDTW distance measure uses Dijkstra's shortest path distance in the standard DTW algorithm. Its best, average case complexity with binary heaps is $O(|E|+|V|\log|V|)$ [266]. For two trajectories of length l_1 and l_2, standard DTW has time complexity of $O(l_1 l_2)$. Remark- There are approximate algorithms such as FastDTW [267] which have a linear time complexity in the average length of trajectories, however, we have not used this implementation in our experiments. The NPR step in Traj-clusiVAT has complexity of $O(n(N-n))$. The computation of RTs has linear time complexity in K. The construction of a Markov model for each cluster is a simple and fast process, which has $O(K)$ time complexity and $O(|E|^2)$ space complexity.

7.6 Experiments

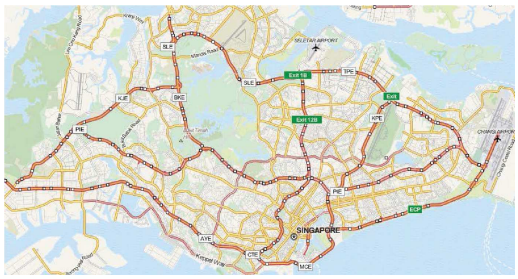
In this section, we conduct an extensive experimental study on two real-life, vehicle trajectory datasets to evaluate the performance of our proposed framework. We first describe the datasets, their preprocessing, evaluation metrics and computational protocols adopted in our empirical study, and then present the experimental results.

7.6.1 Datasets

We performed our experiments on two real trajectory datasets.



(a) T-Drive: Road network in the center of Beijing



(b) Singapore road network

Figure 7.3: Road networks used in our trajectory prediction experiments

7.6.1.1 T-Drive taxi trajectory data [89, 90]

This trajectory dataset is obtained from the T-Drive project which contains one-week trajectories of 10,357 taxis during the period of Feb. 2 to Feb 8, 2008 within Beijing, China. The total number of points is about 15 million, and the total distance of the trajectories is 9 million kilometers. In our experiment, we have taken a subset of this dataset, which contains trajectories from a road network in the center of Beijing city, as shown in Fig. 7.3(a). This road network consists of 100 nodes and 141 road segments (edges). The average sampling interval is 177 seconds with an average distance of about 623 meters, which is quite large for a city traffic environment as the length of many road segments is smaller than the average sampling distance.

7.6.1.2 Singapore taxi trajectory data

This dataset consists of the trajectories of more than 15,000 taxis collected over a duration of 1 month from a road network in Singapore City, as shown in Fig. 7.3(b). This dataset is very dense as it consists of more than 370 million GPS logs. The general format of each data point is as follows: {Time Stamp, Taxi Registration, Latitude, Longitude, Speed, Status}. The Status field contains information about occupation state of Taxi, such as *FREE* and *POB* (Passenger on Board). In order to extract each individual taxi's trip from the raw data, we detect the following sequence: starting from *FREE* to *POB* and ending from *POB* to *FREE*, using the trip extraction framework presented in [268]. This road network consists of 1641 nodes and 2941 edges, with an average edge length of 350m.

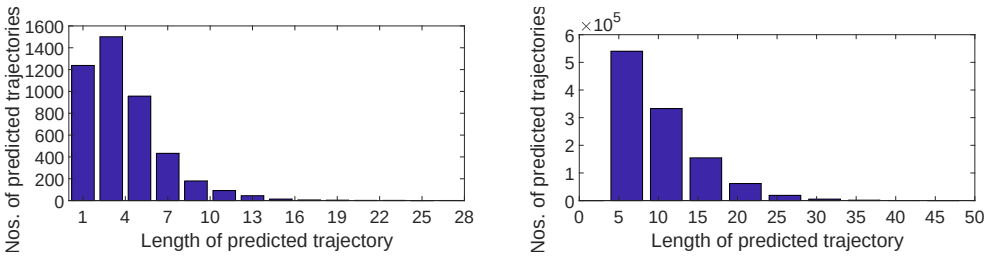
Table 7.2: Training and test set description

	T-Drive Taxi	Singapore Taxi
Training Set	35,501	1,955,573
Test Set	7,904	1,303,717
Total trajectories	43,405	3,259,290

Data Pre-processing

To obtain the trajectories as a sequence of road segments, each of which has a common node with its former and latter road segment, we first map each GPS point to its nearest road segment (commonly known as the Map Matching Problem). We remove duplicate road segments in a trajectory. After removing duplicate nodes, if two consecutive road segments do not have a common node, Dijkstra's algorithm is used to find and insert the minimum length road segment sequence between the two non-adjacent road segments. We use the popular open source map matching tool GraphHopper [269], which provides an implementation of the approach presented in [192].

After pre-processing, we have $N = 43,405$ trajectories in the T-Drive data whose lengths lie in the range of 5 to 200 road segments and have an average of 14 road segments, and $N = 3,259,290$ (3.26 million) trajectories in the Singapore data whose lengths lie in the range of 10 to 250 road segments and have an average of 22 road segments. To prepare training and test sets for both datasets, we first divided the trajectories into two sets based on the day of week viz., weekdays and weekends, during which the trip is being made. For the one-week T-Drive data, we considered trajectories during first 4 weekdays (Monday to Thursday) and first weekend day (Saturday) as the training set, and trajectories during the remaining days (Friday and Sunday) of that week as the test set. For the one-month Singapore data, we considered 60% trajectories randomly as training set and remaining 40% as the test set, for both weekdays and weekend data. The size of training and test sets for both trajectory datasets is shown in Table 7.2. We split each trajectory in a test set into two halves. The first half is used as a partial trajectory (or query trajectory) for predicting its future locations and the second half is used as ground truth to validate our predictions. The distribution of predicted trajectories (second half) in the T-Drive and Singapore test sets is shown in Fig. 7.4.



(a) T-Drive Taxi (b) Singapore Taxi

Figure 7.4: Trajectory distribution of predicted trajectories based on their lengths.

7.6.2 Evaluation Metrics

In our experiments, we assess the performance of our framework for next location prediction (also known as one-step prediction) and long-route prediction using following evaluation metrics:

7.6.2.1 Prediction Accuracy (PA)

The PA is the ratio of correctly predicted locations to the total possible number of predicted locations for each trajectory. Given a predicted trajectory sequence $T_{pred} = \{L_1, L_2, ..., L_m\}$ and a true (actual) trajectory sequence $T_{true} = \{R_1, R_2, ..., R_m\}$, the prediction accuracy is defined as

$$PA = \frac{1}{|T_{pred}|} \sum_{j=1}^{m} H(L_j, R_j), \tag{7.7}$$

where $H(L_j, R_j)$ is 1 if $L_j = R_j$, else 0. The average prediction accuracy is the average of PA for all predicted trajectories in test set $\mathscr{T}^{test}$.

7.6.2.2 Prediction Rate (PR)

The PR is the number of trajectories that are correctly predicted over the total number of trajectories in test set. It is defined as

$$PR = \frac{1}{|\mathscr{T}^{test}|} \sum_{j=1}^{|\mathscr{T}^{tr}|} H(T_{pred_j}, T_{true_j}), \tag{7.8}$$

where $H(T_{pred_j}, T_{true_j})$ is 1 if $T_{pred_j} = T_{true_j}$, else it is 0.

7.6.2.3 Distance error (DE)

Another important performance metric of the long-term prediction system is the capability of continuous route prediction. The *distance error* is defined as the average spatial (*Haversine*) distance between predicted and actual routes. Given a route sequence T_{pred} and T_{true}, the distance error between them is given as

$$DE(T_{pred}, T_{true}) = \frac{1}{|T_{pred}|} \sum_{j=1}^{m} D_H(L_j, R_j), \tag{7.9}$$

where $D_H(L_j, R_j)$ is the Haversine [270] distance between two locations (road segments).

7.6.2.4 One-step accuracy (OA)

This is the ratio of correctly predicted next locations to the total predicted next locations for all trajectories in test set.

7.6.2.5 One-step distance error (ODE)

The ODE defined as the average distance error for one-step (or next location) prediction.

7.6.3 Comparison Methods

Among the plethora of MM and clustering based TP methods available in the literature, we implemented these two approaches for comparison.

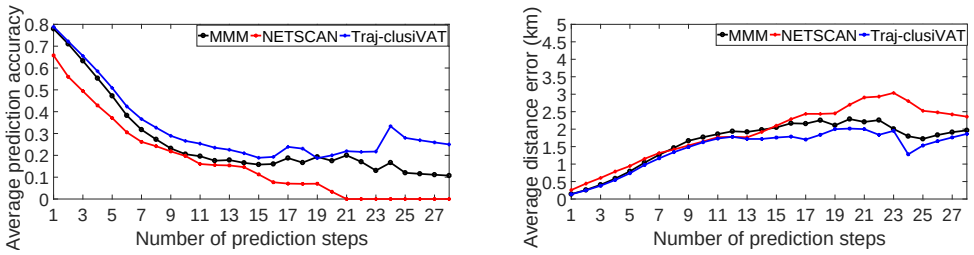
1. Mixed Markov model (MMM) based TP [87]: MMM was proposed as an intermediate model between standard MM and HMM which can encompass all types of movement behaviour present in an input trajectory data. It first clusters the trajectories into groups using the EM algorithm, and then builds an MM for each group, which is subsequently used for prediction. This approach was tested on synthetic and real datasets in [87], which showed 74.1% accuracy for MMM, in comparison to $16.9 - 45.6\%$ for MM and $2.4 - 4.2\%$ for HMM.

2. NETSCAN-based TP: The well-known density-based algorithm DBSCAN and its variants [205, 271–273] have been used extensively as a trajectory clustering method for location prediction [195]. However, they are not suitable for a large number of trajectories as computation of the distance matrix is time intensive. Kharrat *et al.* [88] proposed a trajectory clustering relative of DBSCAN, called NETSCAN which first finds dense road segments based on the moving object counts, merges them to form dense paths on the road network, and then assigns sub-trajectories to the dense paths based on a measure of similarity. This method requires two user-defined parameters: a density threshold - the minimal required density for transition, and a similarity threshold- the maximum density difference between neighbouring road segments. We implement NETSCAN to cluster trajectories into dense road segments, then built an MM for each cluster, and subsequently used them for TP.

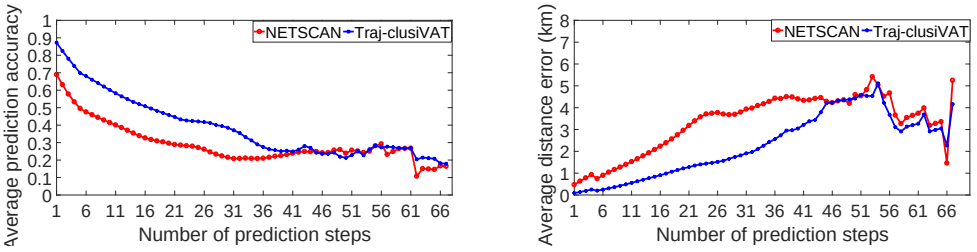
Our proposed method and the baseline methods discussed above are also comparable in terms of prediction time (which will be discussed shortly). They all require a short prediction time and satisfy the requirement of real-time prediction.

7.6.4 Computation Protocols

All algorithms were coded in MATLAB on a PC with the following configuration; OS: Windows 7 (64 bit); processor: Intel Core $i7-4770$ @3.40GHz; RAM: 16GB. We denote the comparison approaches of [87] as MMM, of [88] as NETSCAN, and our Traj-clusiVAT based TP approach as Traj-clusiVAT. All three algorithms were applied to T-Drive data. The MMM method requires the computation and storage of an intermediate matrix of size $|E|\times|E|\times N$, which is very large for Singapore data, so we can not apply MMM to the Singapore data. The number of mixed models of MMM was determined using 10-fold cross-validation. The NETSCAN parameter, density threshold and similarity threshold, were chosen to get as many dense paths (with at least six road segments) as the number of clusters we get using the Traj-clusiVAT algorithm, for a fair comparison. The parameters for Traj-clusiVAT were chosen as follows: $k' = 150$, $n = 500$, and $\alpha = 0.05$ for the T-drive data, and $k' = 300$, $n = 1000$, and $\alpha = 0.06$ for the Singapore data, and $MinT = 30\%$ for both data. It is worth noting that, unlike other clustering algorithms, the clusiVAT algorithm is relatively insensitive to the choice of k' and N [63]. Moreover, we study the effect of α on Traj-clusiVAT performance in our experiments.



(a) T-Drive Dataset



(b) Singapore Taxi Dataset

Figure 7.5: Average prediction accuracy and distance error comparison by prediction steps

7.6.5 Comparison of MMM, NETSCAN, and Traj-clusiVAT for Long-term Predictions

Long-term prediction, also known as continuous route prediction, is a challenging and ongoing research problem in TP. In this experiment, we compare the performance of the MMM, NETSCAN, and Traj-clusiVAT-based prediction approaches for m-step predictions. Specifically, this refers to predicting the next m locations for a given partial trajectory. Fig. 7.5 shows the average prediction accuracy (left panels) and average distance error (right panels) of all three algorithms for increasing prediction steps. The graphs in Fig. 7.5 support these observations:

(i) First, the Traj-clusiVAT outperforms the MMM and NETSCAN-based TP approaches based on the average PA and DE for the T-Drive data, as shown in Fig. 7.5(a). The higher the number of prediction steps, the larger the gap between Traj-clusiVAT and the other two approaches. This means that the Traj-clusiVAT performs better not only for short-term predictions but it performs even better than other two approaches for long-term predictions. This is probably because Maximin sampling in Traj-clusiVAT finds the trajectories which are furthest from each other. As the traj-DTW distance measure yields higher distances for longer trajectories, Maximin sampling tends to pick longer trajectories in its output sample which form separate clusters in subsequent steps. The Markov models trained on these clusters after the NPR step contain all movement be-

haviours similar to those longer trajectory patterns. Therefore, if a query trajectory pattern is not available in any cluster, which is frequent for longer query patterns, then it is assigned to a cluster based on its nearest distance from all cluster RTs. This will assign longer query trajectories to any of the clusters containing longer trajectory patterns, and subsequently, corresponding MMs trained on these clusters contribute towards better predictions for longer query trajectories during the prediction phase. On the other hand, the longer movement rules cannot be easily represented by Markov-based models, especially for irregular trajectory data, due to uncertainty in movement behaviours of vehicles in a complex road network. As there are only a few prediction trajectories available for the T-Drive test set whose lengths are greater than 16 as shown in Fig. 7.4 (a), the performance of all approaches cannot be considered conclusive for longer prediction steps ($m > 16$) based on their performance on the T-drive data.

(ii) Fig. 7.5 (b) shows that the Traj-clusiVAT model also performs better than the NETSCAN-based method based on the average PA and DE values for the Singapore data. The gap between the NETSCAN and Traj-clusiVAT plots increases until 31-th prediction step and then reduces with longer prediction steps. This may be because the trajectory clusters obtained by NETSCAN are usually spread over the entire road network [260], which results in longer dense paths. Therefore, its performance becomes competitive with Traj-clusiVAT for longer prediction lengths compared to its short-term prediction performance.

(iii) The performance of all three approaches deteriorates as the prediction step increases. This may be because the number of frequent trajectory patterns obtained is small for long-term predictions, which do not contain enough information to forecast future locations[1].

In our experiments, we find that most of the clusters contain frequent trajectory patterns whose lengths are less than six or seven. Only a few clusters contain frequent trajectory patterns whose lengths are longer than seven steps. This finding conforms with the real-world situation, where a driver usually predicts only next few locations.

The average long-term prediction performance of all three approaches is summarized in Table 7.3. The best performance is shown in bold for both datasets. Traj-clusiVAT achieves the highest PA, 0.62 and 0.59 and the lowest DE, 0.58km and 0.60km, for the T-Drive and Singapore taxi datasets, respectively. The MMM-based prediction approach is the second best method for

[1] And the other reason, as Niels Bohr said, is that "it is very hard to predict, especially the future"

Table 7.3: Long-term prediction: Comparison of MMM, NETSCAN and Traj-clusiVAT

	T-Drive Data		
	Average PA	Average DE (km)	PR (%)
MMM	0.55	0.68	39.9
NETSCAN	0.41	0.87	24.3
Traj-clusiVAT	**0.62**	**0.58**	**49.8**
	Singapore Data		
NETSCAN	0.34	1.41	5.1
Traj-clusiVAT	**0.59**	**0.60**	**24.8**

Table 7.4: Next location prediction: Comparison of MMM, NETSCAN and Traj-clusiVAT

	T-Drive		Singapore Taxi	
	OA	ODE (km)	OA	ODE (km)
MMM	0.77	0.24	-	-
NETSCAN	0.67	0.54	0.62	0.29
Traj-clusiVAT	**0.80**	**0.23**	**0.86**	**0.05**

T-Drive in terms of all three evaluation metrics. Traj-clusiVAT achieves prediction rates of 49.8% and 24.8% for the T-Drive and Singapore trajectory datasets, respectively. In other words, Traj-clusiVAT is able to predict complete trips for around 50% of the trajectories in T-Drive, and for around 25% of the trajectories in Singapore data. In contrast, MMM predicts about 40% of the total trajectories correctly for the T-Drive dataset. Although NETSCAN performance improved for longer predictions due to longer dense paths, it only predicted about 5% of the total trajectories correctly. Overall, Traj-clusiVAT based prediction approach outperforms both MMM and NETSCAN-based prediction approaches based on all three evaluation metrics.

7.6.6 Next location predictions

In this experiment, we compare Traj-clusiVAT to the other two comparison approaches for predicting next locations. Given a taxi's current location, the next location prediction is to forecast the next location where the taxi may go. Table 7.4 shows the one-step accuracy (OA) and one-step distance error (ODE) on the T-Drive and Singapore trajectory datasets. The Traj-clusiVAT-based approach predicts next location with more than 80% accuracy and with distance error of less than a quarter of km for both T-Drive and Singapore data. The long-term prediction performance (Table 7.3) of NETSCAN and Traj-clusiVAT is better for T-Drive than for the Singapore data.

Conversely, the next location prediction performance of both approaches is better for the Singapore data than the T-Drive data. This may be because Singapore data contains a large number of longer trajectories that span entire Singapore city, whereas T-drive contains partial trajectories belonging to small part of the entire road network, hence modeling is not that efficient for T-drive data. In summary, Traj-clusiVAT outperforms both MMM and NETSCAN for next location prediction.

7.6.7 Effect of latest locations of partial trajectory for prediction

In the prediction step of Traj-clusiVAT, a partial trajectory $T^p = \{R_1, R_2, ..., R_l\}$ is assigned to one of the K clusters using our hybrid NPR approach. For a T^p, the best cluster is chosen based on either its path probability $P^i(T^p)$ in each cluster or its trajDTW distance from each cluster (if T^p is not fully contained in any cluster). The length of known partial trajectory T^p increases after each next location prediction as T^p is updated with a predicted location after each prediction, and subsequently, the updated T^p is used for next location prediction, and so on.

We conduct an experiment in which instead of using full known partial trajectory T^p, we use only the latest movement steps or latest subsequence of T^p until prediction to choose the best matching cluster in the hybrid NPR step. In this regard, we choose a different number of latest locations of known partial trajectories until prediction and investigate the effect on the performance for trajectory prediction.

Fig 7.6 shows the average distance error for a different number of latest locations of known partial trajectories until prediction for the T-drive and Singapore data. It can be inferred from the figure that the best performance is achieved when only the latest two or three locations of partial trajectory are used to find the best matching cluster. The average distance error increases if more than three latest locations are used to find the best cluster in the hybrid NPR step. This is because as the length of T^p increases, its path probability in each cluster decreases, which means that the chance of sequence T^p being fully contained in any cluster decreases. Moreover, if T^p is not fully contained in any cluster representative trajectory, its distance from all the clusters increases with increasing length. This may result in wrong cluster assignment, which in turn, may degrade Traj-clusiVAT's prediction performance.

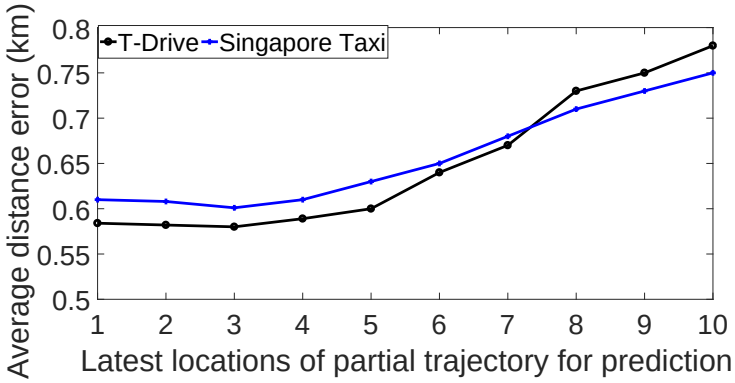
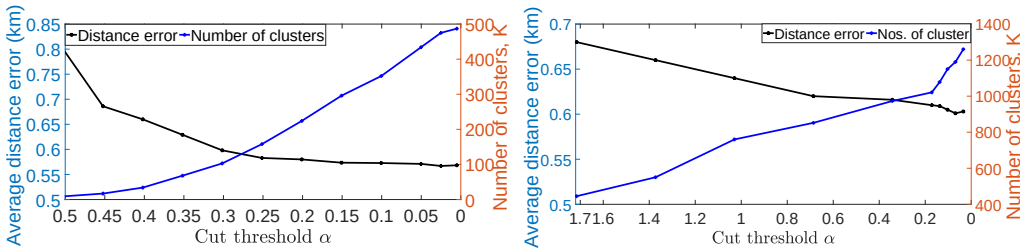


Figure 7.6: Average DE vs latest locations of partial trajectory used to select best cluster in the hybrid NPR step.

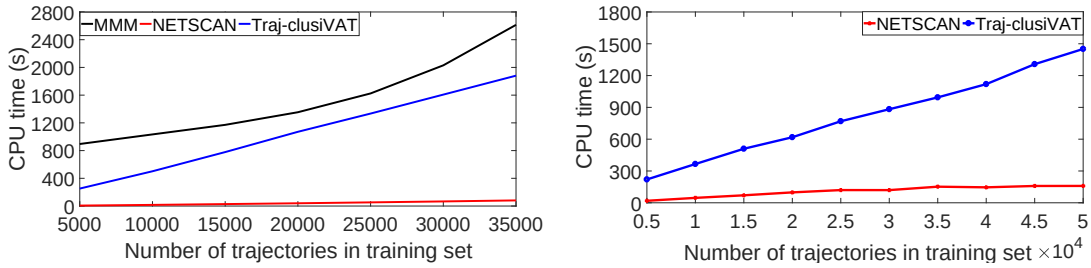


(a) T-Drive (b) Singapore trajecory data

Figure 7.7: Effect of cut threshold α

7.6.8 Effect of Cut threshold α

In this experiment, we study the effect of cut threshold α. The parameter α in Traj-clusiVAT controls how far two groups of data points should be from each other to be considered as different clusters. Figure 7.7 shows the average DE and the number of clusters K for different values of α for the T-Drive and Singapore data. The lower the cut threshold, the tighter the cluster boundaries, and hence, the higher the number of clusters. As the number of clusters K increases, the average DE reduces. This is primarily because the higher K corresponds to a larger number of unique frequent patterns, which improves the prediction performance. Figure 7.7 shows that the Traj-clusiVAT performance improves with lower cut threshold α or with the higher number of clusters. However, with large K, more MM needs to be trained, and hence, system complexity increases. Moreover, Traj-clusiVAT performance does not improve significantly below a certain value of α for either dataset. The procedure to find an optimal value of α is described in [252].



(a) T-Drive (b) Singapore

Figure 7.8: Training time comparison

Table 7.5: Prediction time in seconds for all three algorithms

	MMM	NETSCAN	Traj-clusiVAT
T-Drive Taxi	0.0014s	0.0011s	0.0012s
Singapore Taxi	-	0.063s	0.066s

7.6.9 Time performance analysis

The training time of all three algorithms on different-size training sets is shown in Fig. 7.8. The CPU-time for MMM increases most with the training data size because the computation of an intermediate matrix of size $|E|\times|E|\times N$ incurs high computational overhead and space complexity for large N. On the other hand, NETSCAN incurs the lowest computation time among all three methods. This is because it just computes dense paths based on the movement counts and density threshold, and assigns all trajectories to these dense paths based on similarity. Although it takes less time for training, it suffers from lower prediction accuracy. Traj-clusiVAT scales almost linearly in the number of trajectories, which make it scalable for big trajectory datasets.

Prediction-time is also an important criterion in real-time trajectory prediction. The average prediction time for all three approaches is presented in Table 7.5. We can see that all three approaches take similar times to forecast each trajectory for the T-drive dataset. The response time is less than 1.5ms for T-drive, which suggests that all three approaches satisfy the requirement for real-time prediction. The average prediction time is higher for the Singapore dataset due to a large number of clusters identified by both NETSCAN and Traj-clusiVAT algorithms, but at ~ 0.06 seconds, it is negligible in terms of real-time prediction utility.

7.7 Summary

This chapter presents a novel, scalable, hybrid architecture for short-term and long-term trajectory prediction, which can handle a large number of trajectories from a large-scale dense road network. The proposed framework is based on a scalable clustering approach, called Traj-clusiVAT, which is a modified version of clusiVAT for trajectory prediction. In particular, Traj-clusiVAT develops a novel algorithm to compute a representative trajectory for each cluster. We also presented a new, hybrid nearest prototyping approach for accurate trajectory assignment to (one of) the clusters identified in previous steps of Traj-clusiVAT. Finally, we also propose a hybrid prediction framework based on hybrid NPR which can assign a query trajectory to the best-matching cluster in a robust way to improve prediction performance.

We demonstrated the superiority of our proposed approach by comparing it with mixed Markov model-based and NETSCAN-based TP approaches on two real trajectory datasets, including a large-scale trajectory dataset containing 3.28 million trajectories of passenger trips obtained from $15,061$ taxis within Singapore over a period of one month. Our experimental results on both trajectory datasets show that Traj-clusiVAT based TP approach outperforms the other two approaches based on the prediction accuracy and distance error for short-term and long-term prediction for these two datasets. Our experimental results also suggest that Traj-clusiVAT satisfies the requirement for real-time predictions.

Chapter 8
Conclusions

8.1 Summary of Contributions

Everyday an abundant amount of data is generated from various sources such as IoT networks, smartphones, and social network activities. Making sense of such an unprecedented amount of data is essential for many businesses, services, and applications, and almost for every domain such as health care, transportation, finance, and energy sectors. Therefore, scalable and efficient algorithms are required to manage and extract useful information from a huge amount of data.

This thesis focused on mining information from a large volume of data that is possibly unlabeled, anomalous, streaming, and high-dimensional. Cluster analysis is the best unsupervised approach to extract actionable knowledge and timely detection of interesting events from unlabeled data. This thesis developed a suite of novel algorithms to solve each of the three problems of cluster analysis, namely, cluster tendency assessment, clustering, and cluster validity for large-scale, high-dimensional data, including a novel scalable framework for vehicle trajectory prediction. Chapters 3-7 presented the main contributions of this thesis for big data cluster analysis.

In Chapter 3, we introduced a simple and computationally efficient framework, CAFCM, for high-dimensional data clustering. The CAFCM framework employs FCM clustering on an ensemble of random projections to obtain multiple fuzzy membership matrices and then aggregates them based on their quality, which is determined using cluster validity indices (CVIs). The CAFCM algorithm eliminates the complexity involved in dealing with a big affinity matrix and a final time-consuming clustering step, such as the ones reported in three state-of-the-art approaches, using a cumulative agreement based aggregation approach. We demonstrated the superiority of the CAFCM approach by comparing it with three existing approaches on two Gaussian mixtures

and six real, high-dimensional datasets. Experimental results showed that CAFCM outperforms the other three approaches in terms of accuracy, stability, space, and time complexity. Moreover, CAFCM does not require any prior knowledge of the number of clusters that might be present in the dataset, which makes it attractive for real clustering problems.

Chapter 4 presented a novel hybrid framework, FensiVAT, for fast cluster tendency assessment and subsequent clustering on large volumes of high-dimensional data. FensiVAT integrates VAT with an intelligent sampling scheme, called *Maximin Random Sampling* (MMRS) and a new random projection (RP)-based ensemble method, in an efficient and effective manner. FensiVAT was compared with nine clustering approaches including six big data clustering methods, viz., clusiVAT, spkm, MBKM, CLARA, CURE, GARDENkm, FatSpec, and two high-dimensional data clustering methods, PROCLUS, and RP-EN. Experiments performed on several synthetic and real (labeled and unlabeled) datasets, which have the large sample size and high dimensions, demonstrated that FensiVAT provides a reliable estimate of the number of clusters (k) by the number of dark blocks along the reordered dissimilarity image, in a few seconds. FensiVAT is up to several order of magnitudes faster than the nine (except MBKM) big data clustering approaches, without compromising clustering accuracy.

Chapter 5 addressed the cluster validity problem for big data. Dunn's index is a popular cluster validity index, but its computation is infeasible for large values of N due to its quadratic complexity $O(N^2)$. In Chapter 5, we presented six novel algorithms including two incremental approaches for approximating Dunn's index for big data. First four methods viz., αMMRS, αn*MMRS*, *iMMRS*, and *in*MMRS, used variations of the MMRS sampling to identify the *approximate boundary points* in each cluster, which are used to compute Dunn's index (DI) for big data. Two additional methods, QMS+ and BEPS+, were presented that are based on the unsupervised training of one class support vector machines. We compared our four MMRS methods with two boundary point estimations methods, QMS+ and BEPS+, based on approximation accuracy and CPU time. Our experiments on several labeled datasets of varying sizes showed that computing approximations to DI with MMRS methods are both tractable and accurate. The incremental algorithm *in*MMRS offered an average speedup of about $1000:1$ estimating literal Dunn's index with an error of ± 0.01. All four MMRS methods for estimation of Dunn's index are linear in N, the number of samples in the data.

Chapter 6 contributed a novel cluster tendency assessment algorithm, inc-siVAT, for incremen-

tal and time efficient visualization of evolving cluster structures in high-velocity, data streams. The inc-siVAT algorithm deals with the big data streams in chunks (of configurable size). First, it generates a static *reordered dissimilarity image* (RDI) of an initial smart sample obtained using MMRS sampling. Then, it incrementally updates the MMRS sample on the fly and produces its (incrementally built) RDI image, using our novel inc-MMRS algorithm, and inc-VAT/inc-iVAT and dec-VAT/dec-iVAT algorithms, to track changes in cluster structure after each chunk. The applicability of inc-siVAT was demonstrated for visualizing evolving cluster structure and anomaly detection for dynamic streams of four big datasets, including a real IoT dataset.

Chapter 7 demonstrated big data clustering for a real-world application, particularly for intelligent transportation systems. In this chapter, we developed a scalable framework for vehicle trajectory prediction, based on Markov chain models and a big data clustering algorithm, Traj-clusiVAT, which is suitable for a large number of overlapping trajectories in a dense road network, typically for major cities around the world. Traj-clusiVAT is a modified version of the clusiVAT, implemented for *trajectory prediction* (TP) task, which developed a novel method to compute a representative trajectory for each cluster and a hybrid nearest prototyping scheme for robust assignment of a trajectory to one of the clusters. Experiments performed on two real-life, large-scale taxi trajectory datasets from the Beijing and Singapore Road networks demonstrated that the Traj-clusiVAT based TP approach outperforms two current trajectory prediction schemes, based on the prediction accuracy and distance error for short-term and long-term prediction. Also, the average prediction time (< 1.5ms) of Traj-clusiVAT based TP method for both the datasets suggests that it satisfies the requirement for real-time predictions.

8.2 Future Research Directions

This thesis has made significant contributions to knowledge advancement by proposing a suite of efficient and scalable cluster analysis algorithms for big data. The proposed algorithms were verified on several real, big datasets that were possibly unlabeled, high-dimensional, noisy and streaming, which makes our proposed algorithms suitable for various applications. Specifically, we also demonstrated the utility of big data clustering for trajectory prediction as a smart city application. Many further interesting work can be built over the scientific contribution of this

thesis. Some of them are listed below:

- It is clear from our experiments in Chapter 5 that our approximation methods for Dunn's cluster validity index are not so useful for some of the generalized Dunn's indices, because they do not depend only on extreme points in the data. Perhaps the most intriguing possibility emerging from this contribution is that other internal CVIs can be usefully estimated in the manner we computed Dunn's index for big data. There are many other internal CVIs such as Davies-Bouldin index, Xie-Beni, Silhouette, Alternative Silhouette, point bi-serial, McClain-Rao, Gamma, and Tau, that are $O(N^2)$ or worse. In the era of big data, where the number of samples can easily reach $N \geq 10^8$, computation of all of these measures becomes problematic. A future possibility is to develop approximation algorithms to compute these CVIs for big data.

- The inc-siVAT model presented in this thesis addresses the first two problems of cluster analysis, cluster tendency assessment and subsequent clustering (for anomaly detection), for high-velocity data streams. At present, there is no cluster validation model available in the literature for high-velocity, streaming data. This can be one of possible future work.

- The inc-siVAT model considers the data from the beginning until current time instant, which causes an increased number of data points in seen data by inc-siVAT model after each chunk. This may slow down the computation process over the time due to memory constraints. This problem can be handled by adapting a sliding window based concept with existing inc-siVAT algorithm (as discussed in Chapter 6). An alternative and better way to solve this problem would be to develop an incremental model which uses only the summarized information or a few selected data points from the past chunks with the new data points (from a new chunk) to obtain an updated smart (MMRS) sample and its iVAT image.

- Another future line of work would be to adapt incremental siVAT approach for online training of Traj-clusiVAT based trajectory prediction model to updated clusters and corresponding Markov models in real-time. One more possible extension is to include additional factors such as speed, time, and user information in our prediction system to improve its prediction performance.

Bibliography

[1] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 6, pp. 1130–1146, 2012.

[2] R. Chitta, *Kernel-based clustering of big data*. Michigan State University, 2015.

[3] D. Laney, "3d data management: Controlling data volume, velocity and variety," *META group research note*, vol. 6, no. 70, p. 1, 2001.

[4] H. Chen, R. H. Chiang, and V. C. Storey, "Business intelligence and analytics: from big data to big impact," *MIS quarterly*, pp. 1165–1188, 2012.

[5] R. Kitchin, "The real-time city? big data and smart urbanism," *GeoJournal*, vol. 79, no. 1, pp. 1–14, 2014.

[6] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma, "The role of big data in smart city," *International Journal of Information Management*, vol. 36, no. 5, pp. 748–758, 2016.

[7] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[8] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, vol. 2, no. 3, p. 4, 2006.

[9] S. K. Halgamuge and L. Wang, *Classification and clustering for knowledge discovery*. Springer Science & Business Media, 2005, vol. 4.

[10] M. Moshtaghi, S. Rajasegarar, C. Leckie, and S. Karunasekera, "Anomaly detection by clustering ellipsoids in wireless sensor networks," in *5th International Conference on Intelligent Sensors, Sensor Networks and Informations Processing (ISSNIP)*, 2009, pp. 331–336.

[11] J. C. Bezdek, T. C. Havens, J. M. Keller, C. Leckie, L. Park, M. Palaniswami, and S. Rajasegarar, "Clustering elliptical anomalies in sensor networks," in *IEEE international conference on Fuzzy systems (FUZZ)*, 2010, pp. 1–8.

[12] S. M. Erfani, M. Baktashmotlagh, S. Rajasegarar, S. Karunasekera, and C. Leckie, "R1SVM: a Randomised Nonlinear Approach to Large-Scale Anomaly Detection," in *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*, 2015, pp. 432–438.

[13] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM SIGMOD Record*, vol. 30, no. 2, pp. 151–162, 2001.

[14] Q. Du and J. E. Fowler, "Hyperspectral image compression using jpeg2000 and principal component analysis," *IEEE Geoscience and Remote Sensing Letters*, vol. 4, no. 2, pp. 201–205, 2007.

[15] O. Zamir and O. Etzioni, "Grouper: a dynamic clustering interface to web search results," *Computer Networks*, vol. 31, no. 11-16, pp. 1361–1374, 1999.

[16] P. J. Carrington, J. Scott, and S. Wasserman, *Models and methods in social network analysis*. Cambridge university press, 2005, vol. 28.

[17] W. Wu, H. Xiong, and S. Shekhar, *Clustering and information retrieval*. Springer Science & Business Media, 2013, vol. 11.

[18] M. Jakobsson and N. A. Rosenberg, "Clumpp: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure," *Bioinformatics*, vol. 23, no. 14, pp. 1801–1806, 2007.

[19] A. Ben-Dor and Z. Yakhini, "Clustering gene expression patterns," in *Proceedings of the third annual international conference on Computational molecular biology*. ACM, 1999, pp. 33–42.

[20] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. Pal, *Fuzzy models and algorithms for pattern recognition and image processing*. Springer Science & Business Media, 1999, vol. 4.

[21] G. Punj and D. W. Stewart, "Cluster analysis in marketing research: Review and suggestions for application," *Journal of marketing research*, pp. 134–148, 1983.

[22] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural computation*, vol. 12, no. 10, pp. 2385–2404, 2000.

[23] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[24] Y. Luo, Y. Jiang, S. Khan, S. Peng, Y. Feng, and B. Han, "Analysis of urban heat island effect using k-means clustering," in *Information Science and Engineering (ICISE), 2010 2nd International Conference on*. IEEE, 2010, pp. 3543–3546.

[25] P. Hoffmann and K. H. Schlünzen, "Weather pattern classification to represent the urban heat island in present and future climate," *Journal of Applied Meteorology and Climatology*, vol. 52, no. 12, pp. 2699–2714, 2013.

[26] Y. Zheng, S. Rajasegarar, C. Leckie, and M. Palaniswami, "Smart car parking: temporal clustering and anomaly detection in urban car parking," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*. IEEE, 2014, pp. 1–6.

[27] W. Shao, F. D. Salim, A. Song, and A. Bouguettaya, "Clustering big spatiotemporal-interval data," *IEEE Transactions on Big Data*, vol. 2, no. 3, pp. 190–203, 2016.

[28] N. Piovesan, L. Turi, E. Toigo, B. Martinez, and M. Rossi, "Data analytics for smart parking applications," *Sensors*, vol. 16, no. 10, p. 1575, 2016.

[29] D. Kumar, H. Wu, Y. Lu, S. Krishnaswamy, and M. Palaniswami, "Understanding urban mobility via taxi trip clustering," in *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, vol. 1. IEEE, 2016, pp. 318–324.

[30] J. Tang, F. Liu, Y. Wang, and H. Wang, "Uncovering urban human mobility from large scale taxi gps data," *Physica A: Statistical Mechanics and its Applications*, vol. 438, pp. 140–153, 2015.

[31] A. Lavin and D. Klabjan, "Clustering time-series energy data from smart meters," *Energy efficiency*, vol. 8, no. 4, pp. 681–689, 2015.

[32] M. Azaza and F. Wallin, "Smart meter data clustering using consumption indicators: responsibility factor and consumption variability," *Energy Procedia*, vol. 142, pp. 2236–2242, 2017.

[33] S. Haben, C. Singleton, and P. Grindrod, "Analysis and clustering of residential customers energy behavioral demand using smart meter data," *IEEE transactions on smart grid*, vol. 7, no. 1, pp. 136–144, 2016.

[34] S. T. M. Bourobou and Y. Yoo, "User activity recognition in smart homes using pattern clustering applied to temporal ann algorithm," *Sensors*, vol. 15, no. 5, pp. 11 953–11 971, 2015.

[35] C. Lee, Z. Luo, K. Y. Ngiam, M. Zhang, K. Zheng, G. Chen, B. C. Ooi, and W. L. J. Yip, "Big healthcare data analytics: Challenges and applications," in *Handbook of Large-Scale Distributed Computing in Smart Healthcare*. Springer, 2017, pp. 11–41.

[36] M. Liao, Y. Li, F. Kianifard, E. Obi, and S. Arcona, "Cluster analysis and its application to healthcare claims data: a study of end-stage renal disease patients who initiated hemodialysis," *BMC nephrology*, vol. 17, no. 1, p. 25, 2016.

[37] V. Ramesh, K. Ramar, and S. Babu, "Parallel k-means algorithm on agricultural databases," *IJCSI International Journal of Computer Science Issues*, vol. 10, no. 1, pp. 1694–0814, 2013.

[38] G. Ruß, R. Kruse, and M. Schneider, "A clustering approach for management zone delineation in precision agriculture," in *Proceedings of the Int. Conf. on Precision Agriculture*, 2010.

[39] B.-I. Kim, S. Kim, and S. Sahoo, "Waste collection vehicle routing problem with time windows," *Computers & Operations Research*, vol. 33, no. 12, pp. 3624–3642, 2006.

[40] A. Parchitelli, F. Nocera, G. Iacobellis, M. Mongiello, T. Di Noia, and E. Di Sciascio, "A pre-process clustering methods for the waste collection problem," in *Service Operations and Logistics, and Informatics (SOLI), 2017 IEEE International Conference on*. IEEE, 2017, pp. 242–247.

[41] J. C. Bezdek, *Primer on Cluster Analysis: Four Basic Methods that (Usually) Work*. First Edition Design Publishing, 2017, vol. 1.

[42] B. Hopkins and J. G. Skellam, "A new method for determining the type of distribution of plant individuals," *Annals of Botany*, vol. 18, no. 2, pp. 213–227, 1954.

[43] A. Banerjee and R. N. Dave, "Validating clusters using the hopkins statistic," in *Fuzzy systems, 2004. Proceedings. 2004 IEEE international conference on*, vol. 1. IEEE, 2004, pp. 149–153.

[44] J. C. Bezdek and R. J. Hathaway, "Vat: A tool for visual assessment of (cluster) tendency," in *Proc. IJCNN*, 2002, pp. 2225–2230.

[45] T. C. Havens and J. C. Bezdek, "An efficient formulation of the improved visual assessment of cluster tendency (ivat) algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 813–822, 2012.

[46] L. Wang, X. Geng, J. Bezdek, C. Leckie, and R. Kotagiri, "Enhanced visual analysis for cluster tendency assessment and data partitioning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1401–1414, 2010.

[47] R. J. Hathaway, J. C. Bezdek, and J. M. Huband, "Scalable visual assessment of cluster tendency for large data sets," *Pattern Recognition*, vol. 39, no. 7, pp. 1315–1324, 2006.

[48] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973.

[49] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[50] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 841–847, 1991.

[51] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[52] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[53] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[54] C. J. Van Rijsbergen, "A theoretical basis for the use of co-occurrence data in information retrieval," *Journal of documentation*, vol. 33, no. 2, pp. 106–119, 1977.

[55] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 245–250.

[56] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in Statistical Physics*. Springer, 2004, pp. 273–309.

[57] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, "Big data clustering: a review," in *International Conference on Computational Science and Its Applications*. Springer, 2014, pp. 707–720.

[58] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE transactions on emerging topics in computing*, vol. 2, no. 3, pp. 267–279, 2014.

[59] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 1, p. 1, 2009.

[60] I. Assent, "Clustering high dimensional data," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 340–350, 2012.

[61] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 169–178.

[62] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of intelligent information systems*, vol. 17, no. 2, pp. 107–145, 2001.

[63] D. Kumar, J. C. Bezdek, M. Palaniswami, S. Rajasegarar, C. Leckie, and T. C. Havens, "A hybrid approach to clustering in big data," *IEEE transactions on cybernetics*, vol. 46, no. 10, pp. 2372–2385, 2016.

[64] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings 2003 VLDB Conference*. Elsevier, 2003, pp. 81–92.

[65] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams: Theory and practice," *IEEE transactions on knowledge and data engineering*, vol. 15, no. 3, pp. 515–528, 2003.

[66] L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 3, p. 12, 2009.

[67] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 133–142.

[68] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 2006, pp. 328–339.

[69] D. Kumar, J. C. Bezdek, S. Rajasegarar, M. Palaniswami, C. Leckie, J. Chan, and J. Gubbi, "Adaptive cluster tendency visualization and anomaly detection for streaming data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 2, p. 24, 2016.

[70] A. Katal, M. Wazid, and R. Goudar, "Big data: issues, challenges, tools and good practices," in *Contemporary Computing (IC3), 2013 Sixth International Conference on*. IEEE, 2013, pp. 404–409.

[71] R. Avogadri and G. Valentini, "Fuzzy ensemble clustering based on random projections for dna microarray data analysis," *Artificial Intelligence in Medicine*, vol. 45, no. 2, pp. 173–183, 2009.

[72] M. Popescu, J. Keller, J. Bezdek, and A. Zare, "Random projections fuzzy c-means (rpfcm) for big data clustering," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2015, pp. 1–6.

[73] M. Ye, W. Liu, J. Wei, and X. Hu, "Fuzzy-means and cluster ensemble with random projection for big data clustering," *Mathematical Problems in Engineering*, 2016.

[74] P. Hore, L. O. Hall, and D. B. Goldgof, "Single pass fuzzy c means," in *2007 IEEE International Fuzzy Systems Conference*. IEEE, 2007, pp. 1–7.

[75] P. S. Bradley, U. M. Fayyad, C. Reina *et al.*, "Scaling clustering algorithms to large databases," in *KDD*, 1998, pp. 9–15.

[76] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 1177–1178.

[77] P. J. Rousseeuw and L. Kaufman, *Finding Groups in Data*. Wiley Online Library, 1990.

[78] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," *Information Systems*, vol. 26, no. 1, pp. 35–58, 2001.

[79] Y. Lai, R. Orlandic, W. G. Yee, and S. Kulkarni, "Scalable clustering for large high-dimensional data based on data summarization," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, 2007, pp. 456–461.

[80] T. Sakai and A. Imiya, "Fast spectral clustering with random projection and sampling," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2009, pp. 372–384.

[81] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," in *ACM SIGMoD Record*, vol. 28, no. 2. ACM, 1999, pp. 61–72.

[82] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 3, 2003, pp. 186–193.

[83] R. L. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[84] Z. Ghafoori, S. M. Erfani, S. Rajasegarar, J. C. Bezdek, S. Karunasekera, and C. Leckie, "Efficient unsupervised parameter estimation for one-class support vector machines," *IEEE Transactions on Neural Networks and Learning Systems (in review)*, 2017.

[85] Y. Li and L. Maguire, "Selecting critical patterns based on local geometrical and statistical information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 6, pp. 1189–1201, 2011.

[86] J. C. Bezdek, S. Rajasegarar, M. Moshtaghi, C. Leckie, M. Palaniswami, and T. C. Havens, "Anomaly detection in environmental monitoring networks [application notes]," *IEEE Computational Intelligence Magazine*, vol. 6, no. 2, pp. 52–58, 2011.

[87] A. Asahara, K. Maruyama, A. Sato, and K. Seto, "Pedestrian-movement prediction based on mixed markov-chain model," in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2011, pp. 25–33.

[88] A. Kharrat, I. S. Popa, K. Zeitouni, and S. Faiz, "Clustering algorithm for network constraint trajectories," in *Headway in Spatial Data Handling*. Springer, 2008, pp. 631–647.

[89] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*. ACM, 2010, pp. 99–108.

[90] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 316–324.

[91] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The computer journal*, vol. 41, no. 8, pp. 578–588, 1998.

[92] H. H. Bock, "Classification and clustering: Problems for the future," in *New Approaches in Classification and Data Analysis*. Springer, 1994, pp. 3–24.

[93] S. Still and W. Bialek, "How many clusters? an information-theoretic perspective," *Neural computation*, vol. 16, no. 12, pp. 2483–2506, 2004.

[94] G. Claeskens, N. L. Hjort *et al.*, "Model selection and model averaging," *Cambridge Books*, 2008.

[95] W. Pan, "Akaike's information criterion in generalized estimating equations," *Biometrics*, vol. 57, no. 1, pp. 120–125, 2001.

[96] W. S. Cleveland, *Visualizing data*. Hobart Press, 1993.

[97] J. Czekanowski, *Zur differentialdiagnose der neandertalgruppe*. Friedr. Vieweg & Sohn, 1909.

[98] R. L. Ling, "A computer generated aid for cluster analysis," *Communications of the ACM*, vol. 16, no. 6, pp. 355–361, 1973.

[99] I. S. Dhillon, D. S. Modha, and W. S. Spangler, "Visualizing class structure of multidimensional data," *Computing Science and Statistics*, pp. 488–493, 1998.

[100] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Labs Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[101] R. G. Casey and G. Nagy, "An autonomous reading machine," *IEEE Transactions on Computers*, vol. 100, no. 5, pp. 492–503, 1968.

[102] R. W. Kennard and L. A. Stone, "Computer aided design of experiments," *Technometrics*, vol. 11, no. 1, pp. 137–148, 1969.

[103] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[104] I. Steponavičė, M. Shirazi-Manesh, R. J. Hyndman, K. Smith-Miles, and L. Villanova, "On sampling methods for costly multi-objective black-box optimization," in *Advances in Stochastic and Deterministic Global Optimization*. Springer, 2016, pp. 273–296.

[105] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[106] A. K. Jain, "Data clustering: 50 years beyond k-means," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 3–4.

[107] R. Xu and D. Wunsch, *Clustering*. John Wiley & Sons, 2008, vol. 10.

[108] C. C. Aggarwal and C. K. Reddy, *Data clustering: algorithms and applications*. CRC Press, 2013.

[109] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[110] ——, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[111] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[112] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on*. IEEE, 1979, pp. 761–766.

[113] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids [w:] statistical data analysis based on the ll-norm and related methods, red. y. dodge," 1987.

[114] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[115] B. S. Everitt, *The Cambridge dictionary of statistics in the medical sciences*. Cambridge University Press Cambridge, 1995.

[116] H. Seifoddini and P. M. Wolfe, "Application of the similarity coefficient method in group technology," *IIE transactions*, vol. 18, no. 3, pp. 271–277, 1986.

[117] H. K. Seifoddini, "Single linkage versus average linkage clustering in machine cells formation applications," *Computers & Industrial Engineering*, vol. 16, no. 3, pp. 419–426, 1989.

[118] R. Sibson, "Slink: an optimally efficient algorithm for the single-link cluster method," *The computer journal*, vol. 16, no. 1, pp. 30–34, 1973.

[119] D. Defays, "An efficient algorithm for a complete link method," *The Computer Journal*, vol. 20, no. 4, pp. 364–366, 1977.

[120] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[121] J. C. Gower and G. J. Ross, "Minimum spanning trees and single linkage cluster analysis," *Applied statistics*, pp. 54–64, 1969.

[122] P. Legendre and L. Legendre, "Numerical ecology: second english edition," *Developments in environmental modelling*, vol. 20, 1998.

[123] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *The Computer Journal*, vol. 26, no. 4, pp. 354–359, 1983.

[124] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[125] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[126] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod record*, vol. 28, no. 2. ACM, 1999, pp. 49–60.

[127] A. Hinneburg, D. A. Keim *et al.*, "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, vol. 98, 1998, pp. 58–65.

[128] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (methodological)*, pp. 1–38, 1977.

[129] M. Meila and D. Heckerman, "An experimental comparison of several clustering and initialization methods," *arXiv preprint arXiv:1301.7401*, 2013.

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

[131] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *IEEE transactions on knowledge and data engineering*, vol. 14, no. 5, pp. 1003–1016, 2002.

[132] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[133] T. C. Havens, J. C. Bezdek, J. M. Keller, and M. Popescu, "Dunn's cluster validity index as a contrast measure of vat images," in *19th International Conference on Pattern Recognition (ICPR)*. IEEE, 2008, pp. 1–4.

[134] T. C. Havens, J. C. Bezdek, and M. Palaniswami, "Scalable single linkage hierarchical clustering for big data," in *IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE, 2013, pp. 396–401.

[135] D. Kumar, M. Palaniswami, S. Rajasegarar, C. Leckie, J. C. Bezdek, and T. C. Havens, "clusivat: A mixed visual/numerical clustering algorithm for big data," in *IEEE International Conference on Big Data*. IEEE, 2013, pp. 112–117.

[136] M. Charikar, L. O'Callaghan, and R. Panigrahy, "Better streaming algorithms for clustering problems," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM, 2003, pp. 30–39.

[137] D. Barbará and P. Chen, "Using the fractal dimension to cluster datasets," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 260–264.

[138] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding windows," *Knowledge and Information Systems*, vol. 15, no. 2, pp. 181–214, 2008.

[139] O. Nasraoui and C. Rojas, "Robust clustering for tracking noisy evolving data streams," in *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 2006, pp. 619–623.

[140] P. Hore, L. O. Hall, and D. B. Goldgof, "A fuzzy c means variant for clustering evolving data streams," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*. IEEE, 2007, pp. 360–365.

[141] J. A. Hartigan, "Clustering algorithms (probability & mathematical statistics)," 1975.

[142] C. C. Aggarwal, *Data streams: models and algorithms*. Springer Science & Business Media, 2007, vol. 31.

[143] ——, "A survey of stream clustering algorithms." 2013.

[144] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "Dbdc: Density based distributed clustering," in *International Conference on Extending Database Technology*. Springer, 2004, pp. 88–105.

[145] G. Karypis and V. Kumar, "Parallel multilevel series k-way partitioning scheme for irregular graphs," *Siam Review*, vol. 41, no. 2, pp. 278–300, 1999.

[146] ——, "Multilevelk-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[147] G. Andrade, G. Ramos, D. Madeira, R. Sachetto, R. Ferreira, and L. Rocha, "G-dbscan: A gpu accelerated algorithm for density-based clustering," *Procedia Computer Science*, vol. 18, pp. 369–378, 2013.

[148] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *IEEE International Conference on Cloud Computing*. Springer, 2009, pp. 674–679.

[149] A. Ene, S. Im, and B. Moseley, "Fast clustering using mapreduce," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 681–689.

[150] R. L. Ferreira Cordeiro, C. Traina Junior, A. J. Machado Traina, J. López, U. Kang, and C. Faloutsos, "Clustering very large multi-dimensional datasets with mapreduce," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 690–698.

[151] C. Jin, M. M. A. Patwary, A. Agrawal, W. Hendrix, W.-k. Liao, and A. Choudhary, "Disc: A distributed single-linkage hierarchical clustering algorithm using mapreduce," *work*, vol. 23, p. 27, 2013.

[152] Y. He, H. Tan, W. Luo, S. Feng, and J. Fan, "Mr-dbscan: a scalable mapreduce-based dbscan algorithm for heavily skewed data," *Frontiers of Computer Science*, vol. 8, no. 1, pp. 83–99, 2014.

[153] J. Béjar Alonso, "Strategies and algorithms for clustering large datasets: a review," 2013.

[154] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, vol. 24, no. 6, p. 417, 1933.

[155] S. Kaski, "Dimensionality reduction by random mapping: Fast similarity computation for clustering," in *Proceedings of International Joint Conference on Neural Networks*, vol. 1, 1998, pp. 413–418.

[156] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2001, pp. 274–281.

[157] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," in *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1998, pp. 159–168.

[158] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[159] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

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

[161] J. B. Kruskal and M. Wish, *Multidimensional scaling*. Sage, 1978, vol. 11.

[162] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary Mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[163] S. Dasgupta, "Experiments with random projection," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 2000, pp. 143–151.

[164] J. C. Bezdek, X. Ye, M. Popescu, J. Keller, and A. Zare, "Random projection below the JL limit," in *Proceedings of International Joint Conference on Neural Network (IJCNN)*, 2016, pp. 2414–2423.

[165] W. Wang and Y. Zhang, "On fuzzy cluster validity indices," *Fuzzy sets and systems*, vol. 158, no. 19, pp. 2095–2117, 2007.

[166] R. Dubes and A. K. Jain, "Validity studies in clustering methodologies," *Pattern recognition*, vol. 11, no. 4, pp. 235–254, 1979.

[167] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243–256, 2013.

[168] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2837–2854, 2010.

[169] J. C. Bezdek and N. R. Pal, "Some new indexes of cluster validity," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 301–315, 1998.

[170] Y. Lei, J. C. Bezdek, J. Chan, N. X. Vinh, S. Romano, and J. Bailey, "Generalized information theoretic cluster validity indices for soft clusterings," in *IEEE Symposium on*

Proceedings of the Eighth International Conference on Numerical Taxonomy, 2014, pp. 24–31.

[171] J. C. Bezdek, M. Moshtaghi, T. Runkler, and C. Leckie, "The generalized C index for internal fuzzy cluster validity," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 6, pp. 1500–1512, 2016.

[172] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 7, pp. 773–780, 1989.

[173] J. C. Bezdek, "Objective function clustering," in *Pattern recognition with fuzzy objective function algorithms*. Springer, 1981, pp. 43–93.

[174] M. Roubens, "Pattern classification problems and fuzzy sets," *Fuzzy Sets and Systems*, vol. 1, no. 4, pp. 239–253, 1978.

[175] J. C. Bezdek, "Mathematical models for systematics and taxonomy," in *Proceedings of the Eighth International Conference on Numerical Taxonomy*, 1975, pp. 143–66.

[176] K. Pearson, *On the theory of contingency and its relation to association and normal correlation; On the general theory of skew correlation and non-linear regression*. Cambridge University Press, 1904.

[177] D. T. Anderson, J. C. Bezdek, M. Popescu, and J. M. Keller, "Comparing fuzzy, probabilistic, and possibilistic partitions," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 5, pp. 906–918, 2010.

[178] E. Hullermeier, M. Rifqi, S. Henzgen, and R. Senge, "Comparing fuzzy partitions: A generalization of the rand index and related measures," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 3, pp. 546–556, 2012.

[179] M. Tlili and T. M. Hamdani, "Big data clustering validity," in *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of.* IEEE, 2014, pp. 348–352.

[180] J. M. Luna-Romera, M. del Mar Martínez-Ballesteros, J. García-Gutiérrez, and J. C. Riquelme-Santos, "An approach to silhouette and dunn clustering indices applied to big data

in spark," in *Conference of the Spanish Association for Artificial Intelligence*. Springer, 2016, pp. 160–169.

[181] M. C. Pham, Y. Cao, R. Klamma, and M. Jarke, "A clustering approach for collaborative filtering recommendation using social network analysis." *J. UCS*, vol. 17, no. 4, pp. 583–604, 2011.

[182] F. Z. Benchara, M. Youssfi, O. Bouattane, H. Ouajji, and M. O. Bensalah, "Distributed c-means algorithm for big data image segmentation on a massively parallel and distributed virtual machine based on cooperative mobile agents," *Journal of Software Engineering and Applications*, vol. 8, no. 03, p. 103, 2015.

[183] G. Kerr, H. J. Ruskin, M. Crane, and P. Doolan, "Techniques for clustering gene expression data," *Computers in biology and medicine*, vol. 38, no. 3, pp. 283–293, 2008.

[184] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 312–321.

[185] J.-I. Won, S.-W. Kim, J.-H. Baek, and J. Lee, "Trajectory clustering in road network environment," in *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*. IEEE, 2009, pp. 299–305.

[186] D. Guo, S. Liu, and H. Jin, "A graph-based approach to vehicle trajectory analysis," *Journal of Location Based Services*, vol. 4, no. 3-4, pp. 183–199, 2010.

[187] M. R. Evans, D. Oliver, S. Shekhar, and F. Harvey, "Fast and exact network trajectory similarity computation: a case-study on bicycle corridor planning," in *Proceedings of the 2nd ACM SIGKDD international workshop on urban computing*. ACM, 2013, p. 9.

[188] G.-P. Roh and S.-w. Hwang, "Nncluster: An efficient clustering algorithm for road network trajectories," in *International Conference on Database Systems for Advanced Applications*. Springer, 2010, pp. 47–61.

[189] S. Song, D. Kwak, Y. Kwak, K. Bok, and D. Ko, "Segmentation based trajectory clustering in road network with location sensing technology," *Sensor Letters*, vol. 11, no. 9, pp. 1779–1782, 2013.

[190] B. Han, L. Liu, and E. Omiecinski, "Road-network aware trajectory clustering: Integrating locality, flow, and density," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 416–429, 2015.

[191] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2009, pp. 352–361.

[192] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2009, pp. 336–343.

[193] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[194] M. Morzy, "Mining frequent trajectories of moving objects for location prediction," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2007, pp. 667–680.

[195] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou, "A hybrid prediction model for moving objects," in *IEEE 24th International Conference on Data Engineering (ICDE)*. Ieee, 2008, pp. 70–79.

[196] Y. Ishikawa, Y. Tsukamoto, and H. Kitagawa, "Extracting mobility statistics from indexed spatio-temporal datasets." in *STDBM*, 2004, pp. 9–16.

[197] R. Simmons, B. Browning, Y. Zhang, and V. Sadekar, "Learning to predict driver route and destination intent," in *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. IEEE, 2006, pp. 127–132.

[198] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility markov chains," in *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*. ACM, 2012, p. 3.

[199] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous computing*, vol. 7, no. 5, pp. 275–286, 2003.

[200] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden markov models," in *Proceedings of the 2012 ACM conference on ubiquitous computing*. ACM, 2012, pp. 911–918.

[201] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 2017.

[202] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 593–604.

[203] Y. Wang, Q. Han, and H. Pan, "A clustering scheme for trajectories in road networks," in *Advanced Technology in Teaching-Proceedings of the 2009 3rd International Conference on Teaching and Computational Science (WTCS 2009)*. Springer, 2012, pp. 11–18.

[204] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer, "Probabilistic trajectory prediction with gaussian mixture models," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 141–146.

[205] P.-R. Lei, T.-J. Shen, W.-C. Peng, and J. Su, "Exploring spatial-temporal trajectory model for location prediction," in *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, vol. 1. IEEE, 2011, pp. 58–67.

[206] S. Qiao, N. Han, J. Wang, R.-H. Li, L. A. Gutierrez, and X. Wu, "Predicting long-term trajectories of connected vehicles via the prefix-projection technique," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[207] E. P. Xing, M. I. Jordan, R. M. Karp *et al.*, "Feature selection for high-dimensional genomic microarray data," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 1, 2001, pp. 601–608.

[208] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[209] X. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *Proceedings of the twenty-first ACM Twenty-first International Conference on Machine Learning*, 2004, p. 36.

[210] A. Topchy, A. K. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1866–1881, 2005.

[211] A. L. Fred and A. K. Jain, "Data clustering using evidence accumulation," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 4, 2002, pp. 276–280.

[212] S. Dudoit and J. Fridlyand, "Bagging to improve the accuracy of a clustering procedure," *Bioinformatics*, vol. 19, no. 9, pp. 1090–1099, 2003.

[213] E. Dimitriadou, A. Weingessel, and K. Hornik, "A combination scheme for fuzzy clustering," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 16, no. 07, pp. 901–912, 2002.

[214] H. G. Ayad and M. S. Kamel, "Cumulative voting consensus method for partitions with variable number of clusters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 160–173, 2008.

[215] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[216] J. Wall, "Generalized inverses of stochastic matrices," *Linear Algebra and its Applications*, vol. 10, no. 2, pp. 147–154, 1975.

[217] P. Courrieu, "Fast computation of moore-penrose inverse matrices," *CoRR*, vol. abs/0804.4809, 2008. [Online]. Available: http://arxiv.org/abs/0804.4809

[218] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*, 2009.

[219] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, vol. 43, no. 10, pp. 3605–3620, 2010.

[220] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and Electronics in Agriculture*, vol. 24, no. 3, pp. 131–151, 1999.

[221] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist dataset of handwritten digits," *URL http://yann. lecun. com/exdb/mnist*, 1998.

[222] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *ESANN*, 2013.

[223] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*. Citeseer, 2009.

[224] N. Zahid, M. Limouri, and A. Essaid, "A new cluster-validity for fuzzy clustering," *Pattern Recognition*, vol. 32, no. 7, pp. 1089–1097, 1999.

[225] M. G. Kendall, *Rank correlation methods.* Griffin, 1948.

[226] T. Urruty, C. Djeraba, and D. A. Simovici, "Clustering by random projections," in *Industrial Conference on Data Mining*. Springer, 2007, pp. 107–119.

[227] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data," *Data Mining and Knowledge Discovery*, vol. 11, no. 1, pp. 5–33, 2005.

[228] B. L. Milenova and M. M. Campos, "O-cluster: Scalable clustering of large high dimensional data sets," in *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2002, pp. 290–297.

[229] S. Gilpin, B. Qian, and I. Davidson, "Efficient hierarchical clustering of large high dimensional datasets," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 1371–1380.

[230] E. J. Otoo, A. Shoshani, and S.-w. Hwang, "Clustering high dimensional massive scientific datasets," *Journal of Intelligent Information Systems*, vol. 17, no. 2-3, pp. 147–168, 2001.

[231] L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek, "Automatically determining the number of clusters in unlabeled data sets," *IEEE Transactions on knowledge and Data Engineering*, vol. 21, no. 3, pp. 335–350, 2009.

[232] I. J. Sledge, T. C. Havens, J. M. Huband, J. C. Bezdek, and J. M. Keller, "Finding the number of clusters in ordered dissimilarities," *Soft Computing*, vol. 13, no. 12, pp. 1125–1142, 2009.

[233] T. C. Havens, J. C. Bezdek, J. M. Keller, and M. Popescu, "Clustering in ordered dissimilarity data," *International Journal of Intelligent Systems*, vol. 24, no. 5, pp. 504–528, 2009.

[234] T. C. Havens, J. C. Bezdek, J. M. Keller, M. Popescu, and J. M. Huband, "Is vat really single linkage in disguise?" *Annals of Mathematics and Artificial Intelligence*, vol. 55, no. 3, pp. 237–251, 2009.

[235] Y. S. Ahmed, *Multiple random projection for fast, approximate nearest neighbor search in high dimensions*. University of Toronto, 2004.

[236] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[237] O. Pele and M. Werman, "The quadratic-chi histogram distance family," in *European conference on computer vision*. Springer, 2010, pp. 749–762.

[238] K. Chen and L. Liu, "Detecting the change of clustering structure in categorical data streams," in *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 2006, pp. 504–508.

[239] ——, "ivibrate: Interactive visualization-based framework for clustering large datasets," *ACM Transactions on Information Systems (TOIS)*, vol. 24, no. 2, pp. 245–294, 2006.

[240] S. Mahallati, J. C. Bezdek, D. Kumar, M. R. Popovic, and T. A. Valiante, *Interpreting Cluster Structure in Waveform Data with Visual Assessment and Dunn's Index*. Springer, 2018, pp. 73–101.

[241] R. O. Duda and P. E. Hart, *Pattern Elesslfication and Scene Analysis*. Wiley, 1973.

[242] S. Theodoridis, A. Pikrakis, K. Koutroumbas, and D. Cavouras, *Introduction to pattern recognition: a matlab approach*. Academic Press, 2010.

[243] G. W. Milligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, no. 2, pp. 159–179, 1985.

[244] I. Gurrutxaga, J. Muguerza, O. Arbelaitz, J. M. Pérez, and J. I. Martín, "Towards a standard methodology to evaluate internal cluster validity indices," *Pattern Recognition Letters*, vol. 32, no. 3, pp. 505–515, 2011.

[245] E. Dimitriadou, S. Dolničar, and A. Weingessel, "An examination of indexes for determining the number of clusters in binary data sets," *Psychometrika*, vol. 67, no. 1, pp. 137–159, 2002.

[246] J. C. Bezdek, W. Li, Y. Attikiouzel, and M. Windham, "A geometric approach to cluster validity for normal mixtures," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 1, no. 4, pp. 166–179, 1997.

[247] L. Vendramin, R. J. Campello, and E. R. Hruschka, "Relative clustering validity criteria: A comparative overview," *Statistical analysis and data mining: the ASA data science journal*, vol. 3, no. 4, pp. 209–235, 2010.

[248] "Plexon Neurotechnolgy Research Systems," http://www.plexon.com/.

[249] M. Lavielle, "Detection of multiple changes in a sequence of dependent variables," *Stochastic Processes and their Applications*, vol. 83, no. 1, pp. 79–102, 1999.

[250] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[251] N. Elmqvist, P. Dragicevic, and J.-D. Fekete, "Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation," *IEEE transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1539–1148, 2008.

[252] D. Kumar, J. C. Bezdek, S. Rajasegarar, C. Leckie, and M. Palaniswami, "A visual-numeric approach to clustering and anomaly detection for trajectory data," *The Visual Computer*, vol. 33, no. 3, pp. 265–281, 2017.

[253] H. Song, Z. Jiang, A. Men, and B. Yang, "A hybrid semi-supervised anomaly detection model for high-dimensional data," *Computational intelligence and neuroscience*, vol. 2017, 2017.

[254] S. Gninenko, "Miniboone anomaly and heavy neutrino decay," *Physical review letters*, vol. 103, no. 24, p. 241802, 2009.

[255] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 637–646.

[256] M. Chen, Y. Liu, and X. Yu, "Predicting next locations with object clustering and trajectory clustering," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2015, pp. 344–356.

[257] J. J.-C. Ying, W.-C. Lee, T.-C. Weng, and V. S. Tseng, "Semantic trajectory mining for location prediction," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2011, pp. 34–43.

[258] Q. Lv, Y. Qiao, N. Ansari, J. Liu, and J. Yang, "Big data driven hidden markov model based individual mobility prediction at points of interest," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5204–5216, 2017.

[259] L. Chen, M. Lv, and G. Chen, "A system for destination and future route prediction based on trajectory mining," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 657–676, 2010.

[260] D. Kumar, S. Rajasegarar, M. Palaniswami, X. Wang, and C. Leckie, "A scalable framework for clustering vehicle trajectories in a dense road network," in *The 4th International Work-*

shop on Urban Computing (UrbComp), Held in conjunction with the 21th ACM SIGKDD, 2015.

[261] G. Yavaş, D. Katsaros, Ö. Ulusoy, and Y. Manolopoulos, "A data mining approach for location prediction in mobile environments," *Data & Knowledge Engineering*, vol. 54, no. 2, pp. 121–146, 2005.

[262] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1081–1094, 2008.

[263] E. H.-C. Lu, V. S. Tseng, and S. Y. Philip, "Mining cluster-based temporal mobile sequential patterns in location-based service environments," *IEEE transactions on knowledge and data engineering*, vol. 23, no. 6, pp. 914–927, 2011.

[264] C. Sung, D. Feldman, and D. Rus, "Trajectory clustering for motion prediction," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1547–1552.

[265] N. Ferreira, J. T. Klosowski, C. E. Scheidegger, and C. T. Silva, "Vector field k-means: Clustering trajectories by fitting multiple vector fields," in *Computer Graphics Forum*, vol. 32, no. 3pt2. Wiley Online Library, 2013, pp. 201–210.

[266] M. Barbehenn, "A note on the complexity of dijkstra's algorithm for graphs with weighted vertices," *IEEE transactions on computers*, vol. 47, no. 2, p. 263, 1998.

[267] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[268] Y. Lu, S. Xiang, and W. Wu, "Taxi queue, passenger queue or no queue?" in *Proc. of 18th International Conference on Extending Database Technology (EDBT). Brussels, Belgium*, 2015, pp. 593–604.

[269] "GraphHopper, "Map-matching"," http://www.unhabitat.org/pmss/listItem Details.aspx?publicationID=3387, 2017.

[270] P. C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Destination prediction by trajectory distribution-based model," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[271] Q. Huang, "Mining online footprints to predict user's next location," *International Journal of Geographical Information Science*, vol. 31, no. 3, pp. 523–541, 2017.

[272] S. Qiao, N. Han, W. Zhu, and L. A. Gutierrez, "Traplan: an effective three-in-one trajectory-prediction model in transportation networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1188–1198, 2015.

[273] H. Jeung, H. T. Shen, and X. Zhou, "Mining trajectory patterns using hidden markov models," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2007, pp. 470–480.